在 Amazon RDS和 Amazon Aurora 中調校 PostgreSQL 參數

AWS 規範指引

AWS 規範指引: 在 Amazon RDS和 Amazon Aurora 中調校 PostgreSQL 參數

Table of Contents

調整 Amazon RDS 和 Amazon Aurora 中的 PostgreSQL 參數

亞曼德拉、拉姆賈基尼和羅希特·卡普爾、Amazon Web Services ()AWS

2024 年 2 月 (文件歷史記錄)

適用於 PostgreSQL 的亞馬 Amazon Aurora PostgreSQL 版和 Amazon Relational Database Service 服務 (Amazon RDS) 是精密的開放原始碼關聯式資料庫服務，可提供全方位功能。您可以使用這些服務在各種平台和應用程式上設定 PostgreSQL 資料庫。

Aurora 和 Amazon RDS 提供了一種簡化的方式來管理和操作 PostgreSQL 資料庫。它們旨在管理資料庫基礎結構，並在您專注於應用程式開發時，提供高可用性、耐久性和可擴充性。但是，這些服務的預設組態可能不適合所有工作負載。根據預設，這些服務會設定為以最少的資源在任何地方執行，而且不會產生弱點。調整參數可協助您達到更好的效能、減少停機時間，並改善整體資料庫效率。透過針對特定工作負載最佳化參數，您可以充分利用 Amazon RDS 和 Aurora 提供的功能，並將其優勢發揮到最大。

例如，您可以透過最佳化適用於 PostgreSQL 的 Aurora 和 Amazon RDS 並設定其參數來提升效能。您也應該在建立資料庫查詢時考量效能。即使您最佳化資料庫設定，如果您的查詢執行完整的資料表掃描、使用索引，或是執行昂貴的聯結或彙總作業，系統仍可能執行不佳。

本指南適用於想要調整 PostgreSQL 資料庫的記憶體、自動真空、記錄和邏輯複寫參數的資料庫開發人員、資料庫工程師和管理員。本指南也涵蓋 Amazon RDS for PostgreSQL 版和 Aurora 相容的特定參數。調整這些參數可協助您最佳化資料庫效能，並減少特定工作負載的資源使用量，進而提高效能並節省成本。

使用資料庫和資料庫叢集參數群組

Amazon RDS 和 Aurora 可以根據您的資料庫執行個體大小，針對特定設定自動判斷最適合的參數值。它們還支援參數自訂，以透過資料庫執行個體和叢集的參數群組進行效能調整。

您可以使用資料庫和資料庫叢集參數群組來修改控制資料庫引擎行為各個層面的參數，例如記憶體使用量、磁碟 I/O、網路和鎖定。透過調整這些參數，您可以針對特定工作負載最佳化資料庫引擎，並改善效能。

您可以使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 Amazon RDS API 來建立和設定資料庫和資料庫叢集參數群組。本指南假設您正在使用 AWS CLI. 如需主控台和 API 指示，請參閱 Amazon RDS 文件中的使用資料庫參數群組和使用資料庫叢集參數群組。

> **Important**
>
> 若要使用本指南中提供的指 AWS CLI 令，您必須先安裝並設定 AWS CLI.

若要建立和設定資料庫參數群組：

```
# 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
```

建立和設定資料庫叢集參數群組：

```
# 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 和 Amazon RDS 提供預設參數群組，其中包含無法變更的預先設定值。

參數群組可設定為靜態或動態。無論是否啟用此`ApplyMethod=immediate`選項，都會立即套用動態參數。靜態參數需要手動重新啟動才能生效。

調整記憶體參數

調整記憶體參數是最佳化 Amazon RDS和 Aurora Postgre SQL相容資料庫效能的重要任務。為各種資料庫操作正確配置記憶體，例如執行查詢、排序、索引和快取，可以大幅改善資料庫效能。本節涵蓋 Amazon RDS和 Aurora Postgre 中一些最關鍵的記憶體參數 SQL- 相容，包括其預設值、用於計算適當值的公式，以及如何變更這些參數。如需參數的完整清單，請參閱 Amazon RDS 文件中的使用 RDS for PostgreSQL 資料庫執行個體上的參數，以及 Aurora 文件中的 Amazon Aurora PostgreSQL 參數。

最佳化這些參數需要深入了解資料庫工作負載，以及 Aurora 或 Amazon RDS 資料庫執行個體上可用的資源。系統效能受到兩大類參數的影響：重要參數和或有參數。

重要參數是不可或缺的參數，會對系統效能造成重大且直接的影響，對於達到最佳結果至關重要。

- shared_buffers
- temp_buffers
- effective_cache_size
- work_mem
- maintenance_work_mem

或有參數是案例和業務特定的參數。它們取決於其他因素，並在支援重要參數和最大化整體系統效能方面扮演間接但重要的角色。

- random_page_cost
- seq_page_cost
- track_activity_query_size
- idle_in_transaction_session_timeout
- statement_timeout
- search_path
- max_connections

以下各節會更詳細地討論這些參數。

shared_buffers

shared_buffers 參數控制 PostgreSQL 用來快取記憶體中資料的記憶體量。將此參數設定為適當的值有助於改善查詢效能。

對於 Amazon RDS，根據資料庫執行個體的可用記憶體， 的預設值shared_buffers會設定為{DBInstanceClassMemory/32768}位元組。對於 Aurora，預設值會根據資料庫執行個體的{DBInstanceClassMemory/12038,-50003}可用記憶體設定為 。此參數的最佳值取決於多個因素，包括資料庫的大小、並行連線的數量，以及可用的執行個體記憶體。

> **Note**
>
> 當您縮減資料庫執行個體規模時，請務必調整 shared_buffers以符合新的記憶體限制。否則，PostgreSQL 可能無法啟動，或者您可能會遇到 PostgreSQL 引擎的問題。記錄所有變更，以便日後進行更簡單的調整。

AWS CLI 語法

下列命令shared_buffers會針對特定資料庫參數群組變更。此變更適用於使用 參數群組的所有執行個體或叢集。

```
# 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"
```

類型：靜態 （套用變更需要重新啟動）

預設值：Amazon RDS for Postgre 中的{DBInstanceClassMemory/32768}位元組SQL，Aurora Postgre {DBInstanceClassMemory/12038,-50003}中的位元組 SQL- 相容。在大多數情況下，此方程式可運作約佔系統記憶體的 25%。遵循本指南，參數群組中的shared_buffers設定是使用 Postgre SQL的預設 8K 緩衝區單位，而不是位元組或千位元組來設定。

shared_buffers 參數設定可能會對效能產生重大影響，因此建議您徹底測試變更，以確保該值適合您的工作負載。

範例

假設您的金融服務應用程式正在 Amazon RDS或 Aurora 上執行 PostgreSQL 資料庫。此資料庫用於存放客戶交易資料。它有大量的資料表，且由大量伺服器上的多個應用程式存取。應用程式正在經歷緩慢的查詢效能和高CPU用量。您判斷調校 shared_buffers 參數可能有助於改善效能。

在 Amazon RDS for Postgre 中SQL， 的預設值shared_buffers會設定為 中可用記憶體的{DBInstanceClassMemory/32768}位元組 db.r5.xlarge (例如 3 GB) 。若要判斷 的適當值shared_buffers，您會執行一系列具有不同值的測試shared_buffers，從可用記憶體的預設值開始，並逐漸增加值。針對每個測試，您可以測量資料庫的查詢效能和CPU用量。

根據測試結果，您可以判斷將 值設定為 shared_buffers 8 GB 可為您的工作負載提供最佳的整體查詢效能和CPU用量。此值是透過測試和分析工作負載特性的組合來決定，包括資料庫的大小、查詢的數量和複雜性、並行使用者的數量，以及可用的系統資源。在您進行變更之後，您的監控系統會檢查資料庫的效能，以確保新值適合您的工作負載。然後，您可以視需要微調其他參數，以進一步改善效能。

temp_buffers

temp_buffers 是 Aurora Postgre SQL-Compatible 和 Amazon RDS for PostgreSQL 中的關鍵組態參數，可顯著影響暫時資料表上涉及排序、雜湊和彙總操作的工作負載的效能。此參數會決定為臨時緩衝區配置的記憶體量，進而影響此類操作的效率和速度。如果沒有為 配置足夠的記憶體temp_buffers，則系統可能需要在暫存資料表上使用較慢、效率較低的排序、雜湊和彙總操作方法，導致效能欠佳。

AWS CLI 語法

下列命令temp_buffers會針對特定資料庫參數群組變更。此變更適用於使用 參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態（如果您設定，會立即套用變更ApplyMethod=immediate）

預設值：8 MB

如需此參數的詳細資訊，請參閱 PostgreSQL 文件中的資源使用。

範例

如果您的工作負載涉及暫存資料表上的大量排序、雜湊和彙總操作，temp_buffers可能無法配置足夠的記憶體。在此情況下，系統可能需要對臨時資料表執行排序操作，這些資料表會導致較慢的磁碟型方法，而不是記憶體內排序、雜湊和彙總操作。這可能會導致查詢效能顯著降低，尤其是涉及大型資料集的查詢。增加 的值temp_buffers可確保有足夠的記憶體可在記憶體中執行此類操作，進而大幅改善效能。

若要尋找 的最佳值temp_buffers，請監控您的系統效能，並識別效能欠佳的區域。如果您注意到查詢回應時間緩慢或CPU高使用率，請考慮調整 temp_buffers。例如，如果您的工作負載涉及許多臨時資料表，增加 的值temp_buffers有助於確保這些資料表儲存在記憶體中。這比從儲存體使用讀取/寫入 I/O 要快得多。

以temp_buffers小增量實驗不同值，並在每次變更後仔細監控系統效能。分析不同值對效能的影響，並根據工作負載的特定特性微調設定。

effective_cache_size

effective_cache_size 參數指定 PostgreSQL 應假設可用於快取資料的記憶體量。正確設定此參數可以讓 PostgreSQL 更好地使用可用的記憶體，進而改善效能。

AWS CLI 語法

下列命令effective_cache_size會針對特定資料庫參數群組變更。此變更適用於使用 參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態（如果您設定，會立即套用變更ApplyMethod=immediate）

預設值：SUM(DBInstanceClassMemory/12038,-50003)KB

範例

線上學習平台具有使用者經常存取的課程材料、學生資料和其他內容的大型資料庫。應用程式會在具有 32 GB 記憶體的 Amazon RDS for PostgreSQL db.r5.xlarge執行個體上執行。當使用者嘗試讀取經常存取的內容時，應用程式會體驗到緩慢的效能。分析資料庫伺服器的資源用量後，您判斷 PostgreSQL 未充分利用可用的記憶體。

Amazon RDS for PostgreSQL 中的 effective_cache_size 參數會控制伺服器用於磁碟快取的記憶體量。執行個體類別 的預設值設定為 SUM({DBInstanceClassMemory/12038},-50003) KBdb.r5.xlarge，但此預設值可能不適用於所有工作負載。在此範例中，資料庫伺服器可能會儲存大量經常存取的課程材料和學生資料。增加 effective_cache_size 參數的值可能會導致更多資料快取在記憶體中，從而減少所需的磁碟讀取數量並改善查詢效能。

當您執行查詢時，Amazon RDS for PostgreSQL 會先檢查查詢所需的資料是否已在快取中。如果是這樣，可以從記憶體讀取資料，而不是從磁碟讀取。如果資料不在快取中，則必須從磁碟讀取，這可能是慢操作。

對於線上學習平台，您可能會在測試和分析後決定effective_cache_size將 設為 16 GB（可用記憶體的一半）。此值可讓 PostgreSQL 更好地使用可用的記憶體，從而減少所需的磁碟讀取數量並改善查詢效能。

work_mem

work_mem 參數控制查詢用於排序和雜湊操作的記憶體量。其預設值為 4 MB。如果查詢包含多個操作，則每個操作最多可以使用 4 MB。增加 的值work_mem可以改善需要排序或雜湊的查詢效能，因為這些操作需要更多記憶體。不過，將此參數設為過高可能會導致記憶體使用量過高，進而降低效能。

若要計算 的最佳值work_mem，您可以使用下列公式：

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

其中 available_memory是伺服器上可用的記憶體總量，max_connections 是允許的連線數量上限，而 work_mem_fraction 是決定應配置給每個連線的可用記憶體數量的一小部分。

AWS CLI 語法

下列命令work_mem會針對特定資料庫參數群組變更。此變更適用於使用 參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態（如果您設定 ，會立即套用變更ApplyMethod=immediate）

預設值：4 MB

範例

社交媒體分析工具會處理大量資料，以及涉及複雜排序和聯結操作的查詢，會導致磁碟 I/O 很高，並溢出到磁碟。如果您將 的值work_mem從 4 MB 增加到 16 MB，PostgreSQL 可以為這些操作使用更多記憶體。這可減少 I/O 量，並改善查詢效能。

maintenance_work_mem

maintenance_work_mem 參數控制維護操作使用的記憶體量VACUUM，例如 ANALYZE、 和索引建立。Amazon RDS和 Aurora 中此參數的預設值為 64 MB。

若要計算此參數的適當值，您可以使用此公式：

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

Aurora Postgre SQL-Compatible Edition 和 Amazon RDS for PostgreSQL 套用下列公式以設定最佳值：

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

AWS CLI 語法

下列命令maintenance_work_mem會針對特定資料庫參數群組變更。此變更適用於使用 參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態 （如果您設定 ，會立即套用變更ApplyMethod=immediate）

預設值：64 MB

範例

您的大規模應用程式會使用託管在 Aurora 或 Amazon 上的 PostgreSQL 資料庫RDS。您注意到資料庫在諸如清空和索引等維護活動期間緩慢且沒有回應。您可以監控記憶體用量、維護操作時間和CPU用量等指標，以判斷 的目前值是否maintenance_work_mem造成問題。

若要判斷 的最佳值maintenance_work_mem，您可以調整 參數並監控其影響。如果在維護操作期間，記憶體用量持續很高或操作時間超過預期，增加 maintenance_work_mem可能會有所幫助。相反地，如果在維護操作期間CPU使用量持續很高，減少用量maintenance_work_mem可能會有所幫助。透過反覆進行調整和測試，您可以找到 的最佳值maintenance_work_mem，為記憶體用量、維護操作時間和CPU用量提供最佳平衡。

調查期間，假設您判斷 的預設值 maintenance_work_mem 64 MB 太低，無法滿足您的資料庫大小。因此，維護操作需要更長的時間才能完成，導致過度停機，並降低應用程式的效能。若要解決此問題，您可以決定透過將maintenance_work_mem參數從 64 MB 增加到 512 MB （您識別為最佳值） 來調

整 參數。套用變更可以提高三分之二的維護操作時間。例如，先前完成的真空操作需要 30 分鐘，現在可能只需要 10 分鐘。由於此最佳化，您的資料庫現在可以更有效率地處理維護活動。

random_page_cost

random_page_cost 參數有助於判斷執行隨機頁面存取的成本。Amazon RDS和 Aurora 中的查詢規劃器使用此參數以及有關資料表的其他統計資料，來判斷執行查詢的最有效計劃。

seq_page_cost 和 random_page_cost 參數密切相關，通常由規劃器一起使用，以比較不同存取方法的成本，並決定哪個方法最有效率。因此，如果您變更其中一個參數，您也應該考慮是否需要調整其他參數。

一般而言，查詢規劃器會嘗試將執行查詢的成本降至最低。成本是使用磁碟頁面讀取數目和 值的組合來決定random_page_cost。較高的值random_page_cost傾向於依序掃描，而較低的值傾向於偏好索引掃描。較低的值也會偏好巢狀迴圈聯結，而不是雜湊聯結。

除非在random_page_cost參數群組或本機工作階段中設定值，否則 參數會使用預設 PostgreSQL 引擎值 （4）。您可以根據伺服器和工作負載的特定特性來調整此值。如果工作負載中使用的大多數索引都符合記憶體或 Aurora 分層快取，請將 的值變更為接近 random_page_cost的值seq_page_cost是適當的。

AWS CLI 語法

下列命令random_page_cost會針對特定資料庫參數群組變更。此變更適用於使用 參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態 （如果您設定 ，會立即套用變更ApplyMethod=immediate）

預設值：4

範例

假設您的資料庫將大量資料存放在資料表中，該資料表經常查詢為非索引資料欄上的篩選條件。查詢需要很長時間才能完成，而查詢規劃器不會選擇存取資料最有效的計畫。

改善效能的一種方法是減少 random_page_cost 參數。如果您將其設定為 1，隨機存取頁面的成本會比預設值便宜四倍。如果您random_page_cost保留預設值 4，隨機頁面存取的費用會是連續頁面存取的四倍 （依 seq_page_cost 參數而定，預設為 1.0）。不過，在這種情況下，根據儲存類型，隨機存取頁面實際上可能更昂貴。

降低 random_page_cost 參數的值可讓查詢規劃器更可能選取索引型計畫，或使用更適合資料表特定特徵的不同存取方法。

建議您在變更 參數後監控查詢效能，並視需要進行調整。您也應該使用 EXPLAIN陳述式檢查查詢規劃器，以檢查它是否正在選取有效的計畫。

這只是一個範例。最佳設定取決於工作負載的特定特性。此外，這只是效能調校的一個方面；您也應該考慮可能會影響查詢效能的其他參數和組態選項。

seq_page_cost

參數seq_page_cost有助於判斷執行循序頁面存取的成本。Amazon RDS和 Aurora 中的查詢規劃器使用此參數以及有關資料表的其他統計資料，來判斷執行查詢的最有效計劃。

seq_page_cost 和 random_page_cost 參數密切相關，通常由規劃器一起使用，以比較不同存取方法的成本，並決定哪個方法最有效率。因此，如果您變更其中一個參數，您也應該考慮是否需要調整其他參數。

以循序方式存取資料表時，PostgreSQL 可以使用作業系統的檔案系統快取來提供更快速的存取。依預設， seq_page_cost 會設定為 1.0，其假設循序頁面存取與單一磁碟區塊讀取一樣便宜。如果您的資料表主要以循序方式存取，但由於受到較少 的限制，磁碟存取速度很慢IOPS，您可能想要增加 的值seq_page_cost，以反映存取磁碟的額外成本。

變更此參數的值會影響系統中執行的所有查詢，因此建議您使用不同的值測試查詢，以判斷特定使用案例的最佳值。

AWS CLI 語法

下列命令seq_page_cost會針對特定資料庫參數群組變更。此變更適用於使用 參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態（如果您設定 ，會立即套用變更ApplyMethod=immediate）

預設值：1.0

範例

假設您的資料庫在資料表中存放大量資料，主要是按順序存取。此表主要用於報告，查詢執行速度非常慢，查詢規劃器無法選取存取資料最有效的計劃。

改善效能的一種方法是減少 seq_page_cost 參數。預設值為 1.0，假設循序頁面存取與單一磁碟區塊讀取一樣便宜。不過，在此特定情況下，由於儲存類型（取決於 I/O），循序頁面存取實際上可能比該類型更昂貴。如果您將 seq_page_cost 設為 0.5，循序頁面存取的成本為預設值的一半。此變更可能會讓查詢規劃器更可能選擇使用順序存取方法的計劃，而該方法更適合資料表的特定特徵。

建議您在變更 參數後監控查詢效能，並視需要進行調整。您也應該使用 EXPLAIN陳述式檢查查詢計劃，以檢查它是否正在選取有效的計劃。

這只是一個範例。最佳設定取決於工作負載的特定特性。此外，這只是效能調校的一個方面；您也應該考慮影響查詢效能的其他參數和組態選項。

track_activity_query_size

track_activity_query_size 參數控制pg_stat_activity檢視中每個作用中工作階段記錄的查詢字串大小。根據預設，只有查詢字串的前 1，024 個位元組會記錄在 Amazon RDS for Postgre 中 SQL，而 4，096 個位元組會記錄在 Aurora Postgre SQL-Compatible 中。如果您想要記錄較長的查詢，您可以將此參數設定為較高的值。

AWS CLI 語法

下列命令track_activity_query_size會針對特定資料庫參數群組變更。此變更適用於使用 參數群組的所有執行個體或叢集。

```
# 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"
```

類型：靜態（套用變更需要重新啟動）

預設值：1，024 位元組（Amazon RDS for Postgre SQL）、4，096 位元組（Aurora Postgre SQL-相容）

範例

您的 Amazon RDS for PostgreSQL 資料庫正在經歷緩慢的查詢效能，而您懷疑問題可能與長時間執行的查詢有關。您可以將較長的查詢記錄到pg_stat_activity檢視，以進一步調查。

增加 track_activity_query_size 參數的值可能會導致日誌記錄增加，這可能會對資料庫造成效能影響。建議您在問題解決後，將 參數設回其預設 1，024 值。

idle_in_transaction_session_timeout

idle_in_transaction_session_timeout 參數控制閒置交易停止前的等待時間。

Amazon RDS for PostgreSQL 和 Aurora Postgre SQL-Compatible 中此參數的預設值為 86，400，000 毫秒。

AWS CLI 語法

下列命令idle_in_transaction_session_timeout會針對特定資料庫參數群組變更。此變更適用於使用 參數群組的所有執行個體或叢集。

```
# 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
```

類型：動態（如果您設定 ，會立即套用變更ApplyMethod=immediate）

預設值：86，400，000 毫秒（Aurora Postgre SQL- 相容）

範例

您有處理線上訂單的電子商務應用程式。應用程式會使用託管在 Amazon RDS或 Aurora 上的 PostgreSQL 資料庫。每當客戶下訂單時，應用程式會啟動新的交易來更新庫存和訂單記錄。

如果交易長時間保持閒置狀態，可能會阻止其他交易存取相同的記錄，從而導致效能問題，甚至可能導致應用程式停機。此外，未正確停止的閒置交易可能會耗用寶貴的系統資源，例如記憶體和 CPU。

若要避免此類問題，您可以將 idle_in_transaction_session_timeout 參數設定為對應用程式有意義的值。例如，您可以將其設定為 5 分鐘（300 秒），以便讓閒置超過 5 分鐘的任何交易自動停止。這有助於確保有效使用系統資源，且應用程式可以處理大量訂單，而不會拖慢速度。透過為 設定適當的值idle_in_transaction_session_timeout，您可以協助確保您的應用程式在 Amazon RDS或 Aurora 上以最佳狀態執行。

statement_timeout

statement_timeout 參數會設定查詢在停止之前可以執行的時間上限。

AWS CLI 語法

下列命令statement_timeout會針對特定資料庫參數群組變更。此變更適用於使用 參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態（如果您設定 ，會立即套用變更ApplyMethod=immediate）

預設值：0 毫秒（無逾時）

範例

您的 Web 應用程式允許使用者搜尋大型的產品資料庫。搜尋查詢有時可能需要很長時間才能完成，導致使用者回應時間變慢。若要解決此問題，您可以將 statement_timeout 參數設定為低值，例如 10 秒，這會強制停止任何需要超過 10 秒的查詢。

這看起來像是一個激進的指標，但實際上可以非常有效地改善效能。在許多情況下，長時間執行的查詢是由最佳化不佳的SQL查詢或效率不佳的索引所造成。透過設定低statement_timeout值，您可以識別這些有問題的查詢，並採取步驟將其最佳化。

例如，假設您發現特定搜尋查詢持續逾時。您可以使用 EXPLAIN和 等工具EXPLAIN ANALYZE來分析查詢，並識別任何效能瓶頸。當您發現問題時，您可以採取步驟，透過新增索引、重寫查詢或使用不同的搜尋演算法來最佳化查詢。透過以這種方式持續分析和最佳化SQL查詢，您可以大幅提升應用程式的效能。

search_path

search_path 參數會決定在SQL陳述式中搜尋結構描述的順序。預設值為 $user, public，這表示 PostgreSQL 會先在符合使用者名稱的結構描述中搜尋物件，然後在公有結構描述中搜尋物件。

如果您有大量結構描述，或需要存取特定結構描述中的物件，變更 search_path 參數有助於改善效能。當您search_path設定為特定結構描述時，PostgreSQL 可以更快地尋找物件，而無需搜尋多個結構描述。

若要變更 Amazon RDS和 Aurora 中的 search_path 參數，您可以使用下列命令進行 ROLE 層級：

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

AWS CLI 語法

下列命令search_path會針對特定資料庫參數群組變更。此變更適用於使用 參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態 （ 如果您設定 ，會立即套用變更ApplyMethod=immediate ）

預設值：$user, public

範例

每個使用 Amazon RDS for PostgreSQL 或 Aurora Postgre SQL相容資料庫的租用戶都有具有個別結構描述的多租用戶應用程式，而且您需要執行涉及從多個結構描述加入資料的查詢。

根據預設，Amazon RDS和 Aurora 會使用搜尋路徑來決定要用於指定資料表的結構描述。搜尋路徑是結構描述名稱的清單，當您參考資料表時，PostgreSQL 會依序搜尋這些名稱，但不符合結構描述名稱的資格。根據預設，Amazon RDS和 Aurora 會先在結構描述中尋找與目前使用者具有相同名稱的資料表，然後尋找公有結構描述。

假設您想要執行查詢，其中包含從多個結構描述加入資料表，名稱為 tenant1、 tenant2和 tenant3。若要使用租戶結構描述，您可以在查詢中包含結構描述名稱：

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

不過，更有效率的方法是使用AWS CLI 語法區段中的 命令，將 search_path 參數變更為包含租戶結構描述。您也可以在 PostgreSQL 工作階段中使用 SET命令：

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

然後，您可以撰寫查詢，而不需要驗證結構描述名稱：

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

這可讓您的查詢更簡潔且易於讀取，而且如果您有許多查詢涉及從多個結構描述加入資料表，也可以簡化您的應用程式程式碼。

max_connections

max_connections 參數會設定 PostgreSQL 資料庫的並行連線數目上限。

> **Note**
>
> 當您縮減資料庫執行個體規模時，請務必調整 max_connections以符合新的記憶體限制。否則，PostgreSQL 可能無法啟動，或者您可能會遇到 PostgreSQL 引擎的問題。記錄所有變更，以便日後進行更簡單的調整。

AWS CLI 語法

下列命令max_connections會針對特定資料庫參數群組變更。此變更適用於使用 參數群組的所有執行個體或叢集。

```
# 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"
```

類型：靜態 （套用變更需要重新啟動）

預設值：LEAST(DBInstanceClassMemory/9531392,5000)連線

若要最佳化在 Amazon RDS或 Aurora max_connections中的 使用，並將對效能的影響降至最低，請考慮下列最佳實務：

- 根據可用的系統資源設定參數值。
- 監控連線用量，以防止快速達到限制。
- 使用連線集區減少所需的連線數。
- 使用 Amazon RDS Proxy 進行連線集區。

當您max_connections在 Amazon RDS for PostgreSQL 或 Amazon Aurora Postgre SQL相容中調整 時，請考慮可用的執行個體類型及其配置的資源，並專注於記憶體和CPU容量。儲存體和 I/O 詳細資訊由 管理 AWS，因此您可以監控一般工作負載特性和系統指標，例如FreeableMemory透過 Amazon CloudWatch 或 Amazon RDS主控台，以確認有足夠的記憶體進行連線。監控高CPUUtilization值，這可能表示需要調整。避免設定max_connections過高，因為它可能會影響記憶體用量，並可能間接影響 I/O。請記住，每個連線都會耗用記憶體。若要找到正確的平衡，請緩慢增加max_connections並查看它如何影響您的系統。留意效能較慢或更高CPU用量的跡象。檢查您的應用程式是否仍然運作良好。使用 Aurora 中的僅供讀取複本等功能來分發僅供讀取流量並減少主要執行個體的負載。根據觀察到的使用模式max_connections定期檢閱和調整，以確保在給定的資源限制內獲得最佳的資料庫效能。

調整自動清空參數

Amazon RDS for PostgreSQL 資料庫和 Aurora PostgreSQL 相容需要定期維護，稱為清空。Autovacuum 是內建 PostgreSQL 公用程式，可移除過時或不必要的資料，以釋放資料庫中的空間。自動清空程序會定期在背景執行VACUUM命令。

調整自動清空設定是維持 Amazon RDS for PostgreSQL 或 Aurora PostgreSQL 相容資料庫系統的效能、穩定性和可用性的關鍵步驟。透過調整自動清空參數以符合工作負載和資料庫大小，您可以最佳化自動清空程序的效能，並減少其對系統資源的影響，進而改善資料庫的整體運作狀態。

除了調整自動清空設定之外，使用 Amazon RDS 和 Aurora 中可用的工具和指標來監控資料庫及其元件的效能也很重要。透過監控效能指標，例如 bloat、可用空間和查詢執行時間，您可以在潛在問題變成嚴重問題之前加以識別，並採取適當的動作來解決這些問題。

本節討論下列自動清空主題和參數：

- VACUUM 和 ANALYZE 命令
- 檢查 bloat
- 自動清空
- 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

如需自動清空的詳細資訊，請參閱下列連結：

- 了解 Amazon RDS for PostgreSQL 環境中的自動清空（部落格文章）
- 在 Amazon RDS for PostgreSQL (Amazon Word 文件）上使用 PostgreSQL 自動清空 RDS
- 在 Amazon RDS for PostgreSQL 和 Amazon Aurora PostgreSQL 中平行清除（部落格文章）

VACUUM 和 ANALYZE 命令

VACUUM garbage-collects 並選擇性地分析資料庫。對於大多數應用程式，讓自動清空常駐程式執行清空已足夠。不過，有些管理員可能會想要修改自動清空的資料庫參數，或使用可根據排程器執行的手動受管VACUUM命令來補充或取代常駐程式的活動。

VACUUM 會回收無效元組佔用的儲存體。在標準 PostgreSQL 操作中，當 更新刪除或淘汰了組群時，它們在執行VACUUM操作之前不會從資料表中實際移除。因此，我們建議您VACUUM定期執行，尤其是經常更新的資料表。

調校VACUUM參數在 Amazon RDS for PostgreSQL 和 Aurora PostgreSQL 相容中尤其重要，因為這些受管資料庫服務與自我管理的 PostgreSQL 資料庫具有不同的特性。這些差異可能會影響真空操作的效能。調校VACUUM參數對於最佳化資源的使用至關重要，並確保真空操作不會對資料庫系統的效能和可用性產生負面影響。

以下是您可以在 Aurora PostgreSQL 相容和 Amazon RDS for PostgreSQL 中與 VACUUM命令搭配使用的一些參數：

- FULL
- FREEZE
- VERBOSE
- ANALYZE
- DISABLE_PAGE_SKIPPING
- table_name
- column_name

VACUUM ANALYZE 會為每個選取的資料表執行 VACUUM 操作，然後執行 ANALYZE操作。它提供了一種執行例行維護的有效方法。

在沒有 FULL選項的情況下使用 VACUUM命令，可回收空間以供重複使用。它不需要資料表上的專屬鎖定，因此您可以在標準讀取和寫入操作期間執行此命令。不過，在大多數情況下， 命令不會將額外的空間傳回作業系統，而是讓它在相同的資料表內重複使用。 會將資料表的完整內容VACUUM FULL重寫到新的磁碟檔案中，而不會有額外的空間，並允許未使用的空間傳回作業系統。此表單速度較慢，且每個資料表上都需要ACCESS EXCLUSIVE鎖定。

如需這些參數的完整資訊，請參閱 PostgreSQL 文件。

在 Aurora 和 Amazon RDS 中，自動清空是常駐程式（背景公用程式）程序，會定期執行 VACUUM和 ANALYZE命令，以清除資料庫和伺服器中的備援資料。即使您依賴自動清空，仍建議您檢閱和調整下列各節中討論的自動清空設定，以確保最佳效能。

檢查 bloat

下列 SQL 查詢會檢查 XML 結構描述中的每個資料表，並識別浪費磁碟空間的無效資料列（組）：

```
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;
```

如果此查詢傳回高百分比 (pfrag) 的無效元組，您可以使用 VACUUM命令來回收空間。

若要監控交易前後的資料大小，請在連線至特定資料庫後，在 Shell 中執行下列查詢：

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

autovacuum

您可以使用autovacuum組態參數全域設定自動清空，也可以將pg_class資料表的資料autovacuum_enabled欄設定為特定資料表的 true或 false ，依每個資料表進行變更。

當您在資料表上啟用自動清空時，資料庫伺服器會定期掃描資料表是否有無效資料列和組群，並在背景中移除這些資料，而不需要資料庫管理員的任何介入。這有助於保持資料表較小、改善查詢效能，並減少備份的大小。

AWS CLI 語法

下列命令autovacuum會針對特定資料庫參數群組啟用 。此變更適用於使用 參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態 （如果您設定 ，會立即套用變更ApplyMethod=immediate)

預設值：已啟用

您也可以使用 psql 停用或啟用特定資料表上的自動清空：

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

太多的清空可能會影響效能，因此監控自動清空程序的效能以及資料庫的效能，並視需要調整設定非常重要。

範例

您的 PostgreSQL 資料庫有一個接收大量寫入和刪除操作的資料表。如果沒有自動清空，此資料表最終會填滿無效資料列 （也就是說，已標記為刪除但尚未從資料表中實際移除的資料列）。這些無效資料列會佔用磁碟上的空間、減慢查詢速度，並增加備份的大小。您可以在資料表上啟用自動清空，以自動掃描無效資料列，並移除它們以緩解這些問題。

autovacuum_work_mem

autovacuum_work_mem 是 PostgreSQL 組態參數，可控制自動清空程序執行清空或分析等資料表維護任務時所使用的記憶體量。

在 Aurora 和 Amazon RDS 中，您可以調整 的值autovacuum_work_mem以最佳化效能。

AWS CLI 語法

下列命令autovacuum_work_mem 會針對特定資料庫參數群組啟用 。此變更適用於使用 參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態（如果您設定，會立即套用變更ApplyMethod=immediate)

預設值：Aurora PostgreSQL 相容中的 GREATEST({DBInstanceClassMemory/32768},131072) KB、Amazon RDS for PostgreSQL 中的 64 MB。不過，預設值可能會因您使用的特定 Amazon RDS 或 Aurora 版本而有所不同。

範例

您的 Amazon RDS for PostgreSQL 資料庫有一個經常更新的大型資料表。隨著時間的推移，您注意到資料庫變慢，而且您懷疑自動清空花了太長時間才能完成。

調查過程中，您會檢查系統日誌、使用 pg_stat_activity 檢視來查看目前正在執行的查詢和程序、檢查pg_stat_user_tables檢視來查看每個資料表的統計資料、使用 pg_settings 檢視來比較 的值autovacuum_work_mem與系統上可用的記憶體，以及監控尖峰的記憶體用量。收集此資訊後，您可以將 autovacuum_work_mem設定為工作負載所需的最佳值。若要在記憶體用量和效能之間找到適當的平衡，您可以決定將其設定為系統上可用記憶體的四分之一。變更值後，您可以監控資料庫的效能，並可能發現自動清空完成的速度比以前快得多，而且資料庫的整體執行速度更快。

autovacuum_naptime

autovacuum_naptime 參數控制自動清空程序的連續執行之間的時間間隔。Amazon RDS for PostgreSQL 的預設值為 15 秒，Aurora PostgreSQL 相容則為 5 秒。

例如，假設您的 Amazon RDS for PostgreSQL 資料庫有一個接收大量寫入和刪除操作的資料表。如果您保留預設設定，頻繁的自動清空掃描將破壞此高度交互的資料表。如果您將此參數設定為高值，則連續掃描之間的間隔會較長，而無效資料列的移除頻率會較低。

您可以使用 autovacuum_naptime 管理真空程序導致的負載，特別是如果您的忙碌伺服器已經具有高 CPU 或 I/O 負載。您設定午休時間的時間越長，自動清空執行頻率就越低，這會減少伺服器上的負載。不過，autovacuum_naptime將 設定為非常高的值可能會導致 PostgreSQL 資料表增長，並累積無效資料列，進而降低效能。建議您監控自動清空程序的效能，並視需要調整autovacuum_naptime設定。

AWS CLI 語法

下列命令autovacuum_naptime會針對特定資料庫參數群組變更。此變更適用於使用 參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態 （ 如果您設定 ，會立即套用變更ApplyMethod=immediate)

預設值：15 秒 (Amazon RDS for PostgreSQL)、5 秒 (Aurora PostgreSQL 相容)

autovacuum_max_workers

autovacuum_max_workers 參數控制自動清空程序可以建立的工作者程序數目上限。每個工作者程序都負責清空或分析單一資料表。

例如，假設您有一個大型資料庫，其中包含許多經常更新和刪除的資料表。如果您autovacuum_max_workers將 設定為諸如 1 的低值，一次只能清空一個資料表，而且需要更長的時間才能清除所有資料表。如果您autovacuum_max_workers將 設定為高值，例如 8，則最多可以同時清空八個資料表。這可讓包含許多資料表的資料庫的清理程序更快。

AWS CLI 語法

下列命令autovacuum_max_workers會針對特定資料庫參數群組變更。此變更適用於使用 參數群組的所有執行個體或叢集。

```
# 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"
```

類型：靜態（套用變更需要重新啟動）

預設值：GREATEST(DBInstanceClassMemory/64371566592,3)工作者

增加autovacuum_max_workers設定可能會增加伺服器上的負載，如果您沒有足夠的資源，可能會影響效能。最佳設定取決於資料庫的特定需求、其大小及其包含的資料表數量。建議您嘗試不同的值並監控效能，以尋找最適合使用案例的設定。

autovacuum_vacuum_scale_factor

autovacuum_vacuum_scale_factor 組態參數會控制自動清空程序在清空資料表時的積極程度。

真空比例因數是資料表中必須修改的元組總數的一小部分，之後自動清空才會清除資料表。預設值為 0.1（亦即必須修改 10% 的組分）。例如，如果資料表具有 1，000，000 個元組，且其中 100，000 個元組標記為無效或刪除，則自動清空會清空資料表，取決於 的值autovacuum_vacuum_threshold作為控制因素。

參數autovacuum_vacuum_scale_factor可協助您控制真空程序執行的頻率。如果資料表收到許多寫入操作，您可能想要降低真空比例係數，以便自動清空執行頻率更高，並保持資料表較小。相反地，如果資料表收到很少的寫入操作，您可能想要提高真空比例係數，以便自動清空執行頻率較低，並節省資源。

AWS CLI 語法

下列命令autovacuum_vacuum_scale_factor會針對特定資料庫參數群組變更。此變更適用於使用參數群組的所有執行個體或叢集。

```
# 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
```

類型：動態（如果您設定，會立即套用變更ApplyMethod=immediate)

預設值：0.1

autovacuum_vacuum_scale_factor 參數可與 autovacuum_vacuum_threshold、autovacuum_vacuum_cost_limit, and autovacuum_naptime 參數搭配使用。如需此參數的詳細資訊，請參閱 AWS 部落格文章了解 Amazon RDS for PostgreSQL 環境中的自動清空。

autovacuum_vacuum_threshold

autovacuum_vacuum_threshold 參數控制在自動清空之前，資料表上必須執行的組群更新或刪除操作數目下限。此設定有助於防止對這些操作速率不高的資料表進行不必要的清空。對於 Amazon RDS for PostgreSQL 和 Aurora PostgreSQL 相容，預設值為 50，即 PostgreSQL 引擎預設值。

例如，假設您的資料表有 100，000 個資料列，且autovacuum_vacuum_threshold設定為 50。如果資料表僅收到 49 個更新或刪除，則自動清空不會將其清空。如果資料表收到 50 個以上的更新或刪除，自動清空會將其清空，這取決於資料表資料列數目autovacuum_vacuum_scale_factor乘以作為控制因素的值。

設定此參數太高可能會導致資料表增長和無效資料列累積，這可能會影響效能。

AWS CLI 語法

下列命令autovacuum_vacuum_threshold會針對特定資料庫參數群組變更。此變更適用於使用 參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態（如果您設定，會立即套用變更ApplyMethod=immediate)

預設值：50 個操作

autovacuum_vacuum_threshold 參數可與 autovacuum_vacuum_scale_factor、autovacuum_vacuum_cost_limit, and autovacuum_naptime 參數搭配使用。最佳設定取決於資料庫和資料表大小的特定需求。

如需此參數的詳細資訊，請參閱 AWS 部落格文章了解 Amazon RDS for PostgreSQL 環境中的自動清空。

autovacuum_analyze_scale_factor

autovacuum_analyze_scale_factor 參數控制分析時自動清空程序的積極程度（收集資料表中資料分佈的統計資料）。

自動清空程序會使用此參數，根據資料表中的元組數來計算閾值。如果組合插入、更新或刪除的數量超過此閾值，自動清空會分析資料表。對於 Amazon RDS for PostgreSQL 和 Aurora PostgreSQL 相容，預設值為 0.05（即必須修改 5% 的元組）。

例如，假設您的資料表有 1，000，000 個元組，而您保持autovacuum_analyze_scale_factor預設值為 0.05。如果資料表收到 50，000 個以上的更新或刪除，則根據autovacuum_analyze_threshold值自動清空它，並新增資料表列數作為控制因素。

AWS CLI 語法

下列命令autovacuum_analyze_scale_factor會針對特定資料庫參數群組變更。此變更適用於使用 參數群組的所有執行個體或叢集。

```
# 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
```

類型：動態 （ 如果您設定 ，會立即套用變更ApplyMethod=immediate)

預設值：0.05 (5%)

查詢規劃工具收集統計資料以做出明智決策至關重要，例如如何存取資料以及如何組織資料，因此建議您監控自動清空程序的效能，並視需要調整設定，以確保統計資料為最新狀態。

autovacuum_analyze_scale_factor 參數可與 autovacuum_analyze_threshold、autovacuum_analyze_cost_limit, and autovacuum_naptime 參數搭配使用。最佳設定取決於資料庫和資料表大小的特定需求，以及更新的頻率。如需此參數的詳細資訊，請參閱 AWS 部落格文章了解 Amazon RDS for PostgreSQL 環境中的自動清空。

autovacuum_analyze_threshold

autovacuum_analyze_threshold 參數類似於 autovacuum_vacuum_threshold。它控制在自動清空分析資料表之前，必須發生的組合插入、更新或刪除數目下限。此設定有助於防止對這些操作速率不高的資料表進行不必要的清空。對於 Amazon RDS for PostgreSQL 和 Aurora PostgreSQL 相容，預設值為 50，即 PostgreSQL 引擎預設值。

例如，假設您的資料表有 100，000 個資料列，且autovacuum_analyze_threshold預設值維持在 50。如果資料表僅收到 49 個插入、更新或刪除，則自動清空不會分析它。如果資料表收到 50 個以上的插入、更新或刪除，則自動清空會對其進行分析，並將 的值autovacuum_analyze_scale_factor乘以資料表資料列數目作為控制因素。

AWS CLI 語法

下列命令autovacuum_analyze_threshold會針對特定資料庫參數群組變更。此變更適用於使用參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態（如果您設定，會立即套用變更ApplyMethod=immediate)

預設值：50 個操作

此參數可與 autovacuum_analyze_scale_factor 參數搭配使用，因此當您設定自動清空時，請將這兩個設定都納入考量。

查詢規劃工具必須收集統計資料，才能做出明智的決策，例如如何存取資料以及如何組織資料。設定autovacuum_analyze_threshold過高可能會導致統計資料過時，導致效能不佳。建議您監控自動清空程序的效能，並視需要調整設定。

如需此參數的詳細資訊，請參閱 AWS 部落格文章了解 Amazon RDS for PostgreSQL 環境中的自動清空。

autovacuum_vacuum_cost_limit

autovacuum_vacuum_cost_limit 參數控制自動清空工作者可以取用的 CPU 和 I/O 資源量。

限制自動清空程序的資源用量有助於防止它們耗用過多的 CPU 或磁碟 I/O，這可能會影響在相同系統上執行的其他查詢的效能。參數會指定成本限制，這是工作者在必須暫停之前可以執行的工作單位，並檢查是否仍在限制之下。例如，如果參數設定為 2，000，工作者可以在暫停之前處理 2，000 個單位的工作。

您可以在 PostgreSQL 工作階段中使用 SET命令或使用 AWS CLI 命令來設定 autovacuum_vacuum_cost_limit 參數。

AWS CLI 語法

下列命令autovacuum_vacuum_cost_limit會針對特定資料庫參數群組變更。此變更適用於使用參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態 （如果您設定 ，會立即套用變更ApplyMethod=immediate)

預設值：工作GREATEST({log(DBInstanceClassMemory/21474836480)*600},200)單位

如果您將值設定為autovacuum_vacuum_cost_limit過高，自動清空程序可能會耗用太多資源，並拖慢其他查詢。如果您將其設定為過低，自動清空程序可能無法回收足夠的空間，這會導致資料表隨著時間的推移變大。找到適用於您系統的正確平衡至關重要。

此參數只會影響自動清空程序，不會影響手動VACUUM命令。此外，它僅適用於 的自動清空程序VACUUM，但不適用於 ANALYZE。

調整記錄參數

調整 PostgreSQL 中的記錄參數有助於確保您收集正確的資訊，而不會產生大量記錄，使系統不堪重負。

最佳化記錄參數對於平衡記錄詳細資料與系統效能和磁碟使用量至關重要。您可以自訂下列記錄參數，在記錄檔中擷取適當的詳細資料層級、診斷問題，以及有效調查事件，同時將對系統效能和磁碟使用的影響降到最低。

- RDS.force_ 自動吸塵記錄
- rds.force_ 管理 _ 日誌級別
- 日誌持續時間
- 日誌 _ 分鐘持續時間 _ 聲明
- 日誌錯誤詳細程度
- 日誌聲明
- 日誌聲明統計
- 日誌 _ 分鐘錯誤聲明
- 日誌分鐘消息
- 日誌臨時文件
- 日誌連接
- 日誌解散

這些參數將在以下各節中更詳細地討論。

> **Warning**
>
> 這些參數的最佳設定取決於您組織的原則和合規性需求。不過，啟用記錄參數可能會產生大量的記錄檔和訊息，這些記錄檔和訊息可能會耗盡儲存空間並影響效能，尤其是對於忙碌的資料庫而言。我們建議您仔細使用這些參數。例如，您可能會決定暫時啟用它們，以縮小執行緩慢 SQL 敘述句的問題範圍，並在監督期間結束時將其關閉。

RDS.force_ 自動吸塵記錄

rds.force_autovacuum_logging參數 (僅適用於 PostgreSQL 的 Amazon RDS) 可控制伺服器日誌中是否記錄自動真空動作。它的值是disabled、debug5、debug4、debug3、debug2、debug1、info、notice、warning、、error、預設值為 warning。

啟用時rds.force_autovacuum_logging，會記錄自動真空處理程序的所有動作，例如程序開始時間、結束時間以及它吸塵的列數。這有助於偵錯或疑難排解自動真空效能問題。

AWS CLI 語法

下列指令會rds.force_autovacuum_logging針對特定資料庫參數群組進行變更。此變更適用於使用參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態 (如果已設定，則會立即套用變更ApplyMethod=immediate)

預設值：warning

範例

您可以使用此rds.force_autovacuum_logging參數來分析自動真空在寫入速率非常高的表格上的效能。例如，如果您的資料表每秒收到大量的寫入和刪除作業，而您遇到效能降低的情況，您可以啟用此參數來記錄每次自動真空執行的開始和結束時間，並判斷已清除多少資料列。這可以提供有關 autovacuum 運行頻率，運行需要多長時間以及吸塵多少行的有價值信息。然後，您可以使用此資訊微調自動真空設定 (例如autovacuum_vacuum_scale_factor、)autovacuum_vacuum_threshold，以及最佳化效autovacuum_naptime能。

rds.force_admin_logging_level

rds.force_admin_logging_level參數 (僅適用 Amazon RDS for PostgreSQL) 可控制管理操作 (例如吸塵、分析和重新建立索引) 所產生的日誌中的詳細資料層級。它接受值debug5、debug4、debug3、debug2、debug1、log、、info、notice、warning、errorlogfata和off （預設值）。最佳設定取決於您的使用案例。例如，如果您要疑難排解問題，您可能想要將參數設定為偵錯層級。否則，您可以使用loginfo、或warning設定。

如果設定rds.force_admin_logging_level為debug1，則可以記錄重新建立索引作業的詳細資訊，例如開始和結束時間、處理的資料列數目，以及在處理期間發生的任何錯誤或警告。這可以提供有關重新建立索引程序如何執行的重要資訊，並協助您疑難排解發生的任何問題。

AWS CLI 語法

下列指令會rds.force_admin_logging_level針對特定資料庫參數群組進行變更。此變更適用於使用參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態 (如果已設定，則會立即套用變更ApplyMethod=immediate)

預設值：off

範例

您可以用rds.force_admin_logging_level來監視和分析大型資料庫中多個表格的管理作業效能。例如，假設您有一個包含許多表的大型數據庫，並且要通過定期對它們運行真空和分析操作來優化這些表的性能。將rds.force_admin_logging_level參數設定為info或log，您可以記錄每個作業的開始和結束時間，以及受影響的表格。您可以使用此資訊來追蹤不同表格之間的管理作業效能，並識別可能需要更頻繁或更積極維護的表格。

某些記錄層級會產生大量記錄檔和訊息，這些記錄檔和訊息可能會快速填滿磁碟空間，尤其是在資料庫忙碌的情況下。我們建議您仔細使用此參數，並在監視期結束時將其關閉。

log_duration

此log_duration參數可控制每個查詢的持續時間 (也就是執行所需的時間) 是否與查詢一起記錄。當您將此參數設定為時on，執行每個查詢所需的時間會隨著查詢文字一起包含在記錄輸出中。時間以毫秒為單位測量。

log_duration參數的主要使用案例是協助進行效能調整和疑難排解。藉由記錄每個查詢的持續時間，您可以識別執行時間最長的查詢，然後將您的精力集中在最佳化這些查詢上。這可協助您識別並修正效能瓶頸，並協助改善資料庫的整體效能。

AWS CLI 語法

下列指令會log_duration針對特定資料庫參數群組進行變更。此變更適用於使用參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態 (如果已設定，則會立即套用變更ApplyMethod=immediate)

預設值：off

範例

如果您懷疑特定查詢或查詢集導致效能問題，則可以使用此參數。藉由啟用log_duration參數並檢查記錄輸出，您可以查看哪些查詢執行時間最長，然後採取適當的動作，例如最佳化索引、新增索引或重新撰寫查詢。

啟用log_duration可以增加記錄輸出的音量。我們建議您僅在需要時使用它，並在標準操作期間將其關閉，以避免填滿存儲空間或使日誌難以閱讀。

log_min_duration_statement

此log_min_duration_statement參數可控制 SQL 敘述句在記錄之前執行的時間下限 (以毫秒為單位)。

此參數可協助您識別可能導致效能問題的長時間執行查詢。您可以將其設定為臨界值 (特定工作負載視為太長的執行階段)，以擷取超過該臨界值的查詢，並識別潛在的效能瓶頸。如需範例使用案例，請參閱本指南稍後的 < 使用記錄參數擷取繫結變數 > 。

AWS CLI 語法

下列指令會log_min_duration_statement針對特定資料庫參數群組進行變更。此變更適用於使用參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態 (如果已設定，則會立即套用變更ApplyMethod=immediate)

預設值：1 (停用，這是 PostgreSQL 引擎的預設值)

範例

下列命令會記錄執行時間超過 100 毫秒的任何陳述式：

```
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

對於錯誤層級或更高層級記錄的錯誤和訊息，此log_error_verbosity參數可控制記錄輸出中包含的詳細資訊層級。此參數可以採用以下三個值之一：tersedefault、或verbose。

- terse只包含訊息文字、錯誤層級以及發生錯誤的檔案和行號。
- default包括訊息文字、錯誤層級、檔案和行號，以及錯誤內容。
- verbose包括訊息文字、錯誤層級、檔案和行號、錯誤內容，以及完整的錯誤訊息。

將參數設定為verbose以取得在非生產環境中進行疑難排解和偵錯的最詳細資訊。在生產環境中，您可能想要將其設定為terse，default或者只提供基本資訊，而且不會填滿太多詳細資料的記錄儲存空間。

AWS CLI 語法

下列指令會log_error_verbosity針對特定資料庫參數群組進行變更。此變更適用於使用參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態 (如果已設定，則會立即套用變更ApplyMethod=immediate)

預設值：default

log_statement

此log_statement 參數可控制伺服器記錄檔中記錄哪些 SQL 陳述式。參數可以採用下列其中一個值：

- none（默認）不記錄任何語句
- ddl僅記錄資料定義語言 (DDL) 陳述式，例如CREATE TABLE和 ALTER TABLE
- mod僅記錄資料修改陳述式，例如INSERTUPDATE、和 DELETE
- all記錄所有 SQL 敘述句

您可以使用log_statement參數控制寫入記錄檔的資訊量，方法是僅記錄與您的使用案例相關的特定陳述式類型。

AWS CLI 語法

下列指令會log_statement針對特定資料庫參數群組進行變更。此變更適用於使用參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態 (如果已設定，則會立即套用變更ApplyMethod=immediate)

預設值：none

範例

在實際執行環境中，您可能想log_statementddl要設定為僅記錄 DDL 陳述式，並追蹤對資料庫結構描述所做的任何變更。在開發環境中，您可能想要將參數設定為all以記錄所有陳述式，以協助偵錯和疑難排解。如需其他範例使用案例，請參閱本指南稍後的 < 使用記錄參數擷取繫結變數 >。

啟用log_statement可以增加記錄輸出量，因此請僅在需要時使用它，並將其關閉以避免填滿儲存空間或使記錄難以讀取。

我們建議您監控系統並調整此參數的值，以在記錄的資訊量與系統的儲存和效能之間取得適當的平衡。

log_statement_stats

此log_statement_stats參數控制是否要將與執行 SQL 敘述句相關聯的統計資料與敘述句一起記錄。當您開啟這個參數時，記錄輸出中會包含受影響的資料列數目、讀取和寫入的磁碟區塊數目，以及執行陳述式所需的時間等統計資料。

您可以使用此log_statement_stats參數來收集有關個別敘述句效能和整體工作負載的其他資訊。藉由記錄陳述式統計資料，您可以識別查詢效能和資源使用狀況中的模式，並使用這些資訊來最佳化資料庫並改善整體效能。

AWS CLI 語法

下列指令會log_statement_stats針對特定資料庫參數群組進行變更。此變更適用於使用參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態 (如果已設定，則會立即套用變更ApplyMethod=immediate)

預設值：off(PostgreSQL 引擎預設值)；在參數群組中使用 0 或 1 (布林值) 來設定

範例

您可以用log_statement_stats來分析特定查詢的行為、查看查詢如何使用 CPU、記憶體和磁碟 I/O 等資源，以及識別是否可以最佳化查詢。您也可以使用這個參數來查看是否經常讀取特定資料表 (這可能表示需要在特定資料行上建立索引)，或是否掃描資料表的頻率太頻繁。

啟用log_statement_stats可以增加記錄輸出量，因此請僅在需要時使用它，並將其關閉以避免填滿儲存空間或使記錄難以讀取。

log_min_error_statement

log_min_error_statement參數控制將記錄哪些導致錯誤的 SQL 敘述句。它的值為debug5debug4、debug3、debug2、debug1、info、notice、warning、error、log、fatal、和panic。這些設定會控制寫入記錄檔的資訊量，因此您可以篩選出嚴重性較低的訊息。您可以將此參數設定為較高的嚴重性層級，以減少記錄輸出量並更輕鬆地尋找重要訊息。

AWS CLI 語法

下列指令會log_min_error_statement針對特定的 DB 參數群組進行變更。此變更適用於使用參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態 (如果已設定，則會立即套用變更ApplyMethod=immediate)

預設值 PostgreSQL 設值error)

範例

log_min_error_statement當您疑難排解特定問題，並想要查看造成錯誤之 SQL 陳述式的錯誤訊息時，可能會考慮使用。

log_min_messages

此log_min_messages參數可控制寫入記錄的嚴重性層級。您可以將參數設定為debug5debug4debug3、debug2、debug1、info、notice、warning、、error、log、fatal、或panic。這些設定會控制寫入記錄檔的資訊量，因此您可以篩選出嚴重性較低的訊息。您可以將此參數設定為較高的嚴重性層級，以減少記錄輸出量並更輕鬆地尋找重要訊息。

AWS CLI 語法

下列指令會log_min_messages針對特定的 DB 參數群組進行變更。此變更適用於使用參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態 (如果已設定，則會立即套用變更ApplyMethod=immediate)

預設值：notice

範例

如果您要對特定問題進行疑難排解，並且想要查看所有錯誤訊息，可以將此參數設定error為僅記錄錯誤和更高層級的嚴重性問題。如果您有興趣監視系統的效能，您可以將info此參數設定為查看更詳細的資訊，例如每個陳述式的持續時間和統計資料。

設定log_min_messages為較高的嚴重性層級會減少記錄檔的數量。我們建議您根據特定使用案例、要檢查的記錄大小以及擁有的磁碟空間量來調整此參數。

log_temp_files

此log_temp_files參數可控制暫存檔案名稱和大小的記錄。它適用於為目的（例如排序，哈希和臨時查詢結果）創建的臨時文件。啟用此參數時，會在刪除時為每個暫存檔產生記錄項目，包括其檔案大小 (以位元組為單位)。您可以將此參數設定為 0 (零) 以完整記錄所有暫存檔資訊，或將超過該大小的記錄檔設定為正值 (如果未指定單位，則以 KB 為單位)。這對於識別和解決效能瓶頸或與臨時儲存相關的其他問題非常有用。依預設，會停用暫存檔案的記錄。

AWS CLI 語法

下列指令會log_temp_files針對特定的 DB 參數群組進行變更。此變更適用於使用參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態 (如果已設定，則會立即套用變更ApplyMethod=immediate)

預設值：-1 PostgreSQL 引擎)

範例

如果您懷疑系統使用過多的暫存儲空間，或者暫存檔沒有被正確刪除，則可以啟用此參數。當您檢查記錄輸出時，您可以看到產生暫存檔的查詢或作業，以及這些檔案的使用方式。

某些查詢或作業會建立大量暫存檔案，因此啟用log_temp_files可能會影響系統的整體效能。

log_connections

此log_connections參數可控制是否記錄與資料庫的連線。當您將此參數設定為時on，記錄檔會包含每個成功連線到資料庫的相關資訊，例如用戶端的 IP 位址、使用者名稱、資料庫名稱，以及連線的日期和時間。

您可以使用log_connections參數來監視和疑難排解與資料庫的連線。您可以查看連線到資料庫的使用者、應用程式、終端機和機器人、使用者連線的來源以及頻率。此資訊對於識別和解決連線相關問題或追蹤使用模式非常有用。

AWS CLI 語法

下列指令會log_connections針對特定的 DB 參數群組進行變更。此變更適用於使用參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態 (如果已設定，則會立即套用變更ApplyMethod=immediate)

預設值 PostgreSQL 設值off)

範例

如果您懷疑與資料庫的連線過多，或連線頻率太頻繁的特定使用者或 IP 位址會影響效能，則可以使用此參數。藉由啟用log_connections參數並檢查記錄輸出，您可以看到所有連線的編號和詳細資料。

啟用此參數之前，請檢查組織的政策，並考慮記錄 IP 位址和使用者名稱的安全隱患。

log_disconnections

此log_disconnections參數可控制與資料庫中斷連線的記錄。當您將此參數設定為時on，它會記錄每個工作階段結束的相關資訊，例如用戶端的 IP 位址、使用者名稱、資料庫名稱，以及中斷連線的日期和時間。

您可以使用log_disconnections參數來監視和疑難排解資料庫工作階段終止。您可以查看與資料庫中斷連線的使用者、應用程式、終端機和機器人、時間和原因。例如，您可以檢閱未預期的終止，例如當機或系統管理員啟動的中斷連線。此資訊對於識別和解決與中斷連線或追蹤使用模式相關的問題非常有用。

AWS CLI 語法

下列指令會log_disconnections針對特定的 DB 參數群組進行變更。此變更適用於使用參數群組的所有執行個體或叢集。

```
# 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"
```

類型：動態 (如果已設定，則會立即套用變更ApplyMethod=immediate)

預設值 PostgreSQL 設值off)

範例

log_disconnections如果您懷疑有太多使用者與資料庫中斷連線，或特定使用者或 IP 位址中斷連線的頻率過於頻繁，則可以使用此功能。藉由啟用log_disconnections參數並檢查記錄輸出，您可以查看所有中斷連線的編號和詳細資料，包括斷線前發生的人員、時間，以及是否有任何錯誤發生。

啟用此參數之前，請檢查組織的政策，並考慮記錄 IP 位址和使用者名稱的安全隱患。

使用記錄參數擷取繫結變數

在 PostgreSQL 中擷取繫結變數的典型使用案例是偵錯和效能調整 SQL 查詢。繫結變數可讓您在執行查詢時將資料傳遞至查詢。藉由擷取繫結變數，您可以看到傳遞至查詢的輸入資料，這可協助您識別資料或查詢效能的任何問題。捕獲綁定變量還可以幫助您審核輸入數據並檢測潛在的安全風險或惡意活動。

有幾種方法可以捕獲 PostgreSQL 的綁定變量。一種方法是啟用debug_print_parse和debug_print_rewritten參數。這會導致 PostgreSQL 將 SQL 語句的解析和重寫版本以及綁定的變量發送到服務器日誌。

- debug_print_parse：啟用此參數時，傳入查詢的剖析樹狀結構會列印至伺服器記錄檔。這對於瞭解查詢的結構和任何繫結參數的值非常有用。
- debug_print_rewritten：啟用此參數時，重寫的傳入查詢格式會列印至伺服器記錄檔。這對於瞭解查詢規劃工具如何解譯查詢以及任何繫結參數的值非常有用。

您可以在 Amazon RDS 和 Aurora 中使用兩個其他參數來擷取 PostgreSQL 資料庫中的繫結變數：

- log_min_duration_statement: 此參數會設定陳述式在記錄前的最短持續時間 (以毫秒為單位)。當陳述式所花費的時間超過指定的持續時間時，其繫結值會包含在記錄輸出中。

- log_statement：此參數控制要記錄哪些 SQL 敘述句。將此參數設定為all或 bind，以在記錄檔中包含繫結值。提高記錄層級會影響效能，因此建議您在疑難排解後還原變更。

您也可以使用pg_stat_statements擴充功能，它會為伺服器執行的所有 SQL 敘述句提供效能統計資料，包括查詢文字和繫結值。此擴展允許您使用 pgAdmin 或類似工具來監視和分析查詢性能。

另一個選項是使用該pg_bind_parameter_status()函數從準備好的語句中獲取綁定參數的值，或者使用該pg_get_parameter_status (paramname)函數來檢索特定運行時參數的狀態或值。

此外，您可以使用第三方工具 (例如 pgBadger) 來分析 PostgreSQL 記錄檔，並擷取繫結變數和其他資訊以供進一步分析。

調整複製參數

在 PostgreSQL 中，您可以使用邏輯複寫而不是以檔案為基礎的實體複寫，將資料變更從一個 PostgreSQL 資料庫複寫到另一個資料庫。邏輯複寫使用預寫記錄 (WAL) 來擷取變更，並支援選取的資料表或整個資料庫的複寫。

Amazon RDS for PostgreSQL 版和 Aurora PostgreSQL 相容兩者都支援邏輯複寫，因此您可以設定高可用性和可擴展的資料庫架構，以處理來自多個來源的讀取和寫入流量。這些服務會使用 pglogical (開放原始碼 PostgreSQL 延伸模組) 來實作邏輯複寫。

在 Aurora 和 Amazon RDS 中調整邏輯複寫對於實現最佳效能、可擴展性和可用性非常重要。您可以調整 pglogical 延伸模組中的參數，以管理邏輯複寫的效能。例如，您可以：

- 透過增加背景工作處理序的數目或調整其記憶體配置來改善複寫的效能。
- 調整來源資料庫與複本資料庫之間的同步處理頻率，以降低複寫延遲的風險。
- 調整背景工作處理序的記憶體和 CPU 配置，以最佳化資源的使用。
- 請確定複寫處理作業不會對來源資料庫的效能造成不當的影響。

您可以在 Aurora 和 Amazon RDS 中使用下列參數來控制和設定邏輯複寫：

- `max_replication_slots`設定可在伺服器上建立的最大複製插槽數目。複寫插槽是用於將 WAL 資料傳送至複本的複寫連線的具名永久保留區。
- `max_wal_senders`設置同時連接的 WAL 發送器進程的最大數量。WAL 發送方進程用於將 WAL 從主服務器流式傳輸到副本。
- `wal_sender_timeout`設置 WAL 發送者在放棄和重新連接之前等待複本響應的最大時間（以毫秒為單位）。
- `wal_receiver_timeout`設定複本在逾時之前等待來自主資料庫 WAL 資料的最大時間 (以毫秒為單位)。
- `log_replication_commands`，當設定為時`on`，會執行複製相關的 SQL 敘述句。

當您啟用`rds.logical_replication`參數 (將其設定為 1) 時，`wal_level`參數會設定為`logical`，這表示對資料庫所做的所有變更都會以可讀取並套用至複本的格式寫入 WAL。若要啟用邏輯複製，需要此設定。此設定也允許複寫`SELECT`陳述式。

設定`wal_level`為`logical`可增加寫入 WAL 的資料量，因此會增加寫入磁碟的資料量，這會影響系統效能。我們建議您在啟用邏輯複寫時考慮可用的磁碟空間和系統效能。

範例

您想要將資料從主要資料庫複製到次要資料庫，以供備份和災難復原之用。不過，次要資料庫具有大量的讀取作業，因此您想要確定複寫程序盡可能快速且有效率，而不會影響資料完整性。

Amazon RDS 和 Aurora 中邏輯複寫的預設值優先於效能的一致性，因此對於此使用案例來說，它們可能不是最佳選擇。若要針對速度和效率最佳化邏輯複製設定，您可以依照下列方式自訂參數：

- `max_replication_slots`從 10 (Amazon RDS 的預設值) 或 20 (Aurora 的預設值) 增加到 30，以因應 future 可能的成長和複寫需求。
- `max_wal_senders`從 10 (預設值) 增加到 20，以確保有足夠的 WAL 寄件者處理序可以滿足複寫需求。
- `wal_sender_timeout`從 30 秒 (預設值) 降低為 15 秒，以確保閒置的 WAL 寄件者處理程序能夠更快地終止，從而釋放資源以進行主動複寫。
- `wal_receiver_timeout`從 30 秒 (預設值) 減少到 15 秒，以確保閒置的 WAL 接收器處理程序更快終止，從而釋放資源以進行主動複製。
- `max_logical_replication_workers`從 4 (預設值) 增加到 8，以確保有足夠的邏輯複寫工作者處理序可以滿足複寫需求。

這些最佳化可提供更快速、更有效率的資料複製，同時維持資料完整性和安全性

例如，如果發生災難且主要資料庫變得無法使用，則由於最佳化的複寫程序，次要資料庫就已經擁有最新的可用資料。這將使您的業務運營能夠繼續提供關鍵服務，而不會中斷。

最佳實務

使用龐大的工作負載調整邏輯複寫可能是一項複雜的工作，這取決於各種因素，包括資料集的大小、複寫的資料表數目、複本數目以及可用的資源。以下是調整具有龐大工作負載的邏輯複寫的一些一般秘訣：

- 監視複寫延遲。複製延遲是主要伺服器與待命伺服器之間的時差。監控複寫延遲可協助您識別潛在瓶頸，並採取行動以改善複寫效能。您可以使用此`pg_current_wal_lsn()` 功能來檢查目前的複寫延遲。
- 調整沃爾瑪設置。`pg_logical`延伸功能使用 WAL 將變更從主要伺服器傳輸到待命伺服器。如果 WAL 設定未正確調整，複寫可能會變得緩慢且不可靠。確保將`max_wal_senders`和`max_replication_slots`參數設置為適當的值，具體取決於您的工作負載。

- 有一個索引策略。在主要伺服器上建立適當的索引可協助改善邏輯複寫的效能、減少主要伺服器上的 I/O，並減少系統的負載。
- 使用 parallel 複寫。使用 parallel 複寫可讓多個 parallel 背景工作者處理序複寫資料，有助於提高複寫速度。此功能適用 PostgreSQL 上版本。

後續步驟

針對 Amazon RDS for PostgreSQL 或 Aurora PostgreSQL 相容資料庫最佳化記憶體、複寫、自動真空和記錄參數之後，請考慮下列步驟以進一步改善資料庫的效能：

- 監控您的資料庫。使用內建的監視工具或協力廠商解決方案，隨著時間的推移追蹤資料庫效能。監控 CPU 使用率、磁碟 I/O、記憶體使用率和查詢執行階段等關鍵效能指標，以識別潛在的瓶頸和需要改善的領域。
- 連續調整參數。隨著工作負載的發展，請繼續監控和調整資料庫參數，以確保最佳效能。定期檢查系統記錄、錯誤訊息和效能指標，以識別新的調整機會。
- 實現緩存。使用快取來減少點擊資料庫的查詢數量。您可以使用 Memcached 或 Redis 等工具在應用程式層級實作快取，也可以使用 Amazon 為資料庫 ElastiCache 提供記憶體內快取。
- 最佳化您的查詢。設計不當的查詢會大幅影響資料庫效能。使用`EXPLAIN`和其他查詢調整工具來識別緩慢的查詢、對其進行最佳化，並消除任何不必要的查詢。

透過遵循這些準則，您可以優化 Aurora 或 Amazon RDS for PostgreSQL 資料庫的效能，並透過改善的資料庫效能、提高可靠性、減少停機時間、更佳的安全性和節省成本，確保其符合應用程式的需求。透過最佳化組態參數以符合您的工作負載，您可以確保資料庫有效率地執行，並有效運用資源，進而獲得更好的效能和回應速度更快的應用程式。此外，正確設定的參數可以降低發生錯誤和漏洞的可能性，進而提高可靠性和更好的安全性。在減少維護和停機時間，以及更好的整體用戶體驗和滿意度方面，這可以節省成本。

資源

- Amazon Aurora PostgreSQL 參數，第 1 部分：記憶體和查詢計劃管理 (部AWS 落格文章)
- Amazon Aurora PostgreSQL 參數，第 2 部分：複寫、安全和記錄 (部AWS 落格文章)
- Amazon Aurora PostgreSQL 參數，第 3 部分：優化器參數（ AWS 博客文章 ）
- Amazon Aurora PostgreSQL 參數，第 4 部分：ANSI 相容性選項 (部AWS 落格文章)
- 使用 Amazon Aurora (文件)AWS
- 使用適用 Amazon RDS for PostgreSQL (AWS 文件)
- 使用 Amazon RDS 上的 Performance Insights 來監控資料庫負載 (AWS 文件)
- 使用 Amazon CloudWatch 指標（ AWS 文檔 ）
- 狀態語 PostgreSQL ）

文件歷史紀錄

下表描述了本指南的重大變更。如果您想收到有關未來更新的通知，可以訂閱 RSS 摘要。

變更	描述	日期
關於記憶體和自動真空參數的更新資訊	更新了 random_page_cost 參數的描述；將遺失的單位新增至記憶體和自動真空參數的預設值；更新 max_connections 參數的 AWS CLI 語法。	2024年2月27日
更新有關的信息 autovacuum	更正了自動真空預設設定（啟用）。	2023 年 12 月 27 日
更新有關的信息 max_connections	更新 max_connections 區段，其中包含調整此參數的新指引。	2023 年 11 月 15 日
初次出版	—	2023 年 10 月 31 日

AWS 規範性指導詞彙表

以下是 AWS Prescriptive Guidance 所提供策略、指南和模式的常用詞彙。若要建議項目，請使用詞彙表末尾的提供意見回饋連結。

數字

7 R

將應用程式移至雲端的七種常見遷移策略。這些策略以 Gartner 在 2011 年確定的 5 R 為基礎，包括以下內容：

- 重構/重新架構 – 充分利用雲端原生功能來移動應用程式並修改其架構，以提高敏捷性、效能和可擴展性。這通常涉及移植作業系統和資料庫。範例：將您的內部部署 Oracle 資料庫遷移至 Amazon Aurora Postgre SQL相容版本。
- 平台轉換 (隨即重塑) – 將應用程式移至雲端，並引入一定程度的優化以利用雲端功能。範例：將您的內部部署 Oracle 資料庫遷移至 中的 Oracle 的 Amazon Relational Database Service (Amazon RDS) AWS 雲端。
- 重新購買 (捨棄再購買) – 切換至不同的產品，通常從傳統授權移至 SaaS 模型。範例：將您的客戶關係管理 (CRM) 系統遷移至 Salesforce.com。
- 主機轉換 (隨即轉移) – 將應用程式移至雲端，而不進行任何變更以利用雲端功能。範例：將您的現場部署 Oracle 資料庫遷移至 中EC2執行個體上的 Oracle AWS 雲端。
- 重新放置 (虛擬機器監視器等級隨即轉移) – 將基礎設施移至雲端，無需購買新硬體、重寫應用程式或修改現有操作。您可以將伺服器從內部部署平台遷移到相同平台的雲端服務。範例：遷移 Microsoft Hyper-V 應用程式 AWS。
- 保留 (重新檢視) – 將應用程式保留在來源環境中。其中可能包括需要重要重構的應用程式，且您希望將該工作延遲到以後，以及您想要保留的舊版應用程式，因為沒有業務理由來進行遷移。
- 淘汰 – 解除委任或移除來源環境中不再需要的應用程式。

A

ABAC

請參閱屬性型存取控制。

抽象服務

請參閱 受管服務。

ACID

請參閱原子、一致性、隔離、耐久性。

主動-主動式遷移

一種資料庫遷移方法，其中來源和目標資料庫保持同步 (透過使用雙向複寫工具或雙重寫入操作)，且兩個資料庫都在遷移期間處理來自連接應用程式的交易。此方法支援小型、受控制批次的遷移，而不需要一次性切換。它更靈活，但需要比主動-被動遷移更多的工作。

主動-被動式遷移

一種資料庫遷移方法，其中來源和目標資料庫保持同步，但只有來源資料庫處理來自連接應用程式的交易，同時將資料複寫至目標資料庫。目標資料庫在遷移期間不接受任何交易。

彙總函數

在一組資料列上操作並計算群組單一傳回值的SQL函數。彙總函數的範例包括 SUM和 MAX。

AI

請參閱人工智慧。

AIOps

請參閱人工智慧操作。

匿名化

在資料集中永久刪除個人資訊的程序。匿名化有助於保護個人隱私權。匿名資料不再被視為個人資料。

反模式

經常性問題的常用解決方案，其解決方案具有反效益、無效或效果不如替代方案。

應用程式控制

一種安全方法，僅允許使用核准的應用程式，以協助保護系統免受惡意軟體攻擊。

應用程式組合

有關組織使用的每個應用程式的詳細資訊的集合，包括建置和維護應用程式的成本及其商業價值。此資訊是產品組合探索和分析程序的關鍵，有助於識別要遷移、現代化和優化的應用程式並排定其優先順序。

人工智慧 (AI)

電腦科學領域，致力於使用運算技術來執行通常與人類相關的認知功能，例如學習、解決問題和識別模式。如需詳細資訊，請參閱什麼是人工智慧？

人工智慧操作 (AIOps)

使用機器學習技術解決操作問題、減少操作事件和人工干預以及提高服務品質的程序。如需如何在遷移策略AIOps中使用 AWS 的詳細資訊，請參閱 操作整合指南。

非對稱加密

一種加密演算法，它使用一對金鑰：一個用於加密的公有金鑰和一個用於解密的私有金鑰。您可以共用公有金鑰，因為它不用於解密，但對私有金鑰存取應受到高度限制。

原子、一致性、隔離、耐久性 (ACID)

一組軟體屬性，即使在出現錯誤、電源故障或其他問題的情況下，也能確保資料庫的資料有效性和操作可靠性。

屬性型存取控制 (ABAC)

根據使用者屬性 (例如部門、工作職責和團隊名稱) 建立精細許可的實務。如需詳細資訊，請參閱 AWS Identity and Access Management (IAM) 文件ABAC AWS中的 。

授權資料來源

您存放主要版本資料的位置，被視為最可靠的資訊來源。您可以將資料從授權資料來源複製到其他位置，以處理或修改資料，例如匿名、修訂或假名化資料。

可用區域

在 內的不同位置 AWS 區域 ，可隔離其他可用區域中的故障，並對相同區域中的其他可用區域提供經濟實惠的低延遲網路連線。

AWS 雲端採用架構 (AWS CAF)

的指導方針和最佳實務架構 AWS ，可協助組織制定有效率且有效的計劃，以成功地移至雲端。AWS CAF 將指導方針整理成六個重點領域：商業、人員、治理、平台、安全和操作。業務、人員和控管層面著重於業務技能和程序；平台、安全和操作層面著重於技術技能和程序。例如，人員層面針對處理人力資源 (HR)、人員配備功能和人員管理的利害關係人。為此， AWS CAF 提供人員開發、訓練和通訊的指導，協助組織成功採用雲端。如需詳細資訊，請參閱 AWS CAF網站和AWS CAF白皮書。

AWS 工作負載資格架構 (AWS WQF)

評估資料庫遷移工作負載、建議遷移策略並提供工作預估的工具。 AWS WQF 隨附於 AWS Schema Conversion Tool (AWS SCT)。它會分析資料庫結構描述和程式碼物件、應用程式程式碼、相依性和效能特性，並提供評估報告。

B

錯誤的機器人

旨在中斷或傷害個人或組織的機器人。

BCP

請參閱業務持續性規劃。

行為圖

資源行為的統一互動式檢視，以及一段時間後的互動。您可以搭配 Amazon Detective 使用行為圖表來檢查失敗的登入嘗試、可疑API的呼叫和類似的動作。如需詳細資訊，請參閱偵測文件中的行為圖中的資料。

大端序系統

首先儲存最高有效位元組的系統。另請參閱結尾。

二進制分類

預測二進制結果的過程 (兩個可能的類別之一)。例如，ML 模型可能需要預測諸如「此電子郵件是否是垃圾郵件？」等問題 或「產品是書還是汽車？」

Bloom 篩選條件

一種機率性、記憶體高效的資料結構，用於測試元素是否為集的成員。

藍/綠部署

一種部署策略，您可以在其中建立兩個不同但相同的環境。您可以在一個環境 （藍色） 中執行目前的應用程式版本，並在另一個環境 （綠色） 中執行新的應用程式版本。此策略可協助您快速復原，並將影響降至最低。

機器人

軟體應用程式，透過網際網路執行自動化任務，並模擬人類活動或互動。有些機器人很有用或很有幫助，例如在網際網路上為資訊編製索引的 Web 爬蟲程式。有些其他機器人，稱為不良機器人，旨在中斷或傷害個人或組織。

殭屍網路

受到惡意軟體感染且由單一方控制的機器人網路，稱為機器人繼承者或機器人運算子。殭屍網路是擴展機器人及其影響的最佳已知機制。

分支

程式碼儲存庫包含的區域。儲存庫中建立的第一個分支是主要分支。您可以從現有分支建立新分支，然後在新分支中開發功能或修正錯誤。您建立用來建立功能的分支通常稱為功能分支。當準備好發佈功能時，可以將功能分支合併回主要分支。如需詳細資訊，請參閱關於分支 (GitHub 文件）。

碎片存取

在特殊情況下，透過核准的程序，使用者可以快速存取 AWS 帳戶 他們通常沒有存取許可的 。如需詳細資訊，請參閱 Well-Architected 指南中的 AWS 實作碎片程序指標。

棕地策略

環境中的現有基礎設施。對系統架構採用棕地策略時，可以根據目前系統和基礎設施的限制來設計架構。如果正在擴展現有基礎設施，則可能會混合棕地和綠地策略。

緩衝快取

儲存最常存取資料的記憶體區域。

業務能力

業務如何創造價值 (例如，銷售、客戶服務或營銷)。業務能力可驅動微服務架構和開發決策。如需詳細資訊，請參閱在 AWS上執行容器化微服務白皮書的圍繞業務能力進行組織部分。

業務持續性規劃 (BCP)

一種解決破壞性事件 (如大規模遷移) 對營運的潛在影響並使業務能夠快速恢復營運的計畫。

C

CAF

請參閱AWS 雲端採用架構。

Canary 部署

版本向最終使用者緩慢且遞增的版本。當您有信心時，您可以部署新版本並完全取代目前的版本。

CCoE

請參閱雲端卓越中心。

CDC

請參閱變更資料擷取。

變更資料擷取 (CDC)

追蹤對資料來源 (例如資料庫表格) 的變更並記錄有關變更的中繼資料的程序。您可以使用 CDC進行各種用途，例如稽核或複寫目標系統中的變更，以維持同步。

混亂工程

故意引入故障或破壞性事件，以測試系統的彈性。您可以使用 AWS Fault Injection Service (AWS FIS) 來執行實驗，以對您的 AWS 工作負載造成壓力，並評估其回應。

CI/CD

請參閱持續整合和持續交付。

分類

有助於產生預測的分類程序。用於分類問題的 ML 模型可預測離散值。離散值永遠彼此不同。例如，模型可能需要評估影像中是否有汽車。

用戶端加密

在目標 AWS 服務 接收資料之前，在本機加密資料。

雲端卓越中心 (CCoE)

一個多學科團隊，可推動整個組織的雲端採用工作，包括開發雲端最佳實務、調動資源、制定遷移時間表以及領導組織進行大規模轉型。如需詳細資訊，請參閱 AWS 雲端 企業策略部落格上的CCoE文章。

雲端運算

通常用於遠端資料儲存和 IoT 裝置管理的雲端技術。雲端運算通常連接到邊緣運算技術。

雲端操作模型

在 IT 組織中，用於建置、成熟和最佳化一或多個雲端環境的操作模型。如需詳細資訊，請參閱建置您的雲端操作模型。

採用雲端階段

組織在遷移至 時通常會經歷的四個階段 AWS 雲端：

- 專案 – 執行一些與雲端相關的專案以進行概念驗證和學習用途
- 基礎 – 進行基礎投資以擴展雲端採用 （例如，建立登陸區域、定義 CCoE、建立操作模型）
- 遷移 – 遷移個別應用程式
- 重塑 – 優化產品和服務，並在雲端中創新

這些階段由 Stephen Orban 在部落格文章中定義： AWS 雲端 企業策略部落格上的邁向 Cloud-First 之旅和採用階段。如需有關它們如何與 AWS 遷移策略相關的詳細資訊，請參閱遷移準備指南。

CMDB

請參閱組態管理資料庫。

程式碼儲存庫

透過版本控制程序來儲存及更新原始程式碼和其他資產 (例如文件、範例和指令碼) 的位置。常見的雲端儲存庫包括 GitHub 或 Bitbucket Cloud。 每個版本的程式碼稱為分支。在微服務結構中，每個儲存庫都專用於單個功能。單一 CI/CD 管道可以使用多個儲存庫。

冷快取

一種緩衝快取，它是空的、未填充的，或者包含過時或不相關的資料。這會影響效能，因為資料庫執行個體必須從主記憶體或磁碟讀取，這比從緩衝快取讀取更慢。

冷資料

很少存取的資料，通常是歷史資料。查詢這類資料時，通常可接受慢查詢。將此資料移至效能較低且成本較低的儲存層或類別，可以降低成本。

電腦視覺 (CV)

AI 欄位???，使用機器學習來分析和擷取數位影像和影片等視覺化格式的資訊。例如， AWS Panorama 提供將 CV 新增至內部部署攝影機網路的裝置，而 Amazon SageMaker AI 則提供 CV 的影像處理演算法。

組態偏離

對於工作負載，組態會從預期狀態變更。這可能會導致工作負載變得不合規，而且通常是漸進和無意的。

組態管理資料庫 (CMDB)

儲存和管理有關資料庫及其 IT 環境的資訊的儲存庫，同時包括硬體和軟體元件及其組態。您通常會在遷移CMDB的產品組合探索和分析階段使用來自 的資料。

一致性套件

您可以組合的 AWS Config 規則和修補動作集合，以自訂您的合規和安全檢查。您可以使用 YAML 範本，將一致性套件部署為 AWS 帳戶 和 區域中或整個組織的單一實體。如需詳細資訊，請參閱 AWS Config 文件中的一致性套件。

持續整合和持續交付 (CI/CD)

自動化軟體版本程序的來源、建置、測試、預備和生產階段的程序。 CI/CD is commonly described as a pipeline. CI/CD可協助您自動化程序、提高生產力、改善程式碼品質，並更快速交付。如需詳細資訊，請參閱持續交付的優點。CD 也可表示持續部署。如需詳細資訊，請參閱持續交付與持續部署。

CV

請參閱電腦視覺。

D

靜態資料

網路中靜止的資料，例如儲存中的資料。

資料分類

根據重要性和敏感性來識別和分類網路資料的程序。它是所有網路安全風險管理策略的關鍵組成部分，因為它可以協助您確定適當的資料保護和保留控制。資料分類是 AWS Well-Architected Framework 中安全支柱的元件。如需詳細資訊，請參閱資料分類。

資料偏離

生產資料與用於訓練 ML 模型的資料之間有意義的變化，或輸入資料隨時間有意義的變更。資料偏離可以降低 ML 模型預測的整體品質、準確性和公平性。

傳輸中的資料

在您的網路中主動移動的資料，例如在網路資源之間移動。

資料網格

架構架構，提供分散式、分散式資料擁有權與集中式管理。

資料最小化

僅收集和處理嚴格必要資料的原則。在 中實作資料最小化 AWS 雲端 可以降低隱私權風險、成本和分析碳足跡。

資料周邊

AWS 環境中的一組預防性護欄，可協助確保只有信任的身分才能從預期的網路存取信任的資源。如需詳細資訊，請參閱在 上建置資料周邊 AWS。

資料預先處理

將原始資料轉換成 ML 模型可輕鬆剖析的格式。預處理資料可能意味著移除某些欄或列，並解決遺失、不一致或重複的值。

資料來源

在整個生命週期中追蹤資料的原始伺服器和歷史記錄的程序，例如資料的產生、傳輸和儲存方式。

資料主體

正在收集和處理資料的個人。

資料倉儲

支援商業智慧的資料管理系統，例如分析。資料倉儲通常包含大量歷史資料，通常用於查詢和分析。

資料庫定義語言 (DDL)

用於建立或修改資料庫中資料表和物件之結構的陳述式或命令。

資料庫操作語言 (DML)

用於修改 (插入、更新和刪除) 資料庫中資訊的陳述式或命令。

DDL

請參閱資料庫定義語言。

深度整體

結合多個深度學習模型進行預測。可以使用深度整體來獲得更準確的預測或估計預測中的不確定性。

深度學習

一個機器學習子領域，它使用多層人工神經網路來識別感興趣的輸入資料與目標變數之間的對應關係。

defense-in-depth

這是一種資訊安全方法，其中一系列的安全機制和控制項會在整個電腦網路中精心分層，以保護網路和其中資料的機密性、完整性和可用性。當您在 上採用此策略時 AWS，您可以在 AWS Organizations 結構的不同層新增多個控制項，以協助保護資源。例如，方法 defense-in-depth可能會結合多重要素驗證、網路分割和加密。

委派的管理員

在 中 AWS Organizations，相容的服務可以註冊 AWS 成員帳戶來管理組織的帳戶，並管理該服務的許可。此帳戶稱為該服務的委派管理員。如需詳細資訊和相容服務清單，請參閱 AWS Organizations 文件中的可搭配 AWS Organizations運作的服務。

部署

在目標環境中提供應用程式、新功能或程式碼修正的程序。部署涉及在程式碼庫中實作變更，然後在應用程式環境中建置和執行該程式碼庫。

開發環境

請參閱 環境。

偵測性控制

一種安全控制，用於在事件發生後偵測、記錄和提醒。這些控制是第二道防線，提醒您注意繞過現有預防性控制的安全事件。如需詳細資訊，請參閱在 AWS上實作安全控制中的偵測性控制。

開發值串流映射 (DVSM)

一種程序，用於識別和排定限制的優先順序，這些限制會對軟體開發生命週期中的速度和品質造成負面影響。 DVSM擴展了最初為精實生產實務設計的價值串流映射程序。它著重於透過軟體開發程序建立和移動價值所需的步驟和團隊。

數位分身

真實世界系統的虛擬呈現，例如建築物、工廠、工業設備或生產線。數位分身支援預測性維護、遠端監控和生產最佳化。

維度資料表

在星狀結構描述中，較小的資料表包含有關事實資料表中量化資料的資料屬性。維度資料表屬性通常是文字欄位或離散數字，其行為與文字相似。這些屬性通常用於查詢限制、篩選和結果集標籤。

災難

防止工作負載或系統在其主要部署位置中實現其業務目標的事件。這些事件可能是自然災難、技術故障或人為動作的結果，例如意外的錯誤設定或惡意軟體攻擊。

災難復原 (DR)

您用來將災難造成的停機時間和資料遺失降至最低的策略和程序。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的在雲端中復原工作負載的災難 AWS復原。

DML

請參閱資料庫操作語言。

領域驅動的設計

一種開發複雜軟體系統的方法，它會將其元件與每個元件所服務的不斷發展的領域或核心業務目標相關聯。Eric Evans 在其著作 Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003) 中介紹了這一概念。如需如何使用網域驅動設計搭配 strangler fig 模式的詳細資訊，請參閱使用容器和 Amazon API Gateway 逐步現代化舊版 Microsoft ASP.NET (ASMX) Web 服務。

DR

請參閱災難復原。

偏離偵測

追蹤與基準組態的偏差。例如，您可以使用 AWS CloudFormation 來偵測系統資源中的偏離，也可以使用 AWS Control Tower 來偵測登陸區域中可能會影響對控管要求合規性的變更。 https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-stack-drift.html

DVSM

請參閱開發值串流映射。

E

EDA

請參閱探索性資料分析。

EDI

請參閱電子資料交換。

邊緣運算

提升 IoT 網路邊緣智慧型裝置運算能力的技術。與雲端運算相比，邊緣運算可以減少通訊延遲並縮短回應時間。

電子資料交換 (EDI)

組織之間商業文件的自動交換。如需詳細資訊，請參閱什麼是電子資料交換。

加密

將純文字資料轉換為人類可讀取的運算程序。

加密金鑰

由加密演算法產生的隨機位元的加密字串。金鑰長度可能有所不同，每個金鑰的設計都是不可預測且唯一的。

端序

位元組在電腦記憶體中的儲存順序。大端序系統首先儲存最高有效位元組。小端序系統首先儲存最低有效位元組。

端點

請參閱 服務端點。

端點服務

您可以在虛擬私有雲端 (VPC) 中託管以與其他使用者共用的服務。您可以使用 建立端點服務，AWS PrivateLink 並將許可授予其他 AWS 帳戶 或 AWS Identity and Access Management (IAM) 委託人。這些帳戶或主體可以透過建立介面端點，私下連接到端點服務VPC。如需詳細資訊，請參閱《Amazon Virtual Private Cloud (AmazonVPC) 文件》中的建立端點服務。

企業資源規劃 (ERP)

可自動化和管理企業關鍵業務流程 （ 例如會計MES、 和 專案管理 ） 的系統。

信封加密

使用另一個加密金鑰對某個加密金鑰進行加密的程序。如需詳細資訊，請參閱 AWS Key Management Service (AWS KMS) 文件中的信封加密。

環境

執行中應用程式的執行個體。以下是雲端運算中常見的環境類型：

- 開發環境 – 執行中應用程式的執行個體，只有負責維護應用程式的核心團隊才能使用。開發環境用來測試變更，然後再將開發環境提升到較高的環境。此類型的環境有時稱為測試環境。
- 較低的環境 – 應用程式的所有開發環境，例如用於初始建置和測試的開發環境。

- 生產環境 – 最終使用者可以存取的執行中應用程式的執行個體。在 CI/CD 管道中，生產環境是最後一個部署環境。
- 較高的環境 – 核心開發團隊以外的使用者可存取的所有環境。這可能包括生產環境、生產前環境以及用於使用者接受度測試的環境。

epic

在敏捷方法中，有助於組織工作並排定工作優先順序的功能類別。epic 提供要求和實作任務的高層級描述。例如， AWS CAF安全特徵包括身分和存取管理、偵測控制、基礎設施安全、資料保護和事件回應。如需有關 AWS 遷移策略中的 Epic 的詳細資訊，請參閱計畫實作指南。

ERP

請參閱企業資源規劃。

探索性資料分析 (EDA)

分析資料集以了解其主要特性的過程。您可以收集或彙總資料，然後執行初始調查，以尋找模式、偵測異常狀況，以及檢查假設。 EDA 是透過計算摘要統計資料和建立資料視覺化來執行。

F

事實資料表

星狀結構描述中的中央資料表。它存放有關業務操作的量化資料。一般而言，事實資料表包含兩種類型的資料欄：包含量值的資料，以及包含維度資料表外部索引鍵的資料欄。

快速失敗

使用頻繁且增量測試來縮短開發生命週期的理念。這是敏捷方法的關鍵部分。

故障隔離界限

在 中 AWS 雲端，像是可用區域 AWS 區域、控制平面或資料平面等邊界，會限制故障的影響，並有助於改善工作負載的彈性。如需詳細資訊，請參閱AWS 故障隔離界限。

功能分支

請參閱分支。

特徵

用來進行預測的輸入資料。例如，在製造環境中，特徵可能是定期從製造生產線擷取的影像。

功能重要性

特徵對於模型的預測有多重要。這通常表示為數值分數，可透過各種技術計算，例如 Shapley 附加說明 (SHAP) 和整合漸層。如需詳細資訊，請參閱使用機器學習模型解譯能力 AWS。

特徵轉換

優化 ML 程序的資料，包括使用其他來源豐富資料、調整值、或從單一資料欄位擷取多組資訊。這可讓 ML 模型從資料中受益。例如，如果將「2021-05-27 00:15:37」日期劃分為「2021」、「五月」、「週四」和「15」，則可以協助學習演算法學習與不同資料元件相關聯的細微模式。

少量擷取提示

提供LLM少數範例，示範任務和所需輸出，然後再要求它執行類似的任務。此技術是內文學習的應用，其中模型會從內嵌在提示中的範例 (快照) 中學習。對於需要特定格式、推理或網域知識的任務，少量提示是有效的。另請參閱零鏡頭提示。

FGAC

請參閱精細存取控制。

精細存取控制 (FGAC)

使用多個條件來允許或拒絕存取請求。

閃切遷移

一種資料庫遷移方法，透過變更資料擷取使用連續資料複寫，以盡可能在最短的時間內遷移資料，而不是使用分階段方法。目標是將停機時間降至最低。

FM

請參閱基礎模型。

基礎模型 (FM)

大型深度學習神經網路，已在廣義和未標記資料的大量資料集上進行訓練。 FMs 能夠執行各種一般任務，例如了解語言、產生文字和影像，以及自然語言的交談。如需詳細資訊，請參閱什麼是基礎模型。

G

生成式 AI

經過大量資料訓練的 AI 模型子集，可使用簡單的文字提示建立新的內容和成品，例如影像、影片、文字和音訊。如需詳細資訊，請參閱什麼是生成式 AI。

地理封鎖

請參閱地理限制。

地理限制 (地理封鎖)

在 Amazon 中 CloudFront，此選項可防止特定國家/地區的使用者存取內容分發。您可以使用允許清單或封鎖清單來指定核准和禁止的國家/地區。如需詳細資訊，請參閱 CloudFront 文件中的限制內容的地理分佈。

Gitflow 工作流程

這是一種方法，其中較低和較高環境在原始碼儲存庫中使用不同分支。Gitflow 工作流程會被視為舊版，而以中繼為基礎的工作流程是現代、偏好的方法。

金色影像

系統或軟體的快照，做為部署該系統或軟體新執行個體的範本。例如，在製造中，黃金映像可用於在多個裝置上佈建軟體，並有助於提高裝置製造操作的速度、可擴展性和生產力。

綠地策略

新環境中缺乏現有基礎設施。對系統架構採用綠地策略時，可以選擇所有新技術，而不會限制與現有基礎設施的相容性，也稱為棕地。如果正在擴展現有基礎設施，則可能會混合棕地和綠地策略。

防護機制

高階規則，可協助管理跨組織單位 (OUs) 的資源、政策和合規。預防性防護機制會強制執行政策，以確保符合合規標準。其實作方式是使用服務控制政策和IAM許可界限。偵測性防護機制可偵測政策違規和合規問題，並產生提醒以便修正。它們是透過使用 AWS Config、Amazon AWS Security Hub、 GuardDuty AWS Trusted Advisor、Amazon Inspector 和自訂 AWS Lambda 檢查來實作。

H

HA

請參閱高可用性。

異質資料庫遷移

將來源資料庫遷移至使用不同資料庫引擎的目標資料庫 (例如，Oracle 至 Amazon Aurora)。異質遷移通常是重新架構工作的一部分，而轉換結構描述可能是一項複雜任務。AWS 提供有助於結構描述轉換的 AWS SCT。

高可用性 (HA)

工作負載在遇到挑戰或災難時持續運作的能力，無需介入。HA 系統的設計目的是自動容錯移轉、持續提供高品質的效能，以及處理不同的負載和故障，同時將效能影響降至最低。

歷史現代化

一種方法，用於現代化和升級操作技術 (OT) 系統，以更好地滿足製造業的需求。歷史資料是一種資料庫，用於從工廠中的各種來源收集和存放資料。

保留資料

從用於訓練機器學習模型的資料集中保留的歷史、標記資料的一部分。您可以使用保留資料，透過比較模型預測與保留資料來評估模型效能。

異質資料庫遷移

將您的來源資料庫遷移至共用相同資料庫引擎的目標資料庫（例如，Microsoft SQL Server 遷移至 Amazon RDS for SQL Server)。同質遷移通常是主機轉換或平台轉換工作的一部分。您可以使用原生資料庫公用程式來遷移結構描述。

熱資料

經常存取的資料，例如即時資料或最近的轉譯資料。此資料通常需要高效能儲存層或類別，才能提供快速的查詢回應。

修補程序

緊急修正生產環境中的關鍵問題。由於其緊迫性，修正程式通常在典型 DevOps 的發行工作流程之外建立。

超級護理期間

在切換後，遷移團隊在雲端管理和監控遷移的應用程式以解決任何問題的時段。通常，此期間的長度為 1-4 天。在超級護理期間結束時，遷移團隊通常會將應用程式的責任轉移給雲端營運團隊。

I

IaC

將基礎設施視為程式碼。

身分型政策

連接至一或多個IAM主體的政策，可定義其在 AWS 雲端 環境中的許可。

閒置應用程式

在 90 天內，平均 CPU和記憶體用量介於 5% 到 20% 的應用程式。在遷移專案中，通常會淘汰這些應用程式或將其保留在內部部署。

IIoT

請參閱工業物聯網。

不可變的基礎設施

為生產工作負載部署新基礎設施的模型，而不是更新、修補或修改現有的基礎設施。與可變基礎設施相比，不可避免的基礎設施本質上更一致、可靠且可預測。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的使用不可變基礎設施的部署最佳實務。

傳入（傳入）VPC

在 AWS 多帳戶架構中， VPC接受、檢查和路由來自應用程式外部的網路連線。AWS 安全參考架構建議設定具有傳入、傳出和檢查的網路帳戶VPCs，以保護應用程式與更廣泛的網際網路之間的雙向界面。

增量遷移

一種切換策略，您可以在其中將應用程式分成小部分遷移，而不是執行單一、完整的切換。例如，您最初可能只將一些微服務或使用者移至新系統。確認所有項目都正常運作之後，您可以逐步移動其他微服務或使用者，直到可以解除委任舊式系統。此策略可降低與大型遷移關聯的風險。

工業 4.0

由 Klaus Schwab 於 2016 年引進的術語，透過連線能力、即時資料、自動化、分析和 AI/ML 的進步，來指製造程序的現代化。

基礎設施

應用程式環境中包含的所有資源和資產。

基礎設施即程式碼 (IaC)

透過一組組態檔案來佈建和管理應用程式基礎設施的程序。IaC 旨在協助您集中管理基礎設施，標準化資源並快速擴展，以便新環境可重複、可靠且一致。

工業物聯網 (IIoT)

在製造業、能源、汽車、醫療保健、生命科學和農業等產業領域使用網際網路連線的感測器和裝置。如需詳細資訊，請參閱建置工業物聯網 (IIoT) 數位轉型策略。

檢查 VPC

在 AWS 多帳戶架構中，集中VPC管理 VPCs（在相同或不同的 中 AWS 區域)、網際網路和內部部署網路之間的網路流量檢查。AWS 安全參考架構建議設定具有傳入、傳出和檢查的網路帳戶 VPCs，以保護應用程式與更廣泛的網際網路之間的雙向界面。

物聯網 (IoT)

具有內嵌式感測器或處理器的相連實體物體網路，其透過網際網路或本地通訊網路與其他裝置和系統進行通訊。如需詳細資訊，請參閱什麼是 IoT？

可解釋性

機器學習模型的一個特徵，描述了人類能夠理解模型的預測如何依賴於其輸入的程度。如需詳細資訊，請參閱使用 的機器學習模型可解譯性AWS。

IoT

請參閱物聯網。

IT 資訊庫 (ITIL)

一組最佳實務，用於交付 IT 服務，並將這些服務與業務需求保持一致。 為 ITIL提供了基礎ITSM。

IT 服務管理 (ITSM)

與組織的設計、實作、管理和支援 IT 服務關聯的活動。如需整合雲端操作與ITSM工具的相關資訊，請參閱 操作整合指南。

ITIL

請參閱 IT 資訊庫。

ITSM

請參閱 IT 服務管理。

L

標籤型存取控制 (LBAC)

強制存取控制 (MAC) 的實作，其中使用者和資料本身都會獲得明確指派的安全標籤值。使用者安全標籤和資料安全標籤之間的交集決定使用者可以看到哪些資料列和資料欄。

登陸區域

登陸區域是架構良好的多帳戶 AWS 環境，可擴展且安全。這是一個起點，您的組織可以從此起點快速啟動和部署工作負載與應用程式，並對其安全和基礎設施環境充滿信心。如需有關登陸區域的詳細資訊，請參閱設定安全且可擴展的多帳戶 AWS 環境。

大型語言模型 (LLM)

預先訓練大量資料的深度學習 AI 模型。LLM 可以執行多個任務，例如回答問題、彙整文件、將文字翻譯為其他語言，以及完成句子。如需詳細資訊，請參閱什麼是 LLMs。

大型遷移

遷移 300 部或更多伺服器。

LBAC

請參閱標籤型存取控制。

最低權限

授予執行任務所需之最低許可的安全最佳實務。如需詳細資訊，請參閱 IAM 文件中的套用最低權限許可。

隨即轉移

請參閱 7 個 R。

小端序系統

首先儲存最低有效位元組的系統。另請參閱結尾。

LLM

請參閱大型語言模型。

較低的環境

請參閱 環境。

M

機器學習 (ML)

一種使用演算法和技術進行模式識別和學習的人工智慧。機器學習會進行分析並從記錄的資料 (例如物聯網 (IoT) 資料) 中學習，以根據模式產生統計模型。如需詳細資訊，請參閱機器學習。

主要分支

請參閱分支。

惡意軟體

旨在危及電腦安全或隱私權的軟體。惡意軟體可能會中斷電腦系統、洩露敏感資訊或取得未經授權的存取。惡意軟體的範例包括病毒、蠕蟲、勒索軟體、特洛伊木馬程式、間諜軟體和鍵盤記錄器。

受管服務

AWS 服務 可 AWS 操作基礎設施層、作業系統和平台，而且您可以存取端點來存放和擷取資料。Amazon Simple Storage Service (Amazon S3) 和 Amazon DynamoDB 是受管服務的範例。這些也稱為抽象服務。

製造執行系統 (MES)

一種軟體系統，用於追蹤、監控、記錄和控制生產程序，將原物料轉換為現場成品。

MAP

請參閱遷移加速計劃。

機制

建立工具、推動工具採用，然後檢查結果以進行調整的完整程序。機制是一種循環，可在操作時強化和改善自身。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的建置機制。

成員帳戶

除了屬於 組織一部分的管理帳戶 AWS 帳戶 之外的所有 AWS Organizations。一個帳戶一次只能是一個組織的成員。

MES

請參閱製造執行系統。

訊息佇列遙測傳輸 (MQTT)

根據發佈/訂閱模式的輕量型 machine-to-machine (M2M) 通訊協定，適用於資源受限的 IoT 裝置。

微服務

小型、獨立的服務，透過定義明確的 進行通訊，APIs通常由小型、獨立的團隊擁有。例如，保險系統可能包含對應至業務能力 (例如銷售或行銷) 或子領域 (例如購買、索賠或分析) 的微服務。微服務的優點包括靈活性、彈性擴展、輕鬆部署、可重複使用的程式碼和適應力。如需詳細資訊，請參閱使用無 AWS 伺服器服務整合微服務。

微服務架構

一種使用獨立元件來建置應用程式的方法，這些元件會以微服務形式執行每個應用程式程序。這些微服務使用輕量型 ，透過定義明確的界面進行通訊APIs。此架構中的每個微服務都可以進行更新、部署和擴展，以滿足應用程式特定功能的需求。如需詳細資訊，請參閱在 上實作微服務 AWS。

遷移加速計劃 (MAP)

提供諮詢支援、訓練和服務 AWS ，以協助組織建置強大的營運基礎以遷移至雲端，並協助抵銷遷移的初始成本。 MAP包含以系統化方式執行舊版遷移的遷移方法，以及一組工具，以自動化和加速常見的遷移案例。

大規模遷移

將大部分應用程式組合依波次移至雲端的程序，在每個波次中，都會以更快的速度移動更多應用程式。此階段使用從早期階段學到的最佳實務和經驗教訓來實作團隊、工具和流程的遷移工廠，以透過自動化和敏捷交付簡化工作負載的遷移。這是 AWS 遷移策略的第三階段。

遷移工廠

可透過自動化、敏捷的方法簡化工作負載遷移的跨職能團隊。遷移工廠團隊通常包括操作、業務分析師和擁有者、遷移工程師、開發人員和在衝刺中工作的 DevOps 專業人員。20% 至 50% 之間的企業應用程式組合包含可透過工廠方法優化的重複模式。如需詳細資訊，請參閱此內容集中的遷移工廠的討論和雲端遷移工廠指南。

遷移中繼資料

有關完成遷移所需的應用程式和伺服器的資訊。每種遷移模式都需要一組不同的遷移中繼資料。遷移中繼資料的範例包括目標子網路、安全群組和 AWS 帳戶。

遷移模式

可重複的遷移任務，詳細描述遷移策略、遷移目的地以及所使用的遷移應用程式或服務。範例：EC2使用 AWS Application Migration Service 重新託管遷移至 Amazon。

遷移產品組合評估 (MPA)

提供驗證商業案例以遷移至 的資訊的線上工具 AWS 雲端。 MPA提供詳細的產品組合評估（伺服器大小調整、定價、TCO比較、遷移成本分析）以及遷移規劃（應用程式資料分析和資料收集、應用程式分組、遷移優先順序和波規劃）。MPA 工具（需要登入）可供所有 AWS 顧問和APN合作夥伴顧問免費使用。

遷移就緒狀態評估 (MRA)

使用 取得組織雲端整備狀態的洞見、識別優缺點，以及建立行動計劃以消除已識別差距的程序 AWS CAF。如需詳細資訊，請參閱遷移準備指南。 MRA是AWS 遷移策略的第一個階段。

遷移策略

用來將工作負載遷移至 的方法 AWS 雲端。如需詳細資訊，請參閱本詞彙表中的 7 個 Rs 項目，並請參閱動員您的組織以加速大規模遷移。

機器學習 (ML)

請參閱機器學習。

現代化

將過時的 (舊版或單一) 應用程式及其基礎架構轉換為雲端中靈活、富有彈性且高度可用的系統，以降低成本、提高效率並充分利用創新。如需詳細資訊，請參閱 中的應用程式現代化策略 AWS 雲端。

現代化準備程度評定

這項評估可協助判斷組織應用程式的現代化準備程度；識別優點、風險和相依性；並確定組織能夠在多大程度上支援這些應用程式的未來狀態。評定的結果就是目標架構的藍圖、詳細說明現代化程序的開發階段和里程碑的路線圖、以及解決已發現的差距之行動計畫。如需詳細資訊，請參閱 中的評估應用程式的現代化準備 AWS 雲端程度。

單一應用程式 (單一)

透過緊密結合的程序作為單一服務執行的應用程式。單一應用程式有幾個缺點。如果一個應用程式功能遇到需求激增，則必須擴展整個架構。當程式碼庫增長時，新增或改進單一應用程式的功能也會變得更加複雜。若要解決這些問題，可以使用微服務架構。如需詳細資訊，請參閱將單一體系分解為微服務。

MPA

請參閱遷移產品組合評估。

MQTT

請參閱訊息佇列遙測傳輸。

多類別分類

一個有助於產生多類別預測的過程 (預測兩個以上的結果之一)。例如，機器學習模型可能會詢問「此產品是書籍、汽車還是電話？」 或者「這個客戶對哪種產品類別最感興趣？」

可變基礎設施

更新和修改生產工作負載現有基礎設施的模型。為了提高一致性、可靠性和可預測性， AWS Well-Architected Framework 建議使用不可變基礎設施做為最佳實務。

O

OAC

請參閱原始存取控制。

OAI

請參閱原始存取身分。

OCM

請參閱組織變更管理。

離線遷移

一種遷移方法，可在遷移過程中刪除來源工作負載。此方法涉及延長停機時間，通常用於小型非關鍵工作負載。

OI

請參閱 操作整合。

OLA

請參閱操作層級協議。

線上遷移

一種遷移方法，無需離線即可將來源工作負載複製到目標系統。連接至工作負載的應用程式可在遷移期間繼續運作。此方法涉及零至最短停機時間，通常用於關鍵的生產工作負載。

OPC-UA

請參閱開啟程序通訊 - Unified Architecture。

開放程序通訊 - Unified Architecture (OPC-UA)

工業自動化的 machine-to-machine(M2M) 通訊協定。OPC-UA 提供與資料加密、身分驗證和授權機制的互通性標準。

操作層級協議 (OLA)

釐清哪些功能 IT 群組承諾交付給彼此的協議，以支援服務層級協議 (SLA)。

操作準備度檢閱 (ORR)

問題及相關最佳實務的檢查清單，可協助您了解、評估、預防或減少事件和可能失敗的範圍。如需詳細資訊，請參閱 AWS Well-Architected Framework 中的操作就緒審核 (ORR)。

操作技術 (OT)

使用實體環境控制工業操作、設備和基礎設施的硬體和軟體系統。在製造中，整合 OT 和資訊技術 (IT) 系統是工業 4.0 轉型的關鍵重點。

操作整合 (OI)

在雲端中將操作現代化的程序，其中包括準備程度規劃、自動化和整合。如需詳細資訊，請參閱操作整合指南。

組織追蹤

由 建立的線索 AWS CloudTrail 會記錄 AWS 帳戶 組織中所有 的事件 AWS Organizations。在屬於組織的每個 AWS 帳戶 中建立此追蹤，它會跟蹤每個帳戶中的活動。如需詳細資訊，請參閱 文件中的 CloudTrail為組織建立追蹤。

組織變更管理 (OCM)

從人員、文化和領導角度管理重大、破壞性業務轉型的架構。 透過加速變革採用、解決轉型問題，以及推動文化和組織變革，OCM協助組織準備和轉換新系統和策略。在 AWS 遷移策略中，此架構稱為人員加速，因為雲端採用專案所需的變更速度。如需詳細資訊，請參閱 OCM指南。

原始存取控制 (OAC)

在 中 CloudFront，用於限制存取以保護您的 Amazon Simple Storage Service (Amazon S3) 內容的增強型選項。 OAC支援所有 S3 儲存貯體 AWS 區域中的所有伺服器端加密，以及使用 AWS KMS (SSE-KMS) 的動態PUT和對 S3 儲存貯體的DELETE請求。

原始存取身分 (OAI)

在 中 CloudFront，用於限制存取以保護您的 Amazon S3 內容的選項。當您使用 時OAI， 會 CloudFront 建立 Amazon S3 可驗證的委託人。已驗證的主體只能透過特定 CloudFront 分佈存取 S3 儲存貯體中的內容。另請參閱 OAC，它提供更精細和增強的存取控制。

ORR

請參閱操作整備檢閱。

OT

請參閱操作技術。

傳出 （傳出） VPC

在 AWS 多帳戶架構中， VPC會處理從應用程式內啟動的網路連線。AWS 安全參考架構建議設定您的網路帳戶進行傳入、傳出和檢查，VPCs以保護應用程式與更廣泛的網際網路之間的雙向界面。

P

許可界限

連接到IAM主體的IAM管理政策，以設定使用者或角色可以擁有的最大許可。如需詳細資訊，請參閱 IAM 文件中的許可界限。

個人身分識別資訊 (PII)

直接檢視或與其他相關資料配對時，可用來合理推斷個人身分的資訊。的範例PII包括名稱、地址和聯絡資訊。

PII

請參閱個人識別資訊。

手冊

一組預先定義的步驟，可擷取與遷移關聯的工作，例如在雲端中提供核心操作功能。手冊可以採用指令碼、自動化執行手冊或操作現代化環境所需的程序或步驟摘要的形式。

PLC

請參閱可程式設計邏輯控制器。

PLM

請參閱產品生命週期管理。

政策

可定義許可（請參閱身分型政策）、指定存取條件（請參閱資源型政策）或定義組織中所有帳戶的最大許可 AWS Organizations（請參閱服務控制政策）的物件。

混合持久性

根據資料存取模式和其他需求，獨立選擇微服務的資料儲存技術。如果您的微服務具有相同的資料儲存技術，則其可能會遇到實作挑戰或效能不佳。如果微服務使用最適合其需求的資料儲存，則可以更輕鬆地實作並達到更好的效能和可擴展性。如需詳細資訊，請參閱在微服務中啟用資料持久性。

組合評定

探索、分析應用程式組合並排定其優先順序以規劃遷移的程序。如需詳細資訊，請參閱評估遷移準備程度。

述詞

傳回 `true`或 的查詢條件`false`，通常位於`WHERE`子句中。

述詞下推

一種資料庫查詢最佳化技術，可在傳輸前篩選查詢中的資料。這可降低必須從關聯式資料庫擷取和處理的資料量，並改善查詢效能。

預防性控制

旨在防止事件發生的安全控制。這些控制是第一道防線，可協助防止對網路的未經授權存取或不必要變更。如需詳細資訊，請參閱在 AWS上實作安全控制中的預防性控制。

委託人

中可執行動作和存取資源 AWS 的實體。此實體通常是 AWS 帳戶、 IAM角色或 使用者的根使用者。如需詳細資訊，請參閱 IAM 文件中角色術語和概念中的主體。

依設計的隱私權

透過整個開發程序將隱私權納入考量的系統工程方法。

私有託管區域

容器，其中包含您希望 Amazon Route 53 如何回應一個或多個 內網域及其子網域的DNS查詢的相關資訊VPCs。如需詳細資訊，請參閱 Route 53 文件中的使用私有託管區域。

主動控制

旨在防止部署不合規資源的安全控制。這些控制項會在佈建資源之前對其進行掃描。如果資源不符合控制項，則不會佈建。如需詳細資訊，請參閱 AWS Control Tower 文件中的控制項參考指南，並參閱實作 安全控制項中的主動控制項。 AWS

產品生命週期管理 (PLM)

產品整個生命週期的資料和程序管理，從設計、開發和啟動，到成長和成熟，再到拒絕和移除。

生產環境

請參閱 環境。

可程式設計邏輯控制器 (PLC)

在製造中，高度可靠、適應性強的電腦，可監控機器並自動化製造程序。

提示鏈結

使用一個LLM提示的輸出做為下一個提示的輸入，以產生更好的回應。此技術用於將複雜任務分解為子任務，或反覆精簡或展開初步回應。它有助於提高模型回應的準確性和相關性，並允許更精細、個人化的結果。

擬匿名化

將資料集中的個人識別符取代為預留位置值的程序。假名化有助於保護個人隱私權。假名化資料仍被視為個人資料。

publish/subscribe (pub/sub)

一種模式，可啟用微型服務之間的非同步通訊，以提高可擴展性和回應能力。例如，在以微服務為基礎的 中MES，微服務可以將事件訊息發佈到其他微服務可以訂閱的頻道。系統可以新增新的微服務，而無需變更發佈服務。

Q

查詢計劃

一系列步驟，如指示，用於存取SQL關聯式資料庫系統中的資料。

查詢計劃迴歸

在資料庫服務優化工具選擇的計畫比對資料庫環境進行指定的變更之前的計畫不太理想時。這可能因為對統計資料、限制條件、環境設定、查詢參數繫結的變更以及資料庫引擎的更新所導致。

R

RACI 矩陣

請參閱負責、負責、諮詢、告知 (RACI)。

RAG

請參閱擷取增強的產生。

勒索軟體

一種惡意軟體，旨在阻止對計算機系統或資料的存取，直到付款為止。

RASCI 矩陣

請參閱負責、負責、諮詢、告知 (RACI)。

RCAC

請參閱資料列和資料欄存取控制。

僅供讀取複本

用於唯讀用途的資料庫複本。您可以將查詢路由至僅供讀取複本以減少主資料庫的負載。

重新架構師

請參閱 7 個 R。

復原點目標 (RPO)

自上次資料復原點以來可接受的時間上限。這會決定最後一個復原點與服務中斷之間可接受的資料遺失。

復原時間目標 (RTO)

服務中斷和服務還原之間的可接受延遲上限。

重構

請參閱 7 個 R。

區域

地理區域中的 AWS 資源集合。每個 AWS 區域 都獨立於其他 ，以提供容錯能力、穩定性和彈性。如需詳細資訊，請參閱指定 AWS 區域 您的帳戶可以使用哪些。

迴歸

預測數值的 ML 技術。例如，為了解決「這房子會賣什麼價格？」的問題 ML 模型可以使用線性迴歸模型，根據已知的房屋事實 (例如，平方英尺) 來預測房屋的銷售價格。

重新託管

請參閱 7 個 R。

版本

在部署程序中，它是將變更提升至生產環境的動作。

重新定位

請參閱 7 個 R。

轉換

請參閱 7 個 R。

回購

請參閱 7 個 R。

彈性

應用程式抵禦中斷或從中斷中復原的能力。在 中規劃彈性時，高可用性和災難復原是常見的考量 AWS 雲端。如需詳細資訊，請參閱AWS 雲端 復原。

資源型政策

附接至資源的政策，例如 Amazon S3 儲存貯體、端點或加密金鑰。這種類型的政策會指定允許存取哪些主體、支援的動作以及必須滿足的任何其他條件。

負責、負責、諮詢、知情 (RACI) 矩陣

定義所有涉及遷移活動和雲端操作之各方的角色和責任的矩陣。矩陣名稱衍生自矩陣中定義的責任類型：負責人 (R)、責任 (A)、已諮詢 (C) 和知情 (I)。支援 (S) 類型為選用。如果您包含支援，則矩陣稱為RASCI矩陣，如果您排除它，則稱為RACI矩陣。

回應性控制

一種安全控制，旨在驅動不良事件或偏離安全基準的補救措施。如需詳細資訊，請參閱在 AWS上實作安全控制中的回應性控制。

保留

請參閱 7 個 R。

淘汰

請參閱 7 個 R。

擷取增強產生 (RAG)

一種生成式 AI 技術，其中 LLM會在產生回應之前參考訓練資料來源以外的權威資料來源。例如，RAG模型可能會對組織的知識庫或自訂資料執行語意搜尋。如需詳細資訊，請參閱什麼是 RAG。

輪換

定期更新秘密的程序，讓攻擊者更難存取登入資料。

資料列和資料欄存取控制 (RCAC)

使用已定義存取規則的基本、彈性SQL表達式。 RCAC包含資料列許可和資料欄遮罩。

RPO

請參閱復原點目標。

RTO

請參閱復原時間目標。

執行手冊

執行特定任務所需的一組手動或自動程序。這些通常是為了簡化重複性操作或錯誤率較高的程序而建置。

S

SAML 2.0

許多身分提供者 (IdPs) 使用的開放標準。此功能會啟用聯合單一登入 (SSO)， AWS API讓使用者可以登入 AWS Management Console 或呼叫 操作，而不必IAM為您組織中的每個人建立 中的使用者。如需 SAML 2.0 型聯合的詳細資訊，請參閱 IAM 文件中的關於 SAML 2.0 型聯合。

SCADA

請參閱監督控制和資料擷取。

SCP

請參閱服務控制政策。

秘密

您以加密形式存放的 AWS Secrets Manager機密或限制資訊，例如密碼或使用者登入資料。它包含秘密值及其中繼資料。秘密值可以是二進位、單一字串或多個字串。如需詳細資訊，請參閱 Secrets Manager 文件中的 Secrets Manager 秘密中的什麼內容？。

依設計的安全性

透過整個開發程序將安全性納入考量的系統工程方法。

安全控制

一種技術或管理防護機制，它可預防、偵測或降低威脅行為者利用安全漏洞的能力。安全控制有四種主要類型：預防性、偵測性、回應性和主動性。

安全強化

減少受攻擊面以使其更能抵抗攻擊的過程。這可能包括一些動作，例如移除不再需要的資源、實作授予最低權限的安全最佳實務、或停用組態檔案中不必要的功能。

安全資訊和事件管理 (SIEM) 系統

結合安全資訊管理 (SIM) 和安全事件管理 (SEM) 系統的工具和服務。SIEM 系統會收集、監控和分析來自伺服器、網路、裝置和其他來源的資料，以偵測威脅和安全漏洞，並產生提醒。

安全回應自動化

預先定義和程式設計的動作，旨在自動回應或修復安全事件。這些自動化可做為偵測或回應式安全控制，協助您實作 AWS 安全最佳實務。自動化回應動作的範例包括修改VPC安全群組、修補Amazon EC2執行個體或輪換登入資料。

伺服器端加密

由接收資料的 AWS 服務 加密其目的地的資料。

服務控制政策 (SCP)

提供組織內所有帳戶許可的集中控制政策 AWS Organizations。 SCPs定義管理員可委派給使用者或角色之動作的護欄或設定限制。您可以使用 SCPs做為允許清單或拒絕清單，以指定允許或禁止的服務或動作。如需詳細資訊，請參閱 AWS Organizations 文件中的服務控制政策。

服務端點

URL 的進入點 AWS 服務。您可以使用端點，透過程式設計方式連接至目標服務。如需詳細資訊，請參閱 AWS 一般參考 中的 AWS 服務 端點。

服務層級協議 (SLA)

一份協議，闡明 IT 團隊承諾向客戶提供的服務，例如服務正常執行時間和效能。

服務層級指標 (SLI)

服務效能方面的測量，例如其錯誤率、可用性或輸送量。

服務層級目標 (SLO)

代表服務運作狀態的目標指標，由服務層級指標測量。

共同責任模式

一種模型，描述您與 共同 AWS 承擔的雲端安全與合規責任。 AWS 負責雲端的安全，而 負責雲端的安全。如需詳細資訊，請參閱共同責任模式。

SIEM

請參閱安全資訊和事件管理系統。

單一失敗點 (SPOF)

應用程式的單一關鍵元件中的故障，可能會中斷系統。

SLA

請參閱服務層級協議。

SLI

請參閱服務層級指示器。

SLO

請參閱服務層級目標。

split-and-seed 模型

擴展和加速現代化專案的模式。定義新功能和產品版本時，核心團隊會進行拆分以建立新的產品團隊。這有助於擴展組織的能力和服務，提高開發人員生產力，並支援快速創新。如需詳細資訊，請參閱《》中的階段式應用程式現代化方法 AWS 雲端。

SPOF

請參閱單一故障點。

星狀結構描述

使用一個大型事實資料表來存放交易或測量資料的資料庫組織結構，並使用一或多個較小的維度資料表來存放資料屬性。此結構專為資料倉儲或商業智慧用途而設計。

Strangler Fig 模式

一種現代化單一系統的方法，它會逐步重寫和取代系統功能，直到舊式系統停止使用為止。此模式源自無花果藤，它長成一棵馴化樹並最終戰勝且取代了其宿主。該模式由 Martin Fowler 引入，作為重寫單一系統時管理風險的方式。如需如何套用此模式的範例，請參閱使用容器和 Amazon API Gateway 逐步現代化舊版 Microsoft ASP.NET (ASMX) Web 服務。

子網

中的 IP 地址範圍VPC。子網必須位於單一可用區域。

監控控制和資料擷取 (SCADA)

在製造中，使用硬體和軟體來監控實體資產和生產操作的系統。

對稱加密

使用相同金鑰來加密及解密資料的加密演算法。

合成測試

以模擬使用者互動的方式測試系統，以偵測潛在問題或監控效能。您可以使用 Amazon CloudWatch Synthetics 來建立這些測試。

系統提示

提供內容、指示或指導方針給 LLM以指示其行為的技術。系統提示可協助設定內容，並建立與使用者互動的規則。

T

標籤

做為中繼資料的鍵值對，用於組織您的 AWS 資源。標籤可協助您管理、識別、組織、搜尋及篩選資源。如需詳細資訊，請參閱標記您的 AWS 資源。

目標變數

您嘗試在受監督的 ML 中預測的值。這也被稱為結果變數。例如，在製造設定中，目標變數可能是產品瑕疵。

任務清單

用於透過執行手冊追蹤進度的工具。任務清單包含執行手冊的概觀以及要完成的一般任務清單。對於每個一般任務，它包括所需的預估時間量、擁有者和進度。

測試環境

請參閱 環境。

訓練

為 ML 模型提供資料以供學習。訓練資料必須包含正確答案。學習演算法會在訓練資料中尋找將輸入資料屬性映射至目標的模式 (您想要預測的答案)。它會輸出擷取這些模式的 ML 模型。可以使用 ML 模型，來預測您不知道的目標新資料。

傳輸閘道

可用來互連 VPCs和內部部署網路的網路傳輸中樞。如需詳細資訊，請參閱 AWS Transit Gateway 文件中的什麼是傳輸閘道。

主幹型工作流程

這是一種方法，開發人員可在功能分支中本地建置和測試功能，然後將這些變更合併到主要分支中。然後，主要分支會依序建置到開發環境、生產前環境和生產環境中。

受信任的存取權

將許可授予您指定的服務，以代表您在組織中執行任務 AWS Organizations ，並在其帳戶中執行任務。受信任的服務會在需要該角色時，在每個帳戶中建立服務連結角色，以便為您執行管理工作。如需詳細資訊，請參閱 文件中的 AWS Organizations 搭配使用 AWS Organizations 與其他 AWS 服務。

調校

變更訓練程序的各個層面，以提高 ML 模型的準確性。例如，可以透過產生標籤集、新增標籤、然後在不同的設定下多次重複這些步驟來訓練 ML 模型，以優化模型。

雙比薩團隊

一個小型 DevOps 團隊，您可以使用兩個比薩來餵送。雙披薩團隊規模可確保軟體開發中的最佳協作。

U

不確定性

這是一個概念，指的是不精確、不完整或未知的資訊，其可能會破壞預測性 ML 模型的可靠性。有兩種類型的不確定性：認知不確定性是由有限的、不完整的資料引起的，而隨機不確定性是由資料中固有的噪聲和隨機性引起的。如需詳細資訊，請參閱量化深度學習系統的不確定性指南。

未區分的任務

也稱為繁重的作業，是建立和操作應用程式的必要工作，但不為最終使用者提供直接價值或提供競爭優勢。未區分任務的範例包括採購、維護和容量規劃。

較高的環境

請參閱 環境。

V

清空

一種資料庫維護操作，涉及增量更新後的清理工作，以回收儲存並提升效能。

版本控制

追蹤變更的程序和工具，例如儲存庫中原始程式碼的變更。

VPC 對等互連

兩個 之間的連線VPCs，可讓您使用私有 IP 地址路由流量。如需詳細資訊，請參閱 Amazon VPC 文件中的VPC互連內容。

漏洞

會危害系統安全性的軟體或硬體瑕疵。

W

暖快取

包含經常存取的目前相關資料的緩衝快取。資料庫執行個體可以從緩衝快取讀取，這比從主記憶體或磁碟讀取更快。

暖資料

不常存取的資料。查詢這類資料時，通常可接受中等緩慢的查詢。

視窗函數

在與目前記錄有某種關聯之資料列群組上執行計算的SQL函數。視窗函數適用於處理任務，例如根據目前資料列的相對位置計算移動平均值或存取資料列的值。

工作負載

提供商業價值的資源和程式碼集合，例如面向客戶的應用程式或後端流程。

工作串流

遷移專案中負責一組特定任務的功能群組。每個工作串流都是獨立的，但支援專案中的其他工作串流。例如，組合工作串流負責排定應用程式、波次規劃和收集遷移中繼資料的優先順序。組合工作串流將這些資產交付至遷移工作串流，然後再遷移伺服器和應用程式。

WORM

請參閱寫入一次，多次讀取。

WQF

請參閱AWS工作負載資格架構。

寫入一次，讀取許多 (WORM)

儲存模型，可一次性寫入資料，並防止資料遭到刪除或修改。授權使用者可以視需要多次讀取資料，但無法變更。此資料儲存基礎設施被視為不可變。

Z

零時差漏洞

利用零時差漏洞的攻擊，通常是惡意軟體。

零時差漏洞

生產系統中未緩解的缺陷或漏洞。威脅行為者可以使用這種類型的漏洞來攻擊系統。開發人員經常因為攻擊而意識到漏洞。

零鏡頭提示

提供執行任務LLM的指示，但沒有可協助引導任務的範例 (快照)。LLM 必須使用其預先訓練的知識來處理任務。零鏡頭提示的有效性取決於任務的複雜性和提示的品質。另請參閱微擊提示。

殭屍應用程式

平均CPU和記憶體用量低於 5% 的應用程式。在遷移專案中，通常會淘汰這些應用程式。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。