實作指南

AWS 上的分散式負載測試

AWS 上的分散式負載測試: 實作指南

Table of Contents

自動化大規模測試您的軟體應用程式

發佈日期：2025 年 11 月

AWS 上的分散式負載測試可協助您大規模自動化軟體應用程式的效能測試，以在發佈應用程式之前識別瓶頸。此解決方案模擬數千名連線使用者以持續速率產生 HTTP 請求，而不需要佈建伺服器。

此解決方案利用 AWS Fargate 上的 Amazon Elastic Container Service (Amazon ECS) 部署執行負載測試模擬的容器，並提供下列功能：

- 在獨立執行的 AWS Fargate 容器上部署 Amazon ECS，以測試應用程式的負載容量。
- 跨多個 AWS 區域模擬數萬並行使用者，以連續的速度產生請求。
- 使用 JMeter、K6、Locust 測試指令碼或簡單的 HTTP 端點組態自訂您的應用程式測試。
- 排程載入測試以立即執行、在未來的日期和時間執行，或定期執行。
- 在不同案例和區域同時執行多個負載測試。

此實作指南提供 AWS 解決方案的分散式負載測試概觀、其參考架構和元件、規劃部署的考量，以及將解決方案部署至 Amazon Web Services (AWS) 雲端的組態步驟。它包含 AWS CloudFormation 範本的連結，該範本會啟動和設定使用 AWS 安全與可用性最佳實務部署此解決方案所需的 AWS 服務。

在其環境中使用此解決方案功能的目標受眾包括 IT 基礎設施架構師、管理員和 DevOps 專業人員，這些專業人員在 AWS 雲端中具有實際的架構經驗。

使用此導覽表快速找到這些問題的答案：

如果您想要 . . .	讀取 . . .
了解執行此解決方案的成本。 AWS 資源在美國東部 （維吉尼亞北部） 區域執行此解決方案的估計成本為每月 30.90 USD。	成本
了解此解決方案的安全考量。	安全性
了解如何規劃此解決方案的配額。	配額
了解哪些 AWS 區域支援此解決方案。	支援的 AWS 區域

如果您想要 . . .	讀取 . . .
了解選用的 MCP Server for AI 輔助負載測試分析。	MCP 伺服器整合
檢視或下載此解決方案中包含的 AWS CloudFormation 範本，以自動部署此解決方案的基礎設施資源 (「堆疊」)。	AWS CloudFormation 範本
存取原始程式碼，並選擇性地使用 AWS 雲端開發套件 (AWS CDK) 來部署解決方案。	GitHub 儲存庫

功能

解決方案提供下列功能：

多重測試架構支援

支援 JMeter、K6 和 Locust 測試指令碼，以及簡單的 HTTP 端點測試，而不需要自訂指令碼。如需詳細資訊，請參閱架構詳細資訊區段中的測試類型。

高使用者負載模擬

模擬成千上萬的並行虛擬使用者，以在逼真的載入條件下對您的應用程式進行壓力測試。

多區域負載分佈

將負載測試分散到多個 AWS 區域，以模擬地理分佈的使用者流量並評估全域效能。

彈性測試排程

排程測試以立即執行、在特定的未來日期和時間，或使用 cron 表達式以週期性排程進行自動迴歸測試。

即時監控

提供選用的即時資料串流，以即時指標監控測試進度，包括回應時間、虛擬使用者計數和請求成功率。

完整測試結果

顯示詳細的測試結果，其中包含效能指標、百分位數 (p50、p90、p95、p99)、錯誤分析，以及用於離線分析的可下載成品。

基準比較

指定基準測試執行以進行效能比較，以追蹤一段時間內的改善或迴歸。

端點彈性

測試跨 AWS 區域、內部部署環境或其他雲端供應商的任何 HTTP 或 HTTPS 端點。

直覺式 Web 主控台

提供以 Web 為基礎的主控台來建立、管理和監控測試，而不需要命令列互動。

AI 輔助分析 （選用）

透過模型內容協定 (MCP) 伺服器與 AI 開發工具整合，以智慧分析負載測試資料。

多重通訊協定支援

透過自訂測試指令碼支援各種通訊協定，包括 HTTP、HTTPS、WebSocket、JDBC、JMS、FTP 和 gRPC。

優勢

解決方案提供下列優點：

全方位效能測試

支援負載測試、壓力測試和耐久性測試，以徹底評估各種條件下的應用程式效能。

早期問題偵測

在生產部署之前識別效能瓶頸、記憶體流失和可擴展性問題，降低中斷的風險。

真實世界用量模擬

準確模擬真實世界的使用者行為和流量模式，以在逼真的條件下驗證應用程式效能。

可行的績效詳情

提供詳細的指標、百分位數和錯誤分析，以了解應用程式行為並引導最佳化工作。

自動化測試工作流程

啟用排程和週期性測試，以進行持續效能監控和迴歸測試，無需手動介入。

符合成本效益的基礎設施

使用無伺服器 AWS Fargate 容器搭配pay-per-use定價，無需專用測試基礎設施和持續訂閱費用。

快速測試部署

在幾分鐘內部署和擴展測試基礎設施，無需佈建或管理伺服器。

輕鬆查詢測試結果

透過選用的模型內容協定 (MCP) 伺服器與 AI 開發工具整合，啟用自然語言查詢和負載測試資料的智慧型分析，以更快地洞察和故障診斷。

使用案例

生產前驗證

在類似生產的負載條件下測試 Web 和行動應用程式，再啟動新版本來驗證效能並識別問題。

容量規劃

判斷應用程式可與目前基礎設施支援的並行使用者數量上限，並識別何時需要擴展。

峰值負載驗證

確認您的基礎設施可以處理尖峰負載、季節性流量激增或意外的需求激增，而不會降低效能。

效能最佳化

識別效能瓶頸，例如緩慢的資料庫查詢、無效的程式碼執行、網路延遲或資源限制。

迴歸測試

排程重複負載測試，以偵測新程式碼部署或基礎設施變更所產生的效能迴歸。

全球效能評估

評估多個地理區域的應用程式效能，以確保全球受眾的使用者體驗一致。

API 負載測試

測試 REST APIs、GraphQL 端點或微服務，以驗證負載下的回應時間、輸送量和錯誤率。

CI/CD 管道整合

將自動化效能測試整合到持續整合和部署管道中，以在開發週期的早期發現效能問題。

第三方服務測試

測試應用程式在各種負載條件下所依賴之第三方 APIs 或服務的效能和可靠性。

概念和定義

本節說明關鍵概念並定義此解決方案特有的術語：

案例

測試定義，包括測試的名稱、描述、任務計數、並行、AWS 區域、漸進測試、保留、測試類型、排程日期和週期組態。

任務計數

將在 Fargate 叢集中啟動以執行測試案例的容器數量。一旦達到 Fargate 資源的帳戶限制，就不會建立其他任務。不過，已執行的任務會繼續。

concurrency

並行（每個任務的並行虛擬使用者數量）。根據預設設定的建議並行為 200。並行受限於 CPU 和記憶體。對於以 Apache JMeter 為基礎的測試，較高的並行會增加 ECS 任務上 JVM 使用的記憶體。預設 ECS 任務定義會建立具有 4 GB 記憶體的任務。建議從 1 個任務的較低並行值開始，並監控任務叢集的 ECS CloudWatch 指標。請參閱 Amazon ECS 叢集使用率指標。

漸進測試

從零逐漸增加到目標並行層級的期間。

保留

在漸進測試完成後維持目標並行層級的期間。

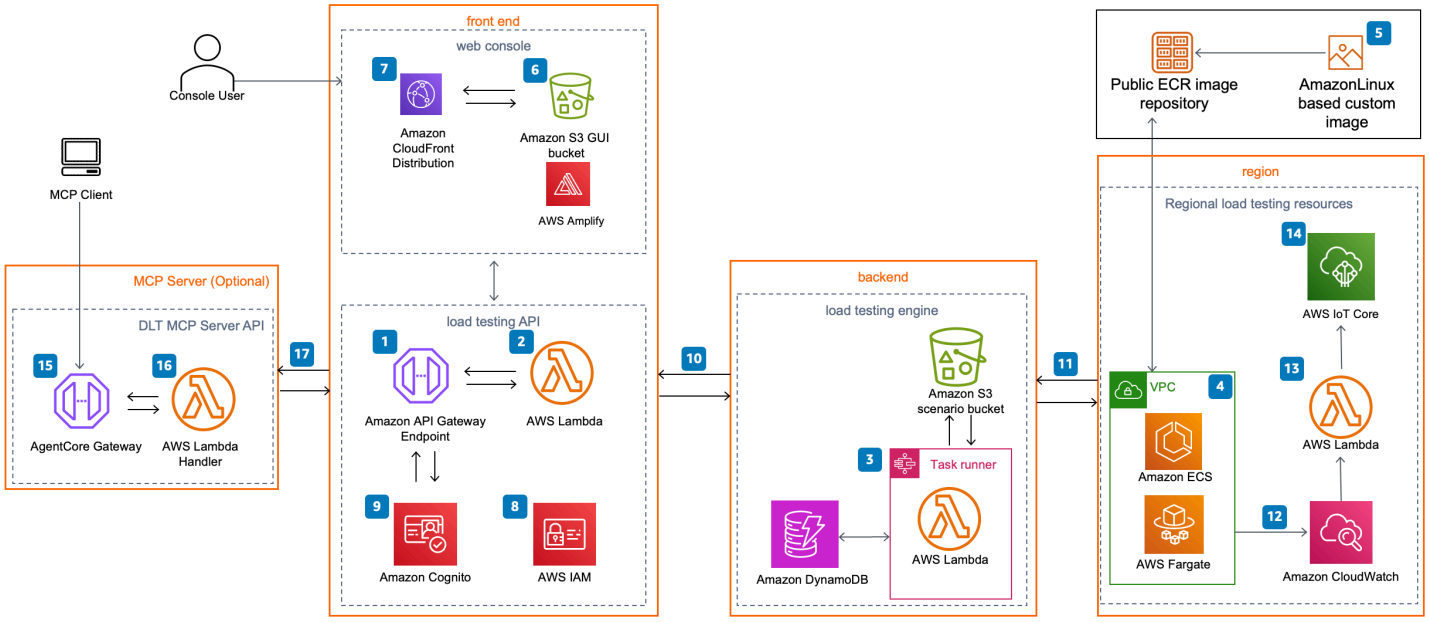
如需 AWS 術語的一般參考，請參閱 AWS 詞彙表。

架構概觀

架構圖

使用預設參數部署此解決方案會在您的 AWS 帳戶中部署下列元件。

AWS 上 AWS 架構的分散式負載測試



> **Note**
>
> AWS CloudFormation 資源是從 AWS 雲端開發套件 (AWS CDK) 建構模組建立。

使用 AWS CloudFormation 範本部署之解決方案元件的高階程序流程如下：

1. 分散式負載測試器 API 利用 Amazon API Gateway 來叫用解決方案的微服務 (AWS Lambda 函數）。
2. 微服務提供商業邏輯來管理測試資料並執行測試。
3. 這些微服務會與 Amazon Simple Storage Service (Amazon S3)、Amazon DynamoDB 和 AWS Step Functions 互動，以存放測試案例詳細資訊和結果，並協調測試執行。

4. Amazon Virtual Private Cloud (Amazon VPC) 網路拓撲部署，其中包含在 AWS Fargate 上執行的解決方案 Amazon Elastic Container Service (Amazon ECS) 容器。
5. 這些容器使用已安裝 Taurus 負載測試架構的 Amazon Linux 2023 基礎映像。Taurus 是一種開放原始碼測試自動化架構，支援 JMeter、K6、Locust 和其他測試工具。容器映像符合 Open Container Initiative (OCI) 規範，並由 AWS 在 Amazon Elastic Container Registry (Amazon ECR) 公有儲存庫中託管。如需詳細資訊，請參閱容器映像自訂。
6. 採用 AWS Amplify 技術的 Web 主控台會部署到設定為靜態 Web 託管的 S3 儲存貯體。
7. Amazon CloudFront 提供對解決方案網站儲存貯體內容的安全公開存取。
8. 在初始組態期間，解決方案會建立預設管理員角色 (IAM 角色），並將存取邀請傳送至客戶指定的使用者電子郵件地址。
9. Amazon Cognito 使用者集區會管理 主控台、分散式負載測試器 API 和 MCP Server 的使用者存取權。
10. 部署此解決方案之後，您可以使用 Web 主控台或 APIs 來建立和執行測試案例，以定義一系列任務。
11. 微服務會使用此測試案例，在指定區域中的 Fargate 上執行 ECS 任務。
12. 測試完成後，解決方案會將結果儲存在 S3 和 DynamoDB 中，並在 Amazon CloudWatch 中記錄輸出。
13. 如果您啟用即時資料選項，解決方案會在測試期間針對執行測試的每個區域，將 CloudWatch 日誌從 Fargate 任務傳送至 Lambda 函數。
14. Lambda 函數會將資料發佈至部署主要堆疊之區域中 AWS IoT Core 中的對應主題。Web 主控台會訂閱主題，並在測試執行時顯示即時資料。

> **Note**
>
> 下列步驟說明 AI 輔助負載測試分析的選用 MCP 伺服器整合。只有在解決方案部署期間選取 MCP 伺服器選項時，才會部署此元件。

15. MCP 用戶端 (AI 開發工具） 會連線至 AWS AgentCore Gateway 端點，以透過模型內容通訊協定存取分散式負載測試解決方案的資料。AgentCore Gateway 驗證使用者的 Cognito 身分驗證字符，以確保授權存取 MCP 伺服器。
16. 身分驗證成功後，AgentCore Gateway 會將 MCP 工具請求轉送至 DLT MCP Server Lambda 函數。Lambda 函數會將結構化資料傳回 AgentCore Gateway，並將其傳回 MCP 用戶端以進行 AI 輔助分析和洞察。
17. Lambda 函數會處理請求並查詢適當的 AWS 資源 (DynamoDB 資料表、S3 儲存貯體或 CloudWatch 日誌），以擷取請求的負載測試資料。

AWS Well-Architected 設計考量事項

此解決方案使用 AWS Well-Architected Framework 的最佳實務，協助客戶在雲端設計及操作可靠、安全、有效率且符合成本效益的工作負載。

本節說明 Well-Architected Framework 的設計原則和最佳實務如何讓此解決方案受益。

卓越營運

本節說明如何使用卓越營運支柱的原則和最佳實務來建構此解決方案。

- 所有資源都定義為使用從 AWS CDK 建構產生之 AWS CloudFormation 範本的基礎設施程式碼。
- 解決方案會在各種階段將指標推送至 CloudWatch，以提供 Lambda 函數、ECS 任務、S3 儲存貯體和其他解決方案元件的可觀測性。

安全

本節說明如何使用安全支柱的原則和最佳實務來建構此解決方案。

- Cognito 會驗證並授權 Web 主控台使用者和 API 請求。
- 所有服務間通訊都使用具有最低權限存取的 AWS Identity and Access Management (IAM) 角色，只包含所需的最低許可。
- 所有資料儲存，包括 S3 儲存貯體和 DynamoDB 資料表，都會使用 AWS 受管金鑰加密靜態資料。
- 在適用於稽核和合規目的的情況下，會啟用記錄、追蹤和版本控制。
- 根據預設，網路存取是私有的，並在可用的情況下啟用 VPC 端點，以將流量保留在 AWS 網路中。

> **Note**
>
> 解決方案會根據日誌量和成本考量，建立具有不同保留期的多個 CloudWatch 日誌群組：

日誌類型	保留期間
ECS 容器洞察	1 天
Step Functions、ECS 自訂日誌、API Gateway 存取日誌	1 年

日誌類型	保留期間
Lambda 執行期日誌	2 年
API Gateway 執行日誌	永不過期

您可以在 CloudWatch 主控台中根據您的需求修改這些保留期間。

可靠性

本節說明如何使用可靠性支柱的原則和最佳實務來建構此解決方案。

- 解決方案會盡可能使用 AWS 無伺服器服務 （例如：Lambda、API Gateway、Amazon S3、AWS Step Functions、Amazon DynamoDB 和 AWS Fargate)，以確保高可用性並從服務故障中復原。
- 所有運算處理都使用 Lambda 函數或 AWS Fargate 上的 Amazon ECS。
- 資料存放在 DynamoDB 和 Amazon S3 中，因此預設會保留在多個可用區域中。

效能效率

本節說明如何使用效能效率支柱的原則和最佳實務來建構此解決方案。

- 解決方案使用無伺服器架構，能夠視需要水平擴展。
- 解決方案可以在支援此解決方案中 AWS 服務的任何區域中啟動，例如：AWS Lambda、Amazon API Gateway、Amazon S3、AWS Step Functions、Amazon DynamoDB、Amazon ECS、AWS Fargate 和 Amazon Cognito。
- 解決方案全面使用 受管服務，以減少資源佈建和管理的操作負擔。
- 解決方案會每天自動測試和部署，以在 AWS 服務變更時達到一致性，並由解決方案架構師和主題專家針對要實驗和改進的領域進行審查。

成本最佳化

本節說明如何使用成本最佳化支柱的原則和最佳實務來建構此解決方案。

- 解決方案使用無伺服器架構；因此，客戶只需支付其使用量的費用。

- Amazon DynamoDB 會隨需擴展容量，因此您只需為所使用的容量付費。
- AWS Fargate 上的 AWS ECS 可讓您僅支付所使用的運算資源，無需預付費用。
- AWS AgentCore Gateway 做為分散式負載測試 API 的成本效益 Lambda 型代理，無需專用基礎設施，並透過無伺服器pay-per-request定價降低成本。

永續性

本節說明如何使用永續性支柱的原則和最佳實務來建構此解決方案。

- 相較於持續操作內部部署服務，解決方案使用受管無伺服器服務，將後端服務對環境的影響降至最低。
- 無伺服器服務可讓您視需要擴展或縮減規模。

架構詳細資訊

本節說明構成此解決方案的元件和 AWS 服務，以及這些元件如何一起運作的架構詳細資訊。

AWS 解決方案上的分散式負載測試包含三個高階元件：前端、後端和選用的 MCP 伺服器。

前端

前端提供與解決方案互動的界面，包括：

- 用於程式設計存取的負載測試 API
- 用於建立、排程和執行效能測試的 Web 主控台
- 用於 AI 輔助分析測試結果和錯誤的選用 MCP 伺服器

負載測試 API

AWS 上的分散式負載測試會將 Amazon API Gateway 設定為託管解決方案的 RESTful API。使用者可以透過隨附的 Web 主控台、RESTful API 和選用的 MCP 伺服器，安全地與負載測試系統互動。API 做為「前門」，用於存取存放在 Amazon DynamoDB 中的測試資料。您也可以使用 APIs來存取您建置到解決方案中的任何延伸功能。

此解決方案會利用 Amazon Cognito 使用者集區的使用者身分驗證功能。成功驗證使用者後，Amazon Cognito 會發出 JSON Web 字符，用於允許主控台向解決方案的 APIs(Amazon API Gateway 端點）提交請求。HTTPS 請求由主控台傳送至具有包含字符的授權標頭的 APIs。

根據請求，API Gateway 會叫用適當的 AWS Lambda 函數，對存放在 DynamoDB 資料表中的資料執行必要的任務、將測試案例儲存為 Amazon S3 中的 JSON 物件、擷取 Amazon CloudWatch 指標影像，並將測試案例提交至 AWS Step Functions 狀態機器。

如需解決方案 API 的詳細資訊，請參閱本指南的分散式負載測試 API 一節。

Web 主控台

此解決方案包含 Web 主控台，可用來設定和執行測試、監控執行中的測試，以及檢視詳細的測試結果。主控台是使用 Cloudscape 建置的 ReactJS 應用程式，這是用於建置直覺式 Web 應用程式的開放原始碼設計系統。主控台託管在 Amazon S3 中，並透過 Amazon CloudFront 存取。應用程式利用

AWS Amplify 與 Amazon Cognito 整合來驗證使用者。Web 主控台也包含檢視執行中測試的即時資料的選項，其中訂閱 AWS IoT Core 中的對應主題。

Web 主控台 URL 是 CloudFront 分佈網域名稱，可在 CloudFormation 輸出中作為主控台找到。啟動 CloudFormation 範本後，您也會收到一封電子郵件，其中包含 Web 主控台 URL 和登入的一次性密碼。

MCP 伺服器（選用）

選用的模型內容通訊協定 (MCP) 伺服器為 AI 開發工具提供額外的界面，以透過自然語言互動存取和分析負載測試資料。只有在解決方案部署期間選取 MCP Server 選項時，才會部署此元件。

MCP Server 可讓 AI 代理器查詢測試結果、分析效能指標，以及使用 Amazon Q、Claude 和其他 MCP 相容 AI 助理等工具深入了解負載測試資料。如需 MCP Server 架構和組態的詳細資訊，請參閱本節中的 MCP Server。

Backend

後端包含容器映像管道和負載測試引擎，您用來為測試產生負載。您可以透過前端與後端互動。此外，針對每個測試啟動的 AWS Fargate 任務上的 Amazon ECS 會以唯一的測試識別符 (ID) 標記。這些測試 ID 標籤可用來協助您監控此解決方案的成本。如需詳細資訊，請參閱《AWS Billing and Cost Management 使用者指南》中的使用者定義的成本分配標籤。

容器映像管道

此解決方案使用以 Amazon Linux 2023 建置的容器映像，做為已安裝 Taurus 負載測試架構的基礎映像。Taurus 是一種開放原始碼測試自動化架構，支援 JMeter、K6、Locust 和其他測試工具。AWS 會在 Amazon Elastic Container Registry (Amazon ECR) 公有儲存庫中託管此映像。解決方案會使用此映像在 AWS Fargate 叢集上的 Amazon ECS 中執行任務。

如需詳細資訊，請參閱本指南的容器映像自訂一節。

測試基礎設施

除了主要 CloudFormation 範本之外，解決方案還提供區域範本，以啟動在多個區域中執行測試所需的資源。解決方案會將此範本存放在 Amazon S3 中，並在 Web 主控台中提供其連結。每個區域堆疊都包含 VPC、AWS Fargate 叢集，以及用於處理即時資料的 Lambda 函數。

如需如何在其他 區域中部署測試基礎設施的詳細資訊，請參閱本指南的多區域部署一節。

負載測試引擎

分散式負載測試解決方案使用 Amazon Elastic Container Service (Amazon ECS) 和 AWS Fargate 模擬多個區域的數千名並行使用者，以持續的速度產生 HTTP 請求。

您可以使用隨附的 Web 主控台來定義測試參數。解決方案使用這些參數來產生 JSON 測試案例，並將其存放在 Amazon S3 中。如需測試指令碼和測試參數的詳細資訊，請參閱本節中的測試類型。

AWS Step Functions 狀態機器會在 AWS Fargate 叢集中執行和監控 Amazon ECS 任務。AWS Step Functions 狀態機器包含 ecr-checker AWS Lambda 函數、task-status-checker AWS Lambda 函數、任務執行器 AWS Lambda 函數、任務取消器 AWS Lambda 函數，以及結果剖析器 AWS Lambda 函數。如需工作流程的詳細資訊，請參閱本指南的測試工作流程一節。如需測試結果的詳細資訊，請參閱本指南的測試結果一節。如需測試取消工作流程的詳細資訊，請參閱本指南的測試取消工作流程一節。

如果您選取即時資料，解決方案會透過對應至該區域中 Fargate 任務的 CloudWatch 日誌，real-time-data-publisher Lambda 函數。解決方案接著會在您啟動主要堆疊的區域中，處理資料並將其發佈至 AWS IoT Core 中的主題。如需詳細資訊，請參閱本指南的即時資料一節。

MCP 伺服器

選用的模型內容通訊協定 (MCP) 伺服器整合可讓 AI 代理器透過自然語言互動，以程式設計方式存取和分析負載測試資料。只有在解決方案部署期間選取 MCP Server 選項時，才會部署此元件。

MCP 伺服器可做為 AI 開發工具和 DLT 部署之間的橋樑，提供標準化界面，以智慧分析效能測試結果。架構整合了數個 AWS 服務，為 AI 代理器互動建立安全、可擴展的界面：

AWS AgentCore Gateway

AWS AgentCore Gateway 是一項全受管服務，可為 MCP 伺服器提供標準化託管和通訊協定管理。在此解決方案中，AgentCore Gateway 會在請求存取您的負載測試資料時，做為 AI 代理器連線的公有端點。

服務會處理所有 MCP 通訊協定通訊，包括工具探索、身分驗證字符驗證和請求路由。AgentCore Gateway 以多租戶服務的形式運作，內建安全保護，可防範公有端點的常見威脅，同時驗證每個請求的 Cognito 字符簽章和宣告。

DLT MCP 伺服器 Lambda

DLT MCP Server Lambda 函數是一種自訂無伺服器元件，可處理來自 AI 代理器的 MCP 請求，並將其轉換為針對 DLT 資源的查詢。

此 Lambda 函數充當 MCP 整合的智慧層、從 DynamoDB 資料表擷取測試結果、存取存放在 S3 儲存貯體中的效能成品，以及查詢 CloudWatch 日誌以取得詳細的執行資訊。Lambda 函數實作唯讀存取模式，並將原始 DLT 資料轉換為結構化的 AI 易記格式，客服人員可以輕鬆解譯和分析。

身分驗證整合

身分驗證系統利用您現有的 Cognito 使用者集區基礎設施，在 Web 主控台和 MCP Server 介面之間維持一致的存取控制。

此整合使用 OAuth 2.0 字符型身分驗證。使用者透過 Cognito 登入程序驗證一次，並接收可用於 UI 互動和 MCP 伺服器存取的字符。系統會維護與 Web 界面相同的許可界限和存取控制，確保使用者只能透過 AI 代理器存取他們可以透過主控台存取的相同負載測試資料。

此解決方案中的 AWS 服務

此解決方案包含下列 AWS 服務：

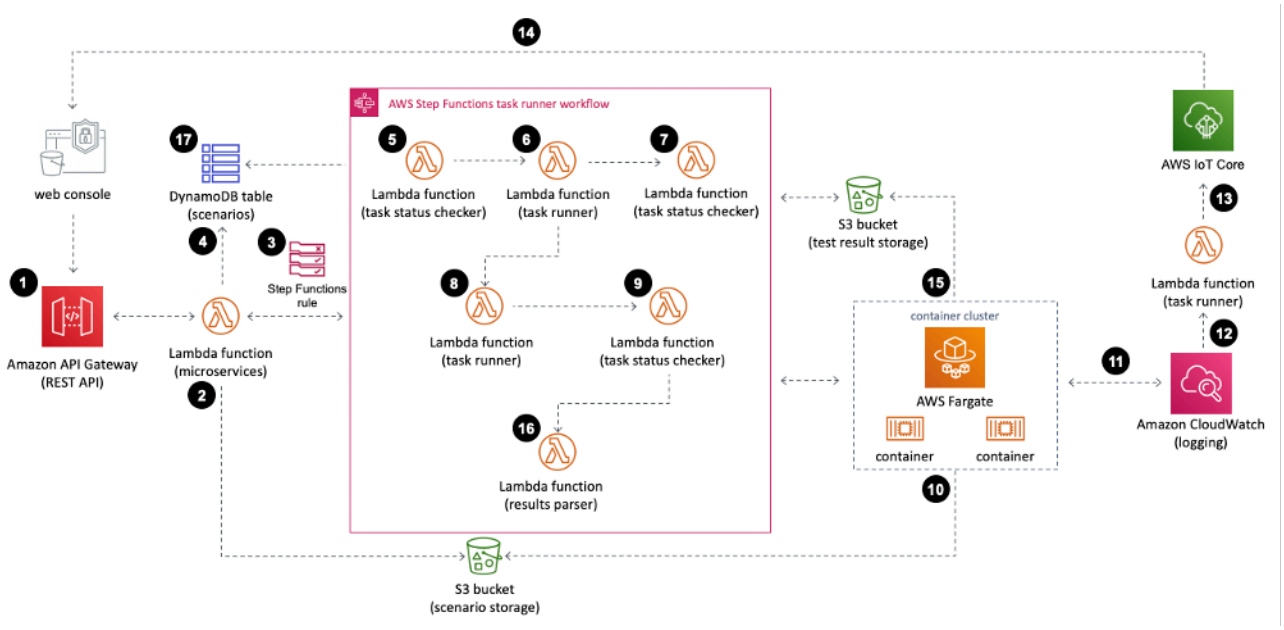
AWS 服務	說明
Amazon API Gateway	核心。在解決方案中託管 REST API 端點。
AWS CloudFormation	核心。管理解決方案基礎設施的部署。
Amazon CloudFront	核心。提供 Amazon S3 中託管的 Web 內容。
Amazon CloudWatch	核心。存放解決方案日誌和指標。
Amazon Cognito	核心。處理 API 的使用者管理和身分驗證。
Amazon DynamoDB	核心。存放部署資訊和測試案例詳細資訊和結果。
Amazon Elastic Container Service	核心。在 AWS Fargate 容器上部署和管理獨立的 Amazon ECS 任務。
AWS Fargate	核心。託管解決方案的 Amazon ECS 容器
AWS Identity and Access Management	核心。處理使用者角色和許可管理。
AWS Lambda	核心。提供 APIs 實作、測試結果剖析和啟動工作者/領導任務的邏輯。

AWS 服務	說明
AWS Step Functions	核心。協調在指定區域中的 AWS Fargate 任務上佈建 Amazon ECS 容器
AWS Amplify	支援。提供採用 AWS Amplify 技術的 Web 主控台。
Amazon CloudWatch Events	支援。將測試排程在指定的日期或週期性日期自動開始。
Amazon Elastic Container Registry	支援。在公有 ECR 儲存庫中託管容器映像。
AWS IoT Core	支援。訂閱 AWS IoT Core 中的對應主題，即可檢視執行中測試的即時資料。
AWS Systems Manager	支援。提供資源操作和成本資料的應用程式層級資源監控和視覺化。
Amazon S3	支援。託管靜態 Web 內容、日誌、指標和測試資料。
Amazon Virtual Private Cloud	支援。包含解決方案在 AWS Fargate 上執行的 Amazon ECS 容器。
Amazon Bedrock AgentCore	支援、選用。託管解決方案選用的遠端模型內容通訊協定 (MCP) Server for AI 代理程式與 API 整合。

AWS 上的分散式負載測試如何運作

下列詳細明細顯示執行測試案例所涉及的步驟。

測試工作流程



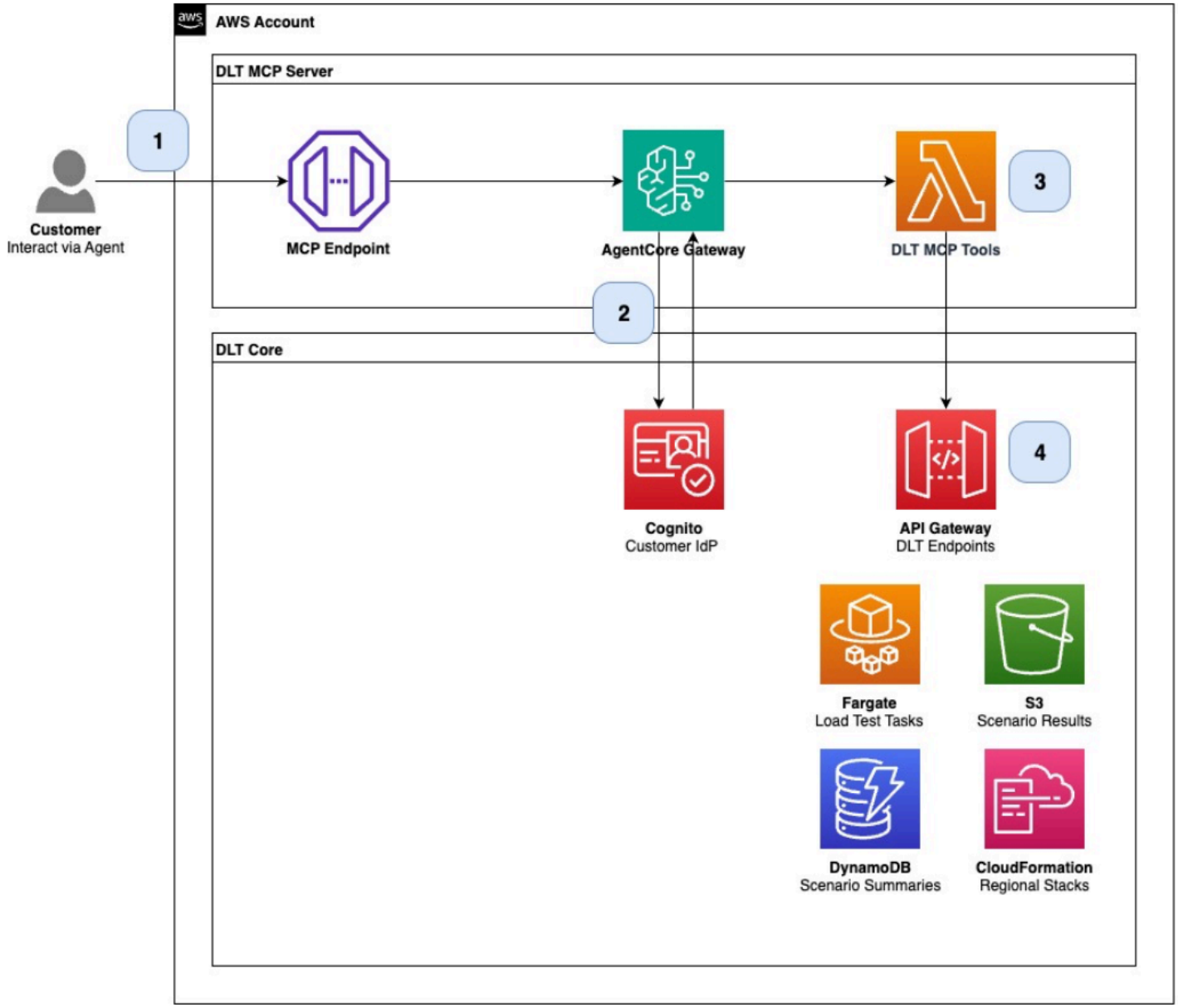
1. 您可以使用 Web 主控台，將包含組態詳細資訊的測試案例提交至解決方案的 API。
2. 測試案例組態會以 JSON 檔案 () 的形式上傳至 Amazon Simple Storage Service (Amazon S3)`s3://<bucket-name>/test-scenarios/<$TEST_ID>/<$TEST_ID>.json`。
3. AWS Step Functions 狀態機器會使用測試 ID、任務計數、測試類型和檔案類型做為 AWS Step Functions 狀態機器輸入來執行。如果已排程測試，它會先建立 CloudWatch Events 規則，該規則會在指定的日期觸發 AWS Step Functions。如需排程工作流程的詳細資訊，請參閱本指南的測試排程工作流程一節。
4. 組態詳細資訊會存放在案例 Amazon DynamoDB 資料表中。
5. 在 AWS Step Functions 任務執行器工作流程中， task-status-checker AWS Lambda 函數會檢查 Amazon Elastic Container Service (Amazon ECS) 任務是否已針對相同的測試 ID 執行。如果發現具有相同測試 ID 的任務正在執行，則會導致錯誤。如果 AWS Fargate 叢集中沒有執行中的 Amazon ECS 任務，則函數會傳回測試 ID、任務計數和測試類型。
6. 任務執行器 AWS Lambda 函數會從上一個步驟取得任務詳細資訊，並在 AWS Fargate 叢集中執行 Amazon ECS 工作者任務。Amazon ECS API 使用 RunTask 動作來執行工作者任務。這些工作者任務會啟動，然後等待領導者任務的開始訊息，以開始測試。RunTask 動作限制為每個定義 10 個任務。如果您的任務計數超過 10，任務定義將執行多次，直到所有工作者任務都已啟動為止。函數也會產生字首，以區分結果剖析 AWS Lambda 函數中的目前測試。
7. task-status-checker AWS Lambda 函數會檢查所有 Amazon ECS 工作者任務是否以相同的測試 ID 執行。如果任務仍在佈建，它會等待一分鐘並再次檢查。執行所有 Amazon ECS 任務後，它會傳回測試 ID、任務計數、測試類型、所有任務 IDs和字首，並將其傳遞給任務執行器函數。

8. 任務執行器 AWS Lambda 函數會再次執行，這次啟動單一 Amazon ECS 任務以做為領導節點。此 ECS 任務會將啟動測試訊息傳送至每個工作者任務，以同時啟動測試。
9. task-status-checker AWS Lambda 函數會再次檢查 Amazon ECS 任務是否以相同的測試 ID 執行。如果任務仍在執行中，它會等待一分鐘並再次檢查。一旦沒有執行中的 Amazon ECS 任務，它會傳回測試 ID、任務計數、測試類型和字首。
10. 當任務執行器 AWS Lambda 函數在 AWS Fargate 叢集中執行 Amazon ECS 任務時，每個任務會從 Amazon S3 下載測試組態並開始測試。
11. 測試執行後，Amazon CloudWatch 會記錄每個任務的平均回應時間、並行使用者數量、成功請求數量，以及失敗的請求數量，並且可以在 CloudWatch 儀表板中檢視。
12. 如果您在測試中包含即時資料，解決方案會使用訂閱篩選條件在 CloudWatch 中篩選即時測試結果。然後，解決方案會將資料傳遞至 Lambda 函數。
13. Lambda 函數接著會建構收到的資料，並將其發佈至 AWS IoT Core 主題。
14. Web 主控台會訂閱測試的 AWS IoT Core 主題，並接收發佈至主題的資料，以在測試執行時繪製即時資料的圖表。
15. 測試完成後，容器映像會將詳細報告匯出為 XML 檔案至 Amazon S3。每個檔案都會獲得檔案名稱的 UUID。例如，s3：//dlte-bucket/test-scenarios/<$TEST_ID>/results/<$UUID>.json。
16. 當 XML 檔案上傳到 Amazon S3 時，結果剖析器 AWS Lambda 函數會從字首開始讀取 XML 檔案中的結果，並剖析和彙總所有結果為一個摘要結果。
17. 結果剖析器 AWS Lambda 函數會將彙總結果寫入 Amazon DynamoDB 資料表。

MCP 伺服器工作流程 （選用）

如果您部署選用的 MCP Server 整合，AI 代理器可以透過下列工作流程存取和分析負載測試資料：

MCP 伺服器架構



1. 客戶互動 - 客戶透過 AWS AgentCore Gateway 託管的 MCP 端點與 DLT 的 MCP 互動。AI 代理器會連線至此端點，請求存取以載入測試資料。
2. 授權 - AgentCore Gateway 會對 Solution Cognito 使用者集區應用程式用戶端處理授權。閘道會驗證使用者的 Cognito 權杖，以確保他們具有存取 DLT MCP 伺服器的許可。授權使用者可透過僅限唯讀操作的代理工具存取獲得存取權。
3. 工具規格 - AgentCore Gateway 連接到 DLT MCP Server Lambda 函數。工具規格定義 AI 代理器可用來與負載測試資料互動的可用工具。
4. 唯讀 API 存取 - Lambda 函數的範圍限定為透過現有 DLT API Gateway 端點的唯讀 API 存取。函數提供四個主要操作：

- 列出案例 - 從 DynamoDB 案例資料表擷取測試案例的清單
- 取得案例測試結果 - 從 DynamoDB 和 S3 存取特定案例的詳細測試結果
- 取得 Fargate 負載測試執行器 - 查詢在 ECS 叢集中執行 Fargate 任務的相關資訊
- 取得可用的區域堆疊 - 從 CloudFormation 擷取已部署區域基礎設施的相關資訊

MCP Server 整合利用現有的 DLT 基礎設施 (API Gateway、Cognito、DynamoDB、S3)，提供安全、唯讀的存取權，以測試 AI 驅動的分析和洞察資料。

設計考量

本節說明 AWS 上分散式負載測試解決方案的重要設計決策和組態選項，包括支援的應用程式、測試類型、排程選項和部署考量。

支援的應用程式

只要您有從 AWS 帳戶到應用程式的網路連線，此解決方案支援測試雲端型應用程式和內部部署應用程式。解決方案支援使用 HTTP 或 HTTPS 通訊協定APIs。

測試類型

AWS 上的分散式負載測試支援多種測試類型：簡易 HTTP 端點測試、JMeter、K6 和 Locust。

簡單 HTTP 端點測試

Web 主控台提供 HTTP 端點組態介面，可讓您測試任何 HTTP 或 HTTPS 端點，而無需撰寫自訂指令碼。您可以定義端點 URL，從下拉式選單中選取 HTTP 方法 (GET、POST、PUT、DELETE 等），並選擇性地新增自訂請求標頭和內文承載。此組態可讓您使用自訂授權字符、內容類型，或應用程式所需的任何其他 HTTP 標頭和請求內文來測試 APIs。

JMeter 測試

使用 Web 主控台建立測試案例時，您可以上傳 JMeter 測試指令碼。解決方案會將指令碼上傳至案例 S3 儲存貯體。當 Amazon ECS 任務執行時，他們會從 S3 下載 JMeter 指令碼並執行測試。

> **Important**
>
> 雖然您的 JMeter 指令碼可能定義並行（虛擬使用者）、交易速率 (TPS)、漸進測試時間和其他負載參數，但解決方案會使用您在測試建立期間在流量形狀畫面中指定的值來覆寫這些組

態。流量形狀組態可控制測試執行的任務計數、並行（每個任務的虛擬使用者）、漸進測試持續時間和保留持續時間。

如果您有 JMeter 輸入檔案，您可以將輸入檔案與 JMeter 指令碼一起壓縮。您可以在建立測試案例時選擇 zip 檔案。

如果您想要包含外掛程式，任何包含在綁定 zip 檔案中 /plugins 子目錄中的 .jar 檔案都會複製到 JMeter 擴充功能目錄，並可用於負載測試。

> **Note**
>
> 如果您在 JMeter 指令碼檔案中包含 JMeter 輸入檔案，則必須在 JMeter 指令碼檔案中包含輸入檔案的相對路徑。此外，輸入檔案必須位於相對路徑。例如，當您的 JMeter 輸入檔案和指令碼檔案位於 /home/user 目錄中，而您參考 JMeter 指令碼檔案中的輸入檔案時，輸入檔案的路徑必須為 。/INPUT_FILES。如果您改為使用 /home/user/INPUT_FILES，則測試將會失敗，因為它將無法找到輸入檔案。

如果您包含 JMeter 外掛程式，.jar 檔案必須封裝在 zip 檔案根目錄中名為 /plugins 的子目錄中。相對於 zip 檔案的根目錄，jar 檔案的路徑必須是 ./plugins/BUNDLED_PLUGIN.jar。

如需如何使用 JMeter 指令碼的詳細資訊，請參閱 JMeter 使用者手冊。

K6 測試

解決方案支援 K6 架構型測試。K6 是根據 AGPL-3.0 授權發行。解決方案會在建立新的 K6 測試時顯示授權確認訊息。您可以在封存檔案中上傳 K6 測試檔案以及任何必要的輸入檔案。

> **Important**
>
> 雖然 K6 指令碼可能會定義並行（虛擬使用者）、階段、閾值和其他負載參數，但解決方案會使用您在測試建立期間在流量形狀畫面中指定的值來覆寫這些組態。流量形狀組態可控制測試執行的任務計數、並行（每個任務的虛擬使用者）、漸進測試持續時間和保留持續時間。

Locust 測試

解決方案支援 Locust 架構型測試。您可以上傳 Locust 測試檔案，以及封存檔案中任何必要的輸入檔案。

> **Important**
>
> 雖然您的 Locust 指令碼可能會定義並行（使用者計數）、產生率和其他負載參數，但解決方案會使用您在測試建立期間在流量形狀畫面中指定的值來覆寫這些組態。流量形狀組態可控制測試執行的任務計數、並行（每個任務的虛擬使用者）、漸進測試持續時間和保留持續時間。

排程測試

解決方案提供三種執行負載測試的執行時間選項：

- 立即執行 - 在建立後立即執行載入測試
- 執行一次 - 在未來的特定日期和時間執行測試
- 在排程上執行 - 使用 Cron 表達式建立週期性測試以定義排程

選取執行一次時，您會以 24 小時格式指定執行時間，以及負載測試應開始執行的執行日期。

當您選取在排程上執行時，您可以手動輸入 Cron 表達式或從常見的 Cron 模式選取（例如每小時、每天特定時間、工作日或每月）。Cron 表達式使用精細的排程格式，其中包含分鐘、小時、月中的日、月、週中的日和年的欄位。您也必須指定過期日期，以定義排程測試何時應停止執行。如需排程如何運作的詳細資訊，請參閱本指南的測試排程工作流程一節。

> **Note**
>
> - 測試持續時間：考慮排程時的總測試持續時間。例如，具有 10 分鐘漸進測試時間和 40 分鐘保留時間的測試大約需要 80 分鐘才能完成。
> - 最小間隔：確保排程測試之間的間隔長於預估測試持續時間。例如，如果測試大約需要 80 分鐘，請將其排程為每 3 小時執行一次。
> - 每小時限制：系統不允許只以一小時的差異排程測試，即使預估測試持續時間少於一小時。

並行測試

此解決方案會為每個測試建立 Amazon CloudWatch 儀表板，以即時顯示 Amazon ECS 叢集中執行之所有任務的合併輸出。CloudWatch 儀表板會顯示平均回應時間、並行使用者數、成功請求數和失敗請求數。解決方案會依秒彙總每個指標，並每分鐘更新儀表板。

使用者管理

在初始組態期間，您會提供 Amazon Cognito 用來授予您存取解決方案 Web 主控台的使用者名稱和電子郵件地址。主控台不提供使用者管理。若要新增其他使用者，您必須使用 Amazon Cognito 主控台。如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的在使用者集區中管理使用者。

如需將現有使用者遷移至 Amazon Cognito 使用者集區，請參閱 AWS 部落格方法，以將使用者遷移至 Amazon Cognito 使用者集區。

區域部署

此解決方案使用 Amazon Cognito，僅適用於特定 AWS 區域。因此，您必須在可使用 Amazon Cognito 的區域部署此解決方案。如需依區域分類的最新服務可用性，請參閱 AWS 區域服務清單。

規劃您的部署

本節說明部署解決方案之前應檢閱的成本、安全性、支援的區域、配額和其他考量事項。

成本

您必須負責執行此解決方案時所使用的 AWS 服務成本。總成本取決於負載測試執行的數量、這些測試的持續時間，以及產生的資料量。截至此修訂，在美國東部（維吉尼亞北部）區域中使用預設設定執行此解決方案的預估成本約為每月 30.90 美元。

下表提供在美國東部（維吉尼亞北部）區域中使用預設參數部署此解決方案一個月的範例成本明細。

AWS 服務	維度	成本 【美元】
AWS Fargate	執行 30 小時的 10 個隨需任務（使用兩個 vCPUs 和 4 GB 記憶體）	29.62 美元
Amazon DynamoDB	1，000 個隨需寫入容量單位 1，000 個隨需讀取容量單位	0.0015 美元
AWS Lambda	1，000 個請求 總持續時間為 10 分鐘	1.25 美元
AWS Step Functions	1，000 個狀態轉換	$0.025
總計：		每月 30.90 美元

解決方案資源會以 Key=SolutionId 和 Value=SO0062 標記。您可以依照文件啟用標籤來啟用標籤金鑰 SolutionId。標籤啟用後，您可以依照文件建立成本類別來建立成本類別規則。您可以監控成本類別主控台並選擇成本類別名稱，以檢視解決方案產生的成本。

我們建議您透過 AWS Cost Explorer 建立預算，以協助管理成本。價格可能變動。如需完整詳細資訊，請參閱此解決方案中使用的每個 AWS 服務的定價網頁。

> **Note**
>
> 預設任務組態每個任務使用 2 個 vCPUs和 4 GB 的記憶體。如果您的負載測試不需要這些資源，您可以降低這些資源以降低成本。相反地，您可以增加資源，以支援每個任務更高的並行。如需詳細資訊，請參閱本指南中的增加容器資源一節。

> **Note**
>
> 此解決方案提供在執行測試時包含即時資料的選項。此功能需要額外的 AWS Lambda 函數和會產生額外費用的 AWS IoT Core 主題。

MCP 伺服器額外費用 （選用）

下表提供 MCP Server 與美國東部 （維吉尼亞北部） 區域中定價整合一個月的成本明細。

服務元件	維度	成本 【美元】
AgentCore Gateway - 工具索引	10 個工具 × 每 100 個工具 $0.02	0.002 美元
AgentCore Gateway - 搜尋 API	每 1，000 個 10，000 個互動 × 0.025 美元	0.25 美元
AgentCore Gateway - API 調用	每 1，000 個 50，000 個調用 × 0.005 美元	0.25 美元
AWS Lambda 函式	根據用量的變數 （典型工作負載）	5.00 - 20.00 美元
預估總額外成本：		每月 5.50 - 20.50 美元

價格可能變動。如需 AgentCore Gateway 定價的完整詳細資訊，請參閱 Amazon Bedrock 定價 (AgentCore Gateway 區段）。如需 Lambda 定價，請參閱 AWS Lambda 定價。

安全

當您在 AWS 基礎設施上建置系統時，安全責任將由您與 AWS 共同承擔。此共同責任模型可減少您的營運負擔，因為 AWS 會操作、管理和控制元件，包括主機作業系統、虛擬化層，以及服務營運所在設施的實體安全性。如需 AWS 安全性的詳細資訊，請造訪 AWS Cloud Security。

IAM 角色

AWS Identity and Access Management (IAM) 角色可讓客戶將精細的存取政策和許可指派給 AWS 雲端上的服務和使用者。此解決方案會建立 IAM 角色，授予解決方案的 AWS Lambda 函數建立區域資源的存取權。

Amazon CloudFront

此解決方案會部署託管在 Amazon S3 儲存貯體中的 Web UI，該儲存貯體由 Amazon CloudFront 分發。為了協助減少延遲並改善安全性，此解決方案包含具有原始存取身分的 CloudFront 分佈，這是提供解決方案網站儲存貯體內容公開存取權的 CloudFront 使用者。根據預設，CloudFront 分佈會使用 TLS 1.2 來強制執行最高層級的安全通訊協定。如需詳細資訊，請參閱《Amazon CloudFront 開發人員指南》中的限制對 Amazon S3 原始伺服器的存取。 Amazon CloudFront

CloudFront 會啟用其他安全緩解措施，將 HTTP 安全標頭附加至每個檢視器回應。如需詳細資訊，請參閱在 CloudFront 回應中新增或移除 HTTP 標頭。

此解決方案使用預設 CloudFront 憑證，其具有 TLS v1.0 的最低支援安全通訊協定。若要強制使用 TLS v1.2 或 TLS v1.3，您必須使用自訂 SSL 憑證，而非預設 CloudFront 憑證。如需詳細資訊，請參閱如何將 CloudFront 分佈設定為使用 SSL/TLS 憑證。

Amazon API Gateway

此解決方案部署邊緣最佳化的 Amazon API Gateway 端點，以使用預設 APIs Gateway 端點而非自訂網域為負載測試功能提供 RESTful API。對於使用預設端點的邊緣最佳化 APIs，API Gateway 會使用 TLS-1-0 安全政策。如需詳細資訊，請參閱《Amazon APIs》中的使用 REST API。 Amazon API Gateway

此解決方案使用預設 API Gateway 憑證，其具有 TLS v1.0 的最低支援安全通訊協定。若要強制使用 TLS v1.2 或 TLS v1.3，您必須使用具有自訂 SSL 憑證的自訂網域，而非預設 API Gateway 憑證。如需詳細資訊，請參閱設定 REST APIs 的自訂網域名稱。

AWS Fargate 安全群組

根據預設，此解決方案會將 AWS Fargate 安全群組的傳出規則開放給大眾。如果您想要封鎖 AWS Fargate 到處傳送流量，請將傳出規則變更為特定無類別網域間路由 (CIDR)。

此安全群組也包含傳入規則，允許連接埠 50，000 上的本機流量流向屬於相同安全群組的任何來源。這用於允許容器彼此通訊。

網路壓力測試

您有責任在網路壓力測試政策下使用此解決方案。此政策涵蓋的情況包括當您計劃直接從 Amazon EC2 執行個體執行大量網路測試到其他位置，例如其他 Amazon EC2 執行個體、AWS 屬性/服務或外部端點。這些測試有時稱為壓力測試、負載測試或遊戲日測試。大多數客戶測試不會屬於此政策；不過，如果您認為產生的流量總計維持超過 1 分鐘、超過 1 Gbps（每秒 10 億位元）或超過 1 Gpps（每秒 10 億封包）。

限制對公有使用者介面的存取

若要限制存取 IAM 和 Amazon Cognito 提供的身分驗證和授權機制以外的公開使用者介面，請使用 AWS WAF (Web 應用程式防火牆）安全自動化解決方案。

此解決方案會自動部署一組可篩選常見 Web 型攻擊的 AWS WAF 規則。使用者可以從預先設定的保護功能中選擇，這些功能定義 AWS WAF Web 存取控制清單 (Web ACL) 中包含的規則。

MCP 伺服器安全性（選用）

如果您部署選用的 MCP Server 整合，解決方案會使用 AWS AgentCore Gateway 提供安全的存取，以載入 AI 代理器的測試資料。AgentCore Gateway 會驗證每個請求的 Amazon Cognito 身分驗證字符，確保只有授權使用者可以存取 MCP 伺服器。MCP Server Lambda 函數實作唯讀存取模式，防止 AI 代理器修改測試組態或結果。所有 MCP 伺服器互動都使用與 Web 主控台相同的許可界限和存取控制。

支援的 AWS 區域

此解決方案使用 Amazon Cognito 服務，目前尚未在所有 AWS 區域提供。如需各區域 AWS 服務的最新可用性，請參閱 AWS 區域服務清單。

AWS 上的分散式負載測試可在下列 AWS 區域使用：

區域名稱	
美國東部 (俄亥俄)	亞太地區 (東京)
美國東部 (維吉尼亞北部)	加拿大 (中部)
美國西部 (加利佛尼亞北部)	歐洲 (法蘭克福)
美國西部 (奧勒岡)	歐洲 (愛爾蘭)
亞太地區 (孟買)	歐洲 (倫敦)
亞太地區 (首爾)	Europe (Paris)
亞太地區 (新加坡)	歐洲 (斯德哥爾摩)
亞太地區 (雪梨)	南美洲 (聖保羅)

MCP 伺服器支援的 AWS 區域（選用）

如果您打算部署選用的 MCP Server 整合，您必須在可使用 AWS AgentCore Gateway 的 AWS 區域中部署解決方案。MCP 伺服器功能僅適用於下列 AWS 區域：

區域名稱	區域代碼
美國東部 (維吉尼亞北部)	us-east-1
美國西部 (奧勒岡)	us-west-2
亞太區域 (新加坡)	ap-southeast-1
亞太區域 (雪梨)	ap-southeast-2
亞太區域 (東京)	ap-northeast-1
歐洲 (法蘭克福)	eu-central-1
歐洲 (愛爾蘭)	eu-west-1
歐洲 (倫敦)	eu-west-2

區域名稱	區域代碼
歐洲 (巴黎)	eu-west-3

如需依區域分類的 AWS AgentCore Gateway 最新可用性，請參閱《AWS AgentCore Gateway 開發人員指南》中的 AWS AgentCore Gateway 端點和配額。 AgentCore

配額

服務配額 (也稱為限制) 是您 AWS 帳戶的服務資源或操作數目最大值。

此解決方案中 AWS 服務的配額

請確定您為此解決方案中實作的每個服務有足夠的配額。如需詳細資訊，請參閱 AWS 服務配額。

使用以下連結前往該服務的 頁面。若要在不切換頁面的情況下檢視文件中所有 AWS 服務的服務配額，請改為檢視 PDF 中服務端點和配額頁面中的資訊。

AWS CloudFormation 配額

您的 AWS 帳戶具有在此解決方案中啟動堆疊時應注意的 AWS CloudFormation 配額。透過了解這些配額，您可以避免限制會阻止您成功部署此解決方案的錯誤。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》中的 中的 AWS CloudFormation 配額。 AWS CloudFormation

負載測試配額

可以使用 AWS Fargate 啟動類型在 Amazon ECS 中執行的任務數量上限，取決於任務的 vCPU 大小。AWS 上分散式負載測試的預設任務大小為 2 個 vCPU。若要查看目前的預設配額，請參閱 Amazon ECS 服務配額。目前的帳戶配額可能與列出的配額不同。若要檢查帳戶特定的配額，請在 AWS 管理主控台中檢查 Fargate 隨需 vCPU 資源計數的服務配額。如需如何請求增加的說明，請參閱《AWS 一般參考指南》中的 AWS 服務配額。

Amazon Linux 2023 容器映像 （ 已安裝 Taurus) 不會限制每個任務的並行連線，但這並不表示它可以支援不限數量的使用者。若要判斷容器可以為測試產生的並行使用者數量，請參閱本指南的判斷使用者數量一節。

> **Note**
>
> 根據預設設定，並行使用者的建議限制為 200 個使用者。

並行測試

此解決方案會為每個測試建立 Amazon CloudWatch 儀表板，以即時顯示 Amazon ECS 叢集中執行之所有任務的合併輸出。CloudWatch 儀表板會顯示平均回應時間、並行使用者數、成功請求數和失敗請求數。解決方案會依秒彙總每個指標，並每分鐘更新儀表板。

Amazon EC2 測試政策

只要您的網路流量低於 1 Gbps，就不需要 AWS 的核准，即可使用此解決方案執行負載測試。如果您的測試會產生超過 1 Gbps，請聯絡 AWS。如需詳細資訊，請參閱 Amazon EC2 測試政策。

Amazon CloudFront 負載測試政策

如果您打算對 CloudFront 端點進行負載測試，請參閱《Amazon CloudFront 開發人員指南》中的負載測試準則。我們也建議將流量分散到多個任務和區域。為負載測試提供至少 30 分鐘的漸進測試時間。對於每秒傳送超過 500，000 個請求或要求超過 300 Gbps 資料的負載測試，我們建議您先取得傳送流量的預先核准。CloudFront 可能會調節影響 CloudFront 服務可用性的未核准負載測試流量。

監控部署後的解決方案

部署解決方案之後，建議您使用 Amazon CloudWatch 警示和指標持續監控解決方案的資源。

設定 CloudWatch 警示

您可以設定 CloudWatch 警示來監控關鍵指標，並在超過閾值時接收通知。請考慮為下列資源設定警示：

Amazon CloudFront 分佈指標

監控 CloudFront 分佈效能和錯誤。如需詳細資訊，請參閱《Amazon CloudFront 開發人員指南》中的 CloudFront 分佈指標。 Amazon CloudFront

Amazon API Gateway 指標

監控 API 請求率、延遲和錯誤。如需詳細資訊，請參閱《Amazon API Gateway 開發人員指南》中的 Amazon API Gateway 維度和指標。 Amazon API Gateway

AWS Lambda 函數指標

監控解決方案微服務的 Lambda 函數叫用、持續時間、錯誤和調節。

Amazon ECS 和 AWS Fargate 指標

在負載測試期間監控任務 CPU 和記憶體使用率，以確保有足夠的資源。

Amazon DynamoDB 指標

監控讀取和寫入容量耗用量、限流請求和延遲。

部署解決方案

此解決方案使用 AWS CloudFormation 範本和堆疊來自動化其部署。CloudFormation 範本會指定此解決方案中包含的 AWS 資源及其屬性。CloudFormation 堆疊會佈建範本中所述的資源。

部署程序概觀

在部署解決方案之前，請檢閱本指南先前討論的成本、架構、安全性和其他考量事項。

部署時間：主要堆疊大約 15 分鐘，加上每個額外區域 5 分鐘

AWS CloudFormation 範本

您可以在部署之前下載此解決方案的 CloudFormation 範本。此解決方案使用 AWS CloudFormation 在 AWS 上自動化分散式負載測試的部署。它包含下列 AWS CloudFormation 範本，您可以在部署之前下載：

View template

distributed-load-testing-on-aws.template - 使用此範本啟動解決方案和所有相關元件。預設組態會部署此解決方案區段中 AWS 服務的核心和支援服務，但您可以自訂範本以符合您的特定需求。

> **Note**
>
> AWS CloudFormation 資源是從 AWS 雲端開發套件 (AWS CDK) 建構模組建立的。如果您先前已部署此解決方案，請參閱更新解決方案以取得更新指示。

啟動 堆疊

請依照下列步驟，在您的帳戶中部署分散式負載測試。

> **Note**
>
> 此解決方案包含 AWS 的資料收集指標。我們使用這些資料更好地了解客戶使用此解決方案、相關服務和產品的方式。AWS 擁有透過此問卷收集的資料。資料收集受 AWS 隱私權聲明約束。

此自動化 AWS CloudFormation 範本會在 AWS 上部署分散式負載測試。

> **Note**
>
> 您需負責支付執行此解決方案時所使用的 AWS 服務成本。如需詳細資訊，請參閱本指南中的成本區段，並參閱此解決方案中使用的每個 AWS 服務的定價網頁。

1. 登入 AWS 管理主控台，然後選取按鈕以啟動 CloudFormation 範本。

   或者，您可以下載範本做為自有實作的起點。
2. 根據預設，範本會在美國東部 (維吉尼亞北部)區域中啟動。若要在不同的 AWS 區域中啟動此解決方案，請使用主控台導覽列中的區域選擇器。

   > **Note**
   >
   > 此解決方案使用 Amazon Cognito，目前僅適用於特定 AWS 區域。因此，您必須在可使用 Amazon Cognito 的 AWS 區域中啟動此解決方案。如需依區域分類的最新服務可用性，請參閱 AWS 區域服務清單。
3. 在建立堆疊頁面上，驗證 Amazon S3 URL 文字方塊中顯示的範本 URL 是否正確，然後選擇下一步。
4. 在指定堆疊詳細資訊頁面上，為您的解決方案堆疊指派名稱。
5. 在參數下，檢閱範本的參數並視需要修改。此解決方案使用下列預設值。

參數	預設	說明
管理員名稱	需要輸入	初始解決方案管理員的使用者名稱。
管理員電子郵件	*<####>*	管理員使用者的電子郵件地址。啟動後，系統會傳送一封電子郵件到此地址，其中包含主控台登入指示。

參數	預設	說明
現有的 VPC ID	<選用輸入>	如果您有要使用且已建立的 VPC，請在部署堆疊的相同區域中輸入現有 VPC 的 ID。例如，vpc-1a2b3c4d5e6f。
第一個現有子網路	<選用輸入>	現有 VPC 中第一個子網路的 ID。此子網路需要網際網路的路由，才能提取容器映像以執行測試。例如，netnet-7h8i9j0k。
第二個現有子網路	<選用輸入>	現有 VPC 內第二個子網路的 ID。此子網路需要網際網路的路由，才能提取容器映像以執行測試。例如，netnet-1x2y3z。
為解決方案提供有效的 CIDR 區塊以建立 VPC	192.168.0.0/16	如果您使用現有的 VPC，則可以將此參數保留空白
為子網路 A 提供有效的 CIDR 區塊，讓解決方案建立 VPC	192.168.0.0/20	AWS Fargate VPC 子網路 A 的 CIDR 區塊
為子網路 B 提供有效的 CIDR 區塊，讓解決方案建立 VPC	192.168.16.0/20	AWS Fargate VPC 子網路 B 的 CIDR 區塊
提供 CIDR 區塊以允許 Fargate 任務的傳出流量	0.0.0.0/0	限制 Amazon ECS 容器傳出存取的 CIDR 區塊。
自動更新容器映像	No	自動使用最新且安全的映像，直到下一個次要版本為止。選取 No會提取最初發行的映像，而不會進行任何安全性更新。

參數	預設	說明
部署選用 MCP 伺服器	No	部署選用的遠端 MCP 伺服器，使用 AgentCore Gateway 將 AI 應用程式連線至 AWS 上的分散式負載測試。

6. 選擇下一步。
7. 在 Configure stack options (設定堆疊選項) 頁面，選擇 Next (下一步)。
8. 在檢視 頁面上，檢視和確認的設定。勾選確認範本將建立 AWS Identity and Access Management (IAM) 資源的方塊。
9. 選擇 Create stack (建立堆疊) 以部署堆疊。

您可以在狀態欄的 AWS CloudFormation 主控台中檢視堆疊的狀態。您應該會在大約 15 分鐘內收到 CREATE_COMPLETE 狀態。

> **Note**
>
> 除了主要 AWS Lambda 函數之外，此解決方案還包含自訂資源 Lambda 函數，僅在初始組態期間或資源更新或刪除時執行。

執行此解決方案時，自訂資源 Lambda 函數處於非作用中狀態。不過，請勿刪除此函數，因為需要管理相關聯的資源。

多區域部署

部署時間：每個區域大約 5 分鐘

您可以跨多個區域執行測試。當您部署分散式負載測試解決方案時，它會建立區域 CloudFormation 範本，並將其存放在案例 S3 儲存貯體中。

> **Note**
>
> 案例儲存貯體名稱包含您的堆疊名稱和關鍵字「案例」。您可以透過導覽至 S3 主控台並在名稱中搜尋包含「案例」的儲存貯體來找到它。

若要執行多區域部署，您必須部署區域 CloudFormation 範本，該範本存放在您要執行測試的區域中的 Amazon S3 案例儲存貯體中。您可以執行下列動作來安裝區域範本：

1. 在解決方案的 Web 主控台中，導覽至左側選單中的儀表板。
2. 使用剪貼簿圖示來複製 Amazon S3 中的 CloudFormation 範本連結。
3. 登入 AWS CloudFormation 主控台，然後選取正確的區域。
4. 在建立堆疊頁面上，驗證 Amazon S3 URL 文字方塊中顯示的範本 URL 是否正確，然後選擇下一步。
5. 在指定堆疊詳細資訊頁面上，為您的解決方案堆疊指派名稱。
6. 在參數下，檢閱範本的參數，並視需要修改。此解決方案使用下列預設值。

參數	預設	說明
現有的 VPC ID	<選用輸入>	如果您有要使用且已建立的 VPC，請在部署堆疊的相同區域中輸入現有 VPC 的 ID。例如，vpc-1a2b3c4d5e6f。
第一個現有子網路	<選用輸入>	現有 VPC 中第一個子網路的 ID。此子網路需要網際網路的路由，才能提取容器映像以執行測試。例如，netnet-7h8i9j0k。
第二個現有子網路	<選用輸入>	現有 VPC 內第二個子網路的 ID。此子網路需要網際網路的路由，才能提取容器映像以執行測試。例如，netnet-1x2y3z。
為建立 VPC 的解決方案提供有效的 CIDR 區塊	192.168.0.0/16	如果您未提供現有 VPC 的值，則解決方案建立的 Amazon VPC 的 CIDR 區塊會包含 AWS Fargate 的 IP 地址。

參數	預設	說明
提供 CIDR 區塊以允許 Fargate 任務的傳出流量	0.0.0.0/0	限制 Amazon ECS 容器傳出存取的 CIDR 區塊。

7. 選擇下一步。
8. 在 Configure stack options (設定堆疊選項) 頁面，選擇 Next (下一步)。
9. 在檢視 頁面上，檢視和確認的設定。請務必勾選確認範本將建立 AWS Identity and Access Management (IAM) 資源的核取方塊。
10. 選擇 Create stack (建立堆疊) 以部署堆疊。

    您可以在狀態欄的 AWS CloudFormation 主控台中檢視堆疊的狀態。您應該會在大約五分鐘內收到 CREATE_COMPLETE 狀態。

當成功部署區域時，它們會出現在 Web 主控台中。當您建立測試時，所有可用的區域都會列在儀表板和案例建立中。您可以在案例建立的流量形狀步驟中，將區域新增至測試。

解決方案會為案例資料表中的每個部署區域建立 DynamoDB 項目，其中包含該區域中測試資源的必要資訊。您可以在 Web 主控台中依區域排序測試結果。若要檢視多區域測試中所有區域的彙總結果，請使用 Amazon CloudWatch 指標。測試完成後，您可以在測試結果中找到圖形的原始碼。

> **Note**
>
> 您可以在不使用 Web 主控台的情況下啟動區域堆疊。在 Amazon S3 案例儲存貯體中取得區域範本的連結，並在所需區域中啟動區域堆疊時將其做為來源提供。或者，您可以下載範本並將其上傳為您想要的區域來源。

更新解決方案

更新解決方案會將最新的功能、安全修補程式和錯誤修正套用至您的部署。如果您先前已部署解決方案，請依照此程序將 CloudFormation 堆疊更新至最新版本。

> **Important**
>
> 更新之前，請確定目前未執行任何負載測試。更新程序可能會暫時中斷解決方案的可用性。

1. 登入 CloudFormation 主控台，選取您現有的 CloudFormation 堆疊，然後選取更新堆疊。
2. 選取直接更新。
3. 選取取代現有範本。
4. 在指定範本下：
   a. 選取 Amazon S3 URL。
   b. 複製最新範本的連結。
   c. 將連結貼到 Amazon S3 URL 方塊中。
   d. 驗證Amazon Amazon S3 URL 文字方塊中顯示的範本 URL 是否正確。
   e. 選擇下一步。
   f. 再次選擇 Next (下一步)。
5. 在參數下，檢閱範本的參數並視需要修改。如需參數的詳細資訊，請參閱啟動堆疊。
6. 選擇下一步。
7. 在 Configure stack options (設定堆疊選項) 頁面，選擇 Next (下一步)。
8. 在檢視 頁面上，檢視和確認的設定。
9. 選取確認範本可能會建立 IAM 資源的方塊。
10. 選擇檢視變更集並驗證變更。
11. 選擇更新堆疊以部署堆疊。

您可以在狀態欄的 AWS CloudFormation 主控台中檢視堆疊的狀態。您應該會在大約 15 分鐘內收到 `UPDATE_COMPLETE` 狀態。

> **Note**
>
> 如果您在堆疊升級後從瀏覽器登入時遇到 Amazon Cognito 身分驗證問題，請重新整理您的瀏覽器 (Windows/Linux 上的 Ctrl+Shift+R 或 Mac 上的 Cmd+Shift+R)，以清除快取的資料，然後再試一次。

針對 v3.3.0 之前版本的更新進行故障診斷

> **Note**
>
> 本節僅適用於 v3.3.0 之前的版本更新。如果您是從 v3.3.0 或更新版本更新，請遵循上述標準更新程序。

1. 下載 distributed-load-testing-on-aws.template。
2. 開啟範本並導覽至條件：並尋找 DLTCommonResourcesAppRegistryCondition
3. 您應該會看到類似下列的內容：

```
Conditions:
DLTCommonResourcesAppRegistryConditionCCEF54F8:
Fn::Equals:
- "true"
- "true"
```

4. 將第二個 true 值變更為 false：

```
Conditions:
DLTCommonResourcesAppRegistryConditionCCEF54F8:
Fn::Equals:
- "true"
- "false"
```

5. 使用自訂範本來更新您的堆疊。
6. 此更新會從堆疊中移除應用程式登錄檔相關的資源，讓更新順利完成。
7. 使用最新的範本 URL 執行另一個堆疊更新，將應用程式登錄應用程式資源新增至您的堆疊。

更新區域堆疊

如果您已在多個區域中部署解決方案，則必須分別更新每個區域堆疊。在您部署測試基礎設施的區域中，請遵循每個區域 CloudFormation 堆疊的標準更新程序。

AWS Systems Manager Application Manager

更新解決方案後，AWS Systems Manager Application Manager 會提供解決方案及其資源的應用程式層級檢視。您可以使用 Application Manager 來：

- 監控資源、跨堆疊和 AWS 帳戶部署資源的成本，以及來自中央位置的日誌。
- 在應用程式內容中檢視解決方案資源的操作資料，例如部署狀態、CloudWatch 警示、資源組態和操作問題。

疑難排解

已知問題解決提供減輕已知錯誤的指示。如果這些指示無法解決您的問題，請聯絡 AWS Support 提供為此解決方案開啟 AWS Support 案例的說明。

已知問題解決方案

問題：您正在使用現有的 VPC，而且您的測試失敗，狀態為失敗，導致下列錯誤訊息：

```
Test might have failed to run.
```

- 解決方法：

確保子網路存在於指定的 VPC 中，並且具有具有網際網路閘道或 NAT 閘道的網際網路路由。AWS Fargate 需要存取才能從公有儲存庫提取容器映像，以成功執行測試。

問題：測試執行時間過長或無限期停滯

- 解決方法：

取消測試並檢查 AWS Fargate，以確保所有任務都已停止。如果尚未停止，請手動停止所有 Fargate 任務。檢查您帳戶的隨需 Fargate 任務限制，以確保您可以啟動所需的任務數量。您也可以檢查 Lambda 任務執行器函數的 CloudWatch 日誌，以深入了解啟動 Fargate 任務時的失敗。檢查 CloudWatch ECS 日誌，了解正在執行的 Fargate 容器中發生的情況的詳細資訊。

問題：測試正在開始，但無法完成或 ECS 任務的狀態不明

- 解決方法：

如果您選擇在部署解決方案的帳戶中提供現有 VPC 的選項，請確定 ECS 任務使用的 VPC 有足夠的可用 IP 地址，以開始測試輸入中提供的任務數量。ECS 任務定義使用需要網際網路閘道或網際網路路由的 ECR 映像，以便 ECS 服務可以透過從 aws-solutions/distributed-load-testing-on-aws-load-tester 下載解決方案 ECR 映像來佈建任務。如果您無法提供網際網路的路由，因為 VPC 中的所有子網路都是私有的，您可以使用 ECR 提取快取在帳戶中託管 ECR 映像。使用新的 ECR 映像 URI 更新任務定義，並建立新的修訂。任務定義更新後，DynamoDB 資料表中的解決方案組態需要更新，才能使用新的修訂。您可以在金鑰 ScenariosTable 下的 CloudFormation 堆疊輸出索引標籤中找到

DynamoDB 資料表名稱。使用金鑰 testId 和值區域-【SOLUTION-DEPLOYED-REGION】 更新項目的屬性 taskDefinition。

問題：測試需要使用私有或無法透過網際網路閘道使用的端點

- 解決方法：

測試無法透過網際網路閘道存取的私有 API 端點時，請考慮下列方法：

1. 網路組態：確保 ECS 任務使用的子網路路由表已更新為要測試之私有端點的 IP 地址範圍的路由。這可讓測試流量到達 VPC 中的私有端點。
2. DNS 解析：對於自訂網域，請在 VPC 中設定 DNS 設定，以解析私有端點的網域名稱。如需詳細說明，請參閱 VPC DNS 文件。
3. VPC 端點：如果測試 AWS 服務，請考慮使用 VPC 端點 (AWS PrivateLink) 來建立私有連線。例如，若要測試私有 API Gateway，您可以為 API Gateway 建立 VPC 端點。請參閱私有 API Gateway 文件。
4. VPC 對等互連：如果私有端點位於不同的 VPC 中，請在部署解決方案的 VPC 與包含私有端點的 VPC 之間建立 VPC 對等互連。在兩個 VPCs中設定適當的路由表。請參閱 VPC 對等互連文件。
5. 傳輸閘道：對於涉及多個 VPCs的更複雜聯網案例，請考慮使用 AWS Transit Gateway 在解決方案的 VPC 和包含私有端點的 VPC 之間路由流量。請參閱 Transit Gateway 文件。
6. 安全群組：確保與 ECS 任務相關聯的安全群組允許對私有端點的傳出流量，而私有端點的安全群組允許來自 ECS 任務的傳入流量。

若要測試內部 Application Load Balancer 或 EC2 執行個體，請確保 VPC CIDR 範圍不重疊，且已在路由表中設定必要的路由。

問題：測試正在完成，但 UI 上無法使用結果

- 解決方法：

如果測試已完成，但在 UI 中無法使用結果，則結果檔案應該仍可從執行測試的 ECS 任務，在 S3 儲存貯體中使用。這是 解決方案中的已知限制。在目前的架構中，解決方案會使用結果剖析 Lambda 函數來摘要來自多個 ECS 任務的結果，然後儲存為 DynamoDB 資料表中的項目。DynamoDB 資料表的項目大小上限為 400 KB。根據測試指令碼的複雜性、並行和使用的任務數量，達到此限制。錯誤並不表示測試失敗；它表示摘要結果並將其存放在 DynamoDB 資料表中用於 CRUD 操作的程序已失敗。結果仍可在測試案例的 S3 儲存貯體中使用。

聯絡 AWS Support

如果您有 AWS 開發人員支援、AWS Business Support 或 AWS Enterprise Support，您可以使用 支援中心來取得此解決方案的專家協助。以下章節將提供說明。

建立案例

1. 登入支援中心。
2. 選擇建立案例。

如何提供協助？

1. 選擇技術
2. 針對服務，選取解決方案。
3. 針對類別，選取 AWS 上的分散式負載測試。
4. 針對嚴重性，選取最符合您使用案例的選項。
5. 當您輸入服務、類別和嚴重性時，界面會填入常見故障診斷問題的連結。如果您無法使用這些連結解決您的問題，請選擇下一步：其他資訊。

其他資訊

1. 針對主旨，輸入摘要您的問題的文字。
2. 針對描述，請詳細說明問題。
3. 選擇連接檔案。
4. 連接 AWS Support 處理請求所需的資訊。

協助我們更快解決您的案例

1. 輸入請求的資訊。
2. 選擇下一步驟：立即解決或聯絡我們。

立即解決或聯絡我們

1. 檢閱立即解決解決方案。

2. 如果您無法解決這些解決方案的問題，請選擇聯絡我們，輸入請求的資訊，然後選擇提交。

解除安裝解決方案

您可以從 AWS 管理主控台或使用 AWS 命令列界面解除安裝 AWS 解決方案上的分散式負載測試。您必須手動刪除此解決方案建立的主控台、案例和記錄 Amazon Simple Storage Service (Amazon S3) 儲存貯體。如果您有要保留的資料，AWS 解決方案實作不會自動將其刪除。

> **Note**
>
> 如果您已部署區域堆疊，您必須先刪除這些區域中的堆疊，才能刪除主要堆疊。

使用 AWS 管理主控台

1. 登入 AWS CloudFormation 主控台。
2. 在堆疊頁面上，選取此解決方案的安裝堆疊。
3. 選擇 刪除。

使用 AWS 命令列界面

判斷您的環境中是否可使用 AWS Command Line Interface (AWS CLI)。如需安裝說明，請參閱《AWS CLI 使用者指南》中的什麼是 AWS 命令列界面。 確認 AWS CLI 可用後，請執行下列命令。

```
$ aws cloudformation delete-stack --stack-name <installation-stack-name>
```

刪除 Amazon S3 儲存貯體

如果您決定刪除 AWS CloudFormation 堆疊以防止意外資料遺失，此解決方案會設定為保留解決方案建立的 Amazon S3 儲存貯體 （用於在選擇加入區域中部署）。解除安裝解決方案之後，如果您不需要保留資料，您可以手動刪除此 S3 儲存貯體。請依照下列步驟刪除 Amazon S3 儲存貯體。

1. 登入 Amazon S3 主控台。
2. 從左側導覽窗格中選擇儲存貯體。
3. 在依名稱尋找儲存貯體欄位中，輸入此解決方案堆疊的名稱。
4. 選取其中一個解決方案的 S3 儲存貯體，然後選擇空白。

5. 在驗證欄位中輸入永久刪除，然後選擇空白。
6. 選取您剛清空的 S3 儲存貯體，然後選擇刪除。
7. 在驗證欄位中輸入 S3 儲存貯體名稱，然後選擇刪除儲存貯體。

重複步驟 4 到 7，直到您刪除所有 S3 儲存貯體為止。

若要使用 AWS CLI 刪除 S3 儲存貯體，請執行下列命令：

```
$ aws s3 rb s3://<bucket-name> --force
```

使用 解決方案

本節提供在 AWS 上使用分散式負載測試解決方案的完整指南，從建立第一個測試案例到分析詳細結果。工作流程包括建立測試案例、執行測試，以及探索測試結果。

建立測試案例

建立測試案例包含四個主要步驟：設定一般設定、定義案例、塑造流量模式，以及檢閱您的組態。

步驟 1：一般設定

為您的負載測試設定基本參數，包括測試名稱、描述和一般組態選項。

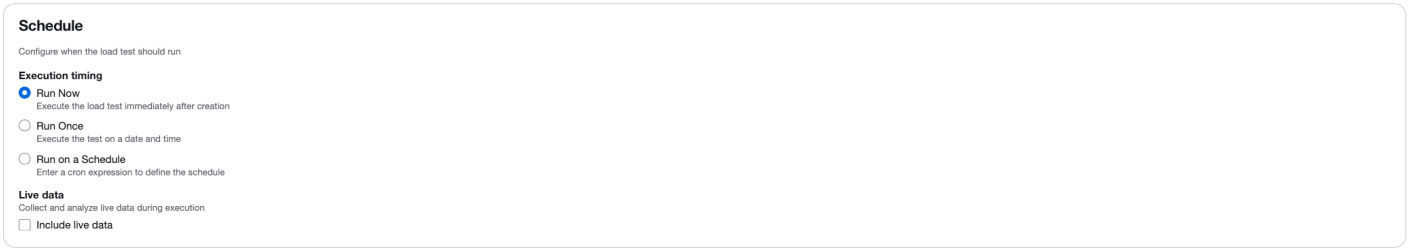
測試識別

- 測試名稱 （必要） - 測試案例的描述性名稱
- 測試描述 （必要） - 有關測試用途和組態的其他詳細資訊
- 標籤 （選用） - 新增最多 5 個標籤來分類和整理您的測試案例

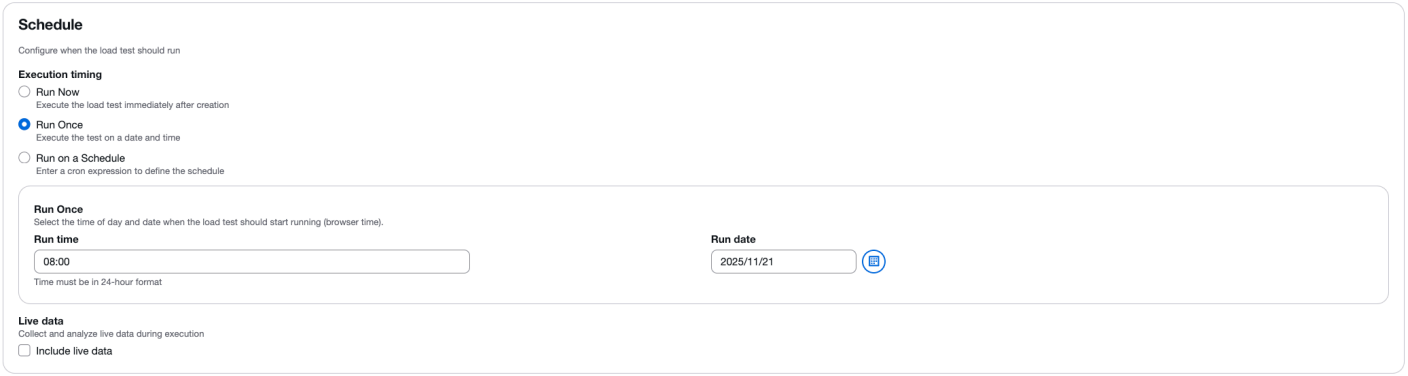
排程選項

設定測試應執行的時間：

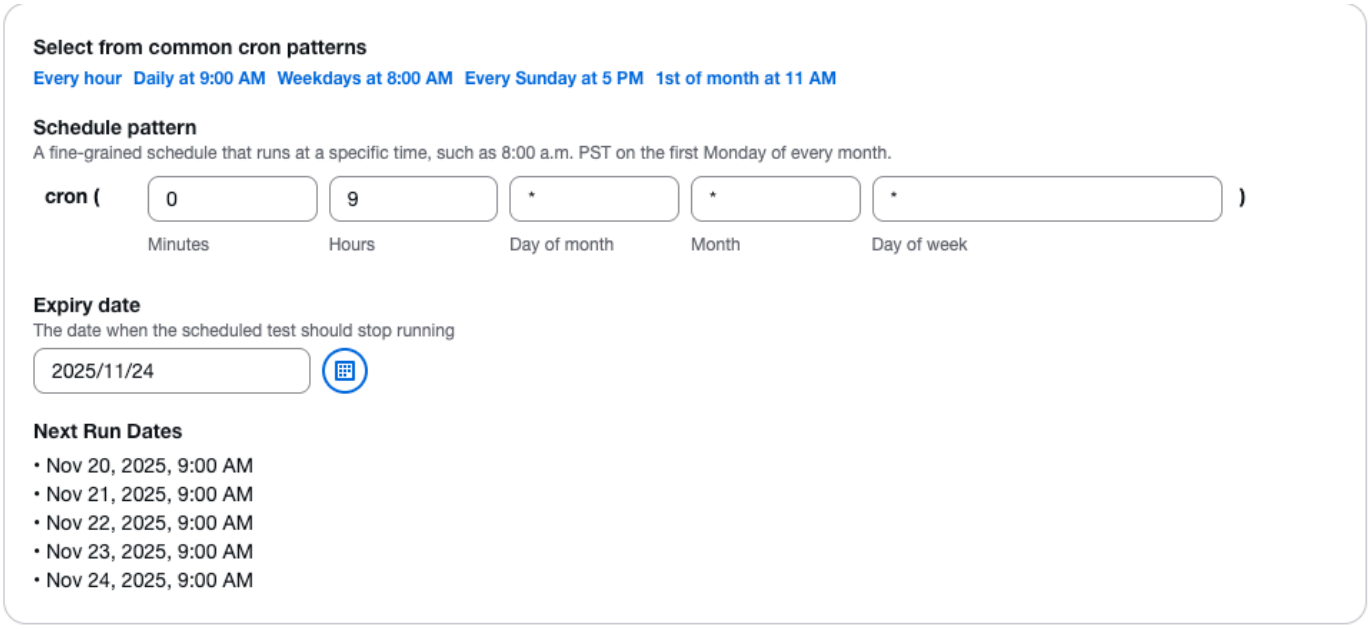
- 立即執行 - 在建立後立即執行測試。



- 執行一次 - 排定在特定日期和時間執行測試。



- 在排程上執行 - 使用以 Cron 為基礎的排程，定期自動執行測試。您可以從常見模式（每小時、每日、每週）中選取，或定義自訂 Cron 表達式。



排程工作流程

當您排程測試時，會發生下列工作流程：

- 排程參數會透過 Amazon API Gateway 傳送至解決方案的 API。
- API 會將參數傳遞至 Lambda 函數，該函數會建立排程在指定日期執行的 CloudWatch Events 規則。
- 對於一次性測試（執行一次），CloudWatch Events 規則會在指定的日期執行，Lambda `api-services` 函數會執行測試。
- 對於週期性測試（在排程上執行），CloudWatch Events 規則會在指定的日期啟用，而 `api-services` Lambda 函數會根據指定的頻率建立立即且重複執行的新規則。

即時資料

選取包含即時資料核取方塊，以在測試執行時檢視即時指標。啟用時，您可以監控：

- 平均回應時間。
- 虛擬使用者計數。
- 成功的請求計數。
- 失敗的請求計數。

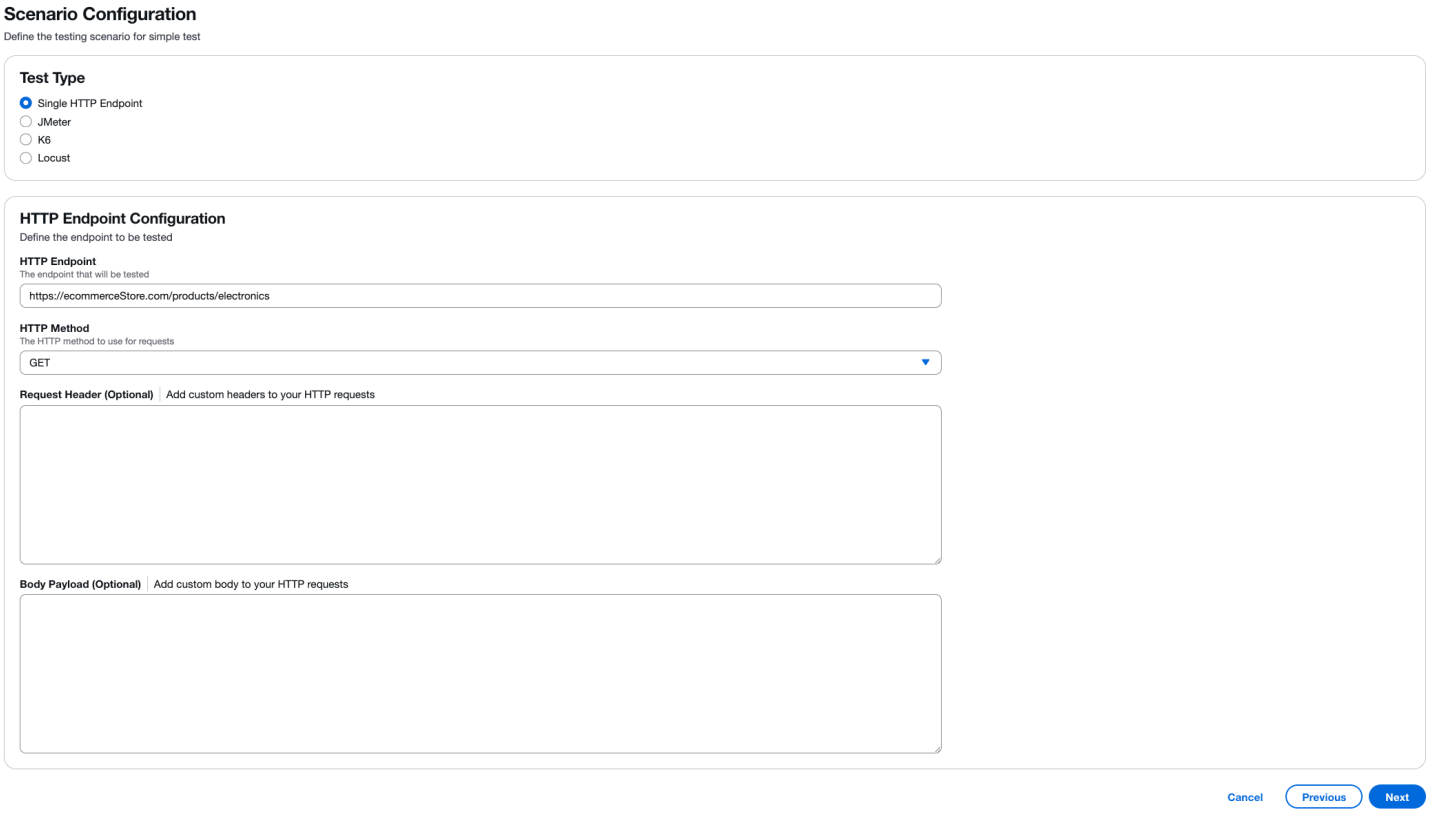
即時資料功能提供即時圖表，並以一秒的間隔彙總資料。如需詳細資訊，請參閱使用即時資料進行監控。

步驟 2：案例組態

定義特定測試案例，然後選取您偏好的測試架構。

測試類型選擇

選擇您要執行的負載測試類型：



- 單一 HTTP 端點 - 使用簡單的組態測試單一 API 端點或網頁。

- JMeter - 上傳 JMeter 測試指令碼 (.jmx 檔案或 .zip 封存檔）。
- K6 - 上傳 K6 測試指令碼 (.js 檔案或 .zip 封存檔）。
- Locust - 上傳 Locust 測試指令碼 (.py 檔案或 .zip 封存檔）。

HTTP 端點組態映像：：images/test-types.png【選取要執行的測試類型】 選取「單一 HTTP 端點」時，請設定這些設定：

HTTP 端點（必要）

輸入您要測試之端點的完整 URL。例如 `https://api.example.com/users` 。確保可從 AWS 基礎設施存取端點。

HTTP 方法（必要）

選取請求的 HTTP 方法。預設值為 GET。其他選項包括 POST、PUT、DELETE、HEAD、 PATCH和 OPTIONS。

請求標頭（選用）

將自訂 HTTP 標頭新增至您的請求。常見範例包括：

- `Content-Type: application/json`
- `Authorization: Bearer <token>`
- `User-Agent: LoadTest/1.0`

選擇新增標頭以包含多個標頭。

內文承載（選用）

新增 POST 或 PUT 請求的請求內文內容。支援 JSON、XML 或純文字格式。例如：`{"userId": 123, "action": "test"}`。

測試架構指令碼

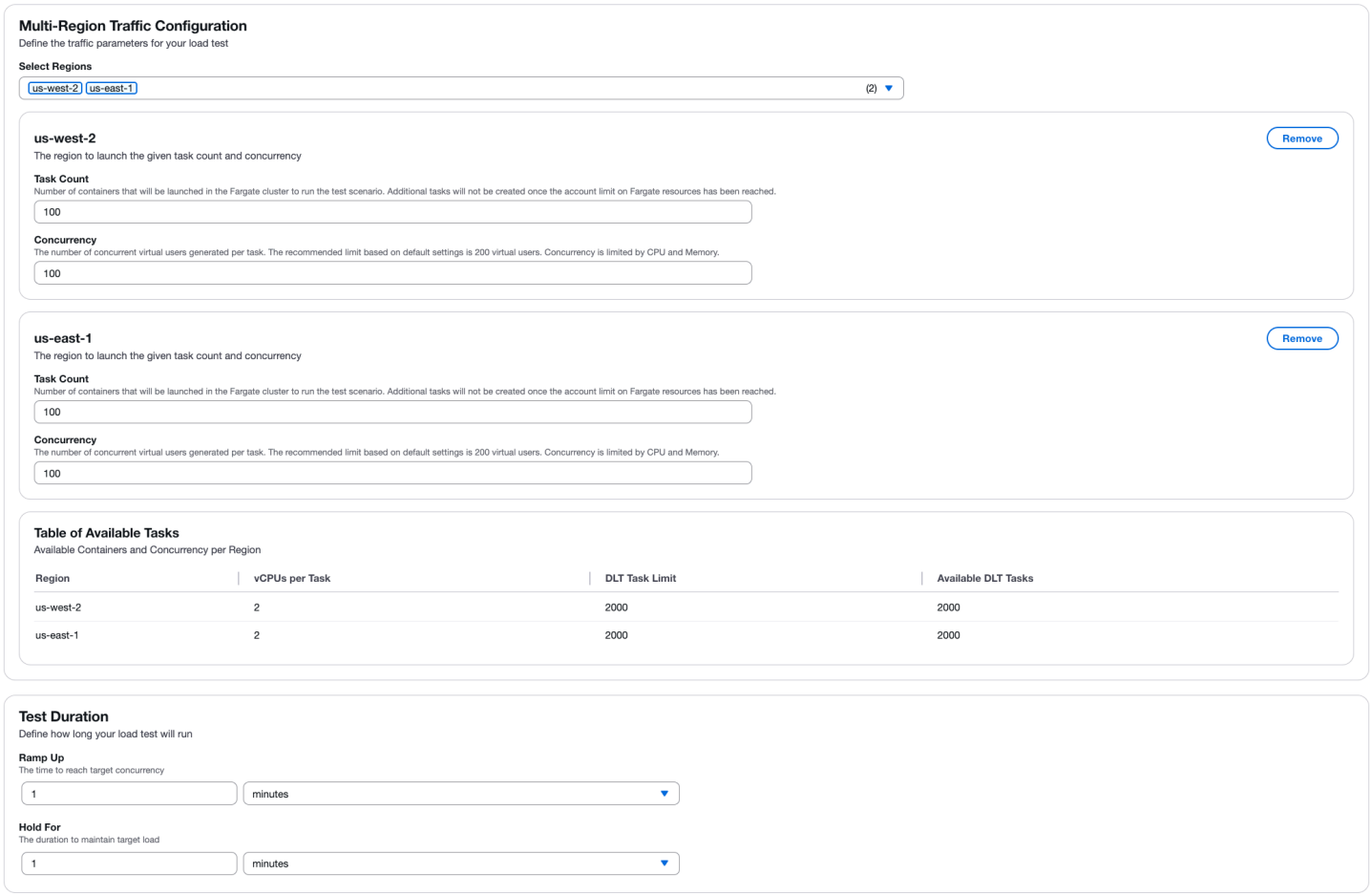
使用 JMeter、K6 或 Locust 時，請上傳您的測試指令碼檔案或包含測試指令碼和支援檔案的 .zip 封存檔。對於 JMeter，您可以在 .zip 封存檔的`/plugins`資料夾中包含自訂外掛程式。

> **Important**
>
> 雖然您的測試指令碼 (JMeter、K6 或 Locust) 可能會定義並行（虛擬使用者）、交易速率 (TPS)、漸進測試時間和其他負載參數，但解決方案會使用您在測試建立期間在流量形狀畫面

中指定的值來覆寫這些組態。流量形狀組態控制測試執行的任務計數、並行（每個任務的虛擬使用者）、漸進測試持續時間和保留持續時間。

步驟 3：流量形狀

設定測試期間流量的分佈方式，包括多區域支援。



多區域流量組態

選取一或多個 AWS 區域，以依地理分佈負載測試。針對每個選取的區域，設定：

任務計數

將在 Fargate 叢集中針對測試案例啟動的容器（任務）數量。一旦帳戶達到「遠的資源」限制，就不會建立其他任務。

並行數量

每個任務產生的並行虛擬使用者數量。建議的限制是根據每個任務 2 個 vCPUs的預設設定。並行受限於 CPU 和記憶體資源。

判斷使用者數量

容器可支援用於測試的使用者數量，可透過逐步增加使用者數量並在 Amazon CloudWatch 中監控效能來判斷。一旦觀察到 CPU 和記憶體效能接近其限制，您已達到容器在其預設組態 (2 個 vCPU 和 4 GB 的記憶體） 中可支援該測試的使用者數量上限。

校正程序

您可以使用下列範例，開始判斷測試的並行使用者限制：

1. 建立不超過 200 個使用者的測試。
2. 測試執行時，請使用 CloudWatch 主控台監控 CPU 和記憶體：
   a. 從左側導覽窗格的容器洞見下，選取效能監控。
   b. 在效能監控頁面的左側下拉式選單中，選取 ECS 叢集。
   c. 從右側下拉式選單中，選取您的 Amazon Elastic Container Service (Amazon ECS) 叢集。
3. 監控時，請監看 CPU 和記憶體。如果 CPU 未超過 75% 或記憶體未超過 85%（忽略一次性峰值），您可以對更多使用者執行另一個測試。

如果測試未超過資源限制，請重複步驟 1-3。或者，您可以增加容器資源，以允許更多並行使用者。不過，這會產生較高的成本。如需詳細資訊，請參閱 開發人員指南。

> **Note**
>
> 為了獲得準確的結果，在確定並行使用者限制時，一次只執行一個測試。所有測試都使用相同的叢集，CloudWatch 容器洞察會根據叢集彙總效能資料。這會導致這兩個測試同時向 CloudWatch Container Insights 報告，這會導致單一測試的資源使用率指標不準確。

如需校正每個引擎使用者的詳細資訊，請參閱 BlazeMeter 文件中的校正 Taurus 測試。

> **Note**
>
> 解決方案會顯示每個區域的可用容量資訊，協助您在可用限制內規劃測試組態。

可用任務的資料表

可用任務表會顯示每個所選區域的資源可用性：

- 區域 - AWS 區域名稱。
- 每個任務vCPUs - 分配給每個任務CPUs 數量（預設值：2)。
- DLT 任務限制 - 根據帳戶的 Fargate 限制可建立的任務數量上限（預設值：2000)。
- 可用的 DLT 任務 - 區域中目前可用的任務數量（預設值：2000)。

Table of Available Tasks

Available Containers and Concurrency per Region

Region	vCPUs per Task	DLT Task Limit	Available DLT Tasks
us-west-2	2	2000	2000
us-east-1	2	2000	2000

若要增加每個任務的可用任務或 vCPUs 數量，請參閱 開發人員指南。

測試持續時間

定義您的負載測試執行的時間長度：

漸進測試

達到目標並行的時間。在此期間，負載會從 0 逐漸增加到設定的並行層級。

保留

維持目標負載的持續時間。測試在此期間會繼續以完整並行狀態進行。

步驟 4：檢閱和建立

在建立測試案例之前，請先檢閱您的所有組態。驗證：

- 一般設定（名稱、描述、排程）。
- 案例組態（測試類型、端點或指令碼）。

- 流量形狀（任務、使用者、持續時間、區域）。

檢閱後，選擇建立以儲存您的測試案例。

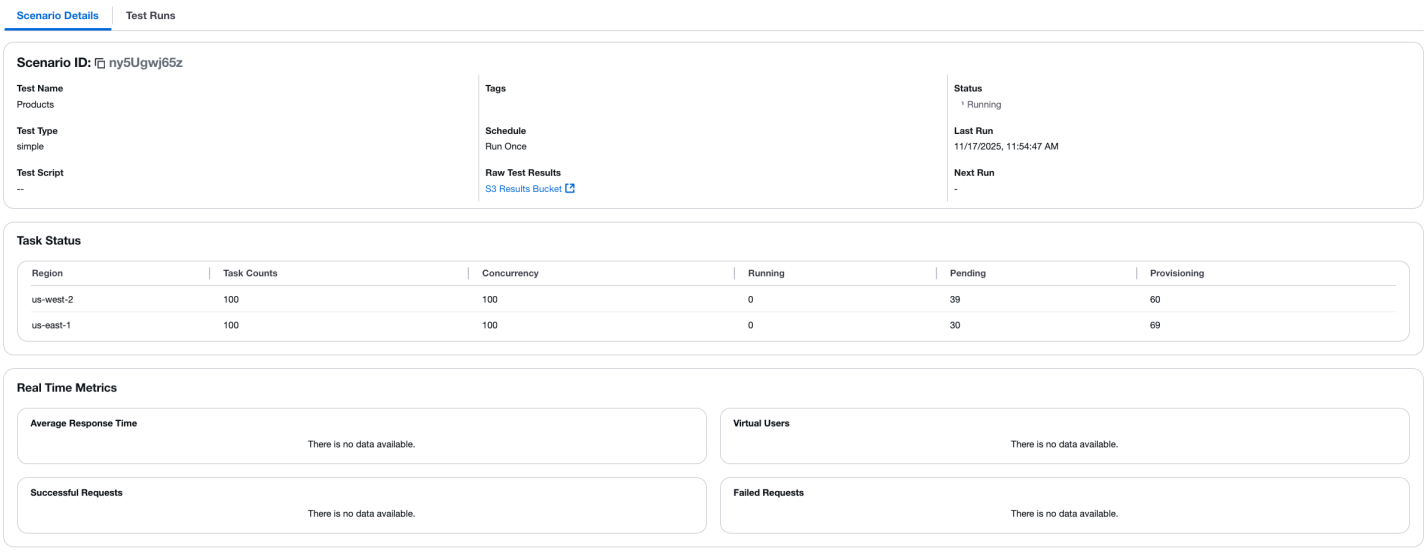
管理測試案例

建立測試案例後，您可以：

- 編輯 - 修改測試組態。常用案例包括：
  - 精簡流量形狀以達到所需的交易速率。
- 複製 - 複製現有的測試案例以建立變化。常用案例包括：
  - 更新端點或新增標頭/內文參數。
  - 新增或修改測試指令碼。
- 刪除 - 移除您不再需要的測試案例。

執行測試案例

建立測試案例之後，您可以立即執行它，或排定它在未來的特定時間執行。當您導覽至執行中的測試時，主控台會顯示案例詳細資訊索引標籤，其中包含即時任務狀態和指標。



案例詳細資訊檢視

案例詳細資訊索引標籤會顯示測試的重要資訊。每個區域的任務 staus 資料表即時資訊。

任務狀態資料表

任務狀態資料表顯示每個區域的即時資訊：

- 區域 - 執行任務的 AWS 區域
- 任務計數 - 針對區域設定的任務總數
- 並行 - 每個任務的虛擬使用者數量
- 執行中 - 目前正在執行測試的任務數量
- 待處理 - 等待啟動的任務數量
- 佈建 - 正在佈建的任務數量

測試執行工作流程

當測試開始時，會發生下列工作流程：

1. 任務佈建 - 解決方案會在指定的 AWS 區域中佈建容器（任務）。任務會出現在「佈建」欄中。
2. 任務啟動 - 解決方案會繼續佈建任務，直到達到每個區域的目標任務計數為止。任務會從 "Provisioning" 移至 "Pending" 到 "Running"。
3. 流量產生 - 在解決方案佈建區域中的所有任務之後，它們會開始將流量傳送到您的目標端點。
4. 測試執行 - 測試在設定的持續時間（漸進 + 保留時間）內執行。
5. 結果剖析 - 測試結束時，背景剖析任務會彙總和處理來自所有區域的結果。

測試執行狀態

測試執行可以有下列狀態：

- 已排程 - 測試排程於未來執行。
- 執行中 - 測試目前正在進行中。
- 已取消 - 使用者已取消進行中的測試執行。
- 發生錯誤 - 測試執行發生錯誤。
- 完成 - 測試執行已成功完成，且結果已準備就緒。

使用即時資料進行監控

如果您在建立測試案例時啟用即時資料，您可以在測試執行時檢視即時指標。即時指標區段會顯示四個圖形，隨著測試進行而持續更新，並以一秒的間隔彙總資料。

圖形描述

平均回應時間

顯示每個區域處理之請求的平均回應時間，以秒為單位。Y 軸顯示以秒為單位的回應時間，而 X 軸顯示一天中的時間。每個區域在圖例中以不同的顏色表示。

虛擬使用者

顯示在每個區域中主動產生負載的並行虛擬使用者數量。圖表顯示虛擬使用者在測試期間如何加速並維持目標並行層級。

成功的請求

顯示每個區域隨時間成功請求的累積計數。圖表顯示處理成功請求的速率。

失敗的請求

顯示每個區域隨時間失敗請求的累積計數。低計數或零計數表示測試執行狀態良好。

多區域視覺化

跨多個區域執行測試時，每個圖形會同時顯示所有區域的資料。每個圖形底部的圖例可識別代表每個區域的顏色（例如 us-west-2 和 us-east-1)。

技術實作

Fargate 任務的 CloudWatch 日誌群組包含擷取測試結果的訂閱篩選條件。偵測到模式時，Lambda 函數會建構資料並將其發佈至 AWS IoT Core 主題。Web 主控台會訂閱此主題，並即時顯示指標。

> **Note**
>
> 即時資料是暫時性的，只有在測試執行時才能使用。Web 主控台最多會保留 5，000 個資料點，之後會將最舊的資料取代為最新的資料點。如果頁面重新整理，則圖表將為空白，並從下一個可用的資料點開始。測試完成後，解決方案會將結果資料存放在 DynamoDB 和 Amazon S3 中。如果還沒有可用的資料，圖形會顯示「沒有可用的資料」。

取消測試

您可以從 Web 主控台取消執行中的測試。當您取消測試時，會發生下列工作流程：

1. 取消請求會傳送至 `microservices` API
2. `microservices` API 會呼叫 `task-canceler` Lambda 函數，以停止所有目前啟動的任務
3. 如果 `task-runner` Lambda 函數在初始取消呼叫後繼續執行，任務可能會短暫地繼續啟動
4. `task-runner` Lambda 函數完成後，AWS Step Functions 會繼續`Cancel Test`執行步驟，再次執行 `task-canceler` Lambda 函數以停止任何剩餘的任務

> **Note**
>
> 當解決方案終止所有容器時，取消的測試需要一些時間來完成關閉程序。清除所有資源後，測試狀態會變更為「已取消」。

探索測試結果

剖析任務完成後，即可分析測試結果。解決方案提供全方位的指標和工具，協助您了解應用程式在負載下的效能。

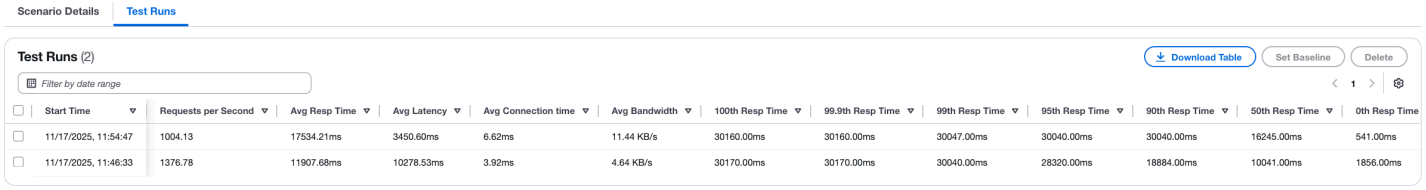
測試執行摘要指標

當測試完成時，解決方案會產生包含下列指標的摘要：

- 平均回應時間 - 測試產生之所有請求的平均回應時間，以秒為單位。

- 平均延遲 - 測試產生之所有請求的平均延遲，以秒為單位。
- 平均連線時間 - 所有請求連線到主機所需的平均時間，以秒為單位。
- 平均頻寬 - 測試產生之所有請求的平均頻寬。
- 總數 - 請求總數。
- 成功計數 - 成功請求的總數。
- 錯誤計數 - 錯誤總數。
- 每秒請求數 - 測試產生之所有請求的平均每秒請求數。
- 百分位數 - 回應時間百分位數，包括 p50 （中位數）、p90、p95 和 p99，顯示所有請求的回應時間分佈。

測試執行資料表



測試執行資料表會顯示案例的所有歷史測試執行。您可以：

- 檢視每個測試執行的摘要指標。
- 設定基準測試執行以進行效能比較。
- 將資料表下載為 CSV 檔案。
- 切換資料欄以自訂檢視。
- 選取測試執行以檢視詳細結果。

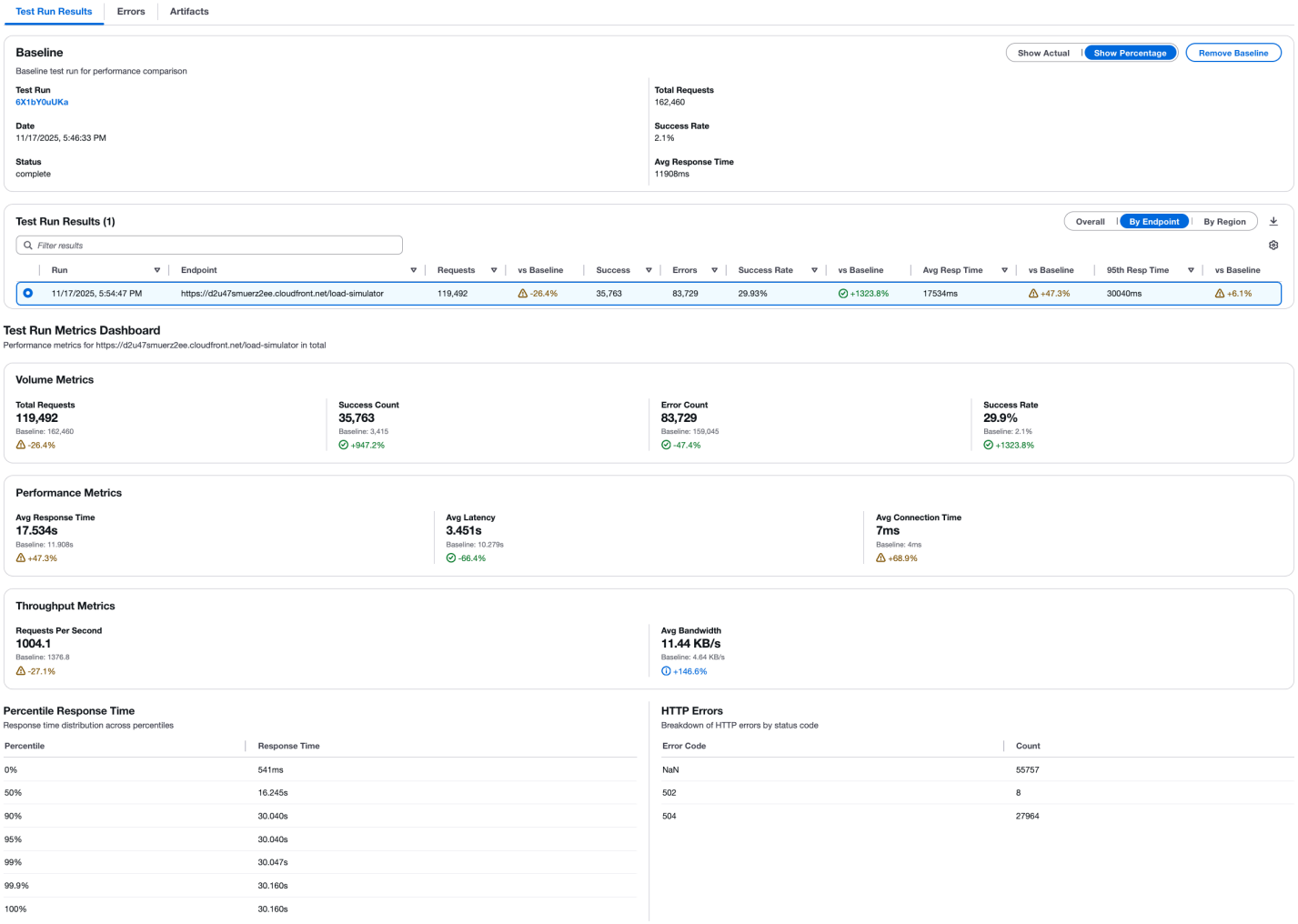
基準比較

您可以指定測試執行做為基準，以比較未來的測試執行。設定基準時：

- 測試執行資料表顯示與每個指標的基準相比的百分比差異 (+/-%)。
- 基準指標可協助您快速識別效能改善或迴歸。
- 您可以隨時變更或清除基準。

詳細測試結果

選取測試執行會開啟具有三個標籤的詳細結果檢視：測試執行結果、錯誤和成品。



基準資訊

如果已設定基準測試執行，則會顯示在頁面頂端。您可以選擇顯示實際、顯示百分比或移除基準，以控制基準比較的顯示方式。

測試執行結果資料表

結果資料表提供具有下列功能的詳細指標：

維度檢視

使用維度按鈕在三個檢視之間切換：

- **整體** - 所有端點和區域的彙總結果

- 依端點 - 依個別端點細分的結果
- 依區域 - 依 AWS 區域細分的結果

動作按鈕

- 顯示實際 - 顯示實際指標值
- 顯示百分比 - 顯示與基準的差異百分比
- 移除基準 - 清除基準比較

資料匯出和自訂

- 將結果資料表下載為 CSV 檔案
- 切換資料欄以自訂檢視
- 篩選和排序資料以專注於特定指標
- 篩選和排序資料以專注於特定指標。

錯誤索引標籤

錯誤索引標籤提供詳細的錯誤分析：

- 依類型檢視錯誤計數。
- 請參閱依整體測試或端點彙總的錯誤。
- 識別失敗請求中的模式。
- 針對特定端點或區域的問題進行故障診斷。

成品索引標籤

成品索引標籤可讓您存取測試執行期間產生的所有檔案：

- 檢視個別成品（日誌、結果檔案）。
- 下載特定成品以進行離線分析。
- 將所有測試執行成品下載為單一封存。

S3 結果結構

在 4.0 版中，S3 結果結構已變更以改善組織：

- 新結構 - `scenario-id/test-run-id/results-files`。
- 舊版結構 - 4.0 版之前執行的測試會在案例 ID 層級顯示所有結果檔案。

> **Note**
>
> 測試結果會顯示在 主控台中。您也可以直接在 `Results` 資料夾下的 Amazon S3 儲存貯體中存取原始測試結果。如需 Taurus 測試結果的詳細資訊，請參閱《Taurus 使用者手冊》中的產生測試報告。

MCP 伺服器整合

如果您在解決方案部署期間部署了選用的 MCP 伺服器元件，您可以將分散式負載測試解決方案與支援模型內容通訊協定的 AI 開發工具整合。MCP 伺服器提供透過 AI 助理擷取、管理和分析負載測試的程式設計存取權。

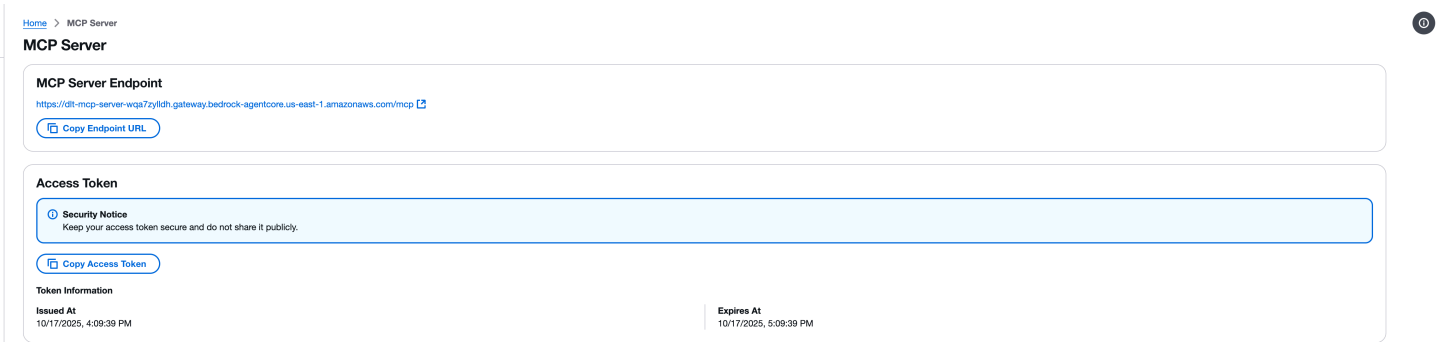
客戶可以使用自己選擇的用戶端 (Amazon Q、Claude 等） 連線到 DLT MCP 伺服器，每個用戶端的組態指示略有不同。本節提供 MCP Inspector、Amazon Q CLI、Cline 和 Amazon Q Suite 的設定指示。

步驟 1：取得 MCP 端點和存取權杖

設定任何 MCP 用戶端之前，您需要從 DLT Web 主控台擷取 MCP 伺服器端點和存取權杖。

1. 導覽至分散式負載測試 Web 主控台中的 MCP 伺服器頁面。
2. 找到 MCP 伺服器端點區段。
3. 使用複製端點 URL 按鈕複製端點 URL。端點 URL 遵循以下格式： `https://[api-id].execute-api.[region].amazonaws.com/[stage]/gateway/backend-agent/sse/mcp`
4. 找到存取字符區段。
5. 使用複製存取權杖按鈕複製存取權杖。

> **Important**
>
> 保護您的存取權杖，不要公開共用。字符透過 MCP 介面提供對分散式負載測試解決方案的唯讀存取權。



步驟 2：使用 MCP Inspector 進行測試

模型內容通訊協定提供 MCP Inspector，這是一種工具，可直接連線至 MCP 伺服器並叫用工具。這可提供方便的 UI 和範例網路請求，以便在設定 AI 用戶端之前測試 MCP 伺服器連線。

> **Note**
>
> MCP Inspector 需要 0.17 版或更新版本。所有請求也可以直接使用 JSON RPC 提出，但 MCP Inspector 提供更易於使用的界面。

安裝和啟動 MCP Inspector

1. 如有必要，請安裝 npm。
2. 執行下列命令來啟動 MCP Inspector：

```
npx @modelcontextprotocol/inspector
```

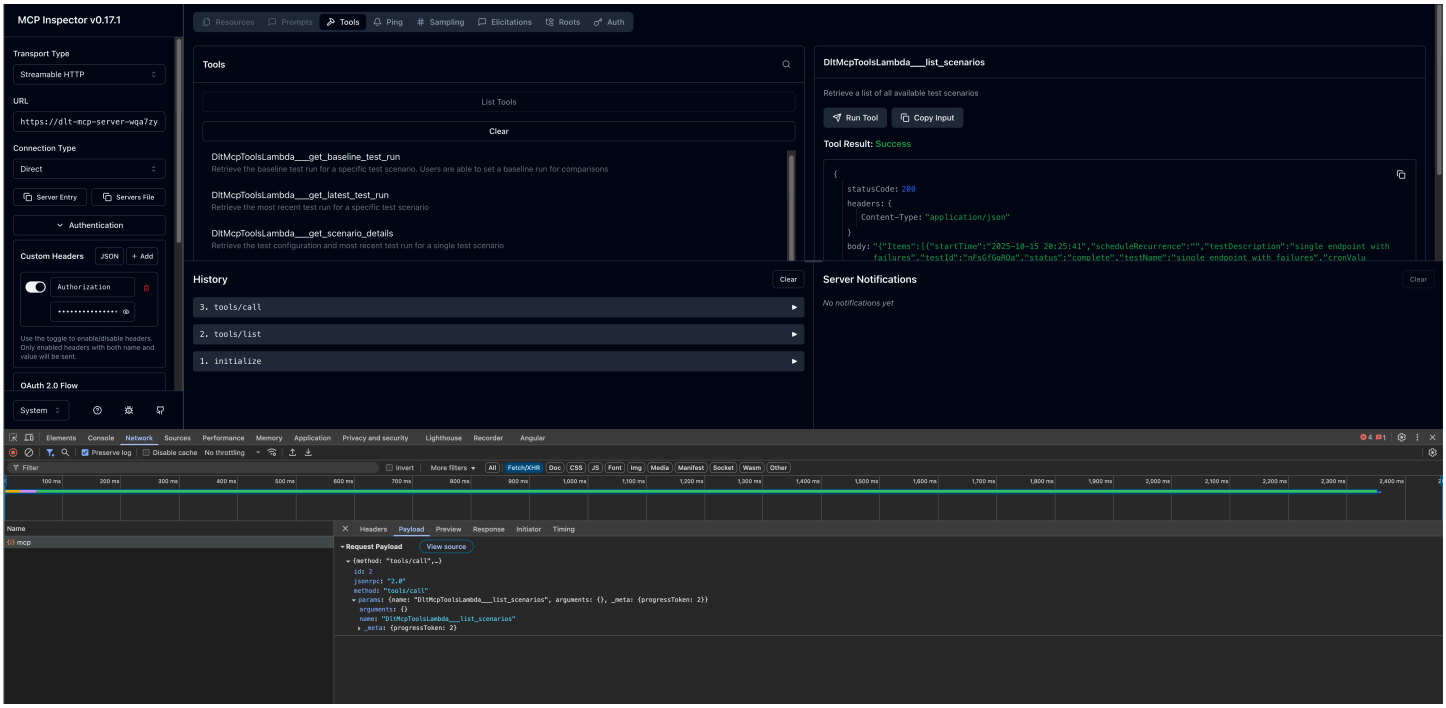
設定連線

1. 在 MCP Inspector 界面中，輸入您的 MCP 伺服器端點 URL。
2. 使用存取權杖新增授權標頭。
3. 按一下連線以建立連線。

叫用工具

連線後，您可以測試可用的 MCP 工具：

1. 瀏覽左側面板中可用工具的清單。
2. 選取工具（例如，`list_scenarios`)。
3. 提供任何必要的參數。
4. 按一下叫用以執行工具並檢視回應。



步驟 3：設定 AI 開發用戶端

使用 MCP Inspector 驗證 MCP 伺服器連線後，您可以設定偏好的 AI 開發用戶端。

Amazon Q CLI

Amazon Q CLI 透過 MCP 伺服器整合，提供 AI 輔助開發的命令列存取。

組態步驟

1. 編輯mcp.json組態檔案。如需組態檔案位置的詳細資訊，請參閱《Amazon Q 開發人員使用者指南》中的設定遠端 MCP 伺服器。
2. 新增您的 DLT MCP 伺服器組態：

```
{
  "mcpServers": {
    "dlt-mcp": {
      "type": "http",
      "url": "https://[api-id].execute-api.[region].amazonaws.com/[stage]/gateway/
backend-agent/sse/mcp",
      "headers": {
        "Authorization": "your_access_token_here"
      }
    }
```

```
  }
}
```

驗證組態

1. 在終端機中，輸入 q以啟動 Amazon Q CLI。
2. 輸入 /mcp 查看所有可用的 MCP 伺服器。
3. 輸入 /tools 查看 dlt-mcp和其他設定的 MCP 伺服器提供的可用工具。
4. 確認 dlt-mcp已成功初始化。

曲線

Cline 是支援 MCP 伺服器整合的 AI 編碼助理。

組態步驟

1. 在 Cline 中，導覽至管理 MCP 伺服器 > 設定 > 設定 MCP 伺服器。
2. 更新 cline_mcp_settings.json 檔案：

```
{
  "mcpServers": {
    "dlt-mcp": {
      "type": "streamableHttp",
      "url": "https://[api-id].execute-api.[region].amazonaws.com/[stage]/gateway/
backend-agent/sse/mcp",
      "headers": {
        "Authorization": "your_access_token_here"
      }
    }
  }
}
```

3. 儲存組態檔案。
4. 重新啟動 Cline 以套用變更。

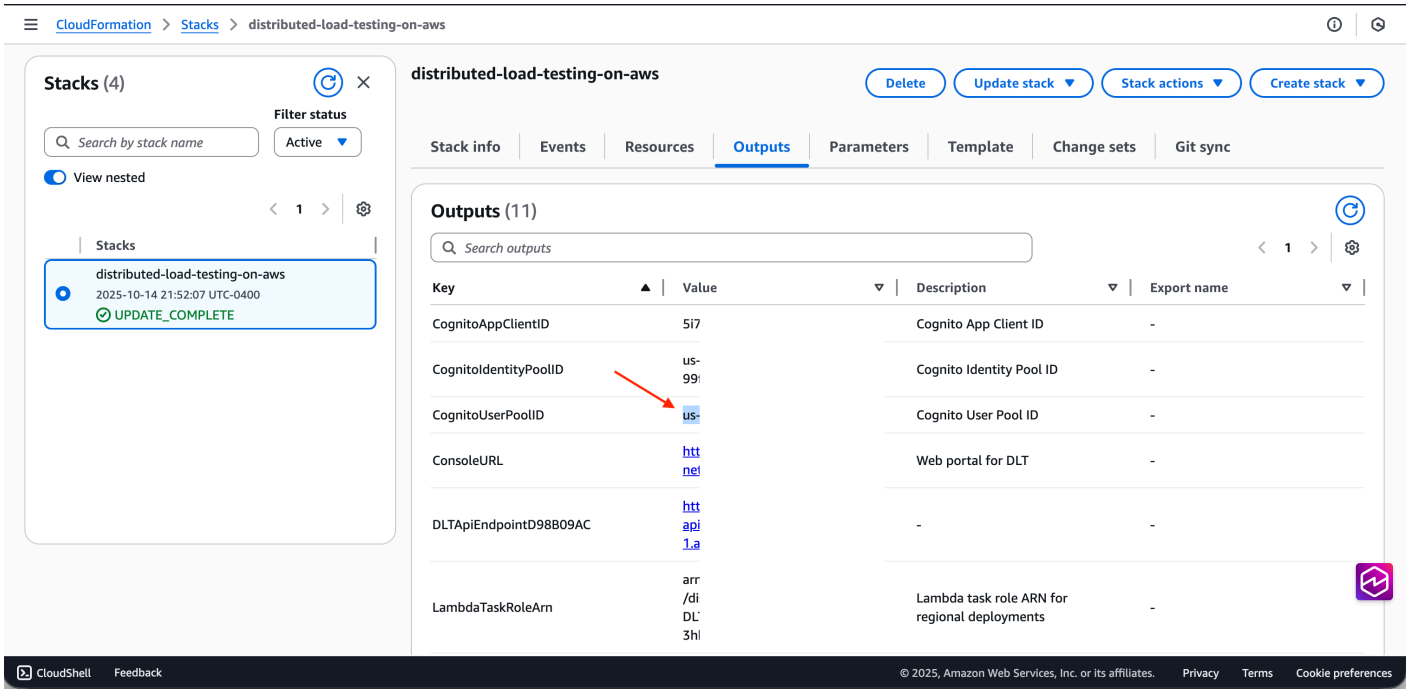
Amazon Q Suite

Amazon Q Suite 提供全方位的 AI 助理平台，並支援 MCP 伺服器動作。

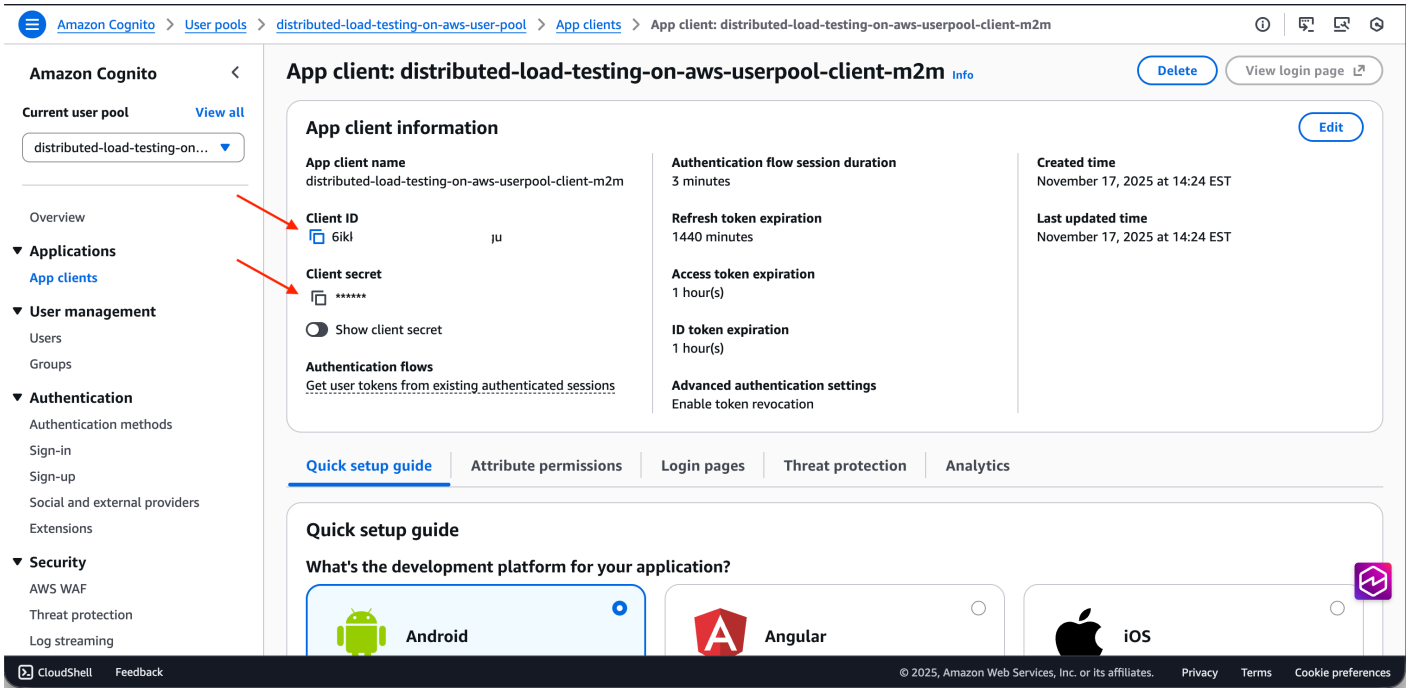
先決條件

在 Amazon Q Suite 中設定 MCP 伺服器之前，您需要從 DLT 部署的 Cognito 使用者集區擷取 OAuth 憑證：

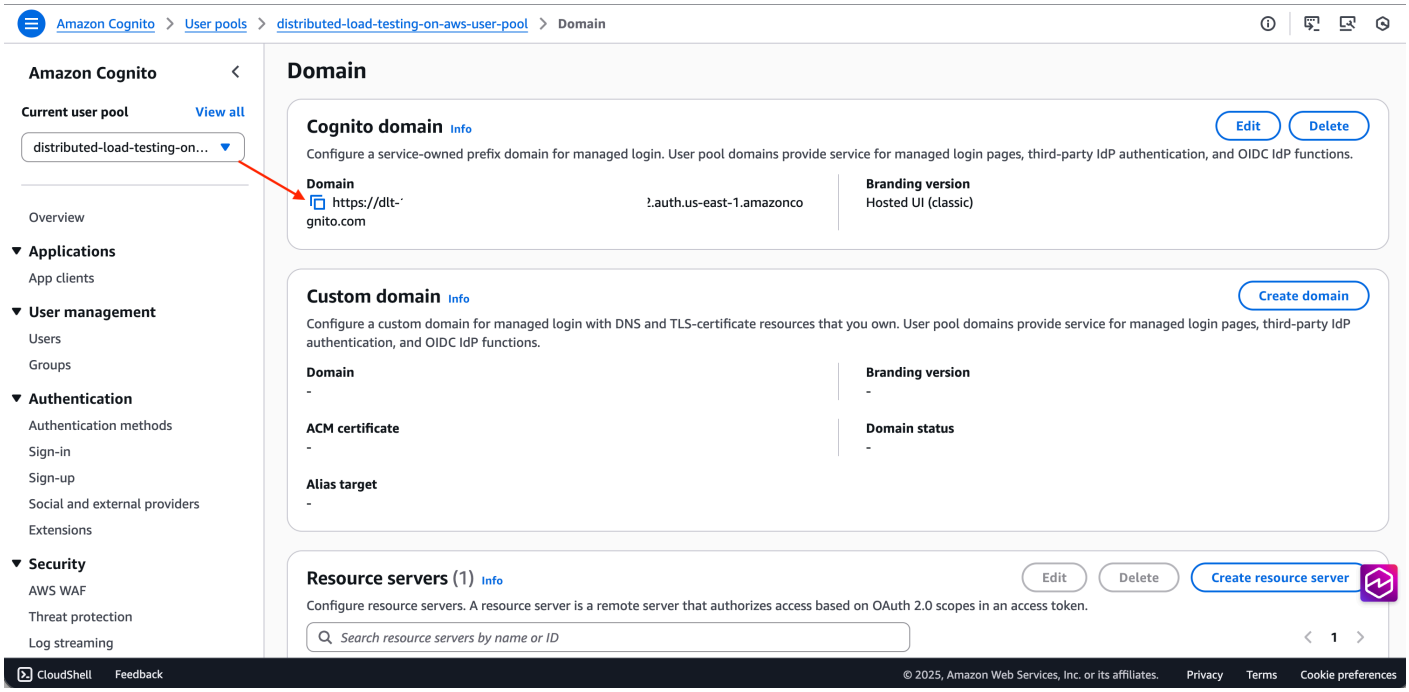
1. 導覽至 AWS CloudFormation 主控台。
2. 選取分散式負載測試堆疊。
3. 在輸出索引標籤中，尋找並複製與您的 DLT 部署相關聯的 Cognito 使用者集區 ID。



4. 導覽至 Amazon Cognito 主控台。
5. 使用 CloudFormation 輸出中的使用者集區 ID 選取使用者集區。
6. 在左側導覽中，選取應用程式整合 > 應用程式用戶端。



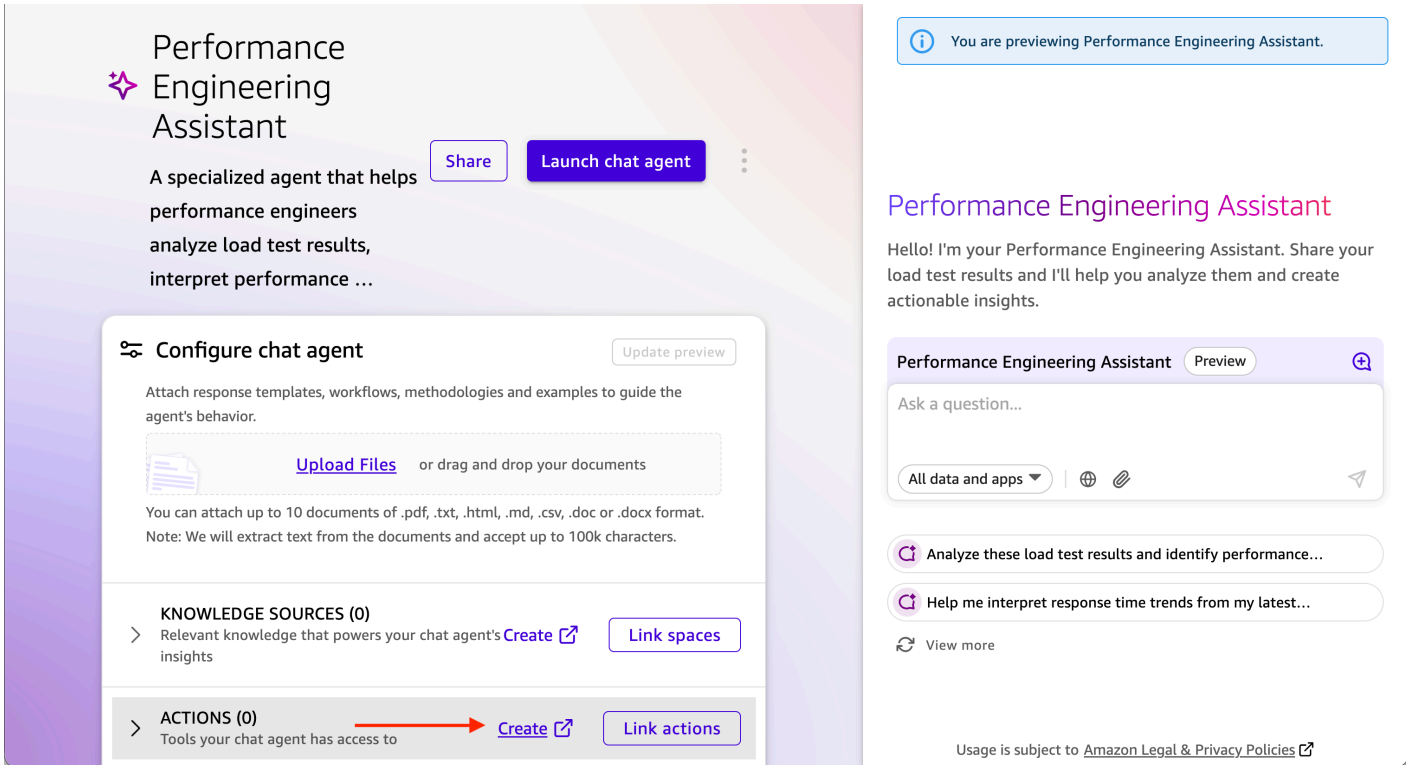
7. 找到名稱結尾為 m2m(machine-to-machine) 的應用程式用戶端。
8. 複製用戶端 ID 和用戶端秘密。
9. 從網域索引標籤取得使用者集區網域。

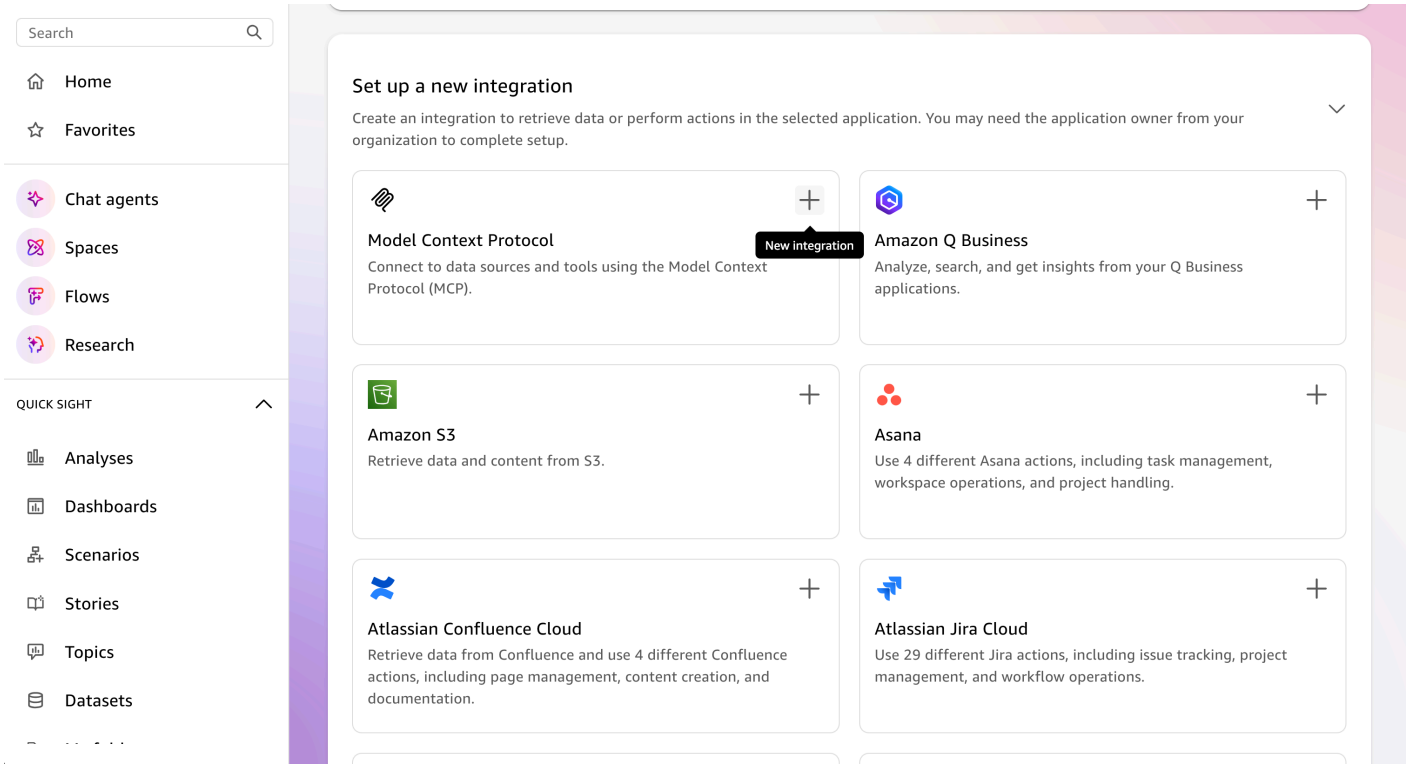


10透過附加/oauth2/token到網域的結尾來建構字符端點 URL。

組態步驟

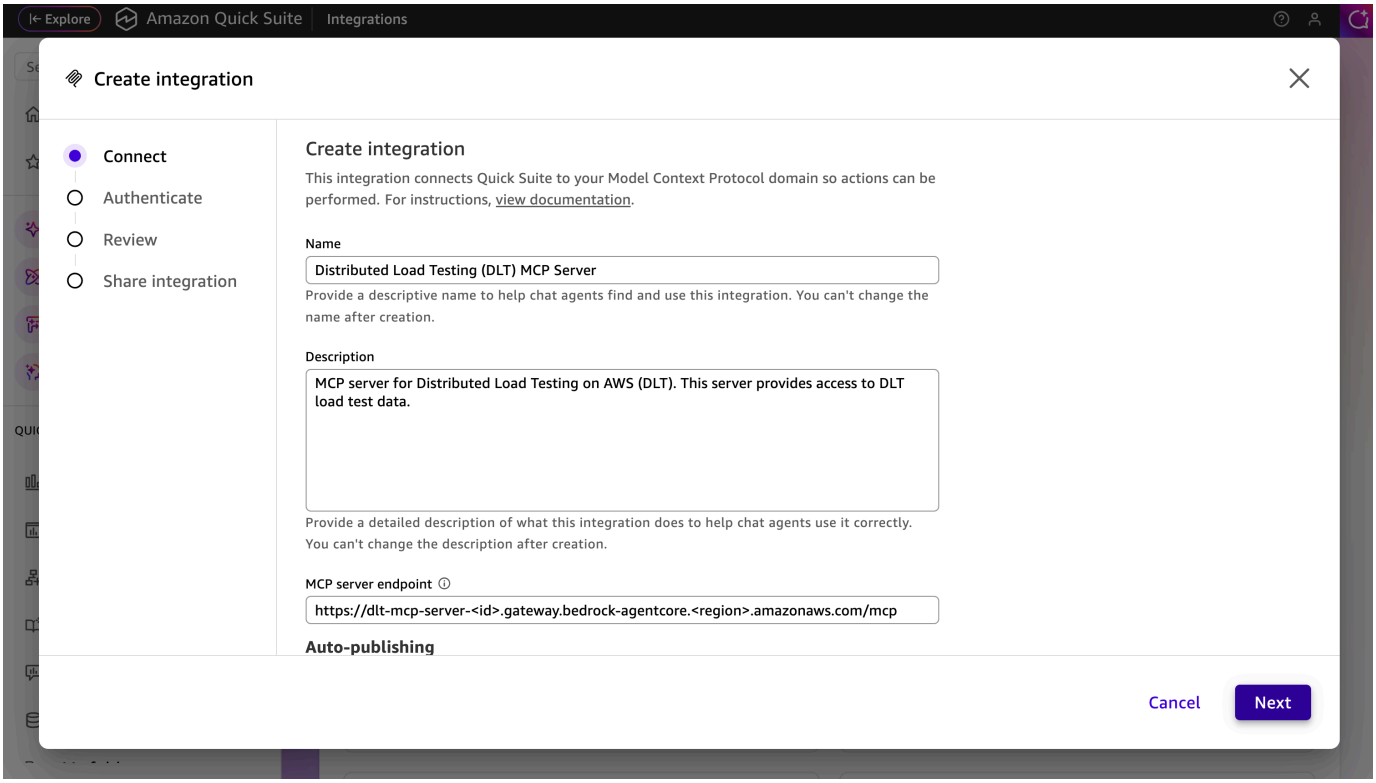
1. 在 Amazon Q Suite 中，建立新的客服人員或選取現有的客服人員。
2. 新增客服人員提示，說明如何與 DLT MCP 伺服器互動。
3. 新增動作，然後選取 MCP 伺服器動作。



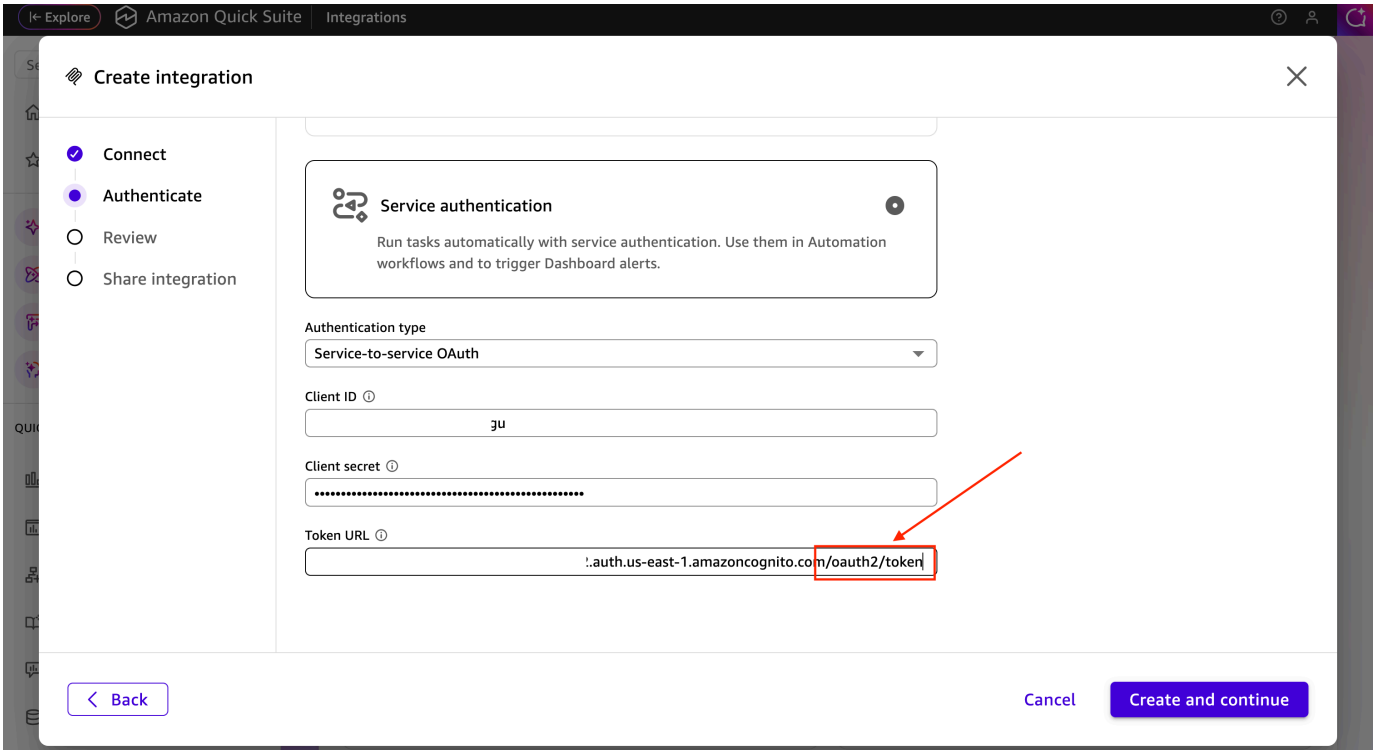


4. 設定 MCP 伺服器詳細資訊：

- MCP 伺服器 URL：您的 DLT MCP 端點



- 身分驗證類型：服務型身分驗證
- 權杖端點：您的 Cognito 權杖端點 URL
- 用戶端 ID：來自 m2m 應用程式用戶端的用戶端 ID
- 用戶端秘密：來自 m2m 應用程式用戶端的用戶端秘密



5. 儲存 MCP 伺服器動作組態。
6. 將新的 MCP 伺服器動作新增至您的代理程式。

啟動和測試代理程式

1. 在 Amazon Q Suite 中啟動代理程式。
2. 使用自然語言提示與客服人員開始對話。
3. 代理程式將使用 MCP 工具來擷取和分析負載測試資料。

範例提示

下列範例示範如何與 AI 助理互動，以透過 MCP 界面分析負載測試資料。自訂測試 IDs、日期範圍和條件，以符合您的特定測試需求。

如需可用 MCP 工具及其參數的詳細資訊，請參閱《 開發人員指南》中的 MCP 工具規格。

簡單測試結果查詢

與 MCP Server 的正式語言互動可以像 一樣簡單`Show me the load tests that have completed in the last 24 hours with their associated completion status`，也可以更描述性，例如

```
Use list_scenarios to find my load tests. Then use get_latest_test_run to show me the
 basic execution data and performance metrics for the most recent test. If the results
 look concerning, also get the detailed performance metrics using get_test_run.
```

互動式效能分析與漸進式揭露

```
I need to analyze my load test performance, but I'm not sure which specific tests to
 focus on. Please help me by:

1. First, use list_scenarios to show me available test scenarios
2. Ask me which tests I want to analyze based on the list you show me
3. For my selected tests, use list_test_runs to get the test run history
4. Then use get_test_run with the test_run_id to get detailed response times,
 throughput, and error rates
5. If I want to compare tests, use get_baseline_test_run to compare against the
 baseline
6. If there are any issues, use get_test_run_artifacts to help me understand what went
 wrong

Please guide me through this step by step, asking for clarification whenever you need
 more specific information.
```

生產準備驗證

```
Help me validate if my API is ready for production deployment:

1. Use list_scenarios to find recent test scenarios
2. For the most recent test scenario, use get_latest_test_run to get basic execution
 data
3. Use get_test_run with that test_run_id to get detailed response times, error rates,
 and throughput
4. Use get_scenario_details with the test_id to show me what load patterns and
 endpoints were tested
5. If I have a baseline, use get_baseline_test_run to compare current results with the
 baseline
6. Provide a clear go/no-go recommendation based on the performance data
7. If there are any concerns, use get_test_run_artifacts to help identify potential
 issues

My SLA requirements are: response time under [X]ms, error rate under [Y]%.
```

效能趨勢分析

```
Analyze the performance trend for my load tests over the past [TIME_PERIOD]:

1. Use list_scenarios to get all test scenarios
2. For each scenario, use list_test_runs with start_date and end_date to get tests from
 that period
3. Use get_test_run for the key test runs to get detailed metrics
4. Use get_baseline_test_run to compare against the baseline
5. Identify any significant changes in response times, error rates, or throughput
6. If you detect performance degradation, use get_test_run_artifacts on the problematic
 tests to help identify causes
7. Present the trend analysis in a clear format showing whether performance is
 improving, stable, or degrading

Focus on completed tests and limit results to [N] tests if there are too many.
```

故障診斷失敗的測試

```
Help me troubleshoot my failed load tests:

1. Use list_scenarios to find test scenarios
2. For each scenario, use list_test_runs to find recent test runs
3. Use get_test_run with the test_run_id to get the basic execution data and failure
 information
4. Use get_test_run_artifacts to get detailed error messages and logs
5. Use get_scenario_details to understand what was being tested when it failed
6. If I have a similar test that passed, use get_baseline_test_run to identify
 differences
7. Summarize the causes of failure and suggest next steps for resolution

Show me the most recent [N] failed tests from the past [TIME_PERIOD].
```

開發人員指南

本節提供解決方案的原始程式碼和其他自訂項目。

來源碼

請造訪我們的 GitHub 儲存庫，下載此解決方案的範本和指令碼，並與他人共用您的自訂項目。

Maintenance (維護)

此解決方案使用 Docker 映像搭配對應於每個解決方案版本的固定版本。根據預設，自動更新會停用，讓您完全控制何時以及將哪些版本更新套用至部署。AWS 解決方案團隊使用 Amazon ECR 增強型掃描來偵測基礎映像和已安裝套件中的常見漏洞與暴露 (CVEs)。如果可能，團隊會發佈具有相同版本標籤的修補映像，以解決 CVEs而不會中斷與發行解決方案版本的相容性。

在相同的次要版本上修補映像時，會自動更新穩定標籤，並以 格式建立額外的映像標籤`<solution-version>_<date-of-fix>`。如果發行主要或次要版本，您必須執行完整堆疊更新以取得最新的映像版本，因為穩定標籤會遞增以符合解決方案版本。

如果您選擇在部署期間自動更新，映像的變更，包括 CVE 修補程式和次要錯誤修正，會自動套用到最新的相符次要版本。

版本

根據預設，此解決方案會部署並停用自動更新。這表示容器映像版本會鎖定至符合您部署解決方案版本的特定版本，讓您完全控制版本更新。

如果您選擇在 CloudFormation 部署期間選取是來啟用自動更新，解決方案會自動收到安全修補程式和直到最新相符次要版本的次要、不中斷的錯誤修正。例如，如果您在啟用自動更新的情況下部署 4.0.0 版，您會收到最多 4.0.x 的更新，但不會收到 4.1.0 或更新版本的更新。

若要手動控制容器映像版本，您可以編輯任務定義，以使用標記的版本格式指定特定的映像版本。這可讓您鎖定特定映像版本，無論自動更新設定為何。

容器映像自訂

此解決方案使用由 AWS 管理的公有 Amazon Elastic Container Registry (Amazon ECR) 映像儲存庫來存放用來執行已設定測試的映像。如果您想要自訂容器映像，您可以重建映像並將其推送到您自己的 AWS 帳戶中的 ECR 映像儲存庫。

如果您想要自訂此解決方案，您可以使用預設容器映像，或編輯此容器以符合您的需求。如果您自訂解決方案，請在建置自訂解決方案之前，使用以下程式碼範例宣告環境變數。

```
#!/bin/bash
export REGION=aws-region-code # the AWS region to launch the solution (e.g. us-east-1)
export BUCKET_PREFIX=my-bucket-name # prefix of the bucket name without the region code
export BUCKET_NAME=$BUCKET_PREFIX-$REGION # full bucket name where the code will reside
export SOLUTION_NAME=my-solution-name
export VERSION=my-version # version number for the customized code
export PUBLIC_ECR_REGISTRY=public.ecr.aws/awssolutions/distributed-load-testing-on-
aws-load-tester # replace with the container registry and image if you want to use a
 different container image export PUBLIC_ECR_TAG=v3.1.0 # replace with the container
 image tag if you want to use a different container image
```

如果您選擇自訂容器映像，可以在私有映像儲存庫中託管它，也可以在 AWS 帳戶中託管公有映像儲存庫。映像資源位於 `deployment/ecr/distributed-load-testing-on-aws-load-tester`目錄中，位於程式碼庫中。

您可以建置映像並將其推送至主機目的地。

- 對於私有 Amazon ECR 儲存庫和映像，請參閱《Amazon ECR 使用者指南》中的 Amazon ECR 私有儲存庫和私有映像。
- 如需公有 Amazon ECR 儲存庫和映像，請參閱《Amazon ECR 公有使用者指南》中的 Amazon ECR 公有儲存庫和公有映像。

建立自己的映像後，您可以在建置自訂解決方案之前宣告下列環境變數。

```
#!/bin/bash
export PUBLIC_ECR_REGISTRY=YOUR_ECR_REGISTRY_URI # e.g. YOUR_ACCOUNT_ID.dkr.ecr.us-
east-1.amazonaws.com/YOUR_IMAGE_NAME
export PUBLIC_ECR_TAG=YOUR_ECR_TAG # e.g. latest, v3.4.0
```

下列範例顯示容器檔案。

```
FROM public.ecr.aws/amazonlinux/amazonlinux:2023-minimal

RUN dnf update -y && \
    dnf install -y python3.11 python3.11-pip java-21-amazon-corretto bc procps jq
 findutils unzip && \
    dnf clean all

ENV PIP_INSTALL="pip3.11 install --no-cache-dir"


# install bzt
RUN $PIP_INSTALL --upgrade bzt awscli setuptools==78.1.1 h11 urllib3==2.2.2 && \
    $PIP_INSTALL --upgrade bzt
COPY ./.bzt-rc /root/.bzt-rc
RUN chmod 755 /root/.bzt-rc

# install bzt tools
RUN bzt -install-tools -o modules.install-
checker.exclude=selenium,gatling,tsung,siege,ab,k6,external-results-
loader,locust,junit,testng,rspec,mocha,nunit,xunit,wdio,robot,newman
RUN rm -rf /root/.bzt/selenium-taurus
RUN mkdir /bzt-configs /tmp/artifacts
ADD ./load-test.sh /bzt-configs/
ADD ./*.jar /bzt-configs/
ADD ./*.py /bzt-configs/

RUN chmod 755 /bzt-configs/load-test.sh
RUN chmod 755 /bzt-configs/ecslistener.py
RUN chmod 755 /bzt-configs/ecscontroller.py
RUN chmod 755 /bzt-configs/jar_updater.py
RUN python3.11 /bzt-configs/jar_updater.py

# Remove jar files from /tmp
RUN rm -rf /tmp/jmeter-plugins-manager-1* && \
    rm -rf /usr/local/lib/python3.11/site-packages/setuptools-65.5.0.dist-info && \
    rm -rf /usr/local/lib/python3.11/site-packages/urllib3-1.26.17.dist-info

# Add settings file to capture the output logs from bzt cli
RUN mkdir -p /etc/bzt.d && echo '{"settings": {"artifacts-dir": "/tmp/artifacts"}}' > /
etc/bzt.d/90-artifacts-dir.json

WORKDIR /bzt-configs
ENTRYPOINT ["./load-test.sh"]
```

除了容器檔案之外，目錄還包含下列 bash 指令碼，可在執行 Taurus/Blazemeter 程式之前從 Amazon S3 下載測試組態。

```
#!/bin/bash

# set a uuid for the results xml file name in S3
UUID=$(cat /proc/sys/kernel/random/uuid)
pypid=0
echo "S3_BUCKET:: ${S3_BUCKET}"
echo "TEST_ID:: ${TEST_ID}"
echo "TEST_TYPE:: ${TEST_TYPE}"
echo "FILE_TYPE:: ${FILE_TYPE}"
echo "PREFIX:: ${PREFIX}"
echo "UUID:: ${UUID}"
echo "LIVE_DATA_ENABLED:: ${LIVE_DATA_ENABLED}"
echo "MAIN_STACK_REGION:: ${MAIN_STACK_REGION}"

cat /proc/self/cgroup
TASK_ID=$(grep -oE '[a-f0-9]{32}' /proc/self/cgroup | head -n 1)
echo $TASK_ID

sigterm_handler() {
  if [ $pypid -ne 0 ]; then
    echo "container received SIGTERM."
    kill -15 $pypid
    wait $pypid
    exit 143 #128 + 15
  fi
}
trap 'sigterm_handler' SIGTERM

echo "Download test scenario"
aws s3 cp s3://$S3_BUCKET/test-scenarios/$TEST_ID-$AWS_REGION.json test.json --region
 $MAIN_STACK_REGION

# Set the default log file values to jmeter
LOG_FILE="jmeter.log"
OUT_FILE="jmeter.out"
ERR_FILE="jmeter.err"
KPI_EXT="jtl"

# download JMeter jmx file
if [ "$TEST_TYPE" != "simple" ]; then
```

```
  # setting the log file values to the test type
  LOG_FILE="${TEST_TYPE}.log"
  OUT_FILE="${TEST_TYPE}.out"
  ERR_FILE="${TEST_TYPE}.err"

  # set variables based on TEST_TYPE
  if [ "$TEST_TYPE" == "jmeter" ]; then
    EXT="jmx"
    TYPE_NAME="JMeter"
    # Copy *.jar to JMeter library path. See the Taurus JMeter path: https://
gettaurus.org/docs/JMeter/
    JMETER_LIB_PATH=`find ~/.bzt/jmeter-taurus -type d -name "lib"`
    echo "cp $PWD/*.jar $JMETER_LIB_PATH"
    cp $PWD/*.jar $JMETER_LIB_PATH
  elif [ "$TEST_TYPE" == "k6" ]; then
    curl --output /tmp/artifacts/k6.rpm https://dl.k6.io/rpm/x86_64/k6-v0.58.0-
amd64.rpm
    rpm -ivh /tmp/artifacts/k6.rpm
    dnf install -y k6
    rm -rf /tmp/artifacts/k6.rpm
    EXT="js"
    KPI_EXT="csv"
    TYPE_NAME="K6"
  elif [ "$TEST_TYPE" == "locust" ]; then
    EXT="py"
    TYPE_NAME="Locust"

  fi

  if [ "$FILE_TYPE" != "zip" ]; then
    aws s3 cp s3://$S3_BUCKET/public/test-scenarios/$TEST_TYPE/$TEST_ID.$EXT ./ --
region $MAIN_STACK_REGION
  else
    aws s3 cp s3://$S3_BUCKET/public/test-scenarios/$TEST_TYPE/$TEST_ID.zip ./ --region
 $MAIN_STACK_REGION
    unzip $TEST_ID.zip
    echo "UNZIPPED"
    ls -l

    # If zip and locust, make sure to pick locustfile
    if [ "$TEST_TYPE" != "locust" ]; then
      TEST_SCRIPT=$(find . -name "*.${EXT}" | head -n 1)
    else
      TEST_SCRIPT=$(find . -name "locustfile.py" | head -n 1)
```

```
    fi
    # only looks for the first test script file.
    TEST_SCRIPT=`find . -name "*.${EXT}" | head -n 1`
    echo $TEST_SCRIPT
    if [ -z "$TEST_SCRIPT" ]; then
      echo "There is no test script (.${EXT}) in the zip file."
      exit 1
    fi

    sed -i -e "s|$TEST_ID.$EXT|$TEST_SCRIPT|g" test.json

    # copy bundled plugin jars to jmeter extension folder to make them available to
 jmeter
    BUNDLED_PLUGIN_DIR=`find $PWD -type d -name "plugins" | head -n 1`
    # attempt to copy only if a /plugins folder is present in upload
    if [ -z "$BUNDLED_PLUGIN_DIR" ]; then
      echo "skipping plugin installation (no /plugins folder in upload)"
    else
      # ensure the jmeter extensions folder exists
      JMETER_EXT_PATH=`find ~/.bzt/jmeter-taurus -type d -name "ext"`
      if [ -z "$JMETER_EXT_PATH" ]; then
        # fail fast - if plugins bundled they will be needed for the tests
        echo "jmeter extension path (~/.bzt/jmeter-taurus/**/ext) not found - cannot
 install bundled plugins"
        exit 1
      fi
      cp -v $BUNDLED_PLUGIN_DIR/*.jar $JMETER_EXT_PATH
    fi
  fi
fi

#Download python script
if [ -z "$IPNETWORK" ]; then
    python3.11 -u $SCRIPT  $TIMEOUT &
    pypid=$!
    wait $pypid
    pypid=0
else
    aws s3 cp s3://$S3_BUCKET/Container_IPs/${TEST_ID}_IPHOSTS_${AWS_REGION}.txt ./ --
region $MAIN_STACK_REGION
    export IPHOSTS=$(cat ${TEST_ID}_IPHOSTS_${AWS_REGION}.txt)
    python3.11 -u $SCRIPT $IPNETWORK $IPHOSTS
fi
```

```
echo "Running test"

stdbuf -i0 -o0 -e0 bzt test.json -o modules.console.disable=true | stdbuf -i0 -o0 -e0
 tee -a result.tmp | sed -u -e "s|^|$TEST_ID $LIVE_DATA_ENABLED |"
CALCULATED_DURATION=`cat result.tmp | grep -m1 "Test duration" | awk -F ' ' '{ print
 $5 }' | awk -F ':' '{ print ($1 * 3600) + ($2 * 60) + $3 }'`

# upload custom results to S3 if any
# every file goes under $TEST_ID/$PREFIX/$UUID to distinguish the result correctly
if [ "$TEST_TYPE" != "simple" ]; then
  if [ "$FILE_TYPE" != "zip" ]; then
    cat $TEST_ID.$EXT | grep filename > results.txt
  else
    cat $TEST_SCRIPT | grep filename > results.txt
  fi

  if [ -f results.txt ]; then
    sed -i -e 's/<stringProp name="filename">//g' results.txt
    sed -i -e 's/<\/stringProp>//g' results.txt
    sed -i -e 's/ //g' results.txt

    echo "Files to upload as results"
    cat results.txt

    files=(`cat results.txt`)
    extensions=()
    for f in "${files[@]}"; do
      ext="${f##*.}"
      if [[ ! " ${extensions[@]} " =~ " ${ext} " ]]; then
        extensions+=("$ext")
      fi
    done

    # Find all files in the current folder with the same extensions
    all_files=()
    for ext in "${extensions[@]}"; do
      for f in *."$ext"; do
        all_files+=("$f")
      done
    done

    for f in "${all_files[@]}"; do
      p="s3://$S3_BUCKET/results/$TEST_ID/${TYPE_NAME}_Result/$PREFIX/$UUID/$f"
      if [[ $f = /* ]]; then
```

```
        p="s3://$S3_BUCKET/results/$TEST_ID/${TYPE_NAME}_Result/$PREFIX/$UUID$f"
      fi

        echo "Uploading $p"
        aws s3 cp $f $p --region $MAIN_STACK_REGION
    done
    fi
fi

if [ -f /tmp/artifacts/results.xml ]; then

  # Insert the Task ID at the same level as <FinalStatus>
  curl -s $ECS_CONTAINER_METADATA_URI_V4/task
  Task_CPU=$(curl -s $ECS_CONTAINER_METADATA_URI_V4/task | jq '.Limits.CPU')
  Task_Memory=$(curl -s $ECS_CONTAINER_METADATA_URI_V4/task | jq '.Limits.Memory')
  START_TIME=$(curl -s "$ECS_CONTAINER_METADATA_URI_V4/task" | jq -r
 '.Containers[0].StartedAt')
  # Convert start time to seconds since epoch
  START_TIME_EPOCH=$(date -d "$START_TIME" +%s)
  # Calculate elapsed time in seconds
  CURRENT_TIME_EPOCH=$(date +%s)
  ECS_DURATION=$((CURRENT_TIME_EPOCH - START_TIME_EPOCH))


  sed -i.bak 's/<\/FinalStatus>/<TaskId>'"$TASK_ID"'<\/TaskId><\/FinalStatus>/' /tmp/
artifacts/results.xml
  sed -i 's/<\/FinalStatus>/<TaskCPU>'"$Task_CPU"'<\/TaskCPU><\/FinalStatus>/' /tmp/
artifacts/results.xml
  sed -i 's/<\/FinalStatus>/<TaskMemory>'"$Task_Memory"'<\/TaskMemory><\/
FinalStatus>/' /tmp/artifacts/results.xml
  sed -i 's/<\/FinalStatus>/<ECSDuration>'"$ECS_DURATION"'<\/ECSDuration><\/
FinalStatus>/' /tmp/artifacts/results.xml

  echo "Validating Test Duration"
  TEST_DURATION=$(grep -E '<TestDuration>[0-9]+.[0-9]+</TestDuration>' /tmp/artifacts/
results.xml | sed -e 's/<TestDuration>//' | sed -e 's/<\/TestDuration>//')

  if (( $(echo "$TEST_DURATION > $CALCULATED_DURATION" | bc -l) )); then
    echo "Updating test duration: $CALCULATED_DURATION s"
    sed -i.bak.td 's/<TestDuration>[0-9]*\.[0-9]*<\/TestDuration>/
<TestDuration>'"$CALCULATED_DURATION"'<\/TestDuration>/' /tmp/artifacts/results.xml
  fi

  if [ "$TEST_TYPE" == "simple" ]; then
```

```
    TEST_TYPE="jmeter"
  fi

  echo "Uploading results, bzt log, and JMeter log, out, and err files"
  aws s3 cp /tmp/artifacts/results.xml s3://$S3_BUCKET/results/${TEST_ID}/${PREFIX}-
${UUID}-${AWS_REGION}.xml --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/bzt.log s3://$S3_BUCKET/results/${TEST_ID}/bzt-${PREFIX}-
${UUID}-${AWS_REGION}.log --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/$LOG_FILE s3://$S3_BUCKET/results/${TEST_ID}/${TEST_TYPE}-
${PREFIX}-${UUID}-${AWS_REGION}.log --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/$OUT_FILE s3://$S3_BUCKET/results/${TEST_ID}/${TEST_TYPE}-
${PREFIX}-${UUID}-${AWS_REGION}.out --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/$ERR_FILE s3://$S3_BUCKET/results/${TEST_ID}/${TEST_TYPE}-
${PREFIX}-${UUID}-${AWS_REGION}.err --region $MAIN_STACK_REGION
  aws s3 cp /tmp/artifacts/kpi.${KPI_EXT} s3://$S3_BUCKET/results/${TEST_ID}/kpi-
${PREFIX}-${UUID}-${AWS_REGION}.${KPI_EXT} --region $MAIN_STACK_REGION

else
  echo "An error occurred while the test was running."
fi
```

除了 Dockerfile 和 bash 指令碼之外，目錄中還包含兩個 Python 指令碼。每個任務從 bash 指令碼內執行 Python 指令碼。工作者任務會執行`ecslistener.py`指令碼，而領導任務則會執行`ecscontroller.py`指令碼。`ecslistener.py` 指令碼會在連接埠 50000 上建立通訊端，並等待訊息。`ecscontroller.py` 指令碼會連線至通訊端，並將啟動測試訊息傳送至工作者任務，讓它們可以同時啟動。

分散式負載測試 API

此負載測試解決方案可協助您以安全的方式公開測試結果資料。API 做為「前門」，用於存取存放在 Amazon DynamoDB 中的測試資料。您也可以使用 APIs來存取您建置到解決方案中的任何延伸功能。

此解決方案使用與 Amazon API Gateway 整合的 Amazon Cognito 使用者集區來識別和授權。Amazon API Gateway 當使用者集區與 API 搭配使用時，用戶端只能在提供有效的身分字符之後呼叫使用者集區啟用的方法。

如需直接透過 API 執行測試的詳細資訊，請參閱 Amazon API Gateway REST API 參考文件中的簽署請求。

下列操作可在解決方案的 API 中使用。

> **Note**
>
> 如需 `testScenario` 和其他參數的詳細資訊，請參閱 GitHub 儲存庫中的案例和承載範例。

堆疊資訊

- GET /stack-info

案例

- GET /案例
- POST/案例
- OPTIONS/案例
- GET /scenarios/{testId}
- POST /scenarios/{testId}
- DELETE /scenarios/{testId}
- OPTIONS /scenarios/{testId}

測試執行

- GET /scenarios/{testId}/testruns
- GET /scenarios/{testId}/testruns/{testRunId}
- DELETE /scenarios/{testId}/testruns/{testRunId}

基準

- GET /scenarios/{testId}/baseline
- PUT /scenarios/{testId}/baseline
- DELETE /scenarios/{testId}/baseline

工作

- GET /任務
- 選項/任務

區域

- GET/區域
- 選項/區域

GET /stack-info

說明

GET /stack-info 操作會擷取已部署堆疊的相關資訊，包括建立時間、區域和版本。前端會使用此端點。

回應

200 - 成功

名稱	描述
created_time	建立堆疊時的 ISO 8601 時間戳記（例如 2025-09-09T19:40:22Z ）
region	部署堆疊的 AWS 區域（例如 us-east-1 ）
version	部署解決方案的版本（例如 v4.0.0)

錯誤回應

- 403 - 禁止：存取堆疊資訊的許可不足
- 404 - 找不到：堆疊資訊無法使用
- 500 - 內部伺服器錯誤

GET /案例

說明

GET /scenarios 操作可讓您擷取測試案例的清單。

回應

名稱	描述
data	案例清單，包括每個測試的 ID、名稱、描述、狀態、執行時間、標籤、總執行次數和上次執行次數

POST/情境

說明

POST /scenarios 操作可讓您建立或排程測試案例。

請求內文

名稱	描述
testName	測試的名稱
testDescription	測試的描述
testTaskConfigs	指定 concurrency （平行執行的數目）、taskCount （執行測試所需的任務數目）以及 案例region的物件
testScenario	測試定義，包括測試的並行、測試時間、主機和方法
testType	測試類型（例如，simple、jmeter)
fileType	上傳檔案類型（例如 、nonescript、zip)
tags	用於分類測試的字串陣列。長度上限為 5 的選用欄位（例如 ["blue", "3.0", "critical"] ）

名稱	描述
scheduleDate	執行測試的日期。僅在排程測試時提供（例如，2021-02-28 ）
scheduleTime	執行測試的時間。僅在排程測試時提供（例如，21:07)
scheduleStep	排程程序中的步驟。僅在排程週期性測試時提供。（可用的步驟包括 create和 start)
cronvalue	自訂週期性排程的 Cron 值。如果使用，請省略 scheduleDate 和 scheduleTime。
cronExpiryDate	必要日期，讓 Cron 過期且不會無限期執行。
recurrence	排程測試的週期。僅在排程重複測試時提供（例如，、biweekly、 daily weekly或 monthly)

回應

名稱	描述
testId	測試的唯一 ID
testName	測試的名稱
status	測試的狀態

選項/案例

說明

OPTIONS /scenarios 操作為具有正確 CORS 回應標頭的請求提供回應。

回應

名稱	描述
testId	測試的唯一 ID
testName	測試的名稱
status	測試的狀態

GET /scenarios/{testId}

說明

GET /scenarios/{testId} 操作可讓您擷取特定測試案例的詳細資訊。

請求參數

testId

- 測試的唯一 ID

  類型：字串

  必要：是

latest

- 僅傳回最新測試執行的查詢參數。預設值為 true

  類型：布林值

  必要：否

history

- 查詢參數以在回應中包含測試執行歷史記錄。預設值為 true。設定為 false以排除歷史記錄

  類型：布林值

  必要：否

回應

名稱	描述
testId	測試的唯一 ID
testName	測試的名稱
testDescription	測試的描述
testType	執行的測試類型（例如，simple、jmeter)
fileType	上傳的檔案類型（例如 none、script、zip)
tags	用於分類測試的字串陣列
status	測試的狀態
startTime	上次測試開始的時間和日期
endTime	上次測試結束的時間和日期
testScenario	測試定義，包括測試的並行、測試時間、主機和方法
taskCount	執行測試所需的任務數量
taskIds	執行測試的任務 IDs 清單
results	測試的最終結果
history	過去測試的最終結果清單（當 時除外history=false ）
totalRuns	此案例的測試執行總數
lastRun	上次測試執行的時間戳記
errorReason	發生錯誤時產生的錯誤訊息

名稱	描述
nextRun	下一個排定的執行（例如 2017-04-22 17:18:00)
scheduleRecurrence	測試的週期（例如，、dailyweekly、biweekly、monthly)

POST /scenarios/{testId}

說明

POST /scenarios/{testId} 操作可讓您取消特定測試案例。

請求參數

testId

- 測試的唯一 ID

  類型：字串

  必要：是

回應

名稱	描述
status	測試的狀態

DELETE /scenarios/{testId}

說明

DELETE /scenarios/{testId} 此操作可讓您刪除與特定測試案例相關的所有資料。

請求參數

`testId`

- 測試的唯一 ID

  類型：字串

  必要：是

回應

名稱	描述
`status`	測試的狀態

OPTIONS /scenarios/{testId}

說明

`OPTIONS /scenarios/{testId}` 操作為具有正確 CORS 回應標頭的請求提供回應。

回應

名稱	描述
`testId`	測試的唯一 ID
`testName`	測試的名稱
`testDescription`	測試的描述
`testType`	執行的測試類型（例如，`simple`、`jmeter`)
`fileType`	上傳的檔案類型（例如 `none`、`script`、`zip`)
`status`	測試的狀態
`startTime`	上次測試開始的時間和日期

名稱	描述
endTime	上次測試結束的時間和日期
testScenario	測試定義，包括測試的並行、測試時間、主機和方法
taskCount	執行測試所需的任務數量
taskIds	執行測試的任務 IDs 清單
results	測試的最終結果
history	過去測試的最終結果清單
errorReason	發生錯誤時產生的錯誤訊息

GET /scenarios/{testId}/testruns

說明

GET /scenarios/{testId}/testruns 操作會擷取特定測試案例的測試執行 IDs，選擇性地依時間範圍篩選。當 時latest=true， 只會傳回最新的單一測試執行。

請求參數

testId

- 測試案例 ID

  類型：字串

  必要：是

latest

- 僅傳回最新的測試執行 ID

  類型：布林值

  預設：false

  必要：否

`start_timestamp`

- ISO 8601 時間戳記，用於篩選從（包含）執行的測試。例如 `2024-01-01T00:00:00Z`

  類型：字串（日期時間格式）

  必要：否

`end_timestamp`

- 篩選測試執行的 ISO 8601 時間戳記，直到（包含）。例如 `2024-12-31T23:59:59Z`

  類型：字串（日期時間格式）

  必要：否

`limit`

- 要傳回的測試執行數目上限（當 時忽略`latest=true`)

  類型：整數（最小值：1，最大值：100)

  預設：`20`

  必要：否

`next_token`

- 上一個回應的分頁字符以取得下一頁

  類型：字串

  必要：否

回應

200 - 成功

名稱	描述
`testRuns`	測試執行物件陣列，每個都包含 `testRunId`（字串）和 `startTime` (ISO 8601 日期時間)

名稱	描述
pagination	物件包含 limit（整數）和 next_token （字串或 null)。如果沒有更多結果，字符為 null

錯誤回應

- 400 - 無效的時間戳記格式或參數
- 404 - 找不到測試案例
- 500 - 內部伺服器錯誤

範例使用方式

- 僅限最新測試執行： GET /scenarios/test123/testruns?latest=true
- 時間範圍內的最新時間： GET /scenarios/test123/testruns?latest=true&start_timestamp=2024-01-01T00:00:00Z
- 下一頁請求： GET /scenarios/test123/testruns?limit=20&next_token=eyJ0ZXN0SWQiOiJzZVFVeTEyTEtMIiwic3RhcnRUaW1lIjoiMjAyNC0wMS0

GET /scenarios/{testId}/testruns/{testRunId}

說明

GET /scenarios/{testId}/testruns/{testRunId} 操作會擷取特定測試執行的完整結果和指標。選擇性地使用 省略歷史記錄結果history=false，以加快回應速度。

請求參數

testId

- 測試案例 ID

  類型：字串

  必要：是

testRunId

- 特定的測試執行 ID

  類型：字串

  必要：是

history

- 在回應中包含歷史記錄陣列。設定為 false可省略歷史記錄以加快回應速度

  類型：布林值

  預設：true

  必要：否

回應

200 - 成功

名稱	描述
testId	測試的唯一 ID （例如 seQUy12LKL ）
testRunId	特定的測試執行 ID （例如 2DEwHItEne ）
testDescription	負載測試的描述
testType	測試類型 （例如，simple、jmeter)
status	測試執行的狀態：complete、failed、running或 cancelled
startTime	測試開始的時間和日期 （例如 2025-09-09 21:01:00)
endTime	測試結束的時間和日期 （例如 2025-09-09 21:18:29)
succPercent	成功百分比 （例如 100.00)

名稱	描述
testTaskConfigs	包含 region、 taskCount 和 的任務組態物件陣列 concurrency
completeTasks	物件映射區域到已完成的任務計數
results	包含詳細指標的物件，包括 avg_lt（平均延遲）、百分位數 (p0_0、p50_0、p90_0p95_0、、p99_9、p100_0、)p avg_rt （平均回應時間）、 avg_ct （平均連線時間）、 stdev_rt （標準差回應時間）concurrency 、、throughput 、 succ（成功計數）、 fail （失敗計數）bytes、、testDuration 、metricS3Location 、 rc（回應代碼陣列）和 labels陣列
testScenario	物件包含具有 execution 、 reporting 和 scenarios 屬性的測試組態
history	歷史測試結果陣列（當 時排除history=false ）

錯誤回應

- 400 - 無效的 testId 或 testRunId
- 404 - 找不到測試執行
- 500 - 內部伺服器錯誤

DELETE /scenarios/{testId}/testruns/{testRunId}

說明

DELETE /scenarios/{testId}/testruns/{testRunId} 操作會刪除與特定測試執行相關的所有資料和成品。測試執行資料會從 DynamoDB 中移除，而 S3 中的實際測試資料保持不變。

請求參數

`testId`

- 測試案例 ID

  類型：字串

  必要：是

`testRunId`

- 要刪除的特定測試執行 ID

  類型：字串

  必要：是

回應

204 - 成功

測試執行已成功刪除（未傳回內容）

錯誤回應

- `400` - 無效的 testId 或 testRunId
- `403` - 禁止：刪除測試執行的許可不足
- `404` - 找不到測試執行
- `409` - 衝突：測試執行目前正在執行，無法刪除
- `500` - 內部伺服器錯誤

GET /scenarios/{testId}/baseline

說明

`GET /scenarios/{testId}/baseline` 操作會擷取案例的指定基準測試結果。根據 `data` 參數傳回基準測試執行 ID 或完整基準結果。

請求參數

`testId`

- 測試案例 ID

  類型：字串

  必要：是

`data`

- 如果 傳回完整的基準測試執行資料`true`，否則僅傳回 testRunId

  類型：布林值

  預設：`false`

  必要：否

回應

200 - 成功

時間 `data=false`（預設）：

名稱	描述
`testId`	測試案例 ID （例如 `seQUy12LKL` ）
`baselineTestRunId`	基準測試執行 ID （例如 `2DEwHItEne` ）

當 `data=true`時：

名稱	描述
`testId`	測試案例 ID （例如 `seQUy12LKL` ）
`baselineTestRunId`	基準測試執行 ID （例如 `2DEwHItEne` ）

名稱	描述
baselineData	完成測試執行結果物件（與 的結構相同GET /scenarios/{testId}/testruns/{testRunId}）

錯誤回應

- 400 - 無效的 testId 參數
- 404 - 找不到測試案例或未設定基準
- 500 - 內部伺服器錯誤

PUT /scenarios/{testId}/baseline

說明

PUT /scenarios/{testId}/baseline 操作會指定特定測試執行做為效能比較的基準。每個案例只能設定一個基準。

請求參數

testId

- 測試案例 ID

  類型：字串

  必要：是

請求內文

名稱	描述
testRunId	要設定為基準的測試執行 ID（例如，2DEwHItEne）

回應

200 - 成功

名稱	描述
message	確認訊息（例如 Baseline set successfully）
testId	測試案例 ID（例如 seQUy12LKL）
baselineTestRunId	設定的基準測試執行 ID（例如 2DEwHItEne）

錯誤回應

- 400 - 無效的 testId 或 testRunId
- 404 - 找不到測試案例或測試執行
- 409 - 衝突：測試執行無法設定為基準（例如，測試失敗）
- 500 - 內部伺服器錯誤

DELETE /scenarios/{testId}/baseline

說明

DELETE /scenarios/{testId}/baseline 操作會將其設定為空字串，以清除案例的基準值。

請求參數

testId

- 測試案例 ID

  類型：字串

  必要：是

回應

204 - 成功

基準已成功清除（未傳回內容）

錯誤回應

- 400 - 無效的 testId
- 500 - 內部伺服器錯誤

GET /任務

說明

GET /tasks 此操作可讓您擷取執行中 Amazon Elastic Container Service (Amazon ECS) 任務的清單。

回應

名稱	描述
tasks	執行測試的任務 IDs 清單

選項/任務

說明

OPTIONS /tasks 任務操作為具有正確 CORS 回應標頭的請求提供回應。

回應

名稱	描述
taskIds	執行測試的任務 IDs 清單

GET/區域

說明

GET /regions 操作可讓您擷取在該區域中執行測試所需的區域資源資訊。

回應

名稱	描述
testId	區域 ID
ecsCloudWatchLogGroup	區域中 Amazon Fargate 任務的 Amazon CloudWatch 日誌群組名稱
region	資料表中資源存在的區域
subnetA	區域中其中一個子網路的 ID
subnetB	區域中其中一個子網路的 ID
taskCluster	區域中 AWS Fargate 叢集的名稱
taskDefinition	區域中任務定義的 ARN
taskImage	區域中任務映像的名稱
taskSecurityGroup	區域中安全群組的 ID

選項/區域

說明

OPTIONS /regions 操作為具有正確 CORS 回應標頭的請求提供回應。

回應

名稱	描述
testId	區域 ID
ecsCloudWatchLogGroup	區域中 Amazon Fargate 任務的 Amazon CloudWatch 日誌群組名稱
region	資料表中資源存在的區域

名稱	描述
subnetA	區域中其中一個子網路的 ID
subnetB	區域中其中一個子網路的 ID
taskCluster	區域中 AWS Fargate 叢集的名稱
taskDefinition	區域中任務定義的 ARN
taskImage	區域中任務映像的名稱
taskSecurityGroup	區域中安全群組的 ID

增加容器資源

若要增加負載測試可以模擬的並行虛擬使用者（並行）數量，您需要增加分配給每個 Amazon ECS 任務的 CPU 和記憶體資源。這包括建立具有較高資源限制的新任務定義修訂，然後更新解決方案的 DynamoDB 組態，以使用新的任務定義進行未來的測試執行。

建立新的任務定義修訂

請依照下列步驟，使用增加的 CPU 和記憶體資源建立新的任務定義：

1. 登入 Amazon Elastic Container Service 主控台。
2. 在左側導覽功能表中，選取任務定義。
3. 選取對應至此解決方案的任務定義旁的核取方塊。例如，[replaceable]<stackName>`-EcsTaskDefinition-*<system-generated-random-Hash>*。
4. 選擇 Create new revision (建立新的修訂)。
5. 在建立新的修訂頁面上，採取下列動作：
   a. 在任務大小下，將任務記憶體和任務 CPU 修改為所需的值。較高的值允許每個任務有更多並行虛擬使用者。
   b. 在容器定義下，檢閱硬/軟記憶體限制。如果此限制低於所需的記憶體，請選擇容器。
   c. 在編輯容器對話方塊中，移至記憶體限制，並更新硬性限制以符合或小於任務記憶體配置。
   d. 選擇更新。
6. 在建立新的修訂頁面上，選擇建立。

7. 成功建立任務定義後，請記錄完整的任務定義 ARN，包括版本編號。例如：[replaceable]<stackName>`-EcsTaskDefinition-*<system-generated-random-Hash>*：【replaceable】<system-generated-versionNumber>。

更新 DynamoDB 資料表

建立新的任務定義修訂版之後，您必須更新解決方案的 DynamoDB 資料表，以便未來的測試執行使用新的任務定義。針對您要使用更新任務定義的每個 AWS 區域重複這些步驟：

1. 導覽至 DynamoDB 主控台。
2. 在左側導覽窗格中，選取資料表下的探索項目。
3. 選取與此解決方案相關聯的 scenarios-table DynamoDB 資料表。例如，[replaceable]<stackName>`-DLTTestRunnerStorageDLTScenariosTable-*<system-generated-random-Hash>*。
4. 選取對應至您建立新任務定義修訂的區域的項目。例如，region-[replaceable]<region-name>`。
5. 在項目編輯器中，找到 taskDefinition 屬性，並使用您在上一節中記錄的完整任務定義 ARN 更新其值（包括版本編號）。
6. 選擇儲存變更。

> **Note**
>
> 更新的任務定義只會用於新的測試執行。目前正在執行或排程的任何測試都會繼續使用先前的任務定義。

MCP 工具規格

分散式負載測試解決方案公開了一組 MCP 工具，可讓 AI 代理器與測試案例和結果互動。這些工具提供高階的抽象功能，符合 AI 代理器處理資訊的方式，讓他們能夠專注於分析和洞見，而不是詳細的 API 合約。

> **Note**
>
> 所有 MCP 工具都提供對解決方案資料的唯讀存取。不支援透過 MCP 介面修改測試案例或組態。

list_scenarios

說明

此`list_scenarios`工具會擷取所有可用測試案例的清單，其中包含基本中繼資料。

Endpoint

```
GET /scenarios
```

Parameters

無

回應

名稱	描述
`testId`	測試案例的唯一識別符
`testName`	測試案例的名稱
`status`	測試案例的目前狀態
`startTime`	測試建立或上次執行的時間
`testDescription`	測試案例的描述

get_scenario_details

說明

此`get_scenario_details`工具會擷取單一測試案例的測試組態和最新的測試執行。

Endpoint

```
GET /scenarios/<test_id>?history=false&results=false
```

請求參數

`test_id`

- 測試案例的唯一識別符

  類型：字串

  必要：是

回應

名稱	描述
`testTaskConfigs`	每個區域的任務組態
`testScenario`	測試定義和參數
`status`	目前的測試狀態
`startTime`	測試開始時間戳記
`endTime`	測試結束時間戳記（如果已完成）

list_test_runs

說明

此`list_test_runs`工具會擷取特定測試案例的測試執行清單，將最新到最舊排序。傳回最多 30 個結果。

Endpoint

```
GET /scenarios/<testid>/testruns/?limit=<limit>
```

或

```
GET /scenarios/<testid>/testruns/?
limit=30&start_date=<start_date>&end_date=<end_date>
```

請求參數

`test_id`

- 測試案例的唯一識別符

  類型：字串

  必要：是

`limit`

- 要傳回的測試執行數目上限

  類型：整數

  預設：20

  上限：30

  必要：否

`start_date`

- 篩選從特定日期執行的 ISO 8601 時間戳記

  類型：字串 (日期時間格式)

  必要：否

`end_date`

- 篩選執行至特定日期的 ISO 8601 時間戳記

  類型：字串 (日期時間格式)

  必要：否

回應

名稱	描述
testRuns	具有每次執行效能指標和百分位數的測試執行摘要陣列

get_test_run

說明

此get_test_run工具會擷取具有區域和端點明細的單一測試執行的詳細結果。

Endpoint

```
GET /scenarios/<testid>/testruns/<testrunid>
```

請求參數

test_id

- 測試案例的唯一識別符

  類型：字串

  必要：是

test_run_id

- 特定測試執行的唯一識別符

  類型：字串

  必要：是

回應

名稱	描述
results	完整的測試執行資料，包括區域結果明細、端點特定指標、效能百分位數 (p50、p90、

名稱	描述
	p95、p99)、成功和失敗計數、回應時間和延遲，以及用於執行的測試組態

get_latest_test_run

說明

此`get_latest_test_run`工具會擷取特定測試案例的最新測試執行。

Endpoint

```
GET /scenarios/<testid>/testruns/?limit=1
```

> **Note**
>
> 結果會使用全域次要索引 (GSI) 依時間排序，確保傳回最新的測試執行。

請求參數

`test_id`

- 測試案例的唯一識別符

  類型：字串

  必要：是

回應

名稱	描述
`results`	與 具有相同格式的最新測試執行資料 `get_test_run`

get_baseline_test_run

說明

此get_baseline_test_run工具會擷取特定測試案例的基準測試執行。基準用於效能比較目的。

Endpoint

```
GET /scenarios/<test_id>/baseline
```

請求參數

test_id

- 測試案例的唯一識別符

  類型：字串

  必要：是

回應

名稱	描述
baselineData	用於比較的基準測試執行資料，包括來自指定基準執行的所有指標和組態

get_test_run_artifacts

說明

此get_test_run_artifacts工具會擷取 Amazon S3 儲存貯體資訊，以存取測試成品，包括日誌、錯誤檔案和結果。

Endpoint

```
GET /scenarios/<testid>/testruns/<testrunid>
```

請求參數

`test_id`

- 測試案例的唯一識別符

  類型：字串

  必要：是

`test_run_id`

- 特定測試執行的唯一識別符

  類型：字串

  必要：是

回應

名稱	描述
`bucketName`	存放成品的 S3 儲存貯體名稱
`testRunPath`	目前成品儲存的路徑字首 (4.0+ 版)
`testScenarioPath`	舊版成品儲存的路徑字首 (4.0 版前)

> **Note**
>
> 所有 MCP 工具都會利用現有的 API 端點。不需要修改基礎 APIs 即可支援 MCP 功能。

參考資料

本節包含資料收集的相關資訊、相關資源的指標，以及有助於此解決方案的建置器清單。

資料收集

此解決方案會將有關使用此解決方案的操作指標傳送給 AWS (「資料」)。我們使用此資料來更好地了解客戶如何使用此解決方案和相關的服務和產品。AWS 收集此資料受 AWS 隱私權聲明約束。

貢獻者

- Tom Nightingale
- Fernando Dingler
- Beomseok Lee
- George Lenz
- Erin McGill
- Dimitri Lopez
- Kamyar Ziabari
- Bassem Wanis
- Garvit Singh
- Nikhil Reddy
- Simon Kroll
- Ahern Knox
- Ian Downard
- 歐文布雷迪
- Jim Thario
- Thyag Ramachandran
- 長春
- 詹姆士王

詞彙表

此詞彙表定義了 AWS 實作指南上分散式負載測試中使用的縮寫和縮寫。

技術通訊協定和格式

AGPL

Affero 一般公有授權。K6 使用的開放原始碼軟體授權。

API

應用程式程式設計界面。一組通訊協定和工具，用於建置軟體應用程式並啟用不同系統之間的通訊。

CLI

命令列界面。文字型界面，用於與軟體和作業系統互動。

CORS

跨來源資源共用。一種安全功能，允許或限制在某個原始伺服器執行的 Web 應用程式存取來自不同原始伺服器的資源。

CSV

逗號分隔值。用於以純文字存放表格式資料的檔案格式，通常用於資料匯出。

gRPC

gRPC 遠端程序呼叫。適用於遠端程序呼叫的高效能開放原始碼架構。

HTTP

超文字傳輸通訊協定。用於在全球資訊網傳輸資料的基礎通訊協定。

HTTPS

HTTP 安全。HTTP 的延伸，使用加密透過網路進行安全通訊。

JSON

JavaScript 物件標記法。輕量型資料交換格式，方便人類讀取和寫入，也可輕鬆讓機器剖析和產生。

JWT

JSON Web 權杖。一種簡潔且 URL 安全的方法，代表要在雙方之間傳輸以進行身分驗證和授權的宣告。

OAuth

開啟授權。存取委派的開放標準，通常用於字符型身分驗證和授權。

REST

表示式狀態轉移。使用無狀態通訊和標準 HTTP 方法設計聯網應用程式的架構樣式。

SSE

伺服器傳送事件。伺服器推送技術可讓用戶端透過 HTTP 連線接收來自伺服器的自動更新。

UI

使用者介面。使用者與軟體應用程式互動的視覺化元素和控制項。

URL

統一資源定位器。用來存取網際網路資源的地址。

XML

可擴展標記語言。標記語言，定義以人類可讀和機器可讀的格式編碼文件的規則。

測試和資料庫條款

FTP

檔案傳輸通訊協定。用於在用戶端和伺服器之間傳輸檔案的標準網路通訊協定。

GSI

全域次要索引。DynamoDB 功能可讓您使用備用金鑰查詢資料。

JDBC

Java 資料庫連線。用於使用資料庫連線和執行查詢的 Java API。

JMS

Java 訊息服務。用於在兩個或多個用戶端之間傳送訊息的 Java API。

TPS

每秒交易數。衡量系統在一秒內可處理的交易數量。

AWS 和系統條款

ARN

Amazon Resource Name (Amazon 資源名稱)。用於跨 AWS 服務指定資源的 AWS 資源的唯一識別符。

ISO

國際標準化組織。開發國際標準的獨立、非政府組織。本指南中參考的 ISO 8601 時間戳記格式。

SLA

服務水準協議。服務提供者與客戶之間的承諾，定義預期的服務水準。

UUID

全域唯一識別符。用於唯一識別電腦系統中資訊的 128 位元數字。

vCPU

虛擬中央處理器。指派給虛擬機器或容器的虛擬處理器，代表實體 CPU 處理能力的一部分。

負載測試條款

concurrency

每個任務的並行虛擬使用者數量。此參數控制每個 Fargate 任務在負載測試期間產生的模擬使用者數量。

區域堆疊

部署在 AWS 區域中的 CloudFormation 堆疊，以提供多區域負載測試的測試基礎設施。

任務計數

啟動以執行測試案例的 Fargate 容器（任務）數量。產生的總負載等於任務計數乘以並行。

測試案例

已設定的負載測試，包括測試類型、目標端點、任務計數、並行、持續時間和其他參數。

修訂

請造訪 GitHub 儲存庫中的 CHANGELOG.md，以追蹤版本特定的改善和修正。

注意

客戶有責任對本文件中的資訊進行自己的獨立評定。本文件：(a) 僅供參考，(b) 代表 AWS 目前的產品產品和實務，這些產品和實務可能會有所變更，恕不另行通知，且 (c) 不會從 AWS 及其附屬公司、供應商或授權方建立任何承諾或保證。AWS 產品或服務「原樣」提供，不做任何明示或暗示的保證、表示或條件。AWS 對其客戶的責任和義務由 AWS 協議控制，本文件不屬於 AWS 與其客戶之間的任何協議，也不會對其進行修改。

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。