Ajuste dos SQL parâmetros do Postgre na Amazon RDS e no Amazon Aurora

AWS Orientação prescritiva

AWS Orientação prescritiva: Ajuste dos SQL parâmetros do Postgre na Amazon RDS e no Amazon Aurora

Table of Contents

Ajuste dos parâmetros do PostgreSQL no Amazon RDS e no Amazon Aurora

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

Fevereiro de 2024 (histórico do documento)

A edição compatível com o Amazon Aurora PostgreSQL e o Amazon Relational Database Service (Amazon RDS) para PostgreSQL são serviços sofisticados de banco de dados relacional de código aberto que oferecem uma gama completa de recursos. Você pode usar esses serviços para configurar seu banco de dados PostgreSQL em uma variedade de plataformas e aplicativos.

O Aurora e o Amazon RDS oferecem uma forma simplificada de gerenciar e operar seus bancos de dados PostgreSQL. Eles foram projetados para gerenciar a infraestrutura do banco de dados e fornecer alta disponibilidade, durabilidade e escalabilidade enquanto você se concentra no desenvolvimento de aplicativos. No entanto, as configurações padrão desses serviços podem não ser ideais para todas as cargas de trabalho. Por padrão, esses serviços são configurados para serem executados em qualquer lugar com o mínimo de recursos possível e sem introduzir vulnerabilidades. O ajuste dos parâmetros pode ajudá-lo a obter melhor desempenho, reduzir o tempo de inatividade e melhorar a eficiência geral do banco de dados. Ao otimizar os parâmetros para sua carga de trabalho específica, você pode aproveitar ao máximo os recursos que o Amazon RDS e o Aurora oferecem e maximizar seus benefícios.

Por exemplo, você pode melhorar o desempenho otimizando o Aurora e o Amazon RDS para o Amazon RDS for PostgreSQL e configurando seus parâmetros. Você também deve levar em consideração o desempenho ao criar consultas ao banco de dados. Mesmo quando você otimiza as configurações do banco de dados, o sistema pode ter um desempenho ruim se suas consultas realizarem varreduras completas da tabela, quando utilizarem um índice ou se executarem operações caras de junção ou agregação.

Este guia é para desenvolvedores de banco de dados, engenheiros de banco de dados e administradores que desejam ajustar parâmetros de memória, autovacuum, registro e replicação lógica para seus bancos de dados PostgreSQL. O guia também aborda parâmetros específicos do Amazon RDS for PostgreSQL e do Aurora PostgreSQL compatíveis. O ajuste desses parâmetros pode ajudá-lo a otimizar o desempenho do banco de dados e reduzir o uso de recursos para sua carga de trabalho específica, resultando em melhor desempenho e economia de custos.

Usando grupos de parâmetros de banco de dados e cluster de banco de dados

O Amazon RDS e o Aurora podem determinar automaticamente os valores de parâmetros mais adequados para determinadas configurações com base no tamanho da instância do seu banco de dados. Eles também oferecem suporte à personalização de parâmetros para ajuste de desempenho por meio de grupos de parâmetros para instâncias e clusters de banco de dados.

Você pode usar grupos de parâmetros do banco de dados e do cluster de banco de dados para modificar os parâmetros que controlam vários aspectos do comportamento do mecanismo de banco de dados, como uso de memória, E/S de disco, rede e bloqueio. Ao ajustar esses parâmetros, você pode otimizar o mecanismo de banco de dados para sua carga de trabalho específica e melhorar o desempenho.

Você pode criar e configurar grupos de parâmetros de bancos de dados e clusters de banco de dados usando a AWS Management Console API, the AWS Command Line Interface (AWS CLI) ou Amazon RDS. Este guia pressupõe que você esteja usando o. AWS CLI Para obter instruções de console e API, consulte Como trabalhar com grupos de parâmetros de banco de dados e Trabalhar com grupos de parâmetros de cluster de banco de dados na documentação do Amazon RDS.

> **Important**
>
> Para usar os AWS CLI comandos fornecidos neste guia, você deve primeiro instalar e configurar AWS CLI o.

Para criar e configurar um grupo de parâmetros de banco de dados:

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

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

```
  --parameters "ParameterName=max_connections<parameter-
name>,ParameterValue=<value>,ApplyMethod=immediate"

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

Para criar e configurar um grupo de parâmetros de cluster de banco de dados:

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

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

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

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

> **Note**
>
> O Aurora e o Amazon RDS fornecem um grupo de parâmetros padrão com valores pré-configurados que não podem ser alterados.

Os grupos de parâmetros podem ser definidos como estáticos ou dinâmicos. Os parâmetros dinâmicos são aplicados imediatamente, independentemente de a `ApplyMethod=immediate` opção estar ativada. Os parâmetros estáticos exigem uma reinicialização manual para entrarem em vigor.

Ajustando os parâmetros de memória

Ajustar os parâmetros de memória é uma tarefa essencial para otimizar o desempenho dos bancos de dados compatíveis com Amazon RDS e Aurora PostgreSQL. A alocação adequada de memória para várias operações de banco de dados, como execução de consultas, classificação, indexação e armazenamento em cache, pode melhorar significativamente o desempenho do banco de dados. Esta seção aborda alguns dos parâmetros de memória mais importantes compatíveis com Amazon RDS e Aurora PostgreSQL, incluindo seus valores padrão, fórmulas para calcular valores apropriados e como alterá-los. Para obter uma lista completa de parâmetros, consulte Como trabalhar com parâmetros em sua instância de banco de dados RDS para PostgreSQL na documentação do Amazon RDS e Parâmetros do Amazon Aurora PostgreSQL na documentação do Aurora.

A otimização desses parâmetros requer uma compreensão profunda da carga de trabalho do seu banco de dados, bem como dos recursos disponíveis na sua instância de banco de dados Aurora ou Amazon RDS. O desempenho do sistema é influenciado por duas grandes categorias de parâmetros: parâmetros vitais e parâmetros contingentes.

Os parâmetros vitais são parâmetros indispensáveis que exercem uma influência significativa e direta no desempenho do sistema e são essenciais para alcançar os melhores resultados.

- bufferes_compartilhados
- tamp_buffers
- tamanho_de_cache efetivo
- work_mem
- maintenance_work_mem

Os parâmetros contingentes são parâmetros específicos do cenário e do negócio. Eles dependem de outros fatores e desempenham um papel indireto, mas fundamental, no suporte a parâmetros vitais e na maximização do desempenho geral do sistema.

- custo_de_página aleatória
- seq_page_cost
- rastreie o tamanho da consulta de atividade
- Tempo limite da sessão de inatividade na transação

- statement_timeout
- caminho_de_pesquisa
- conexões máximas

Esses parâmetros são discutidos com mais detalhes nas seções a seguir.

shared_buffers

O `shared_buffers` parâmetro controla a quantidade de memória que o PostgreSQL usa para armazenar dados em cache na memória. Definir esse parâmetro com um valor apropriado pode ajudar a melhorar o desempenho da consulta.

Para o Amazon RDS, o valor padrão para `shared_buffers` é definido como `{DBInstanceClassMemory/32768}` bytes, com base na memória disponível para a instância de banco de dados. Para o Aurora, o valor padrão é definido como`{DBInstanceClassMemory/12038,-50003}`, com base na memória disponível para a instância de banco de dados. O valor ideal para esse parâmetro depende de vários fatores, incluindo o tamanho do banco de dados, o número de conexões simultâneas e a memória de instância disponível.

AWS CLI sintaxe

O comando a seguir é alterado `shared_buffers` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Estático (a aplicação das alterações requer uma reinicialização)

Valor padrão: `{DBInstanceClassMemory/32768}` bytes no Amazon RDS for PostgreSQL, no Aurora compatível com `{DBInstanceClassMemory/12038,-50003}` PostgreSQL. Na maioria dos casos, essa equação representa cerca de 25% da memória do seu sistema. Seguindo essa diretriz, a `shared_buffers` configuração no grupo de parâmetros é definida usando as unidades padrão do PostgreSQL de buffers de 8K em vez de bytes ou kilobytes.

A configuração dos `shared_buffers` parâmetros pode ter um impacto significativo no desempenho, por isso recomendamos que você teste suas alterações minuciosamente para garantir que o valor seja adequado à sua carga de trabalho.

Exemplo

Digamos que você tenha um aplicativo de serviços financeiros executando um banco de dados PostgreSQL no Amazon RDS ou no Aurora. Esse banco de dados é usado para armazenar dados de transações do cliente. Ele tem um grande número de tabelas e é acessado por vários aplicativos em um grande número de servidores. O aplicativo está apresentando baixo desempenho de consulta e alto uso da CPU. Você determina que o ajuste do `shared_buffers` parâmetro pode ajudar a melhorar o desempenho.

No Amazon RDS for PostgreSQL, o `shared_buffers` valor padrão de é `{DBInstanceClassMemory/32768}` definido como bytes de `db.r5.xlarge` memória disponível em (por exemplo, 3 GB). Para determinar o valor apropriado para`shared_buffers`, você executa uma série de testes com valores variados de`shared_buffers`, começando com o valor padrão da memória disponível e aumentando o valor gradualmente. Para cada teste, você mede o desempenho da consulta e o uso da CPU do banco de dados.

Com base nos resultados do teste, você determina que definir o valor de `shared_buffers` 8 GB resulta no melhor desempenho geral da consulta e no melhor uso da CPU para sua carga de trabalho. O valor é determinado por meio de uma combinação de testes e análises das características da carga de trabalho, incluindo o tamanho do banco de dados, o número e a complexidade das consultas, o número de usuários simultâneos e os recursos disponíveis do sistema. Depois de fazer a alteração, seus sistemas de monitoramento verificam o desempenho do banco de dados para garantir que o novo valor seja adequado à sua carga de trabalho. Em seguida, você pode ajustar parâmetros adicionais conforme necessário para melhorar ainda mais o desempenho.

temp_buffers

`temp_buffers`é um parâmetro de configuração fundamental no Aurora PostgreSQL compatível e no Amazon RDS for PostgreSQL que pode afetar significativamente o desempenho de cargas de trabalho que envolvem classificações, hashes e operações agregadas em tabelas temporárias. Esse parâmetro determina a quantidade de memória alocada para buffers temporários, o que, por sua vez, afeta a eficiência e a velocidade dessas operações. Se não houver memória suficiente alocada`temp_buffers`, o sistema talvez precise usar métodos mais lentos e menos eficientes para classificações, hashes e operações de agregação em tabelas temporárias, resultando em um desempenho abaixo do ideal.

AWS CLI sintaxe

O comando a seguir é alterado `temp_buffers` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: 8 MB

Para obter mais informações sobre esse parâmetro, consulte Consumo de recursos na documentação do PostgreSQL.

Exemplo

Se sua carga de trabalho envolver muitas operações de classificação, hashing e agregação em tabelas temporárias, `temp_buffers` talvez não aloque memória suficiente. Nesse caso, o sistema

pode ter que realizar operações de classificação em tabelas temporárias que levam a métodos mais lentos baseados em disco, em vez de operações de classificação, hashing e agregação na memória. Isso pode causar uma redução significativa no desempenho das consultas, especialmente para consultas que envolvem grandes conjuntos de dados. Aumentar o valor de `temp_buffers` pode garantir que haja memória suficiente disponível para realizar essas operações na memória, levando a uma melhoria significativa no desempenho.

Para encontrar o valor ideal de`temp_buffers`, monitore o desempenho do sistema e identifique áreas de desempenho abaixo do ideal. Se você observar tempos de resposta de consulta lentos ou alta utilização da CPU, considere fazer ajustes`temp_buffers`. Por exemplo, se sua carga de trabalho envolve muitas tabelas temporárias, aumentar o valor de `temp_buffers` pode ajudar a garantir que essas tabelas sejam armazenadas na memória. Isso pode ser muito mais rápido do que usar E/S de leitura/gravação do armazenamento.

Experimente diferentes valores de `temp_buffers` em pequenos incrementos e monitore cuidadosamente o desempenho do sistema após cada alteração. Analise o impacto de diferentes valores no desempenho e ajuste a configuração com base nas características específicas da sua carga de trabalho.

effective_cache_size

O `effective_cache_size` parâmetro especifica a quantidade de memória que o PostgreSQL deve considerar disponível para armazenar dados em cache. Definir esse parâmetro corretamente pode melhorar o desempenho, permitindo que o PostgreSQL faça melhor uso da memória disponível.

AWS CLI sintaxe

O comando a seguir é alterado `effective_cache_size` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

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

Exemplo

Uma plataforma de aprendizado on-line tem um grande banco de dados de materiais didáticos, dados de alunos e outros conteúdos que são acessados com frequência pelos usuários. O aplicativo é executado em uma instância do Amazon RDS `db.r5.xlarge` for PostgreSQL com 32 GB de memória. O aplicativo apresenta um desempenho lento quando os usuários tentam ler o conteúdo acessado com frequência. Depois de analisar o uso de recursos do servidor de banco de dados, você determina que o PostgreSQL não está fazendo o melhor uso da memória disponível.

O `effective_cache_size` parâmetro no Amazon RDS for PostgreSQL controla a quantidade de memória usada pelo servidor para armazenamento em cache de disco. O valor padrão é definido como `SUM({DBInstanceClassMemory/12038},-50003)` KB para a classe de instância`db.r5.xlarge`, mas esse padrão pode não ser apropriado para todas as cargas de trabalho. Neste exemplo, o servidor de banco de dados pode estar armazenando grandes quantidades de materiais do curso e dados de alunos acessados com frequência. Aumentar o valor do `effective_cache_size` parâmetro pode resultar em mais dados sendo armazenados em cache na memória, o que reduz o número de leituras de disco necessárias e melhora o desempenho da consulta.

Quando você executa uma consulta, o Amazon RDS for PostgreSQL primeiro verifica se os dados exigidos pela consulta já estão no cache. Nesse caso, os dados podem ser lidos da memória em vez de serem lidos do disco. Se os dados não estiverem no cache, eles precisarão ser lidos do disco, o que pode ser uma operação lenta.

Para a plataforma de aprendizado on-line, você pode decidir configurar `effective_cache_size` para 16 GB (metade da memória disponível) após o teste e a análise. Esse valor permite que o PostgreSQL faça melhor uso da memória disponível, o que reduz o número de leituras de disco necessárias e melhora o desempenho da consulta.

work_mem

O `work_mem` parâmetro controla a quantidade de memória usada pelas consultas para operações de classificação e hashing. Seu valor padrão é 4 MB. Se uma consulta incluir várias operações, ela poderá usar até 4 MB para cada operação. Aumentar o valor de `work_mem` pode melhorar o desempenho de consultas que exigem classificação ou hashing, porque essas operações exigem mais memória. No entanto, definir esse parâmetro muito alto pode causar uso excessivo de memória, levando à degradação do desempenho.

Para calcular o valor ideal para`work_mem`, você pode usar a seguinte fórmula:

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

onde `available_memory` é a quantidade total de memória disponível no servidor, max_connections é o número máximo de conexões permitidas e work_mem_fraction é uma fração que determina quanto da memória disponível deve ser alocada para cada conexão.

AWS CLI sintaxe

O comando a seguir é alterado `work_mem` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: 4 MB

Exemplo

Uma ferramenta de análise de mídia social processa uma grande quantidade de dados, e consultas que envolvem operações complexas de classificação e junção causam alta E/S de disco e vazam para o disco. Se você aumentar o valor `work_mem` de 4 MB para 16 MB, o PostgreSQL poderá usar mais memória para essas operações. Isso reduz a quantidade de E/S e melhora o desempenho da consulta.

maintenance_work_mem

O `maintenance_work_mem` parâmetro controla a quantidade de memória usada pelas operações de manutenção`VACUUM`, como`ANALYZE`, e criação de índice. O valor padrão desse parâmetro no Amazon RDS e no Aurora é 64 MB.

Para calcular o valor apropriado para esse parâmetro, você pode usar esta fórmula:

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

O Aurora PostgreSQL Compatible Edition e o Amazon RDS for PostgreSQL aplicam a seguinte fórmula para definir o valor ideal:

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

AWS CLI sintaxe

O comando a seguir é alterado `maintenance_work_mem` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: 64 MB

Exemplo

Seu aplicativo de grande escala usa um banco de dados PostgreSQL hospedado no Aurora ou no Amazon RDS. Você percebe que o banco de dados está lento e não responde durante as atividades de manutenção, como limpeza e indexação. Você pode monitorar métricas como uso de memória, tempos de operação de manutenção e uso da CPU para determinar se o valor atual de `maintenance_work_mem` está causando problemas.

Para determinar o valor ideal para`maintenance_work_mem`, você pode ajustar o parâmetro e monitorar seu impacto. Se o uso de memória for consistentemente alto ou os tempos de operação forem maiores do que o esperado durante as operações de manutenção, o aumento `maintenance_work_mem` pode ajudar. Por outro lado, se o uso da CPU for consistentemente alto durante as operações de manutenção, diminuir `maintenance_work_mem` pode ajudar. Ao iterar por meio de ajustes e testes, você pode encontrar o valor ideal `maintenance_work_mem` que fornece o melhor equilíbrio entre uso de memória, tempos de operação de manutenção e uso da CPU.

Durante sua investigação, digamos que você determine que o valor padrão de 64 MB `maintenance_work_mem` é muito baixo para o tamanho do seu banco de dados. Como resultado, as operações de manutenção demoram mais para serem concluídas, causam tempo de inatividade excessivo e diminuem o desempenho do seu aplicativo. Para resolver esse problema, você pode decidir ajustar o `maintenance_work_mem` parâmetro aumentando-o de 64 MB para 512 MB (que você identifica como o valor ideal). A aplicação da alteração pode melhorar os tempos de operação de manutenção em dois terços. Por exemplo, uma operação de vácuo que antes levava 30 minutos para ser concluída agora pode levar apenas 10 minutos. Como resultado dessa otimização, seu banco de dados agora pode lidar com as atividades de manutenção com mais eficiência.

random_page_cost

O `random_page_cost` parâmetro ajuda a determinar o custo do acesso aleatório à página. O planejador de consultas no Amazon RDS e no Aurora usa esse parâmetro, junto com outras estatísticas sobre a tabela, para determinar o plano mais eficiente para executar uma consulta.

Os `random_page_cost` parâmetros `seq_page_cost` e estão intimamente relacionados e geralmente são usados juntos pelo planejador para comparar os custos de diferentes métodos de

acesso e decidir qual deles é o mais eficiente. Portanto, se você alterar um desses parâmetros, também deverá considerar se o outro parâmetro precisa ser ajustado.

Em geral, o planejador de consultas tenta minimizar o custo da execução de uma consulta. O custo é determinado usando uma combinação do número de leituras de páginas do disco e do valor de`random_page_cost`. Um valor mais alto de `random_page_cost` tende a favorecer as varreduras sequenciais, enquanto um valor mais baixo tende a favorecer as varreduras de índice. Um valor mais baixo também tende a favorecer junções de loop aninhadas em vez de junções de hash.

O `random_page_cost` parâmetro usa o valor padrão do mecanismo PostgreSQL (4), a menos que um valor seja definido no grupo de parâmetros ou na sessão local. Você pode ajustar esse valor dependendo das características específicas do servidor e da carga de trabalho. Se a maioria dos índices usados em sua carga de trabalho couber na memória ou no cache hierárquico do Aurora, alterar o valor de `random_page_cost` para um valor próximo é apropriado. `seq_page_cost`

AWS CLI sintaxe

O comando a seguir é alterado `random_page_cost` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: 4

Exemplo

Digamos que você tenha um banco de dados que armazena uma grande quantidade de dados em uma tabela que é frequentemente consultada com filtros em colunas não indexadas. As consultas demoram muito para serem concluídas e o planejador de consultas não seleciona o plano mais eficiente para acessar os dados.

Uma forma de melhorar o desempenho seria diminuir o `random_page_cost` parâmetro. Se você defini-lo como 1, o custo do acesso aleatório à página seria quatro vezes menor do que o valor padrão. Se você `random_page_cost` deixasse o valor padrão de 4, o acesso aleatório à página seria quatro vezes mais caro do que o acesso sequencial à página (conforme determinado pelo `seq_page_cost` parâmetro, que é 1,0 por padrão). No entanto, nesse caso específico, o acesso aleatório à página pode, na verdade, ser muito mais caro, dependendo do tipo de armazenamento.

Diminuir o valor do `random_page_cost` parâmetro pode aumentar a probabilidade de o planejador de consultas selecionar um plano baseado em índice ou usar um método de acesso diferente que seja mais adequado às características específicas da tabela.

Recomendamos que você monitore o desempenho da consulta depois de alterar o parâmetro e fazer os ajustes necessários. Você também deve verificar o planejador de consultas com uma `EXPLAIN` declaração para verificar se ele está selecionando um plano eficiente.

Esse é apenas um exemplo. A configuração ideal depende das características específicas da sua carga de trabalho. Além disso, esse é apenas um aspecto do ajuste de desempenho; você também deve considerar outros parâmetros e opções de configuração que podem afetar o desempenho da consulta.

seq_page_cost

O `seq_page_cost` parâmetro ajuda a determinar o custo do acesso sequencial à página. O planejador de consultas no Amazon RDS e no Aurora usa esse parâmetro, junto com outras estatísticas sobre a tabela, para determinar o plano mais eficiente para executar uma consulta.

Os `random_page_cost` parâmetros `seq_page_cost` e estão intimamente relacionados e geralmente são usados juntos pelo planejador para comparar os custos de diferentes métodos de acesso e decidir qual deles é o mais eficiente. Portanto, se você alterar um desses parâmetros, também deverá considerar se o outro parâmetro precisa ser ajustado.

Quando uma tabela é acessada de forma sequencial, o PostgreSQL pode usar o cache do sistema de arquivos do sistema operacional para fornecer acesso mais rápido. Por padrão, `seq_page_cost` é definido como 1.0, o que pressupõe que o acesso sequencial à página seja tão barato quanto a

leitura de um único bloco de disco. Se você tem uma tabela que é acessada principalmente de forma sequencial, mas o acesso ao disco é lento porque é limitado por menos IOPS, convém aumentar o valor de `seq_page_cost` para refletir o custo adicional de acessar o disco.

A alteração do valor desse parâmetro afeta todas as consultas executadas no sistema, portanto, recomendamos que você teste suas consultas com valores diferentes para determinar o valor ideal para seu caso de uso específico.

AWS CLI sintaxe

O comando a seguir é alterado `seq_page_cost` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: 1.0

Exemplo

Digamos que você tenha um banco de dados que armazena uma grande quantidade de dados em uma tabela que é acessada principalmente sequencialmente. A tabela é usada principalmente para gerar relatórios, as consultas são executadas muito lentamente e o planejador de consultas não consegue selecionar o plano mais eficiente para acessar os dados.

Uma forma de melhorar o desempenho seria diminuir o `seq_page_cost` parâmetro. O valor padrão é 1,0, o que pressupõe que o acesso sequencial à página seja tão barato quanto a leitura de um único bloco de disco. No entanto, nesse caso específico, o acesso sequencial à página pode, na

verdade, ser mais caro do que isso devido ao tipo de armazenamento (dependendo da E/S). Se você `seq_page_cost` definir como 0,5, o custo do acesso sequencial à página será metade do valor padrão. Essa alteração pode aumentar a probabilidade de o planejador de consultas selecionar um plano que use um método de acesso sequencial mais adequado às características específicas da tabela.

Recomendamos que você monitore o desempenho da consulta depois de alterar o parâmetro e fazer os ajustes necessários. Você também deve verificar o plano de consulta com uma `EXPLAIN` declaração para verificar se ele está selecionando um plano eficiente.

Esse é apenas um exemplo. A configuração ideal depende das características específicas da sua carga de trabalho. Além disso, esse é apenas um aspecto do ajuste de desempenho; você também deve considerar outros parâmetros e opções de configuração que afetam o desempenho da consulta.

track_activity_query_size

O `track_activity_query_size` parâmetro controla o tamanho da sequência de caracteres de consulta que é registrada para cada sessão ativa na `pg_stat_activity` exibição. Por padrão, somente os primeiros 1.024 bytes da string de consulta são registrados no Amazon RDS for PostgreSQL, e 4.096 bytes são registrados no Aurora compatível com PostgreSQL. Se quiser registrar consultas mais longas, você pode definir esse parâmetro para um valor maior.

AWS CLI sintaxe

O comando a seguir é alterado `track_activity_query_size` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Estático (a aplicação das alterações requer uma reinicialização)

Valor padrão: 1.024 bytes (Amazon RDS for PostgreSQL), 4.096 bytes (compatível com Aurora PostgreSQL)

Exemplo

Seu banco de dados Amazon RDS for PostgreSQL está apresentando um desempenho de consulta lento e você suspeita que o problema possa estar relacionado a consultas de longa duração. Você pode investigar mais detalhadamente registrando consultas mais longas na `pg_stat_activity` exibição.

Aumentar o valor do `track_activity_query_size` parâmetro pode resultar no aumento do registro, o que pode ter um impacto no desempenho do banco de dados. Recomendamos que você redefina o parâmetro para seu valor padrão de 1.024 depois que o problema for resolvido.

idle_in_transaction_session_timeout

O `idle_in_transaction_session_timeout` parâmetro controla a quantidade de tempo que uma transação ociosa espera antes de ser interrompida.

O valor padrão desse parâmetro no Amazon RDS for PostgreSQL e compatível com Aurora PostgreSQL é 86.400.000 milissegundos.

AWS CLI sintaxe

O comando a seguir é alterado `idle_in_transaction_session_timeout` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: 86.400.000 milissegundos (compatível com Aurora PostgreSQL)

Exemplo

Você tem um aplicativo de comércio eletrônico que processa pedidos on-line. O aplicativo usa um banco de dados PostgreSQL hospedado no Amazon RDS ou no Aurora. Sempre que um cliente faz um pedido, o aplicativo inicia uma nova transação para atualizar os registros de estoque e pedidos.

Se uma transação ficar inativa por muito tempo, ela pode impedir que outras transações acessem os mesmos registros, causando problemas de desempenho e, potencialmente, até mesmo tempo de inatividade do aplicativo. Além disso, transações ociosas que não são interrompidas adequadamente podem consumir recursos valiosos do sistema, como memória e CPU.

Para evitar esses problemas, você pode definir o `idle_in_transaction_session_timeout` parâmetro com um valor que faça sentido para seu aplicativo. Por exemplo, você pode configurá-lo para 5 minutos (300 segundos) para que qualquer transação que fique inativa por mais de 5 minutos seja automaticamente interrompida. Isso pode ajudar a garantir que os recursos do sistema sejam usados com eficiência e que o aplicativo possa lidar com um grande número de pedidos sem diminuir a velocidade. Ao definir o valor apropriado para`idle_in_transaction_session_timeout`, você pode ajudar a garantir que seu aplicativo tenha um desempenho ideal no Amazon RDS ou no Aurora.

statement_timeout

O `statement_timeout` parâmetro define o tempo máximo que uma consulta pode ser executada antes de ser interrompida.

AWS CLI sintaxe

O comando a seguir é alterado `statement_timeout` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: 0 milissegundos (sem tempo limite)

Exemplo

Seu aplicativo web permite que os usuários pesquisem em um grande banco de dados de produtos. Às vezes, as consultas de pesquisa podem levar muito tempo para serem concluídas, causando tempos de resposta lentos para os usuários. Para resolver esse problema, você poderia definir o `statement_timeout` parâmetro para um valor baixo, como 10 segundos, o que forçaria a interrupção de qualquer consulta que levasse mais de 10 segundos.

Isso pode parecer uma medida drástica, mas na verdade pode ser muito eficaz para melhorar o desempenho. Em muitos casos, consultas de longa duração são causadas por consultas SQL mal otimizadas ou índices ineficientes. Ao definir um `statement_timeout` valor baixo, você pode identificar essas consultas problemáticas e tomar medidas para otimizá-las.

Por exemplo, suponha que você descubra que uma consulta de pesquisa específica está sempre atingindo o tempo limite. Você pode usar ferramentas como `EXPLAIN` e `EXPLAIN ANALYZE` para analisar a consulta e identificar quaisquer gargalos de desempenho. Depois de identificar o problema, você pode tomar medidas para otimizar a consulta adicionando novos índices, reescrevendo a consulta ou usando um algoritmo de pesquisa diferente. Ao analisar e otimizar continuamente suas consultas SQL dessa forma, você pode melhorar significativamente o desempenho do seu aplicativo.

search_path

O `search_path` parâmetro determina a ordem na qual os esquemas são pesquisados por objetos nas instruções SQL. O valor padrão é`$user, public`, o que significa que o PostgreSQL pesquisa objetos primeiro no esquema que corresponde ao nome do usuário e depois no esquema público.

Se você tiver um grande número de esquemas ou precisar acessar objetos em um esquema específico, alterar o `search_path` parâmetro pode ajudar a melhorar o desempenho. Quando

você define `search_path` um esquema específico, o PostgreSQL pode encontrar os objetos mais rapidamente sem precisar pesquisar vários esquemas.

Para alterar o `search_path` parâmetro no Amazon RDS e no Aurora, você pode usar o seguinte comando `ROLE` para o nível:

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

AWS CLI sintaxe

O comando a seguir é alterado `search_path` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: `$user, public`

Exemplo

Você tem um aplicativo multilocatário com esquemas separados para cada locatário que usa um banco de dados compatível com Amazon RDS for PostgreSQL ou Aurora PostgreSQL, e você precisa executar uma consulta que envolva a união de dados de vários esquemas.

Por padrão, o Amazon RDS e o Aurora usam o caminho de pesquisa para determinar qual esquema usar para uma determinada tabela. O caminho de pesquisa é uma lista de nomes de esquema que o PostgreSQL pesquisa em ordem quando você se refere a uma tabela sem qualificar o nome do

esquema. Por padrão, o Amazon RDS e o Aurora primeiro procuram a tabela no esquema que tem o mesmo nome do usuário atual e, em seguida, examinam o esquema público.

Digamos que você queira executar uma consulta que envolva a junção de tabelas de vários esquemas `tenant1tenant2`, chamados e. `tenant3` Para usar os esquemas de inquilinos, você pode incluir os nomes dos esquemas na consulta:

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

No entanto, um método mais eficiente é alterar o `search_path` parâmetro para incluir os esquemas do inquilino usando os comandos na seção de AWS CLI sintaxe. Você também pode usar o `SET` comando em uma sessão do PostgreSQL:

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

Em seguida, você pode escrever a consulta sem qualificar os nomes dos esquemas:

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

Isso pode tornar sua consulta mais concisa e fácil de ler, além de simplificar o código do aplicativo se você tiver muitas consultas que envolvam a junção de tabelas de vários esquemas.

max_connections

O `max_connections` parâmetro define o número máximo de conexões simultâneas para seu banco de dados PostgreSQL.

AWS CLI sintaxe

O comando a seguir é alterado `max_connections` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

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

Tipo: Estático (a aplicação das alterações requer uma reinicialização)

Valor padrão: `LEAST(DBInstanceClassMemory/9531392,5000)` conexões

Para otimizar o uso do `max_connections` Amazon RDS ou do Aurora e minimizar seu impacto no desempenho, considere as seguintes melhores práticas:

- Defina o valor do parâmetro com base nos recursos disponíveis do sistema.
- Monitore o uso da conexão para evitar atingir o limite rapidamente.
- Use o pool de conexões para reduzir o número de conexões necessárias.
- Use o Amazon RDS Proxy para agrupamento de conexões.

Ao ajustar o `max_connections` Amazon RDS for PostgreSQL ou compatível com Amazon Aurora PostgreSQL, considere os tipos de instância disponíveis e seus recursos alocados e concentre-se na memória e na capacidade da CPU. Os detalhes de armazenamento e E/S são gerenciados pelo AWS, portanto, você pode monitorar as características gerais da carga de trabalho e as métricas do sistema, como por meio da `FreeableMemory` Amazon CloudWatch ou do console do Amazon RDS, para confirmar se há memória suficiente para as conexões. Monitore `CPUUtilization` valores altos, o que pode indicar que ajustes são necessários. Evite uma configuração `max_connections` muito alta, pois ela pode afetar o uso da memória e potencialmente influenciar a E/S indiretamente. Lembre-se de que cada conexão consome memória. Para encontrar o equilíbrio certo, aumente lentamente `max_connections` e veja como isso afeta seu sistema. Fique atento aos sinais de desempenho mais lento ou maior uso da CPU. Verifique se seu aplicativo ainda funciona bem. Use recursos como réplicas de leitura no Aurora para distribuir o tráfego de leitura e reduzir a carga na instância primária. Revise e ajuste `max_connections` regularmente com base nos padrões de uso observados para garantir o desempenho ideal do banco de dados dentro das restrições de recursos determinadas.

Ajustando os parâmetros de autovacuum

Os bancos de dados Amazon RDS for PostgreSQL e os compatíveis com o Aurora PostgreSQL exigem manutenção periódica conhecida como aspiração. O Autovacuum é um utilitário PostgreSQL integrado que remove dados desatualizados ou desnecessários para liberar espaço no banco de dados. O processo de autovacuum executa o `VACUUM` comando em segundo plano em intervalos regulares.

Ajustar as configurações de autovacuum é uma etapa crucial para manter o desempenho, a estabilidade e a disponibilidade do seu sistema de banco de dados compatível com Amazon RDS for PostgreSQL ou Aurora PostgreSQL. Ao ajustar os parâmetros de autovacuum para se adequarem à carga de trabalho e ao tamanho do banco de dados, você pode otimizar o desempenho do processo de autovacuum e reduzir seu impacto nos recursos do sistema, melhorando assim a integridade geral do banco de dados.

Além de ajustar as configurações de autovacuum, é importante monitorar o desempenho do seu banco de dados e seus componentes usando as ferramentas e métricas disponíveis no Amazon RDS e no Aurora. Ao monitorar métricas de desempenho, como inchaço, espaço livre e tempos de execução de consultas, você pode identificar possíveis problemas antes que eles se tornem problemas graves e tomar as medidas apropriadas para resolvê-los.

Esta seção aborda os seguintes tópicos e parâmetros de autovacuum:

- Comandos VACUUM e ANALYZE
- Verificando se há inchaço
- aspirador automático
- autovacuum_work_mem
- autovacuum_naptime
- autovacuum_max_workers
- fator_de_escala de vácuo automático
- limiar de vácuo automático
- fator_de_escala de análise automática
- limiar de análise de vácuo automático
- limite de custo do vácuo automático

Para obter informações adicionais sobre autovacuum, consulte os links a seguir:

- Entendendo o autovacuum nos ambientes Amazon RDS for PostgreSQL (postagem no blog)
- Trabalhando com o autovacuum do PostgreSQL no Amazon RDS para PostgreSQL (documentação do Amazon RDS)
- Aspiração paralela no Amazon RDS for PostgreSQL e no Amazon Aurora PostgreSQL (postagem do blog)

Comandos VACUUM e ANALYZE

`VACUUM`coleta lixo e, opcionalmente, analisa um banco de dados. Para a maioria das aplicações, basta deixar o daemon autovacuum realizar a limpeza. No entanto, alguns administradores podem querer modificar os parâmetros do banco de dados para autovacuum ou complementar ou substituir as atividades do daemon usando `VACUUM` comandos gerenciados manualmente que podem ser executados de acordo com um agendador.

`VACUUM`recupera o armazenamento que é ocupado por tuplas mortas. Nas operações padrão do PostgreSQL, quando as tuplas são excluídas ou tornadas obsoletas por uma atualização, elas não são removidas fisicamente das tabelas até `VACUUM` que uma operação seja executada. Portanto, recomendamos que você execute `VACUUM` periodicamente, especialmente em tabelas que são atualizadas com frequência.

O ajuste de `VACUUM` parâmetros é particularmente importante no Amazon RDS for PostgreSQL e compatível com o Aurora PostgreSQL, porque esses serviços de banco de dados gerenciados têm características diferentes em comparação aos bancos de dados PostgreSQL autogerenciados. Essas diferenças podem afetar o desempenho das operações de vácuo. O ajuste `VACUUM` dos parâmetros é essencial para otimizar o uso de recursos e garantir que as operações de vácuo não afetem negativamente o desempenho e a disponibilidade do seu sistema de banco de dados.

Aqui estão alguns dos parâmetros que você pode usar com o `VACUUM` comando no Aurora PostgreSQL compatível e no Amazon RDS for PostgreSQL:

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

- `DISABLE_PAGE_SKIPPING`
- `table_name`
- `column_name`

`VACUUM ANALYZE`executa uma `VACUUM` operação seguida por uma `ANALYZE` operação para cada tabela selecionada. Ele fornece uma maneira eficiente de realizar a manutenção de rotina.

Usar o `VACUUM` comando sem a `FULL` opção recupera espaço para reutilização. Ele não requer um bloqueio exclusivo na tabela, portanto, você pode executar esse comando durante as operações padrão de leitura e gravação. No entanto, na maioria dos casos, o comando não retorna espaço extra para o sistema operacional, mas o mantém disponível para reutilização na mesma tabela. `VACUUM FULL`reescreve todo o conteúdo da tabela em um novo arquivo de disco sem espaço extra e permite que o espaço não utilizado seja devolvido ao sistema operacional. Esse formulário é muito mais lento e requer um `ACCESS EXCLUSIVE` bloqueio em cada mesa.

Para obter informações completas sobre esses parâmetros, consulte a documentação do PostgreSQL.

No Aurora e no Amazon RDS, o autovacuum é um processo daemon (utilitário em segundo plano) que executa os `ANALYZE` comandos `VACUUM` e regularmente para limpar dados redundantes no banco de dados e no servidor. Mesmo que você confie na aspiração automática, recomendamos que você revise e ajuste as configurações de autoaspiração discutidas nas seções a seguir para garantir o desempenho ideal.

Verificando se há inchaço

A consulta SQL a seguir examina cada tabela no esquema XML e identifica linhas inativas (tuplas) que desperdiçam espaço em disco:

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

Se essa consulta retornar uma alta porcentagem (pfrag) de tuplas mortas, você poderá usar o `VACUUM` comando para recuperar espaço.

Para monitorar os tamanhos dos dados antes e depois das transações, execute a seguinte consulta no shell depois de se conectar a um banco de dados específico:

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

autovacuum

Você pode definir o autovacuum globalmente usando o parâmetro de `autovacuum` configuração ou alterá-lo por tabela definindo a `autovacuum_enabled` coluna da `pg_class` tabela `false` para `true` ou para uma tabela específica.

Quando você ativa o autovacuum em uma tabela, o servidor do banco de dados examina periodicamente a tabela em busca de linhas e tuplas inativas e as remove em segundo plano, sem qualquer intervenção do administrador do banco de dados. Isso ajuda a manter a tabela pequena, melhorar o desempenho das consultas e reduzir o tamanho dos backups.

AWS CLI sintaxe

O comando a seguir habilita um grupo `autovacuum` de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: Ativado

Você também pode desativar ou ativar o autovacuum em uma tabela específica usando o psql:

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

A limpeza excessiva pode afetar o desempenho, por isso é importante monitorar o desempenho do processo de autovacuum, bem como o desempenho do seu banco de dados, e ajustar as configurações conforme necessário.

Exemplo

Seu banco de dados PostgreSQL tem uma tabela que recebe um grande volume de operações de gravação e exclusão. Sem o autovacuum, essa tabela acabaria ficando cheia de linhas mortas (ou seja, linhas que foram marcadas para exclusão, mas ainda não foram removidas fisicamente da tabela). Essas linhas inativas ocupariam espaço no disco, diminuiriam a velocidade das consultas e aumentariam o tamanho dos backups. Você pode ativar o autovacuum na tabela para verificar automaticamente as linhas inativas e removê-las para mitigar esses problemas.

autovacuum_work_mem

`autovacuum_work_mem`é um parâmetro de configuração do PostgreSQL que controla a quantidade de memória usada pelo processo de autovacuum quando ele executa tarefas de manutenção de tabelas, como limpeza ou análise.

No Aurora e no Amazon RDS, você pode ajustar o valor de `autovacuum_work_mem` para otimizar o desempenho.

AWS CLI sintaxe

O comando a seguir habilita um grupo `autovacuum_work_mem` de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: `GREATEST({DBInstanceClassMemory/32768},131072)` KB no Aurora compatível com PostgreSQL, 64 MB no Amazon RDS for PostgreSQL. No entanto, o valor padrão pode variar dependendo da versão específica do Amazon RDS ou do Aurora que você está usando.

Exemplo

Seu banco de dados Amazon RDS for PostgreSQL tem uma tabela grande que é atualizada com frequência. Com o tempo, você percebe que o banco de dados fica mais lento e suspeita que o autovacuum esteja demorando muito para ser concluído.

Como parte de sua investigação, você verifica os registros do sistema, usa a `pg_stat_activity` visualização para ver quais consultas e processos estão sendo executados no momento, verifica a `pg_stat_user_tables` visualização para ver as estatísticas de cada tabela, usa a `pg_settings` visualização para comparar o valor da `autovacuum_work_mem` memória disponível no sistema e monitora o uso da memória em busca de picos. Depois de coletar essas informações, você pode `autovacuum_work_mem` definir o valor ideal que sua carga de trabalho precisa. Para encontrar o equilíbrio certo entre o uso e o desempenho da memória, você pode decidir configurá-lo para um quarto da memória disponível no sistema. Depois de alterar o valor, você monitora o desempenho do banco de dados e pode ver que o autovacuum é concluído muito mais rápido do que antes e que seu banco de dados tem um desempenho geral mais rápido.

autovacuum_naptime

O `autovacuum_naptime` parâmetro controla o intervalo de tempo entre execuções sucessivas do processo de autovacuum. O valor padrão é 15 segundos para Amazon RDS for PostgreSQL e 5 segundos para Aurora compatível com PostgreSQL.

Por exemplo, digamos que seu banco de dados Amazon RDS for PostgreSQL tenha uma tabela que receba um grande volume de operações de gravação e exclusão. Se você mantiver a configuração padrão, as varreduras frequentes de autovacuum prejudicarão essa tabela altamente transativa. Se você definir esse parâmetro como um valor alto, haverá um intervalo maior entre as varreduras sucessivas e as linhas inativas serão removidas com menos frequência.

Você pode usar `autovacuum_naptime` para gerenciar a carga causada pelo processo de limpeza, especialmente se você tiver um servidor ocupado que já tenha uma carga alta de CPU ou E/

S. Quanto mais tempo você definir o tempo de soneca, menos frequentemente o autovacuum é executado, o que reduz a carga no servidor. No entanto, `autovacuum_naptime` definir um valor muito alto pode fazer com que suas tabelas do PostgreSQL cresçam e as linhas mortas se acumulem, levando a uma diminuição no desempenho. Recomendamos que você monitore o desempenho do processo de autovacuum e ajuste a `autovacuum_naptime` configuração conforme necessário.

AWS CLI sintaxe

O comando a seguir é alterado `autovacuum_naptime` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: 15 segundos (Amazon RDS for PostgreSQL), 5 segundos (compatível com Aurora PostgreSQL)

autovacuum_max_workers

O `autovacuum_max_workers` parâmetro controla o número máximo de processos de trabalho que o processo de autovacuum pode criar. Cada processo de trabalho é responsável por limpar ou analisar uma única mesa.

Por exemplo, digamos que você tenha um banco de dados grande com muitas tabelas que são atualizadas e excluídas com frequência. Se você `autovacuum_max_workers` definir um valor

baixo, como 1, somente uma mesa poderá ser aspirada por vez e levará mais tempo para que todas as mesas sejam limpas. Se você `autovacuum_max_workers` definir um valor alto, como 8, até oito tabelas poderão ser limpas simultaneamente. Isso pode tornar o processo de limpeza mais rápido para bancos de dados que contêm muitas tabelas.

AWS CLI sintaxe

O comando a seguir é alterado `autovacuum_max_workers` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Estático (a aplicação das alterações requer uma reinicialização)

Valor padrão: `GREATEST(DBInstanceClassMemory/64371566592,3)` workers

Aumentar a `autovacuum_max_workers` configuração pode aumentar a carga no servidor, o que pode afetar o desempenho se você não tiver recursos suficientes. A configuração ideal depende dos requisitos específicos do seu banco de dados, do tamanho e do número de tabelas que ele contém. Recomendamos que você experimente valores diferentes e monitore o desempenho para encontrar a configuração ideal para seu caso de uso.

autovacuum_vacuum_scale_factor

O parâmetro `autovacuum_vacuum_scale_factor` de configuração controla o quão agressivo o processo de autovacuum deve ser ao aspirar uma mesa.

O fator de escala de vácuo é uma fração do número total de tuplas em uma tabela que deve ser modificada antes que o autovacuum limpe a mesa. O valor padrão é 0,1 (ou seja, 10% das tuplas

devem ser modificadas). Por exemplo, se uma tabela tiver 1.000.000 de tuplas e 100.000 dessas tuplas estiverem marcadas como inativas ou excluídas, o autovacuum limpa a tabela, dependendo do valor de como fator de controle. `autovacuum_vacuum_threshold`

O `autovacuum_vacuum_scale_factor` parâmetro ajuda a controlar a frequência com que o processo de vácuo é executado. Se uma tabela receber muitas operações de gravação, talvez você queira diminuir o fator de escala de vácuo para que o autovacuum seja executado com mais frequência e mantenha a tabela menor. Por outro lado, se uma tabela receber poucas operações de gravação, talvez você queira aumentar o fator de escala de vácuo para que o autovacuum seja executado com menos frequência e economize recursos.

AWS CLI sintaxe

O comando a seguir é alterado `autovacuum_vacuum_scale_factor` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: 0.1

O `autovacuum_vacuum_scale_factor` parâmetro funciona em conjunto com os `autovacuum_vacuum_threshold` `autovacuum_vacuum_cost_limit, and` `autovacuum_naptime` parâmetros. Para obter informações adicionais sobre esse parâmetro, consulte a postagem do AWS blog Entendendo o autovacuum nos ambientes Amazon RDS for PostgreSQL.

autovacuum_vacuum_threshold

O `autovacuum_vacuum_threshold` parâmetro controla o número mínimo de operações de atualização ou exclusão de tuplas que devem ocorrer em uma tabela antes que o autovacuum a limpe. Essa configuração pode ser útil para evitar a limpeza desnecessária em tabelas que não têm uma alta taxa dessas operações. O valor padrão é 50, que é o padrão do mecanismo PostgreSQL, tanto para Amazon RDS for PostgreSQL quanto para Aurora PostgreSQL compatível.

Por exemplo, digamos que você tenha uma tabela com 100.000 linhas e `autovacuum_vacuum_threshold` esteja definida como 50. Se a tabela receber apenas 49 atualizações ou exclusões, o autovacuum não a limpará. Se a tabela receber 50 ou mais atualizações ou exclusões, o autovacuum a limpará, dependendo do valor `autovacuum_vacuum_scale_factor` multiplicado pelo número de linhas da tabela como fator de controle.

Definir esse parâmetro muito alto pode fazer com que a tabela cresça e as linhas mortas se acumulem, o que pode afetar o desempenho.

AWS CLI sintaxe

O comando a seguir é alterado `autovacuum_vacuum_threshold` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: 50 operações

O `autovacuum_vacuum_threshold` parâmetro funciona em conjunto com os `autovacuum_vacuum_scale_factor` `autovacuum_vacuum_cost_limit, and` `autovacuum_naptime` parâmetros. As configurações ideais dependem dos requisitos específicos do banco de dados e do tamanho da tabela.

Para obter informações adicionais sobre esse parâmetro, consulte a postagem do AWS blog Entendendo o autovacuum nos ambientes Amazon RDS for PostgreSQL.

autovacuum_analyze_scale_factor

O `autovacuum_analyze_scale_factor` parâmetro controla o quão agressivo o processo de autovacuum deve ser ao analisar (coletar) estatísticas sobre a distribuição de dados em uma tabela.

O processo de autovacuum usa esse parâmetro para calcular um limite com base no número de tuplas em uma tabela. Se o número de inserções, atualizações ou exclusões de tuplas exceder esse limite, o autovacuum analisará a tabela. O valor padrão é 0,05 (ou seja, 5% das tuplas devem ser modificadas) tanto para Amazon RDS for PostgreSQL quanto para Aurora PostgreSQL compatível.

Por exemplo, digamos que sua tabela tenha 1.000.000 de tuplas e você mantenha o `autovacuum_analyze_scale_factor` valor padrão em 0,05. Se a tabela receber 50.000 ou mais atualizações ou exclusões, o autovacuum a limpa, dependendo do `autovacuum_analyze_threshold` valor e adicionando o número de linhas da tabela como fator de controle.

AWS CLI sintaxe

O comando a seguir é alterado `autovacuum_analyze_scale_factor` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

```
  --parameters
"ParameterName=autovacuum_analyze_scale_factor,ParameterValue=<new_value>,ApplyMethod=immediat
```

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: 0,05 (5 por cento)

É essencial que o planejador de consultas colete estatísticas para tomar decisões informadas, como acessar os dados e organizá-los. Por isso, recomendamos que você monitore o desempenho do processo de autovacuum e ajuste as configurações conforme necessário para garantir que as estatísticas estejam atualizadas.

O `autovacuum_analyze_scale_factor` parâmetro funciona em conjunto com os `autovacuum_analyze_threshold` `autovacuum_analyze_cost_limit, and` `autovacuum_naptime` parâmetros. A configuração ideal depende dos requisitos específicos do banco de dados e do tamanho da tabela e da frequência das atualizações. Para obter informações adicionais sobre esse parâmetro, consulte a postagem do AWS blog Entendendo o autovacuum nos ambientes Amazon RDS for PostgreSQL.

autovacuum_analyze_threshold

O `autovacuum_analyze_threshold` parâmetro é semelhante `autovacuum_vacuum_threshold` a. Ele controla o número mínimo de inserções, atualizações ou exclusões de tuplas que devem ocorrer em uma tabela antes que o autovacuum a analise. Essa configuração pode ser útil para evitar a limpeza desnecessária em tabelas que não têm uma alta taxa dessas operações. O valor padrão é 50, que é o padrão do mecanismo PostgreSQL, tanto para Amazon RDS for PostgreSQL quanto para Aurora PostgreSQL compatível.

Por exemplo, digamos que você tenha uma tabela com 100.000 linhas e mantenha o `autovacuum_analyze_threshold` padrão em 50. Se a tabela receber apenas 49 inserções, atualizações ou exclusões, o autovacuum não a analisará. Se a tabela receber 50 ou mais inserções, atualizações ou exclusões, o autovacuum a analisará, mantendo o valor de `autovacuum_analyze_scale_factor` multiplicado pelo número de linhas da tabela como fator de controle.

AWS CLI sintaxe

O comando a seguir é alterado `autovacuum_analyze_threshold` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: 50 operações

Esse parâmetro funciona em conjunto com o `autovacuum_analyze_scale_factor` parâmetro, portanto, leve em consideração as duas configurações ao configurar o autovacuum.

É essencial que o planejador de consultas colete estatísticas para tomar decisões informadas, como acessar os dados e organizá-los. Uma configuração `autovacuum_analyze_threshold` muito alta pode fazer com que as estatísticas se tornem obsoletas, levando a um desempenho ruim. Recomendamos que você monitore o desempenho do processo de autovacuum e ajuste as configurações conforme necessário.

Para obter informações adicionais sobre esse parâmetro, consulte a postagem do AWS blog Entendendo o autovacuum nos ambientes Amazon RDS for PostgreSQL.

autovacuum_vacuum_cost_limit

O `autovacuum_vacuum_cost_limit` parâmetro controla a quantidade de recursos de CPU e E/S que um trabalhador de autovacuum pode consumir.

Limitar o uso de recursos dos processos de autovacuum pode ajudar a evitar que eles consumam muita CPU ou E/S de disco, o que pode afetar o desempenho de outras consultas executadas no

mesmo sistema. O parâmetro especifica um limite de custo, que é uma unidade de trabalho que o trabalhador pode realizar antes de fazer uma pausa e verificar se ainda está abaixo do limite. Por exemplo, se o parâmetro for definido como 2.000, um trabalhador poderá processar 2.000 unidades de trabalho antes da pausa.

Você pode definir o `autovacuum_vacuum_cost_limit` parâmetro usando o `SET` comando em uma sessão do PostgreSQL ou usar um comando. AWS CLI

AWS CLI sintaxe

O comando a seguir é alterado `autovacuum_vacuum_cost_limit` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: `GREATEST({log(DBInstanceClassMemory/21474836480)*600},200)` unidades de trabalho

Se você definir um valor `autovacuum_vacuum_cost_limit` muito alto, o processo de autovacuum poderá consumir muitos recursos e atrasar outras consultas. Se você definir um valor muito baixo, o processo de autovacuum pode não recuperar espaço suficiente, o que faz com que a mesa fique maior com o tempo. É essencial encontrar o equilíbrio certo que funcione para o seu sistema.

Esse parâmetro afeta somente o processo de autovacuum, não os `VACUUM` comandos manuais. Além disso, só se aplica aos processos de autovacuum para`VACUUM`, mas não para`ANALYZE`.

Ajustando os parâmetros de registro

Ajustar os parâmetros de registro no PostgreSQL ajuda a garantir que você esteja coletando as informações corretas sem produzir registros grandes que sobrecarreguem seu sistema.

A otimização dos parâmetros de registro é crucial para equilibrar os detalhes do registro com o desempenho do sistema e o uso do disco. Você pode personalizar os seguintes parâmetros de registro para capturar o nível adequado de detalhes nos registros, diagnosticar problemas e investigar incidentes de forma eficaz, minimizando o impacto no desempenho do sistema e no uso do disco.

- rds.force_autovacuum_logging
- rds.force_admin_logging_level
- duração_de_log
- log_min_duration_statement
- log_error_verbosity
- declaração_de_log
- estatísticas_de_log
- log_min_error_statement
- log_min_messages
- arquivos_temp_log
- conexões_de_log
- log_desconexões

Esses parâmetros são discutidos com mais detalhes nas seções a seguir.

> **Warning**
>
> As melhores configurações para esses parâmetros dependem das políticas e dos requisitos de conformidade da sua organização. No entanto, ativar os parâmetros de registro pode resultar em um grande número de registros e mensagens, que podem consumir o armazenamento e afetar o desempenho, especialmente em um banco de dados ocupado. Recomendamos que você use esses parâmetros com cuidado. Por exemplo, você pode

decidir ativá-los temporariamente para reduzir um problema com uma instrução SQL de baixo desempenho e desativá-los quando o período de monitoramento terminar.

rds.force_autovacuum_logging

O `rds.force_autovacuum_logging` parâmetro (disponível somente no Amazon RDS for PostgreSQL) controla se as ações de autovacuum são registradas no log do servidor. Seus valores são`disabled`,`debug5`,`debug4`,`debug3`,`debug2`,`debug1`,`info`,`notice`,`warning`,`error`,`log`,`fatal`,`pani`
O valor padrão é `warning`.

Quando você ativa`rds.force_autovacuum_logging`, todas as ações do processo de autovacuum, como quando o processo começa, quando termina e quantas linhas ele limpa, são registradas. Isso é útil para depurar ou solucionar problemas de desempenho do autovacuum.

AWS CLI sintaxe

O comando a seguir é alterado `rds.force_autovacuum_logging` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: `warning`

Exemplo

Você pode usar o `rds.force_autovacuum_logging` parâmetro para analisar o desempenho do autovacuum em uma tabela com uma taxa de gravação muito alta. Por exemplo, se sua tabela receber um grande número de operações de gravação e exclusão por segundo e você tiver um desempenho lento, você poderá ativar o parâmetro para registrar os horários de início e término de cada execução de autovacuum e determinar quantas linhas foram limpas. Isso pode fornecer informações valiosas sobre a frequência com que o autovacuum é executado, quanto tempo leva para ser executado e quantas linhas ele aspira. Em seguida, você pode usar essas informações para ajustar as configurações de autovacuum`autovacuum_vacuum_scale_factor`, como,`autovacuum_vacuum_threshold`, e `autovacuum_naptime` para otimizar o desempenho.

rds.force_admin_logging_level

O `rds.force_admin_logging_level` parâmetro (disponível somente no Amazon RDS for PostgreSQL) controla o nível de detalhe nos registros produzidos por operações administrativas, como limpeza, análise e reindexação. Ele aceita os valores `debug5`,`debug4`,`debug3`,`debug2`,`debug1`,`log`,`info`,`notice`,`warning`,`error`, `logfatal`, e `off` (padrão). A configuração ideal depende do seu caso de uso. Por exemplo, se você estiver solucionando um problema, talvez queira definir o parâmetro para um nível de depuração. Caso contrário, você pode usar a `warning` configuração `loginfo`, ou.

Se você definir `rds.force_admin_logging_level` como`debug1`, poderá registrar informações detalhadas para uma operação de reindexação, como os horários de início e término, o número de linhas processadas e quaisquer erros ou avisos que ocorram durante o processo. Isso pode fornecer informações valiosas sobre o desempenho do processo de reindexação e ajudá-lo a solucionar quaisquer problemas que ocorram.

AWS CLI sintaxe

O comando a seguir é alterado `rds.force_admin_logging_level` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: `off`

Exemplo

Você pode usar `rds.force_admin_logging_level` para monitorar e analisar o desempenho das operações administrativas em várias tabelas em um grande banco de dados. Por exemplo, digamos que você tenha um banco de dados grande com muitas tabelas e queira otimizar o desempenho dessas tabelas executando regularmente operações de limpeza e análise nelas. Ao definir o `rds.force_admin_logging_level` parâmetro como `info` ou`log`, você pode registrar os horários de início e término de cada operação e das tabelas afetadas. Você pode usar essas informações para monitorar o desempenho das operações administrativas em diferentes tabelas e identificar as tabelas que podem exigir manutenção mais frequente ou mais agressiva.

Alguns dos níveis de registro geram um grande número de arquivos e mensagens de log que podem ocupar espaço em disco rapidamente, especialmente se você tiver um banco de dados ocupado. Recomendamos que você use esse parâmetro com cuidado e o desative quando o período de monitoramento terminar.

log_duration

O `log_duration` parâmetro controla se a duração de cada consulta (ou seja, o tempo necessário para ser executada) é registrada com a consulta. Quando você define esse parâmetro comoon, o tempo necessário para executar cada consulta é incluído na saída do log junto com o texto da consulta. O tempo é medido em milissegundos.

O principal caso de uso do `log_duration` parâmetro é ajudar no ajuste de desempenho e na solução de problemas. Ao registrar a duração de cada consulta, você pode identificar as consultas que estão demorando mais para serem executadas e, em seguida, concentrar seus esforços na otimização dessas consultas. Isso pode ajudá-lo a identificar e corrigir gargalos de desempenho e a melhorar o desempenho geral do seu banco de dados.

AWS CLI sintaxe

O comando a seguir é alterado `log_duration` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: `off`

Exemplo

Você pode usar esse parâmetro se suspeitar que uma consulta específica ou um conjunto de consultas esteja causando problemas de desempenho. Ao ativar o `log_duration` parâmetro e examinar a saída do log, você pode ver quais consultas estão demorando mais para serem executadas e, em seguida, tomar as medidas apropriadas, como otimizar índices, adicionar novos índices ou reescrever a consulta.

A ativação `log_duration` pode aumentar o volume da saída do log. Recomendamos que você o use somente quando necessário e o desative durante as operações padrão para evitar encher o armazenamento ou dificultar a leitura dos registros.

log_min_duration_statement

O `log_min_duration_statement` parâmetro controla o tempo mínimo, em milissegundos, que uma instrução SQL é executada antes de ser registrada.

Esse parâmetro ajuda a identificar consultas de longa duração que podem causar problemas de desempenho. Você pode defini-lo como um valor limite (um tempo de execução considerado muito longo para uma carga de trabalho específica) para capturar consultas que excedam esse limite e identificar possíveis gargalos de desempenho. Para ver um exemplo de caso de uso, consulte Como usar parâmetros de registro para capturar variáveis de vinculação posteriormente neste guia.

AWS CLI sintaxe

O comando a seguir é alterado `log_min_duration_statement` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: 1 (desativado, que é o padrão do mecanismo PostgreSQL)

Exemplo

O comando a seguir registra qualquer instrução que leve mais de 100 milissegundos para ser executada:

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

log_error_verbosity

O `log_error_verbosity` parâmetro controla o nível de detalhe incluído na saída do log para erros e mensagens que são registradas no nível de erro ou superior. Esse parâmetro pode assumir um dos três valores: `terse`, `default` ou `verbose`.

- `terse` inclui somente o texto da mensagem, o nível do erro e o número do arquivo e da linha em que o erro ocorreu.
- `default` inclui o texto da mensagem, o nível do erro, o número do arquivo e da linha e o contexto do erro.
- `verbose` inclui o texto da mensagem, o nível do erro, o número do arquivo e da linha, o contexto do erro e a mensagem de erro completa.

Defina o parâmetro `verbose` para obter as informações mais detalhadas para solução de problemas e depuração em um ambiente que não seja de produção. Em um ambiente de produção, talvez você queira configurá-lo para `default` `terse` que ele forneça apenas informações essenciais e não preencha o armazenamento de registros com muitos detalhes.

AWS CLI sintaxe

O comando a seguir é alterado `log_error_verbosity` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: `default`

log_statement

O `log_statement` parâmetro controla quais instruções SQL são registradas no log do servidor. O parâmetro pode assumir um dos seguintes valores:

- `none`(padrão) não registra nenhuma declaração
- `ddl`registra somente instruções de linguagem de definição de dados (DDL), como e `CREATE TABLE` `ALTER TABLE`
- `mod`registra somente instruções de modificação de dados, como, e `INSERT` `UPDATE` `DELETE`
- `all`registra todas as instruções SQL

Você pode usar o `log_statement` parâmetro para controlar a quantidade de informações gravadas no registro documentando somente os tipos específicos de declarações que são relevantes para seu caso de uso.

AWS CLI sintaxe

O comando a seguir é alterado `log_statement` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: `none`

Exemplo

Em um ambiente de produção, talvez você queira configurar `log_statement` `ddl` para registrar somente instruções DDL e rastrear todas as alterações feitas no esquema do banco de dados. Em um ambiente de desenvolvimento, talvez você queira definir o parâmetro para registrar todas as instruções `all` para ajudar na depuração e na solução de problemas. Para ver outro exemplo de caso de uso, consulte Como usar parâmetros de registro para capturar variáveis de vinculação posteriormente neste guia.

A ativação `log_statement` pode aumentar o volume da saída de registros, portanto, use-a somente quando necessário e desative-a para evitar encher o armazenamento ou dificultar a leitura dos registros.

Recomendamos que você monitore seu sistema e ajuste o valor desse parâmetro para obter o equilíbrio adequado entre a quantidade de informações registradas e o armazenamento e o desempenho do sistema.

log_statement_stats

O `log_statement_stats` parâmetro controla se as estatísticas associadas à execução de uma instrução SQL são registradas com a instrução. Quando você ativa esse parâmetro, estatísticas como o número de linhas afetadas, o número de blocos de disco lidos e gravados e o tempo necessário para executar a instrução são incluídos na saída do log.

Você pode usar o `log_statement_stats` parâmetro para coletar informações adicionais sobre o desempenho de declarações individuais e a carga de trabalho geral. Ao registrar estatísticas de instruções, você pode identificar padrões no desempenho da consulta e no uso de recursos e usar essas informações para otimizar seu banco de dados e melhorar o desempenho geral.

AWS CLI sintaxe

O comando a seguir é alterado `log_statement_stats` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: `off` (padrão do mecanismo PostgreSQL); use 0 ou 1 (booleano) para definir grupos de parâmetros

Exemplo

Você pode usar `log_statement_stats` para analisar o comportamento de uma consulta específica, ver como ela usa recursos como CPU, memória e E/S de disco e identificar se a consulta pode ser otimizada. Você também pode usar esse parâmetro para ver se uma tabela específica é lida com frequência (o que pode indicar a necessidade de criar um índice em uma coluna específica) ou se uma tabela é escaneada com muita frequência.

A ativação `log_statement_stats` pode aumentar o volume da saída de registros, portanto, use-a somente quando necessário e desative-a para evitar encher o armazenamento ou dificultar a leitura dos registros.

log_min_error_statement

O `log_min_error_statement` parâmetro controla quais instruções SQL que resultam em um erro serão registradas. Seus valores são `debug5``debug4`,`debug3`,`debug2`,`debug1`,`info`,`notice`, `warning``error`,`log`,`fatal`,, `panic` e. Essas configurações controlam a quantidade de informações gravadas no registro para que você possa filtrar mensagens de menor gravidade. Você pode definir esse parâmetro para um nível de severidade mais alto para reduzir a quantidade de saída de log e encontrar mensagens importantes com mais facilidade.

AWS CLI sintaxe

O comando a seguir é alterado `log_min_error_statement` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: `error` (padrão do mecanismo PostgreSQL)

Exemplo

Você pode considerar o uso `log_min_error_statement` ao solucionar um problema específico e quiser ver mensagens de erro de instruções SQL que estão causando erros.

log_min_messages

O `log_min_messages` parâmetro controla o nível de severidade gravado no registro. Você pode definir o parâmetro como`debug5`,`debug4`, `debug3``debug2`,`debug1`,`info`,`notice`,`warning`,`error`,`log`,`fatal`, ou`panic`. Essas configurações controlam a quantidade de informações gravadas no registro para que você possa filtrar mensagens de menor gravidade. Você pode definir esse parâmetro para um nível de severidade mais alto para reduzir a quantidade de saída de log e encontrar mensagens importantes com mais facilidade.

AWS CLI sintaxe

O comando a seguir é alterado `log_min_messages` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: `notice`

Exemplo

Se você estiver solucionando um problema específico e quiser ver todas as mensagens de erro, defina esse parâmetro `error` para registrar somente erros e problemas de gravidade de nível superior. Se você estiver interessado em monitorar o desempenho do sistema, você pode definir esse parâmetro `info` para ver informações mais detalhadas, como a duração e as estatísticas de cada declaração.

`log_min_messages`A configuração para um nível de severidade mais alto diminui o volume de registros. Recomendamos que você ajuste esse parâmetro dependendo do seu caso de uso específico, do tamanho do log que você deseja verificar e da quantidade de espaço em disco que você tem.

log_temp_files

O `log_temp_files` parâmetro controla o registro de nomes e tamanhos de arquivos temporários. Ela se aplica a arquivos temporários criados para fins como classificações, hashes e resultados de consultas temporárias. Quando esse parâmetro é ativado, uma entrada de registro é gerada para cada arquivo temporário após a exclusão, incluindo o tamanho do arquivo, em bytes. Você pode definir esse parâmetro como 0 (zero) para um registro abrangente de todas as informações do arquivo temporário ou como um valor positivo para arquivos de log que excedam esse tamanho (em kilobytes, se as unidades não forem especificadas). Isso pode ser útil para identificar e resolver gargalos de desempenho ou outros problemas relacionados ao armazenamento temporário. Por padrão, o registro de arquivos temporários está desativado.

AWS CLI sintaxe

O comando a seguir é alterado `log_temp_files` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: -1 (padrão do mecanismo PostgreSQL)

Exemplo

Você pode ativar esse parâmetro se suspeitar que o sistema está usando muito armazenamento temporário ou que os arquivos temporários não estão sendo excluídos adequadamente. Ao examinar a saída do log, você pode ver as consultas ou operações que estão gerando arquivos temporários e como esses arquivos estão sendo usados.

Algumas consultas ou operações criam um grande número de arquivos temporários, portanto, a ativação `log_temp_files` pode afetar o desempenho geral do sistema.

log_connections

O `log_connections` parâmetro controla se as conexões com o banco de dados são registradas. Quando você define esse parâmetro comoon, o log contém informações sobre cada conexão bem-sucedida com o banco de dados, como o endereço IP do cliente, o nome de usuário, o nome do banco de dados e a data e a hora da conexão.

Você pode usar o `log_connections` parâmetro para monitorar e solucionar problemas de conexões com o banco de dados. Você pode ver os usuários, aplicativos, terminais e bots que se conectam ao banco de dados, de onde estão se conectando e com que frequência. Essas informações podem ser úteis para identificar e resolver problemas relacionados à conexão ou rastrear padrões de uso.

AWS CLI sintaxe

O comando a seguir é alterado `log_connections` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: `off` (padrão do mecanismo PostgreSQL)

Exemplo

Você pode usar esse parâmetro se suspeitar que muitas conexões com o banco de dados ou um usuário ou endereço IP específico que esteja se conectando com muita frequência estejam afetando o desempenho. Ao ativar o `log_connections` parâmetro e examinar a saída do log, você pode ver o número e os detalhes de todas as conexões.

Antes de ativar esse parâmetro, verifique as políticas da sua organização e considere as implicações de segurança do registro de endereços IP e nomes de usuário.

log_disconnections

O `log_disconnections` parâmetro controla o registro de desconexões do banco de dados. Quando você define esse parâmetro comoon, ele registra informações sobre o final de cada sessão, como o endereço IP do cliente, o nome do nome de usuário, o nome do banco de dados e a data e a hora da desconexão.

Você pode usar o `log_disconnections` parâmetro para monitorar e solucionar problemas de encerramentos de sessões de banco de dados. Você pode ver os usuários, aplicativos, terminais e bots que se desconectam do banco de dados, quando e por quê. Por exemplo, você pode analisar encerramentos inesperados, como uma falha ou desconexões iniciadas pelo administrador. Essas informações podem ser úteis para identificar e resolver problemas relacionados a desconexões ou rastrear padrões de uso.

AWS CLI sintaxe

O comando a seguir é alterado `log_disconnections` para um grupo de parâmetros de banco de dados específico. Essa alteração se aplica a todas as instâncias ou clusters que usam o grupo de parâmetros.

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

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

Tipo: Dinâmico (as alterações são aplicadas imediatamente se você definir`ApplyMethod=immediate`)

Valor padrão: `off` (padrão do mecanismo PostgreSQL)

Exemplo

Você pode usar `log_disconnections` se suspeitar que há muitos usuários se desconectando do banco de dados ou que um usuário ou endereço IP específico está se desconectando com muita

frequência. Ao ativar o `log_disconnections` parâmetro e examinar a saída do log, você pode ver o número e os detalhes de todas as desconexões, incluindo quem, quando e se houve algum erro antes da desconexão.

Antes de ativar esse parâmetro, verifique as políticas da sua organização e considere as implicações de segurança do registro de endereços IP e nomes de usuário.

Usando parâmetros de registro para capturar variáveis de vinculação

Um caso de uso típico para capturar variáveis de vinculação no PostgreSQL é depurar e ajustar o desempenho de consultas SQL. Uma variável de associação permite que você passe dados para uma consulta ao executá-la. Ao capturar as variáveis de associação, você pode ver os dados de entrada que foram passados para uma consulta, o que pode ajudá-lo a identificar quaisquer problemas com os dados ou com o desempenho da consulta. A captura das variáveis de vinculação também pode ajudá-lo a auditar os dados de entrada e detectar possíveis riscos de segurança ou atividades maliciosas.

Há várias maneiras de capturar variáveis de vinculação para o PostgreSQL. Um método é ativar os `debug_print_rewritten` parâmetros `debug_print_parse` e. Isso faz com que o PostgreSQL envie as versões analisadas e reescritas das instruções SQL, junto com as variáveis vinculadas, para o log do servidor.

- `debug_print_parse`: quando você ativa esse parâmetro, a árvore de análise das consultas recebidas é impressa no log do servidor. Isso pode ser útil para entender a estrutura de uma consulta e os valores de qualquer parâmetro vinculado.
- `debug_print_rewritten`: quando você ativa esse parâmetro, as formas reescritas das consultas recebidas são impressas no log do servidor. Isso pode ser útil para entender como o planejador de consultas interpreta uma consulta e os valores de quaisquer parâmetros vinculados.

Você pode usar dois parâmetros adicionais no Amazon RDS e no Aurora para capturar variáveis de associação em seus bancos de dados PostgreSQL:

- `log_min_duration_statement`: esse parâmetro define a duração mínima de uma instrução antes de ser registrada, em milissegundos. Quando uma instrução demora mais do que a duração especificada, seus valores de vinculação são incluídos na saída do log.

- `log_statement`: esse parâmetro controla quais instruções SQL são registradas. Defina esse parâmetro como `all` ou vincule para incluir os valores vinculados no registro. O aumento do nível de registro afeta o desempenho, por isso recomendamos que você reverta as alterações após a solução de problemas.

Você também pode usar a `pg_stat_statements` extensão, que fornece estatísticas de desempenho para todas as instruções SQL executadas por um servidor, incluindo o texto da consulta e os valores vinculados. Essa extensão permite que você use o pGADmin ou ferramentas similares para monitorar e analisar o desempenho da consulta.

Outra opção é usar a `pg_bind_parameter_status()` função para obter os valores dos parâmetros vinculados de uma instrução preparada ou usar a `pg_get_parameter_status (paramname)` função para recuperar o status ou o valor de um parâmetro de tempo de execução específico.

Além disso, você pode usar ferramentas de terceiros, como o pGBadger, para analisar os registros do PostgreSQL e extrair as variáveis de vinculação e outras informações para análise posterior.

Ajustando os parâmetros de replicação

No PostgreSQL, você pode replicar as alterações de dados de um banco de dados PostgreSQL para outro usando a replicação lógica em vez da replicação física baseada em arquivos. A replicação lógica usa o log de gravação antecipada (WAL) para capturar alterações e oferece suporte à replicação de tabelas selecionadas ou bancos de dados inteiros.

Tanto o Amazon RDS for PostgreSQL quanto o Aurora PostgreSQL, compatíveis com o Aurora PostgreSQL, oferecem suporte à replicação lógica, para que você possa configurar uma arquitetura de banco de dados altamente disponível e escalável que possa lidar com tráfego de leitura e gravação de várias fontes. Esses serviços usam pglogical, que é uma extensão PostgreSQL de código aberto, para implementar a replicação lógica.

Ajustar a replicação lógica no Aurora e no Amazon RDS é importante para alcançar desempenho, escalabilidade e disponibilidade ideais. Você pode ajustar os parâmetros na extensão pglogical para gerenciar o desempenho da replicação lógica. Por exemplo, é possível:

- Melhore o desempenho da replicação aumentando o número de processos de trabalho ou ajustando sua alocação de memória.
- Reduza o risco de atraso na replicação ajustando a frequência de sincronização entre os bancos de dados de origem e de réplica.
- Otimize o uso de recursos ajustando a alocação de memória e CPU dos processos de trabalho.
- Certifique-se de que o processo de replicação não cause impacto indevido no desempenho do banco de dados de origem.

Você pode usar os seguintes parâmetros no Aurora e no Amazon RDS para controlar e configurar a replicação lógica:

- `max_replication_slots`define o número máximo de slots de replicação que podem ser criados no servidor. Um slot de replicação é uma reserva nomeada e persistente para que uma conexão de replicação envie dados WAL para uma réplica.
- `max_wal_senders`define o número máximo de processos de remetente WAL conectados simultaneamente. Os processos do remetente do WAL são usados para transmitir o WAL do servidor primário para a réplica.
- `wal_sender_timeout`define o tempo máximo, em milissegundos, que um remetente do WAL espera por uma resposta da réplica antes de desistir e se reconectar.

- `wal_receiver_timeout`define o tempo máximo, em milissegundos, que uma réplica espera pelos dados do WAL do banco de dados primário antes de atingir o tempo limite.
- `log_replication_commands`, quando definido comoon, executa as instruções SQL relacionadas à replicação.

Quando você ativa o `rds.logical_replication` parâmetro (definindo-o como 1), o `wal_level` parâmetro é definido como`logical`, o que significa que todas as alterações feitas no banco de dados são gravadas no WAL em um formato que pode ser lido e aplicado a uma réplica. Essa configuração é necessária para habilitar a replicação lógica. Essa configuração também permite a replicação de `SELECT` instruções.

A configuração `wal_level` para `logical` pode aumentar a quantidade de dados gravados no WAL e, portanto, no disco, o que pode afetar o desempenho do sistema. Recomendamos que você considere o espaço em disco disponível e o desempenho do sistema ao habilitar a replicação lógica.

Exemplo

Você deseja replicar dados do seu banco de dados principal para um banco de dados secundário para fins de backup e recuperação de desastres. No entanto, o banco de dados secundário tem um alto volume de operações de leitura, então você quer garantir que o processo de replicação seja o mais rápido e eficiente possível sem comprometer a integridade dos dados.

Os valores padrão para replicação lógica no Amazon RDS e no Aurora priorizam a consistência em detrimento do desempenho, portanto, eles podem não ser ideais para esse caso de uso. Para otimizar suas configurações de replicação lógica em termos de velocidade e eficiência, você pode personalizar os parâmetros da seguinte forma:

- Aumente `max_replication_slots` de 10 (padrão para Amazon RDS) ou 20 (padrão para Aurora) para 30 para acomodar possíveis necessidades futuras de crescimento e replicação.
- Aumente `max_wal_senders` de 10 (padrão) para 20 para garantir que haja processos de remetente WAL suficientes para acompanhar a demanda de replicação.
- Diminua `wal_sender_timeout` de 30 segundos (padrão) para 15 segundos para garantir que os processos ociosos do remetente do WAL sejam encerrados mais rapidamente, o que libera recursos para a replicação ativa.

- Diminua `wal_receiver_timeout` de 30 segundos (padrão) para 15 segundos para garantir que os processos ociosos do receptor WAL sejam encerrados mais rapidamente, o que libera recursos para a replicação ativa.
- Aumente `max_logical_replication_workers` de 4 (padrão) para 8 para garantir que haja processos de trabalho de replicação lógica suficientes para atender à demanda de replicação.

Essas otimizações fornecem uma replicação de dados mais rápida e eficiente, mantendo a integridade e a segurança dos dados.

Por exemplo, se um desastre ocorresse e o banco de dados principal ficasse indisponível, o banco de dados secundário já teria os dados mais recentes disponíveis devido ao processo de replicação otimizado. Isso permitiria que suas operações comerciais continuassem fornecendo serviços essenciais sem interrupção.

Práticas recomendadas

Ajustar a replicação lógica com grandes cargas de trabalho pode ser uma tarefa complexa que depende de vários fatores, incluindo o tamanho do conjunto de dados, o número de tabelas que estão sendo replicadas, o número de réplicas e os recursos disponíveis. Aqui estão algumas dicas gerais para ajustar a replicação lógica com grandes cargas de trabalho:

- Monitore o atraso na replicação. O atraso na replicação é a diferença horária entre o servidor primário e os servidores em espera. O monitoramento do atraso na replicação pode ajudá-lo a identificar possíveis gargalos e tomar medidas para melhorar o desempenho da replicação. Você pode usar a `pg_current_wal_lsn()` função para verificar o atraso de replicação atual.
- Ajuste as configurações do WAL. A `pg_logical` extensão usa o WAL para transmitir as alterações do servidor primário para o servidor em espera. Se as configurações do WAL não forem ajustadas adequadamente, a replicação pode se tornar lenta e não confiável. Certifique-se de definir os `max_replication_slots` parâmetros `max_wal_senders` e para valores adequados, dependendo de suas cargas de trabalho.
- Tenha uma estratégia de indexação. Ter índices adequados no servidor primário pode ajudar a melhorar o desempenho da replicação lógica, reduzir a E/S no servidor primário e reduzir a carga no sistema.
- Use a replicação paralela. O uso da replicação paralela pode ajudar a aumentar a velocidade de replicação, permitindo que vários processos de trabalho paralelos repliquem dados. Esse recurso está disponível no PostgreSQL 12 e versões posteriores.

Próximas etapas

Depois de otimizar os parâmetros de memória, replicação, autovacuum e registro para seu banco de dados compatível com Amazon RDS for PostgreSQL ou Aurora PostgreSQL, considere estas etapas para melhorar ainda mais o desempenho do seu banco de dados:

- Monitore seu banco de dados. Acompanhe o desempenho do seu banco de dados ao longo do tempo usando ferramentas de monitoramento integradas ou soluções de terceiros. Monitore as principais métricas de desempenho, como utilização da CPU, E/S de disco, uso de memória e tempos de execução de consultas, para identificar possíveis gargalos e áreas de melhoria.
- Ajuste continuamente os parâmetros. À medida que sua carga de trabalho evolui, continue monitorando e ajustando os parâmetros do banco de dados para garantir o desempenho ideal. Verifique regularmente os registros do sistema, as mensagens de erro e as métricas de desempenho para identificar novas oportunidades de ajuste.
- Implemente o armazenamento em cache. Use o armazenamento em cache para reduzir o número de consultas que chegam ao banco de dados. Você pode implementar o armazenamento em cache no nível do aplicativo usando ferramentas como Memcached ou Redis, ou você pode usar a Amazon ElastiCache para fornecer um cache na memória para seu banco de dados.
- Otimize suas consultas. Consultas mal projetadas podem afetar significativamente o desempenho do banco de dados. Use `EXPLAIN` outras ferramentas de ajuste de consultas para identificar consultas lentas, otimizá-las e eliminar consultas desnecessárias.

Seguindo essas diretrizes, você pode otimizar o desempenho do seu banco de dados Aurora ou Amazon RDS for PostgreSQL e garantir que ele atenda às necessidades do seu aplicativo com melhor desempenho do banco de dados, maior confiabilidade, menor tempo de inatividade, melhor segurança e economia de custos. Ao otimizar os parâmetros de configuração para se adequar à sua carga de trabalho, você pode garantir que seu banco de dados esteja funcionando com eficiência e usando os recursos de forma eficaz, levando a um melhor desempenho e a um aplicativo mais responsivo. Além disso, parâmetros configurados adequadamente podem reduzir a probabilidade de erros e vulnerabilidades, resultando em maior confiabilidade e melhor segurança. Isso pode se traduzir em economia de custos em termos de redução de manutenção e tempo de inatividade, bem como melhor experiência geral e satisfação do usuário.

Recursos

- Parâmetros do Amazon Aurora PostgreSQL, parte 1: gerenciamento de planos de consulta e memória (publicação no blog)AWS
- Parâmetros do Amazon Aurora PostgreSQL, parte 2: replicação, segurança e registro (publicação no blog)AWS
- Parâmetros do Amazon Aurora PostgreSQL, parte 3: parâmetros do otimizador (postagem do blog)AWS
- Parâmetros do Amazon Aurora PostgreSQL, parte 4: opções de compatibilidade ANSI (postagem do blog)AWS
- Trabalhando com o Amazon Aurora PostgreSQL (documentação)AWS
- Trabalhando com o Amazon RDS for PostgreSQL (documentação)AWS
- Monitorando a carga do banco de dados com Performance Insights no Amazon RDS (AWS documentação)
- Usando CloudWatch métricas da Amazon (AWS documentação)
- pg_stats_statements (documentação do PostgreSQL)

Histórico do documento

A tabela a seguir descreve alterações significativas feitas neste guia. Se desejar receber notificações sobre futuras atualizações, inscreva-se em um feed RSS.

Alteração	Descrição	Data
Informações atualizadas sobre parâmetros de memória e autovacuum	Atualizou a descrição do parâmetro random_page_cost; adicionou unidades ausentes aos valores padrão dos parâmetros de memória e autovacuum; atualizou a sintaxe do AWS CLI parâmetro max_connections.	27 de fevereiro de 2024
Informações atualizadas sobre `autovacuum`	Corrigida a configuração padrão de autovacuum (ativada).	27 de dezembro de 2023
Informações atualizadas sobre `max_connections`	A seção max_connections foi atualizada com novas orientações sobre o ajuste desse parâmetro.	15 de novembro de 2023
Publicação inicial	—	31 de outubro de 2023

AWS Glossário de orientação prescritiva

A seguir estão os termos comumente usados em estratégias, guias e padrões fornecidos pela Orientação AWS Prescritiva. Para sugerir entradas, use o link Fornecer feedback no final do glossário.

Números

7 Rs

Sete estratégias comuns de migração para mover aplicações para a nuvem. Essas estratégias baseiam-se nos 5 Rs identificados pela Gartner em 2011 e consistem em:

- Refatorar/rearquitetar: mova uma aplicação e modifique sua arquitetura aproveitando ao máximo os recursos nativos de nuvem para melhorar a agilidade, a performance e a escalabilidade. Isso normalmente envolve a portabilidade do sistema operacional e do banco de dados. Exemplo: migre seu banco de dados Oracle local para a edição compatível com o Amazon Aurora PostgreSQL.
- Redefinir a plataforma (mover e redefinir [mover e redefinir (lift-and-reshape]): mova uma aplicação para a nuvem e introduza algum nível de otimização a fim de aproveitar os recursos da nuvem. Exemplo: Migre seu banco de dados Oracle local para o Amazon Relational Database Service (Amazon RDS) for Oracle no. Nuvem AWS
- Recomprar (drop and shop): mude para um produto diferente, normalmente migrando de uma licença tradicional para um modelo SaaS. Exemplo: migre seu sistema de gerenciamento de relacionamento com o cliente (CRM) para a Salesforce.com.
- Redefinir a hospedagem (mover sem alterações [lift-and-shift])mover uma aplicação para a nuvem sem fazer nenhuma alteração a fim de aproveitar os recursos da nuvem. Exemplo: Migre seu banco de dados Oracle local para o Oracle em uma instância do EC2 no. Nuvem AWS
- Realocar (mover o hipervisor sem alterações [hypervisor-level lift-and-shift]): mover a infraestrutura para a nuvem sem comprar novo hardware, reescrever aplicações ou modificar suas operações existentes. Você migra servidores de uma plataforma local para um serviço em nuvem para a mesma plataforma. Exemplo: migrar um Microsoft Hyper-V aplicativo para o. AWS
- Reter (revisitar): mantenha as aplicações em seu ambiente de origem. Isso pode incluir aplicações que exigem grande refatoração, e você deseja adiar esse trabalho para um

momento posterior, e aplicações antigas que você deseja manter porque não há justificativa comercial para migrá-las.

- Retirar: desative ou remova aplicações que não são mais necessárias em seu ambiente de origem.

A

ABAC

Consulte controle de acesso baseado em atributos.

serviços abstratos

Veja os serviços gerenciados.

ACID

Veja atomicidade, consistência, isolamento, durabilidade.

migração ativa-ativa

Um método de migração de banco de dados no qual os bancos de dados de origem e de destino são mantidos em sincronia (por meio de uma ferramenta de replicação bidirecional ou operações de gravação dupla), e ambos os bancos de dados lidam com transações de aplicações conectadas durante a migração. Esse método oferece suporte à migração em lotes pequenos e controlados, em vez de exigir uma substituição única. É mais flexível, mas exige mais trabalho do que a migração ativa-passiva.

migração ativa-passiva

Um método de migração de banco de dados no qual os bancos de dados de origem e de destino são mantidos em sincronia, mas somente o banco de dados de origem manipula as transações das aplicações conectadas enquanto os dados são replicados no banco de dados de destino. O banco de dados de destino não aceita nenhuma transação durante a migração.

função agregada

Uma função SQL que opera em um grupo de linhas e calcula um único valor de retorno para o grupo. Exemplos de funções agregadas incluem `SUM` e. `MAX`

AI

Veja inteligência artificial.

AIOps

Veja as operações de inteligência artificial.

anonimização

O processo de excluir permanentemente informações pessoais em um conjunto de dados. A anonimização pode ajudar a proteger a privacidade pessoal. Dados anônimos não são mais considerados dados pessoais.

antipadrões

Uma solução frequentemente usada para um problema recorrente em que a solução é contraproducente, ineficaz ou menos eficaz do que uma alternativa.

controle de aplicativos

Uma abordagem de segurança que permite o uso somente de aplicativos aprovados para ajudar a proteger um sistema contra malware.

portfólio de aplicações

Uma coleção de informações detalhadas sobre cada aplicação usada por uma organização, incluindo o custo para criar e manter a aplicação e seu valor comercial. Essas informações são fundamentais para o processo de descoberta e análise de portfólio e ajudam a identificar e priorizar as aplicações a serem migradas, modernizadas e otimizadas.

inteligência artificial (IA)

O campo da ciência da computação que se dedica ao uso de tecnologias de computação para desempenhar funções cognitivas normalmente associadas aos humanos, como aprender, resolver problemas e reconhecer padrões. Para obter mais informações, consulte O que é inteligência artificial?

operações de inteligência artificial (AIOps)

O processo de usar técnicas de machine learning para resolver problemas operacionais, reduzir incidentes operacionais e intervenção humana e aumentar a qualidade do serviço. Para obter mais informações sobre como as AIOps são usadas na estratégia de migração para a AWS , consulte o guia de integração de operações.

criptografia assimétrica

Um algoritmo de criptografia que usa um par de chaves, uma chave pública para criptografia e uma chave privada para descriptografia. É possível compartilhar a chave pública porque ela não é usada na descriptografia, mas o acesso à chave privada deve ser altamente restrito.

atomicidade, consistência, isolamento, durabilidade (ACID)

Um conjunto de propriedades de software que garantem a validade dos dados e a confiabilidade operacional de um banco de dados, mesmo no caso de erros, falhas de energia ou outros problemas.

controle de acesso por atributo (ABAC)

A prática de criar permissões minuciosas com base nos atributos do usuário, como departamento, cargo e nome da equipe. Para obter mais informações, consulte ABAC AWS na documentação AWS Identity and Access Management (IAM).

fonte de dados autorizada

Um local onde você armazena a versão principal dos dados, que é considerada a fonte de informações mais confiável. Você pode copiar dados da fonte de dados autorizada para outros locais com o objetivo de processar ou modificar os dados, como anonimizá-los, redigi-los ou pseudonimizá-los.

Availability Zone (zona de disponibilidade)

Um local distinto dentro de um Região da AWS que está isolado de falhas em outras zonas de disponibilidade e fornece conectividade de rede barata e de baixa latência a outras zonas de disponibilidade na mesma região.

AWS Estrutura de adoção da nuvem (AWS CAF)

Uma estrutura de diretrizes e melhores práticas AWS para ajudar as organizações a desenvolver um plano eficiente e eficaz para migrar com sucesso para a nuvem. AWS O CAF organiza a orientação em seis áreas de foco chamadas perspectivas: negócios, pessoas, governança, plataforma, segurança e operações. As perspectivas de negócios, pessoas e governança têm como foco habilidades e processos de negócios; as perspectivas de plataforma, segurança e operações concentram-se em habilidades e processos técnicos. Por exemplo, a perspectiva das pessoas tem como alvo as partes interessadas que lidam com recursos humanos (RH), funções de pessoal e gerenciamento de pessoal. Nessa perspectiva, o AWS CAF fornece orientação para desenvolvimento, treinamento e comunicação de pessoas para ajudar a preparar a organização

para a adoção bem-sucedida da nuvem. Para obter mais informações, consulte o site da AWS CAF e o whitepaper da AWS CAF.

AWS Estrutura de qualificação da carga de trabalho (AWS WQF)

Uma ferramenta que avalia as cargas de trabalho de migração do banco de dados, recomenda estratégias de migração e fornece estimativas de trabalho. AWS O WQF está incluído com AWS Schema Conversion Tool ()AWS SCT. Ela analisa esquemas de banco de dados e objetos de código, código de aplicações, dependências e características de performance, além de fornecer relatórios de avaliação.

B

bot ruim

Um bot destinado a perturbar ou causar danos a indivíduos ou organizações.

BCP

Veja o planejamento de continuidade de negócios.

gráfico de comportamento

Uma visualização unificada e interativa do comportamento e das interações de recursos ao longo do tempo. É possível usar um gráfico de comportamento com o Amazon Detective para examinar tentativas de login malsucedidas, chamadas de API suspeitas e ações similares. Para obter mais informações, consulte Dados em um gráfico de comportamento na documentação do Detective.

sistema big-endian

Um sistema que armazena o byte mais significativo antes. Veja também endianness.

classificação binária

Um processo que prevê um resultado binário (uma de duas classes possíveis). Por exemplo, seu modelo de ML pode precisar prever problemas como “Este e-mail é ou não é spam?” ou “Este produto é um livro ou um carro?”

filtro de bloom

Uma estrutura de dados probabilística e eficiente em termos de memória que é usada para testar se um elemento é membro de um conjunto.

blue/green deployment (implantação azul/verde)

Uma estratégia de implantação em que você cria dois ambientes separados, mas idênticos. Você executa a versão atual do aplicativo em um ambiente (azul) e a nova versão do aplicativo no outro ambiente (verde). Essa estratégia ajuda você a reverter rapidamente com o mínimo de impacto.

bot

Um aplicativo de software que executa tarefas automatizadas pela Internet e simula a atividade ou interação humana. Alguns bots são úteis ou benéficos, como rastreadores da Web que indexam informações na Internet. Alguns outros bots, conhecidos como bots ruins, têm como objetivo perturbar ou causar danos a indivíduos ou organizações.

botnet

Redes de bots infectadas por malware e sob o controle de uma única parte, conhecidas como pastor de bots ou operador de bots. As redes de bots são o mecanismo mais conhecido para escalar bots e seu impacto.

ramo

Uma área contida de um repositório de código. A primeira ramificação criada em um repositório é a ramificação principal. Você pode criar uma nova ramificação a partir de uma ramificação existente e, em seguida, desenvolver recursos ou corrigir bugs na nova ramificação. Uma ramificação que você cria para gerar um recurso é comumente chamada de ramificação de recurso. Quando o recurso estiver pronto para lançamento, você mesclará a ramificação do recurso de volta com a ramificação principal. Para obter mais informações, consulte Sobre filiais (GitHub documentação).

acesso em vidro quebrado

Em circunstâncias excepcionais e por meio de um processo aprovado, um meio rápido para um usuário obter acesso a um Conta da AWS que ele normalmente não tem permissão para acessar. Para obter mais informações, consulte o indicador Implementar procedimentos de quebra de vidro na orientação do Well-Architected AWS .

estratégia brownfield

A infraestrutura existente em seu ambiente. Ao adotar uma estratégia brownfield para uma arquitetura de sistema, você desenvolve a arquitetura de acordo com as restrições dos sistemas e da infraestrutura atuais. Se estiver expandindo a infraestrutura existente, poderá combinar as estratégias brownfield e greenfield.

cache do buffer

A área da memória em que os dados acessados com mais frequência são armazenados.

capacidade de negócios

O que uma empresa faz para gerar valor (por exemplo, vendas, atendimento ao cliente ou marketing). As arquiteturas de microsserviços e as decisões de desenvolvimento podem ser orientadas por recursos de negócios. Para obter mais informações, consulte a seção Organizados de acordo com as capacidades de negócios do whitepaper Executar microsserviços conteinerizados na AWS.

planejamento de continuidade de negócios (BCP)

Um plano que aborda o impacto potencial de um evento disruptivo, como uma migração em grande escala, nas operações e permite que uma empresa retome as operações rapidamente.

C

CAF

Consulte Estrutura de adoção da AWS nuvem.

implantação canária

O lançamento lento e incremental de uma versão para usuários finais. Quando estiver confiante, você implanta a nova versão e substituirá a versão atual em sua totalidade.

CCoE

Veja o Centro de Excelência em Nuvem.

CDC

Veja a captura de dados de alterações.

captura de dados de alterações (CDC)

O processo de rastrear alterações em uma fonte de dados, como uma tabela de banco de dados, e registrar metadados sobre a alteração. É possível usar o CDC para várias finalidades, como auditar ou replicar alterações em um sistema de destino para manter a sincronização.

engenharia do caos

Introduzir intencionalmente falhas ou eventos disruptivos para testar a resiliência de um sistema. Você pode usar AWS Fault Injection Service (AWS FIS) para realizar experimentos que estressam suas AWS cargas de trabalho e avaliar sua resposta.

CI/CD

Veja a integração e a entrega contínuas.

classificação

Um processo de categorização que ajuda a gerar previsões. Os modelos de ML para problemas de classificação predizem um valor discreto. Os valores discretos são sempre diferentes uns dos outros. Por exemplo, um modelo pode precisar avaliar se há ou não um carro em uma imagem.

criptografia no lado do cliente

Criptografia de dados localmente, antes que o alvo os AWS service (Serviço da AWS) receba.

Centro de Excelência da Nuvem (CCoE)

Uma equipe multidisciplinar que impulsiona os esforços de adoção da nuvem em toda a organização, incluindo o desenvolvimento de práticas recomendas de nuvem, a mobilização de recursos, o estabelecimento de cronogramas de migração e a liderança da organização em transformações em grande escala. Para obter mais informações, consulte as postagens do CCoE no blog de estratégia Nuvem AWS corporativa.

computação em nuvem

A tecnologia de nuvem normalmente usada para armazenamento de dados remoto e gerenciamento de dispositivos de IoT. A computação em nuvem geralmente está conectada à tecnologia de computação de ponta.

modelo operacional em nuvem

Em uma organização de TI, o modelo operacional usado para criar, amadurecer e otimizar um ou mais ambientes de nuvem. Para obter mais informações, consulte Criar seu modelo operacional de nuvem.

estágios de adoção da nuvem

As quatro fases pelas quais as organizações normalmente passam quando migram para o Nuvem AWS:

- Projeto: executar alguns projetos relacionados à nuvem para fins de prova de conceito e aprendizado
- Fundação: realizar investimentos fundamentais para escalar sua adoção da nuvem (por exemplo, criar uma zona de pouso, definir um CCoE, estabelecer um modelo de operações)
- Migração: migrar aplicações individuais
- Reinvenção: otimizar produtos e serviços e inovar na nuvem

Esses estágios foram definidos por Stephen Orban na postagem do blog The Journey Toward Cloud-First & the Stages of Adoption no blog de estratégia Nuvem AWS empresarial. Para obter informações sobre como eles se relacionam com a estratégia de AWS migração, consulte o guia de preparação para migração.

CMDB

Consulte o banco de dados de gerenciamento de configuração.

repositório de código

Um local onde o código-fonte e outros ativos, como documentação, amostras e scripts, são armazenados e atualizados por meio de processos de controle de versão. Os repositórios de nuvem comuns incluem GitHub ou AWS CodeCommit. Cada versão do código é chamada de ramificação. Em uma estrutura de microsserviços, cada repositório é dedicado a uma única peça de funcionalidade. Um único pipeline de CI/CD pode usar vários repositórios.

cache frio

Um cache de buffer que está vazio, não está bem preenchido ou contém dados obsoletos ou irrelevantes. Isso afeta a performance porque a instância do banco de dados deve ler da memória principal ou do disco, um processo que é mais lento do que a leitura do cache do buffer.

dados frios

Dados que raramente são acessados e geralmente são históricos. Ao consultar esse tipo de dados, consultas lentas geralmente são aceitáveis. Mover esses dados para níveis ou classes de armazenamento de baixo desempenho e menos caros pode reduzir os custos.

visão computacional (CV)

Um campo da IA que usa aprendizado de máquina para analisar e extrair informações de formatos visuais, como imagens e vídeos digitais. Por exemplo, AWS Panorama oferece dispositivos que adicionam CV às redes de câmeras locais, e a Amazon SageMaker fornece algoritmos de processamento de imagem para CV.

desvio de configuração

Para uma carga de trabalho, uma alteração de configuração em relação ao estado esperado. Isso pode fazer com que a carga de trabalho se torne incompatível e, normalmente, é gradual e não intencional.

banco de dados de gerenciamento de configuração (CMDB)

Um repositório que armazena e gerencia informações sobre um banco de dados e seu ambiente de TI, incluindo componentes de hardware e software e suas configurações. Normalmente, os dados de um CMDB são usados no estágio de descoberta e análise do portfólio da migração.

pacote de conformidade

Um conjunto de AWS Config regras e ações de remediação que você pode montar para personalizar suas verificações de conformidade e segurança. Você pode implantar um pacote de conformidade como uma entidade única em uma Conta da AWS região ou em uma organização usando um modelo YAML. Para obter mais informações, consulte Pacotes de conformidade na documentação. AWS Config

integração contínua e entrega contínua (CI/CD)

O processo de automatizar os estágios de origem, criação, teste, preparação e produção do processo de lançamento do software. O CI/CD é comumente descrito como um pipeline. O CI/CD pode ajudar você a automatizar processos, melhorar a produtividade, melhorar a qualidade do código e entregar com mais rapidez. Para obter mais informações, consulte Benefícios da entrega contínua. CD também pode significar implantação contínua. Para obter mais informações, consulte Entrega contínua versus implantação contínua.

CV

Veja visão computacional.

D

dados em repouso

Dados estacionários em sua rede, por exemplo, dados que estão em um armazenamento.

classificação de dados

Um processo para identificar e categorizar os dados em sua rede com base em criticalidade e confidencialidade. É um componente crítico de qualquer estratégia de gerenciamento de riscos de

segurança cibernética, pois ajuda a determinar os controles adequados de proteção e retenção para os dados. A classificação de dados é um componente do pilar de segurança no AWS Well-Architected Framework. Para obter mais informações, consulte Classificação de dados.

desvio de dados

Uma variação significativa entre os dados de produção e os dados usados para treinar um modelo de ML ou uma alteração significativa nos dados de entrada ao longo do tempo. O desvio de dados pode reduzir a qualidade geral, a precisão e a imparcialidade das previsões do modelo de ML.

dados em trânsito

Dados que estão se movendo ativamente pela sua rede, como entre os recursos da rede.

malha de dados

Uma estrutura arquitetônica que fornece propriedade de dados distribuída e descentralizada com gerenciamento e governança centralizados.

minimização de dados

O princípio de coletar e processar apenas os dados estritamente necessários. Praticar a minimização de dados no Nuvem AWS pode reduzir os riscos de privacidade, os custos e a pegada de carbono de sua análise.

perímetro de dados

Um conjunto de proteções preventivas em seu AWS ambiente que ajudam a garantir que somente identidades confiáveis acessem recursos confiáveis das redes esperadas. Para obter mais informações, consulte Construindo um perímetro de dados em. AWS

pré-processamento de dados

A transformação de dados brutos em um formato que seja facilmente analisado por seu modelo de ML. O pré-processamento de dados pode significar a remoção de determinadas colunas ou linhas e o tratamento de valores ausentes, inconsistentes ou duplicados.

proveniência dos dados

O processo de rastrear a origem e o histórico dos dados ao longo de seu ciclo de vida, por exemplo, como os dados foram gerados, transmitidos e armazenados.

titular dos dados

Um indivíduo cujos dados estão sendo coletados e processados.

data warehouse

Um sistema de gerenciamento de dados que oferece suporte à inteligência comercial, como análises. Os data warehouses geralmente contêm grandes quantidades de dados históricos e geralmente são usados para consultas e análises.

linguagem de definição de dados (DDL)

Instruções ou comandos para criar ou modificar a estrutura de tabelas e objetos em um banco de dados.

linguagem de manipulação de dados (DML)

Instruções ou comandos para modificar (inserir, atualizar e excluir) informações em um banco de dados.

DDL

Consulte a linguagem de definição de banco de dados.

deep ensemble

A combinação de vários modelos de aprendizado profundo para gerar previsões. Os deep ensembles podem ser usados para produzir uma previsão mais precisa ou para estimar a incerteza nas previsões.

Aprendizado profundo

Um subcampo do ML que usa várias camadas de redes neurais artificiais para identificar o mapeamento entre os dados de entrada e as variáveis-alvo de interesse.

defense-in-depth

Uma abordagem de segurança da informação na qual uma série de mecanismos e controles de segurança são cuidadosamente distribuídos por toda a rede de computadores para proteger a confidencialidade, a integridade e a disponibilidade da rede e dos dados nela contidos. Ao adotar essa estratégia AWS, você adiciona vários controles em diferentes camadas da AWS Organizations estrutura para ajudar a proteger os recursos. Por exemplo, uma defense-in-depth abordagem pode combinar autenticação multifatorial, segmentação de rede e criptografia.

administrador delegado

Em AWS Organizations, um serviço compatível pode registrar uma conta de AWS membro para administrar as contas da organização e gerenciar as permissões desse serviço. Essa conta

é chamada de administrador delegado para esse serviço Para obter mais informações e uma lista de serviços compatíveis, consulte Serviços que funcionam com o AWS Organizations na documentação do AWS Organizations .

implantação

O processo de criar uma aplicação, novos recursos ou correções de código disponíveis no ambiente de destino. A implantação envolve a implementação de mudanças em uma base de código e, em seguida, a criação e execução dessa base de código nos ambientes da aplicação

ambiente de desenvolvimento

Veja o ambiente.

controle detectivo

Um controle de segurança projetado para detectar, registrar e alertar após a ocorrência de um evento. Esses controles são uma segunda linha de defesa, alertando você sobre eventos de segurança que contornaram os controles preventivos em vigor. Para obter mais informações, consulte Controles detectivos em Como implementar controles de segurança na AWS.

mapeamento do fluxo de valor de desenvolvimento (DVSM)

Um processo usado para identificar e priorizar restrições que afetam negativamente a velocidade e a qualidade em um ciclo de vida de desenvolvimento de software. O DVSM estende o processo de mapeamento do fluxo de valor originalmente projetado para práticas de manufatura enxuta. Ele se concentra nas etapas e equipes necessárias para criar e movimentar valor por meio do processo de desenvolvimento de software.

gêmeo digital

Uma representação virtual de um sistema real, como um prédio, fábrica, equipamento industrial ou linha de produção. Os gêmeos digitais oferecem suporte à manutenção preditiva, ao monitoramento remoto e à otimização da produção.

tabela de dimensões

Em um esquema em estrela, uma tabela menor que contém atributos de dados sobre dados quantitativos em uma tabela de fatos. Os atributos da tabela de dimensões geralmente são campos de texto ou números discretos que se comportam como texto. Esses atributos são comumente usados para restringir consultas, filtrar e rotular conjuntos de resultados.

desastre

Um evento que impede que uma workload ou sistema cumpra seus objetivos de negócios em seu local principal de implantação. Esses eventos podem ser desastres naturais, falhas técnicas ou o resultado de ações humanas, como configuração incorreta não intencional ou ataque de malware.

Recuperação de desastres (RD)

A estratégia e o processo que você usa para minimizar o tempo de inatividade e a perda de dados causados por um desastre. Para obter mais informações, consulte Recuperação de desastres de cargas de trabalho em AWS: Recuperação na nuvem no AWS Well-Architected Framework.

DML

Consulte linguagem de manipulação de banco de dados.

design orientado por domínio

Uma abordagem ao desenvolvimento de um sistema de software complexo conectando seus componentes aos domínios em evolução, ou principais metas de negócios, atendidos por cada componente. Esse conceito foi introduzido por Eric Evans em seu livro, Design orientado por domínio: lidando com a complexidade no coração do software (Boston: Addison-Wesley Professional, 2003). Para obter informações sobre como usar o design orientado por domínio com o padrão strangler fig, consulte Modernizar incrementalmente os serviços web herdados do Microsoft ASP.NET (ASMX) usando contêineres e o Amazon API Gateway.

DR

Veja a recuperação de desastres.

detecção de deriva

Rastreando desvios de uma configuração básica. Por exemplo, você pode usar AWS CloudFormation para detectar desvios nos recursos do sistema ou AWS Control Tower para detectar mudanças em seu landing zone que possam afetar a conformidade com os requisitos de governança.

DVSM

Veja o mapeamento do fluxo de valor do desenvolvimento.

E

EDA

Veja a análise exploratória de dados.

computação de borda

A tecnologia que aumenta o poder computacional de dispositivos inteligentes nas bordas de uma rede de IoT. Quando comparada à computação em nuvem, a computação de ponta pode reduzir a latência da comunicação e melhorar o tempo de resposta.

Criptografia

Um processo de computação que transforma dados de texto simples, legíveis por humanos, em texto cifrado.

chave de criptografia

Uma sequência criptográfica de bits aleatórios que é gerada por um algoritmo de criptografia. As chaves podem variar em tamanho, e cada chave foi projetada para ser imprevisível e exclusiva.

endianismo

A ordem na qual os bytes são armazenados na memória do computador. Os sistemas big-endian armazenam o byte mais significativo antes. Os sistemas little-endian armazenam o byte menos significativo antes.

endpoint

Veja o endpoint do serviço.

serviço de endpoint

Um serviço que pode ser hospedado em uma nuvem privada virtual (VPC) para ser compartilhado com outros usuários. Você pode criar um serviço de endpoint com AWS PrivateLink e conceder permissões a outros diretores Contas da AWS ou a AWS Identity and Access Management (IAM). Essas contas ou entidades principais podem se conectar ao serviço de endpoint de maneira privada criando endpoints da VPC de interface. Para obter mais informações, consulte Criar um serviço de endpoint na documentação do Amazon Virtual Private Cloud (Amazon VPC).

planejamento de recursos corporativos (ERP)

Um sistema que automatiza e gerencia os principais processos de negócios (como contabilidade, MES e gerenciamento de projetos) para uma empresa.

criptografia envelopada

O processo de criptografar uma chave de criptografia com outra chave de criptografia. Para obter mais informações, consulte Criptografia de envelope na documentação AWS Key Management Service (AWS KMS).

environment (ambiente)

Uma instância de uma aplicação em execução. Estes são tipos comuns de ambientes na computação em nuvem:

- ambiente de desenvolvimento: uma instância de uma aplicação em execução que está disponível somente para a equipe principal responsável pela manutenção da aplicação. Ambientes de desenvolvimento são usados para testar mudanças antes de promovê-las para ambientes superiores. Esse tipo de ambiente às vezes é chamado de ambiente de teste.
- ambientes inferiores: todos os ambientes de desenvolvimento para uma aplicação, como aqueles usados para compilações e testes iniciais.
- ambiente de produção: uma instância de uma aplicação em execução que os usuários finais podem acessar. Em um pipeline de CI/CD, o ambiente de produção é o último ambiente de implantação.
- ambientes superiores: todos os ambientes que podem ser acessados por usuários que não sejam a equipe principal de desenvolvimento. Isso pode incluir um ambiente de produção, ambientes de pré-produção e ambientes para testes de aceitação do usuário.

epic

Em metodologias ágeis, categorias funcionais que ajudam a organizar e priorizar seu trabalho. Os epics fornecem uma descrição de alto nível dos requisitos e das tarefas de implementação. Por exemplo, os épicos de segurança AWS da CAF incluem gerenciamento de identidade e acesso, controles de detetive, segurança de infraestrutura, proteção de dados e resposta a incidentes. Para obter mais informações sobre epics na estratégia de migração da AWS , consulte o guia de implementação do programa.

ERP

Consulte planejamento de recursos corporativos.

análise exploratória de dados (EDA)

O processo de analisar um conjunto de dados para entender suas principais características. Você coleta ou agrega dados e, em seguida, realiza investigações iniciais para encontrar padrões,

detectar anomalias e verificar suposições. O EDA é realizado por meio do cálculo de estatísticas resumidas e da criação de visualizações de dados.

F

tabela de fatos

A tabela central em um esquema em estrela. Ele armazena dados quantitativos sobre operações comerciais. Normalmente, uma tabela de fatos contém dois tipos de colunas: aquelas que contêm medidas e aquelas que contêm uma chave externa para uma tabela de dimensões.

falham rapidamente

Uma filosofia que usa testes frequentes e incrementais para reduzir o ciclo de vida do desenvolvimento. É uma parte essencial de uma abordagem ágil.

limite de isolamento de falhas

No Nuvem AWS, um limite, como uma zona de disponibilidade, Região da AWS um plano de controle ou um plano de dados, que limita o efeito de uma falha e ajuda a melhorar a resiliência das cargas de trabalho. Para obter mais informações, consulte Limites de isolamento de AWS falhas.

ramificação de recursos

Veja a filial.

recursos

Os dados de entrada usados para fazer uma previsão. Por exemplo, em um contexto de manufatura, os recursos podem ser imagens capturadas periodicamente na linha de fabricação.

importância do recurso

O quanto um recurso é importante para as previsões de um modelo. Isso geralmente é expresso como uma pontuação numérica que pode ser calculada por meio de várias técnicas, como Shapley Additive Explanations (SHAP) e gradientes integrados. Para obter mais informações, consulte Interpretabilidade do modelo de aprendizado de máquina com:AWS.

transformação de recursos

O processo de otimizar dados para o processo de ML, incluindo enriquecer dados com fontes adicionais, escalar valores ou extrair vários conjuntos de informações de um único

campo de dados. Isso permite que o modelo de ML se beneficie dos dados. Por exemplo, se a data “2021-05-27 00:15:37” for dividida em “2021”, “maio”, “quinta” e “15”, isso poderá ajudar o algoritmo de aprendizado a aprender padrões diferenciados associados a diferentes componentes de dados.

FGAC

Veja o controle de acesso refinado.

Controle de acesso refinado (FGAC)

O uso de várias condições para permitir ou negar uma solicitação de acesso.

migração flash-cut

Um método de migração de banco de dados que usa replicação contínua de dados por meio da captura de dados alterados para migrar dados no menor tempo possível, em vez de usar uma abordagem em fases. O objetivo é reduzir ao mínimo o tempo de inatividade.

G

bloqueio geográfico

Veja as restrições geográficas.

restrições geográficas (bloqueio geográfico)

Na Amazon CloudFront, uma opção para impedir que usuários em países específicos acessem distribuições de conteúdo. É possível usar uma lista de permissões ou uma lista de bloqueios para especificar países aprovados e banidos. Para obter mais informações, consulte Restringir a distribuição geográfica do seu conteúdo na CloudFront documentação.

Fluxo de trabalho do GitFlow

Uma abordagem na qual ambientes inferiores e superiores usam ramificações diferentes em um repositório de código-fonte. O fluxo de trabalho do Gitflow é considerado legado, e o fluxo de trabalho baseado em troncos é a abordagem moderna e preferida.

estratégia greenfield

A ausência de infraestrutura existente em um novo ambiente. Ao adotar uma estratégia greenfield para uma arquitetura de sistema, é possível selecionar todas as novas tecnologias sem a

restrição da compatibilidade com a infraestrutura existente, também conhecida como brownfield. Se estiver expandindo a infraestrutura existente, poderá combinar as estratégias brownfield e greenfield.

barreira de proteção

Uma regra de alto nível que ajuda a gerenciar recursos, políticas e conformidade em todas as unidades organizacionais (UOs). Barreiras de proteção preventivas impõem políticas para garantir o alinhamento a padrões de conformidade. Elas são implementadas usando políticas de controle de serviço e limites de permissões do IAM. Barreiras de proteção detectivas detectam violações de políticas e problemas de conformidade e geram alertas para remediação. Eles são implementados usando AWS Config, AWS Security Hub, Amazon GuardDuty AWS Trusted Advisor, Amazon Inspector e verificações personalizadas AWS Lambda .

H

HA

Veja a alta disponibilidade.

migração heterogênea de bancos de dados

Migrar seu banco de dados de origem para um banco de dados de destino que usa um mecanismo de banco de dados diferente (por exemplo, Oracle para Amazon Aurora). A migração heterogênea geralmente faz parte de um esforço de redefinição da arquitetura, e converter o esquema pode ser uma tarefa complexa. O AWS fornece o AWS SCT para ajudar nas conversões de esquemas.

alta disponibilidade (HA)

A capacidade de uma workload operar continuamente, sem intervenção, em caso de desafios ou desastres. Os sistemas AH são projetados para realizar o failover automático, oferecer consistentemente desempenho de alta qualidade e lidar com diferentes cargas e falhas com impacto mínimo no desempenho.

modernização de historiador

Uma abordagem usada para modernizar e atualizar os sistemas de tecnologia operacional (OT) para melhor atender às necessidades do setor de manufatura. Um historiador é um tipo de banco de dados usado para coletar e armazenar dados de várias fontes em uma fábrica.

migração homogênea de bancos de dados

Migrar seu banco de dados de origem para um banco de dados de destino que compartilha o mesmo mecanismo de banco de dados (por exemplo, Microsoft SQL Server para Amazon RDS para SQL Server). A migração homogênea geralmente faz parte de um esforço de redefinição da hospedagem ou da plataforma. É possível usar utilitários de banco de dados nativos para migrar o esquema.

dados quentes

Dados acessados com frequência, como dados em tempo real ou dados translacionais recentes. Esses dados normalmente exigem uma camada ou classe de armazenamento de alto desempenho para fornecer respostas rápidas às consultas.

hotfix

Uma correção urgente para um problema crítico em um ambiente de produção. Devido à sua urgência, um hotfix geralmente é feito fora do fluxo de trabalho típico de uma DevOps versão.

período de hipercuidados

Imediatamente após a substituição, o período em que uma equipe de migração gerencia e monitora as aplicações migradas na nuvem para resolver quaisquer problemas. Normalmente, a duração desse período é de 1 a 4 dias. No final do período de hipercuidados, a equipe de migração normalmente transfere a responsabilidade pelas aplicações para a equipe de operações de nuvem.

I

IaC

Veja a infraestrutura como código.

Política baseada em identidade

Uma política anexada a um ou mais diretores do IAM que define suas permissões no Nuvem AWS ambiente.

aplicação ociosa

Uma aplicação que tem um uso médio de CPU e memória entre 5 e 20% em um período de 90 dias. Em um projeto de migração, é comum retirar essas aplicações ou retê-las on-premises.

IIoT

Veja a Internet das Coisas industrial.

infraestrutura imutável

Um modelo que implanta uma nova infraestrutura para cargas de trabalho de produção em vez de atualizar, corrigir ou modificar a infraestrutura existente. Infraestruturas imutáveis são inerentemente mais consistentes, confiáveis e previsíveis do que infraestruturas mutáveis. Para obter mais informações, consulte as melhores práticas de implantação usando infraestrutura imutável no Well-Architected AWS Framework.

VPC de entrada (admissão)

Em uma arquitetura de AWS várias contas, uma VPC que aceita, inspeciona e roteia conexões de rede de fora de um aplicativo. A Arquitetura de referência de segurança da AWS recomenda configurar sua conta de rede com VPCs de entrada, saída e inspeção para proteger a interface bidirecional entre a aplicação e a Internet em geral.

migração incremental

Uma estratégia de substituição na qual você migra a aplicação em pequenas partes, em vez de realizar uma única substituição completa. Por exemplo, é possível mover inicialmente apenas alguns microsserviços ou usuários para o novo sistema. Depois de verificar se tudo está funcionando corretamente, mova os microsserviços ou usuários adicionais de forma incremental até poder descomissionar seu sistema herdado. Essa estratégia reduz os riscos associados a migrações de grande porte.

Indústria 4.0

Um termo que foi introduzido por Klaus Schwab em 2016 para se referir à modernização dos processos de fabricação por meio de avanços em conectividade, dados em tempo real, automação, análise e IA/ML.

infraestrutura

Todos os recursos e ativos contidos no ambiente de uma aplicação.

Infraestrutura como código (IaC)

O processo de provisionamento e gerenciamento da infraestrutura de uma aplicação por meio de um conjunto de arquivos de configuração. A IaC foi projetada para ajudar você a centralizar

o gerenciamento da infraestrutura, padronizar recursos e escalar rapidamente para que novos ambientes sejam reproduzíveis, confiáveis e consistentes.

Internet das Coisas Industrial (IIoT)

O uso de sensores e dispositivos conectados à Internet nos setores industriais, como manufatura, energia, automotivo, saúde, ciências biológicas e agricultura. Para obter mais informações, consulte Construir uma estratégia de transformação digital para a Internet das Coisas Industrial (IIoT).

VPC de inspeção

Em uma arquitetura de AWS várias contas, uma VPC centralizada que gerencia as inspeções do tráfego de rede entre VPCs (na mesma ou em diferentes Regiões da AWS), a Internet e as redes locais. A Arquitetura de referência de segurança da AWS recomenda configurar sua conta de rede com VPCs de entrada, saída e inspeção para proteger a interface bidirecional entre a aplicação e a Internet em geral.

Internet das Coisas (IoT)

A rede de objetos físicos conectados com sensores ou processadores incorporados que se comunicam com outros dispositivos e sistemas pela Internet ou por uma rede de comunicação local. Para obter mais informações, consulte O que é IoT?

interpretabilidade

Uma característica de um modelo de machine learning que descreve o grau em que um ser humano pode entender como as previsões do modelo dependem de suas entradas. Para obter mais informações, consulte Interpretabilidade do modelo de machine learning com a AWS.

IoT

Consulte Internet das Coisas.

Biblioteca de informações de TI (ITIL)

Um conjunto de práticas recomendadas para fornecer serviços de TI e alinhar esses serviços a requisitos de negócios. A ITIL fornece a base para o ITSM.

Gerenciamento de serviços de TI (ITSM)

Atividades associadas a design, implementação, gerenciamento e suporte de serviços de TI para uma organização. Para obter informações sobre a integração de operações em nuvem com ferramentas de ITSM, consulte o guia de integração de operações.

ITIL

Consulte a biblioteca de informações de TI.

ITSM

Veja o gerenciamento de serviços de TI.

L

controle de acesso baseado em etiqueta (LBAC)

Uma implementação do controle de acesso obrigatório (MAC) em que os usuários e os dados em si recebem explicitamente um valor de etiqueta de segurança. A interseção entre a etiqueta de segurança do usuário e a etiqueta de segurança dos dados determina quais linhas e colunas podem ser vistas pelo usuário.

zona de pouso

Uma landing zone é um AWS ambiente bem arquitetado, com várias contas, escalável e seguro. Um ponto a partir do qual suas organizações podem iniciar e implantar rapidamente workloads e aplicações com confiança em seu ambiente de segurança e infraestrutura. Para obter mais informações sobre zonas de pouso, consulte Configurar um ambiente da AWS com várias contas seguro e escalável.

migração de grande porte

Uma migração de 300 servidores ou mais.

LBAC

Veja controle de acesso baseado em etiquetas.

privilégio mínimo

A prática recomendada de segurança de conceder as permissões mínimas necessárias para executar uma tarefa. Para obter mais informações, consulte Aplicar permissões de privilégios mínimos na documentação do IAM.

mover sem alterações (lift-and-shift)

Veja 7 Rs.

sistema little-endian

Um sistema que armazena o byte menos significativo antes. Veja também endianness.

ambientes inferiores

Veja o ambiente.

M

machine learning (ML)

Um tipo de inteligência artificial que usa algoritmos e técnicas para reconhecimento e aprendizado de padrões. O ML analisa e aprende com dados gravados, por exemplo, dados da Internet das Coisas (IoT), para gerar um modelo estatístico baseado em padrões. Para obter mais informações, consulte Machine learning.

ramificação principal

Veja a filial.

malware

Software projetado para comprometer a segurança ou a privacidade do computador. O malware pode interromper os sistemas do computador, vazar informações confidenciais ou obter acesso não autorizado. Exemplos de malware incluem vírus, worms, ransomware, cavalos de Tróia, spyware e keyloggers.

serviços gerenciados

Serviços da AWS para o qual AWS opera a camada de infraestrutura, o sistema operacional e as plataformas, e você acessa os endpoints para armazenar e recuperar dados. O Amazon Simple Storage Service (Amazon S3) e o Amazon DynamoDB são exemplos de serviços gerenciados. Eles também são conhecidos como serviços abstratos.

sistema de execução de manufatura (MES)

Um sistema de software para rastrear, monitorar, documentar e controlar processos de produção que convertem matérias-primas em produtos acabados no chão de fábrica.

MAP

Consulte Migration Acceleration Program.

mecanismo

Um processo completo no qual você cria uma ferramenta, impulsiona a adoção da ferramenta e, em seguida, inspeciona os resultados para fazer ajustes. Um mecanismo é um ciclo que se reforça e se aprimora à medida que opera. Para obter mais informações, consulte Construindo mecanismos no AWS Well-Architected Framework.

conta-membro

Todos, Contas da AWS exceto a conta de gerenciamento, que fazem parte de uma organização em AWS Organizations. Uma conta só pode ser membro de uma organização de cada vez.

MES

Veja o sistema de execução de manufatura.

Transporte de telemetria de enfileiramento de mensagens (MQTT)

Um protocolo de comunicação leve machine-to-machine (M2M), baseado no padrão de publicação/assinatura, para dispositivos de IoT com recursos limitados.

microsserviço

Um serviço pequeno e independente que se comunica por meio de APIs bem definidas e normalmente pertence a equipes pequenas e autônomas. Por exemplo, um sistema de seguradora pode incluir microsserviços que mapeiam as capacidades comerciais, como vendas ou marketing, ou subdomínios, como compras, reclamações ou análises. Os benefícios dos microsserviços incluem agilidade, escalabilidade flexível, fácil implantação, código reutilizável e resiliência. Para obter mais informações, consulte Integração de microsserviços usando serviços sem AWS servidor.

arquitetura de microsserviços

Uma abordagem à criação de aplicações com componentes independentes que executam cada processo de aplicação como um microsserviço. Esses microsserviços se comunicam por meio de uma interface bem definida usando APIs leves. Cada microsserviço nessa arquitetura pode ser atualizado, implantado e escalado para atender à demanda por funções específicas de uma aplicação. Para obter mais informações, consulte Implementação de microsserviços em. AWS

Programa de Aceleração da Migração (MAP)

Um AWS programa que fornece suporte de consultoria, treinamento e serviços para ajudar as organizações a criar uma base operacional sólida para migrar para a nuvem e ajudar a

compensar o custo inicial das migrações. O MAP inclui uma metodologia de migração para executar migrações legadas de forma metódica e um conjunto de ferramentas para automatizar e acelerar cenários comuns de migração.

migração em escala

O processo de mover a maior parte do portfólio de aplicações para a nuvem em ondas, com mais aplicações sendo movidas em um ritmo mais rápido a cada onda. Essa fase usa as práticas recomendadas e lições aprendidas nas fases anteriores para implementar uma fábrica de migração de equipes, ferramentas e processos para agilizar a migração de workloads por meio de automação e entrega ágeis. Esta é a terceira fase da estratégia de migração para a AWS.

fábrica de migração

Equipes multifuncionais que simplificam a migração de workloads por meio de abordagens automatizadas e ágeis. As equipes da fábrica de migração geralmente incluem operações, analistas e proprietários de negócios, engenheiros de migração, desenvolvedores e DevOps profissionais que trabalham em sprints. Entre 20 e 50% de um portfólio de aplicações corporativas consiste em padrões repetidos que podem ser otimizados por meio de uma abordagem de fábrica. Para obter mais informações, consulte discussão sobre fábricas de migração e o guia do Cloud Migration Factory neste conjunto de conteúdo.

metadados de migração

As informações sobre a aplicação e o servidor necessárias para concluir a migração. Cada padrão de migração exige um conjunto de metadados de migração diferente. Exemplos de metadados de migração incluem a sub-rede, o grupo de segurança e AWS a conta de destino.

padrão de migração

Uma tarefa de migração repetível que detalha a estratégia de migração, o destino da migração e a aplicação ou o serviço de migração usado. Exemplo: rehospede a migração para o Amazon EC2 AWS com o Application Migration Service.

Avaliação de Portfólio para Migração (MPA)

Uma ferramenta on-line que fornece informações para validar o caso de negócios para migrar para o. Nuvem AWS O MPA fornece avaliação detalhada do portfólio (dimensionamento correto do servidor, preços, comparações de TCO, análise de custos de migração), bem como planejamento de migração (análise e coleta de dados de aplicações, agrupamento de aplicações, priorização de migração e planejamento de ondas). A ferramenta MPA (requer login) está disponível gratuitamente para todos os AWS consultores e consultores parceiros da APN.

Avaliação de Preparação para Migração (MRA)

O processo de obter insights sobre o status de prontidão de uma organização para a nuvem, identificar pontos fortes e fracos e criar um plano de ação para fechar as lacunas identificadas, usando o CAF. AWS Para mais informações, consulte o guia de preparação para migração. A MRA é a primeira fase da estratégia de migração para a AWS.

estratégia de migração

A abordagem usada para migrar uma carga de trabalho para o. Nuvem AWS Para obter mais informações, consulte a entrada de 7 Rs neste glossário e consulte Mobilize sua organização para acelerar migrações em grande escala.

ML

Veja o aprendizado de máquina.

modernização

Transformar uma aplicação desatualizada (herdada ou monolítica) e sua infraestrutura em um sistema ágil, elástico e altamente disponível na nuvem para reduzir custos, ganhar eficiência e aproveitar as inovações. Para obter mais informações, consulte Estratégia para modernizar aplicativos no Nuvem AWS.

avaliação de preparação para modernização

Uma avaliação que ajuda a determinar a preparação para modernização das aplicações de uma organização. Ela identifica benefícios, riscos e dependências e determina o quão bem a organização pode acomodar o estado futuro dessas aplicações. O resultado da avaliação é um esquema da arquitetura de destino, um roteiro que detalha as fases de desenvolvimento e os marcos do processo de modernização e um plano de ação para abordar as lacunas identificadas. Para obter mais informações, consulte Avaliação da prontidão para modernização de aplicativos no. Nuvem AWS

aplicações monolíticas (monólitos)

Aplicações que são executadas como um único serviço com processos fortemente acoplados. As aplicações monolíticas apresentam várias desvantagens. Se um recurso da aplicação apresentar um aumento na demanda, toda a arquitetura deverá ser escalada. Adicionar ou melhorar os recursos de uma aplicação monolítica também se torna mais complexo quando a base de código cresce. Para resolver esses problemas, é possível criar uma arquitetura de microsserviços. Para obter mais informações, consulte Decompor monólitos em microsserviços.

MAPA

Consulte Avaliação do portfólio de migração.

MQTT

Consulte Transporte de telemetria de enfileiramento de mensagens.

classificação multiclasse

Um processo que ajuda a gerar previsões para várias classes (prevendo um ou mais de dois resultados). Por exemplo, um modelo de ML pode perguntar “Este produto é um livro, um carro ou um telefone?” ou “Qual categoria de produtos é mais interessante para este cliente?”

infraestrutura mutável

Um modelo que atualiza e modifica a infraestrutura existente para cargas de trabalho de produção. Para melhorar a consistência, confiabilidade e previsibilidade, o AWS Well-Architected Framework recomenda o uso de infraestrutura imutável como uma prática recomendada.

O

OAC

Veja o controle de acesso de origem.

CARVALHO

Veja a identidade de acesso de origem.

OCM

Veja o gerenciamento de mudanças organizacionais.

migração offline

Um método de migração no qual a workload de origem é desativada durante o processo de migração. Esse método envolve tempo de inatividade prolongado e geralmente é usado para workloads pequenas e não críticas.

OI

Veja a integração de operações.

OLA

Veja o contrato em nível operacional.

migração online

Um método de migração no qual a workload de origem é copiada para o sistema de destino sem ser colocada offline. As aplicações conectadas à workload podem continuar funcionando durante a migração. Esse método envolve um tempo de inatividade nulo ou mínimo e normalmente é usado para workloads essenciais para a produção.

OPC-UA

Consulte Comunicação de processo aberto — Arquitetura unificada.

Comunicação de processo aberto — Arquitetura unificada (OPC-UA)

Um protocolo de comunicação machine-to-machine (M2M) para automação industrial. O OPC-UA fornece um padrão de interoperabilidade com esquemas de criptografia, autenticação e autorização de dados.

acordo de nível operacional (OLA)

Um acordo que esclarece o que os grupos funcionais de TI prometem oferecer uns aos outros para apoiar um acordo de serviço (SLA).

análise de prontidão operacional (ORR)

Uma lista de verificação de perguntas e melhores práticas associadas que ajudam você a entender, avaliar, prevenir ou reduzir o escopo de incidentes e possíveis falhas. Para obter mais informações, consulte Operational Readiness Reviews (ORR) no Well-Architected AWS Framework.

tecnologia operacional (OT)

Sistemas de hardware e software que funcionam com o ambiente físico para controlar operações, equipamentos e infraestrutura industriais. Na manufatura, a integração dos sistemas OT e de tecnologia da informação (TI) é o foco principal das transformações da Indústria 4.0.

integração de operações (OI)

O processo de modernização das operações na nuvem, que envolve planejamento de preparação, automação e integração. Para obter mais informações, consulte o guia de integração de operações.

trilha organizacional

Uma trilha criada por ela AWS CloudTrail registra todos os eventos de todos Contas da AWS em uma organização em AWS Organizations. Essa trilha é criada em cada Conta da AWS que faz parte da organização e monitora a atividade em cada conta. Para obter mais informações, consulte Criação de uma trilha para uma organização na CloudTrail documentação.

gerenciamento de alterações organizacionais (OCM)

Uma estrutura para gerenciar grandes transformações de negócios disruptivas de uma perspectiva de pessoas, cultura e liderança. O OCM ajuda as organizações a se prepararem e fazerem a transição para novos sistemas e estratégias, acelerando a adoção de alterações, abordando questões de transição e promovendo mudanças culturais e organizacionais. Na estratégia de AWS migração, essa estrutura é chamada de aceleração de pessoas, devido à velocidade de mudança exigida nos projetos de adoção da nuvem. Para obter mais informações, consulte o guia do OCM.

controle de acesso de origem (OAC)

Em CloudFront, uma opção aprimorada para restringir o acesso para proteger seu conteúdo do Amazon Simple Storage Service (Amazon S3). O OAC oferece suporte a todos os buckets S3 Regiões da AWS, criptografia do lado do servidor com AWS KMS (SSE-KMS) e solicitações dinâmicas ao bucket S3. `PUT` `DELETE`

Identidade do acesso de origem (OAI)

Em CloudFront, uma opção para restringir o acesso para proteger seu conteúdo do Amazon S3. Quando você usa o OAI, CloudFront cria um principal com o qual o Amazon S3 pode se autenticar. Os diretores autenticados podem acessar o conteúdo em um bucket do S3 somente por meio de uma distribuição específica. CloudFront Veja também OAC, que fornece um controle de acesso mais granular e aprimorado.

OU

Veja a análise de prontidão operacional.

NÃO

Veja a tecnologia operacional.

VPC de saída (egresso)

Em uma arquitetura de AWS várias contas, uma VPC que gerencia conexões de rede que são iniciadas de dentro de um aplicativo. A Arquitetura de referência de segurança da AWS

recomenda configurar sua conta de rede com VPCs de entrada, saída e inspeção para proteger a interface bidirecional entre a aplicação e a Internet em geral.

P

limite de permissões

Uma política de gerenciamento do IAM anexada a entidades principais do IAM para definir as permissões máximas que o usuário ou perfil podem ter. Para obter mais informações, consulte Limites de permissões na documentação do IAM.

Informações de identificação pessoal (PII)

Informações que, quando visualizadas diretamente ou combinadas com outros dados relacionados, podem ser usadas para inferir razoavelmente a identidade de um indivíduo. Exemplos de PII incluem nomes, endereços e informações de contato.

PII

Veja informações de identificação pessoal.

manual

Um conjunto de etapas predefinidas que capturam o trabalho associado às migrações, como a entrega das principais funções operacionais na nuvem. Um manual pode assumir a forma de scripts, runbooks automatizados ou um resumo dos processos ou etapas necessários para operar seu ambiente modernizado.

PLC

Consulte controlador lógico programável.

AMEIXA

Veja o gerenciamento do ciclo de vida do produto.

política

Um objeto que pode definir permissões (consulte a política baseada em identidade), especificar as condições de acesso (consulte a política baseada em recursos) ou definir as permissões máximas para todas as contas em uma organização em AWS Organizations (consulte a política de controle de serviços).

persistência poliglota

Escolher de forma independente a tecnologia de armazenamento de dados de um microsserviço com base em padrões de acesso a dados e outros requisitos. Se seus microsserviços tiverem a mesma tecnologia de armazenamento de dados, eles poderão enfrentar desafios de implementação ou apresentar baixa performance. Os microsserviços serão implementados com mais facilidade e alcançarão performance e escalabilidade melhores se usarem o armazenamento de dados mais bem adaptado às suas necessidades. Para obter mais informações, consulte Habilitar a persistência de dados em microsserviços.

avaliação do portfólio

Um processo de descobrir, analisar e priorizar o portfólio de aplicações para planejar a migração. Para obter mais informações, consulte Avaliar a preparação para a migração.

predicado

Uma condição de consulta que retorna `true` ou`false`, normalmente localizada em uma `WHERE` cláusula.

pressão de predicados

Uma técnica de otimização de consulta de banco de dados que filtra os dados na consulta antes da transferência. Isso reduz a quantidade de dados que devem ser recuperados e processados do banco de dados relacional e melhora o desempenho das consultas.

controle preventivo

Um controle de segurança projetado para evitar que um evento ocorra. Esses controles são a primeira linha de defesa para ajudar a evitar acesso não autorizado ou alterações indesejadas em sua rede. Para obter mais informações, consulte Controles preventivos em Como implementar controles de segurança na AWS.

principal (entidade principal)

Uma entidade AWS que pode realizar ações e acessar recursos. Essa entidade geralmente é um usuário raiz para um Conta da AWS, uma função do IAM ou um usuário. Para obter mais informações, consulte Entidade principal em Termos e conceitos de perfis na documentação do IAM.

Privacidade por design

Uma abordagem em engenharia de sistemas que leva em consideração a privacidade em todo o processo de engenharia.

zonas hospedadas privadas

Um contêiner que armazena informações sobre como você quer que o Amazon Route 53 responda a consultas ao DNS para um domínio e seus subdomínios dentro de uma ou mais VPCs. Para obter mais informações, consulte Como trabalhar com zonas hospedadas privadas na documentação do Route 53.

controle proativo

Um controle de segurança projetado para impedir a implantação de recursos não compatíveis. Esses controles examinam os recursos antes de serem provisionados. Se o recurso não estiver em conformidade com o controle, ele não será provisionado. Para obter mais informações, consulte o guia de referência de controles na AWS Control Tower documentação e consulte Controles proativos em Implementação de controles de segurança em AWS.

gerenciamento do ciclo de vida do produto (PLM)

O gerenciamento de dados e processos de um produto em todo o seu ciclo de vida, desde o design, desenvolvimento e lançamento, passando pelo crescimento e maturidade, até o declínio e a remoção.

ambiente de produção

Veja o ambiente.

controlador lógico programável (PLC)

Na fabricação, um computador altamente confiável e adaptável que monitora as máquinas e automatiza os processos de fabricação.

pseudonimização

O processo de substituir identificadores pessoais em um conjunto de dados por valores de espaço reservado. A pseudonimização pode ajudar a proteger a privacidade pessoal. Os dados pseudonimizados ainda são considerados dados pessoais.

publicar/assinar (pub/sub)

Um padrão que permite comunicações assíncronas entre microsserviços para melhorar a escalabilidade e a capacidade de resposta. Por exemplo, em um MES baseado em microsserviços, um microsserviço pode publicar mensagens de eventos em um canal no qual outros microsserviços possam se inscrever. O sistema pode adicionar novos microsserviços sem alterar o serviço de publicação.

Q

plano de consulta

Uma série de etapas, como instruções, usadas para acessar os dados em um sistema de banco de dados relacional SQL.

regressão de planos de consultas

Quando um otimizador de serviço de banco de dados escolhe um plano menos adequado do que escolhia antes de uma determinada alteração no ambiente de banco de dados ocorrer. Isso pode ser causado por alterações em estatísticas, restrições, configurações do ambiente, associações de parâmetros de consulta e atualizações do mecanismo de banco de dados.

R

Matriz RACI

Veja responsável, responsável, consultado, informado (RACI).

ransomware

Um software mal-intencionado desenvolvido para bloquear o acesso a um sistema ou dados de computador até que um pagamento seja feito.

Matriz RASCI

Veja responsável, responsável, consultado, informado (RACI).

RCAC

Veja o controle de acesso por linha e coluna.

réplica de leitura

Uma cópia de um banco de dados usada somente para leitura. É possível encaminhar consultas para a réplica de leitura e reduzir a carga no banco de dados principal.

rearquiteta

Veja 7 Rs.

objetivo de ponto de recuperação (RPO).

O máximo período de tempo aceitável desde o último ponto de recuperação de dados. Isso determina o que é considerado uma perda aceitável de dados entre o último ponto de recuperação e a interrupção do serviço.

objetivo de tempo de recuperação (RTO)

O máximo atraso aceitável entre a interrupção e a restauração do serviço.

refatorar

Veja 7 Rs.

Região

Uma coleção de AWS recursos em uma área geográfica. Cada um Região da AWS é isolado e independente dos outros para fornecer tolerância a falhas, estabilidade e resiliência. Para obter mais informações, consulte Especificar o que Regiões da AWS sua conta pode usar.

regressão

Uma técnica de ML que prevê um valor numérico. Por exemplo, para resolver o problema de “Por qual preço esta casa será vendida?” um modelo de ML pode usar um modelo de regressão linear para prever o preço de venda de uma casa com base em fatos conhecidos sobre a casa (por exemplo, a metragem quadrada).

redefinir a hospedagem

Veja 7 Rs.

versão

Em um processo de implantação, o ato de promover mudanças em um ambiente de produção.

realocar

Veja 7 Rs.

redefinir a plataforma

Veja 7 Rs.

recomprar

Veja 7 Rs.

resiliência

A capacidade de um aplicativo de resistir ou se recuperar de interrupções. Alta disponibilidade e recuperação de desastres são considerações comuns ao planejar a resiliência no. Nuvem AWS Para obter mais informações, consulte Nuvem AWS Resiliência.

política baseada em recurso

Uma política associada a um recurso, como um bucket do Amazon S3, um endpoint ou uma chave de criptografia. Esse tipo de política especifica quais entidades principais têm acesso permitido, ações válidas e quaisquer outras condições que devem ser atendidas.

matriz responsável, accountable, consultada, informada (RACI)

Uma matriz que define as funções e responsabilidades de todas as partes envolvidas nas atividades de migração e nas operações de nuvem. O nome da matriz é derivado dos tipos de responsabilidade definidos na matriz: responsável (R), responsabilizável (A), consultado (C) e informado (I). O tipo de suporte (S) é opcional. Se você incluir suporte, a matriz será chamada de matriz RASCI e, se excluir, será chamada de matriz RACI.

controle responsivo

Um controle de segurança desenvolvido para conduzir a remediação de eventos adversos ou desvios em relação à linha de base de segurança. Para obter mais informações, consulte Controles responsivos em Como implementar controles de segurança na AWS.

reter

Veja 7 Rs.

aposentar-se

Veja 7 Rs.

rotação

O processo de atualizar periodicamente um segredo para dificultar o acesso das credenciais por um invasor.

controle de acesso por linha e coluna (RCAC)

O uso de expressões SQL básicas e flexíveis que tenham regras de acesso definidas. O RCAC consiste em permissões de linha e máscaras de coluna.

RPO

Veja o objetivo do ponto de recuperação.

RTO

Veja o objetivo do tempo de recuperação.

runbook

Um conjunto de procedimentos manuais ou automatizados necessários para realizar uma tarefa específica. Eles são normalmente criados para agilizar operações ou procedimentos repetitivos com altas taxas de erro.

S

SAML 2.0

Um padrão aberto que muitos provedores de identidade (IdPs) usam. Esse recurso permite o login único federado (SSO), para que os usuários possam fazer login AWS Management Console ou chamar as operações da AWS API sem que você precise criar um usuário no IAM para todos em sua organização. Para obter mais informações sobre a federação baseada em SAML 2.0, consulte Sobre a federação baseada em SAML 2.0 na documentação do IAM.

SCADA

Veja controle de supervisão e aquisição de dados.

SCP

Veja a política de controle de serviços.

secret

Em AWS Secrets Manager, informações confidenciais ou restritas, como uma senha ou credenciais de usuário, que você armazena de forma criptografada. Ele consiste no valor secreto e em seus metadados. O valor secreto pode ser binário, uma única string ou várias strings. Para obter mais informações, consulte O que há em um segredo do Secrets Manager? na documentação do Secrets Manager.

controle de segurança

Uma barreira de proteção técnica ou administrativa que impede, detecta ou reduz a capacidade de uma ameaça explorar uma vulnerabilidade de segurança. Existem quatro tipos principais de controles de segurança: preventivos, detectivos, responsivos e proativos.

fortalecimento da segurança

O processo de reduzir a superfície de ataque para torná-la mais resistente a ataques. Isso pode incluir ações como remover recursos que não são mais necessários, implementar a prática recomendada de segurança de conceder privilégios mínimos ou desativar recursos desnecessários em arquivos de configuração.

sistema de gerenciamento de eventos e informações de segurança (SIEM)

Ferramentas e serviços que combinam sistemas de gerenciamento de informações de segurança (SIM) e gerenciamento de eventos de segurança (SEM). Um sistema SIEM coleta, monitora e analisa dados de servidores, redes, dispositivos e outras fontes para detectar ameaças e violações de segurança e gerar alertas.

automação de resposta de segurança

Uma ação predefinida e programada projetada para responder ou remediar automaticamente um evento de segurança. Essas automações servem como controles de segurança responsivos ou detectivos que ajudam você a implementar as melhores práticas AWS de segurança. Exemplos de ações de resposta automatizada incluem a modificação de um grupo de segurança da VPC, a correção de uma instância do Amazon EC2 ou a rotação de credenciais.

Criptografia do lado do servidor

Criptografia dos dados em seu destino, por AWS service (Serviço da AWS) quem os recebe.

política de controle de serviços (SCP)

Uma política que fornece controle centralizado sobre as permissões de todas as contas em uma organização no AWS Organizations. As SCPs definem barreiras de proteção ou estabelecem limites para as ações que um administrador pode delegar a usuários ou perfis. É possível usar SCPs como listas de permissão ou de negação para especificar quais serviços ou ações são permitidos ou proibidos. Para obter mais informações, consulte Políticas de controle de serviço na AWS Organizations documentação.

service endpoint (endpoint de serviço)

O URL do ponto de entrada para um AWS service (Serviço da AWS). Você pode usar o endpoint para se conectar programaticamente ao serviço de destino. Para obter mais informações, consulte Endpoints do AWS service (Serviço da AWS) na Referência geral da AWS.

acordo de serviço (SLA)

Um acordo que esclarece o que uma equipe de TI promete fornecer aos clientes, como tempo de atividade e performance do serviço.

indicador de nível de serviço (SLI)

Uma medida de um aspecto de desempenho de um serviço, como taxa de erro, disponibilidade ou taxa de transferência.

objetivo de nível de serviço (SLO)

Uma métrica alvo que representa a integridade de um serviço, conforme medida por um indicador de nível de serviço.

modelo de responsabilidade compartilhada

Um modelo que descreve a responsabilidade com a qual você compartilha AWS pela segurança e conformidade na nuvem. AWS é responsável pela segurança da nuvem, enquanto você é responsável pela segurança na nuvem. Para obter mais informações, consulte o Modelo de responsabilidade compartilhada.

SIEM

Veja informações de segurança e sistema de gerenciamento de eventos.

ponto único de falha (SPOF)

Uma falha em um único componente crítico de um aplicativo que pode interromper o sistema.

SLA

Veja o contrato de nível de serviço.

ESGUIO

Veja o indicador de nível de serviço.

SLO

Veja o objetivo do nível de serviço.

split-and-seed modelo

Um padrão para escalar e acelerar projetos de modernização. À medida que novos recursos e lançamentos de produtos são definidos, a equipe principal se divide para criar novas equipes

de produtos. Isso ajuda a escalar os recursos e os serviços da sua organização, melhora a produtividade do desenvolvedor e possibilita inovações rápidas. Para obter mais informações, consulte Abordagem em fases para modernizar aplicativos no. Nuvem AWS

CUSPE

Veja um único ponto de falha.

esquema de estrelas

Uma estrutura organizacional de banco de dados que usa uma grande tabela de fatos para armazenar dados transacionais ou medidos e usa uma ou mais tabelas dimensionais menores para armazenar atributos de dados. Essa estrutura foi projetada para uso em um data warehouse ou para fins de inteligência comercial.

padrão strangler fig

Uma abordagem à modernização de sistemas monolíticos que consiste em reescrever e substituir incrementalmente a funcionalidade do sistema até que o sistema herdado possa ser desativado. Esse padrão usa a analogia de uma videira que cresce e se torna uma árvore estabelecida e, eventualmente, supera e substitui sua hospedeira. O padrão foi apresentado por Martin Fowler como forma de gerenciar riscos ao reescrever sistemas monolíticos. Para ver um exemplo de como aplicar esse padrão, consulte Modernizar incrementalmente os serviços Web herdados do Microsoft ASP.NET (ASMX) usando contêineres e o Amazon API Gateway.

sub-rede

Um intervalo de endereços IP na VPC. Cada sub-rede fica alocada em uma única zona de disponibilidade.

controle de supervisão e aquisição de dados (SCADA)

Na manufatura, um sistema que usa hardware e software para monitorar ativos físicos e operações de produção.

symmetric encryption (criptografia simétrica)

Um algoritmo de criptografia que usa a mesma chave para criptografar e descriptografar dados.

testes sintéticos

Testar um sistema de forma que simule as interações do usuário para detectar possíveis problemas ou monitorar o desempenho. Você pode usar o Amazon CloudWatch Synthetics para criar esses testes.

T

tags

Pares de valores-chave que atuam como metadados para organizar seus recursos. AWS As tags podem ajudar você a gerenciar, identificar, organizar, pesquisar e filtrar recursos. Para obter mais informações, consulte Marcar seus recursos do AWS.

variável-alvo

O valor que você está tentando prever no ML supervisionado. Ela também é conhecida como variável de resultado. Por exemplo, em uma configuração de fabricação, a variável-alvo pode ser um defeito do produto.

lista de tarefas

Uma ferramenta usada para monitorar o progresso por meio de um runbook. Uma lista de tarefas contém uma visão geral do runbook e uma lista de tarefas gerais a serem concluídas. Para cada tarefa geral, ela inclui o tempo estimado necessário, o proprietário e o progresso.

ambiente de teste

Veja o ambiente.

treinamento

O processo de fornecer dados para que seu modelo de ML aprenda. Os dados de treinamento devem conter a resposta correta. O algoritmo de aprendizado descobre padrões nos dados de treinamento que mapeiam os atributos dos dados de entrada no destino (a resposta que você deseja prever). Ele gera um modelo de ML que captura esses padrões. Você pode usar o modelo de ML para obter previsões de novos dados cujo destino você não conhece.

gateway de trânsito

Um hub de trânsito de rede que pode ser usado para interconectar as VPCs e as redes on-premises. Para obter mais informações, consulte O que é um gateway de trânsito na AWS Transit Gateway documentação.

fluxo de trabalho baseado em troncos

Uma abordagem na qual os desenvolvedores criam e testam recursos localmente em uma ramificação de recursos e, em seguida, mesclam essas alterações na ramificação principal. A

ramificação principal é então criada para os ambientes de desenvolvimento, pré-produção e produção, sequencialmente.

Acesso confiável

Conceder permissões a um serviço que você especifica para realizar tarefas em sua organização AWS Organizations e em suas contas em seu nome. O serviço confiável cria um perfil vinculado ao serviço em cada conta, quando esse perfil é necessário, para realizar tarefas de gerenciamento para você. Para obter mais informações, consulte Usando AWS Organizations com outros AWS serviços na AWS Organizations documentação.

tuning (ajustar)

Alterar aspectos do processo de treinamento para melhorar a precisão do modelo de ML. Por exemplo, você pode treinar o modelo de ML gerando um conjunto de rótulos, adicionando rótulos e repetindo essas etapas várias vezes em configurações diferentes para otimizar o modelo.

equipe de duas pizzas

Uma pequena DevOps equipe que você pode alimentar com duas pizzas. Uma equipe de duas pizzas garante a melhor oportunidade possível de colaboração no desenvolvimento de software.

U

incerteza

Um conceito que se refere a informações imprecisas, incompletas ou desconhecidas que podem minar a confiabilidade dos modelos preditivos de ML. Há dois tipos de incertezas: a incerteza epistêmica é causada por dados limitados e incompletos, enquanto a incerteza aleatória é causada pelo ruído e pela aleatoriedade inerentes aos dados. Para obter mais informações, consulte o guia Como quantificar a incerteza em sistemas de aprendizado profundo.

tarefas indiferenciadas

Também conhecido como trabalho pesado, trabalho necessário para criar e operar um aplicativo, mas que não fornece valor direto ao usuário final nem oferece vantagem competitiva. Exemplos de tarefas indiferenciadas incluem aquisição, manutenção e planejamento de capacidade.

ambientes superiores

Veja o ambiente.

V

aspiração

Uma operação de manutenção de banco de dados que envolve limpeza após atualizações incrementais para recuperar armazenamento e melhorar a performance.

controle de versões

Processos e ferramentas que rastreiam mudanças, como alterações no código-fonte em um repositório.

emparelhamento de VPC

Uma conexão entre duas VPCs que permite rotear tráfego usando endereços IP privados. Para ter mais informações, consulte O que é emparelhamento de VPC? na documentação da Amazon VPC.

Vulnerabilidade

Uma falha de software ou hardware que compromete a segurança do sistema.

W

cache quente

Um cache de buffer que contém dados atuais e relevantes que são acessados com frequência. A instância do banco de dados pode ler do cache do buffer, o que é mais rápido do que ler da memória principal ou do disco.

dados mornos

Dados acessados raramente. Ao consultar esse tipo de dados, consultas moderadamente lentas geralmente são aceitáveis.

função de janela

Uma função SQL que executa um cálculo em um grupo de linhas que se relacionam de alguma forma com o registro atual. As funções de janela são úteis para processar tarefas, como calcular uma média móvel ou acessar o valor das linhas com base na posição relativa da linha atual.

workload

Uma coleção de códigos e recursos que geram valor empresarial, como uma aplicação voltada para o cliente ou um processo de back-end.

workstreams

Grupos funcionais em um projeto de migração que são responsáveis por um conjunto específico de tarefas. Cada workstream é independente, mas oferece suporte aos outros workstreams do projeto. Por exemplo, o workstream de portfólio é responsável por priorizar aplicações, planejar ondas e coletar metadados de migração. O workstream de portfólio entrega esses ativos ao workstream de migração, que então migra os servidores e as aplicações.

MINHOCA

Veja escrever uma vez, ler muitas.

WQF

Consulte o AWS Workload Qualification Framework.

escreva uma vez, leia muitas (WORM)

Um modelo de armazenamento que grava dados uma única vez e evita que os dados sejam excluídos ou modificados. Os usuários autorizados podem ler os dados quantas vezes forem necessárias, mas não podem alterá-los. Essa infraestrutura de armazenamento de dados é considerada imutável.

Z

exploração de dia zero

Um ataque, geralmente malware, que tira proveito de uma vulnerabilidade de dia zero.

vulnerabilidade de dia zero

Uma falha ou vulnerabilidade não mitigada em um sistema de produção. Os agentes de ameaças podem usar esse tipo de vulnerabilidade para atacar o sistema. Os desenvolvedores frequentemente ficam cientes da vulnerabilidade como resultado do ataque.

aplicação zumbi

Uma aplicação que tem um uso médio de CPU e memória inferior a 5%. Em um projeto de migração, é comum retirar essas aplicações.

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.