套用適用於 Amazon Neptune 的 AWS Well-Architected 架構

AWS 方案指引

AWS 方案指引: 套用適用於 Amazon Neptune 的 AWS Well-Architected 架構

Table of Contents

套用適用於 Amazon Neptune 的 AWS Well-Architected 架構

Amazon Web Services （貢獻者）

2026 年 1 月 (文件歷史記錄)

您可以使用 Amazon Amazon Neptune 在 Amazon Web Services (AWS) 上建置圖形型解決方案。本指南提供在規劃 Neptune 部署時套用 AWS Well-Architected Framework 原則的規範性指導。

AWS Well-Architected Framework 可協助您為各種應用程式和工作負載建置安全、高效能、彈性且高效率的基礎設施。它也提供一致的方法來評估架構並實作可擴展的設計。

AWS Well-Architected 架構是以下列六個支柱為基礎：

- 卓越營運
- 安全
- 可靠性
- 效能效率
- 成本最佳化
- 永續性

本指南提供 AWS Well-Architected Framework 設計支柱和最佳實務的資訊，以及在 上部署 Neptune 時應謹記的考量事項 AWS。

目標對象

本指南適用於設計並實作使用圖形之解決方案的資料工程師、解決方案架構師和資料分析師 AWS。

目標

本指南可協助您和組織執行下列動作：

- 根據您的使用案例和查詢模式，從支援的部署選項和查詢語言中進行選擇。
- 遵循 AWS Well-Architected 設計模式，以協助改善彈性和安全性。
- 設計您的查詢以獲得最佳效能並節省成本。
- 了解如何在生產環境中管理 Neptune 叢集時達到營運效率。

卓越營運支柱

AWS Well-Architected Framework 的卓越營運支柱著重於執行和監控系統，並持續改善流程和程序。其中包括有效支援開發和執行工作負載、深入了解其操作，以及持續改善支援程序和程序以交付商業價值的能力。您可以透過自我修復工作負載來降低操作複雜性，無需人工介入即可偵測和修復大多數問題。您可以遵循本節所述的最佳實務來實現此目標。當您的工作負載偏離預期行為時，請使用 Amazon Neptune 指標、API 和機制來正確回應。 APIs

此卓越營運支柱的討論著重於下列關鍵領域：

- 基礎設施即程式碼 (IaC)
- 變更管理
- 彈性策略
- 事件管理
- 合規稽核報告
- 日誌記錄和監控

使用 IaC 方法自動化部署

使用 IaC 在 Neptune 上自動化部署的最佳實務包括下列項目：

- 盡可能套用基礎設施即程式碼 (IaC) 來部署 Neptune 叢集。若要取得一致的環境組態，請使用 AWS CloudFormation 範本、 AWS Cloud Development Kit (AWS CDK)或 HashiCorp Terraform 為您的叢集建立所有必要的資源。
- 盡可能自動化 Neptune 操作程序，例如調整執行個體大小、新增或移除僅供讀取複本，或在全域資料表上執行手動容錯移轉。
- 從用戶端外部存放連線字串。使用擷取、轉換和載入 (ETL) 程序，以促進藍/綠部署策略、災難復原 (DR) 以及近乎零的停機時間遷移至新叢集。連線字串可以存放在 AWS Secrets Manager、Amazon DynamoDB 或任何可以動態變更它們的位置。
- 使用標籤將中繼資料新增至 Neptune 資源，並根據標籤追蹤用量。如需詳細資訊，請參閱標記 Amazon Neptune 資源。

進行頻繁、小型、可逆的變更

下列建議著重於小型、可逆的變更，以將複雜性降至最低，並降低工作負載中斷的可能性：

- 在來源控制服務中存放 IaC 範本和指令碼，例如 GitHub 或 GitLab。

> **Important**
>
> 請勿在來源控制中存放 AWS 登入資料。

- 要求 IaC 部署使用持續整合和持續交付 (CI/CD) 服務，例如 AWS CodePipeline或 AWS CodeBuild。這些服務會在非生產環境中編譯、測試和部署程式碼，其中包含暫時性 Neptune 叢集，然後再影響您的生產 Amazon Neptune 叢集。
- 在較低的環境中測試基礎設施和應用程式查詢，然後再將其部署到生產環境。這將最大限度地降低中斷的可能性，並有助於確保它們在工作負載和規模方面表現良好。

預期失敗

自我修復基礎設施透過預測故障並嘗試在不介入的情況下解決任何問題，來示範卓越營運。下列建議可協助您使用 Neptune 實現該成熟度：

- 建立監控計畫，使用 Amazon CloudWatch 指標來監控資料庫執行個體的 CPU 和記憶體用量，並了解用量模式。為您的應用程式日誌中找到的關鍵指標和 Neptune 用戶端回應建立 CloudWatch 儀表板和警示。如需高或低 CPU 使用率指標的詳細資訊，請參閱 Neptune 文件中的使用 CloudWatch 在 Neptune 中監控資料庫執行個體效能。

  如果您經常在查詢中遇到out-of-memory例外狀況，請考慮減少查詢周遊的節點總數，或嘗試使用 X2 系列執行個體，其 RAM-to-CPU比率較高。
- 設定通知以監控 Neptune 叢集的運作狀態。例如， `BufferCacheHitRatio` 應該持續處於高（大於 99.9%），而 `MainRequestQueuePendingRequests`應該持續處於低（理想情況下為 0，但取決於您的需求和延遲容錯能力）。
- 請考慮使用僅供讀取複本，以在 Neptune 中實現高可用性。您應該在與寫入器執行個體不同的可用區域中至少有兩個僅供讀取複本，以確保執行個體始終可用於在容錯移轉事件期間提供讀取查詢。
- 根據使用率指標自動擴展僅供讀取複本。如需詳細資訊，請參閱自動調整 Amazon Neptune 資料庫叢集中的複本數量。
- 測試資料庫執行個體的容錯移轉，以了解您的使用案例執行此程序需時多長。
- 如果您的應用程式需要完全中斷 AWS 區域 ，請考慮使用全域資料庫做為 DR 計畫的一部分。

從所有操作失敗中學習

自我修復基礎設施是一種長期工作，會在發生罕見問題或回應效果不如預期時，在反覆運算中發展。採用下列實務可推動專注於該目標：

- 透過從所有失敗中學習來推動改進。
- 跨團隊和組織分享學到的內容。如果組織中的多個團隊使用 Neptune，請建立通用聊天室或使用者群組來共用學習和最佳實務。

使用記錄功能來監控未經授權的或異常的活動

若要觀察異常效能和活動模式，請將日誌存放在 Amazon CloudWatch Logs 中。請考慮下列最佳實務：

- 啟用慢查詢記錄。定期檢閱日誌，並診斷特定查詢緩慢的原因。使用 Gremlin、SPARQL 或 openCypher 的 Neptune 解釋和設定檔端點，深入了解這些查詢緩慢的原因。
- 啟用 Neptune 稽核日誌，並定期檢閱日誌是否有未經授權的存取或異常。
- 如果您使用慢查詢記錄或稽核記錄，請啟用發佈至 CloudWatch Logs。這可協助您避免執行個體上的磁碟空間用盡。Neptune 執行個體的日誌儲存容量有限，且會在超過日誌空間時覆寫較舊的日誌檔案。CloudWatch Logs 支援長期保留日誌。CloudWatch Logs 中的增強型監控功能將改善您查詢日誌和診斷問題的能力。
- 為了為您的稽核日誌提供更好的分析工具，您可以設定 Neptune 資料庫叢集，將稽核日誌資料發佈至 CloudWatch Logs 中的日誌群組。使用 CloudWatch Logs，您可以執行日誌資料的即時分析、使用 CloudWatch 來建立警示和檢視指標，並使用 CloudWatch Logs 將您的日誌記錄儲存到高耐用性儲存體。如需詳細資訊，請參閱將 Neptune 日誌發佈至 Amazon CloudWatch Logs。
- Neptune 支援使用 記錄控制平面動作 AWS CloudTrail。如需詳細資訊，請參閱使用 記錄 Amazon Neptune API 呼叫 AWS CloudTrail。

安全支柱

的雲端安全 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，這些架構專為滿足最安全敏感組織的需求而建置。

安全性是 AWS 與您之間的共同責任。共同責任模型將此描述為雲端本身的安全和雲端內部的安全：

- 雲端的安全性 – AWS 負責保護在 AWS 服務 中執行的基礎設施 AWS 雲端。 AWS 也為您提供可安全使用的服務。在AWS 合規計畫中，第三方稽核人員會定期測試和驗證 AWS 安全性的有效性。若要了解適用於 Amazon Neptune 的合規計劃，請參閱合規計劃範圍內的AWS 服務。
- 雲端的安全性 – 您的責任取決於您使用 AWS 服務 的 。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。如需有關資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需歐洲資料保護的相關資訊，請參閱AWS 共同責任模型和 GDPR 部落格文章。

安全支柱可協助您了解如何在使用 Neptune 時套用共同責任模型。下列主題說明如何將 Neptune 設定為實現您的安全及合規目標。您也會了解如何使用其他 AWS 服務 來協助您監控和保護 Neptune 資源。

安全支柱包含下列主要重點領域：

- 資料安全
- 網路安全
- 身分驗證和授權

實作資料安全性

資料外洩和違規會讓您的客戶面臨風險，並可能對您的公司造成重大負面影響。下列最佳實務有助於保護您的客戶資料免於意外和惡意暴露：

- 叢集名稱、標籤、參數群組、 AWS Identity and Access Management (IAM) 角色和其他中繼資料不應包含機密或敏感資訊，因為該資料可能會出現在帳單或診斷日誌中。
- 存放為 Neptune 資料之外部伺服器的 URIs或連結不應包含登入資料資訊來驗證請求。
- Neptune 加密的執行個體可以協助保護您的資料免於發生未經授權的基礎儲存體存取，為資料提供另一層保護。您可以使用 Neptune 加密來提高部署在雲端之應用程式的資料保護。您也可以進行 Neptune 加密，以滿足靜態資料的合規要求。

若要啟用新 Neptune 資料庫執行個體的加密，請在 Neptune 主控台的啟用加密區段中選擇是（預設為選取），或在其中設定 AWS::Neptune::DBCluster::StorageEncrypted 屬性 CloudFormation。如果啟用加密，Neptune 預設將使用 Amazon Relational Database Service (Amazon RDS) AWS 受管金鑰，或者您可以建立客戶受管金鑰。如需建立 Neptune 資料庫執行個體的資訊，請參閱建立新的 Neptune 資料庫叢集。如需詳細資訊，請參閱加密 Neptune 資源靜態。您的自動化和手動快照會使用您為 Neptune 叢集選取的相同加密。

- 使用 SPARQL 和 openCypher 語言時，請練習適當的輸入驗證和參數化技術，以防止 SQL Injection 和其他形式的攻擊。避免使用使用者提供的輸入來建構使用字串串連的查詢。使用參數化查詢或預備陳述式，安全地將輸入參數傳遞至圖形資料庫。如需詳細資訊，請參閱 openCypher 參數化查詢和 SPARQL Injection Defence 的範例。 https://owasp.org/www-pdf-archive/Onofri-NapolitanoOWASPDayItaly2012.pdf
- 對於 Gremlin 語言，請使用 Gremlin 語言變體，而不是直接傳遞字串型 Gremlin 指令碼，以避免潛在的注入問題。

保護您的網路

Amazon Neptune 資料庫叢集只能在 的虛擬私有雲端 (VPC) 中建立 AWS。在 Neptune 1.4.6.0 之前，Neptune 資料庫叢集的端點只能在該 VPC 內存取。從 Neptune 1.4.6.0 及更新版本開始，Neptune 執行個體可設定為透過網際網路公開存取。最佳實務是僅在非生產環境中使用此功能，以為您的開發人員啟用 Neptune 的簡化存取（不過在 上一律需要 IAM 身分驗證才能啟用公有存取）。如果您已啟用公有可存取性，請考慮將資料庫連接埠的傳入安全群組規則設定為僅已知 IP 地址流量。在生產環境或使用包含敏感資料的叢集中，不允許公開存取並限制對 Neptune 資料庫叢集所在 VPC 的存取，以保護您的 Neptune 資料。如需詳細資訊，請參閱連線至 Amazon Neptune 圖形。

為了保護您的傳輸中資料，Neptune 會使用安全通訊協定和密碼，透過 HTTPS 強制 SSL 連線至任何執行個體或叢集端點。Neptune 為您的 Neptune 資料庫執行個體提供 SSL 憑證。Neptune SSL 憑證僅支援叢集端點、讀取器端點和執行個體端點主機名稱。

如果您使用負載平衡器或代理伺服器（例如 HAProxy），則必須使用 SSL 終止，並在代理伺服器上擁有自己的 SSL 憑證。SSL 傳遞沒有作用，因為提供的 SSL 憑證與代理伺服器主機名稱不符。如需使用 SSL 連線至 Neptune 端點的詳細資訊，請參閱使用 HTTP REST 端點連線至 Neptune 資料庫執行個體。

實作身分驗證和授權

若要控制誰可以在 Neptune 資料庫叢集和資料庫執行個體上執行 Neptune 管理 動作，請啟用 IAM 資料庫身分驗證並使用 IAM 登入資料。當您 AWS 使用 IAM 登入資料連線至 時，您的 IAM 角色必須具有授予執行 Neptune 管理操作所需許可的 IAM 政策。確保您遵循最低權限原則，僅授予完成任務所需的許可。如需詳細資訊，請參閱使用不同類型的 IAM 政策來控制使用暫時登入資料的 Neptune 和 IAM 身分驗證存取權。

若要控制誰可以連線至 Neptune 叢集並查詢資料，您可以使用 IAM 向 Neptune 資料庫執行個體或資料庫叢集進行身分驗證。如果您在 Neptune 資料庫叢集中啟用 IAM 身分驗證，則必須先驗證存取資料庫叢集的任何人。如需詳細資訊，請參閱在 Neptune 中啟用 IAM 資料庫身分驗證，以取得啟用 IAM 身分驗證的步驟。

當 IAM 資料庫身分驗證啟用時，每個請求都必須使用 AWS Signature 第 4 版進行簽署。若要了解如何將簽署的請求傳送至已啟用 IAM 身分驗證的所有 Neptune 端點，請參閱使用 AWS 簽章版本 4 連接和簽署。許多程式庫和工具，例如 awscurl，已經支援 AWS Signature 第 4 版。

為了與其他 互動 AWS 服務，Amazon Neptune 使用 IAM 服務連結角色。服務連結角色是直接連結至 Neptune 的一種獨特 IAM 角色類型。服務連結角色是由 Neptune 預先定義，並包含服務代表您呼叫其他 AWS 服務 所需的所有許可。如需詳細資訊，請參閱使用 Neptune 的服務連結角色。

可靠性支柱

可靠性支柱包含工作負載在預期情況下正確且一致地執行其預期功能的能力。包括在整個生命週期中執行及測試工作負載。

可靠的工作負載始於對軟體和基礎設施的前期設計決策。您的架構選擇會影響所有 AWS Well-Architected 支柱的工作負載行為。為求可靠性，您必須依循特定模式。

可靠性支柱著重於下列關鍵領域：

- 工作負載架構，包括服務配額和部署模式
- 變更管理
- 故障管理

了解 Neptune 服務配額

除了中國和 GovCloud AWS 區域 之外，所有支援的 Neptune 叢集磁碟區大小上限為 128 TB (TiB)，其中配額為 64 TiB。

128 TiB 配額足以在圖形中存放約 200-4000 億個物件。在已標記的屬性圖表 (LPG) 中，物件是節點或節點或邊緣上的節點、邊緣或屬性。在資源描述架構 (RDF) 圖形中，物件是四元組。

對於任何 Neptune Serverless 叢集，您可以同時設定 Neptune 容量單位 (NCUs的最小和最大數量。每個 NCU 包含 2 GB (GiB) 的記憶體和相關聯的 vCPU 和聯網。最小和最大 NCU 值適用於叢集中的任何無伺服器執行個體。您可以設定的最大 NCU 值最高為 128.0 NCU，而最小值最低為 1.0 NCU。透過觀察 Amazon CloudWatch 指標`ServerlessDatabaseCapacity`並擷取您經常在其中執行的範圍，並關聯該範圍內的不良行為或成本`NCUUtilization`，來最佳化最適合您應用程式的 NCU 範圍。在許多工作負載中，1.0 NCU 的起點太低，並在閒置一段時間後導致不可靠的行為。如果您發現工作負載擴展速度不夠快，請增加最小 NCUs，以便在擴展時為初始突增提供足夠的處理。

對於您可以建立的資料庫資源數量，每個 AWS 帳戶 都有每個區域的配額。這些資源包括資料庫執行個體和資料庫叢集。在您到達某項資源的上限時，所發出用來建立該資源的額外呼叫都會伴隨著異常而失敗。有些配額是可依請求增加的軟性配額。如需 Amazon Neptune 與 Amazon RDS、Amazon Aurora 和 Amazon DocumentDB （具有 MongoDB 相容性） 之間共用的配額清單，以及可用時請求增加配額的連結，請參閱 Amazon RDS 中的配額。

了解 Neptune 部署模式

在 Neptune 資料庫叢集中，有一個主要資料庫執行個體和最多 15 個 Neptune 複本。主要資料庫執行個體支援讀取和寫入操作，並對叢集磁碟區執行所有資料修改。Neptune 複本會連線至與主要資料庫執行個體相同的儲存磁碟區，而且僅支援讀取操作。Neptune 複本可以從主要資料庫執行個體卸載讀取工作負載。

若要實現高可用性，請使用僅供讀取複本。在不同的可用區域中擁有一或多個僅供讀取複本執行個體可提高可用性，因為僅供讀取複本可做為主要執行個體的容錯移轉目標。如果寫入器執行個體失敗，Neptune 會將僅供讀取複本執行個體提升為主要執行個體。發生這種情況時，提升的執行個體重新啟動時會短暫中斷（通常少於 30 秒），在此期間，對主要執行個體發出的讀取和寫入請求會失敗，但有例外。為了取得最高的可靠性，請考慮使用不同可用區域中的兩個僅供讀取複本。如果可用區域 1 中的主要執行個體離線，可用區域 2 中的執行個體會提升為主要執行個體，但在發生這種情況時無法處理查詢。因此，在轉換期間需要可用區域 3 中的執行個體來處理讀取查詢。

如果您使用的是 Neptune Serverless，所有可用區域中的讀取器和寫入器執行個體都會根據其資料庫負載來獨立擴展和縮減規模。您可以將讀取器執行個體的提升層設定為 0 或 1，以便隨著寫入器執行個體的容量向上和向下擴展。這可讓您隨時接管目前的工作負載。

如果您的應用程式具有全球足跡或需要多區域容錯移轉，請考慮使用 Neptune 全域資料庫。Amazon Neptune 全域資料庫跨越多個 AWS 區域，啟用低延遲全域讀取，並在中斷影響整個 的罕見情況下提供快速復原 AWS 區域。Neptune 全域資料庫包含一個區域中的主要資料庫叢集，以及不同區域中最多五個次要資料庫叢集。

管理和擴展 Neptune 叢集

您可以使用 Neptune 自動調整規模來自動調整資料庫叢集中的 Neptune 複本數量，以根據 CPU 使用率閾值滿足您的連線和工作負載需求。透過自動擴展，Neptune 資料庫叢集可以處理工作負載突然增加的情況。當工作負載減少時，自動擴展會移除不必要的複本，這樣您就不會支付未使用的容量。請注意，新的執行個體啟動可能需要長達 15 分鐘的時間，因此僅自動擴展並不足以因應快速的需求變化。

您只能將自動擴展與已有一個主要寫入器執行個體和至少一個僅供讀取複本執行個體的 Neptune 資料庫叢集搭配使用（請參閱 Amazon Neptune 資料庫叢集和執行個體）。此外，叢集中的所有僅供讀取複本執行個體都必須處於可用狀態。如果任何僅供讀取複本處於可用以外的狀態，Neptune 自動擴展不會執行任何動作，直到叢集中的每個僅供讀取複本都可用為止。

如果您遇到需求快速變化，請考慮使用無伺服器執行個體。無伺服器執行個體可以在短時間內垂直擴展，而自動擴展會在較長的期間水平擴展。此組態提供最佳可擴展性，因為無伺服器執行個體垂直擴

展，而自動擴展會執行個體化新的僅供讀取複本，以處理超過單一無伺服器執行個體最大容量的工作負載。如需 Amazon Neptune Serverless 容量擴展的詳細資訊，請參閱 Neptune Serverless 資料庫叢集中的容量擴展。

如果您的擴展需要在可預測的時間變更，您可以將變更排程到最小執行個體、最大執行個體和閾值，以更好地處理這些轉移需求。請記得至少提前 15 分鐘排程橫向擴展事件，以允許這些執行個體在需要時上線。

您可以使用參數群組中的參數，在 Amazon Neptune 中管理資料庫組態。參數群組扮演引擎組態值的容器，以套用至一或多個資料庫執行個體。修改參數群組中的叢集參數時，請了解靜態和動態參數之間的差異，以及套用它們的方式和時間。使用狀態端點來查看目前套用的組態。

管理備份和容錯移轉事件

Neptune 會自動備份叢集磁碟區，並在備份保留期間保留備份的資料。Neptune 備份具有連續性和增量性，因此，您可以快速還原到備份保留期內的任何時間點。您可以在建立或修改資料庫叢集時指定 1–35 天的備份保留期。

若要在備份保留期間之後保留備份，您也可以擷取叢集磁碟區中資料的快照。儲存快照需要支付 Neptune 的標準儲存費用。

當您建立資料庫叢集的 Amazon Neptune 快照時，Neptune 會建立叢集的儲存磁碟區快照，備份其所有資料，而不只是個別執行個體。您稍後可以從這個資料庫叢集快照進行還原來建立新的資料庫叢集。當您還原資料庫叢集時，請提供要還原的資料庫叢集快照名稱，然後提供還原所建立之新資料庫叢集的名稱。

測試您的系統如何回應容錯移轉事件。使用 Neptune API 強制容錯移轉事件。當您想要模擬資料庫執行個體的故障進行測試，或在容錯移轉發生後將操作還原至原始可用區域時，使用容錯移轉重新啟動會很有幫助。如需詳細資訊，請參閱設定和管理異地同步備份部署。當您重新啟動資料庫寫入器執行個體時，它會容錯移轉至待命複本。重新啟動 Neptune 複本不會啟動容錯移轉。

為您的用戶端設計可靠性。在容錯移轉事件期間測試其行為。在用戶端中以指數退避邏輯實作重試邏輯。您可以在 AWS Lambda Amazon Neptune 函數範例下的文件中找到實作此邏輯的程式碼範例。

AWS Backup 如果您有一組通用的備份需求，可套用至多個資料庫引擎，請考慮使用 。

效能效率支柱

AWS Well-Architected Framework 的效能效率支柱著重於如何在擷取或查詢資料時最佳化效能。效能最佳化是下列的增量和持續程序：

- 確認業務需求
- 測量工作負載效能
- 識別效能不佳的元件
- 調校元件以符合您的業務需求

效能效率支柱提供使用案例特定的指導方針，可協助識別正確的圖形資料模型和要使用的查詢語言。它還包括從 Amazon Neptune 擷取資料和取用資料的最佳實務。

效能效率支柱著重於下列關鍵領域：

- 圖形建模
- 查詢最佳化
- 叢集大小適中
- 寫入最佳化

了解圖形建模

了解已標記屬性圖 (LPG) 與資源描述架構 (RDF) 模型之間的差異。在大多數情況下，這是偏好事項。不過，有幾個使用案例，其中一個模型比另一個模型更適合。如果您需要了解連接圖形中兩個節點的路徑，請選擇 LPG。如果您想要跨 Neptune 叢集或其他圖形三元存放區聯合資料，請選擇 RDF。

如果您要建置軟體即服務 (SaaS) 應用程式或需要多租用戶的應用程式，請考慮在資料模型中整合租用戶的邏輯分離，而不是每個叢集有一個租用戶。若要實現該類型的設計，您可以使用名為 SPARQL 的圖形和標籤策略，例如在標籤前面加上客戶識別符，或新增代表租用戶識別符的屬性鍵/值對。請確定您的用戶端層注入這些值，以保持該邏輯分隔。如需多租用建議的詳細資訊，請參閱執行 Amazon Neptune 資料庫ISVs 的多租用指引。

查詢的效能取決於處理查詢時需要評估的圖形物件數量（節點、邊緣、屬性）。因此，圖形模型可能會對應用程式的效能產生重大影響。盡可能使用精細標籤，並僅存放實現路徑確定或篩選所需的屬性。若要達到更高的效能，請考慮預先計算圖形的部分，例如建立摘要節點或連接常見路徑的更多直接邊緣。

嘗試避免在具有異常大量邊緣且具有相同標籤的節點之間導覽。這類節點通常有數千個邊緣（其中大多數節點的邊緣計數為十）。結果是運算和資料複雜性更高。這些節點在某些查詢模式中可能不會有問題，但建議您以不同的方式建立資料模型，以避免這種情況，尤其是如果您將導覽到節點做為中繼步驟時。您可以使用慢查詢日誌來協助識別瀏覽這些節點的查詢。您可能會觀察到比平均查詢模式高出許多的延遲和資料存取指標，特別是如果您使用偵錯模式時。

如果您的使用案例支援節點和邊緣，請使用決定性節點 IDs，而不是使用 Neptune 為 IDs 指派隨機 GUID 值。依 ID 存取節點是最有效率的方法。

最佳化查詢

openCypher 和 Gremlin 語言可以交替用於 LPG 模型。如果效能是首要考量，請考慮交替使用兩種語言，因為對於特定查詢模式，一種可能比另一種更好。

Neptune 正在轉換為其替代查詢引擎 (DFE)。openCypher 僅在 DFE 上執行，但 Gremlin 和 SPARQL 查詢都可以選擇使用查詢註釋在 DFE 上執行。考慮在啟用 DFE 的情況下測試查詢，並在不使用 DFE 時比較查詢模式的效能。

Neptune 已針對從單一節點或一組節點開始的交易類型查詢進行最佳化，並從那裡散出，而不是評估整個圖形的分析查詢。對於分析查詢工作負載，請使用 Neptune Analytics。Neptune Analytics 是調查、探索性或資料科學工作負載的理想選擇，這些工作負載需要快速迭代以進行資料、分析和演算法處理。它也可以對圖形資料執行向量搜尋，並直接從 Neptune 資料庫執行個體載入資料。如果 Neptune Analytics 不符合您的需求，您也可以考慮AWS 適用於 Pandas 的 SDK，或使用 neptune-export 結合 AWS Glue或 Amazon EMR。

若要識別模型和查詢中的效率低下和瓶頸，請針對每種查詢語言使用 `profile`和 `explain` API，以取得查詢計劃和查詢指標的詳細說明。 APIs 如需詳細資訊，請參閱 Gremlin 描述檔、openCypher explain 和 SPARQL explain。

了解您的查詢模式。如果圖形中的不同邊緣數量變大，預設 Neptune 存取策略可能會變得效率低下。下列查詢可能會變得非常沒有效率：

- 未提供邊緣標籤時，跨邊緣向後導覽的查詢。
- 在內部使用此相同模式的子句，例如 `.both()` Gremlin，或捨棄任何語言節點的子句（需要捨棄傳入邊緣而不了解標籤）。
- 存取屬性值而不指定屬性標籤的查詢。這些查詢可能會變得非常沒有效率。如果這符合您的使用模式，請考慮啟用 OSGP 索引（物件、主旨、圖形、述詞）。

使用慢查詢記錄來識別慢查詢。緩慢的查詢可能是由未最佳化的查詢計劃或不必要的大量索引查詢所造成，這可能會增加 I/O 成本。Gremlin、SPARQL 或 openCypher 的 Neptune 解釋和設定檔端點可協助您了解這些查詢緩慢的原因。原因可能包括下列項目：

- 與圖形中的平均節點相比，邊緣數量異常高的節點（例如，千個節點與數十個節點相比）可能會增加運算複雜性，因此延遲更長，資源耗用量也更高。判斷這些節點是否已正確建模，或是否可以改善存取模式，以減少必須周遊的邊緣數量。
- 未最佳化的查詢將包含未最佳化特定步驟的警告。重寫這些查詢以使用最佳化步驟可能會改善效能。
- 備援篩選條件可能會導致不必要的索引查詢。同樣地，備援模式可能會導致重複的索引查詢，可透過改善查詢進行最佳化（請參閱設定檔輸出`Index Operations - Duplication ratio`中的）。
- Gremlin 等某些語言沒有強式輸入的數值，而是改用類型提升。例如，如果值為 55，Neptune 會尋找整數、長數、浮點數和其他相當於 55 的數值類型。這會導致額外的操作。如果您事先知道您的類型相符，您可以使用查詢提示來避免這種情況。
- 您的圖形模型可能會大幅影響效能。考慮使用更精細的標籤或將捷徑預先計算為多躍點線性路徑，以減少需要評估的物件數量。

如果僅查詢最佳化不允許您達到效能需求，請考慮搭配 Neptune 使用各種快取技術來達成這些需求。

Neptune 效能會隨著每個版本持續改善。檢閱版本備註，以查看每個版本改進的詳細資訊。請考慮規劃 Neptune 資料庫叢集的定期更新，以協助達到最佳效能。較新版本也支援較新的執行個體。請考慮升級至 1.4.5.0 或更新版本，以利用`r8g`執行個體。如需如何改善工作負載效能的詳細資訊，請參閱使用 Amazon Neptune 1.4.5 版搭配 AWS Graviton4 R8g 執行個體的 4.7 倍更好的寫入查詢價格效能。

適當大小的叢集

根據您的並行和輸送量需求調整叢集的大小。叢集中每個執行個體可處理的並行查詢數量等於該執行個體上虛擬 CPUs(vCPUs) 數量的兩倍。在佔用所有工作者執行緒時抵達的其他查詢會放入伺服器端佇列。當工作者執行緒可用時，這些查詢會以first-in-first-out(FIFO) 為基礎來處理。`MainRequestQueuePendingRequests` Amazon CloudWatch 指標會顯示每個執行個體目前的佇列深度。如果此值經常高於零，請考慮選擇具有更多 vCPU 的執行個體。 vCPUs 如果佇列深度超過 8，192，Neptune 將傳回`ThrottlingException`錯誤。

每個執行個體大約有 65% 的 RAM 保留給緩衝區快取。緩衝區快取會保留有效的資料集（而非整個圖形；僅是查詢的資料）。若要判斷從緩衝區快取而非儲存體擷取的資料百分比，請監控 CloudWatch 指標 `BufferCacheHitRatio`。如果此指標通常低於 99.9%，請考慮嘗試具有更多記憶體的執行個體，以判斷其是否降低您的延遲和 I/O 成本。

僅供讀取複本的大小不必與寫入器執行個體相同。不過，繁重的寫入工作負載可能會導致較小的複本落後並重新啟動，因為它們無法跟上複寫的進度。因此，我們建議讓複本等於或大於寫入器執行個體。

為僅供讀取複本使用自動擴展時，請記住，最多可能需要 15 分鐘才能上線新的僅供讀取複本。當用戶端流量快速增加但可預測時，請考慮使用排程擴展來設定較高的僅供讀取複本數目下限，以考慮該初始化時間。

無伺服器執行個體支援數個不同的使用案例和工作負載。針對下列案例，請考慮在佈建執行個體上無伺服器：

- 您的工作負載經常在一天中波動。
- 您已建立新的應用程式，且不確定工作負載大小為何。
- 您正在執行開發和測試。

請務必注意，無伺服器執行個體的成本高於每 GB RAM 的同等佈建執行個體。每個無伺服器執行個體都包含 2 GB 的 RAM，以及相關聯的 vCPU 和聯網。在您的選項之間執行成本分析，以避免意外帳單。一般而言，只有當您的工作負載每天只有幾個小時非常繁重，且一天的其餘時間幾乎為零，或您的工作負載在一天中大幅波動時，您才能透過無伺服器節省成本。

利用 Amazon Neptune 定價計算器，根據queries-per-second(QPS) 需求等因素，協助評估叢集的正確組態。

最佳化寫入

若要最佳化寫入，請考慮下列事項：

- Neptune 大量載入器是最初載入資料庫或附加至現有資料的最佳方式。Neptune 載入器不是交易式且無法刪除資料，因此如果這些是您的需求，請勿使用它。
- 您可以使用支援的查詢語言進行交易更新。若要最佳化寫入 I/O 操作，請批次寫入每個遞交 50-100 個物件的資料。物件是 LPG 中節點或邊緣上的節點、邊緣或屬性，或 RDF 中的三重存放區或四元組。
- 所有交易 Neptune 寫入操作都是每個連線的單一執行緒。傳送大量資料至 Neptune 時，請考慮具有多個平行連線，每個連線都會寫入資料。當您選擇 Neptune 佈建執行個體時，執行個體大小會與多個 vCPUs相關聯。Neptune 會為執行個體上的每個 vCPU 建立兩個資料庫執行緒，因此在測試最佳平行處理時，從 vCPUs 數目的兩倍開始。 無伺服器執行個體會以大約每 4 個 NCUs 一個的速率擴展 vCPUs 數量。

> **Note**
>
> 這不適用於大量載入 API，僅適用於直接連線。

- 在所有寫入程序期間規劃並有效率地處理 ConcurrentModificationExceptions，即使一次只有單一連線正在寫入資料。在`ConcurrentModificationExceptions`發生時為用戶端設計可靠性。
- 如果您想要刪除所有資料，請考慮使用快速重設 API，而不是發出並行刪除查詢。與前者相比，後者需要更長的時間並產生大量的 I/O 成本。
- 如果您想要刪除大部分的資料，請考慮使用 neptune-export 將您要保留的資料匯出，以將資料載入新的叢集。然後刪除原始叢集。

成本最佳化支柱

AWS Well-Architected Framework 的成本最佳化支柱著重於避免不必要的成本。下列建議可協助您符合 Amazon Neptune 的成本最佳化設計原則和架構最佳實務。

成本最佳化支柱著重於下列關鍵領域：

- 了解一段時間內的支出並控制資金分配
- 選取正確類型和數量的資源
- 擴展以滿足業務需求，而不會過度花費

了解所需的用量模式和服務

如果您的資料模型具有可識別的圖形結構，且您的查詢需要探索關係並周遊多個躍點，Neptune 非常適合您的工作負載。圖形資料庫不適合下列模式：

- 主要是單躍查詢（考慮您的資料是否可更好地表示為物件的屬性）
- 存放為屬性的 JSON 或 BLOB 資料
- 跨資料集彙總的查詢，例如計算大量節點的數值屬性總和

考慮將多個專用資料庫結合用於特定存取模式，是否可以滿足您的所有需求。例如：

- 使用 Neptune、DynamoDB 或 Amazon DocumentDB 中的一或多個項目，最適合顯示需要較少複雜圖形導覽以及高度並行擷取單一節點屬性的 API。
- 關聯式資料庫可以與 Neptune 共存，以維護現有的功能，但僅在關聯式資料庫中執行和擴展不良的多躍點周遊使用 Neptune。

了解與 Neptune 互動和補充的 服務相關的成本，包括下列項目：

- 大量載入 Neptune 的資料檔案的 Amazon Simple Storage Service (Amazon S3) 儲存成本
- 用於插入或 upsert 查詢、讀取查詢和 Neptune 串流處理的 Lambda 函數
- 建置在 Neptune 上的 API 層，可與 Amazon API Gateway 中的用戶端應用程式（而非直接連線至資料庫）互動，或 AWS AppSync
- AWS Glue 用來往返 Neptune 傳輸資料的 任務

- Amazon Kinesis 或 Amazon Managed Streaming for Apache Kafka (Amazon MSK) 執行個體接收串流資料，以近乎即時的方式擷取至 Neptune。
- AWS Database Migration Service 用於將關聯式資料遷移至 Neptune
- Jupyter 筆記本和深度圖形程式庫機器學習模型的 Amazon SageMaker 執行期成本

選取關注成本的資源

Neptune 定價是根據每小時執行個體成本（或無伺服器使用的 Neptune 運算單位）、資料 I/O 和儲存體用量。執行個體平均佔整體成本的 85%，因此適當調整大小可能會對成本產生重大影響。適當大小執行個體的最佳方法是在各種執行個體上測試應用程式效能，並比較下列因素：

- `MainRequestQueuePendingRequests` CloudWatch 指標是否保持在接近零的一致低數？
- `BufferCacheHitRatio` CloudWatch 指標大部分時間是否保持在 99.9% 以上？
- 執行個體成本和相關聯資料 I/O 成本的成本和效能曲線是多少？ 需要經常將緩衝區快取交換與儲存體的執行個體大小過小，可能會大幅增加資料讀取成本。在這些情況下，`BufferCacheHitRatio`會經常下降。

執行個體成本會隨著相同執行個體系列中的大小線性擴展。`db.r6i.2xlarge` 執行個體的每小時成本是`db.r6i.xlarge`執行個體的兩倍，也是資源配置的兩倍。`db.r6i.24xlarge` 執行個體是`db.r6i.xlarge`執行個體每小時成本的 24 倍。

估計您必須支援的並行查詢數量。您可以有 0 到 15 個僅供讀取複本來處理唯讀查詢。如果您的需求因一天、一週或一個月的時間而有所不同，您可以使用多個較小的執行個體來根據排程進行擴展。執行個體上的每個 vCPU 提供兩個執行緒來處理並行查詢。三個僅供`db.r6i.xlarge`讀取複本各有 4 個 vCPU，可以處理 24 個並行查詢。

如果您的流量改為以每秒查詢數 (QPS) 來測量，您必須實驗以確定查詢的平均延遲。Neptune 叢集可支援的每秒查詢數等於 `vCPU × 2 × (1 second/average query latency)`。例如，如果您有 4 個 vCPU 且查詢延遲為 100 毫秒 (0.1 秒），則 `QPS = 4 × 2 × (1s/0.1s) = 80 queries per second`。

對於連續、穩定且可預測的工作負載，佈建的執行個體比無伺服器便宜。當您的工作負載每天只需要數小時（例如 `db.r6i.4xlarge`) 使用量非常高，然後當天剩餘時間幾乎沒有流量（例如 1 個 Neptune 運算單位），則無伺服器提供最佳化成本的機會。無伺服器執行個體會擴展數小時，然後再縮減，比起整天使用佈建的`db.r6i.4xlarge`執行個體來得便宜。

考慮升級至 Neptune 1.4.5.0 或更新版本，並利用r8g執行個體以低於 r7g或 等較舊世代執行個體的成本實現更佳的讀取和寫入輸送量r6g。如需詳細資訊，請參閱使用 Amazon Neptune v1.4.5 的 AWS Graviton4 R8g 執行個體寫入查詢價格效能提升 4.7 倍 (AWS 部落格文章）。

Neptune 叢集預設為使用標準儲存體建立 （ 如果您使用主控台建立 ，則預設為選取 I/O 最佳化儲存體 ）。使用 I/O 最佳化儲存體時，您需要為儲存體和執行個體支付稍高的成本，但沒有 I/O 成本。這會導致更可預測的經常性成本，但如果您的 I/O 用量通常很低，利用標準儲存可能更具成本效益。如果您想要最初載入大量資料，您可以選擇 I/O 最佳化儲存、執行初始資料載入，然後切換到標準儲存，以最佳化成本。儲存類型只會影響帳單模型，且在 Neptune 資料庫叢集或執行個體組態中沒有技術差異。您可以每 30 天變更一次儲存類型。30 天後，請檢查您的詳細 Neptune 成本，並使用 Neptune 定價頁面來計算使用 I/O 最佳化儲存的成本是否會更高。如果可以的話，請繼續使用標準儲存，否則請切換回 I/O 最佳化。

為您的工作負載選擇最佳的 Neptune 執行個體組態

如果您在 2025 年 7 月 15 AWS 帳戶 日之前建立 ，則可以使用 AWS 免費方案進行 Neptune 的入門級實驗。750 小時的免費 db.t3.medium和 db.t4g.medium執行個體用量足以讓您以低規模充分了解 Neptune。您的叢集將在免費試用期結束後保留，但從該時間點開始，您將需要支付使用費。

db.t3.medium 和 db.t4g.medium執行個體適用於您未使用 openCypher、Graph Explorer 或各種生成式 AI 整合的低成本開發環境。這些執行個體的 RAM-to-vCPU比率 (2：1) 小於R系列執行個體 (8：1) 或X系列執行個體 (16：1)。這可降低比率，防止使用啟用 openCypher 效能、GenAI 整合 （ 通知 LLM 圖形結構描述 ） 和 Graph Explorer 的 DFE 引擎統計資料。使用T系列執行個體時，效能描述檔可能會明顯不同，尤其是先前提到的工作負載。這些執行個體也可以增加查詢瀏覽圖形大部分OutOfMemoryExceptions時的發生次數。若要判斷後者條件是否可能受到影響，請檢查 BufferCacheHitRatio CloudWatch 指標。

我們強烈建議您不要對T系列執行個體進行任何效能或負載測試，因為您可能會遇到不一致的結果，這些結果並不表示生產環境。

當您的工作負載相當穩定且可預測時，佈建的執行個體可為您提供最佳的成本和效能組合。根據所需的請求並行和查詢複雜性，選擇執行個體大小。較高的並行需要更多 vCPUs。較高的查詢複雜性需要更多 RAM。使用 MainRequestQueuePendingRequests CloudWatch 指標來判斷前者的影響 （ 大於零表示的並行請求多於可處理的請求 ）。使用 BufferCacheHitRatio CloudWatch 指標來判斷後者的影響。經常低於 99.9% 的比率表示沒有足夠的 RAM 來包含正在評估的圖形工作部分，這會導致更頻繁的快取交換。如果 R 系列執行個體提供足夠的並行但沒有足夠的 RAM，請考慮嘗試X一系列執行個體。

Neptune 文件說明無伺服器執行個體的理想使用案例。如果您不確定佈建或無伺服器是否最適合您，而成本是您的主要考量，請在無伺服器中測試工作負載，以判斷使用的 NCUs 數量，並將佈建 (N hours × hourly provisioned cost) 的成本與無伺服器 () 進行比較sum of NCUs × hourly cost per NCU。如果您不確定同等大小的佈建執行個體，一個 NCU 相當於大約 2 GB 的 RAM 和相關聯的 vCPU 和聯網。如果您的佈建執行個體來自 r6i 系列，則比率為每 8 GB RAM 1 個 vCPU，或 4 NCUs，以及相關聯的聯網。Amazon Neptune 定價計算器也提供比較，協助您決定最佳成本組態。

將無伺服器用於主要和複本執行個體時，請記住，提升層 0 和 1 中的僅供讀取複本會根據寫入器執行個體擴展其 NCUs，以便在容錯移轉事件發生時適當擴展。根據寫入器或讀取器接收最多流量的執行個體，設定這些執行個體的 NCU 限制。

在每週 7 天、每天 24 小時不需要叢集的環境中，請考慮撰寫指令碼，以在不使用時關閉 Neptune 執行個體，並在使用前再次啟動它們。Neptune 執行個體會每 7 天自動重新啟動一次，以確保套用必要的維護更新。如果您想要讓執行個體長時間關閉，請使用每週指令碼再次將其關閉。

適當大小的資料儲存和傳輸

更有效率的查詢 （例如，需要接觸圖形中較少節點、邊緣和屬性的查詢） 需要較少的 I/O 傳輸，而且可能會利用較小的執行個體，因為需要較少的緩衝區快取。使用設定檔或解釋查詢語言的端點來最佳化查詢，並考慮針對查詢效能最佳化您的圖形模型。

Neptune 在大型字串上使用字典編碼，該字典針對效能進行了最佳化，而非效率。如果您有大型 BLOBs、JSON 或經常變更屬性的字串，請考慮將它們存放在 Amazon S3、Amazon DynamoDB 或 Amazon DocumentDB 的 Neptune 外部，並在 Neptune 節點中僅存放參考。

在某些情況下，選擇較大的執行個體大小可能會比較便宜。如果您的 I/O 成本因為 較低而很高BufferCacheHitRatio，則較大的緩衝區快取可能會大幅降低該成本。這是因為所有資料都適合快取，而不是經常從儲存體交換並產生 I/O 傳輸速率。

Neptune 使用copy-on-write。複製將圖形分割成多個碎片時，最好不要刪除複製叢集上不需要的資料，因為這將涉及建立新資料頁面，進而增加儲存成本。在複製事件之前未變更的資料將存在於跨兩個叢集共用的單一資料頁面中，並且只會針對該單一複本收費。

請勿啟用 OSGP 索引或使用 R5d 執行個體，除非您已測試確認它們對您的工作負載造成重大影響。兩者都專為極少發生的案例而設計，而且它們可能會以最少或無收益的方式增加您的成本。

永續性支柱

永續性支柱著重於將執行雲端工作負載的環境影響降至最低。關鍵主題包括永續性的共同責任模型、了解影響，以及最大化使用量，以盡可能減少所需資源並減少下游影響。

永續性支柱包含下列主要重點領域：

- 您的影響
- 永續性目標
- 最大化用量
- 預測和採用新的、更有效率的硬體和軟體產品
- 使用 受管服務
- 減少下游影響

本指南著重於您的影響。如需其他永續性設計原則的詳細資訊，請參閱 AWS Well-Architected Framework。

您的選擇和要求會影響環境。如果您選擇 AWS 區域 碳強度較低的 ，而且如果您的需求反映實際的工作負載需求，而不只是最大化運作時間和耐用性，工作負載的永續性就會增加。下一節討論最佳實務和周到考量，如果在工作負載設計和持續操作中採用，將對環境產生正面影響。

AWS 區域 選擇

有些 AWS 區域 接近 Amazon 可再生能源專案，或位於電網已發佈的碳強度低於其他項目的位置。考慮可能適用於工作負載的區域對永續性的影響，並將您的清單與 可使用 Neptune 的區域交叉參考。

根據使用者行為模式的使用量

適當調整用量大小以符合使用者的流量和行為，有助於將服務對環境的影響 AWS 降至最低。設計解決方案時，請考慮下列最佳實務：

- 監控 `CPUUtilization`、 `MainRequestQueuePendingRequests`和 等 Amazon CloudWatch 指標`TotalRequestsPerSec`，以判斷您的需求何時最高和最低，並確保您的叢集資源在這段期間大小正確。
- 在未使用非生產環境的時段內，自動停止非生產環境。如需詳細資訊，請參閱部落格文章使用資源標籤自動停止和啟動 Amazon Neptune 環境資源。

- 如果您的流量模式經常變化且無法預測，請考慮使用隨需擴展和縮減的 Neptune Serverless 執行個體，而不是使用為尖峰流量佈建的執行個體。
- 除了業務連續性目標之外，請考慮將您的服務層級協議與永續性目標保持一致。消除多區域災難復原、高可用性或長期備份保留等需求，尤其是非生產環境或非任務關鍵工作負載，可以減少實現這些目標所需的資源量。

最佳化軟體開發和架構模式

若要避免浪費，請最佳化模型和查詢，並共用運算資源，以便使用 Neptune 執行個體和叢集中可用的所有資源。特定最佳實務包括：

- 讓開發人員共用 Neptune 執行個體和 Jupyter Notebook 應用程式執行個體，而不是每個建立自己的執行個體。透過使用多租用分割策略，在單一 Neptune 叢集中為每個開發人員提供自己的邏輯分割區，並在單一 Jupyter 執行個體上為每個開發人員建立個別的筆記本資料夾。
- 實作可將資源使用量最大化並將閒置時間降至最低的模式，例如將資料和批次記錄載入大型交易的平行執行緒。
- 最佳化您的查詢和圖形模型，將計算結果所需的資源降至最低。
- 對於 Gremlin 查詢結果，請使用結果快取功能，將重新計算分頁或經常重複查詢所花費的資源降至最低。
- 將您的 Neptune 環境保持在最新狀態。最新版本的 Neptune 支援更有效率的最新 Amazon EC2 執行個體，例如 Graviton。它們也具有查詢最佳化改進和錯誤修正，可減少計算查詢所需的資源量。

Resources

參考

- AWS Well-Architected
- AWS Well-Architected Framework 文件
- Neptune 最新更新
- 最佳實務：充分利用 Neptune
- Amazon Neptune 定價計算器

部落格文章

- 使用 Apache TinkerPop Gremlin 自動化測試 Amazon Neptune 資料存取
- 使用資源標籤自動停止和啟動 Amazon Neptune 環境資源
- Amazon Neptune 資料平面動作的精細存取控制
- 使用 Amazon Neptune 1.4.5 版搭配 AWS Graviton4 R8g 執行個體的寫入查詢價格效能提升 4.7 倍
- Orca Security 如何最佳化其 Amazon Neptune 資料庫效能
- 使用 Amazon Neptune 公有端點更快速地建置圖形應用程式
- 新的 Amazon Neptune 引擎版本為 openCypher 查詢效能提供高達 9 倍的速度和 10 倍的輸送量

免費 AWS 技能建置器課程

- Amazon Neptune 入門
- 在 Amazon Neptune 上建置應用程式
- Amazon Neptune 的資料建模

貢獻者

本指南的貢獻者包括：

- Brian O'Keefe，首席 Neptune 解決方案架構師， AWS
- Abhishek Mishra，資深 Neptune 解決方案架構師， AWS
- Ganesh Sawhney | 策略合作夥伴成功解決方案架構師 AWS
- Michael Havey，資深 Neptune 解決方案架構師 AWS
- Kevin Phillips，Neptune 解決方案架構師， AWS
- Melissa Kwok，Neptune 解決方案架構師， AWS
- Sakti Mishra，首席解決方案架構師 AWS
- Javed Ali，資深解決方案架構師 AWS

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
Neptune 版本更新	我們更新文件以包含 Amazon Neptune 1.4.6.0 及更新版本的相關資訊。	2026 年 1 月 2 日
初次出版	—	2023 年 9 月 27 日

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。