Ottimizzazione dei parametri Postgre SQL in Amazon e Amazon RDS Aurora

AWS Guida prescrittiva

AWS Guida prescrittiva: Ottimizzazione dei parametri Postgre SQL in Amazon e Amazon RDS Aurora

Table of Contents

Introduzione 1
Utilizzo dei gruppi di parametri del cluster DB e DB 2
Ottimizzazione dei parametri della memoria 4
shared_buffers 5
temp_buffers 6
effective_cache_size 8
work_mem 9
maintenance_work_mem 10
random_page_cost 12
seq_page_cost 14
track_activity_query_size 16
idle_in_transaction_session_timeout 17
statement_timeout 18
search_path 19
max_connections 21
Regolazione dei parametri dell'autovacuum 23
Comandi VACUUM e ANALYZE 24
Controllo del rigonfiamento 25
autovacuum 26
autovacuum_work_mem 27
autovacuum_naptime 28
autovacuum_max_workers 29
autovacuum_vacuum_scale_factor 30
autovacuum_vacuum_threshold 31
autovacuum_analyze_scale_factor 33
autovacuum_analyze_threshold 34
autovacuum_vacuum_cost_limit 35
Ottimizzazione dei parametri di registrazione 37
rds.force_autovacuum_logging 38
rds.force_admin_logging_level 39
log_duration 40
log_min_duration_statement 41
log_error_verbosity 42
log_statement 43

log_statement_stats 45
log_min_error_statement 46
log_min_messages 47
log_temp_files 48
log_connections 49
log_disconnections 50
Utilizzo dei parametri di registrazione per acquisire le variabili di associazione 51
Ottimizzazione dei parametri di replica 53
Esempio 54
Best practice 55
Passaggi successivi 56
Risorse 57
Cronologia dei documenti 58
Glossario 59
..... 59
A 60
B 63
C 65
D 68
E 72
F 74
G 75
H 76
I 77
L 80
M 81
O 85
P 88
Q 91
R 91
S 94
T 97
U 99
V 99
W 100
Z 101

..... cii

Ottimizzazione dei parametri PostgreSQL in Amazon RDS e Amazon Aurora

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

Febbraio 2024 (storia del documento)

Amazon Aurora PostgreSQL Compatible Edition e Amazon Relational Database Service (Amazon RDS) per PostgreSQL sono sofisticati servizi di database relazionali open source che offrono una gamma completa di funzionalità. È possibile utilizzare questi servizi per configurare il database PostgreSQL su una varietà di piattaforme e applicazioni.

Aurora e Amazon RDS offrono un modo semplificato per gestire e utilizzare i database PostgreSQL. Sono progettati per gestire l'infrastruttura del database e fornire disponibilità, durabilità e scalabilità elevate mentre ti concentri sullo sviluppo di applicazioni. Tuttavia, le configurazioni predefinite di questi servizi potrebbero non essere ottimali per tutti i carichi di lavoro. Per impostazione predefinita, questi servizi sono configurati per funzionare ovunque con il minor numero di risorse possibile e senza introdurre vulnerabilità. L'ottimizzazione dei parametri può aiutarti a ottenere prestazioni migliori, ridurre i tempi di inattività e migliorare l'efficienza complessiva del database. Ottimizzando i parametri per il tuo carico di lavoro specifico, puoi sfruttare appieno le funzionalità offerte da Amazon RDS e Aurora e massimizzarne i vantaggi.

Ad esempio, puoi migliorare le prestazioni ottimizzando Aurora e Amazon RDS for PostgreSQL e configurandone i parametri. È inoltre necessario prendere in considerazione le prestazioni quando si creano interrogazioni al database. Anche quando si ottimizzano le impostazioni del database, il sistema può funzionare male se le query eseguono scansioni complete della tabella, quando utilizzano un indice o se eseguono costose operazioni di unione o aggregazione.

Questa guida è destinata agli sviluppatori di database, agli ingegneri di database e agli amministratori che desiderano ottimizzare i parametri di memoria, autovacuum, registrazione e replica logica per i propri database PostgreSQL. La guida descrive anche i parametri specifici di Amazon RDS for PostgreSQL e Aurora PostgreSQL compatibili. L'ottimizzazione di questi parametri può aiutarti a ottimizzare le prestazioni del database e ridurre l'utilizzo delle risorse per il tuo carico di lavoro specifico, con conseguente miglioramento delle prestazioni e risparmi sui costi.

Utilizzo dei gruppi di parametri del cluster DB e DB

Amazon RDS e Aurora possono determinare automaticamente i valori dei parametri più adatti per determinate impostazioni in base alla dimensione dell'istanza del database. Supportano anche la personalizzazione dei parametri per l'ottimizzazione delle prestazioni tramite gruppi di parametri per istanze di database e cluster.

È possibile utilizzare i gruppi di parametri del cluster DB e DB per modificare i parametri che controllano vari aspetti del comportamento del motore di database, come l'utilizzo della memoria, l'I/O del disco, la rete e il blocco. Regolando questi parametri, è possibile ottimizzare il motore di database per il carico di lavoro specifico e migliorare le prestazioni.

Puoi creare e configurare gruppi di parametri del cluster DB e DB utilizzando AWS Management Console, the AWS Command Line Interface (AWS CLI) o l'API Amazon RDS. Questa guida presuppone che tu stia utilizzando. AWS CLI Per istruzioni su console e API, consulta Lavorare con i gruppi di parametri DB e Lavorare con i gruppi di parametri del cluster DB nella documentazione di Amazon RDS.

> **Important**
>
> Per utilizzare i AWS CLI comandi forniti in questa guida, devi prima installare e configurare. AWS CLI

Per creare e configurare un gruppo di parametri DB:

```
# 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
```

Per creare e configurare un gruppo di parametri del cluster DB:

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

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

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

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

> **Note**
>
> Aurora e Amazon RDS forniscono un gruppo di parametri predefinito con valori preconfigurati che non possono essere modificati.

I gruppi di parametri possono essere impostati come statici o dinamici. I parametri dinamici vengono applicati immediatamente, indipendentemente dal fatto che l'`ApplyMethod=immediate`opzione sia abilitata. I parametri statici richiedono un riavvio manuale per avere effetto.

Ottimizzazione dei parametri della memoria

L'ottimizzazione dei parametri di memoria è un'attività essenziale per ottimizzare le prestazioni dei database compatibili con Amazon RDS e Aurora PostgreSQL. L'allocazione corretta della memoria per varie operazioni di database, come l'esecuzione di query, l'ordinamento, l'indicizzazione e la memorizzazione nella cache, può migliorare significativamente le prestazioni del database. Questa sezione tratta alcuni dei parametri di memoria più importanti in Amazon RDS e Aurora, compatibili con PostgreSQL, inclusi i valori predefiniti, le formule per il calcolo dei valori appropriati e come modificarli. Per un elenco completo dei parametri, consulta Lavorare con i parametri sull'istanza DB RDS for PostgreSQL nella documentazione di Amazon RDS e i parametri di Amazon Aurora PostgreSQL nella documentazione di Aurora.

L'ottimizzazione di questi parametri richiede una conoscenza approfondita del carico di lavoro del database e delle risorse disponibili sull'istanza DB Aurora o Amazon RDS. Le prestazioni del sistema sono influenzate da due ampie categorie di parametri: parametri vitali e parametri contingenti.

I parametri vitali sono parametri indispensabili che esercitano un'influenza significativa e diretta sulle prestazioni del sistema e sono essenziali per ottenere risultati ottimali.

- buffer condivisi
- temp_buffer
- dimensione_cache_effettiva
- work_mem
- maintenance_work_mem

I parametri contingenti sono parametri specifici dello scenario e dell'azienda. Dipendono da altri fattori e svolgono un ruolo indiretto ma fondamentale nel supportare i parametri vitali e massimizzare le prestazioni complessive del sistema.

- random_page_cost
- seq_page_cost
- track_activity_query_size
- idle_in_transaction_session_timeout
- statement_timeout

- percorso_di ricerca
- numero massimo di connessioni

Questi parametri sono discussi più dettagliatamente nelle sezioni seguenti.

shared_buffers

Il `shared_buffers` parametro controlla la quantità di memoria utilizzata da PostgreSQL per memorizzare nella cache i dati in memoria. L'impostazione di questo parametro su un valore appropriato può aiutare a migliorare le prestazioni delle query.

Per Amazon RDS, il valore predefinito per `shared_buffers` è impostato su `{DBInstanceClassMemory/32768}` byte, in base alla memoria disponibile per l'istanza DB. Per Aurora, il valore predefinito è impostato su`{DBInstanceClassMemory/12038,-50003}`, in base alla memoria disponibile per l'istanza DB. Il valore ottimale per questo parametro dipende da diversi fattori, tra cui la dimensione del database, il numero di connessioni simultanee e la memoria di istanza disponibile.

AWS CLI sintassi

Il comando seguente viene modificato `shared_buffers` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: statico (l'applicazione delle modifiche richiede il riavvio)

Valore predefinito: `{DBInstanceClassMemory/32768}` byte in Amazon RDS for PostgreSQL, compatibile con Aurora `{DBInstanceClassMemory/12038,-50003}` PostgreSQL. Nella maggior

parte dei casi, questa equazione risulta essere circa il 25 percento della memoria del sistema. Seguendo questa linea guida, l'`shared_buffers`impostazione nel gruppo di parametri viene configurata utilizzando le unità predefinite di buffer da 8K di PostgreSQL anziché byte o kilobyte.

L'impostazione `shared_buffers` dei parametri può avere un impatto significativo sulle prestazioni, quindi ti consigliamo di testare accuratamente le modifiche per assicurarti che il valore sia appropriato per il tuo carico di lavoro.

Esempio

Supponiamo che tu abbia un'applicazione di servizi finanziari che esegue un database PostgreSQL su Amazon RDS o Aurora. Questo database viene utilizzato per archiviare i dati delle transazioni dei clienti. Ha un gran numero di tabelle ed è accessibile da più applicazioni su un gran numero di server. L'applicazione presenta prestazioni di query lente e un utilizzo elevato della CPU. L'utente ritiene che l'ottimizzazione del `shared_buffers` parametro possa contribuire a migliorare le prestazioni.

In Amazon RDS for PostgreSQL, il valore `shared_buffers` predefinito di `{DBInstanceClassMemory/32768}` è impostato su byte di memoria `db.r5.xlarge` disponibile in (ad esempio, 3 GB). Per determinare il valore appropriato per`shared_buffers`, esegui una serie di test con valori diversi di`shared_buffers`, iniziando dal valore predefinito della memoria disponibile e aumentando gradualmente il valore. Per ogni test, si misurano le prestazioni delle query e l'utilizzo della CPU del database.

In base ai risultati del test, si determina che impostando il valore `shared_buffers` fino a 8 GB si ottengono le migliori prestazioni complessive delle query e l'utilizzo della CPU per il carico di lavoro. Il valore viene determinato mediante una combinazione di test e analisi delle caratteristiche del carico di lavoro, tra cui la dimensione del database, il numero e la complessità delle query, il numero di utenti simultanei e le risorse di sistema disponibili. Dopo aver apportato la modifica, i sistemi di monitoraggio controllano le prestazioni del database per assicurarsi che il nuovo valore sia appropriato per il carico di lavoro. È quindi possibile ottimizzare i parametri aggiuntivi, se necessario, per migliorare ulteriormente le prestazioni.

temp_buffers

`temp_buffers`è un parametro di configurazione chiave in Aurora, compatibile con PostgreSQL e Amazon RDS for PostgreSQL, che può influire in modo significativo sulle prestazioni per i carichi di lavoro che coinvolgono ordinamenti, hash e operazioni aggregate su tabelle temporanee. Questo parametro determina la quantità di memoria allocata per i buffer temporanei, che a sua

volta influisce sull'efficienza e sulla velocità di tali operazioni. Se la memoria allocata non è sufficientetemp_buffers, il sistema potrebbe dover utilizzare metodi più lenti e meno efficienti per gli ordinamenti, gli hash e le operazioni di aggregazione sulle tabelle temporanee, con conseguenti prestazioni non ottimali.

AWS CLI sintassi

Il comando seguente viene modificato temp_buffers per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) ApplyMethod=immediate

Valore predefinito: 8 MB

Per ulteriori informazioni su questo parametro, vedere Consumo di risorse nella documentazione di PostgreSQL.

Esempio

Se il tuo carico di lavoro prevede molte operazioni di ordinamento, hashing e aggregazione su tabelle temporanee, potresti non allocare abbastanza memoria. temp_buffers In questo caso, il sistema potrebbe dover eseguire operazioni di ordinamento su tabelle temporanee che comportano metodi più lenti e basati su disco anziché operazioni di ordinamento, hashing e aggregazione in memoria. Ciò può causare un rallentamento significativo delle prestazioni delle query, in particolare per le query che coinvolgono set di dati di grandi dimensioni. L'aumento del valore di temp_buffers può garantire la disponibilità di memoria sufficiente per eseguire tali operazioni in memoria, con un conseguente miglioramento significativo delle prestazioni.

Per trovare il valore ottimale di`temp_buffers`, monitora le prestazioni del sistema e identifica le aree con prestazioni non ottimali. Se notate tempi di risposta alle query lenti o un elevato utilizzo della CPU, prendete in considerazione la possibilità di apportare delle modifiche. `temp_buffers` Ad esempio, se il carico di lavoro prevede molte tabelle temporanee, aumentare il valore di `temp_buffers` può contribuire a garantire che tali tabelle vengano archiviate in memoria. Questa operazione può essere molto più veloce rispetto all'utilizzo di I/O di lettura/scrittura dallo storage.

Sperimenta con valori diversi, con piccoli `temp_buffers` incrementi, e monitora attentamente le prestazioni del sistema dopo ogni modifica. Analizza l'impatto di diversi valori sulle prestazioni e ottimizza l'impostazione in base alle caratteristiche specifiche del tuo carico di lavoro.

effective_cache_size

Il `effective_cache_size` parametro specifica la quantità di memoria che PostgreSQL dovrebbe presumere sia disponibile per la memorizzazione nella cache dei dati. L'impostazione corretta di questo parametro può migliorare le prestazioni consentendo a PostgreSQL di utilizzare meglio la memoria disponibile.

AWS CLI sintassi

Il comando seguente viene modificato `effective_cache_size` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

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

Esempio

Una piattaforma di apprendimento online dispone di un ampio database di materiali didattici, dati degli studenti e altri contenuti a cui gli utenti accedono frequentemente. L'applicazione viene eseguita su un'istanza Amazon `db.r5.xlarge` RDS for PostgreSQL con 32 GB di memoria. L'applicazione presenta prestazioni lente quando gli utenti cercano di leggere contenuti a cui si accede di frequente. Dopo aver analizzato l'utilizzo delle risorse del server di database, si determina che PostgreSQL non sta facendo un uso ottimale della memoria disponibile.

Il `effective_cache_size` parametro in Amazon RDS for PostgreSQL controlla la quantità di memoria utilizzata dal server per la memorizzazione nella cache del disco. Il valore predefinito è impostato su `SUM({DBInstanceClassMemory/12038},-50003)` KB, ad esempio la classe`db.r5.xlarge`, ma questo valore predefinito potrebbe non essere appropriato per tutti i carichi di lavoro. In questo esempio, il server del database potrebbe archiviare grandi quantità di materiali didattici e dati degli studenti a cui si accede di frequente. L'aumento del valore del `effective_cache_size` parametro può comportare una maggiore quantità di dati memorizzati nella cache, il che riduce il numero di letture su disco richieste e migliora le prestazioni delle query.

Quando esegui una query, Amazon RDS for PostgreSQL verifica innanzitutto se i dati richiesti dalla query sono già presenti nella cache. In tal caso, i dati possono essere letti dalla memoria anziché essere letti dal disco. Se i dati non sono nella cache, devono essere letti dal disco, operazione che può essere lenta.

Per la piattaforma di apprendimento online, potresti decidere di `effective_cache_size` impostarla su 16 GB (metà della memoria disponibile) dopo test e analisi. Questo valore consente a PostgreSQL di utilizzare meglio la memoria disponibile, riducendo il numero di letture su disco richieste e migliorando le prestazioni delle query.

work_mem

Il `work_mem` parametro controlla la quantità di memoria utilizzata dalle query per le operazioni di ordinamento e hashing. Il suo valore predefinito è 4 MB. Se una query include più operazioni, può utilizzare fino a 4 MB per ciascuna operazione. L'aumento del valore di `work_mem` può migliorare le prestazioni delle query che richiedono l'ordinamento o l'hashing, poiché queste operazioni richiedono più memoria. Tuttavia, un'impostazione troppo elevata di questo parametro può causare un utilizzo eccessivo della memoria, con conseguente peggioramento delle prestazioni.

Per calcolare il valore ottimale per`work_mem`, puoi utilizzare la seguente formula:

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

dove `available_memory` è la quantità totale di memoria disponibile sul server, max_connections è il numero massimo di connessioni consentite e work_mem_fraction è una frazione che determina la quantità di memoria disponibile da allocare a ciascuna connessione.

AWS CLI sintassi

Il comando seguente viene modificato `work_mem` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: 4 MB

Esempio

Uno strumento di analisi dei social media elabora una grande quantità di dati e le interrogazioni che implicano complesse operazioni di ordinamento e unione causano un elevato I/O del disco e si riversano su disco. Se si aumenta il valore `work_mem` da 4 MB a 16 MB, PostgreSQL può utilizzare più memoria per queste operazioni. Ciò riduce la quantità di I/O e migliora le prestazioni delle query.

maintenance_work_mem

Il `maintenance_work_mem` parametro controlla la quantità di memoria utilizzata dalle operazioni di manutenzione come, e la creazione di indici. `VACUUM` `ANALYZE` Il valore predefinito per questo parametro in Amazon RDS e Aurora è 64 MB.

Per calcolare il valore appropriato per questo parametro, puoi utilizzare questa formula:

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

Aurora PostgreSQL Compatible Edition e Amazon RDS for PostgreSQL applicano la seguente formula per impostare il valore ottimale:

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

AWS CLI sintassi

Il comando seguente viene modificato `maintenance_work_mem` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: 64 MB

Esempio

La tua applicazione su larga scala utilizza un database PostgreSQL ospitato su Aurora o Amazon RDS. Noti che il database è lento e non risponde durante le attività di manutenzione come l'archiviazione e l'indicizzazione. È possibile monitorare parametri quali l'utilizzo della memoria, i tempi delle operazioni di manutenzione e l'utilizzo della CPU per determinare se il valore corrente di causa problemi. `maintenance_work_mem`

Per determinare il valore ottimale di`maintenance_work_mem`, è possibile regolare il parametro e monitorarne l'impatto. Se l'utilizzo della memoria è costantemente elevato o i tempi di funzionamento sono più lunghi del previsto durante le operazioni di manutenzione, un aumento `maintenance_work_mem` potrebbe essere utile. Al contrario, se l'utilizzo della CPU è costantemente elevato durante le operazioni di manutenzione, `maintenance_work_mem` potrebbe essere utile ridurlo. Ripetendo le regolazioni e i test, è possibile trovare il valore ottimale in `maintenance_work_mem` grado di fornire il miglior equilibrio tra utilizzo della memoria, tempi di manutenzione e utilizzo della CPU.

Durante l'indagine, supponiamo che tu determini che il valore predefinito di 64 MB per `maintenance_work_mem` è troppo basso per le dimensioni del database. Di conseguenza, le operazioni di manutenzione richiedono più tempo per essere completate, causano tempi di inattività eccessivi e rallentano le prestazioni dell'applicazione. Per risolvere questo problema, potreste decidere di ottimizzare il `maintenance_work_mem` parametro aumentandolo da 64 MB a 512 MB (che identificate come il valore ottimale). L'applicazione della modifica può migliorare i tempi di manutenzione di due terzi. Ad esempio, un'operazione sotto vuoto che in precedenza richiedeva 30 minuti per essere completata potrebbe ora richiedere solo 10 minuti. Grazie a questa ottimizzazione, il database è ora in grado di gestire le attività di manutenzione in modo più efficiente.

random_page_cost

Il `random_page_cost` parametro aiuta a determinare il costo dell'accesso casuale alle pagine. Il pianificatore di query in Amazon RDS e Aurora utilizza questo parametro, insieme ad altre statistiche sulla tabella, per determinare il piano più efficiente per l'esecuzione di una query.

I `random_page_cost` parametri `seq_page_cost` and sono strettamente correlati e di solito vengono utilizzati insieme dal pianificatore per confrontare i costi dei diversi metodi di accesso e decidere quale sia il più efficiente. Pertanto, se si modifica uno di questi parametri, è necessario considerare anche se è necessario modificare l'altro parametro.

In generale, il pianificatore di query cerca di ridurre al minimo i costi di esecuzione di una query. Il costo viene determinato utilizzando una combinazione del numero di letture di pagine su disco e del valore di. `random_page_cost` Un valore più alto di `random_page_cost` tende a favorire le scansioni sequenziali, mentre un valore più basso tende a favorire le scansioni indicizzate. Un valore più basso tende inoltre a favorire i loop join annidati anziché gli hash join.

Il `random_page_cost` parametro utilizza il valore predefinito del motore PostgreSQL (4) a meno che non venga impostato un valore nel gruppo di parametri o nella sessione locale. È possibile

ottimizzare questo valore in base alle caratteristiche specifiche del server e del carico di lavoro. Se la maggior parte degli indici utilizzati nel carico di lavoro si trova nella memoria o nella cache a più livelli Aurora, è opportuno modificare il valore di con un valore vicino `random_page_cost` a. `seq_page_cost`

AWS CLI sintassi

Il comando seguente viene modificato `random_page_cost` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: 4

Esempio

Supponiamo di avere un database che memorizza una grande quantità di dati in una tabella a cui vengono spesso eseguite interrogazioni con filtri su colonne non indicizzate. Il completamento delle query richiede molto tempo e il pianificatore di query non seleziona il piano più efficiente per l'accesso ai dati.

Un modo per migliorare le prestazioni sarebbe quello di ridurre il `random_page_cost` parametro. Se lo imposti su 1, il costo dell'accesso casuale alla pagina sarebbe quattro volte inferiore rispetto al valore predefinito. Se si mantiene `random_page_cost` il valore predefinito di 4, l'accesso casuale alle pagine sarebbe quattro volte più costoso dell'accesso sequenziale alle pagine (come determinato dal `seq_page_cost` parametro, che per impostazione predefinita è 1,0). Tuttavia, in questo caso specifico, l'accesso casuale alle pagine potrebbe effettivamente essere molto più costoso a seconda del tipo di archiviazione.

La riduzione del valore del `random_page_cost` parametro può aumentare la probabilità che il pianificatore di query selezioni un piano basato su indici o utilizzi un metodo di accesso diverso, più adatto alle caratteristiche specifiche della tabella.

Si consiglia di monitorare le prestazioni delle query dopo aver modificato il parametro e di aver apportato le modifiche necessarie. Dovresti anche controllare il pianificatore di query con una `EXPLAIN` dichiarazione per verificare se sta selezionando un piano efficiente.

Questo è solo un esempio. L'impostazione ottimale dipende dalle caratteristiche specifiche del carico di lavoro. Inoltre, questo è solo un aspetto dell'ottimizzazione delle prestazioni; è necessario prendere in considerazione anche altri parametri e opzioni di configurazione che possono influire sulle prestazioni delle query.

seq_page_cost

Il `seq_page_cost` parametro aiuta a determinare il costo dell'accesso sequenziale alle pagine. Il pianificatore di query in Amazon RDS e Aurora utilizza questo parametro, insieme ad altre statistiche sulla tabella, per determinare il piano più efficiente per l'esecuzione di una query.

I `random_page_cost` parametri `seq_page_cost` and sono strettamente correlati e di solito vengono utilizzati insieme dal pianificatore per confrontare i costi dei diversi metodi di accesso e decidere quale sia il più efficiente. Pertanto, se si modifica uno di questi parametri, è necessario considerare anche se è necessario modificare l'altro parametro.

Quando si accede a una tabella in modo sequenziale, PostgreSQL può utilizzare la cache del file system del sistema operativo per fornire un accesso più rapido. Per impostazione predefinita, `seq_page_cost` è impostato su 1.0, il che presuppone che l'accesso sequenziale alle pagine sia economico quanto la lettura di un singolo blocco di disco. Se si dispone di una tabella a cui si accede principalmente in modo sequenziale, ma l'accesso al disco è lento perché limitato da un numero inferiore di IOPS, è consigliabile aumentare il valore di in modo da riflettere il costo aggiuntivo di accesso `seq_page_cost` al disco.

La modifica del valore di questo parametro influisce su tutte le query eseguite nel sistema, pertanto è consigliabile testare le query con valori diversi per determinare il valore ottimale per il caso d'uso specifico.

AWS CLI sintassi

Il comando seguente viene modificato `seq_page_cost` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: 1.0

Esempio

Supponiamo di avere un database che memorizza una grande quantità di dati in una tabella a cui si accede principalmente in modo sequenziale. La tabella viene utilizzata principalmente per la creazione di report, le query vengono eseguite molto lentamente e il pianificatore di query non è in grado di selezionare il piano più efficiente per l'accesso ai dati.

Un modo per migliorare le prestazioni sarebbe quello di ridurre il `seq_page_cost` parametro. Il valore predefinito è 1.0, che presuppone che l'accesso sequenziale alla pagina sia economico quanto la lettura di un singolo blocco del disco. Tuttavia, in questo caso specifico, l'accesso sequenziale alle pagine potrebbe effettivamente essere più costoso a causa del tipo di archiviazione (a seconda dell'I/O). Se si imposta su `seq_page_cost` 0,5, il costo dell'accesso sequenziale alle pagine è la metà rispetto al valore predefinito. Questa modifica può aumentare la probabilità che il pianificatore di query selezioni un piano che utilizza un metodo di accesso sequenziale più adatto alle caratteristiche specifiche della tabella.

Si consiglia di monitorare le prestazioni delle query dopo aver modificato il parametro e di aver apportato le modifiche necessarie. È inoltre necessario controllare il piano di query con una `EXPLAIN` dichiarazione per verificare se si sta selezionando un piano efficiente.

Questo è solo un esempio. L'impostazione ottimale dipende dalle caratteristiche specifiche del carico di lavoro. Inoltre, questo è solo un aspetto dell'ottimizzazione delle prestazioni; è necessario considerare anche altri parametri e opzioni di configurazione che influiscono sulle prestazioni delle query.

track_activity_query_size

Il `track_activity_query_size` parametro controlla la dimensione della stringa di query che viene registrata per ogni sessione attiva nella vista. `pg_stat_activity` Per impostazione predefinita, solo i primi 1.024 byte della stringa di query vengono registrati in Amazon RDS for PostgreSQL e 4.096 byte vengono registrati in Aurora PostgreSQL compatibile. Se desideri registrare interrogazioni più lunghe, puoi impostare questo parametro su un valore più alto.

AWS CLI sintassi

Il comando seguente viene modificato `track_activity_query_size` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: statico (l'applicazione delle modifiche richiede il riavvio)

Valore predefinito: 1.024 byte (Amazon RDS for PostgreSQL), 4.096 byte (compatibile con Aurora PostgreSQL)

Esempio

Il tuo database Amazon RDS for PostgreSQL presenta prestazioni di query lente e sospetti che il problema possa essere correlato a query a esecuzione prolungata. Puoi indagare ulteriormente registrando le query più lunghe nella vista. `pg_stat_activity`

L'aumento del valore del `track_activity_query_size` parametro può comportare un aumento della registrazione, che può avere un impatto sulle prestazioni del database. Si consiglia di ripristinare il valore predefinito di 1.024 del parametro dopo la risoluzione del problema.

idle_in_transaction_session_timeout

Il parametro controlla la quantità di tempo di attesa di una transazione inattiva prima di essere interrotta. `idle_in_transaction_session_timeout`

Il valore predefinito per questo parametro in Amazon RDS for PostgreSQL e Aurora PostgreSQL Compatible è 86.400.000 millisecondi.

AWS CLI sintassi

Il comando seguente viene modificato `idle_in_transaction_session_timeout` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: 86.400.000 millisecondi (compatibile con Aurora PostgreSQL)

Esempio

Hai un'applicazione di e-commerce che elabora gli ordini online. L'applicazione utilizza un database PostgreSQL ospitato su Amazon RDS o Aurora. Ogni volta che un cliente effettua un ordine, l'applicazione avvia una nuova transazione per aggiornare l'inventario e i record degli ordini.

Se una transazione rimane inattiva per un lungo periodo, può impedire ad altre transazioni di accedere agli stessi record, causando problemi di prestazioni e potenzialmente anche tempi di inattività delle applicazioni. Inoltre, le transazioni inattive che non vengono interrotte correttamente possono consumare preziose risorse di sistema come memoria e CPU.

Per evitare tali problemi, potete impostare il `idle_in_transaction_session_timeout` parametro su un valore adatto alla vostra applicazione. Ad esempio, è possibile impostarlo su 5 minuti (300 secondi) in modo che qualsiasi transazione rimasta inattiva per più di 5 minuti venga interrotta automaticamente. Ciò può contribuire a garantire che le risorse di sistema vengano utilizzate in modo efficiente e che l'applicazione sia in grado di gestire un gran numero di ordini senza rallentamenti. Impostando il valore appropriato per`idle_in_transaction_session_timeout`, puoi contribuire a garantire che la tua applicazione funzioni in modo ottimale su Amazon RDS o Aurora.

statement_timeout

Il `statement_timeout` parametro imposta il tempo massimo di esecuzione di una query prima che venga interrotta.

AWS CLI sintassi

Il comando seguente viene modificato `statement_timeout` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate)
`ApplyMethod=immediate`

Valore predefinito: 0 millisecondi (nessun timeout)

Esempio

La tua applicazione web consente agli utenti di effettuare ricerche in un ampio database di prodotti. Il completamento delle query di ricerca a volte può richiedere molto tempo, causando tempi di risposta lenti per gli utenti. Per risolvere questo problema, è possibile impostare il `statement_timeout` parametro su un valore basso, ad esempio 10 secondi, in modo da forzare l'interruzione di qualsiasi query che impiega più di 10 secondi.

Potrebbe sembrare una misura drastica, ma in realtà può essere molto efficace per migliorare le prestazioni. In molti casi, le query di lunga durata sono causate da query SQL mal ottimizzate o da indici inefficienti. Impostando un `statement_timeout` valore basso, è possibile identificare queste query problematiche e adottare misure per ottimizzarle.

Ad esempio, supponete di scoprire che una determinata query di ricerca scade costantemente. È possibile utilizzare strumenti come `EXPLAIN` e `EXPLAIN ANALYZE` per analizzare la query e identificare eventuali rallentamenti nelle prestazioni. Una volta identificato il problema, è possibile adottare misure per ottimizzare la query aggiungendo nuovi indici, riscrivendo la query o utilizzando un algoritmo di ricerca diverso. Analizzando e ottimizzando continuamente le query SQL in questo modo, è possibile migliorare in modo significativo le prestazioni dell'applicazione.

search_path

Il `search_path` parametro determina l'ordine in cui gli schemi vengono cercati per gli oggetti nelle istruzioni SQL. Il valore predefinito è`$user, public`, il che significa che PostgreSQL cerca gli oggetti prima nello schema che corrisponde al nome dell'utente e poi nello schema pubblico.

Se disponi di un numero elevato di schemi o devi accedere agli oggetti di uno schema specifico, la modifica del `search_path` parametro può contribuire a migliorare le prestazioni. Quando si imposta `search_path` uno schema specifico, PostgreSQL può trovare gli oggetti più rapidamente senza dover cercare tra più schemi.

Per modificare il `search_path` parametro in Amazon RDS e Aurora, puoi utilizzare il seguente comando `ROLE` per il livello:

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

AWS CLI sintassi

Il comando seguente viene modificato `search_path` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: `$user, public`

Esempio

Hai un'applicazione multi-tenant con schemi separati per ogni tenant che utilizza un database compatibile con Amazon RDS for PostgreSQL o Aurora PostgreSQL e devi eseguire una query che prevede l'unione di dati da più schemi.

Per impostazione predefinita, Amazon RDS e Aurora utilizzano il percorso di ricerca per determinare quale schema utilizzare per una determinata tabella. Il percorso di ricerca è un elenco di nomi di schema che PostgreSQL cerca in ordine quando si fa riferimento a una tabella senza specificare il nome dello schema. Per impostazione predefinita, Amazon RDS e Aurora cercano prima la tabella nello schema che ha lo stesso nome dell'utente corrente, quindi cercano nello schema pubblico.

Supponiamo che tu voglia eseguire una query che prevede l'unione di tabelle da più schemi`tenant1`, `tenant2` denominate e. `tenant3` Per utilizzare gli schemi dei tenant, è possibile includere i nomi degli schemi nella query:

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

Tuttavia, un metodo più efficiente consiste nel modificare il *search_path* parametro per includere gli schemi dei tenant utilizzando i comandi nella sezione sulla sintassi.AWS CLI Puoi anche usare il SET comando in una sessione PostgreSQL:

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

È quindi possibile scrivere la query senza qualificare i nomi degli schemi:

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

Ciò può rendere la query più concisa e più facile da leggere e può anche semplificare il codice dell'applicazione se sono presenti molte query che prevedono l'unione di tabelle di più schemi.

max_connections

Il max_connections parametro imposta il numero massimo di connessioni simultanee per il database PostgreSQL.

AWS CLI sintassi

Il comando seguente viene modificato max_connections per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: statico (l'applicazione delle modifiche richiede il riavvio)

Valore predefinito: connessioni `LEAST(DBInstanceClassMemory/9531392,5000)`

Per ottimizzare l'uso di `max_connections` Amazon RDS o Aurora e minimizzarne l'impatto sulle prestazioni, prendi in considerazione le seguenti best practice:

- Imposta il valore del parametro in base alle risorse di sistema disponibili.
- Monitora l'utilizzo della connessione per evitare di raggiungere rapidamente il limite.
- Utilizza il pool di connessioni per ridurre il numero di connessioni richieste.
- Usa Amazon RDS Proxy per il pooling di connessioni.

Quando esegui il `max_connections` tuning con Amazon RDS for PostgreSQL o la compatibilità con Amazon Aurora PostgreSQL, considera i tipi di istanze disponibili e le relative risorse allocate e concentrati sulla memoria e sulla capacità della CPU. I dettagli di storage e I/O sono gestiti da AWS, quindi puoi monitorare le caratteristiche generali del carico di lavoro e i parametri di sistema, ad esempio tramite `FreeableMemory` Amazon o CloudWatch la console Amazon RDS per confermare che c'è abbastanza memoria per le connessioni. Monitora `CPUUtilization` i valori elevati, che potrebbero indicare la necessità di aggiustamenti. Evita di impostare `max_connections` valori troppo alti, poiché possono influire sull'utilizzo della memoria e potenzialmente influenzare l'I/O indirettamente. Tieni presente che ogni connessione consuma memoria. Per trovare il giusto equilibrio, aumenta lentamente `max_connections` e osserva come influisce sul sistema. Fai attenzione ai segnali di un rallentamento delle prestazioni o di un maggiore utilizzo della CPU. Verifica se la tua applicazione funziona ancora bene. Utilizza funzionalità come le repliche di lettura in Aurora per distribuire il traffico di lettura e ridurre il carico sull'istanza principale. Esamina e `max_connections` aggiusta regolarmente in base ai modelli di utilizzo osservati per garantire prestazioni ottimali del database entro i limiti di risorse specificati.

Regolazione dei parametri dell'autovacuum

I database Amazon RDS for PostgreSQL e la compatibilità con Aurora PostgreSQL richiedono una manutenzione periodica nota come vacuuming. Autovacuum è un'utilità PostgreSQL integrata che rimuove i dati obsoleti o non necessari per liberare spazio nel database. Il processo autovacuum esegue il comando in background a intervalli regolari. `VACUUM`

L'ottimizzazione delle impostazioni dell'autovacuum è un passaggio fondamentale per mantenere le prestazioni, la stabilità e la disponibilità del tuo sistema di database compatibile con Amazon RDS for PostgreSQL o Aurora PostgreSQL. Regolando i parametri dell'autovacuum in base al carico di lavoro e alle dimensioni del database, puoi ottimizzare le prestazioni del processo di autovacuum e ridurne l'impatto sulle risorse di sistema, migliorando così lo stato generale del database.

Oltre a regolare le impostazioni dell'autovacuum, è importante monitorare le prestazioni del database e dei suoi componenti utilizzando gli strumenti e i parametri disponibili in Amazon RDS e Aurora. Monitorando parametri prestazionali come bloat, spazio liberabile e tempi di esecuzione delle query, puoi identificare potenziali problemi prima che diventino gravi e intraprendere le azioni appropriate per risolverli.

In questa sezione vengono descritti i seguenti argomenti e parametri relativi all'autovacuum:

- Comandi VACUUM e ANALYZE
- Controllo del gonfiore
- aspirapolvere automatico
- autovacuum_work_mem
- autovacuum_naptime
- autovacuum_max_workers
- autovacuum_vacuum_scale_factor
- autovacuum_vacuum_threshold
- autovacuum_analyze_scale_factor
- autovacuum_analyze_threshold
- autovacuum_vacuum_cost_limit

Per ulteriori informazioni su autovacuum, consulta i seguenti collegamenti:

- Comprendere l'autovacuum negli ambienti Amazon RDS for PostgreSQL (post sul blog)
- Utilizzo dell'autovacuum PostgreSQL su Amazon RDS per PostgreSQL (documentazione Amazon RDS)
- Aspirazione parallela in Amazon RDS per PostgreSQL e Amazon Aurora PostgreSQL (post di blog)

Comandi VACUUM e ANALYZE

`VACUUM`garbage raccoglie e, facoltativamente, analizza un database. Per la maggior parte delle applicazioni, è sufficiente lasciare che il demone autovacuum esegua l'aspirazione. Tuttavia, alcuni amministratori potrebbero voler modificare i parametri del database per autovacuum o integrare o sostituire le attività del demone utilizzando comandi gestiti manualmente che possono essere eseguiti secondo uno scheduler. `VACUUM`

`VACUUM`recupera lo spazio di archiviazione occupato da tuple morte. Nelle operazioni PostgreSQL standard, quando le tuple vengono eliminate o rese obsolete da un aggiornamento, non vengono rimosse fisicamente dalle tabelle finché non viene eseguita un'operazione. `VACUUM` Pertanto, si consiglia di eseguirla `VACUUM` periodicamente, specialmente su tabelle che vengono aggiornate frequentemente.

L'ottimizzazione `VACUUM` dei parametri è particolarmente importante in Amazon RDS for PostgreSQL e Aurora, compatibile con PostgreSQL, perché questi servizi di database gestiti hanno caratteristiche diverse rispetto ai database PostgreSQL autogestiti. Queste differenze possono influire sulle prestazioni delle operazioni di sottovuoto. L'ottimizzazione `VACUUM` dei parametri è essenziale per ottimizzare l'uso delle risorse e garantire che le operazioni di vacuo non influiscano negativamente sulle prestazioni e sulla disponibilità del sistema di database.

Ecco alcuni dei parametri che puoi usare con il `VACUUM` comando in Aurora, compatibile con PostgreSQL e Amazon RDS for PostgreSQL:

- `FULL`
- `FREEZE`
- `VERBOSE`
- `ANALYZE`
- `DISABLE_PAGE_SKIPPING`
- `table_name`

- `column_name`

`VACUUM ANALYZE`esegue un'operazione seguita da un'operazione per ogni tabella `VACUUM` selezionata. `ANALYZE` Fornisce un modo efficiente per eseguire la manutenzione ordinaria.

L'utilizzo del `VACUUM` comando senza l'`FULL`opzione consente di recuperare spazio per il riutilizzo. Non richiede un blocco esclusivo sulla tabella, quindi è possibile eseguire questo comando durante le operazioni di lettura e scrittura standard. Tuttavia, nella maggior parte dei casi, il comando non restituisce spazio aggiuntivo al sistema operativo ma lo mantiene disponibile per il riutilizzo all'interno della stessa tabella. `VACUUM FULL`riscrive l'intero contenuto della tabella in un nuovo file su disco senza spazio aggiuntivo e consente di restituire lo spazio inutilizzato al sistema operativo. Questo modulo è molto più lento e richiede un `ACCESS EXCLUSIVE` blocco su ogni tabella.

Per informazioni complete su questi parametri, consulta la documentazione di PostgreSQL.

In Aurora e Amazon RDS, autovacuum è un processo daemon (utilità in background) che esegue regolarmente `ANALYZE` i comandi `VACUUM` e per pulire i dati ridondanti nel database e nel server. Anche se ti affidi all'autovacuuming, ti consigliamo di rivedere e modificare le impostazioni dell'autovacuum descritte nelle sezioni seguenti per garantire prestazioni ottimali.

Controllo del rigonfiamento

La seguente query SQL esamina ogni tabella dello schema XML e identifica le righe morte (tuple) che occupano spazio su 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 questa query restituisce un'alta percentuale (pfrag) di tuple morte, è possibile utilizzare il comando per recuperare spazio. `VACUUM`

Per monitorare le dimensioni dei dati prima e dopo le transazioni, esegui la seguente query nella shell dopo la connessione a un database specifico:

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

autovacuum

Puoi impostare autovacuum a livello globale utilizzando il parametro di `autovacuum` configurazione oppure puoi modificarlo per tabella impostando la colonna della tabella su o per una tabella specifica. `autovacuum_enabled` `pg_class` `true` `false`

Quando si abilita l'autovacuum su una tabella, il server del database analizza periodicamente la tabella alla ricerca di righe e tuple morte e le rimuove in background, senza alcun intervento da parte dell'amministratore del database. Questo aiuta a mantenere la tabella piccola, a migliorare le prestazioni delle query e a ridurre le dimensioni dei backup.

AWS CLI sintassi

Il comando seguente abilita `autovacuum` uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: abilitato

Puoi anche disabilitare o abilitare l'autovacuum su una tabella specifica usando psql:

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

Un'eccessiva pulizia può influire sulle prestazioni, quindi è importante monitorare le prestazioni del processo di autovacuum e le prestazioni del database e modificare le impostazioni secondo necessità.

Esempio

Il database PostgreSQL ha una tabella che riceve un volume elevato di operazioni di scrittura ed eliminazione. Senza autovacuum, questa tabella finirebbe per riempirsi di righe morte (ovvero righe contrassegnate per l'eliminazione ma che non sono ancora state rimosse fisicamente dalla tabella). Queste righe morte occuperebbero spazio sul disco, rallenterebbero le query e aumenterebbero le dimensioni dei backup. È possibile abilitare l'autovacuum sulla tabella per rilevare automaticamente le righe morte e rimuoverle per mitigare questi problemi.

autovacuum_work_mem

`autovacuum_work_mem`è un parametro di configurazione PostgreSQL che controlla la quantità di memoria utilizzata dal processo autovacuum quando esegue attività di manutenzione delle tabelle come l'vacuumuming o l'analisi.

In Aurora e Amazon RDS, puoi regolare il valore di `autovacuum_work_mem` per ottimizzare le prestazioni.

AWS CLI sintassi

Il comando seguente abilita `autovacuum_work_mem` uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: `GREATEST({DBInstanceClassMemory/32768},131072)` KB in Aurora PostgreSQL compatibile, 64 MB in Amazon RDS for PostgreSQL. Tuttavia, il valore predefinito potrebbe variare a seconda della versione specifica di Amazon RDS o Aurora che stai utilizzando.

Esempio

Il tuo database Amazon RDS for PostgreSQL ha una tabella di grandi dimensioni che viene aggiornata frequentemente. Con il passare del tempo, noti che il database diventa più lento e sospetti che il completamento di autovacuum richieda troppo tempo.

Nell'ambito dell'indagine, controllate i log di sistema, utilizzate la `pg_stat_activity` visualizzazione per vedere quali query e processi sono attualmente in esecuzione, controllate la vista per `pg_stat_user_tables` visualizzare le statistiche per ogni tabella, utilizzate la `pg_settings` vista per confrontare il valore della memoria rispetto alla memoria disponibile sul sistema e monitorate l'utilizzo della memoria `autovacuum_work_mem` per individuare eventuali picchi. Dopo aver raccolto queste informazioni, puoi `autovacuum_work_mem` impostarle sul valore ottimale richiesto dal tuo carico di lavoro. Per trovare il giusto equilibrio tra utilizzo della memoria e prestazioni, potresti decidere di impostarla su un quarto della memoria disponibile sul sistema. Dopo aver modificato il valore, si monitorano le prestazioni del database e si può verificare che il completamento dell'operazione Autovacuum sia molto più rapido rispetto a prima e che il database abbia prestazioni complessivamente più elevate.

autovacuum_naptime

Il `autovacuum_naptime` parametro controlla l'intervallo di tempo tra le esecuzioni successive del processo autovacuum. Il valore predefinito è 15 secondi per Amazon RDS for PostgreSQL e 5 secondi per Aurora PostgreSQL Compatible.

Ad esempio, supponiamo che il tuo database Amazon RDS for PostgreSQL abbia una tabella che riceve un volume elevato di operazioni di scrittura ed eliminazione. Se mantieni l'impostazione predefinita, le frequenti scansioni autovacuum interromperanno questa tabella altamente transattiva. Se imposti questo parametro su un valore elevato, ci sarà un intervallo più lungo tra le scansioni successive e le righe morte verranno rimosse meno frequentemente.

È possibile `autovacuum_naptime` utilizzarlo per gestire il carico causato dal processo di vacuazione, soprattutto se si dispone di un server occupato che ha già un elevato carico di CPU o I/O. Più a lungo si imposta l'ora del sonnellino, meno frequentemente viene eseguito l'autovacuum, il che riduce il carico sul server. Tuttavia, l'impostazione `autovacuum_naptime` su un valore molto alto può causare la crescita delle tabelle PostgreSQL e l'accumulo di righe morte, con conseguente riduzione delle prestazioni. Si consiglia di monitorare le prestazioni del processo di autovacuum e di regolare l'impostazione secondo necessità. `autovacuum_naptime`

AWS CLI sintassi

Il comando seguente viene modificato `autovacuum_naptime` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: 15 secondi (Amazon RDS for PostgreSQL), 5 secondi (compatibile con Aurora PostgreSQL)

autovacuum_max_workers

Il `autovacuum_max_workers` parametro controlla il numero massimo di processi di lavoro che il processo autovacuum può creare. Ogni processo di lavoro è responsabile dell'aspirazione o dell'analisi di una singola tabella.

Ad esempio, supponiamo di disporre di un database di grandi dimensioni con molte tabelle che vengono aggiornate ed eliminate frequentemente. Se si imposta un valore basso, ad esempio 1, è possibile aspirare solo un tavolo alla volta e la pulizia di tutti i tavoli richiede più tempo. `autovacuum_max_workers` Se si imposta un valore elevato, `autovacuum_max_workers` ad esempio 8, è possibile aspirare fino a otto tavoli contemporaneamente. Questo può velocizzare il processo di pulizia dei database che contengono molte tabelle.

AWS CLI sintassi

Il comando seguente viene modificato `autovacuum_max_workers` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: statico (l'applicazione delle modifiche richiede il riavvio)

Valore predefinito: workers `GREATEST(DBInstanceClassMemory/64371566592,3)`

L'aumento dell'`autovacuum_max_workers`impostazione può aumentare il carico sul server, il che può influire sulle prestazioni se non si dispone di risorse sufficienti. L'impostazione ottimale dipende dai requisiti specifici del database, dalle sue dimensioni e dal numero di tabelle che contiene. Ti consigliamo di sperimentare diversi valori e monitorare le prestazioni per trovare l'impostazione ottimale per il tuo caso d'uso.

autovacuum_vacuum_scale_factor

Il parametro di configurazione controlla l'aggressività del processo di aspirazione automatica durante l'aspirazione di una tabella. `autovacuum_vacuum_scale_factor`

Il fattore di scala del vuoto è una frazione del numero totale di tuple in una tabella che deve essere modificata prima che l'autovacuum pulisca la tabella. Il valore predefinito è 0,1 (ovvero, il 10 percento delle tuple deve essere modificato). Ad esempio, se una tabella contiene 1.000.000 di tuple e 100.000 di tali tuple sono contrassegnate come morte o eliminate, l'aspirapolvere automatico aspira la tabella, a seconda del valore di come fattore di controllo. `autovacuum_vacuum_threshold`

Il `autovacuum_vacuum_scale_factor` parametro consente di controllare la frequenza di esecuzione del processo di aspirazione. Se una tabella riceve molte operazioni di scrittura, potresti

voler ridurre il fattore di scala del vuoto in modo che l'autovacuum venga eseguito più frequentemente e mantenere la tabella più piccola. Al contrario, se una tabella riceve poche operazioni di scrittura, potresti voler aumentare il fattore di scala del vuoto in modo che l'autovacuum venga eseguito meno frequentemente e consenta di risparmiare risorse.

AWS CLI sintassi

Il comando seguente viene modificato `autovacuum_vacuum_scale_factor` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: 0,1

Il `autovacuum_vacuum_scale_factor` parametro funziona insieme ai `autovacuum_vacuum_cost_limit, and autovacuum_naptime` parametri`autovacuum_vacuum_threshold`,. Per ulteriori informazioni su questo parametro, consulta il post di AWS blog Understanding autovacuum in Amazon RDS for PostgreSQL Environments.

autovacuum_vacuum_threshold

Il `autovacuum_vacuum_threshold` parametro controlla il numero minimo di operazioni di aggiornamento o eliminazione delle tuple che devono essere eseguite su una tabella prima che autovacuum la aspiri. Questa impostazione può essere utile per evitare l'eliminazione non necessaria

dell'aspirapolvere su tavoli che non hanno una frequenza elevata di queste operazioni. Il valore predefinito è 50, che è l'impostazione predefinita del motore PostgreSQL, sia per Amazon RDS for PostgreSQL che per Aurora PostgreSQL compatibile.

Ad esempio, supponiamo di avere una tabella con 100.000 righe e impostata su 50. `autovacuum_vacuum_threshold` Se la tabella riceve solo 49 aggiornamenti o eliminazioni, autovacuum non la eliminerà. Se la tabella riceve 50 o più aggiornamenti o eliminazioni, autovacuum la eliminerà, a seconda del valore `autovacuum_vacuum_scale_factor` moltiplicato per il numero di righe della tabella come fattore di controllo.

L'impostazione di questo parametro su un valore troppo elevato può causare l'ingrandimento della tabella e l'accumulo di righe morte, con conseguenti ripercussioni sulle prestazioni.

AWS CLI sintassi

Il comando seguente viene modificato `autovacuum_vacuum_threshold` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: 50 operazioni

Il `autovacuum_vacuum_threshold` parametro funziona in combinazione con i `autovacuum_vacuum_cost_limit, and autovacuum_naptime` parametri`autovacuum_vacuum_scale_factor`,. Le impostazioni ottimali dipendono dai requisiti specifici del database e dalle dimensioni della tabella.

Per ulteriori informazioni su questo parametro, consulta il post di AWS blog Understanding autovacuum in Amazon RDS for PostgreSQL Environments.

autovacuum_analyze_scale_factor

Il `autovacuum_analyze_scale_factor` parametro controlla l'aggressività del processo autovacuum durante l'analisi (raccolta) di statistiche sulla distribuzione dei dati in una tabella.

Il processo di autovacuum utilizza questo parametro per calcolare una soglia basata sul numero di tuple in una tabella. Se il numero di inserimenti, aggiornamenti o eliminazioni di tuple supera questa soglia, autovacuum analizza la tabella. Il valore predefinito è 0,05 (ovvero il 5% delle tuple deve essere modificato) sia per Amazon RDS for PostgreSQL che per Aurora PostgreSQL compatibile.

Ad esempio, supponiamo che la tua tabella abbia 1.000.000 di tuple e che tu mantenga il valore predefinito a 0,05. `autovacuum_analyze_scale_factor` Se la tabella riceve 50.000 o più aggiornamenti o eliminazioni, autovacuum la aspira, a seconda del `autovacuum_analyze_threshold` valore e aggiungendo il numero di righe della tabella come fattore di controllo.

AWS CLI sintassi

Il comando seguente viene modificato `autovacuum_analyze_scale_factor` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: 0,05 (5 percento)

È essenziale che il pianificatore di query raccolga statistiche per prendere decisioni informate, ad esempio su come accedere ai dati e come organizzarli, quindi consigliamo di monitorare le prestazioni del processo di autovacuum e di modificare le impostazioni secondo necessità per garantire che le statistiche siano aggiornate.

Il `autovacuum_analyze_scale_factor` parametro funziona in combinazione con i parametri. `autovacuum_analyze_threshold` `autovacuum_analyze_cost_limit, and autovacuum_naptime` L'impostazione ottimale dipende dai requisiti specifici del database e delle dimensioni della tabella e dalla frequenza degli aggiornamenti. Per ulteriori informazioni su questo parametro, consulta il post di AWS blog Understanding autovacuum in Amazon RDS for PostgreSQL Environments.

autovacuum_analyze_threshold

`autovacuum_vacuum_threshold`Il parametro è simile a. `autovacuum_analyze_threshold` Controlla il numero minimo di inserimenti, aggiornamenti o eliminazioni di tuple che devono avvenire su una tabella prima che autovacuum la analizzi. Questa impostazione può essere utile per evitare operazioni di pulizia non necessarie su tavoli che non presentano una frequenza elevata di queste operazioni. Il valore predefinito è 50, che è l'impostazione predefinita del motore PostgreSQL, sia per Amazon RDS for PostgreSQL che per Aurora PostgreSQL compatibile.

Ad esempio, supponiamo di avere una tabella con 100.000 righe e di mantenere il valore predefinito a 50. `autovacuum_analyze_threshold` Se la tabella riceve solo 49 inserimenti, aggiornamenti o eliminazioni, autovacuum non la analizza. Se la tabella riceve 50 o più inserimenti, aggiornamenti o eliminazioni, autovacuum la analizzerà, mantenendo come fattore di controllo il valore `autovacuum_analyze_scale_factor` moltiplicato per il numero di righe della tabella.

AWS CLI sintassi

Il comando seguente viene modificato `autovacuum_analyze_threshold` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: 50 operazioni

Questo parametro funziona in combinazione con il `autovacuum_analyze_scale_factor` parametro, quindi tieni conto di entrambe le impostazioni quando configuri l'autovacuum.

È essenziale che il pianificatore di query raccolga statistiche per prendere decisioni informate, ad esempio come accedere ai dati e come organizzarli. Un'impostazione `autovacuum_analyze_threshold` troppo alta può far sì che le statistiche diventino obsolete, con conseguenti prestazioni scadenti. Si consiglia di monitorare le prestazioni del processo di aspirazione automatica e di regolare le impostazioni secondo necessità.

Per ulteriori informazioni su questo parametro, consulta il post di AWS blog Understanding autovacuum in Amazon RDS for PostgreSQL Environments.

autovacuum_vacuum_cost_limit

Il `autovacuum_vacuum_cost_limit` parametro controlla la quantità di risorse di CPU e I/O che un autovacuum worker può consumare.

Limitare l'utilizzo delle risorse dei processi di autovacuum può aiutare a evitare che consumino troppa CPU o I/O del disco, il che potrebbe influire sulle prestazioni di altre query eseguite sullo stesso sistema. Il parametro specifica un limite di costo, ossia un'unità di lavoro che il lavoratore può svolgere prima di dover fare una pausa e verificare se è ancora al di sotto del limite. Ad esempio, se il parametro è impostato su 2.000, un lavoratore può elaborare 2.000 unità di lavoro prima della pausa.

È possibile impostare il `autovacuum_vacuum_cost_limit` parametro utilizzando il `SET` comando in una sessione PostgreSQL o utilizzare un comando. AWS CLI

AWS CLI sintassi

Il comando seguente viene modificato `autovacuum_vacuum_cost_limit` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: `GREATEST({log(DBInstanceClassMemory/21474836480)*600},200)` unità di lavoro

Se si imposta un valore `autovacuum_vacuum_cost_limit` troppo alto, il processo di autovacuum potrebbe consumare troppe risorse e rallentare altre interrogazioni. Se lo impostate su un valore troppo basso, il processo di autovacuum potrebbe non recuperare spazio sufficiente, con conseguente aumento delle dimensioni della tabella nel tempo. È essenziale trovare il giusto equilibrio adatto al sistema.

Questo parametro influisce solo sul processo di aspirazione automatica, non sui comandi manuali. `VACUUM` Inoltre, si applica solo ai processi di autovacuum for ma non for. `VACUUM ANALYZE`

Ottimizzazione dei parametri di registrazione

L'ottimizzazione dei parametri di registrazione in PostgreSQL aiuta a garantire la raccolta delle informazioni corrette senza produrre log di grandi dimensioni che sovraccaricano il sistema.

L'ottimizzazione dei parametri di registrazione è fondamentale per bilanciare i dettagli dei log con le prestazioni del sistema e l'utilizzo del disco. È possibile personalizzare i seguenti parametri di registrazione per acquisire il livello di dettaglio appropriato nei registri, diagnosticare i problemi e analizzare gli incidenti in modo efficace, riducendo al minimo l'impatto sulle prestazioni del sistema e sull'utilizzo del disco.

- rds.force_autovacuum_logging
- rds.force_admin_logging_level
- log_duration
- dichiarazione_log_min_durata_
- log_error_verbosità
- log_dichiarazione
- log_statement_stats
- log_min_error_statement
- log_min_messaggi
- file_log_temp_
- log_connessioni
- log_disconnessioni

Questi parametri sono discussi più dettagliatamente nelle sezioni seguenti.

> **Warning**
>
> Le impostazioni migliori per questi parametri dipendono dalle politiche e dai requisiti di conformità dell'organizzazione. Tuttavia, l'abilitazione dei parametri di registrazione può comportare la creazione di un gran numero di registri e messaggi, che possono consumare spazio di archiviazione e influire sulle prestazioni, soprattutto per un database occupato. Si consiglia di utilizzare questi parametri con attenzione. Ad esempio, è possibile decidere

di attivarli temporaneamente per limitare un problema relativo a un'istruzione SQL dalle prestazioni lente e disattivarli al termine del periodo di monitoraggio.

rds.force_autovacuum_logging

Il `rds.force_autovacuum_logging` parametro (disponibile solo in Amazon RDS for PostgreSQL) controlla se le azioni di autovacuum vengono registrate nel registro del server. I suoi valori sono`disabled`,,,,`debug5`,,`debug4`,,`debug3`,`debug2`,`debug1`. `info` `notice` `warning` `error` `log` `fatal` `panic` Il valore predefinito è `warning`.

Quando abilitate`rds.force_autovacuum_logging`, vengono registrate tutte le azioni del processo di aspirazione automatica, ad esempio quando il processo inizia, quando termina e quante righe aspira. Ciò è utile per il debug o la risoluzione dei problemi di prestazioni dell'autovacuum.

AWS CLI sintassi

Il comando seguente viene modificato `rds.force_autovacuum_logging` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: `warning`

Esempio

È possibile utilizzare il `rds.force_autovacuum_logging` parametro per analizzare le prestazioni dell'autovacuum su una tabella con una velocità di scrittura molto elevata. Ad esempio, se la tabella riceve un numero elevato di operazioni di scrittura ed eliminazione al secondo e le prestazioni sono lente, è possibile abilitare il parametro per registrare gli orari di inizio e fine di ogni esecuzione dell'autovacuum e per determinare quante righe sono state cancellate. Ciò può fornire informazioni preziose sulla frequenza di funzionamento dell'aspirapolvere automatico, sul tempo necessario per farlo funzionare e su quante righe aspira. È quindi possibile utilizzare queste informazioni per ottimizzare le impostazioni dell'aspirapolvere automatico, ad esempio, e per ottimizzare le prestazioni. `autovacuum_vacuum_scale_factor` `autovacuum_vacuum_threshold` `autovacuum_naptime`

rds.force_admin_logging_level

Il `rds.force_admin_logging_level` parametro (disponibile solo in Amazon RDS for PostgreSQL) controlla il livello di dettaglio nei log prodotti da operazioni amministrative come il vacuuming, l'analisi e la reindicizzazione. Accetta i valori`debug5`,,,,`debug4`,,`debug3`, `debug2` `debug1` `loginfo`, `notice` e (impostazione predefinita). `warning` `error` `log` `fatal` `off` L'impostazione ottimale dipende dal caso d'uso. Ad esempio, se state risolvendo un problema, potreste voler impostare il parametro su un livello di debug. Altrimenti, puoi usare l'`warning`impostazione `loginfo`, o.

Se si imposta su `rds.force_admin_logging_level`debug1, è possibile registrare informazioni dettagliate per un'operazione di reindicizzazione, ad esempio l'ora di inizio e di fine, il numero di righe elaborate ed eventuali errori o avvisi che si verificano durante il processo. Ciò può fornire informazioni preziose sulle prestazioni del processo di reindicizzazione e aiutare a risolvere eventuali problemi che si verificano.

AWS CLI sintassi

Il comando seguente viene modificato `rds.force_admin_logging_level` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: `off`

Esempio

È possibile `rds.force_admin_logging_level` utilizzarlo per monitorare e analizzare le prestazioni delle operazioni amministrative su più tabelle in un database di grandi dimensioni. Ad esempio, supponiamo di avere un database di grandi dimensioni con molte tabelle e di voler ottimizzare le prestazioni di queste tabelle eseguendo regolarmente operazioni di archiviazione e analisi su di esse. Impostando il `rds.force_admin_logging_level` parametro su `info` `olog`, è possibile registrare gli orari di inizio e di fine di ogni operazione e le tabelle interessate. È possibile utilizzare queste informazioni per tenere traccia delle prestazioni delle operazioni amministrative su diverse tabelle e identificare le tabelle che potrebbero richiedere una manutenzione più frequente o più aggressiva.

Alcuni livelli di registrazione generano un gran numero di file di registro e messaggi che possono occupare rapidamente lo spazio su disco, soprattutto se il database è occupato. Si consiglia di utilizzare questo parametro con attenzione e di disattivarlo al termine del periodo di monitoraggio.

log_duration

Il `log_duration` parametro controlla se la durata di ogni query (ovvero il tempo necessario per l'esecuzione) viene registrata con la query. Quando si imposta questo parametro suon, il tempo necessario per eseguire ogni query viene incluso nell'output del registro insieme al testo della query. Il tempo viene misurato in millisecondi.

Il principale caso d'uso del `log_duration` parametro è quello di facilitare l'ottimizzazione delle prestazioni e la risoluzione dei problemi. Registrando la durata di ogni query, è possibile identificare le query che impiegano più tempo a essere eseguite e quindi concentrare gli sforzi sull'ottimizzazione di tali query. In questo modo è possibile identificare e correggere i problemi di prestazioni e migliorare le prestazioni complessive del database.

AWS CLI sintassi

Il comando seguente viene modificato `log_duration` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: `off`

Esempio

È possibile utilizzare questo parametro se si sospetta che una query o un insieme di query specifici causino problemi di prestazioni. Abilitando il `log_duration` parametro ed esaminando l'output del registro, è possibile vedere quali query richiedono più tempo per l'esecuzione e quindi intraprendere le azioni appropriate, ad esempio ottimizzare gli indici, aggiungere nuovi indici o riscrivere la query.

L'attivazione può aumentare il volume dell'output del registro. `log_duration` Si consiglia di utilizzarlo solo quando necessario e di disattivarlo durante le operazioni standard per evitare di riempire lo spazio di archiviazione o rendere i log difficili da leggere.

log_min_duration_statement

Il `log_min_duration_statement` parametro controlla il periodo di tempo minimo, in millisecondi, di esecuzione di un'istruzione SQL prima di essere registrata.

Questo parametro consente di identificare le query di lunga durata che potrebbero causare problemi di prestazioni. È possibile impostarlo su un valore di soglia (un runtime considerato troppo lungo per un carico di lavoro specifico) per acquisire le query che superano tale soglia e identificare potenziali

rallentamenti prestazionali. Per un esempio di utilizzo, vedi Utilizzo dei parametri di registrazione per acquisire variabili di associazione più avanti in questa guida.

AWS CLI sintassi

Il comando seguente viene modificato `log_min_duration_statement` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: 1 (disabilitato, che è l'impostazione predefinita del motore PostgreSQL)

Esempio

Il comando seguente registra qualsiasi istruzione che impiega più di 100 millisecondi per essere eseguita:

```
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

Il `log_error_verbosity` parametro controlla il livello di dettaglio incluso nell'output del registro per errori e messaggi registrati con un livello di errore o superiore. Questo parametro può assumere uno dei tre valori seguenti: `tersedefault`, `overbose`.

- `terse`include solo il testo del messaggio, il livello di errore e il numero di file e di riga in cui si è verificato l'errore.
- `default`include il testo del messaggio, il livello di errore, il numero di file e di riga e il contesto dell'errore.
- `verbose`include il testo del messaggio, il livello di errore, il numero di file e di riga, il contesto dell'errore e il messaggio di errore completo.

Imposta il parametro per `verbose` ottenere le informazioni più dettagliate per la risoluzione dei problemi e il debug in un ambiente non di produzione. In un ambiente di produzione, potresti volerlo impostare in `terse` modo che fornisca solo le informazioni essenziali e non riempia l'archiviazione dei log con troppi dettagli. `default`

AWS CLI sintassi

Il comando seguente viene modificato `log_error_verbosity` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: `default`

log_statement

Il `log_statement` parametro controlla quali istruzioni SQL vengono registrate nel log del server. Il parametro può assumere uno dei seguenti valori:

- `none`(impostazione predefinita) non registra alcuna istruzione
- `ddl`registra solo le istruzioni DDL (Data Definition Language) come e `CREATE TABLE ALTER TABLE`
- `mod`registra solo le istruzioni che modificano i dati come, e `INSERT UPDATE DELETE`
- `all`registra tutte le istruzioni SQL

È possibile utilizzare il `log_statement` parametro per controllare la quantità di informazioni scritte nel registro documentando solo i tipi specifici di istruzioni pertinenti al caso d'uso.

AWS CLI sintassi

Il comando seguente viene modificato `log_statement` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: `none`

Esempio

In un ambiente di produzione, è possibile impostare in modo `log_statement` da `ddl` registrare solo le istruzioni DDL e tenere traccia delle modifiche apportate allo schema del database. In un ambiente di sviluppo, potresti voler impostare il parametro su per registrare tutte le istruzioni `all` per facilitare il debug e la risoluzione dei problemi. Per un altro caso d'uso di esempio, vedi Utilizzo dei parametri di registrazione per acquisire variabili di associazione più avanti in questa guida.

L'attivazione `log_statement` può aumentare il volume dell'output dei log, quindi usatelo solo quando necessario e disattivatelo per evitare di riempire lo spazio di archiviazione o rendere i log difficili da leggere.

Si consiglia di monitorare il sistema e regolare il valore di questo parametro per ottenere il giusto equilibrio tra la quantità di informazioni registrate e l'archiviazione e le prestazioni del sistema.

log_statement_stats

Il `log_statement_stats` parametro controlla se le statistiche associate all'esecuzione di un'istruzione SQL vengono registrate con l'istruzione. Quando si attiva questo parametro, nell'output del registro vengono incluse statistiche quali il numero di righe interessate, il numero di blocchi di disco letti e scritti e il tempo necessario per eseguire l'istruzione.

È possibile utilizzare il `log_statement_stats` parametro per raccogliere informazioni aggiuntive sulle prestazioni delle singole istruzioni e sul carico di lavoro complessivo. Registrando le statistiche sulle istruzioni, è possibile identificare i modelli nelle prestazioni delle query e nell'utilizzo delle risorse e utilizzare tali informazioni per ottimizzare il database e migliorare le prestazioni complessive.

AWS CLI sintassi

Il comando seguente viene modificato `log_statement_stats` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: `off` (impostazione predefinita del motore PostgreSQL); usa 0 o 1 (booleano) per impostare i gruppi di parametri

Esempio

È possibile utilizzarlo `log_statement_stats` per analizzare il comportamento di una query specifica, vedere come utilizza risorse come CPU, memoria e I/O del disco e identificare se la query può essere ottimizzata. È inoltre possibile utilizzare questo parametro per verificare se una tabella specifica viene letta frequentemente (il che potrebbe indicare la necessità di creare un indice su una colonna specifica) o se una tabella viene scansionata troppo spesso.

L'attivazione `log_statement_stats` può aumentare il volume dell'output dei log, quindi utilizzatelo solo quando necessario e disattivatelo per evitare di riempire lo spazio di archiviazione o rendere i log difficili da leggere.

log_min_error_statement

Il `log_min_error_statement` parametro controlla quali istruzioni SQL che generano un errore verranno registrate. I suoi valori sono `debug5debug4`,`debug3`,`debug2`,`debug1`,`info`,`notice`,`warning`,`error`, `logfatal`, `epanic`. Queste impostazioni controllano la quantità di informazioni scritte nel registro in modo da poter filtrare i messaggi con un livello di gravità inferiore. È possibile impostare questo parametro su un livello di gravità più elevato per ridurre la quantità di output del registro e trovare più facilmente i messaggi importanti.

AWS CLI sintassi

Il comando seguente viene modificato `log_min_error_statement` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: `error` (impostazione predefinita del motore PostgreSQL)

Esempio

È consigliabile utilizzarlo `log_min_error_statement` quando si risolve un problema specifico e si desidera visualizzare i messaggi di errore provenienti da istruzioni SQL che causano errori.

log_min_messages

Il `log_min_messages` parametro controlla il livello di gravità scritto nel registro. È possibile impostare il parametro su `debug5`,`debug4`,`debug3`,`debug2`,`debug1`,`info`,`notice`, `warning`,`error`,`log`,`fatal`, o`panic`. Queste impostazioni controllano la quantità di informazioni scritte nel registro in modo da poter filtrare i messaggi con un livello di gravità inferiore. È possibile impostare questo parametro su un livello di gravità più elevato per ridurre la quantità di output del registro e trovare più facilmente i messaggi importanti.

AWS CLI sintassi

Il comando seguente viene modificato `log_min_messages` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: `notice`

Esempio

Se stai risolvendo un problema specifico e desideri visualizzare tutti i messaggi di errore, puoi impostare questo parametro su per `error` registrare solo gli errori e i problemi di gravità superiore. Se sei interessato a monitorare le prestazioni del sistema, puoi impostare questo parametro `info` per visualizzare informazioni più dettagliate come la durata e le statistiche di ogni istruzione.

L'impostazione `log_min_messages` di un livello di gravità più elevato riduce il volume dei log. Si consiglia di ottimizzare questo parametro in base al caso d'uso specifico, alla dimensione del registro che si desidera controllare e alla quantità di spazio su disco disponibile.

log_temp_files

Il `log_temp_files` parametro controlla la registrazione dei nomi e delle dimensioni dei file temporanei. Si applica ai file temporanei creati per scopi quali ordinamenti, hash e risultati di query temporanee. Quando questo parametro è abilitato, al momento dell'eliminazione viene generata una voce di registro per ogni file temporaneo, inclusa la dimensione del file, in byte. È possibile impostare questo parametro su 0 (zero) per la registrazione completa di tutte le informazioni sui file temporanei o su un valore positivo per i file di registro che superano tale dimensione (in kilobyte, se non sono specificate unità). Questo può essere utile per identificare e risolvere problemi di prestazioni o altri problemi relativi all'archiviazione temporanea. Per impostazione predefinita, la registrazione dei file temporanei è disattivata.

AWS CLI sintassi

Il comando seguente viene modificato `log_temp_files` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: -1 (impostazione predefinita del motore PostgreSQL)

Esempio

È possibile abilitare questo parametro se si sospetta che il sistema stia utilizzando troppa memoria temporanea o che i file temporanei non vengano eliminati correttamente. Quando si esamina l'output del registro, è possibile visualizzare le query o le operazioni che generano i file temporanei e il modo in cui questi file vengono utilizzati.

Alcune query o operazioni creano un gran numero di file temporanei, pertanto l'attivazione `log_temp_files` potrebbe influire sulle prestazioni complessive del sistema.

log_connections

Il `log_connections` parametro controlla se le connessioni al database vengono registrate. Quando si imposta questo parametro suon, il registro contiene informazioni su ogni connessione riuscita al database, ad esempio l'indirizzo IP del client, il nome utente, il nome del database e la data e l'ora della connessione.

È possibile utilizzare il `log_connections` parametro per monitorare e risolvere i problemi di connessione al database. Puoi vedere gli utenti, le applicazioni, i terminali e i bot che si connettono al database, da dove si connettono e con quale frequenza. Queste informazioni possono essere utili per identificare e risolvere problemi relativi alla connessione o tenere traccia dei modelli di utilizzo.

AWS CLI sintassi

Il comando seguente viene modificato `log_connections` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: `off` (impostazione predefinita del motore PostgreSQL)

Esempio

Puoi utilizzare questo parametro se sospetti che troppe connessioni al database o che un utente o un indirizzo IP specifico che si connette troppo frequentemente influiscano sulle prestazioni. Abilitando il `log_connections` parametro ed esaminando l'output del registro, è possibile visualizzare il numero e i dettagli di tutte le connessioni.

Prima di abilitare questo parametro, controllate le politiche della vostra organizzazione e considerate le implicazioni in termini di sicurezza della registrazione di indirizzi IP e nomi utente.

log_disconnections

Il `log_disconnections` parametro controlla la registrazione delle disconnessioni dal database. Quando si imposta questo parametro suon, vengono registrate le informazioni sulla fine di ogni sessione, come l'indirizzo IP del client, il nome utente, il nome del database e la data e l'ora della disconnessione.

È possibile utilizzare il `log_disconnections` parametro per monitorare e risolvere i problemi relativi alle terminazioni delle sessioni del database. È possibile visualizzare gli utenti, le applicazioni, i terminali e i bot che si disconnettono dal database, quando e perché. Ad esempio, è possibile esaminare le interruzioni impreviste, ad esempio un arresto anomalo o le disconnessioni avviate dall'amministratore. Queste informazioni possono essere utili per identificare e risolvere problemi relativi alle disconnessioni o al monitoraggio dei modelli di utilizzo.

AWS CLI sintassi

Il comando seguente viene modificato `log_disconnections` per uno specifico gruppo di parametri DB. Questa modifica si applica a tutte le istanze o i cluster che utilizzano il gruppo di parametri.

```
# 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: Dinamico (le modifiche vengono applicate immediatamente se impostate) `ApplyMethod=immediate`

Valore predefinito: `off` (impostazione predefinita del motore PostgreSQL)

Esempio

Potresti utilizzarlo `log_disconnections` se sospetti che ci siano troppi utenti che si disconnettono dal database o che uno specifico utente o indirizzo IP si disconnetta troppo frequentemente. Abilitando il `log_disconnections` parametro ed esaminando l'output del registro, è possibile visualizzare il numero e i dettagli di tutte le disconnessioni, incluso chi, quando e se sono stati generati errori prima della disconnessione.

Prima di abilitare questo parametro, verificate le politiche della vostra organizzazione e considerate le implicazioni in termini di sicurezza della registrazione di indirizzi IP e nomi utente.

Utilizzo dei parametri di registrazione per acquisire le variabili di associazione

Un tipico caso d'uso per l'acquisizione di variabili di associazione in PostgreSQL è il debug e l'ottimizzazione delle prestazioni delle query SQL. Una variabile bind consente di passare dati a una query quando la si esegue. Acquisendo le variabili di associazione, è possibile visualizzare i dati di input passati a una query, il che può aiutare a identificare eventuali problemi relativi ai dati o alle prestazioni delle query. L'acquisizione delle variabili di associazione può inoltre aiutare a controllare i dati di input e a rilevare potenziali rischi per la sicurezza o attività dannose.

Esistono diversi modi per acquisire variabili di associazione per PostgreSQL. Un metodo consiste nell'abilitare i parametri and. `debug_print_parse` `debug_print_rewritten` Ciò fa sì che

PostgreSQL invii le versioni analizzate e riscritte delle istruzioni SQL, insieme alle variabili associate, al log del server.

- `debug_print_parse`: Quando abiliti questo parametro, l'albero di analisi delle query in entrata viene stampato nel log del server. Questo può essere utile per comprendere la struttura di una query e i valori di qualsiasi parametro associato.
- `debug_print_rewritten`: Quando si abilita questo parametro, i moduli riscritti delle query in entrata vengono stampati nel registro del server. Ciò può essere utile per comprendere come il pianificatore di query interpreta una query e i valori di qualsiasi parametro associato.

Puoi utilizzare due parametri aggiuntivi in Amazon RDS e Aurora per acquisire variabili di associazione nei tuoi database PostgreSQL:

- `log_min_duration_statement`: Questo parametro imposta la durata minima di un'istruzione prima che venga registrata, in millisecondi. Quando un'istruzione richiede più tempo della durata specificata, i relativi valori di associazione vengono inclusi nell'output del registro.
- `log_statement`: Questo parametro controlla quali istruzioni SQL vengono registrate. Imposta questo parametro su `all` o associa per includere i valori associati nel registro. L'aumento del livello di registrazione influisce sulle prestazioni, quindi consigliamo di annullare le modifiche dopo la risoluzione dei problemi.

È inoltre possibile utilizzare l'`pg_stat_statements`estensione, che fornisce statistiche sulle prestazioni per tutte le istruzioni SQL eseguite da un server, incluso il testo della query e i valori associati. Questa estensione consente di utilizzare pGAdmin o strumenti simili per monitorare e analizzare le prestazioni delle query.

Un'altra opzione consiste nell'utilizzare la `pg_bind_parameter_status()` funzione per ottenere i valori dei parametri associati da un'istruzione preparata o nell'utilizzare la `pg_get_parameter_status (paramname)` funzione per recuperare lo stato o il valore di un parametro di runtime specifico.

Inoltre, puoi utilizzare strumenti di terze parti come pgBadger per analizzare i log di PostgreSQL ed estrarre le variabili di associazione e altre informazioni per ulteriori analisi.

Ottimizzazione dei parametri di replica

In PostgreSQL, è possibile replicare le modifiche ai dati da un database PostgreSQL a un altro utilizzando la replica logica anziché la replica fisica basata su file. La replica logica utilizza il write-ahead log (WAL) per acquisire le modifiche e supporta la replica di tabelle selezionate o di interi database.

Amazon RDS for PostgreSQL e Aurora, compatibile con PostgreSQL, supportano entrambi la replica logica, quindi puoi configurare un'architettura di database altamente disponibile e scalabile in grado di gestire il traffico di lettura e scrittura da più fonti. Questi servizi utilizzano pglogical, un'estensione PostgreSQL open source, per implementare la replica logica.

L'ottimizzazione della replica logica in Aurora e Amazon RDS è importante per ottenere prestazioni, scalabilità e disponibilità ottimali. È possibile ottimizzare i parametri nell'estensione pglogical per gestire le prestazioni della replica logica. Ad esempio, puoi:

- Migliora le prestazioni della replica aumentando il numero di processi di lavoro o regolando la loro allocazione di memoria.
- Riduci il rischio di ritardi nella replica regolando la frequenza di sincronizzazione tra il database di origine e quello di replica.
- Ottimizza l'uso delle risorse regolando l'allocazione della memoria e della CPU dei processi di lavoro.
- Assicuratevi che il processo di replica non abbia un impatto indebito sulle prestazioni del database di origine.

Puoi utilizzare i seguenti parametri in Aurora e Amazon RDS per controllare e configurare la replica logica:

- `max_replication_slots`imposta il numero massimo di slot di replica che possono essere creati sul server. Uno slot di replica è una prenotazione denominata e persistente per una connessione di replica per inviare dati WAL a una replica.
- `max_wal_senders`imposta il numero massimo di processi mittenti WAL connessi contemporaneamente. I processi di invio WAL vengono utilizzati per trasmettere il WAL dal server primario alla replica.
- `wal_sender_timeout`imposta il tempo massimo, in millisecondi, in cui un mittente WAL attende una risposta dalla replica prima di arrendersi e riconnettersi.

- `wal_receiver_timeout`imposta il tempo massimo, in millisecondi, durante il quale una replica attende i dati WAL dal database primario prima del timeout.
- `log_replication_commands`, se impostato su, esegue le istruzioni SQL relative alla on replica.

Quando si abilita il `rds.logical_replication` parametro (impostandolo su 1), il `wal_level` parametro viene impostato su`logical`, il che significa che tutte le modifiche apportate al database vengono scritte nel WAL in un formato che può essere letto e applicato a una replica. Questa impostazione è necessaria per abilitare la replica logica. Questa impostazione consente anche la replica delle `SELECT` istruzioni.

L'impostazione `wal_level` su `logical` può aumentare la quantità di dati scritti sul WAL e quindi su disco, il che può influire sulle prestazioni del sistema. Si consiglia di considerare lo spazio disponibile su disco e le prestazioni del sistema quando si abilita la replica logica.

Esempio

Si desidera replicare i dati dal database principale a un database secondario per scopi di backup e disaster recovery. Tuttavia, il database secondario presenta un volume elevato di operazioni di lettura, quindi è necessario assicurarsi che il processo di replica sia il più rapido ed efficiente possibile senza compromettere l'integrità dei dati.

I valori predefiniti per la replica logica in Amazon RDS e Aurora danno priorità alla coerenza rispetto alle prestazioni, quindi potrebbero non essere ottimali per questo caso d'uso. Per ottimizzare le impostazioni di replica logica in termini di velocità ed efficienza, puoi personalizzare i parametri come segue:

- `max_replication_slots`Passa da 10 (impostazione predefinita per Amazon RDS) o 20 (impostazione predefinita per Aurora) a 30 per soddisfare le potenziali esigenze future di crescita e replica.
- Passa `max_wal_senders` da 10 (impostazione predefinita) a 20 per garantire che ci siano abbastanza processi di invio WAL per tenere il passo con la domanda di replica.
- Riduci `wal_sender_timeout` da 30 secondi (impostazione predefinita) a 15 secondi per garantire che i processi di invio WAL inattivi vengano terminati più rapidamente, il che libera risorse per la replica attiva.

- Riduci `wal_receiver_timeout` da 30 secondi (impostazione predefinita) a 15 secondi per garantire che i processi di ricezione WAL inattivi vengano terminati più rapidamente, il che libera risorse per la replica attiva.
- Aumentate `max_logical_replication_workers` da 4 (impostazione predefinita) a 8 per garantire che vi siano abbastanza processi di replica logica per stare al passo con la domanda di replica.

Queste ottimizzazioni forniscono una replica dei dati più rapida ed efficiente, mantenendo al contempo l'integrità e la sicurezza dei dati.

Ad esempio, se si verificasse un disastro e il database primario diventasse non disponibile, il database secondario disporrebbe già dei dati più recenti grazie al processo di replica ottimizzato. Ciò consentirebbe alle operazioni aziendali di continuare a fornire servizi critici senza interruzioni.

Best practice

Ottimizzare la replica logica con enormi carichi di lavoro può essere un'attività complessa che dipende da diversi fattori, tra cui la dimensione del set di dati, il numero di tabelle da replicare, il numero di repliche e le risorse disponibili. Ecco alcuni suggerimenti generali per ottimizzare la replica logica con carichi di lavoro enormi:

- Monitora il ritardo di replica. Il ritardo di replica è la differenza di fuso orario tra il server primario e i server di standby. Il monitoraggio del ritardo nella replica può aiutare a identificare potenziali rallentamenti e ad adottare misure per migliorare le prestazioni di replica. È possibile utilizzare la `pg_current_wal_lsn()` funzione per controllare l'attuale ritardo di replica.
- Ottimizza le impostazioni WAL. L'`pg_logical`estensione utilizza WAL per trasmettere le modifiche dal server primario al server di standby. Se le impostazioni WAL non sono regolate correttamente, la replica può diventare lenta e inaffidabile. Assicurati di impostare i `max_replication_slots` parametri `max_wal_senders` and su valori adeguati, a seconda dei tuoi carichi di lavoro.
- Adotta una strategia di indicizzazione. La presenza di indici adeguati sul server primario può contribuire a migliorare le prestazioni della replica logica, ridurre l'I/O sul server primario e ridurre il carico sul sistema.
- Usa la replica parallela. L'utilizzo della replica parallela può contribuire ad aumentare la velocità di replica consentendo a più processi di lavoro paralleli di replicare i dati. Questa funzionalità è disponibile su PostgreSQL 12 e versioni successive.

Passaggi successivi

Dopo aver ottimizzato la memoria, la replica, l'autovacuum e i parametri di registrazione per il tuo database compatibile con Amazon RDS for PostgreSQL o Aurora PostgreSQL, prendi in considerazione questi passaggi per migliorare ulteriormente le prestazioni del tuo database:

- Monitora il tuo database. Tieni traccia delle prestazioni del tuo database nel tempo utilizzando strumenti di monitoraggio integrati o soluzioni di terze parti. Monitora i principali parametri prestazionali come l'utilizzo della CPU, l'I/O del disco, l'utilizzo della memoria e i tempi di esecuzione delle query per identificare potenziali colli di bottiglia e aree di miglioramento.
- Ottimizza continuamente i parametri. Man mano che il carico di lavoro si evolve, continua a monitorare e modificare i parametri del database per garantire prestazioni ottimali. Controlla regolarmente i registri di sistema, i messaggi di errore e le metriche delle prestazioni per identificare nuove opportunità di ottimizzazione.
- Implementa il caching. Utilizza la memorizzazione nella cache per ridurre il numero di query che colpiscono il database. Puoi implementare la memorizzazione nella cache a livello di applicazione utilizzando strumenti come Memcached o Redis oppure puoi utilizzare Amazon ElastiCache per fornire una cache in memoria per il tuo database.
- Ottimizza le tue domande. Le query mal progettate possono influire in modo significativo sulle prestazioni del database. Utilizza `EXPLAIN` altri strumenti di ottimizzazione delle query per identificare le query lente, ottimizzarle ed eliminare quelle non necessarie.

Seguendo queste linee guida, puoi ottimizzare le prestazioni del tuo database Aurora o Amazon RDS for PostgreSQL e assicurarti che soddisfi le esigenze della tua applicazione con prestazioni migliorate del database, maggiore affidabilità, tempi di inattività ridotti, maggiore sicurezza e risparmi sui costi. Ottimizzando i parametri di configurazione in base al carico di lavoro, è possibile garantire che il database funzioni in modo efficiente e utilizzi le risorse in modo efficace, garantendo prestazioni migliori e un'applicazione più reattiva. Inoltre, i parametri configurati correttamente possono ridurre la probabilità di errori e vulnerabilità, con conseguente maggiore affidabilità e migliore sicurezza. Ciò può tradursi in risparmi sui costi in termini di riduzione della manutenzione e dei tempi di inattività, nonché in una migliore esperienza e soddisfazione complessive degli utenti.

Risorse

- Parametri PostgreSQL di Amazon Aurora, parte 1: gestione della memoria e del piano di query (post sul blog)AWS
- Parametri PostgreSQL di Amazon Aurora, parte 2: Replica, sicurezza e registrazione (post sul blog)AWS
- Parametri PostgreSQL di Amazon Aurora, parte 3: parametri dell'ottimizzatore (post sul blog)AWS
- Parametri PostgreSQL di Amazon Aurora, parte 4: opzioni di compatibilità ANSI (post sul blog)AWS
- Lavorare con Amazon Aurora PostgreSQL (documentazione)AWS
- Utilizzo di Amazon RDS per PostgreSQL (documentazione)AWS
- Monitoraggio del carico del DB con Performance Insights su Amazon RDS (AWS documentazione)
- Utilizzo dei CloudWatch parametri di Amazon (AWS documentazione)
- pg_stats_statements (documentazione PostgreSQL)

Cronologia dei documenti

La tabella seguente descrive le modifiche significative apportate a questa guida. Per ricevere notifiche sugli aggiornamenti futuri, puoi abbonarti a un feed RSS.

Modifica	Descrizione	Data
Informazioni aggiornate sulla memoria e sui parametri dell'autovacuum	È stata aggiornata la descrizione del parametro random_page_cost; sono state aggiunte le unità mancanti ai valori predefiniti per i parametri memory e autovacuum; è stata aggiornata la sintassi per il parametro max_connections. AWS CLI	27 febbraio 2024
Informazioni aggiornate su `autovacuum`	È stata corretta l'impostazione predefinita dell'autovacuum (abilitata).	27 dicembre 2023
Informazioni aggiornate su `max_connections`	È stata aggiornata la sezione max_connections con nuove indicazioni sull'ottimizzazione di questo parametro.	15 novembre 2023
Pubblicazione iniziale	—	31 ottobre 2023

AWS Glossario delle linee guida prescrittive

I seguenti sono termini comunemente usati nelle strategie, nelle guide e nei modelli forniti da AWS Prescriptive Guidance. Per suggerire voci, utilizza il link Fornisci feedback alla fine del glossario.

Numeri

7 R

Sette strategie di migrazione comuni per trasferire le applicazioni sul cloud. Queste strategie si basano sulle 5 R identificate da Gartner nel 2011 e sono le seguenti:

- Rifattorizzare/riprogettare: trasferisci un'applicazione e modifica la sua architettura sfruttando appieno le funzionalità native del cloud per migliorare l'agilità, le prestazioni e la scalabilità. Ciò comporta in genere la portabilità del sistema operativo e del database. Esempio: migra il tuo database Oracle locale all'edizione compatibile con Amazon Aurora PostgreSQL.
- Ridefinire la piattaforma (lift and reshape): trasferisci un'applicazione nel cloud e introduci un certo livello di ottimizzazione per sfruttare le funzionalità del cloud. Esempio: migra il tuo database Oracle locale ad Amazon Relational Database Service (Amazon RDS) per Oracle in. Cloud AWS
- Riacquistare (drop and shop): passa a un prodotto diverso, in genere effettuando la transizione da una licenza tradizionale a un modello SaaS. Esempio: migra il tuo sistema di gestione delle relazioni con i clienti (CRM) su Salesforce.com.
- Eseguire il rehosting (lift and shift): trasferisci un'applicazione sul cloud senza apportare modifiche per sfruttare le funzionalità del cloud. Esempio: migra il tuo database Oracle locale a Oracle su un'istanza EC2 in. Cloud AWS
- Trasferire (eseguire il rehosting a livello hypervisor): trasferisci l'infrastruttura sul cloud senza acquistare nuovo hardware, riscrivere le applicazioni o modificare le operazioni esistenti. Esegui la migrazione dei server da una piattaforma locale a un servizio cloud per la stessa piattaforma. Esempio: migra un'applicazione suMicrosoft Hyper-V. AWS
- Riesaminare (mantenere): mantieni le applicazioni nell'ambiente di origine. Queste potrebbero includere applicazioni che richiedono una rifattorizzazione significativa che desideri rimandare a un momento successivo e applicazioni legacy che desideri mantenere, perché non vi è alcuna giustificazione aziendale per effettuarne la migrazione.
- Ritirare: disattiva o rimuovi le applicazioni che non sono più necessarie nell'ambiente di origine.

A

ABAC

Vedi controllo degli accessi basato sugli attributi.

servizi astratti

Vedi servizi gestiti.

ACIDO

Vedi atomicità, consistenza, isolamento, durata.

migrazione attiva-attiva

Un metodo di migrazione del database in cui i database di origine e di destinazione vengono mantenuti sincronizzati (utilizzando uno strumento di replica bidirezionale o operazioni di doppia scrittura) ed entrambi i database gestiscono le transazioni provenienti dalle applicazioni di connessione durante la migrazione. Questo metodo supporta la migrazione in piccoli batch controllati anziché richiedere una conversione una tantum. È più flessibile ma richiede più lavoro rispetto alla migrazione attiva-passiva.

migrazione attiva-passiva

Un metodo di migrazione di database in cui i database di origine e di destinazione vengono mantenuti sincronizzati, ma solo il database di origine gestisce le transazioni provenienti dalle applicazioni di connessione mentre i dati vengono replicati nel database di destinazione. Il database di destinazione non accetta alcuna transazione durante la migrazione.

funzione aggregata

Una funzione SQL che opera su un gruppo di righe e calcola un singolo valore restituito per il gruppo. Esempi di funzioni aggregate includono `SUM` e. `MAX`

Intelligenza artificiale

Vedi intelligenza artificiale.

AIOps

Guarda le operazioni di intelligenza artificiale.

anonimizzazione

Il processo di eliminazione permanente delle informazioni personali in un set di dati. L'anonimizzazione può aiutare a proteggere la privacy personale. I dati anonimi non sono più considerati dati personali.

anti-modello

Una soluzione utilizzata frequentemente per un problema ricorrente in cui la soluzione è controproducente, inefficace o meno efficace di un'alternativa.

controllo delle applicazioni

Un approccio alla sicurezza che consente l'uso solo di applicazioni approvate per proteggere un sistema dal malware.

portfolio di applicazioni

Una raccolta di informazioni dettagliate su ogni applicazione utilizzata da un'organizzazione, compresi i costi di creazione e manutenzione dell'applicazione e il relativo valore aziendale. Queste informazioni sono fondamentali per il processo di scoperta e analisi del portfolio e aiutano a identificare e ad assegnare la priorità alle applicazioni da migrare, modernizzare e ottimizzare.

intelligenza artificiale (IA)

Il campo dell'informatica dedicato all'uso delle tecnologie informatiche per svolgere funzioni cognitive tipicamente associate agli esseri umani, come l'apprendimento, la risoluzione di problemi e il riconoscimento di schemi. Per ulteriori informazioni, consulta la sezione Che cos'è l'intelligenza artificiale?

operazioni di intelligenza artificiale (AIOps)

Il processo di utilizzo delle tecniche di machine learning per risolvere problemi operativi, ridurre gli incidenti operativi e l'intervento umano e aumentare la qualità del servizio. Per ulteriori informazioni su come viene utilizzato AIOps nella strategia di migrazione AWS , consulta la guida all'integrazione delle operazioni.

crittografia asimmetrica

Un algoritmo di crittografia che utilizza una coppia di chiavi, una chiave pubblica per la crittografia e una chiave privata per la decrittografia. Puoi condividere la chiave pubblica perché non viene utilizzata per la decrittografia, ma l'accesso alla chiave privata deve essere altamente limitato.

atomicità, consistenza, isolamento, durabilità (ACID)

Un insieme di proprietà del software che garantiscono la validità dei dati e l'affidabilità operativa di un database, anche in caso di errori, interruzioni di corrente o altri problemi.

Controllo degli accessi basato su attributi (ABAC)

La pratica di creare autorizzazioni dettagliate basate su attributi utente, come reparto, ruolo professionale e nome del team. Per ulteriori informazioni, consulta ABAC for AWS nella documentazione AWS Identity and Access Management (IAM).

fonte di dati autorevole

Una posizione in cui è archiviata la versione principale dei dati, considerata la fonte di informazioni più affidabile. È possibile copiare i dati dalla fonte di dati autorevole in altre posizioni allo scopo di elaborarli o modificarli, ad esempio anonimizzandoli, oscurandoli o pseudonimizzandoli.

Zona di disponibilità

Una posizione distinta all'interno di un edificio Regione AWS che è isolata dai guasti in altre zone di disponibilità e offre una connettività di rete economica e a bassa latenza verso altre zone di disponibilità nella stessa regione.

AWS Cloud Adoption Framework (CAF)AWS

Un framework di linee guida e best practice AWS per aiutare le organizzazioni a sviluppare un piano efficiente ed efficace per passare con successo al cloud. AWS CAF organizza le linee guida in sei aree di interesse chiamate prospettive: business, persone, governance, piattaforma, sicurezza e operazioni. Le prospettive relative ad azienda, persone e governance si concentrano sulle competenze e sui processi aziendali; le prospettive relative alla piattaforma, alla sicurezza e alle operazioni si concentrano sulle competenze e sui processi tecnici. Ad esempio, la prospettiva relativa alle persone si rivolge alle parti interessate che gestiscono le risorse umane (HR), le funzioni del personale e la gestione del personale. In questa prospettiva, AWS CAF fornisce linee guida per lo sviluppo delle persone, la formazione e le comunicazioni per aiutare a preparare l'organizzazione all'adozione del cloud di successo. Per ulteriori informazioni, consulta il sito web di AWS CAF e il white paper AWS CAF.

AWS Workload Qualification Framework (WQF)AWS

Uno strumento che valuta i carichi di lavoro di migrazione dei database, consiglia strategie di migrazione e fornisce stime del lavoro. AWS WQF è incluso in (). AWS Schema Conversion Tool AWS SCT Analizza gli schemi di database e gli oggetti di codice, il codice dell'applicazione, le dipendenze e le caratteristiche delle prestazioni e fornisce report di valutazione.

B

bot difettoso

Un bot che ha lo scopo di disturbare o causare danni a individui o organizzazioni.

BCP

Vedi la pianificazione della continuità operativa.

grafico comportamentale

Una vista unificata, interattiva dei comportamenti delle risorse e delle interazioni nel tempo. Puoi utilizzare un grafico comportamentale con Amazon Detective per esaminare tentativi di accesso non riusciti, chiamate API sospette e azioni simili. Per ulteriori informazioni, consulta Dati in un grafico comportamentale nella documentazione di Detective.

sistema big-endian

Un sistema che memorizza per primo il byte più importante. Vedi anche endianness.

Classificazione binaria

Un processo che prevede un risultato binario (una delle due classi possibili). Ad esempio, il modello di machine learning potrebbe dover prevedere problemi come "Questa e-mail è spam o non è spam?" o "Questo prodotto è un libro o un'auto?"

filtro Bloom

Una struttura di dati probabilistica ed efficiente in termini di memoria che viene utilizzata per verificare se un elemento fa parte di un set.

distribuzioni blu/verdi

Una strategia di implementazione in cui si creano due ambienti separati ma identici. La versione corrente dell'applicazione viene eseguita in un ambiente (blu) e la nuova versione dell'applicazione nell'altro ambiente (verde). Questa strategia consente di ripristinare rapidamente il sistema con un impatto minimo.

bot

Un'applicazione software che esegue attività automatizzate su Internet e simula l'attività o l'interazione umana. Alcuni bot sono utili o utili, come i web crawler che indicizzano le informazioni su Internet. Alcuni altri bot, noti come bot dannosi, hanno lo scopo di disturbare o causare danni a individui o organizzazioni.

botnet

Reti di bot infettate da malware e controllate da un'unica parte, nota come bot herder o bot operator. Le botnet sono il meccanismo più noto per scalare i bot e il loro impatto.

ramo

Un'area contenuta di un repository di codice. Il primo ramo creato in un repository è il ramo principale. È possibile creare un nuovo ramo a partire da un ramo esistente e quindi sviluppare funzionalità o correggere bug al suo interno. Un ramo creato per sviluppare una funzionalità viene comunemente detto ramo di funzionalità. Quando la funzionalità è pronta per il rilascio, il ramo di funzionalità viene ricongiunto al ramo principale. Per ulteriori informazioni, consulta Informazioni sulle filiali (documentazione). GitHub

accesso break-glass

In circostanze eccezionali e tramite una procedura approvata, un mezzo rapido per consentire a un utente di accedere a un sito a Account AWS cui in genere non dispone delle autorizzazioni necessarie. Per ulteriori informazioni, vedere l'indicatore Implementate break-glass procedures nella guida Well-Architected AWS .

strategia brownfield

L'infrastruttura esistente nell'ambiente. Quando si adotta una strategia brownfield per un'architettura di sistema, si progetta l'architettura in base ai vincoli dei sistemi e dell'infrastruttura attuali. Per l'espansione dell'infrastruttura esistente, è possibile combinare strategie brownfield e greenfield.

cache del buffer

L'area di memoria in cui sono archiviati i dati a cui si accede con maggiore frequenza.

capacità di business

Azioni intraprese da un'azienda per generare valore (ad esempio vendite, assistenza clienti o marketing). Le architetture dei microservizi e le decisioni di sviluppo possono essere guidate dalle capacità aziendali. Per ulteriori informazioni, consulta la sezione Organizzazione in base alle funzionalità aziendali del whitepaper Esecuzione di microservizi containerizzati su AWS.

pianificazione della continuità operativa (BCP)

Un piano che affronta il potenziale impatto di un evento che comporta l'interruzione dell'attività, come una migrazione su larga scala, sulle operazioni e consente a un'azienda di riprendere rapidamente le operazioni.

C

CAF

Vedi AWS Cloud Adoption Framework.

implementazione canaria

Il rilascio lento e incrementale di una versione agli utenti finali. Quando sei sicuro, distribuisci la nuova versione e sostituisci la versione corrente nella sua interezza.

CoE

Vedi Cloud Center of Excellence.

CDC

Vedi Change Data Capture.

Change Data Capture (CDC)

Il processo di tracciamento delle modifiche a un'origine dati, ad esempio una tabella di database, e di registrazione dei metadati relativi alla modifica. È possibile utilizzare CDC per vari scopi, ad esempio il controllo o la replica delle modifiche in un sistema di destinazione per mantenere la sincronizzazione.

ingegneria del caos

Introduzione intenzionale di guasti o eventi dirompenti per testare la resilienza di un sistema. Puoi usare AWS Fault Injection Service (AWS FIS) per eseguire esperimenti che stressano i tuoi AWS carichi di lavoro e valutarne la risposta.

CI/CD

Vedi integrazione continua e distribuzione continua.

classificazione

Un processo di categorizzazione che aiuta a generare previsioni. I modelli di ML per problemi di classificazione prevedono un valore discreto. I valori discreti sono sempre distinti l'uno dall'altro. Ad esempio, un modello potrebbe dover valutare se in un'immagine è presente o meno un'auto.

crittografia lato client

Crittografia dei dati a livello locale, prima che il destinatario li Servizio AWS riceva.

centro di eccellenza del cloud (CCoE)

Un team multidisciplinare che guida le iniziative di adozione del cloud in tutta l'organizzazione, tra cui lo sviluppo di best practice per il cloud, la mobilitazione delle risorse, la definizione delle tempistiche di migrazione e la guida dell'organizzazione attraverso trasformazioni su larga scala. Per ulteriori informazioni, consulta i post di CCoE sull' Cloud AWS Enterprise Strategy Blog.

cloud computing

La tecnologia cloud generalmente utilizzata per l'archiviazione remota di dati e la gestione dei dispositivi IoT. Il cloud computing è generalmente collegato alla tecnologia di edge computing.

modello operativo cloud

In un'organizzazione IT, il modello operativo utilizzato per creare, maturare e ottimizzare uno o più ambienti cloud. Per ulteriori informazioni, consulta Building your Cloud Operating Model.

fasi di adozione del cloud

Le quattro fasi che le organizzazioni in genere attraversano quando migrano verso Cloud AWS:

- Progetto: esecuzione di alcuni progetti relativi al cloud per scopi di dimostrazione e apprendimento
- Fondamento: effettuare investimenti fondamentali per dimensionare l'adozione del cloud (ad esempio, creazione di una zona di destinazione, definizione di un CCoE, definizione di un modello operativo)
- Migrazione: migrazione di singole applicazioni
- Reinvenzione: ottimizzazione di prodotti e servizi e innovazione nel cloud

Queste fasi sono state definite da Stephen Orban nel post del blog The Journey Toward Cloud-First & the Stages of Adoption on the Enterprise Strategy. Cloud AWS Per informazioni su come si relazionano alla strategia di AWS migrazione, consulta la guida alla preparazione alla migrazione.

CMDB

Vedi database di gestione della configurazione.

repository di codice

Una posizione in cui il codice di origine e altri asset, come documentazione, esempi e script, vengono archiviati e aggiornati attraverso processi di controllo delle versioni. Gli archivi cloud più comuni includono GitHub o AWS CodeCommit. Ogni versione del codice è denominata ramo. In

una struttura a microservizi, ogni repository è dedicato a una singola funzionalità. Una singola pipeline CI/CD può utilizzare più repository.

cache fredda

Una cache del buffer vuota, non ben popolata o contenente dati obsoleti o irrilevanti. Ciò influisce sulle prestazioni perché l'istanza di database deve leggere dalla memoria o dal disco principale, il che richiede più tempo rispetto alla lettura dalla cache del buffer.

dati freddi

Dati a cui si accede raramente e che in genere sono storici. Quando si eseguono interrogazioni di questo tipo di dati, le interrogazioni lente sono in genere accettabili. Lo spostamento di questi dati su livelli o classi di storage meno costosi e con prestazioni inferiori può ridurre i costi.

visione artificiale (CV)

Un campo dell'intelligenza artificiale che utilizza l'apprendimento automatico per analizzare ed estrarre informazioni da formati visivi come immagini e video digitali. Ad esempio, AWS Panorama offre dispositivi che aggiungono CV alle reti di telecamere locali e Amazon SageMaker fornisce algoritmi di elaborazione delle immagini per CV.

deriva della configurazione

Per un carico di lavoro, una modifica della configurazione rispetto allo stato previsto. Potrebbe causare la non conformità del carico di lavoro e in genere è graduale e involontaria.

database di gestione della configurazione (CMDB)

Un repository che archivia e gestisce le informazioni su un database e il relativo ambiente IT, inclusi i componenti hardware e software e le relative configurazioni. In genere si utilizzano i dati di un CMDB nella fase di individuazione e analisi del portafoglio della migrazione.

Pacchetto di conformità

Una raccolta di AWS Config regole e azioni correttive che puoi assemblare per personalizzare i controlli di conformità e sicurezza. È possibile distribuire un pacchetto di conformità come singola entità in una regione Account AWS and o all'interno di un'organizzazione utilizzando un modello YAML. Per ulteriori informazioni, consulta i Conformance Pack nella documentazione. AWS Config

integrazione e distribuzione continua (continuous integration and continuous delivery, CI/CD)

Il processo di automazione delle fasi di origine, creazione, test, gestione temporanea e produzione del processo di rilascio del software. Il processo CI/CD è comunemente descritto come una

pipeline. CI/CD può aiutare ad automatizzare i processi, migliorare la produttività, migliorare la qualità del codice e velocizzare le distribuzioni. Per ulteriori informazioni, consulta Vantaggi della distribuzione continua. CD può anche significare continuous deployment (implementazione continua). Per ulteriori informazioni, consulta Distribuzione continua e implementazione continua a confronto.

CV

Vedi visione artificiale.

D

dati a riposo

Dati stazionari nella rete, ad esempio i dati archiviati.

classificazione dei dati

Un processo per identificare e classificare i dati nella rete in base alla loro criticità e sensibilità. È un componente fondamentale di qualsiasi strategia di gestione dei rischi di sicurezza informatica perché consente di determinare i controlli di protezione e conservazione appropriati per i dati. La classificazione dei dati è un componente del pilastro della sicurezza nel AWS Well-Architected Framework. Per ulteriori informazioni, consulta Classificazione dei dati.

deriva dei dati

Una variazione significativa tra i dati di produzione e i dati utilizzati per addestrare un modello di machine learning o una modifica significativa dei dati di input nel tempo. La deriva dei dati può ridurre la qualità, l'accuratezza e l'equità complessive nelle previsioni dei modelli ML.

dati in transito

Dati che si spostano attivamente attraverso la rete, ad esempio tra le risorse di rete.

rete di dati

Un framework architettonico che fornisce la proprietà distribuita e decentralizzata dei dati con gestione e governance centralizzate.

riduzione al minimo dei dati

Il principio della raccolta e del trattamento dei soli dati strettamente necessari. Praticare la riduzione al minimo dei dati in the Cloud AWS può ridurre i rischi per la privacy, i costi e l'impronta di carbonio delle analisi.

perimetro dei dati

Una serie di barriere preventive nell' AWS ambiente che aiutano a garantire che solo le identità attendibili accedano alle risorse attendibili delle reti previste. Per ulteriori informazioni, consulta Building a data perimeter on. AWS

pre-elaborazione dei dati

Trasformare i dati grezzi in un formato che possa essere facilmente analizzato dal modello di ML. La pre-elaborazione dei dati può comportare la rimozione di determinate colonne o righe e l'eliminazione di valori mancanti, incoerenti o duplicati.

provenienza dei dati

Il processo di tracciamento dell'origine e della cronologia dei dati durante il loro ciclo di vita, ad esempio il modo in cui i dati sono stati generati, trasmessi e archiviati.

soggetto dei dati

Un individuo i cui dati vengono raccolti ed elaborati.

data warehouse

Un sistema di gestione dei dati che supporta la business intelligence, come l'analisi. I data warehouse contengono in genere grandi quantità di dati storici e vengono generalmente utilizzati per interrogazioni e analisi.

linguaggio di definizione del database (DDL)

Istruzioni o comandi per creare o modificare la struttura di tabelle e oggetti in un database.

linguaggio di manipolazione del database (DML)

Istruzioni o comandi per modificare (inserire, aggiornare ed eliminare) informazioni in un database.

DDL

Vedi linguaggio di definizione del database.

deep ensemble

Combinare più modelli di deep learning per la previsione. È possibile utilizzare i deep ensemble per ottenere una previsione più accurata o per stimare l'incertezza nelle previsioni.

deep learning

Un sottocampo del ML che utilizza più livelli di reti neurali artificiali per identificare la mappatura tra i dati di input e le variabili target di interesse.

defense-in-depth

Un approccio alla sicurezza delle informazioni in cui una serie di meccanismi e controlli di sicurezza sono accuratamente stratificati su una rete di computer per proteggere la riservatezza, l'integrità e la disponibilità della rete e dei dati al suo interno. Quando si adotta questa strategia AWS, si aggiungono più controlli a diversi livelli della AWS Organizations struttura per proteggere le risorse. Ad esempio, un defense-in-depth approccio potrebbe combinare l'autenticazione a più fattori, la segmentazione della rete e la crittografia.

amministratore delegato

In AWS Organizations, un servizio compatibile può registrare un account AWS membro per amministrare gli account dell'organizzazione e gestire le autorizzazioni per quel servizio. Questo account è denominato amministratore delegato per quel servizio specifico. Per ulteriori informazioni e un elenco di servizi compatibili, consulta Servizi che funzionano con AWS Organizations nella documentazione di AWS Organizations .

implementazione

Il processo di creazione di un'applicazione, di nuove funzionalità o di correzioni di codice disponibili nell'ambiente di destinazione. L'implementazione prevede l'applicazione di modifiche in una base di codice, seguita dalla creazione e dall'esecuzione di tale base di codice negli ambienti applicativi.

Ambiente di sviluppo

Vedi ambiente.

controllo di rilevamento

Un controllo di sicurezza progettato per rilevare, registrare e avvisare dopo che si è verificato un evento. Questi controlli rappresentano una seconda linea di difesa e avvisano l'utente in caso di eventi di sicurezza che aggirano i controlli preventivi in vigore. Per ulteriori informazioni, consulta Controlli di rilevamento in Implementazione dei controlli di sicurezza in AWS.

mappatura del flusso di valore dello sviluppo (DVSM)

Un processo utilizzato per identificare e dare priorità ai vincoli che influiscono negativamente sulla velocità e sulla qualità nel ciclo di vita dello sviluppo del software. DVSM estende il processo di

mappatura del flusso di valore originariamente progettato per pratiche di produzione snella. Si concentra sulle fasi e sui team necessari per creare e trasferire valore attraverso il processo di sviluppo del software.

gemello digitale

Una rappresentazione virtuale di un sistema reale, ad esempio un edificio, una fabbrica, un'attrezzatura industriale o una linea di produzione. I gemelli digitali supportano la manutenzione predittiva, il monitoraggio remoto e l'ottimizzazione della produzione.

tabella delle dimensioni

In uno schema a stella, una tabella più piccola che contiene gli attributi dei dati quantitativi in una tabella dei fatti. Gli attributi della tabella delle dimensioni sono in genere campi di testo o numeri discreti che si comportano come testo. Questi attributi vengono comunemente utilizzati per il vincolo delle query, il filtraggio e l'etichettatura dei set di risultati.

disastro

Un evento che impedisce a un carico di lavoro o a un sistema di raggiungere gli obiettivi aziendali nella sua sede principale di implementazione. Questi eventi possono essere disastri naturali, guasti tecnici o il risultato di azioni umane, come errori di configurazione involontari o attacchi di malware.

disaster recovery (DR)

La strategia e il processo utilizzati per ridurre al minimo i tempi di inattività e la perdita di dati causati da un disastro. Per ulteriori informazioni, consulta Disaster Recovery of Workloads su AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML

Vedi linguaggio di manipolazione del database.

progettazione basata sul dominio

Un approccio allo sviluppo di un sistema software complesso collegandone i componenti a domini in evoluzione, o obiettivi aziendali principali, perseguiti da ciascun componente. Questo concetto è stato introdotto da Eric Evans nel suo libro,Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003). Per informazioni su come utilizzare la progettazione basata sul dominio con il modello del fico strangolatore (Strangler Fig), consulta la sezione Modernizzazione incrementale dei servizi Web Microsoft ASP.NET (ASMX) legacy utilizzando container e il Gateway Amazon API.

DOTT.

Vedi disaster recovery.

rilevamento della deriva

Tracciamento delle deviazioni da una configurazione di base. Ad esempio, è possibile AWS CloudFormation utilizzarlo per rilevare deviazioni nelle risorse di sistema oppure AWS Control Tower per rilevare cambiamenti nella landing zone che potrebbero influire sulla conformità ai requisiti di governance.

DVSM

Vedi la mappatura del flusso di valore dello sviluppo.

E

EDA

Vedi analisi esplorativa dei dati.

edge computing

La tecnologia che aumenta la potenza di calcolo per i dispositivi intelligenti all'edge di una rete IoT. Rispetto al cloud computing, l'edge computing può ridurre la latenza di comunicazione e migliorare i tempi di risposta.

crittografia

Un processo di elaborazione che trasforma i dati in chiaro, leggibili dall'uomo, in testo cifrato.

chiave crittografica

Una stringa crittografica di bit randomizzati generata da un algoritmo di crittografia. Le chiavi possono variare di lunghezza e ogni chiave è progettata per essere imprevedibile e univoca.

endianità

L'ordine in cui i byte vengono archiviati nella memoria del computer. I sistemi big-endian memorizzano per primo il byte più importante. I sistemi little-endian memorizzano per primo il byte meno importante.

endpoint

Vedi service endpoint.

servizio endpoint

Un servizio che puoi ospitare in un cloud privato virtuale (VPC) da condividere con altri utenti. Puoi creare un servizio endpoint con AWS PrivateLink e concedere autorizzazioni ad altri Account AWS o a AWS Identity and Access Management (IAM) principali. Questi account o principali possono connettersi al servizio endpoint in privato creando endpoint VPC di interfaccia. Per ulteriori informazioni, consulta Creazione di un servizio endpoint nella documentazione di Amazon Virtual Private Cloud (Amazon VPC).

pianificazione delle risorse aziendali (ERP)

Un sistema che automatizza e gestisce i processi aziendali chiave (come contabilità, MES e gestione dei progetti) per un'azienda.

crittografia envelope

Il processo di crittografia di una chiave di crittografia con un'altra chiave di crittografia. Per ulteriori informazioni, vedete Envelope encryption nella documentazione AWS Key Management Service (AWS KMS).

ambiente

Un'istanza di un'applicazione in esecuzione. Di seguito sono riportati i tipi di ambiente più comuni nel cloud computing:

- ambiente di sviluppo: un'istanza di un'applicazione in esecuzione disponibile solo per il team principale responsabile della manutenzione dell'applicazione. Gli ambienti di sviluppo vengono utilizzati per testare le modifiche prima di promuoverle negli ambienti superiori. Questo tipo di ambiente viene talvolta definito ambiente di test.
- ambienti inferiori: tutti gli ambienti di sviluppo di un'applicazione, ad esempio quelli utilizzati per le build e i test iniziali.
- ambiente di produzione: un'istanza di un'applicazione in esecuzione a cui gli utenti finali possono accedere. In una pipeline CI/CD, l'ambiente di produzione è l'ultimo ambiente di implementazione.
- ambienti superiori: tutti gli ambienti a cui possono accedere utenti diversi dal team di sviluppo principale. Si può trattare di un ambiente di produzione, ambienti di preproduzione e ambienti per i test di accettazione da parte degli utenti.

epica

Nelle metodologie agili, categorie funzionali che aiutano a organizzare e dare priorità al lavoro. Le epiche forniscono una descrizione di alto livello dei requisiti e delle attività di implementazione.

Ad esempio, le epopee della sicurezza AWS CAF includono la gestione delle identità e degli accessi, i controlli investigativi, la sicurezza dell'infrastruttura, la protezione dei dati e la risposta agli incidenti. Per ulteriori informazioni sulle epiche, consulta la strategia di migrazione AWS , consulta la guida all'implementazione del programma.

ERP

Vedi la pianificazione delle risorse aziendali.

analisi esplorativa dei dati (EDA)

Il processo di analisi di un set di dati per comprenderne le caratteristiche principali. Si raccolgono o si aggregano dati e quindi si eseguono indagini iniziali per trovare modelli, rilevare anomalie e verificare ipotesi. L'EDA viene eseguita calcolando statistiche di riepilogo e creando visualizzazioni di dati.

F

tabella dei fatti

Il tavolo centrale in uno schema a stella. Memorizza dati quantitativi sulle operazioni aziendali. In genere, una tabella dei fatti contiene due tipi di colonne: quelle che contengono misure e quelle che contengono una chiave esterna per una tabella di dimensioni.

fallire velocemente

Una filosofia che utilizza test frequenti e incrementali per ridurre il ciclo di vita dello sviluppo. È una parte fondamentale di un approccio agile.

limite di isolamento dei guasti

Nel Cloud AWS, un limite come una zona di disponibilità Regione AWS, un piano di controllo o un piano dati che limita l'effetto di un errore e aiuta a migliorare la resilienza dei carichi di lavoro. Per ulteriori informazioni, consulta AWS Fault Isolation Boundaries.

ramo di funzionalità

Vedi filiale.

caratteristiche

I dati di input che usi per fare una previsione. Ad esempio, in un contesto di produzione, le caratteristiche potrebbero essere immagini acquisite periodicamente dalla linea di produzione.

importanza delle caratteristiche

Quanto è importante una caratteristica per le previsioni di un modello. Di solito viene espresso come punteggio numerico che può essere calcolato con varie tecniche, come Shapley Additive Explanations (SHAP) e gradienti integrati. Per ulteriori informazioni, vedere Interpretabilità del modello di machine learning con:AWS.

trasformazione delle funzionalità

Per ottimizzare i dati per il processo di machine learning, incluso l'arricchimento dei dati con fonti aggiuntive, il dimensionamento dei valori o l'estrazione di più set di informazioni da un singolo campo di dati. Ciò consente al modello di ML di trarre vantaggio dai dati. Ad esempio, se suddividi la data "2021-05-27 00:15:37" in "2021", "maggio", "giovedì" e "15", puoi aiutare l'algoritmo di apprendimento ad apprendere modelli sfumati associati a diversi componenti dei dati.

FGAC

Vedi il controllo granulare degli accessi.

controllo granulare degli accessi (FGAC)

L'uso di più condizioni per consentire o rifiutare una richiesta di accesso.

migrazione flash-cut

Un metodo di migrazione del database che utilizza la replica continua dei dati tramite l'acquisizione dei dati delle modifiche per migrare i dati nel più breve tempo possibile, anziché utilizzare un approccio graduale. L'obiettivo è ridurre al minimo i tempi di inattività.

G

blocco geografico

Vedi restrizioni geografiche.

limitazioni geografiche (blocco geografico)

In Amazon CloudFront, un'opzione per impedire agli utenti di determinati paesi di accedere alle distribuzioni di contenuti. Puoi utilizzare un elenco consentito o un elenco di blocco per specificare i paesi approvati e vietati. Per ulteriori informazioni, consulta Limitare la distribuzione geografica dei contenuti nella CloudFront documentazione.

Flusso di lavoro di GitFlow

Un approccio in cui gli ambienti inferiori e superiori utilizzano rami diversi in un repository di codice di origine. Il flusso di lavoro Gitflow è considerato obsoleto e il flusso di lavoro basato su trunk è l'approccio moderno e preferito.

strategia greenfield

L'assenza di infrastrutture esistenti in un nuovo ambiente. Quando si adotta una strategia greenfield per un'architettura di sistema, è possibile selezionare tutte le nuove tecnologie senza il vincolo della compatibilità con l'infrastruttura esistente, nota anche come brownfield. Per l'espansione dell'infrastruttura esistente, è possibile combinare strategie brownfield e greenfield.

guardrail

Una regola di livello elevato che consente di governare risorse, policy e conformità tra le unità organizzative (OU). I guardrail preventivi applicano le policy per garantire l'allineamento agli standard di conformità. Vengono implementati utilizzando le policy di controllo dei servizi e i limiti delle autorizzazioni IAM. I guardrail di rilevamento rilevano le violazioni delle policy e i problemi di conformità e generano avvisi per porvi rimedio. Sono implementati utilizzando Amazon AWS Config AWS Security Hub GuardDuty AWS Trusted Advisor, Amazon Inspector e controlli personalizzati AWS Lambda .

H

AH

Vedi disponibilità elevata.

migrazione di database eterogenea

Migrazione del database di origine in un database di destinazione che utilizza un motore di database diverso (ad esempio, da Oracle ad Amazon Aurora). La migrazione eterogenea fa in genere parte di uno sforzo di riprogettazione e la conversione dello schema può essere un'attività complessa. AWS offre AWS SCT che aiuta con le conversioni dello schema.

alta disponibilità (HA)

La capacità di un carico di lavoro di funzionare in modo continuo, senza intervento, in caso di sfide o disastri. I sistemi HA sono progettati per il failover automatico, fornire costantemente prestazioni di alta qualità e gestire carichi e guasti diversi con un impatto minimo sulle prestazioni.

modernizzazione storica

Un approccio utilizzato per modernizzare e aggiornare i sistemi di tecnologia operativa (OT) per soddisfare meglio le esigenze dell'industria manifatturiera. Uno storico è un tipo di database utilizzato per raccogliere e archiviare dati da varie fonti in una fabbrica.

migrazione di database omogenea

Migrazione del database di origine in un database di destinazione che condivide lo stesso motore di database (ad esempio, da Microsoft SQL Server ad Amazon RDS per SQL Server). La migrazione omogenea fa in genere parte di un'operazione di rehosting o ridefinizione della piattaforma. Per migrare lo schema è possibile utilizzare le utilità native del database.

dati caldi

Dati a cui si accede frequentemente, ad esempio dati in tempo reale o dati di traduzione recenti. Questi dati richiedono in genere un livello o una classe di storage ad alte prestazioni per fornire risposte rapide alle query.

hotfix

Una soluzione urgente per un problema critico in un ambiente di produzione. A causa della sua urgenza, un hotfix viene in genere creato al di fuori del tipico DevOps flusso di lavoro di rilascio.

periodo di hypercare

Subito dopo la conversione, il periodo di tempo in cui un team di migrazione gestisce e monitora le applicazioni migrate nel cloud per risolvere eventuali problemi. In genere, questo periodo dura da 1 a 4 giorni. Al termine del periodo di hypercare, il team addetto alla migrazione in genere trasferisce la responsabilità delle applicazioni al team addetto alle operazioni cloud.

I

IaC

Considera l'infrastruttura come codice.

Policy basata su identità

Una policy associata a uno o più principi IAM che definisce le relative autorizzazioni all'interno dell' Cloud AWS ambiente.

applicazione inattiva

Un'applicazione che prevede un uso di CPU e memoria medio compreso tra il 5% e il 20% in un periodo di 90 giorni. In un progetto di migrazione, è normale ritirare queste applicazioni o mantenerle on-premise.

IIoT

Vedi Industrial Internet of Things.

infrastruttura immutabile

Un modello che implementa una nuova infrastruttura per i carichi di lavoro di produzione anziché aggiornare, applicare patch o modificare l'infrastruttura esistente. Le infrastrutture immutabili sono intrinsecamente più coerenti, affidabili e prevedibili delle infrastrutture mutabili. Per ulteriori informazioni, consulta la best practice Deploy using immutable infrastructure in Well-Architected AWS Framework.

VPC in ingresso (ingress)

In un'architettura AWS multi-account, un VPC che accetta, ispeziona e indirizza le connessioni di rete dall'esterno di un'applicazione. Nel documento Architettura di riferimento per la sicurezza di AWS si consiglia di configurare l'account di rete con VPC in entrata, in uscita e di ispezione per proteggere l'interfaccia bidirezionale tra l'applicazione e Internet in generale.

migrazione incrementale

Una strategia di conversione in cui si esegue la migrazione dell'applicazione in piccole parti anziché eseguire una conversione singola e completa. Ad esempio, inizialmente potresti spostare solo alcuni microservizi o utenti nel nuovo sistema. Dopo aver verificato che tutto funzioni correttamente, puoi spostare in modo incrementale microservizi o utenti aggiuntivi fino alla disattivazione del sistema legacy. Questa strategia riduce i rischi associati alle migrazioni di grandi dimensioni.

Industria 4.0

Un termine introdotto da Klaus Schwab nel 2016 per riferirsi alla modernizzazione dei processi di produzione attraverso progressi in termini di connettività, dati in tempo reale, automazione, analisi e AI/ML.

infrastruttura

Tutte le risorse e gli asset contenuti nell'ambiente di un'applicazione.

infrastruttura come codice (IaC)

Il processo di provisioning e gestione dell'infrastruttura di un'applicazione tramite un insieme di file di configurazione. Il processo IaC è progettato per aiutarti a centralizzare la gestione dell'infrastruttura, a standardizzare le risorse e a dimensionare rapidamente, in modo che i nuovi ambienti siano ripetibili, affidabili e coerenti.

Internet delle cose industriale (IIoT)

L'uso di sensori e dispositivi connessi a Internet nei settori industriali, come quello manifatturiero, energetico, automobilistico, sanitario, delle scienze della vita e dell'agricoltura. Per ulteriori informazioni, consulta Creazione di una strategia di trasformazione digitale dell'Internet delle cose industriale (IIoT).

VPC di ispezione

In un'architettura AWS multi-account, un VPC centralizzato che gestisce le ispezioni del traffico di rete tra VPC (uguali o diversi Regioni AWS), Internet e reti locali. Nel documento Architettura di riferimento per la sicurezza di AWS si consiglia di configurare l'account di rete con VPC in entrata, in uscita e di ispezione per proteggere l'interfaccia bidirezionale tra l'applicazione e Internet in generale.

Internet of Things (IoT)

La rete di oggetti fisici connessi con sensori o processori incorporati che comunicano con altri dispositivi e sistemi tramite Internet o una rete di comunicazione locale. Per ulteriori informazioni, consulta Cos'è l'IoT?

interpretabilità

Una caratteristica di un modello di machine learning che descrive il grado in cui un essere umano è in grado di comprendere in che modo le previsioni del modello dipendono dai suoi input. Per ulteriori informazioni, consulta la sezione Interpretabilità dei modelli di machine learning con AWS.

IoT

Vedi Internet of Things.

libreria di informazioni IT (ITIL)

Una serie di best practice per offrire servizi IT e allinearli ai requisiti aziendali. ITIL fornisce le basi per ITSM.

gestione dei servizi IT (ITSM)

Attività associate alla progettazione, implementazione, gestione e supporto dei servizi IT per un'organizzazione. Per informazioni sull'integrazione delle operazioni cloud con gli strumenti ITSM, consulta la guida all'integrazione delle operazioni.

ITIL

Vedi la libreria di informazioni IT.

ITSM

Vedi Gestione dei servizi IT.

L

controllo degli accessi basato su etichette (LBAC)

Un'implementazione del controllo di accesso obbligatorio (MAC) in cui agli utenti e ai dati stessi viene assegnato esplicitamente un valore di etichetta di sicurezza. L'intersezione tra l'etichetta di sicurezza utente e l'etichetta di sicurezza dei dati determina quali righe e colonne possono essere visualizzate dall'utente.

zona di destinazione

Una landing zone è un AWS ambiente multi-account ben progettato, scalabile e sicuro. Questo è un punto di partenza dal quale le organizzazioni possono avviare e distribuire rapidamente carichi di lavoro e applicazioni con fiducia nel loro ambiente di sicurezza e infrastruttura. Per ulteriori informazioni sulle zone di destinazione, consulta la sezione Configurazione di un ambiente AWS multi-account sicuro e scalabile.

migrazione su larga scala

Una migrazione di 300 o più server.

BIANCO

Vedi controllo degli accessi basato su etichette.

Privilegio minimo

La best practice di sicurezza per la concessione delle autorizzazioni minime richieste per eseguire un'attività. Per ulteriori informazioni, consulta Applicazione delle autorizzazioni del privilegio minimo nella documentazione di IAM.

eseguire il rehosting (lift and shift)

Vedi 7 R.

sistema little-endian

Un sistema che memorizza per primo il byte meno importante. Vedi anche endianità.

ambienti inferiori

Vedi ambiente.

M

machine learning (ML)

Un tipo di intelligenza artificiale che utilizza algoritmi e tecniche per il riconoscimento e l'apprendimento di schemi. Il machine learning analizza e apprende dai dati registrati, come i dati dell'Internet delle cose (IoT), per generare un modello statistico basato su modelli. Per ulteriori informazioni, consulta la sezione Machine learning.

ramo principale

Vedi filiale.

malware

Software progettato per compromettere la sicurezza o la privacy del computer. Il malware potrebbe interrompere i sistemi informatici, divulgare informazioni sensibili o ottenere accessi non autorizzati. Esempi di malware includono virus, worm, ransomware, trojan horse, spyware e keylogger.

servizi gestiti

Servizi AWS per cui AWS gestisce il livello di infrastruttura, il sistema operativo e le piattaforme e si accede agli endpoint per archiviare e recuperare i dati. Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) e Amazon DynamoDB sono esempi di servizi gestiti. Questi sono noti anche come servizi astratti.

sistema di esecuzione della produzione (MES)

Un sistema software per tracciare, monitorare, documentare e controllare i processi di produzione che convertono le materie prime in prodotti finiti in officina.

MAP

Vedi Migration Acceleration Program.

meccanismo

Un processo completo in cui si crea uno strumento, si promuove l'adozione dello strumento e quindi si esaminano i risultati per apportare le modifiche. Un meccanismo è un ciclo che si rafforza e si migliora man mano che funziona. Per ulteriori informazioni, consulta Creazione di meccanismi nel AWS Well-Architected Framework.

account membro

Tutti gli account Account AWS diversi dall'account di gestione che fanno parte di un'organizzazione in. AWS Organizations Un account può essere membro di una sola organizzazione alla volta.

MEH

Vedi sistema di esecuzione della produzione.

Message Queuing Telemetry Transport (MQTT)

Un protocollo di comunicazione machine-to-machine (M2M) leggero, basato sul modello di pubblicazione/sottoscrizione, per dispositivi IoT con risorse limitate.

microservizio

Un piccolo servizio indipendente che comunica tramite API ben definite ed è in genere di proprietà di piccoli team autonomi. Ad esempio, un sistema assicurativo potrebbe includere microservizi che si riferiscono a funzionalità aziendali, come vendite o marketing, o sottodomini, come acquisti, reclami o analisi. I vantaggi dei microservizi includono agilità, dimensionamento flessibile, facilità di implementazione, codice riutilizzabile e resilienza. Per ulteriori informazioni, consulta Integrazione dei microservizi utilizzando servizi serverless. AWS

architettura di microservizi

Un approccio alla creazione di un'applicazione con componenti indipendenti che eseguono ogni processo applicativo come microservizio. Questi microservizi comunicano tramite un'interfaccia ben definita utilizzando API leggere. Ogni microservizio in questa architettura può essere aggiornato, distribuito e dimensionato per soddisfare la richiesta di funzioni specifiche di un'applicazione. Per ulteriori informazioni, vedere Implementazione dei microservizi su. AWS

Programma di accelerazione della migrazione (MAP)

Un AWS programma che fornisce consulenza, supporto, formazione e servizi per aiutare le organizzazioni a costruire una solida base operativa per il passaggio al cloud e per contribuire a compensare il costo iniziale delle migrazioni. MAP include una metodologia di migrazione per eseguire le migrazioni precedenti in modo metodico e un set di strumenti per automatizzare e accelerare gli scenari di migrazione comuni.

migrazione su larga scala

Il processo di trasferimento della maggior parte del portfolio di applicazioni sul cloud avviene a ondate, con più applicazioni trasferite a una velocità maggiore in ogni ondata. Questa fase utilizza le migliori pratiche e le lezioni apprese nelle fasi precedenti per implementare una fabbrica di migrazione di team, strumenti e processi per semplificare la migrazione dei carichi di lavoro attraverso l'automazione e la distribuzione agile. Questa è la terza fase della strategia di migrazione AWS.

fabbrica di migrazione

Team interfunzionali che semplificano la migrazione dei carichi di lavoro attraverso approcci automatizzati e agili. I team di Migration Factory includono in genere operazioni, analisti e proprietari aziendali, ingegneri addetti alla migrazione, sviluppatori e DevOps professionisti che lavorano nell'ambito degli sprint. Tra il 20% e il 50% di un portfolio di applicazioni aziendali è costituito da schemi ripetuti che possono essere ottimizzati con un approccio di fabbrica. Per ulteriori informazioni, consulta la discussione sulle fabbriche di migrazione e la Guida alla fabbrica di migrazione al cloud in questo set di contenuti.

metadati di migrazione

Le informazioni sull'applicazione e sul server necessarie per completare la migrazione. Ogni modello di migrazione richiede un set diverso di metadati di migrazione. Esempi di metadati di migrazione includono la sottorete, il gruppo di sicurezza e l'account di destinazione. AWS

modello di migrazione

Un'attività di migrazione ripetibile che descrive in dettaglio la strategia di migrazione, la destinazione della migrazione e l'applicazione o il servizio di migrazione utilizzati. Esempio: riorganizza la migrazione su Amazon EC2 AWS con Application Migration Service.

Valutazione del portfolio di migrazione (MPA)

Uno strumento online che fornisce informazioni per la convalida del business case per la migrazione a. Cloud AWS MPA offre una valutazione dettagliata del portfolio (dimensionamento

corretto dei server, prezzi, confronto del TCO, analisi dei costi di migrazione) e pianificazione della migrazione (analisi e raccolta dei dati delle applicazioni, raggruppamento delle applicazioni, prioritizzazione delle migrazioni e pianificazione delle ondate). Lo strumento MPA (richiede l'accesso) è disponibile gratuitamente per tutti i AWS consulenti e i consulenti dei partner APN.

valutazione della preparazione alla migrazione (MRA)

Il processo di acquisizione di informazioni sullo stato di preparazione al cloud di un'organizzazione, l'identificazione dei punti di forza e di debolezza e la creazione di un piano d'azione per colmare le lacune identificate, utilizzando il CAF. AWS Per ulteriori informazioni, consulta la guida di preparazione alla migrazione. MRA è la prima fase della strategia di migrazione AWS.

strategia di migrazione

L'approccio utilizzato per migrare un carico di lavoro verso. Cloud AWS Per ulteriori informazioni, consulta la voce 7 R in questo glossario e consulta Mobilita la tua organizzazione per accelerare le migrazioni su larga scala.

ML

Vedi machine learning.

modernizzazione

Trasformazione di un'applicazione obsoleta (legacy o monolitica) e della relativa infrastruttura in un sistema agile, elastico e altamente disponibile nel cloud per ridurre i costi, aumentare l'efficienza e sfruttare le innovazioni. Per ulteriori informazioni, vedere Strategia per la modernizzazione delle applicazioni in. Cloud AWS

valutazione della preparazione alla modernizzazione

Una valutazione che aiuta a determinare la preparazione alla modernizzazione delle applicazioni di un'organizzazione, identifica vantaggi, rischi e dipendenze e determina in che misura l'organizzazione può supportare lo stato futuro di tali applicazioni. Il risultato della valutazione è uno schema dell'architettura di destinazione, una tabella di marcia che descrive in dettaglio le fasi di sviluppo e le tappe fondamentali del processo di modernizzazione e un piano d'azione per colmare le lacune identificate. Per ulteriori informazioni, vedere Valutazione della preparazione alla modernizzazione per le applicazioni in. Cloud AWS

applicazioni monolitiche (monoliti)

Applicazioni eseguite come un unico servizio con processi strettamente collegati. Le applicazioni monolitiche presentano diversi inconvenienti. Se una funzionalità dell'applicazione registra un

picco di domanda, l'intera architettura deve essere dimensionata. L'aggiunta o il miglioramento delle funzionalità di un'applicazione monolitica diventa inoltre più complessa man mano che la base di codice cresce. Per risolvere questi problemi, puoi utilizzare un'architettura di microservizi. Per ulteriori informazioni, consulta la sezione Scomposizione dei monoliti in microservizi.

MAPPA

Vedi Migration Portfolio Assessment.

MQTT

Vedi Message Queuing Telemetry Transport.

classificazione multiclasse

Un processo che aiuta a generare previsioni per più classi (prevedendo uno o più di due risultati). Ad esempio, un modello di machine learning potrebbe chiedere "Questo prodotto è un libro, un'auto o un telefono?" oppure "Quale categoria di prodotti è più interessante per questo cliente?"

infrastruttura mutabile

Un modello che aggiorna e modifica l'infrastruttura esistente per i carichi di lavoro di produzione. Per migliorare la coerenza, l'affidabilità e la prevedibilità, il AWS Well-Architected Framework consiglia l'uso di un'infrastruttura immutabile come best practice.

O

OAC

Vedi Origin Access Control.

QUERCIA

Vedi Origin Access Identity.

OCM

Vedi gestione delle modifiche organizzative.

migrazione offline

Un metodo di migrazione in cui il carico di lavoro di origine viene eliminato durante il processo di migrazione. Questo metodo prevede tempi di inattività prolungati e viene in genere utilizzato per carichi di lavoro piccoli e non critici.

OI

Vedi l'integrazione delle operazioni.

OLA

Vedi accordo a livello operativo.

migrazione online

Un metodo di migrazione in cui il carico di lavoro di origine viene copiato sul sistema di destinazione senza essere messo offline. Le applicazioni connesse al carico di lavoro possono continuare a funzionare durante la migrazione. Questo metodo comporta tempi di inattività pari a zero o comunque minimi e viene in genere utilizzato per carichi di lavoro di produzione critici.

OPC-UA

Vedi Open Process Communications - Unified Architecture.

Comunicazioni a processo aperto - Architettura unificata (OPC-UA)

Un protocollo di comunicazione machine-to-machine (M2M) per l'automazione industriale. OPC-UA fornisce uno standard di interoperabilità con schemi di crittografia, autenticazione e autorizzazione dei dati.

accordo a livello operativo (OLA)

Un accordo che chiarisce quali sono gli impegni reciproci tra i gruppi IT funzionali, a supporto di un accordo sul livello di servizio (SLA).

revisione della prontezza operativa (ORR)

Un elenco di domande e best practice associate che aiutano a comprendere, valutare, prevenire o ridurre la portata degli incidenti e dei possibili guasti. Per ulteriori informazioni, vedere Operational Readiness Reviews (ORR) nel Well-Architected AWS Framework.

tecnologia operativa (OT)

Sistemi hardware e software che interagiscono con l'ambiente fisico per controllare le operazioni, le apparecchiature e le infrastrutture industriali. Nella produzione, l'integrazione di sistemi OT e di tecnologia dell'informazione (IT) è un obiettivo chiave per le trasformazioni dell'Industria 4.0.

integrazione delle operazioni (OI)

Il processo di modernizzazione delle operazioni nel cloud, che prevede la pianificazione, l'automazione e l'integrazione della disponibilità. Per ulteriori informazioni, consulta la guida all'integrazione delle operazioni.

trail organizzativo

Un percorso creato da noi AWS CloudTrail che registra tutti gli eventi di un'organizzazione per tutti Account AWS . AWS Organizations Questo percorso viene creato in ogni Account AWS che fa parte dell'organizzazione e tiene traccia dell'attività in ogni account. Per ulteriori informazioni, consulta Creazione di un percorso per un'organizzazione nella CloudTrail documentazione.

gestione del cambiamento organizzativo (OCM)

Un framework per la gestione di trasformazioni aziendali importanti e che comportano l'interruzione delle attività dal punto di vista delle persone, della cultura e della leadership. OCM aiuta le organizzazioni a prepararsi e passare a nuovi sistemi e strategie accelerando l'adozione del cambiamento, affrontando i problemi di transizione e promuovendo cambiamenti culturali e organizzativi. Nella strategia di AWS migrazione, questo framework si chiama accelerazione delle persone, a causa della velocità di cambiamento richiesta nei progetti di adozione del cloud. Per ulteriori informazioni, consultare la Guida OCM.

controllo dell'accesso all'origine (OAC)

In CloudFront, un'opzione avanzata per limitare l'accesso per proteggere i contenuti di Amazon Simple Storage Service (Amazon S3). OAC supporta tutti i bucket S3 in generale Regioni AWS, la crittografia lato server con AWS KMS (SSE-KMS) e le richieste dinamiche e dirette al bucket S3. `PUT` `DELETE`

identità di accesso origine (OAI)

Nel CloudFront, un'opzione per limitare l'accesso per proteggere i tuoi contenuti Amazon S3. Quando usi OAI, CloudFront crea un principale con cui Amazon S3 può autenticarsi. I principali autenticati possono accedere ai contenuti in un bucket S3 solo tramite una distribuzione specifica. CloudFront Vedi anche OAC, che fornisce un controllo degli accessi più granulare e avanzato.

O

Vedi la revisione della prontezza operativa.

- NON

Vedi la tecnologia operativa.

VPC in uscita (egress)

In un'architettura AWS multi-account, un VPC che gestisce le connessioni di rete avviate dall'interno di un'applicazione. Nel documento Architettura di riferimento per la sicurezza di

AWS si consiglia di configurare l'account di rete con VPC in entrata, in uscita e di ispezione per proteggere l'interfaccia bidirezionale tra l'applicazione e Internet in generale.

P

limite delle autorizzazioni

Una policy di gestione IAM collegata ai principali IAM per impostare le autorizzazioni massime che l'utente o il ruolo possono avere. Per ulteriori informazioni, consulta Limiti delle autorizzazioni nella documentazione di IAM.

informazioni di identificazione personale (PII)

Informazioni che, se visualizzate direttamente o abbinate ad altri dati correlati, possono essere utilizzate per dedurre ragionevolmente l'identità di un individuo. Esempi di informazioni personali includono nomi, indirizzi e informazioni di contatto.

Informazioni che consentono l'identificazione personale degli utenti

Visualizza le informazioni di identificazione personale.

playbook

Una serie di passaggi predefiniti che raccolgono il lavoro associato alle migrazioni, come l'erogazione delle funzioni operative principali nel cloud. Un playbook può assumere la forma di script, runbook automatici o un riepilogo dei processi o dei passaggi necessari per gestire un ambiente modernizzato.

PLC

Vedi controllore logico programmabile.

PLM

Vedi la gestione del ciclo di vita del prodotto.

policy

Un oggetto in grado di definire le autorizzazioni (vedi politica basata sull'identità), specificare le condizioni di accesso (vedi politicabasata sulle risorse) o definire le autorizzazioni massime per tutti gli account di un'organizzazione in (vedi politica di controllo dei servizi). AWS Organizations

persistenza poliglotta

Scelta indipendente della tecnologia di archiviazione di dati di un microservizio in base ai modelli di accesso ai dati e ad altri requisiti. Se i microservizi utilizzano la stessa tecnologia di archiviazione di dati, possono incontrare problemi di implementazione o registrare prestazioni scadenti. I microservizi vengono implementati più facilmente e ottengono prestazioni e scalabilità migliori se utilizzano l'archivio dati più adatto alle loro esigenze. Per ulteriori informazioni, consulta la sezione Abilitazione della persistenza dei dati nei microservizi.

valutazione del portfolio

Un processo di scoperta, analisi e definizione delle priorità del portfolio di applicazioni per pianificare la migrazione. Per ulteriori informazioni, consulta la pagina Valutazione della preparazione alla migrazione.

predicate

Una condizione di interrogazione che restituisce o, in genere, si trova in una clausola`true`. `false` `WHERE`

predicato pushdown

Una tecnica di ottimizzazione delle query del database che filtra i dati della query prima del trasferimento. Ciò riduce la quantità di dati che devono essere recuperati ed elaborati dal database relazionale e migliora le prestazioni delle query.

controllo preventivo

Un controllo di sicurezza progettato per impedire il verificarsi di un evento. Questi controlli sono la prima linea di difesa per impedire accessi non autorizzati o modifiche indesiderate alla rete. Per ulteriori informazioni, consulta Controlli preventivi in Implementazione dei controlli di sicurezza in AWS.

principale

Un'entità in AWS grado di eseguire azioni e accedere alle risorse. Questa entità è in genere un utente root per un Account AWS ruolo IAM o un utente. Per ulteriori informazioni, consulta Principali in Termini e concetti dei ruoli nella documentazione di IAM.

Privacy fin dalla progettazione

Un approccio all'ingegneria dei sistemi che tiene conto della privacy durante l'intero processo di progettazione.

zone ospitate private

Un container che contiene informazioni su come si desidera che Amazon Route 53 risponda alle query DNS per un dominio e i relativi sottodomini all'interno di uno o più VPC. Per ulteriori informazioni, consulta Utilizzo delle zone ospitate private nella documentazione di Route 53.

controllo proattivo

Un controllo di sicurezza progettato per impedire l'implementazione di risorse non conformi. Questi controlli analizzano le risorse prima del loro provisioning. Se la risorsa non è conforme al controllo, non viene fornita. Per ulteriori informazioni, consulta la guida di riferimento sui controlli nella AWS Control Tower documentazione e consulta Controlli proattivi in Implementazione dei controlli di sicurezza su. AWS

gestione del ciclo di vita del prodotto (PLM)

La gestione dei dati e dei processi di un prodotto durante l'intero ciclo di vita, dalla progettazione, sviluppo e lancio, attraverso la crescita e la maturità, fino al declino e alla rimozione.

Ambiente di produzione

Vedi ambiente.

controllore logico programmabile (PLC)

Nella produzione, un computer altamente affidabile e adattabile che monitora le macchine e automatizza i processi di produzione.

pseudonimizzazione

Il processo di sostituzione degli identificatori personali in un set di dati con valori segnaposto. La pseudonimizzazione può aiutare a proteggere la privacy personale. I dati pseudonimizzati sono ancora considerati dati personali.

pubblica/sottoscrivi (pub/sub)

Un pattern che consente comunicazioni asincrone tra microservizi per migliorare la scalabilità e la reattività. Ad esempio, in un MES basato su microservizi, un microservizio può pubblicare messaggi di eventi su un canale a cui altri microservizi possono abbonarsi. Il sistema può aggiungere nuovi microservizi senza modificare il servizio di pubblicazione.

Q

Piano di query

Una serie di passaggi, come le istruzioni, utilizzati per accedere ai dati in un sistema di database relazionale SQL.

regressione del piano di query

Quando un ottimizzatore del servizio di database sceglie un piano non ottimale rispetto a prima di una determinata modifica all'ambiente di database. Questo può essere causato da modifiche a statistiche, vincoli, impostazioni dell'ambiente, associazioni dei parametri di query e aggiornamenti al motore di database.

R

Matrice RACI

Vedi responsabile, responsabile, consultato, informato (RACI).

ransomware

Un software dannoso progettato per bloccare l'accesso a un sistema informatico o ai dati fino a quando non viene effettuato un pagamento.

Matrice RASCI

Vedi responsabile, responsabile, consultato, informato (RACI).

RCAC

Vedi controllo dell'accesso a righe e colonne.

replica di lettura

Una copia di un database utilizzata per scopi di sola lettura. È possibile indirizzare le query alla replica di lettura per ridurre il carico sul database principale.

riprogettare

Vedi 7 Rs.

obiettivo del punto di ripristino (RPO)

Il periodo di tempo massimo accettabile dall'ultimo punto di ripristino dei dati. Ciò determina quella che viene considerata una perdita di dati accettabile tra l'ultimo punto di ripristino e l'interruzione del servizio.

obiettivo del tempo di ripristino (RTO)

Il ritardo massimo accettabile tra l'interruzione del servizio e il ripristino del servizio.

rifattorizzare

Vedi 7 R.

Regione

Una raccolta di AWS risorse in un'area geografica. Ciascuna Regione AWS è isolata e indipendente dalle altre per fornire tolleranza agli errori, stabilità e resilienza. Per ulteriori informazioni, consulta Specificare cosa può usare Regioni AWS il tuo account.

regressione

Una tecnica di ML che prevede un valore numerico. Ad esempio, per risolvere il problema "A che prezzo verrà venduta questa casa?" un modello di ML potrebbe utilizzare un modello di regressione lineare per prevedere il prezzo di vendita di una casa sulla base di dati noti sulla casa (ad esempio, la metratura).

riospitare

Vedi 7 R.

rilascio

In un processo di implementazione, l'atto di promuovere modifiche a un ambiente di produzione.

trasferisco

Vedi 7 Rs.

ripiattaforma

Vedi 7 Rs.

riacquisto

Vedi 7 Rs.

resilienza

La capacità di un'applicazione di resistere o ripristinare le interruzioni. L'elevata disponibilità e il disaster recovery sono considerazioni comuni quando si pianifica la resilienza in. Cloud AWSPer ulteriori informazioni, vedere Cloud AWS Resilience.

policy basata su risorse

Una policy associata a una risorsa, ad esempio un bucket Amazon S3, un endpoint o una chiave di crittografia. Questo tipo di policy specifica a quali principali è consentito l'accesso, le azioni supportate e qualsiasi altra condizione che deve essere soddisfatta.

matrice di assegnazione di responsabilità (RACI)

Una matrice che definisce i ruoli e le responsabilità di tutte le parti coinvolte nelle attività di migrazione e nelle operazioni cloud. Il nome della matrice deriva dai tipi di responsabilità definiti nella matrice: responsabile (R), responsabile (A), consultato (C) e informato (I). Il tipo di supporto (S) è facoltativo. Se includi il supporto, la matrice viene chiamata matrice RASCI e, se la escludi, viene chiamata matrice RACI.

controllo reattivo

Un controllo di sicurezza progettato per favorire la correzione di eventi avversi o deviazioni dalla baseline di sicurezza. Per ulteriori informazioni, consulta Controlli reattivi in Implementazione dei controlli di sicurezza in AWS.

retain

Vedi 7 R.

andare in pensione

Vedi 7 Rs.

rotazione

Processo di aggiornamento periodico di un segreto per rendere più difficile l'accesso alle credenziali da parte di un utente malintenzionato.

controllo dell'accesso a righe e colonne (RCAC)

L'uso di espressioni SQL di base e flessibili con regole di accesso definite. RCAC è costituito da autorizzazioni di riga e maschere di colonna.

RPO

Vedi l'obiettivo del punto di ripristino.

RTO

Vedi l'obiettivo del tempo di ripristino.

runbook

Un insieme di procedure manuali o automatizzate necessarie per eseguire un'attività specifica. In genere sono progettati per semplificare operazioni o procedure ripetitive con tassi di errore elevati.

S

SAML 2.0

Uno standard aperto utilizzato da molti provider di identità (IdPs). Questa funzionalità abilita il single sign-on (SSO) federato, in modo che gli utenti possano accedere AWS Management Console o chiamare le operazioni AWS API senza che tu debba creare un utente in IAM per tutti i membri dell'organizzazione. Per ulteriori informazioni sulla federazione basata su SAML 2.0, consulta Informazioni sulla federazione basata su SAML 2.0 nella documentazione di IAM.

SCADA

Vedi controllo di supervisione e acquisizione dati.

SCP

Vedi la politica di controllo del servizio.

Secret

In AWS Secrets Manager, informazioni riservate o riservate, come una password o le credenziali utente, archiviate in forma crittografata. È costituito dal valore segreto e dai relativi metadati. Il valore segreto può essere binario, una stringa singola o più stringhe. Per ulteriori informazioni, consulta Cosa c'è in un segreto di Secrets Manager? nella documentazione di Secrets Manager.

controllo di sicurezza

Un guardrail tecnico o amministrativo che impedisce, rileva o riduce la capacità di un autore di minacce di sfruttare una vulnerabilità di sicurezza. Esistono quattro tipi principali di controlli di sicurezza: preventivi, investigativi, reattivi e proattivi.

rafforzamento della sicurezza

Il processo di riduzione della superficie di attacco per renderla più resistente agli attacchi. Può includere azioni come la rimozione di risorse che non sono più necessarie, l'implementazione di

best practice di sicurezza che prevedono la concessione del privilegio minimo o la disattivazione di funzionalità non necessarie nei file di configurazione.

sistema di gestione delle informazioni e degli eventi di sicurezza (SIEM)

Strumenti e servizi che combinano sistemi di gestione delle informazioni di sicurezza (SIM) e sistemi di gestione degli eventi di sicurezza (SEM). Un sistema SIEM raccoglie, monitora e analizza i dati da server, reti, dispositivi e altre fonti per rilevare minacce e violazioni della sicurezza e generare avvisi.

automazione della risposta alla sicurezza

Un'azione predefinita e programmata progettata per rispondere o porre rimedio automaticamente a un evento di sicurezza. Queste automazioni fungono da controlli di sicurezza investigativi o reattivi che aiutano a implementare le migliori pratiche di sicurezza. AWS Esempi di azioni di risposta automatizzate includono la modifica di un gruppo di sicurezza VPC, l'applicazione di patch a un'istanza Amazon EC2 o la rotazione delle credenziali.

Crittografia lato server

Crittografia dei dati a destinazione, da parte di chi li riceve. Servizio AWS

Policy di controllo dei servizi (SCP)

Una policy che fornisce il controllo centralizzato sulle autorizzazioni per tutti gli account di un'organizzazione in AWS Organizations. Le SCP definiscono i guardrail o fissano i limiti alle azioni che un amministratore può delegare a utenti o ruoli. Puoi utilizzare le SCP come elenchi consentiti o elenchi di rifiuto, per specificare quali servizi o azioni sono consentiti o proibiti. Per ulteriori informazioni, consulta le politiche di controllo del servizio nella AWS Organizations documentazione.

endpoint del servizio

L'URL del punto di ingresso per un Servizio AWS. Puoi utilizzare l'endpoint per connetterti a livello di programmazione al servizio di destinazione. Per ulteriori informazioni, consulta Endpoint del Servizio AWS nei Riferimenti generali di AWS.

accordo sul livello di servizio (SLA)

Un accordo che chiarisce ciò che un team IT promette di offrire ai propri clienti, ad esempio l'operatività e le prestazioni del servizio.

indicatore del livello di servizio (SLI)

Misurazione di un aspetto prestazionale di un servizio, ad esempio il tasso di errore, la disponibilità o la velocità effettiva.

obiettivo a livello di servizio (SLO)

Una metrica target che rappresenta lo stato di un servizio, misurato da un indicatore del livello di servizio.

Modello di responsabilità condivisa

Un modello che descrive la responsabilità condivisa AWS per la sicurezza e la conformità del cloud. AWS è responsabile della sicurezza del cloud, mentre tu sei responsabile della sicurezza nel cloud. Per ulteriori informazioni, consulta Modello di responsabilità condivisa.

SIEM

Vedi il sistema di gestione delle informazioni e degli eventi sulla sicurezza.

punto di errore singolo (SPOF)

Un guasto in un singolo componente critico di un'applicazione che può disturbare il sistema.

SLAM

Vedi il contratto sul livello di servizio.

SLI

Vedi l'indicatore del livello di servizio.

LENTA

Vedi obiettivo del livello di servizio.

split-and-seed modello

Un modello per dimensionare e accelerare i progetti di modernizzazione. Man mano che vengono definite nuove funzionalità e versioni dei prodotti, il team principale si divide per creare nuovi team di prodotto. Questo aiuta a dimensionare le capacità e i servizi dell'organizzazione, migliora la produttività degli sviluppatori e supporta una rapida innovazione. Per ulteriori informazioni, vedere Approccio graduale alla modernizzazione delle applicazioni in. Cloud AWS

SPOF

Vedi punto di errore singolo.

schema a stella

Una struttura organizzativa di database che utilizza un'unica tabella dei fatti di grandi dimensioni per archiviare i dati transazionali o misurati e utilizza una o più tabelle dimensionali più piccole per memorizzare gli attributi dei dati. Questa struttura è progettata per l'uso in un data warehouse o per scopi di business intelligence.

modello del fico strangolatore

Un approccio alla modernizzazione dei sistemi monolitici mediante la riscrittura e la sostituzione incrementali delle funzionalità del sistema fino alla disattivazione del sistema legacy. Questo modello utilizza l'analogia di una pianta di fico che cresce fino a diventare un albero robusto e alla fine annienta e sostituisce il suo ospite. Il modello è stato introdotto da Martin Fowler come metodo per gestire il rischio durante la riscrittura di sistemi monolitici. Per un esempio di come applicare questo modello, consulta Modernizzazione incrementale dei servizi Web legacy di Microsoft ASP.NET (ASMX) mediante container e Gateway Amazon API.

sottorete

Un intervallo di indirizzi IP nel VPC. Una sottorete deve risiedere in una singola zona di disponibilità.

controllo di supervisione e acquisizione dati (SCADA)

Nella produzione, un sistema che utilizza hardware e software per monitorare gli asset fisici e le operazioni di produzione.

crittografia simmetrica

Un algoritmo di crittografia che utilizza la stessa chiave per crittografare e decrittografare i dati.

test sintetici

Test di un sistema in modo da simulare le interazioni degli utenti per rilevare potenziali problemi o monitorare le prestazioni. Puoi usare Amazon CloudWatch Synthetics per creare questi test.

T

tags

Coppie chiave-valore che fungono da metadati per l'organizzazione delle risorse. AWS Con i tag è possibile a gestire, identificare, organizzare, cercare e filtrare le risorse. Per ulteriori informazioni, consulta Tagging delle risorse AWS.

variabile di destinazione

Il valore che stai cercando di prevedere nel machine learning supervisionato. Questo è indicato anche come variabile di risultato. Ad esempio, in un ambiente di produzione la variabile di destinazione potrebbe essere un difetto del prodotto.

elenco di attività

Uno strumento che viene utilizzato per tenere traccia dei progressi tramite un runbook. Un elenco di attività contiene una panoramica del runbook e un elenco di attività generali da completare. Per ogni attività generale, include la quantità stimata di tempo richiesta, il proprietario e lo stato di avanzamento.

Ambiente di test

Vedi ambiente.

training

Fornire dati da cui trarre ispirazione dal modello di machine learning. I dati di training devono contenere la risposta corretta. L'algoritmo di apprendimento trova nei dati di addestramento i pattern che mappano gli attributi dei dati di input al target (la risposta che si desidera prevedere). Produce un modello di ML che acquisisce questi modelli. Puoi quindi utilizzare il modello di ML per creare previsioni su nuovi dati di cui non si conosce il target.

Transit Gateway

Un hub di transito di rete che è possibile utilizzare per collegare i VPC e le reti on-premise. Per ulteriori informazioni, consulta Cos'è un gateway di transito nella AWS Transit Gateway documentazione.

flusso di lavoro basato su trunk

Un approccio in cui gli sviluppatori creano e testano le funzionalità localmente in un ramo di funzionalità e quindi uniscono tali modifiche al ramo principale. Il ramo principale viene quindi integrato negli ambienti di sviluppo, preproduzione e produzione, in sequenza.

Accesso attendibile

Concessione delle autorizzazioni a un servizio specificato dall'utente per eseguire attività all'interno dell'organizzazione AWS Organizations e nei suoi account per conto dell'utente. Il servizio attendibile crea un ruolo collegato al servizio in ogni account, quando tale ruolo è necessario, per eseguire attività di gestione per conto dell'utente. Per ulteriori informazioni,

consulta Utilizzo AWS Organizations con altri AWS servizi nella AWS Organizations documentazione.

regolazione

Modificare alcuni aspetti del processo di training per migliorare la precisione del modello di ML. Ad esempio, puoi addestrare il modello di ML generando un set di etichette, aggiungendo etichette e quindi ripetendo questi passaggi più volte con impostazioni diverse per ottimizzare il modello.

team da due pizze

Una piccola DevOps squadra che puoi sfamare con due pizze. Un team composto da due persone garantisce la migliore opportunità possibile di collaborazione nello sviluppo del software.

U

incertezza

Un concetto che si riferisce a informazioni imprecise, incomplete o sconosciute che possono minare l'affidabilità dei modelli di machine learning predittivi. Esistono due tipi di incertezza: l'incertezza epistemica, che è causata da dati limitati e incompleti, mentre l'incertezza aleatoria è causata dal rumore e dalla casualità insiti nei dati. Per ulteriori informazioni, consulta la guida Quantificazione dell'incertezza nei sistemi di deep learning.

compiti indifferenziati

Conosciuto anche come sollevamento di carichi pesanti, è un lavoro necessario per creare e far funzionare un'applicazione, ma che non apporta valore diretto all'utente finale né offre vantaggi competitivi. Esempi di attività indifferenziate includono l'approvvigionamento, la manutenzione e la pianificazione della capacità.

ambienti superiori

Vedi ambiente.

V

vacuum

Un'operazione di manutenzione del database che prevede la pulizia dopo aggiornamenti incrementali per recuperare lo spazio di archiviazione e migliorare le prestazioni.

controllo delle versioni

Processi e strumenti che tengono traccia delle modifiche, ad esempio le modifiche al codice di origine in un repository.

Peering VPC

Una connessione tra due VPC che consente di instradare il traffico tramite indirizzi IP privati. Per ulteriori informazioni, consulta Che cos'è il peering VPC? nella documentazione di Amazon VPC.

vulnerabilità

Un difetto software o hardware che compromette la sicurezza del sistema.

W

cache calda

Una cache del buffer che contiene dati correnti e pertinenti a cui si accede frequentemente. L'istanza di database può leggere dalla cache del buffer, il che richiede meno tempo rispetto alla lettura dalla memoria dal disco principale.

dati caldi

Dati a cui si accede raramente. Quando si eseguono interrogazioni di questo tipo di dati, in genere sono accettabili interrogazioni moderatamente lente.

funzione finestra

Una funzione SQL che esegue un calcolo su un gruppo di righe che si riferiscono in qualche modo al record corrente. Le funzioni della finestra sono utili per l'elaborazione di attività, come il calcolo di una media mobile o l'accesso al valore delle righe in base alla posizione relativa della riga corrente.

Carico di lavoro

Una raccolta di risorse e codice che fornisce valore aziendale, ad esempio un'applicazione rivolta ai clienti o un processo back-end.

flusso di lavoro

Gruppi funzionali in un progetto di migrazione responsabili di una serie specifica di attività. Ogni flusso di lavoro è indipendente ma supporta gli altri flussi di lavoro del progetto. Ad esempio,

il flusso di lavoro del portfolio è responsabile della definizione delle priorità delle applicazioni, della pianificazione delle ondate e della raccolta dei metadati di migrazione. Il flusso di lavoro del portfolio fornisce queste risorse al flusso di lavoro di migrazione, che quindi migra i server e le applicazioni.

VERME

Vedi scrivere una volta, leggere molti.

WQF

Vedi AWS Workload Qualification Framework.

scrivi una volta, leggi molte (WORM)

Un modello di storage che scrive i dati una sola volta e ne impedisce l'eliminazione o la modifica. Gli utenti autorizzati possono leggere i dati tutte le volte che è necessario, ma non possono modificarli. Questa infrastruttura di archiviazione dei dati è considerata immutabile.

Z

exploit zero-day

Un attacco, in genere malware, che sfrutta una vulnerabilità zero-day.

vulnerabilità zero-day

Un difetto o una vulnerabilità assoluta in un sistema di produzione. Gli autori delle minacce possono utilizzare questo tipo di vulnerabilità per attaccare il sistema. Gli sviluppatori vengono spesso a conoscenza della vulnerabilità causata dall'attacco.

applicazione zombie

Un'applicazione che prevede un utilizzo CPU e memoria inferiore al 5%. In un progetto di migrazione, è normale ritirare queste applicazioni.

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.