開發人員指南

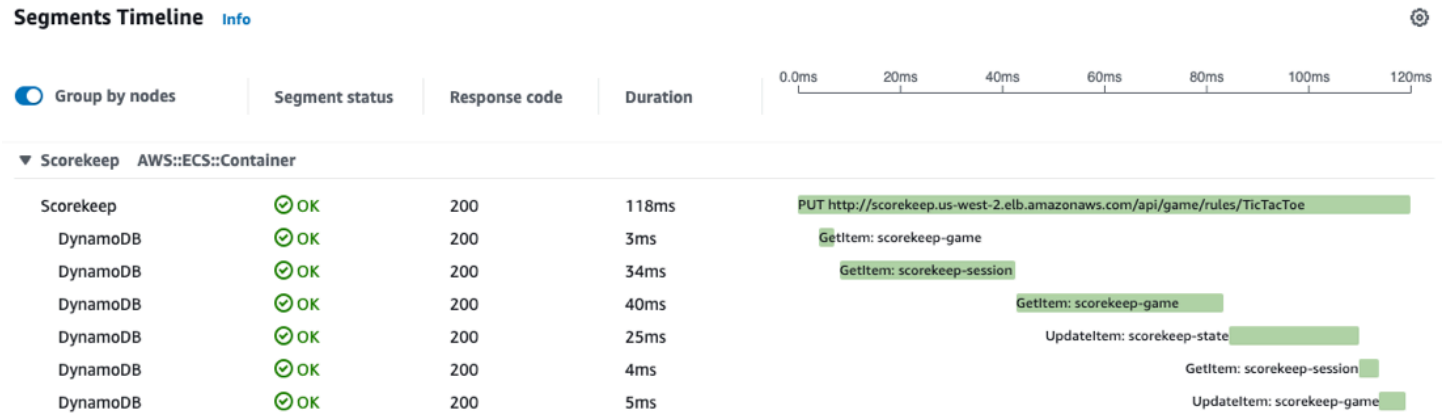
AWS X-Ray

AWS X-Ray: 開發人員指南

Table of Contents

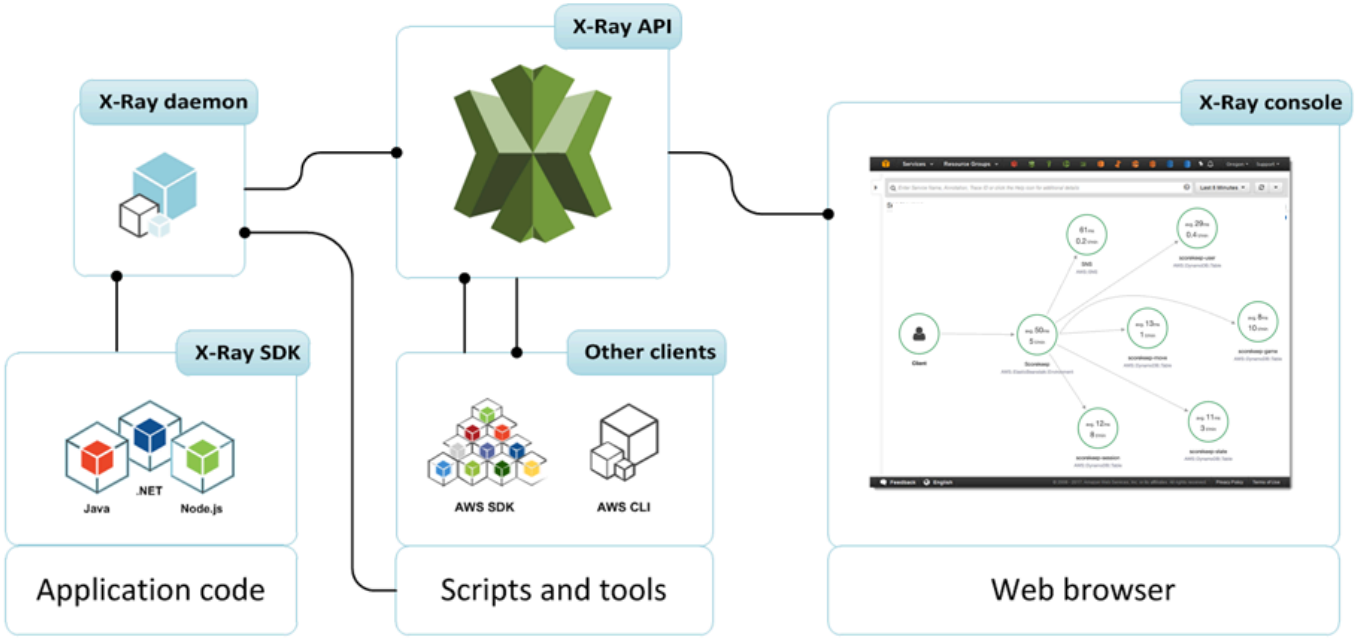
什麼是 AWS X-Ray？

AWS X-Ray 是一項服務，可收集應用程式所提供之要求的相關資料，並提供工具，供您檢視、篩選及深入瞭解該資料，以識別要進行最佳化的問題和機會。對於應用程式的任何追蹤要求，您不僅可以查看要求和回應的詳細資訊，還可以查看應用程式對下游 AWS 資源、微服務、資料庫和 Web API 進行呼叫的詳細資訊。



AWS X-Ray 除了已與 X-Ray 整合的應用程式 AWS 服務 用途外，還會接收來自應用程式的追蹤。檢測您的應用程式需要傳送追蹤資料，以及應用程式內的傳入和輸出要求以及其他事件，以及每個要求的中繼資料。許多檢測案例僅需要組態變更。例如，您可以檢測 Java 應用程式所發出的所有傳入 HTTP 要求和下游呼叫。 AWS 服務 有數個 SDK、代理程式和工具可用來檢測您的應用程式以進行 X-Ray 追蹤。如需詳細資訊，請參閱檢測您的應用程式。

AWS 服務 與 X-Ray 整合的可將追蹤標頭新增至傳入要求、將追蹤資料傳送至 X-Ray，或執行 X-Ray 精靈。例如， AWS Lambda 可以將有關請求的追蹤資料傳送至 Lambda 函數，並在工作站上執行 X-Ray 精靈，讓使用 X-Ray SDK 變得更簡單。



每個用戶端 SDK 不會將追蹤資料直接傳送至 X-Ray，而是將 JSON 區段文件傳送至偵聽 UDP 流量的守護程序。X-Ray 守護程序緩衝隊列中的段，並將其批量上傳到 X-Ray。該守護程序可用於 Linux，視窗和 macOS，並包含在 AWS Elastic Beanstalk 和 AWS Lambda 平台上。

X-Ray 會使用來自為雲端應用程式提供動力的 AWS 資源中的追蹤資料來產生詳細的追蹤圖。追蹤對應會顯示前端服務呼叫以處理要求並保留資料的用戶端、前端服務和後端服務。使用追蹤對應來識別瓶頸、延遲尖峰和其他要解決的問題，以改善應用程式的效能。

開始使用 X-Ray

若要開始使用 AWS X-Ray：

- 啟動已經測試產生追蹤資料的範例應用程式。幾分鐘後，您就可以啟動範例應用程式、產生流量、將區段傳送至 X-Ray，以及在 AWS Management Console.
- 瞭解如何檢測您的應用程式，包括使用 X-Ray SDK 或發行 AWS 版將追蹤資料傳送 OpenTelemetry 至 X-Ray。
- 探索與 X-Ray 整合AWS 服務的其他項目，包括取樣和新增標頭至傳入的要求、執行 X-Ray 精靈，以及自動將追蹤資料傳送至 X-Ray。
- 使用 X-Ray API，該 API 可通過 AWS SDK 或直接通過 HTTPS 訪問所有 X-Ray 功能。 AWS Command Line Interface

AWS X-Ray 概念

AWS X-Ray 以區段形式接收來自服務的資料。然後，X-Ray 會將具有共同要求的區段群組為追蹤。X-Ray 會處理追蹤，以產生提供應用程式視覺化呈現的服務圖表。

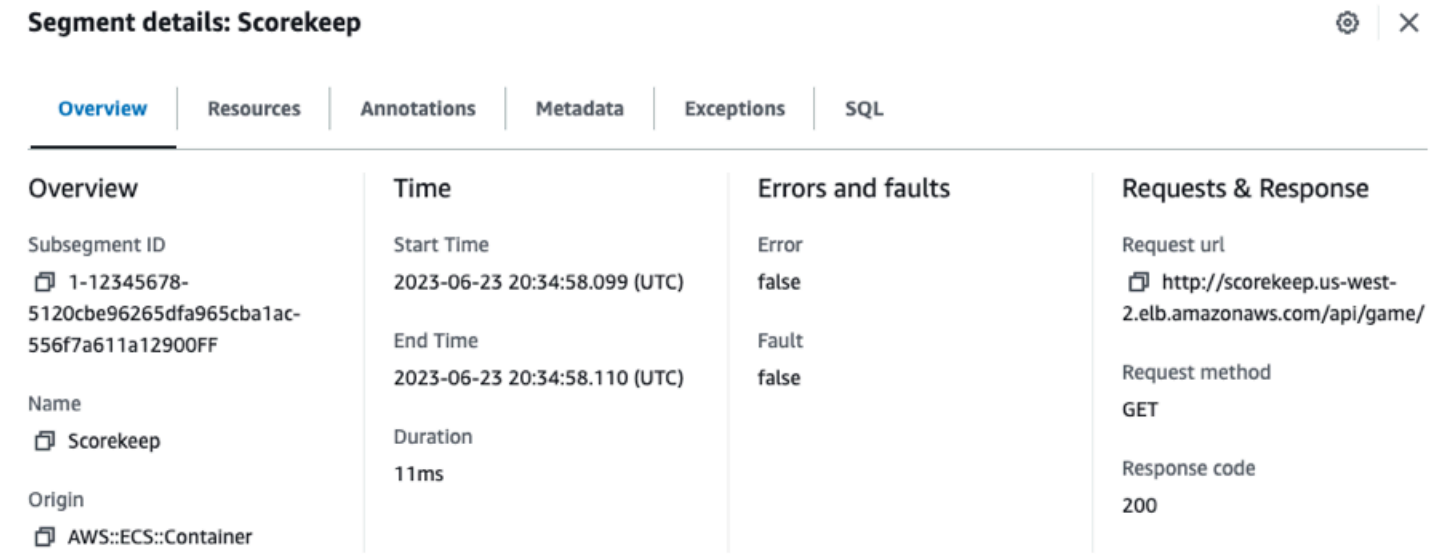
概念

- 客群
- 子區段
- 服務圖表
- 追蹤
- 抽樣
- 追蹤標頭
- 篩選條件表達式
- 群組
- 標註和中繼資料
- 錯誤、故障和例外狀況

客群

執行應用程式邏輯的運算資源會以區段形式傳送與其工作相關的資料。區段中提供資源的名稱、請求的詳細資訊，以及完成的工作詳細資訊。例如，當 HTTP 請求到達您的應用程式時，它可以記錄以下相關資料：

- 主機 — 主機名稱、別名或 IP 位址
- 請求-方法，客戶端地址，路徑，用戶代理
- 響應 — 狀態，內容
- 完成的工作-開始和結束時間，子段
- 發生的問題 — 錯誤、錯誤和例外狀況，包括自動擷取例外狀況堆疊。



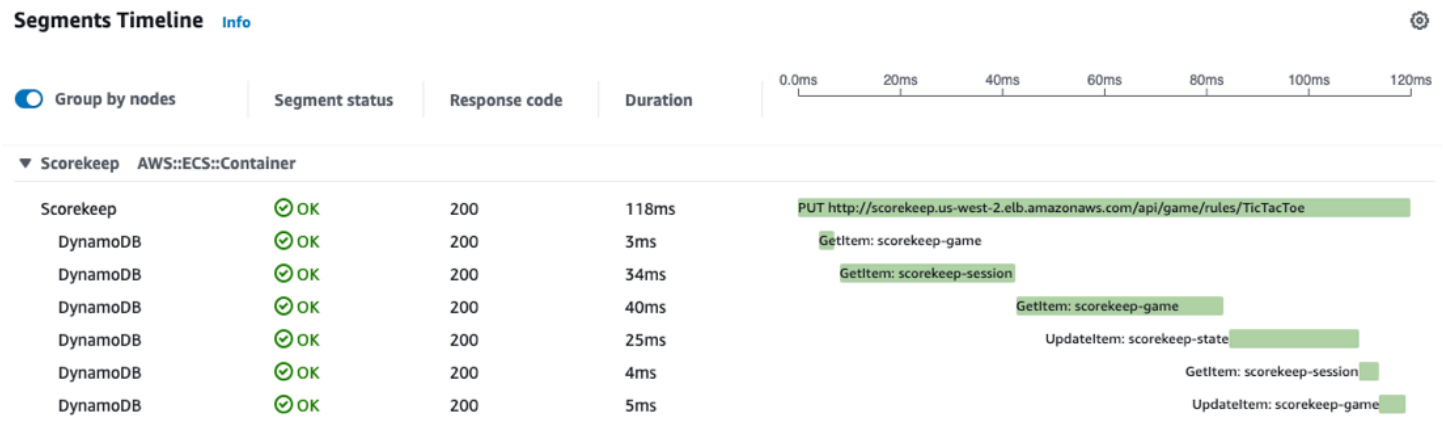
X-Ray SDK 會從要求和回應標頭、應用程式中的程式碼，以及執行其所在 AWS 資源的中繼資料收集資訊。您可以透過修改應用程式設定或程式碼來檢測傳入的要求、下游要求和 AWS SDK 用戶端，以選擇要收集的資料。

> **轉寄的要求**
>
> 如果負載平衡器或其他中介機構將要求轉寄至您的應用程式，X-Ray 會從要求中的`X-Forwarded-For`標頭取得用戶端 IP，而不是從 IP 封包中的來源 IP 取得。為轉寄要求所記錄的用戶端 IP 可以偽造，因此不應該受信任。

您可以使用 X-Ray SDK 來記錄其他資訊，例如註釋和中繼資料。如需結構詳細資訊與區段和子區段中所記錄的資訊，請參閱AWS X-Ray 區段文件。區段文件的大小最多可達 64 kB。

子區段

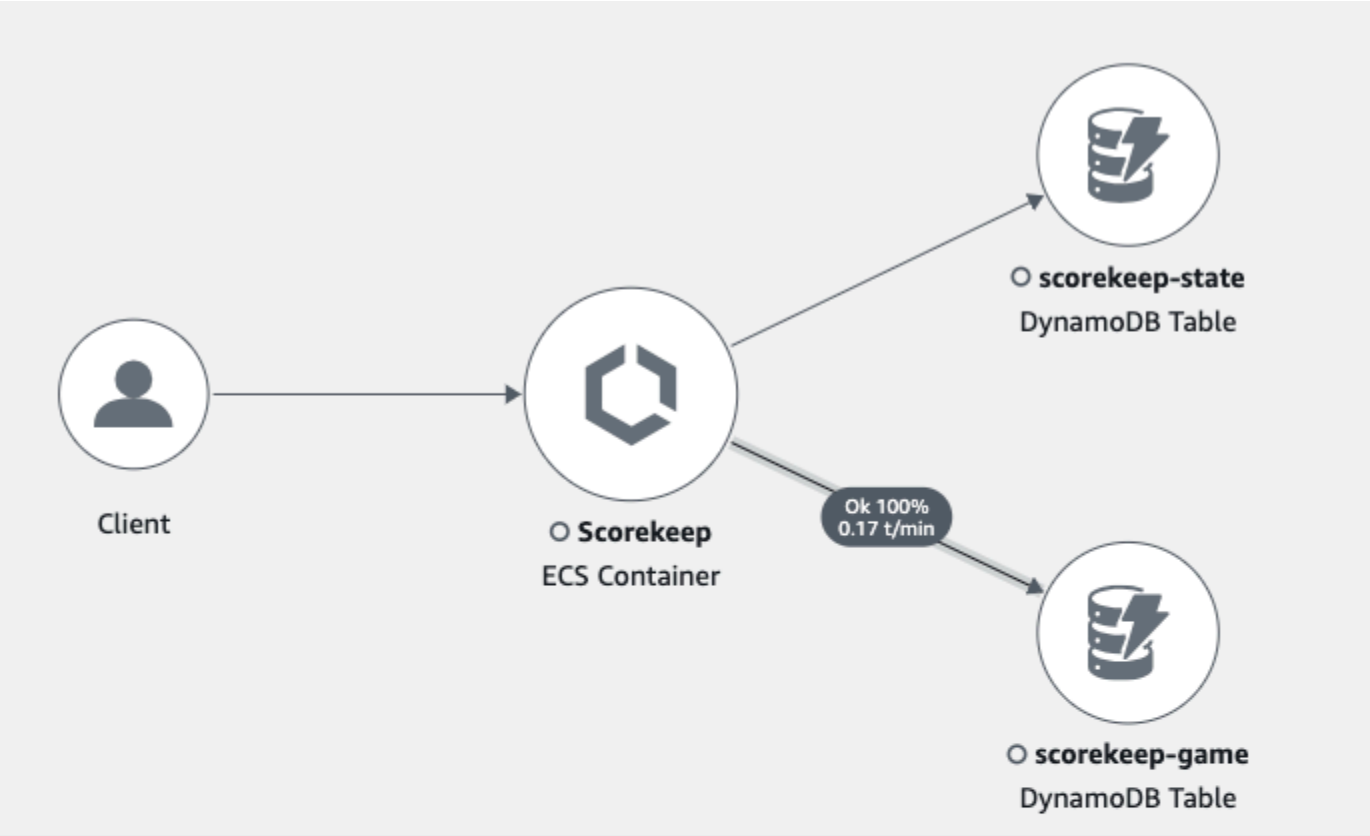
區段可以將完成工作的資料細分為子區段。子區段可提供更精確的計時資訊，以及應用程式為滿足原始請求所做的下游呼叫詳細資訊。子區段可以包含有關呼叫 AWS 服務、外部 HTTP API 或 SQL 資料庫的其他詳細資料。您甚至可以定義任意子區段來檢測應用程式的特定函數或程式碼行。

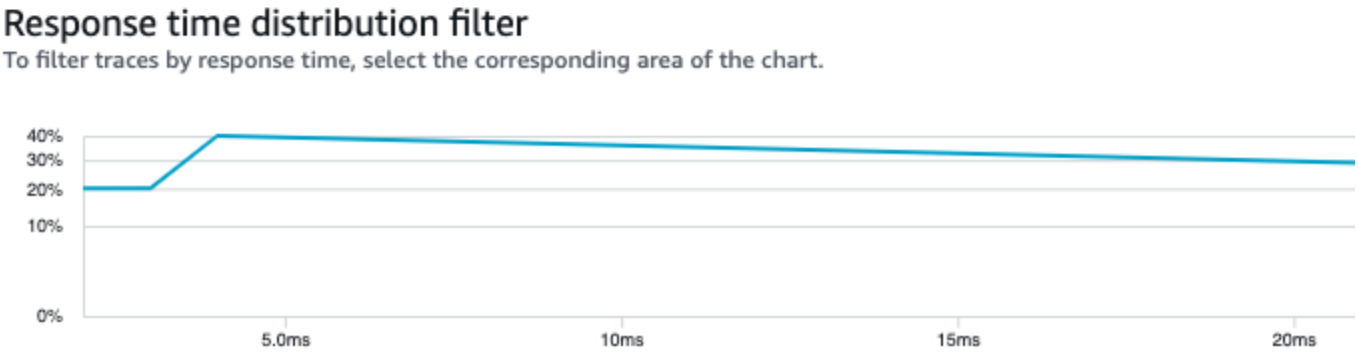


對於不傳送自己區段的服務 (例如 Amazon DynamoDB)，X-Ray 會使用子區段在追蹤對應上產生推斷的區段和下游節點。這可讓您查看所有下游相依性，即使這些相依性不支援追蹤，或屬於外部相依性亦同。

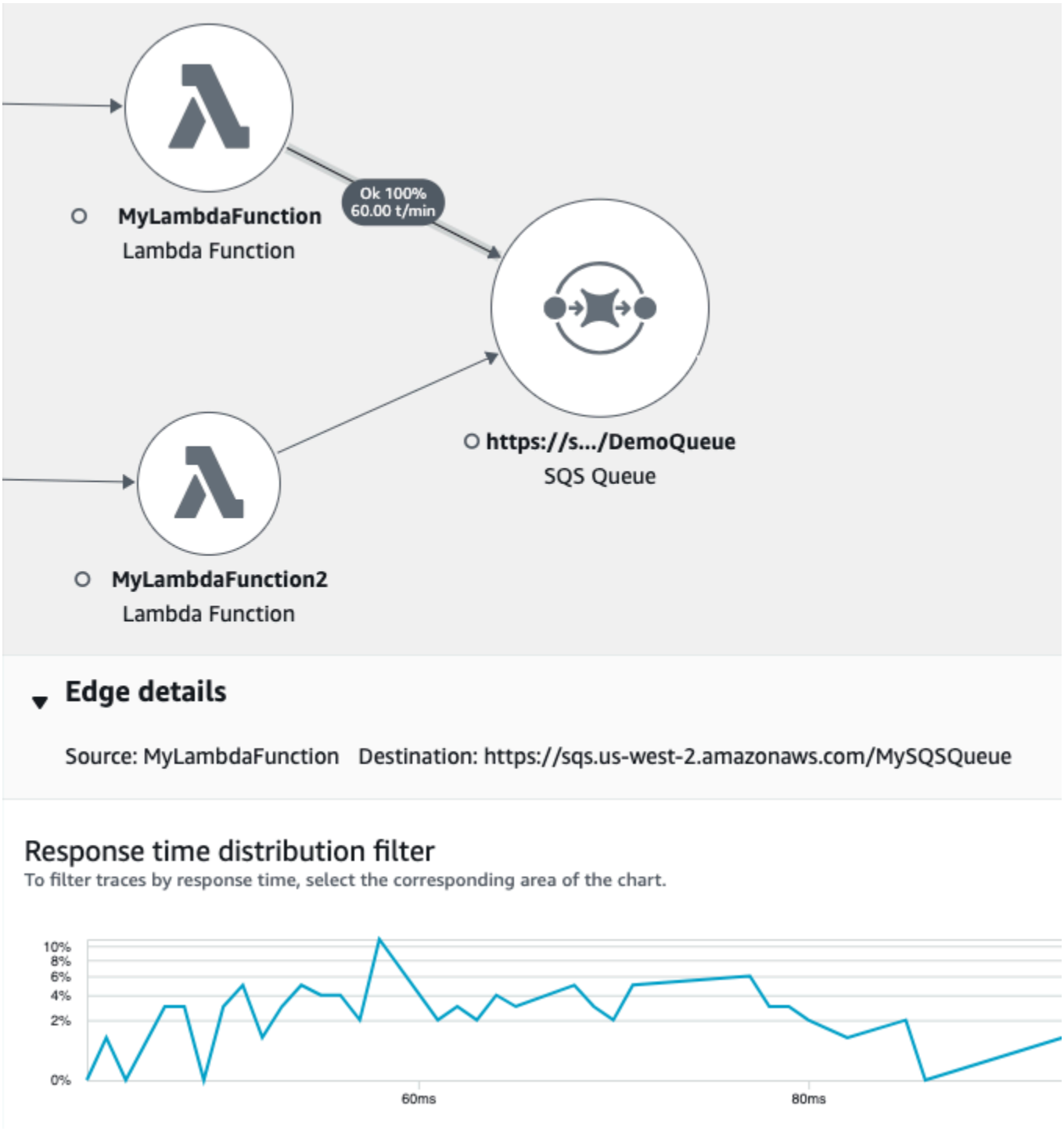
子區段會以用戶端形式來代表您應用程式的下游呼叫檢視。如果下游服務也經過檢測，其傳送的區段則會取代從上游用戶端子區段產生的推斷區段。服務圖表的節點一律使用來自服務區段的資訊；兩個節點之間的邊緣則會使用上游服務的子區段。

例如，當您使用已檢測的 AWS SDK 用戶端呼叫 DynamoDB 時，X-Ray SDK 會記錄該呼叫的子區段。DynamoDB 不會傳送區段，因此追蹤中推斷的區段、服務圖表上的 DynamoDB 節點，以及服務與 DynamoDB 之間的邊緣都包含來自子區段的資訊。





當您使用經檢測的應用程式呼叫其他經檢測的服務時，下游服務會傳送自己的區段以記錄上游服務在子區段中記錄的相同呼叫檢視。在服務圖表中，兩種服務的節點都會包含來自這些服務區段的計時和錯誤資訊，而之間的邊緣則包含來自上游服務子區段的資訊。



這兩種檢視都很實用，因為下游服務會精確記錄請求的工作開始和結束時間，而上游服務會記錄請求的往返延遲，包括請求在這兩項服務之間移動所花的時間。

服務圖表

X-Ray 會使用應用程式傳送的資料來產生服務圖表。將 AWS 資料傳送至 X-Ray 的每個資源都會顯示為圖形中的一項服務。邊緣可連線至各種服務，這些服務則協力為請求提供服務。邊緣會將用戶端連線至您的應用程式，並將您的應用程式連線至其所用的下游服務和資源。

> **服務名稱**
>
> 區段name應與產生區段之服務的網域名稱或邏輯名稱相符。但是，這不是強制執行的。具有PutTraceSegments可傳送任何名稱之區段之權限的任何應用程式。

服務圖表是一種 JSON 文件，其中包含構成您應用程式的服務和資源相關資訊。X-Ray 主控台使用服務圖表產生視覺效果或服務對應。

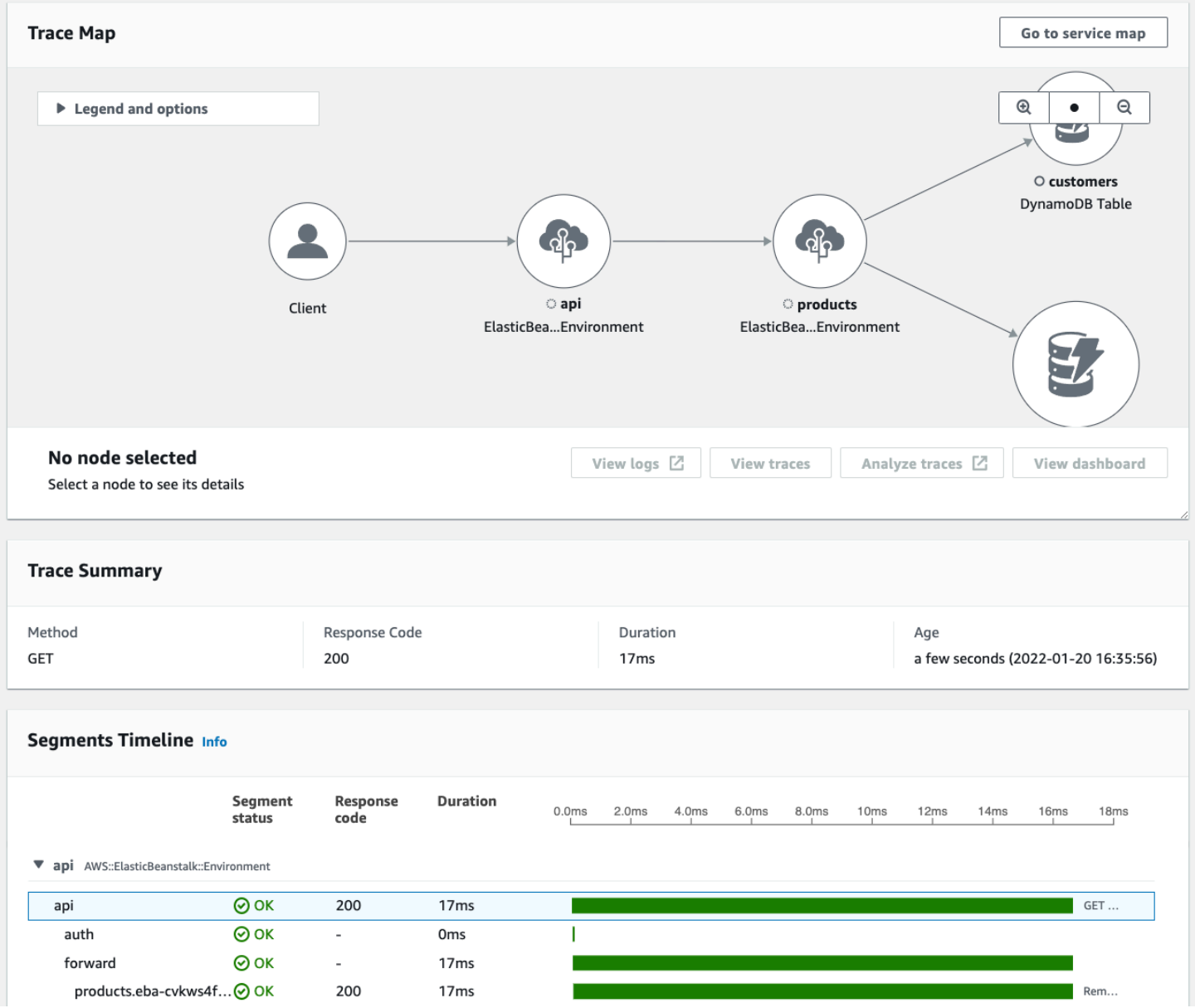
對於分散式應用程式，X-Ray 會將處理具有相同追蹤識別碼之要求的所有服務節點合併為單一服務圖表。請求命中的第一個服務會新增追蹤標頭，而系統會在前端和其呼叫的服務之間傳播此標頭。

例如，Scorekeep 會執行呼叫微服務的 Web API (AWS Lambda 函數)，以使用 Node.js 程式庫來產生隨機名稱。適用於 Java 的 X-Ray 開發套件會產生追蹤識別碼，並將其包含在呼叫 Lambda 中。Lambda 會傳送追蹤資料，並將追蹤識別碼傳遞給函數。Node.js 的 X-Ray SDK 也會使用追蹤識別碼來傳送資料。因此，API、Lambda 服務和 Lambda 函數的節點都會在追蹤對應上顯示為獨立但已連接的節點。

服務圖表資料會保留 30 天。

追蹤

追蹤 ID 可追蹤透過應用程式的請求路徑。追蹤會收集由單一請求所產生的所有區段。該要求通常是 HTTP GET 或 POST 要求，會透過負載平衡器傳送、命中您的應用程式程式碼，並產生對其他 AWS 服務或外部 Web API 的下游呼叫。第一個與 HTTP 請求互動的支援服務會為請求新增追蹤 ID，並向下游傳播以追蹤延遲、處理和其他請求資料。



如需 X-Ray 追蹤計費方式的相關資訊，請參閱AWS X-Ray 定價。追蹤資料會保留 30 天。

抽樣

為了確保有效率的追蹤，並提供應用程式所提供之請求的代表性樣本，X-Ray SDK 會套用取樣演算法來決定要追蹤哪些要求。根據預設，X-Ray SDK 會每秒記錄第一個要求，以及任何其他要求的百分之五。

為了避免開始使用時產生的服務費用，預設的抽樣費率都很保守。您可以設定 X-Ray 來修改預設取樣規則，並設定根據服務或要求內容套用取樣的其他規則。

例如，您可能想要停用取樣，並追蹤所有要求，以修改狀態或處理使用者或交易的呼叫。若是大量唯讀呼叫 (例如背景輪詢、運作狀態檢查或連線維護)，您可以用低費率抽樣但仍獲得足夠的資料，以查看發生的任何問題。

如需詳細資訊，請參閱 設定 取樣規則。

追蹤標頭

所有請求都會追蹤，並到可設定的最低限度為止。到達該最低限度之後，就會追蹤某個百分比的請求，以避免不必要的成本。取樣決策和追蹤識別碼會新增至名為的追蹤標頭中的 HTTP 要求`X-Amzn-Trace-Id`。要求所擊中的第一個 X-ray 整合服務會新增追蹤標頭，而 X-Ray SDK 會讀取並包含在回應中。

Example 含根追蹤 ID 和抽樣決策的追蹤標頭

```
X-Amzn-Trace-Id: Root=1-5759e988-bd862e3fe1be46a994272793;Sampled=1
```

> **追蹤標頭的安全性**
>
> 追蹤標頭可以來自 X-Ray SDK AWS 服務、或用戶端要求。您的應用程式可以從傳入的請求移除 `X-Amzn-Trace-Id`，以避免使用者將追蹤 ID 或抽樣決策新增到請求時發生問題。

如果請求是從經檢測的應用程式產生，則追蹤標頭也可以包含父區段 ID。例如，如果您的應用程式使用已檢測的 HTTP 用戶端呼叫下游 HTTP Web API，X-Ray SDK 會將原始要求的區段識別碼新增至下游要求的追蹤標頭。為下游請求提供服務的經檢測應用程式，可以記錄父區段 ID 以連線這兩個請求。

Example 含根追蹤 ID、父區段 ID 和抽樣決策的追蹤標頭

```
X-Amzn-Trace-Id: Root=1-5759e988-
bd862e3fe1be46a994272793;Parent=53995c3f42cd8ad8;Sampled=1
```

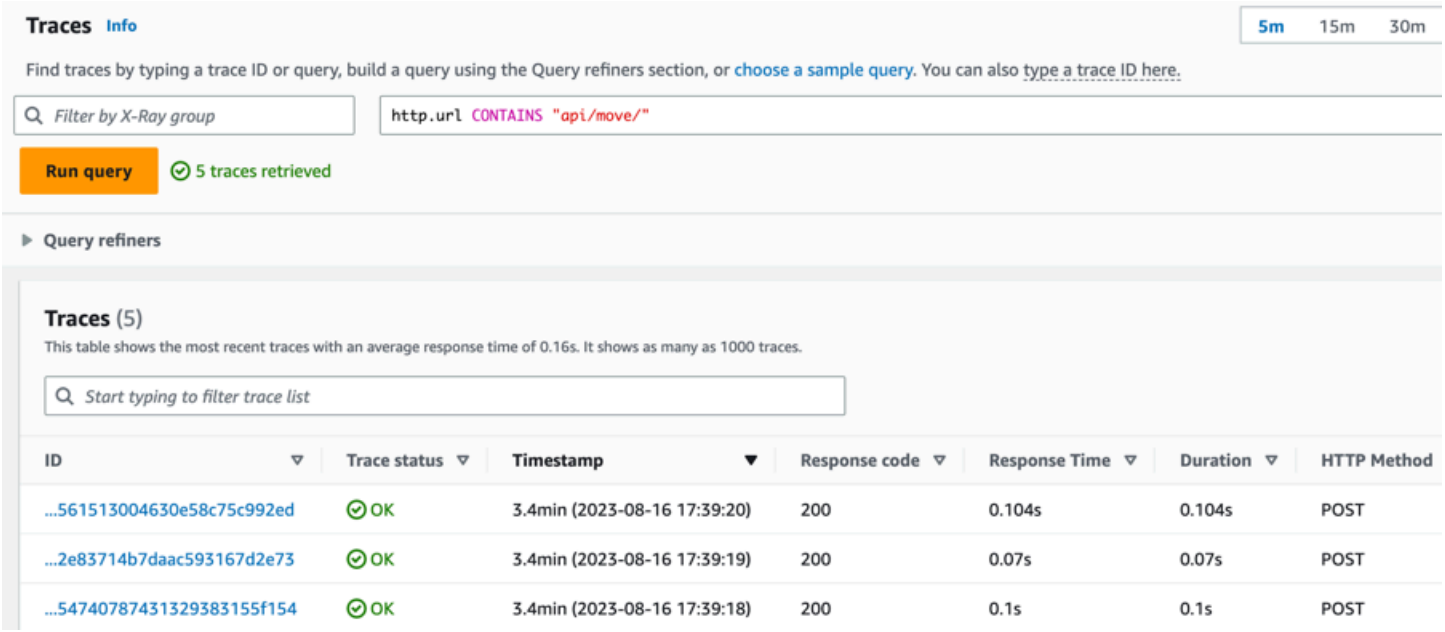
`Lineage`可以由 Lambda 和其他 AWS 服務 作為其處理機制的一部分附加到跟踪標頭，並且不應直接使用。

Example 使用歷程追蹤標頭

```
X-Amzn-Trace-Id: Root=1-5759e988-bd862e3fe1be46a994272793;Sampled=1;Lineage=a87bd80c:1|
68fd508a:5|c512fbe3:2
```

篩選條件表達式

即便使用抽樣，複雜的應用程式仍會產生大量資料。主 AWS X-Ray 控台提供服務圖形的 easy-to-navigate 檢視。它會顯示運作狀況和效能資訊，以協助您識別問題並找出最佳化應用程式的機會。若要進階追蹤，您可以向下切入以追蹤個別請求，或使用篩選條件表達式，以尋找與特定路徑或使用者相關的追蹤。



群組

擴展過濾器表達式，X-Ray 還支持組功能。使用篩選條件表達式，即可以定義條件，以接受追蹤到群組。

您可以依名稱或 Amazon 資源名稱 (ARN) 呼叫群組，以產生自己的服務圖表、追蹤摘要和 Amazon CloudWatch 指標。建立群組之後，傳入的追蹤會在儲存在 X-Ray 服務中時，根據群組的篩選器運算式進行檢查。每分鐘都會發佈符合 CloudWatch 每個條件的追蹤數量指標。

更新群組的篩選條件表達式不會變更已記錄的資料。更新僅適用於後續追蹤。這會導致圖表合併新舊表達式。若要避免這種情況，請刪除目前群組並建立新的群組。

> **Note**
>
> 群組的計費方式是根據符合篩選條件表達式的擷取追蹤。如需詳細資訊，請參閱 AWS X-Ray 定價。

如需群組的詳細資訊，請參閱設定群組。

標註和中繼資料

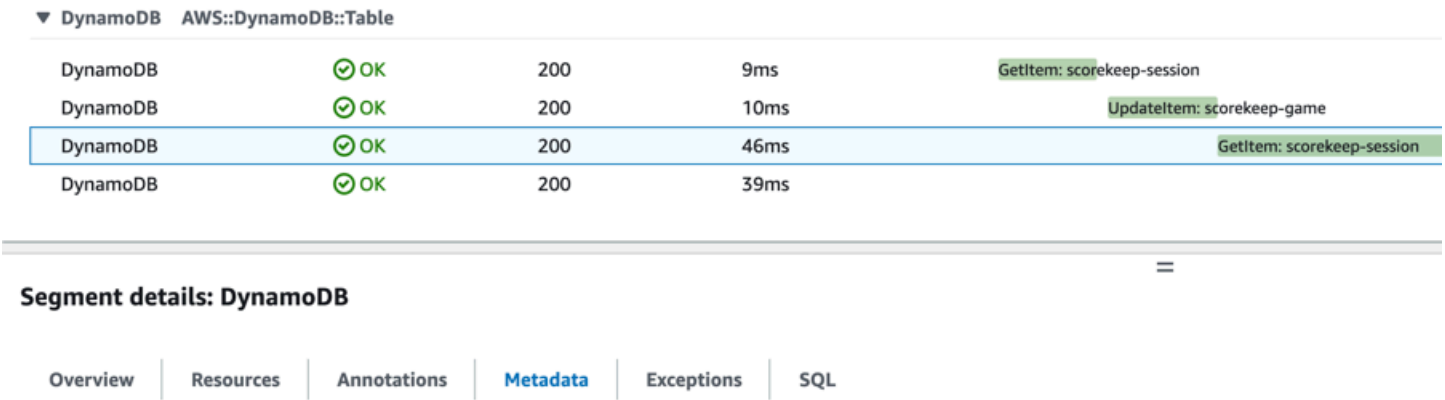
檢測應用程式時，X-Ray SDK 會記錄傳入和傳出要求、使用的 AWS 資源以及應用程式本身的相關資訊。您可以藉由註釋和中繼資料形式，將其他資訊新增至區段文件。註釋和中繼資料會在追蹤層級彙總，並且可以新增至任何區段或子區段。

註釋是簡單的鍵/值對，其會建立索引以與篩選條件表達式搭配使用。使用標記記錄您想要用來在主控台將追蹤分組的資料，或是在呼叫 `GetTraceSummaries` API 時使用標記。

X-Ray 索引每個軌跡最多 50 個註釋。

中繼資料為含有任何類型值的鍵/值對，包括物件和清單，但不會建立索引。您可以使用中繼資料，來記錄想要存放於追蹤但不需用於搜尋追蹤的資料。

您可以在 CloudWatch 主控台的「追蹤詳細資訊」頁面中的區段或子區段詳細資訊視窗中檢視註釋和中繼資料。



錯誤、故障和例外狀況

X-Ray 會追蹤應用程式程式碼中發生的錯誤，以及下游服務傳回的錯誤。錯誤分類如下。

- **`Error`**— 客戶端錯誤（400 系列錯誤）
- **`Fault`**— 伺服器故障 (500 系列錯誤)
- **`Throttle`**-節流錯誤（429 太多請求）

當您的應用程式提供已檢測的要求時發生例外狀況，X-Ray SDK 會記錄有關例外狀況的詳細資訊，包括堆疊追蹤 (如果有的話)。您可以在 X-Ray 主控台的區段詳細資料下檢視例外。

AWS X-Ray 控制台

使用 AWS X-Ray 主控台可檢視應用程式所提供之要求的服務對應和關聯追蹤，以及設定群組和取樣規則，以影響追蹤傳送至 X-Ray 的方式。

> **Note**
>
> CloudWatch 現在包括應用程序信號，它可以發現和監視您的應用程序服務，客戶端，Synthetics 金絲雀和服務依賴關係。使用 Application Signals 查看服務清單或視覺化地圖，根據您的服務等級目標 (SLO) 檢視運作狀態指標，並深入了解相關的 X-Ray 追蹤以取得更詳細的疑難排解。
>
> X-Ray 服務地圖和 CloudWatch ServiceLens 地圖已合併到 Amazon CloudWatch 控制台中的 X-Ray 跟踪地圖中。打開CloudWatch 控制台，然後從左側導航窗格中選擇 X-Ray 跟踪下的跟踪映射。

主要的 X-Ray 主控台頁面是追蹤對應，這是 X-Ray 從應用程式產生的追蹤資料產生的 JSON 服務圖形的視覺化呈現。映射由您帳戶中每個處理請求的應用程式服務節點、代表請求來源的上游用戶端節點，以及代表處理請求過程中應用程式所使用 web 服務及資源的下游服務節點組成。還有其他頁面可供檢視追蹤和追蹤詳細資訊，以及設定群組和取樣規則。

探索 X-Ray 控制台

- 使用 X-Ray 軌跡圖
- 檢視追蹤和追蹤詳細資訊
- 使用篩選運算式
- 跨帳戶追蹤
- 追蹤事件導向的應用程式
- 使用延遲直方圖
- 使用 X-Ray 洞察
- 與 Analytics 主控台互動
- 設定群組
- 設定 取樣規則
- 主控台深層連結

使用 X-Ray 軌跡圖

檢視 X-Ray 追蹤對應，以識別發生錯誤的服務、高延遲的連線，或追蹤失敗的要求。

> **Note**
>
> CloudWatch 現在包括應用程序信號，它可以發現和監視您的應用程序服務，客戶端，合成金絲雀和服務依賴關係。使用 Application Signals 查看服務清單或視覺化地圖，根據您的服務等級目標 (SLO) 檢視運作狀態指標，並深入了解相關的 X-Ray 追蹤以取得更詳細的疑難排解。X-Ray 服務地圖和 CloudWatch ServiceLens 地圖被合併到 Amazon CloudWatch 控制台內的 X-Ray 跟踪地圖中。打開CloudWatch 控制台，然後從左側導航窗格中選擇 X-Ray 跟踪下的「跟踪映射」。

檢視追蹤映射

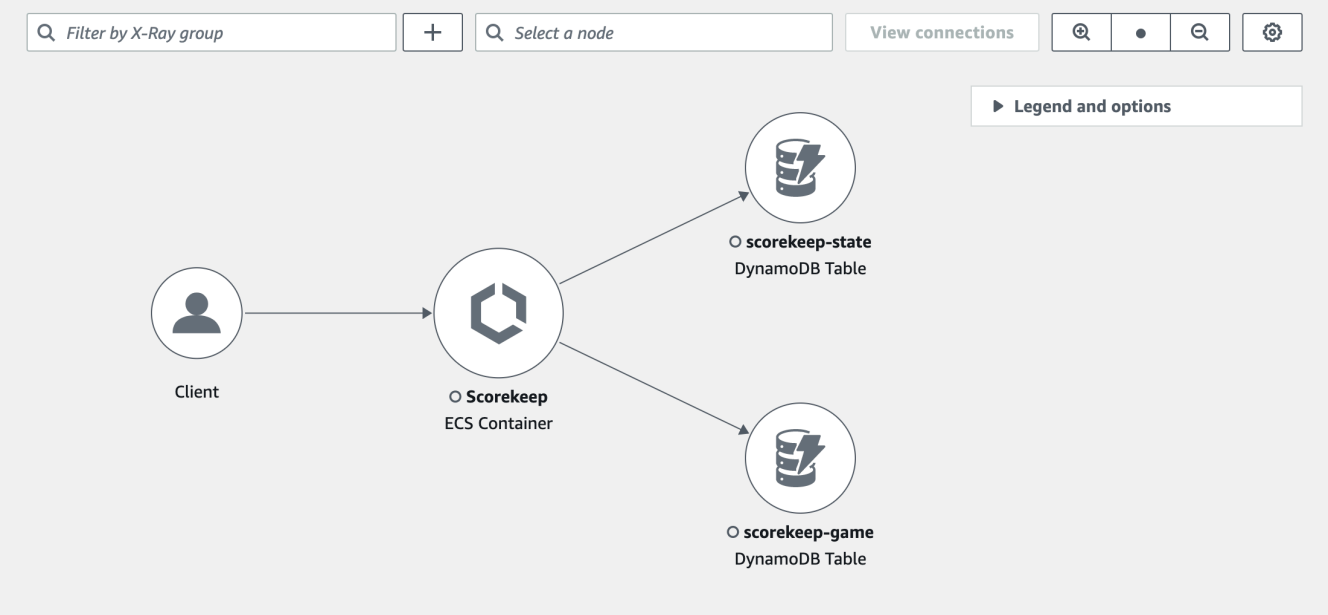
追蹤對應是應用程式所產生之追蹤資料的視覺化呈現方式。此對映會顯示提供要求的服務節點、代表要求來源的上游用戶端節點，以及代表應用程式在處理要求時所使用之 Web 服務和資源的下游服務節點。

追蹤對應顯示使用 Amazon SQS 和 Lambda 之事件導向應用程式之間的追蹤連線檢視。如需詳細資訊，請參閱追蹤事件導向應用程式。追蹤對映也支援跨帳戶追蹤，在單一對應中顯示來自多個帳戶的節點。

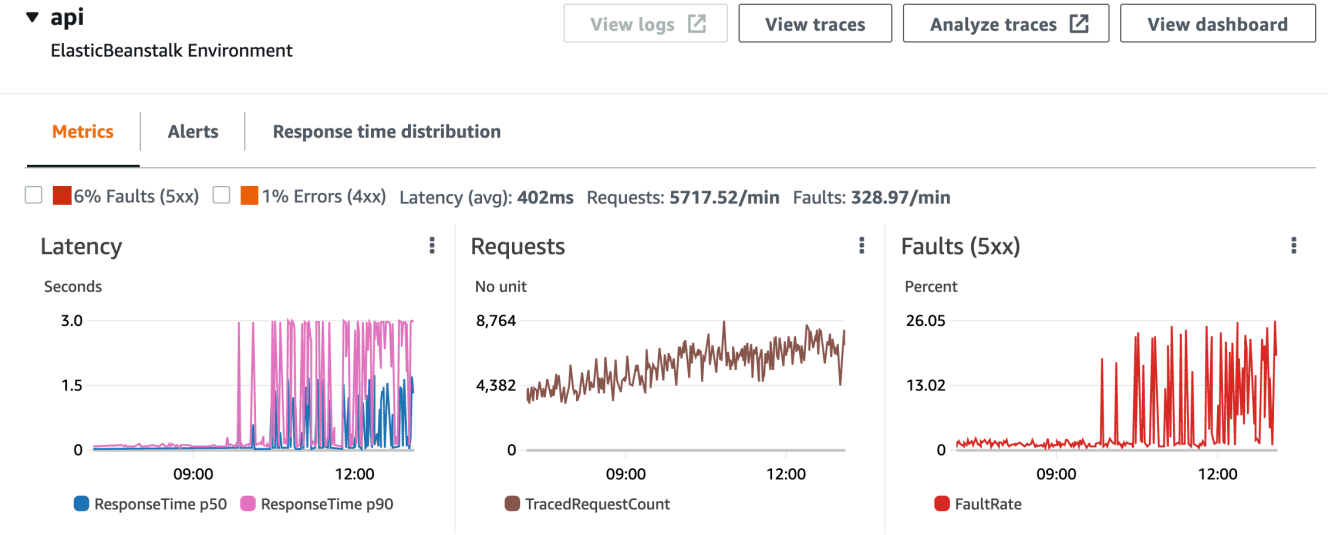
CloudWatch console

若要在 CloudWatch 主控台中檢視軌跡對應

1. 開啟 CloudWatch 主控台。在左側導覽窗格的「X-Ray 追蹤」區段下選擇「追蹤對應」。



2. 選擇服務節點來檢視該節點的請求，或是兩個節點間的邊緣來檢視在該連線上進行的請求。
3. 追蹤對映下方會顯示其他資訊，包括測量結果、警示和回應時間分佈的標籤。在「測量結果」頁籤上，選取每個圖表中的範圍以向下展開以檢視更多詳細資訊，或選擇「錯誤」或「錯誤」選項來篩選追蹤。在「回應時間分佈」標籤上，選取圖形內的範圍，以依回應時間篩選追蹤。



4. 選擇 [檢視追蹤] 來檢視追蹤，或者如果已套用篩選，請選擇 [檢視篩選的追蹤]。
5. 選擇 [檢視記錄檔] 以查看與所選節點相關聯的 CloudWatch 記錄。並非所有追蹤對應節點都支援檢視記錄檔。如需詳細資訊，請參閱疑難排解 CloudWatch 記

跟踪映射通過使用顏色概述每個節點中的問題：

- 紅色表示伺服器故障 (500 系列錯誤)

- 黃色表示用戶端錯誤 (400 系列錯誤)
- 紫色表示調節錯誤 (429 請求數太多)

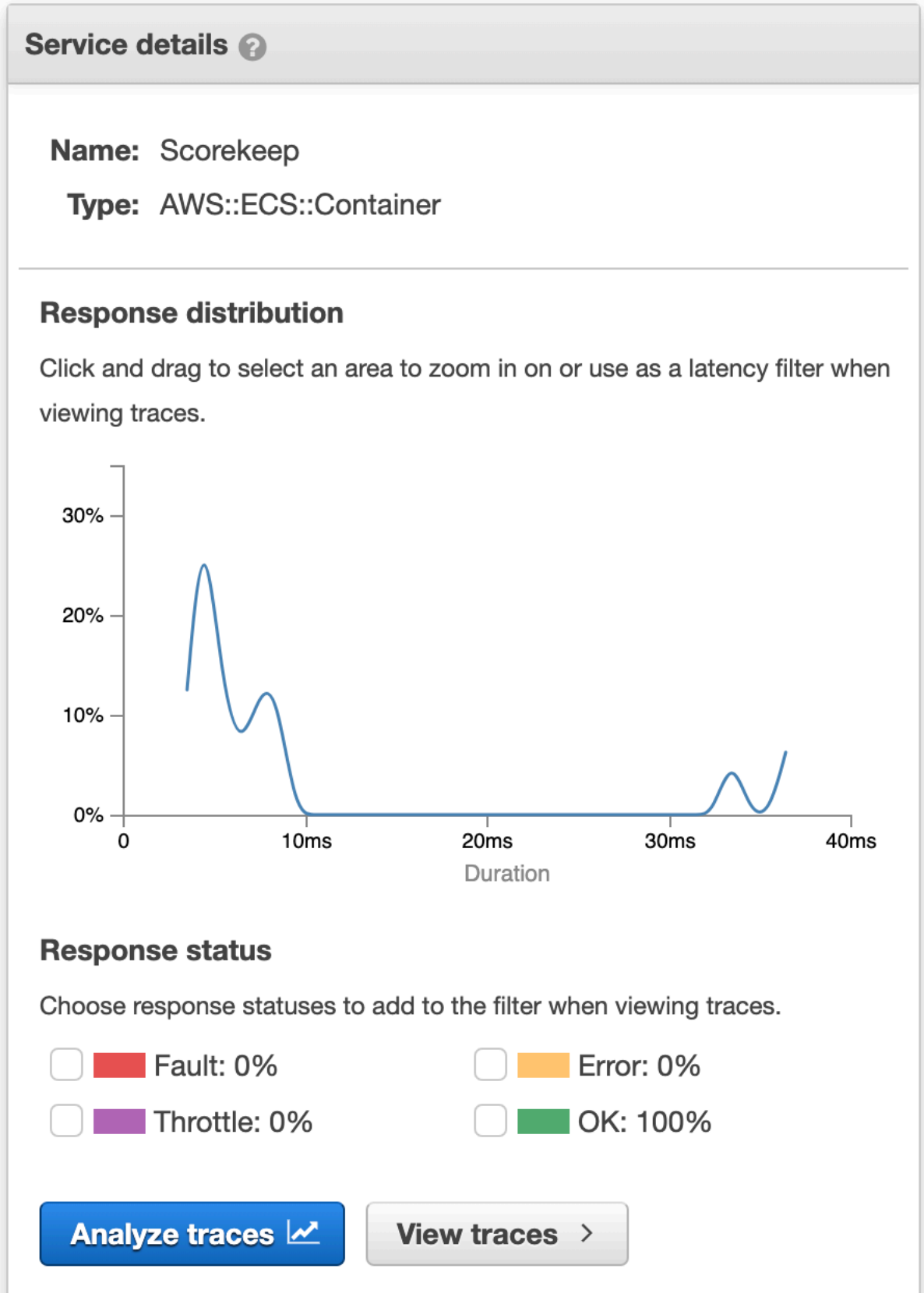
如果您的軌跡地圖很大，請使用螢幕上的控制選項或滑鼠來放大和縮小並移動地圖。

X-Ray console

若要檢視服務對應

1. 開啟 X-Ray 主控台。依預設，會顯示服務對應。您也可以從左側導覽窗格中選擇「服務對應」。

2. 選擇服務節點來檢視該節點的請求，或是兩個節點間的邊緣來檢視在該連線上進行的請求。
3. 使用回應分佈直方圖，依持續時間篩選追蹤，並選取您要檢視繪線的狀態碼。然後選擇 View traces (檢視追蹤) 來使用套用的篩選條件表達式開啟追蹤清單。


Service details
Name: Scorekeep
Type: AWS::ECS::Container
Response distribution
Click and drag to select an area to zoom in on or use as a latency filter when viewing traces.
30%
20%
10%
0%
0
10ms
20ms
30ms
40ms
Duration
Response status
Choose response statuses to add to the filter when viewing traces.
Fault: 0%
Error: 0%
Throttle: 0%
OK: 100%
Analyze traces
View traces

服務映射會透過根據成功呼叫與錯誤及故障的比例，將每個節點標上顏色，來指出每個節點的運作狀態。

- 綠色表示成功呼叫
- 紅色表示伺服器故障 (500 系列錯誤)
- 黃色表示用戶端錯誤 (400 系列錯誤)
- 紫色表示調節錯誤 (429 請求數太多)

如果您的服務對應很大，請使用螢幕上的控制選項或滑鼠來放大和縮小並移動地圖。

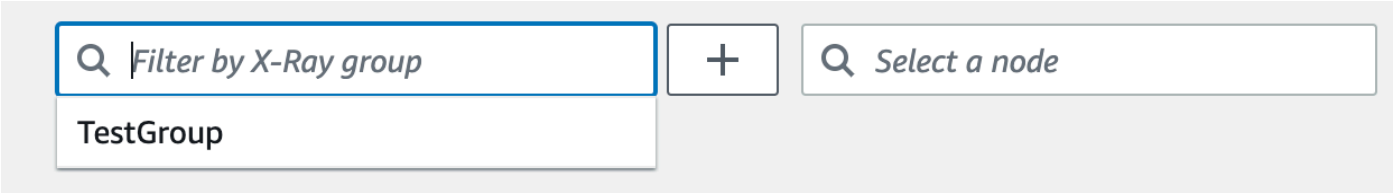
> **Note**
>
> X-Ray 追蹤圖最多可顯示 10,000 個節點。在服務節點總數超過此限制的罕見情況下，您可能會收到錯誤訊息，而且無法在主控台中顯示完整的追蹤對應。

依群組篩選軌跡圖

使用篩選器運算式，您可以定義要在群組中包含追蹤的條件。使用下列步驟，然後在軌跡對映中顯示該特定群組。

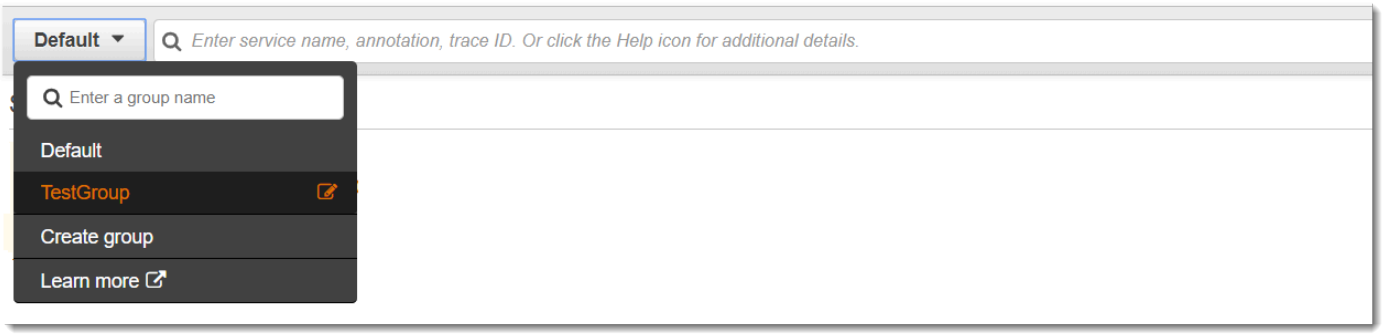
CloudWatch console

從繪圖左上角的群組篩選中選擇群組名稱。



X-Ray console

從下拉式功能表左側的搜尋列，選擇群組名稱。



現在將篩選服務對應，以顯示符合所選群組之篩選器運算式的追蹤。

追蹤地圖圖例和選項

軌跡貼圖包括一個圖例和數個用於自訂地圖顯示的選項。

CloudWatch console

選擇地圖右上方的「圖例」和「選項」下拉式選單。選擇節點內顯示的內容，包括：

- 測量結果顯示所選時間範圍內的平均回應時間和每分鐘傳送的追蹤數目。
- 節點會在每個節點內顯示服務圖示。

從「偏好設定」面板中選擇其他地圖設定，您可以透過地圖右上方的齒輪圖示進行存取。這些設定包括選取用於決定每個節點大小的測量結果，以及應在地圖上顯示哪些金絲雀。

X-Ray console

選擇地圖右上方的地圖圖例連結，即可顯示服務對應圖例。您可以在追蹤對應的右下方選擇服務對應選項，包括：

- 「服務圖示」可切換每個節點內顯示的內容，並顯示服務圖示，或者在所選時間範圍內每分鐘傳送的平均回應時間和追蹤數目。
- 調整節點大小：「無」 會將所有節點設定為相同大小。
- 節點大小調整：Health 全狀況會依受影響要求的數目來調整節點大小，包括錯誤、錯誤或限制的要求。
- 節點大小：流量按請求總數調整節點的大小。

檢視追蹤和追蹤詳細資訊

您可以使用 X-Ray 主控台中的「追蹤」頁面，依 URL、回應代碼或追蹤摘要中的其他資料尋找追蹤。從追蹤清單選取追蹤之後，「追蹤詳細資訊」頁面會顯示與所選追蹤相關聯的服務節點對應，以及追蹤區段的時間表。

檢視追蹤

CloudWatch console

若要在 CloudWatch 主控台中檢視追蹤

1. 請登入 AWS Management Console 並開啟 CloudWatch 主控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在左側導覽窗格中，選擇「X-Ray 繪線」，然後選擇「繪圖」。您可以依群組篩選或輸入篩選運算式。這會篩選頁面底部「追蹤」區段中顯示的追蹤。

   或者，您可以使用服務對應導覽至特定的服務節點，然後檢視追蹤。這會開啟已套用查詢的 [追蹤] 頁面。
3. 在「查詢細化器」區段中精簡您的查詢。若要依共用屬性篩選追蹤，請從「精簡查詢依據」旁的向下箭頭選擇選項。選項包括下列項目：

   - 節點 — 依服務節點篩選追蹤。
   - 資源 ARN — 依與追蹤相關聯的資源篩選追蹤。這些資源的範例包括 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體、 AWS Lambda 函數或 Amazon DynamoDB 表格。
   - 使用者 — 使用使用者 ID 篩選追蹤。
   - 錯誤根本原因訊息 — 依錯誤根本原因篩選追蹤。
   - URL — 依應用程式使用的 URL 路徑篩選追蹤。
   - HTTP 狀態碼 — 依應用程式傳回的 HTTP 狀態碼篩選追蹤。您可以指定自訂回應碼，或從下列選項中選取：
     - 200— 請求已成功。
     - 401-請求缺少有效的身份驗證憑據。
     - 403-請求缺少有效的權限。
     - 404— 伺服器找不到要求的資源。
     - 500— 伺服器遇到非預期的情況並產生內部錯誤。

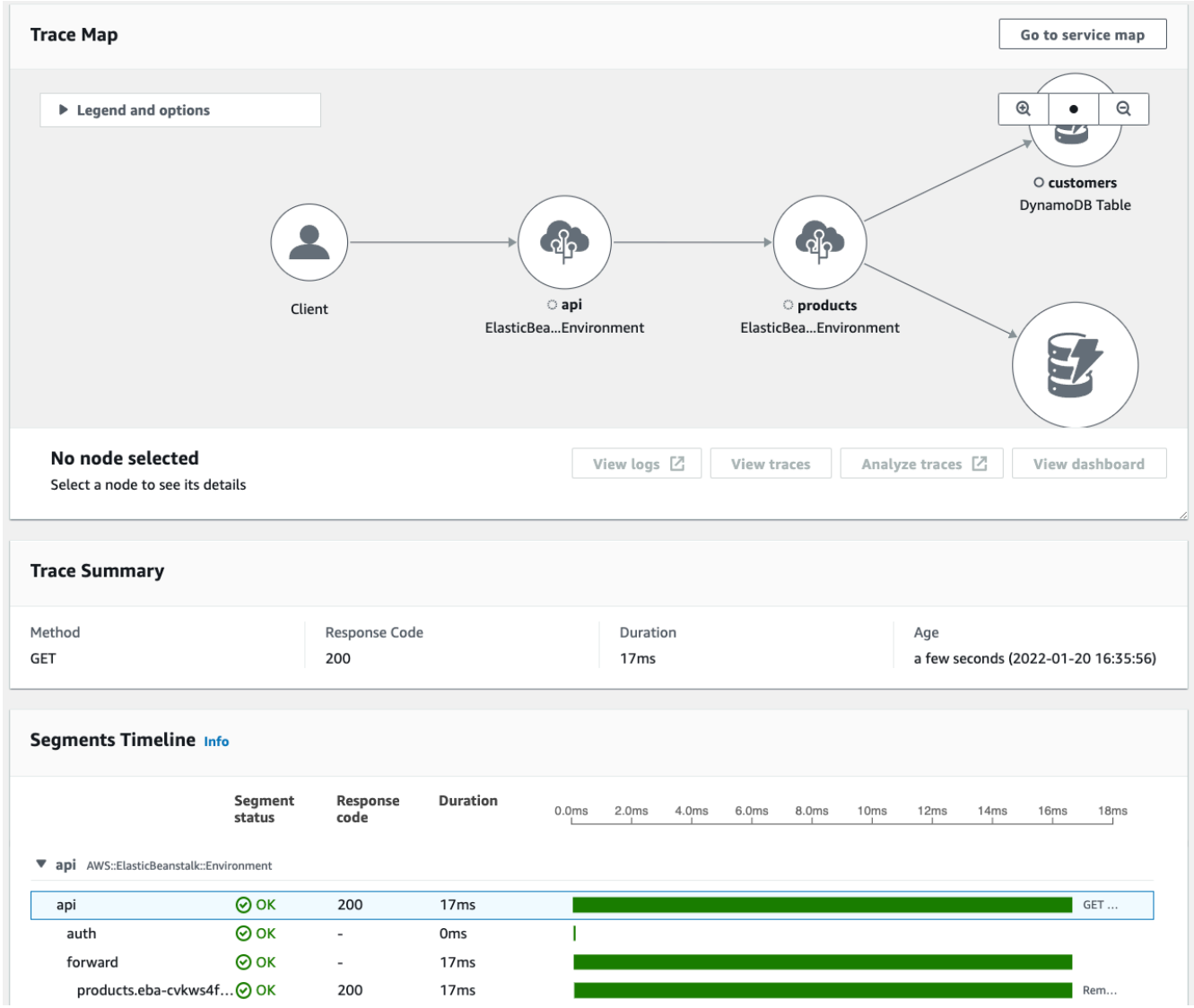
選擇一或多個項目，然後選擇 [新增至查詢]，以新增至頁面頂端的篩選器運算式。

4. 若要尋找單一追蹤，請直接在查詢欄位中輸入追蹤 ID。您可以使用 X-Ray 格式或萬維網聯盟（W3C）格式。例如，使用發行AWS 版所建立的追蹤 OpenTelemetry是 W3C 格式。

> **Note**
>
> 當您查詢使用 W3C 格式追蹤識別碼建立的追蹤時，主控台會以 X-Ray 格式顯示相符的追蹤。例如，如果您以 W3C 格式進`4efaaf4d1e8720b39541901950019ee5`行查詢，則主控台會顯示 X-Ray 對等項目：`1-4efaaf4d-1e8720b39541901950019ee5`。

5. 選擇「隨時執行查詢」，即可在頁面底端的「追蹤」段落中顯示相符追蹤的清單。
6. 若要顯示單一追蹤的「追蹤詳細資訊」頁面，請從清單中選取追蹤 ID。

下圖顯示了一個跟踪映射，其中包含與跟踪相關聯的服務節點和節點之間的邊緣，這些節點表示構成跟踪的段所採取的路徑。追蹤摘要會在追蹤對映之後。摘要包含範例GET作業、其回應碼、追蹤執行所花費的持續時間，以及要求存留期間的相關資訊。「區段時間表」遵循「追蹤摘要」，顯示追蹤區段和子區段的持續時間。



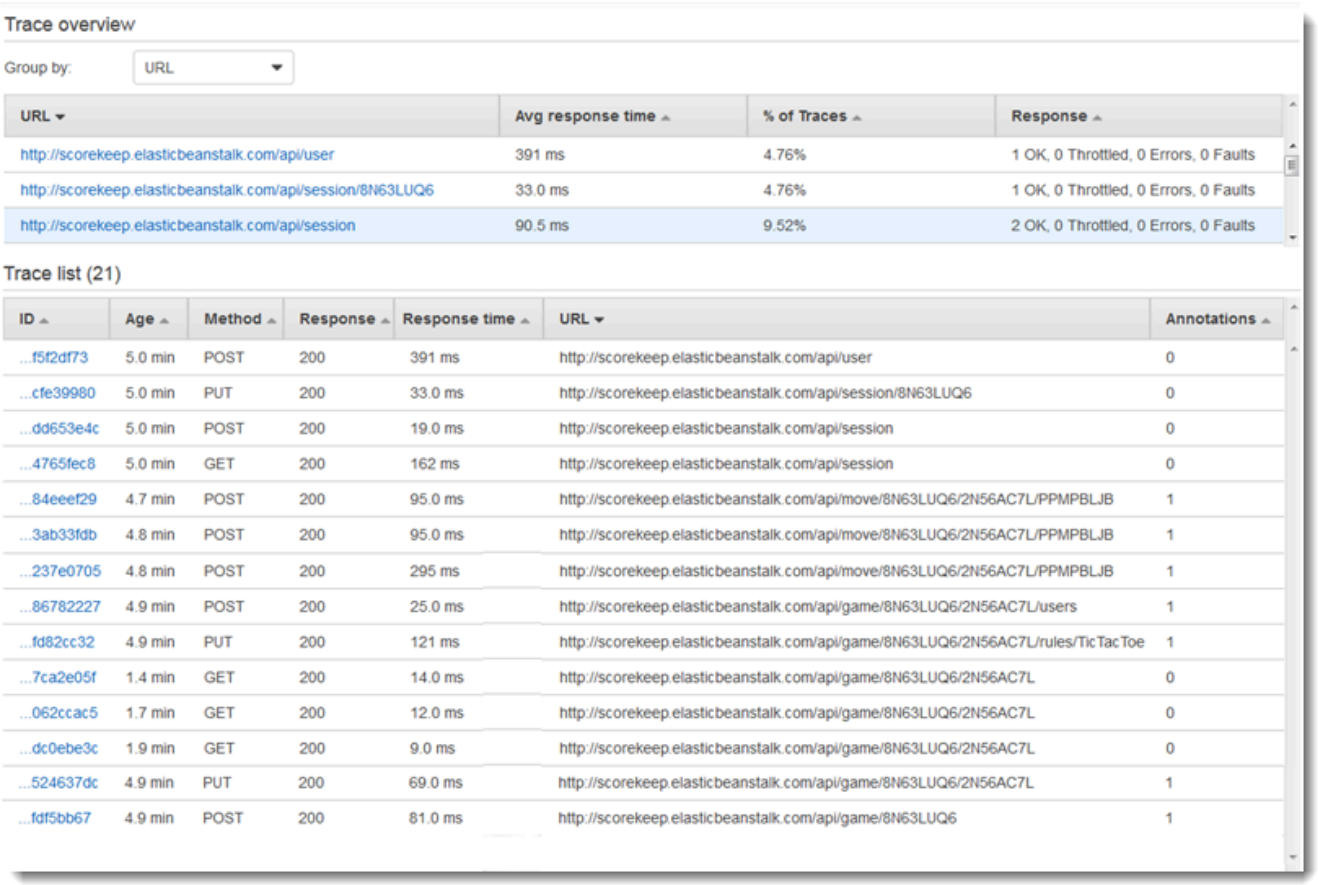
如果您有使用 Amazon SQS 和 Lambda 的事件導向應用程式，則可以在追蹤對應中查看每個請求的連線追蹤檢視。在地圖中，來自訊息產生者的追蹤會連結至來自 AWS Lambda 消費者的追蹤，並顯示為虛線邊緣。如需事件導向應用程式的詳細資訊，請參閱。追蹤事件導向的應用程式

追蹤和追蹤詳細資料頁面也支援跨帳戶追蹤，可列出來自追蹤清單和單一追蹤對應中多個帳戶的追蹤。

X-Ray console

在 X-Ray 主控台中檢視軌跡

1. 在 X-Ray 主控台中開啟「追蹤」頁面。追蹤概觀面板會顯示依常見功能 (包括錯誤根本原因、ResourceARN 和) 分組的追蹤清單。InstanceId
2. 若要選取共用特徵來檢視已群組的繪線集合，請展開「群組依據」(Group By) 旁邊的向下箭頭。下圖顯示依據 URL 分組的追蹤概觀AWS X-Ray 範例應用，以及相關聯追蹤的清單。

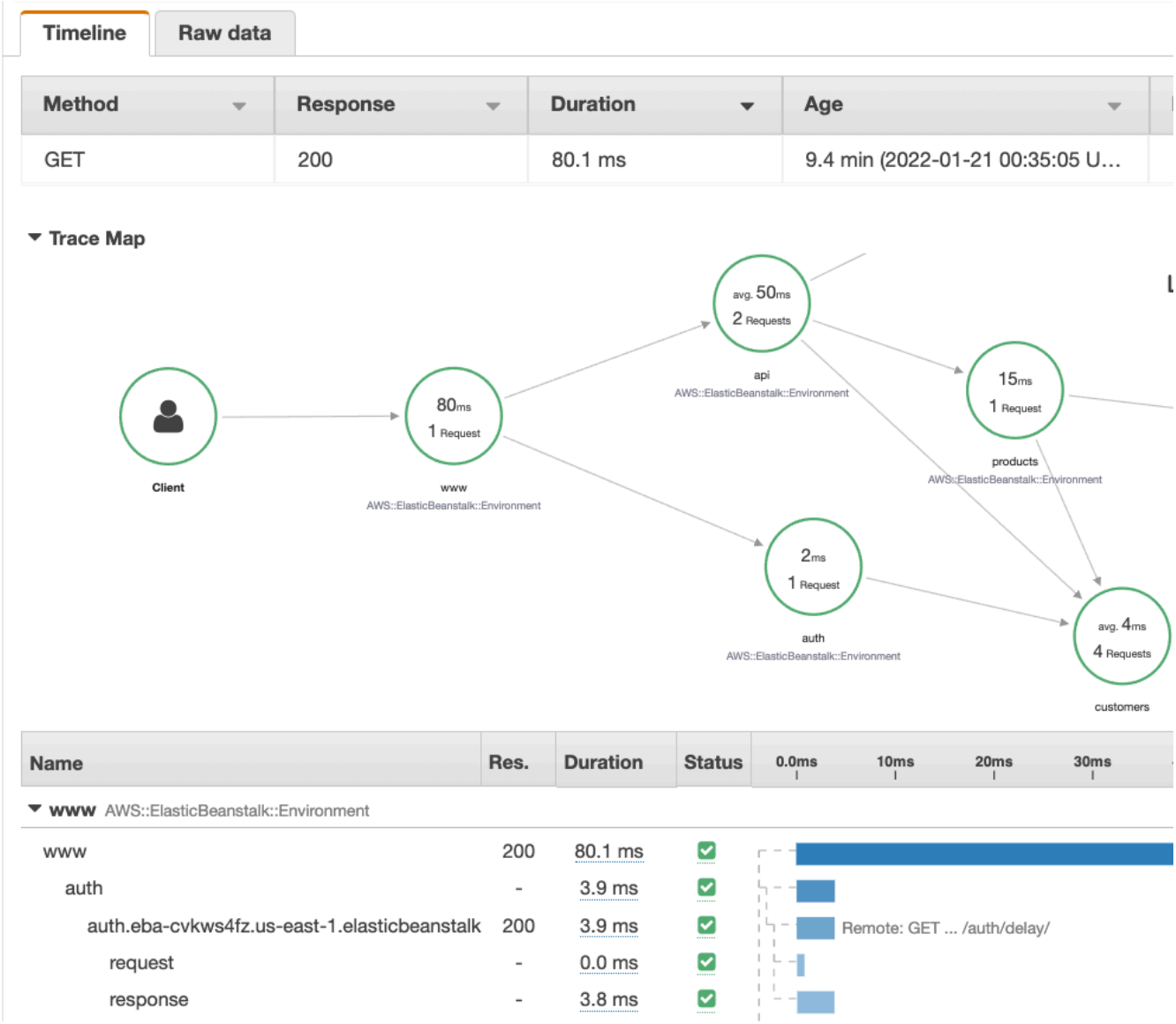


3. 選擇追蹤的 ID，以便在「追蹤」清單下進行檢視。您也可以在導覽窗格中選擇 [服務對應]，以檢視特定服務節點的追蹤。然後，您可以檢視與該節點相關聯的追蹤。

   「時間軸」索引標籤會顯示追蹤的要求流程，並包含下列項目：

   - 軌跡中每個區段的路徑對映。
   - 區段到達追蹤對應中的節點所花費的時間。
   - 追蹤對應中的節點發出了多少要求。

下圖顯示了與對範例應用程式發出的**GET**請求相關聯的「追蹤對應」範例。箭頭會顯示每個區段完成要求所採用的路徑。服務節點會顯示GET要求期間發出的要求數目。



如需有關「時間軸」標籤的詳細資訊，請參閱下面的「探索追蹤時間表」一節。

「原始資料」索引標籤會以JSON格式顯示追蹤的相關資訊，以及構成追蹤的區段和子區段。這些信息可能包括以下內容：

- 時間戳記
- 唯一 ID
- 與區段或子區段相關聯的資源

- 段或子段的來源或原點
- 應用程式要求的其他相關資訊，例如來自 HTTP 要求的回應

探索追蹤時間軸

「時間軸」區段會在水平列旁顯示區段和子區段的階層，該階層與其完成工作所用的時間相對應。清單中的第一個項目是區段，代表服務為單一請求記錄的所有資料。子區段會縮排並列在區段之後。欄包含每個區段的相關資訊。

CloudWatch console

在 CloudWatch 主控台中，「區段時間軸」提供下列資訊：

- 第一個資料欄：列出所選追蹤中的區段和子區段。
- 「區段狀態」欄：列出每個區段與子區段的狀態結果。
- 「回應代碼」資料欄：列出區段或子區段所發出之瀏覽器要求的 HTTP 回應狀態碼 (如果有的話)。
- 「持續時間」欄：列出區段或子區段執行的時間長度。
- 「託管於」欄：列出執行區段或子區段的命名空間或環境 (如果適用)。如需詳細資訊，請參閱 https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/AppSignals-StandardMetrics.html#AppSignals-StandardMetrics-Dimensions。
- 最後一欄：顯示與區段或子區段執行的持續時間相對應的水平列，相對於時間表中的其他區段或子區段。

若要依服務節點將區段和子區段清單分組，請開啟「依節點分組」。

X-Ray console

在追蹤詳細資訊頁面中，選擇「時間軸」標籤，以查看組成追蹤之每個區段和子區段的時間表。

在 X-Ray 主控台中，時間軸提供下列資訊：

- 名稱資料欄：列出追蹤中區段和子區段的名稱。
- Res. 資料欄：列出區段或子區段所發出之瀏覽器要求的 HTTP 回應狀態碼 (如果有的話)。
- 「持續時間」欄：列出區段或子區段執行的時間長度。
- 「狀態」欄位：列出區段或子區段狀態的結果。

- 最後一欄：顯示與區段或子區段執行的持續時間相對應的水平列，相對於時間表中的其他區段或子區段。

若要查看主控台用來產生時間表的原始追蹤資料，請選擇「原始資料」索引標籤。原始資料會顯示追蹤的相關資訊，以及以JSON格式組成追蹤的區段和子區段。這些信息可能包括以下內容：

- 時間戳記
- 唯一 ID
- 與區段或子區段相關聯的資源
- 段或子段的來源或原點
- 有關應用程式要求的其他資訊，例如來自 HTTP 要求的回應。

當您使用已檢測的 AWS SDK 或用SQL戶端來呼叫外部資源時，X-Ray SDK 會自動記錄子區段。HTTP您也可以使用 X-Ray SDK 記錄任何函數或程式碼區塊的自訂子區段。開啟自訂子區段時所記錄的其他子區段會成為自訂子區段的子區段。

檢視區段詳細資訊

從追蹤時間軸中，選擇要檢視其詳細資訊的區段名稱。

區段詳細資料面板會顯示「概觀」、「資源」、「註釋」、「中繼資料」、「例外」和「SQL」 以下情況適用：

- Overview (概觀) 標籤會顯示請求及回應的相關資訊。資訊包括名稱、開始時間、結束時間、持續時間、要求 URL、要求作業、要求回應碼，以及任何錯誤和錯誤。
- 區段的「資源」標籤會顯示 X-Ray SDK 的資訊，以及執行應用程式之 AWS 資源的相關資訊。使用適用於 X-Ray 開發套件的亞馬 Amazon EC2 或 Amazon ECS 外掛程式來記錄服務特定的資源資訊。 AWS Elastic Beanstalk如需外掛程式的詳細資訊，請參閱中的服務外掛程式一節設定適用於 Java 的 X-Ray SDK。
- 其餘標籤會顯示為區段記錄的「註釋」、「中繼資料」和「例外」。從已檢測的請求產生例外狀況時，會自動擷取例外狀況。註釋和中繼資料包含您使用 X-Ray SDK 提供的作業所記錄的其他資訊。若要將註解或中繼資料新增至區段，請使用 X-Ray SDK。如需詳細資訊，請參閱中使 AWS X-Ray 用 SDK 檢測您的應用程式中列出的特定語言連結。檢測您的應用程式 AWS X-Ray

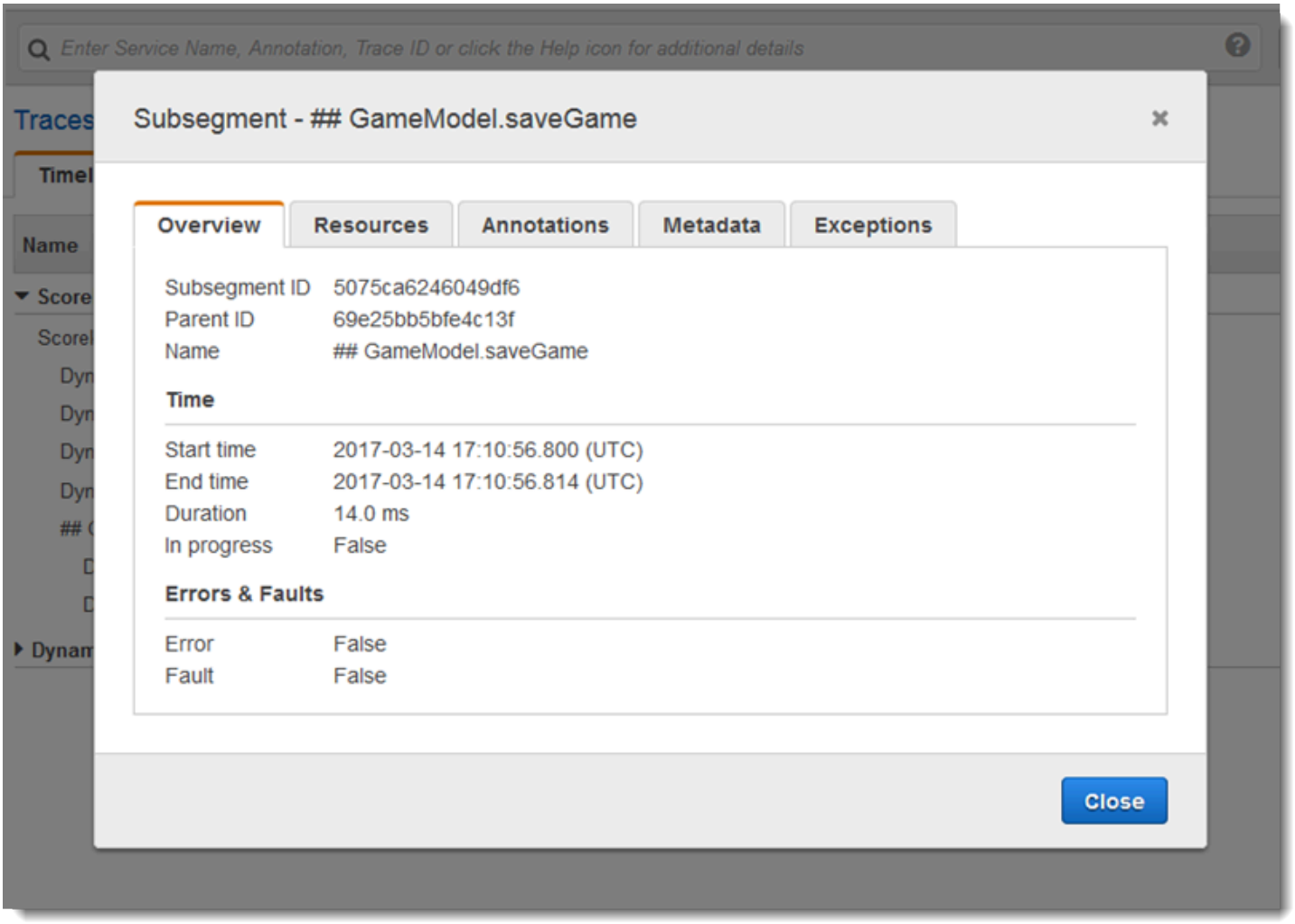
檢視子區段詳細資訊

在追蹤時間表中，選擇子區段的名稱以檢視其詳細資訊：

- [概觀] 索引標籤包含要求和回應的相關資訊。這包括名稱、開始時間、結束時間、持續時間、要求URL、要求作業、要求回應碼，以及任何錯誤和錯誤。針對使用受檢測用戶端產生的子區段，Overview (概觀) 標籤包含從您應用程式觀點的請求和回應相關資訊。
- 子區段的「資源」標籤會顯示用來執行子區段之 AWS 資源的詳細資訊。例如，資源索引標籤可能包括 AWS Lambda 函數 ARN、DynamoDB 表的相關資訊、任何呼叫的作業以及請求識別碼。
- 其餘標籤會顯示記錄在子區段上的「註釋」、「中繼資料」和「例外」。從已檢測的請求產生例外狀況時，會自動擷取例外狀況。註釋和中繼資料包含您使用 X-Ray SDK 提供的作業所記錄的其他資訊。使用 X-Ray SDK 將註釋或中繼資料新增至您的區段。如需詳細資訊，請參閱中使 AWS X-Ray 用 SDK 檢測您的應用程式中列出的特定語言連結。檢測您的應用程式 AWS X-Ray

針對自訂子區段，Overview (概觀) 標籤會顯示子區段的名稱，您可以設定該名稱來指定其記錄的程式碼或函數區域。如需詳細資訊，請參閱中使 AWS X-Ray 用 SDK 檢測您的應用程式中列出的特定語言連結。使用適用於 Java 的 X-Ray SDK 產生自訂子區段

下圖展示了自訂子區段的「概觀」頁籤。概觀包含子區段 ID、父項 ID、名稱、開始和結束時間、持續時間、狀態以及錯誤或錯誤。



自訂子區段的「中繼資料」標籤包含有關該子區段所使用資源的JSON格式資訊。

使用篩選運算式

使用篩選器運算式來檢視特定要求、服務、兩個服務 (邊緣) 之間的連線或滿足條件的要求的追蹤對應或追蹤。X-Ray 提供篩選器運算式語言，可根據要求標頭、回應狀態和原始區段上的索引欄位中的資料篩選要求、服務和邊緣。

當您選擇要在 X-Ray 主控台中檢視的追蹤時段時，可能會得到比主控台顯示的更多結果。主控台會在右上角顯示已掃描的追蹤數目，但實際上可用的追蹤可能更多。您可以使用篩選器運算式將結果縮小為只要尋找的追蹤。

主題

- 篩選條件表達式詳細資訊

- 搭配使用篩選條件表達式與群組
- 篩選條件表達式語法
- 布林值關鍵字
- 數字關鍵字
- 字串關鍵字
- 複雜關鍵字
- id 函數

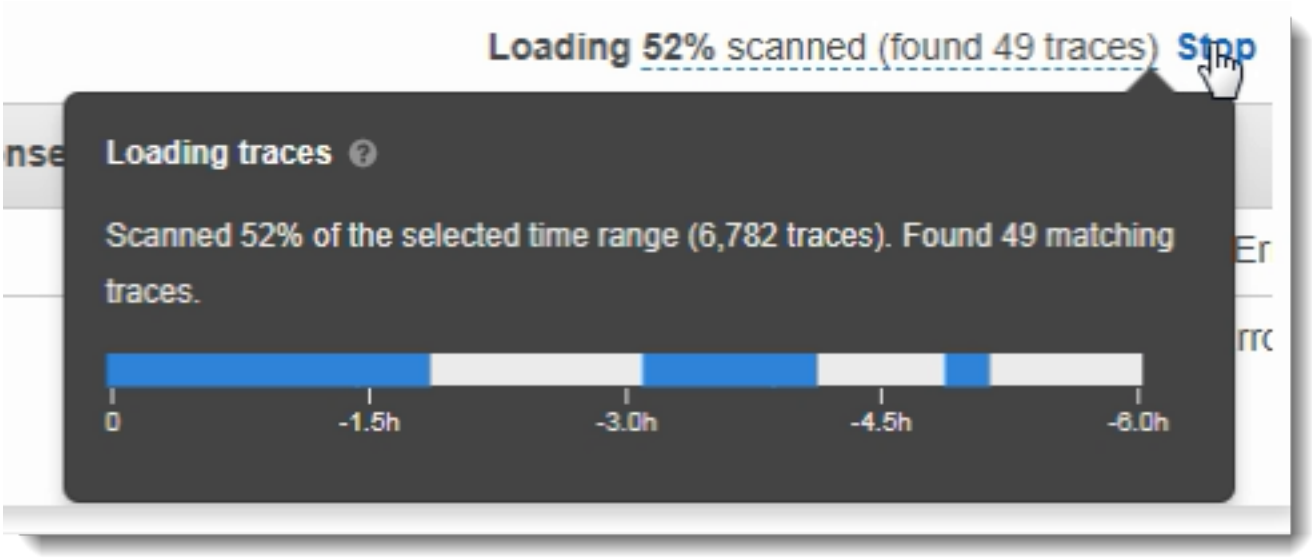
篩選條件表達式詳細資訊

當您在追蹤對應中選擇節點時，主控台會根據節點的服務名稱建構篩選器運算式，以及根據您的選取項目存在的錯誤類型。若要尋找顯示效能問題或與特定請求相關的追蹤，您可以調整主控台提供的表達式，或是自行建立。如果您使用 X-Ray SDK 新增註釋，您也可以根據註釋金鑰的存在或金鑰值進行篩選。

> **Note**
>
> 如果您在追蹤對映中選擇相對時間範圍並選擇節點，主控台會將時間範圍轉換為絕對開始和結束時間。為了確保搜尋結果顯示節點的追蹤，並避免掃描未作用中節點的時間，時間範圍只包含節點傳送追蹤的時間。若要搜尋相對於目前的時間，您可以在追蹤頁面切換回相對時間範圍，並再次掃描。

如果可用結果超過主控台可以顯示的範圍，主控台會顯示相符的追蹤數和已掃描的追蹤數。顯示的百分比為所選時間範圍內已掃描的百分比。若要確保結果中可以顯示所有相符的追蹤，請進一步縮小篩選條件表達式，或選擇較短的時間範圍。

若要先取得最新的結果，主控台會從時間範圍結尾開始掃描，並倒退進行。如果有大量追蹤，但僅有少數結果，主控台會將時間範圍分成區塊並平行掃描。進度列會顯示已掃描的部分時間範圍。



搭配使用篩選條件表達式與群組

群組是一系列追蹤，為篩選條件表達式所定義。您可以使用群組產生其他服務圖表並提供 Amazon CloudWatch 指標。

群組會根據名稱或 Amazon Resource Name (ARN) 進行識別，且包含篩選條件表達式。此服務會比較傳入表達式的追蹤，並依序存放。

您可以在下拉式功能表，篩選條件表達式左側的搜尋列，建立或修改群組。

> **Note**
>
> 如果服務驗證群組資格時發生錯誤，該群組即不會處理傳入的追蹤，且會記錄錯誤指標。

如需群組的詳細資訊，請參閱設定群組。

篩選條件表達式語法

篩選條件表達式可以包含「關鍵字」、一元或二元「運算子」以及用於比較的「值」。

```
keyword operator value
```

不同運算子適用於不同類型的關鍵字。例如，`responsetime` 是數字關鍵字，可相較於與數字相關的運算子。

Example — 回應時間大於 5 秒的要求

```
responsetime > 5
```

您可以使用 AND 或 OR 運算子，將多個表達式結合成一個複合表達式。

Example — 總持續時間為 5—8 秒的請求

```
duration >= 5 AND duration <= 8
```

簡單關鍵字和運算子只能發現追蹤層級的問題。如果下游發生錯誤，但已由您的應用程式處理而未傳回給使用者，搜尋 `error` 時就不會找到此錯誤。

若要尋找含有下游問題的追蹤，您可以使用複雜的關鍵字 `service()` 和 `edge()`。這些關鍵字可讓您將篩選條件表達式套用到所有下游節點、單一下游節點，或是兩個節點之間的邊緣。如需更高的精細程度，您可以依據 id() 函數類型來篩選服務和邊緣。

布林值關鍵字

布林值關鍵字值可為 true 或 false。使用這些關鍵字來尋找導致錯誤的追蹤。

布林值關鍵字

- ok— 響應狀態代碼為 2XX「成功」。
- `error`— 響應狀態代碼為 4XX 客戶端錯誤。
- `throttle`— 響應狀態代碼為 429 請求太多。
- `fault`— 回應狀態碼為 5XX 伺服器錯誤。
- `partial`— 請求具有不完整的區段。
- `inferred`— 請求推斷了區段。
- `first`— 元素是列舉清單的第一個。
- `last`— 元素是列舉清單的最後一個。
- `remote`— 根本原因實體是遠端的。
- `root`— 服務是追蹤的進入點或根區段。

布林值運算子可尋找指定索引鍵為 `true` 或 `false` 的區段。

布林值運算子

- 無 — 如果關鍵字為真，則表示式為真。
- ! — 如果關鍵字為假，則表示式為 true。
- =，!= — 將關鍵字的值與字串`true`或比較`false`。這些運算子的作用與其他運算子相同，但更加明確。

Example — 響應狀態為 2XX「確定」

```
ok
```

Example — 響應狀態不是 2XX「確定」

```
!ok
```

Example — 響應狀態不是 2XX「確定」

```
ok = false
```

Example — 上次列舉的錯誤追蹤具有錯誤名稱「還原序列化」

```
rootcause.fault.entity { last and name = "deserialize" }
```

Example — 具有覆蓋範圍大於 0.7 且服務名稱為「追蹤」之遠端區段的要求

```
rootcause.responsetime.entity { remote and coverage > 0.7 and name = "traces" }
```

Example — 具有推斷區段的請求，其中服務類型為「AWS：DynamoDB」

```
rootcause.fault.service { inferred and name = traces and type = "AWS::DynamoDB" }
```

Example — 具有名稱為「資料平面」作為根區段的請求

```
service("data-plane") {root = true and fault = true}
```

數字關鍵字

使用數字關鍵字以搜尋含特定回應時間、持續時間或回應狀態的請求。

數字關鍵字

- `responsetime`— 伺服器傳送回應所花費的時間。
- `duration`— 總請求持續時間，包括所有下游呼叫。
- `http.status`— 響應狀態代碼。
- `index`— 元素在列舉清單中的位置。
- `coverage`— 根區段回應時間內實體回應時間的十進位百分比。僅適用於回應時間根本原因實體。

數字運算子

數字關鍵字使用標準的對等和比較運算子。

- =、!= — 關鍵字等於或不等於數字值。
- <、<=、>、>= — 關鍵字小於或大於數字值。

Example — 響應狀態不是 200「確定」

```
http.status != 200
```

Example — 請求總持續時間為 5—8 秒

```
duration >= 5 AND duration <= 8
```

Example — 在 3 秒內成功完成的請求，包括所有下游呼叫

```
ok !partial duration <3
```

Example — 索引大於 5 的列舉清單實體

```
rootcause.fault.service { index > 5 }
```

Example — 具有覆蓋範圍大於 0.8 的最後一個實體的請求

```
rootcause.responsetime.entity { last and coverage > 0.8 }
```

字串關鍵字

使用字串關鍵字以尋找請求標頭中含特定文字或特定使用者 ID 的追蹤。

字串關鍵字

- `http.url`— 請求網址。
- `http.method`— 請求方法。
- `http.useragent`— 要求使用者代理字串。
- `http.clientip`— 要求者的 IP 位址。
- `user`— 追蹤中任何區段上的使用者欄位值。
- `name`— 服務或例外的名稱。
- `type`— 服務類型。
- `message`— 例外訊息。
- `availabilityzone`— 追蹤中任何區段上可用區域欄位的值。
- `instance.id`— 追蹤中任何區段上的執行個體 ID 欄位值。
- `resource.arn`— 追蹤中任何區段上資源 ARN 欄位的值。

字串運算子可尋找等於或包含特定文字的值。您必須一律在引號中指定值。

字串運算子

- `=`、`!=` — 關鍵字等於或不等於數字值。
- `CONTAINS`— 關鍵字包含特定字串。
- `BEGINSWITH`, `ENDSWITH` — 關鍵字以特定字串開頭或結尾。

Example -網址過濾器

```
http.url CONTAINS "/api/game/"
```

若要測試追蹤上是否存在欄位 (無論其值為何)，請檢查欄位是否包含空白字串。

Example -用戶過濾器

尋找所有含使用者 ID 的追蹤。

```
user CONTAINS ""
```

Example — 選取具有錯誤根本原因的追蹤，其中包含名為「Auth」的服務

```
rootcause.fault.service { name = "Auth" }
```

Example — 選取具有回應時間根本原因的追蹤，其上一個服務具有 DynamoDB 類型

```
rootcause.responsetime.service { last and type = "AWS::DynamoDB" }
```

Example — 選取具有錯誤根本原因的追蹤，其最後一個例外狀況會顯示「account_id 的存取遭拒：1234567890」

```
rootcause.fault.exception { last and message = "Access Denied for account_id:
 1234567890"
```

複雜關鍵字

使用複雜關鍵字以根據服務名稱、邊緣名稱或註釋值來尋找請求。若是服務和邊緣，您可以指定額外篩選條件表達式以套用到服務或邊緣。若是註釋，您可以使用布林值、數字或字串運算子，篩選含特定索引鍵的註釋值。

複雜關鍵字

- `annotation.`*`key`*— 含欄位*#*之註釋的值。註釋值可以是布林值、數字或字串，所以您可以使用任何這些類型的比較運算子。您可以將此關鍵字與 `service` or 關鍵edge字結合使用。
- `edge(`*`source`*`, `*`destination`*`) {`*`filter`*`}`— 服務*#*和*###*之間的連接。選用的大括號可包含篩選表達式，以套用到此連線的服務。
- `group.`*`name`* `/ group.`*`arn`*— 群組篩選運算式的值，依群組名稱或群組 ARN 參照。
- `json`— JSON 根本原因物件。如需以程式設計方式建立 JSON 實體的步驟，請參閱從 AWS X-Ray 取得
- `service(`*`name`*`) {`*`filter`*`}`— 具有名*#*的服務。選用的大括號可包含篩選表達式，以套用到服務所建立的區段。

使用 service 關鍵字來尋找追蹤對應上某個節點的要求追蹤。

複雜的關鍵字運算子會尋找已設定或未設定指定索引鍵的區段。

複雜關鍵字運算

- 無 — 如果已設定關鍵字，則表示式為 true。如果關鍵字是布爾類型，它將評估為布爾值。
- !— 如果未設定關鍵字，則表示式為 true。如果關鍵字是布爾類型，它將評估為布爾值。
- =，!=— 比較關鍵字的值。
- edge(*source*, *destination*) {*filter*}— 服務#和###之間的連接。選用的大括號可包含篩選表達式，以套用到此連線的服務。
- annotation.*key*— 含欄位#之註釋的值。註釋值可以是布林值、數字或字串，所以您可以使用任何這些類型的比較運算子。您可以將此關鍵字與 service or 關鍵edge字結合使用。
- json— JSON 根本原因物件。如需以程式設計方式建立 JSON 實體的步驟，請參閱從 AWS X-Ray 取得

使用 service 關鍵字來尋找追蹤對應上某個節點的要求追蹤。

Example — 服務過濾器

包含對 api.example.com 的呼叫且發生故障 (500 系列錯誤) 的請求。

```
service("api.example.com") { fault }
```

您可以排除服務名稱，以將篩選條件表達式套用到服務地圖中的所有節點。

Example — 服務過濾器

在追蹤對應上的任何位置造成錯誤的要求。

```
service() { fault }
```

邊緣關鍵字可將篩選條件表達式套用到兩個節點之間的連線。

Example — 邊緣濾鏡

api.example.com 服務對 backend.example.com 發出呼叫但因錯誤而失敗的請求。

```
edge("api.example.com", "backend.example.com") { error }
```

您也可以搭配使用 ！運算子與服務和邊緣關鍵字，以排除其他篩選條件表達式結果的服務或邊緣。

Example — 服務和請求過濾器

URL 開頭為 `http://api.example.com/` 並包含 `/v2/` 但未到達名稱為 `api.example.com` 之服務的請求。

```
http.url BEGINSWITH "http://api.example.com/" AND http.url CONTAINS "/v2/" AND !
service("api.example.com")
```

Example — 服務和響應時間過濾器

尋找`http url`已設定且回應時間大於 2 秒的追蹤。

```
http.url AND responseTime > 2
```

對於註釋，您可以調用設置的所有跟踪，或使用對應於值類型的比較運算符。`annotation.`*`key`*

Example -帶有字符串值的註釋

含名稱為 `gameid`、字串值為 `"817DL6V0"` 之註釋的請求。

```
annotation.gameid = "817DL6V0"
```

Example — 註釋被設置

具有名為 age set 之註釋的請求。

```
annotation.age
```

Example — 未設定註解

沒有註釋命名為 age set 的請求。

```
!annotation.age
```

Example -帶有數字值的註釋

含註釋存留期且數值大於 29 的請求。

```
annotation.age > 29
```

Example -與服務或邊緣組合的註釋

```
service { annotation.request_id = "917DL6VO" }
```

```
edge { source.annotation.request_id = "916DL6VO" }
```

```
edge { destination.annotation.request_id = "918DL6VO" }
```

Example — 群組與使用者

追蹤符合high_response_time群組篩選條件的要求 (例如responseTime > 3)，且使用者名為 Alice。

```
group.name = "high_response_time" AND user = "alice"
```

Example — 具有根本原因實體的 JSON

有相符根本原因實體的請求。

```
rootcause.json = #[{ "Services": [ { "Name": "GetWeatherData", "EntityPath": [{ "Name":
 "GetWeatherData" }, { "Name": "get_temperature" } ] }, { "Name": "GetTemperature",
 "EntityPath": [ { "Name": "GetTemperature" } ] } ] }]
```

id 函數

當您將服務名稱提供給 service 或 edge 關鍵字時，您可取得具有該名稱之所有節點的結果。如需更精確的篩選，除了名稱之外，您還可以使用 id 函數來指定服務類型，以區分名稱相同的節點。

檢視監視帳戶中多個帳戶的追蹤時，請使用此account.id函數來指定服務的特定帳戶。

```
id(name: "service-name", type:"service::type", account.id:"account-ID")
```

您可以使用 id 函數來代替服務和邊緣篩選條件中的服務名稱。

```
service(id(name: "service-name", type:"service::type")) { filter }
```

```
edge(id(name: "service-one", type:"service::type"), id(name: "service-two",
 type:"service::type")) { filter }
```

例如， AWS Lambda 函數會在追蹤對映中產生兩個節點；一個用於函數叫用，另一個用於 Lambda 服務。這兩個節點的名稱相同，但類型不同。標準的服務篩選條件可尋找這兩種追蹤。

Example — 服務過濾器

包含任何名為 random-name 服務上的錯誤的請求。

```
service("function-name") { error }
```

使用 id 函數來縮小搜尋範圍至函數本身的錯誤，而不含服務的錯誤。

Example — 具有 id 功能的服務過濾器

包含名為 random-name、類型為 AWS::Lambda::Function 服務上的錯誤的請求。

```
service(id(name: "random-name", type: "AWS::Lambda::Function")) { error }
```

若要依據類型來搜尋節點，您也可以完全排除名稱。

Example — 具有 id 功能和服務類型的服務過濾器

包含類型為 AWS::Lambda::Function 服務上的錯誤的請求。

```
service(id(type: "AWS::Lambda::Function")) { error }
```

若要搜尋特定節點 AWS 帳戶，請指定帳號 ID。

Example — 具有 ID 功能和帳戶 ID 的服務過濾器

在特定帳戶 ID 中包含服務的要求AWS::Lambda::Function。

```
service(id(account.id: "account-id"))
```

跨帳戶追蹤

AWS X-Ray 支援跨帳戶觀察能力，使您能夠監控和疑難排解跨多個帳戶的應用程式。 AWS 區域您可以無縫搜索，可視化和分析任何鏈接帳戶中的指標，日誌和跟踪，就像您在單個帳戶中進行操作一樣。這提供了跨多個帳戶傳送的請求的完整視圖。您可以在 X-Ray 追蹤地圖中檢視跨帳戶追蹤，並在CloudWatch主控台內檢視追蹤頁面。

共用的可觀測性資料可包含下列任何類型的遙測：

- Amazon 指標 CloudWatch
- Amazon CloudWatch 日誌中的日誌群組

- 追蹤中 AWS X-Ray
- Amazon 應用程式洞察中的 CloudWatch 應用

設定跨帳戶觀察能力

若要開啟跨帳戶可觀察性，請設定一或多個 AWS 監控帳戶，並將其與多個來源帳戶連結。監視帳戶是一個中心， AWS 帳戶 可以檢視來源帳戶產生的可觀測性資料，並與之互動。來源帳戶是為其包含 AWS 帳戶 的資源產生可觀測性資料的個人。

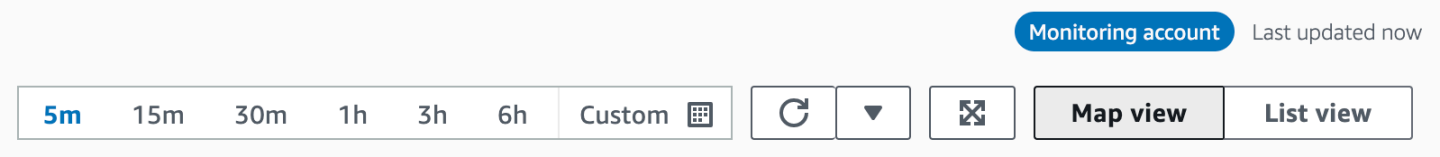
來源帳戶會與監控帳戶共用其觀察性資料。追蹤會從每個來源帳戶複製到最多五個監視帳戶。從源帳戶到第一個監視帳戶的跟踪副本是免費的。傳送至其他監控帳戶的追蹤副本會根據標準定價向每個來源帳戶收取費用。如需詳細資訊，請參閱AWS X-Ray 定價和 Amazon CloudWatch 定價。

若要在監視帳戶和來源帳戶之間建立連結，請使用和 API 中的 CloudWatch 主控台或新的可觀察性存取管理員命令。 AWS CLI 如需詳細資訊，請參閱 CloudWatch CloudWatch 跨帳戶可觀測性。

> **Note**
>
> X-Ray 追蹤會計費至接收 AWS 帳戶 位置。如果取樣的要求跨越多個服務 AWS 帳戶，則每個帳戶都會記錄個別追蹤，而且所有追蹤都共用相同的追蹤 ID。若要進一步了解跨帳戶觀察定價，請參閱AWS X-Ray 定價和 Amazon CloudWatch 定價。

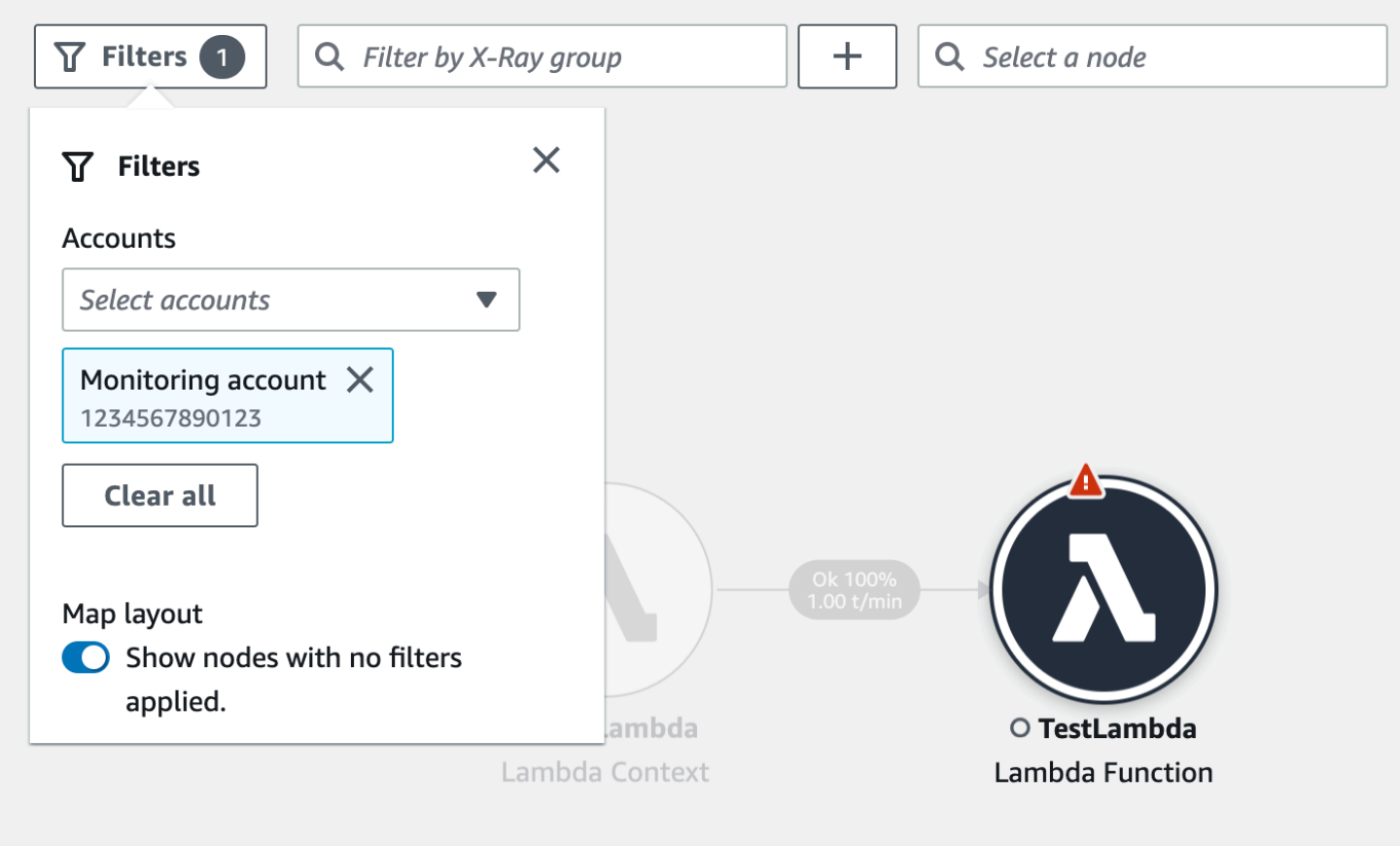
檢視跨帳戶追蹤

跨帳戶跟踪顯示在監視帳戶中。每個來源帳戶只會顯示該特定帳戶的本機追蹤。以下各節假設您已登入監控帳戶並開啟 Amazon 主 CloudWatch 控台。在追蹤對應和追蹤頁面上，監控帳戶徽章會顯示在右上角。

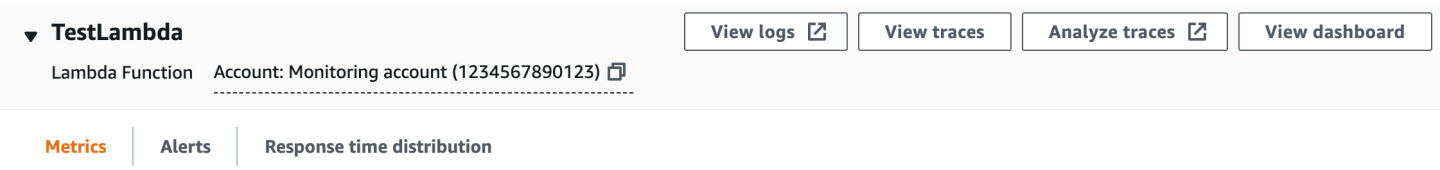


軌跡圖

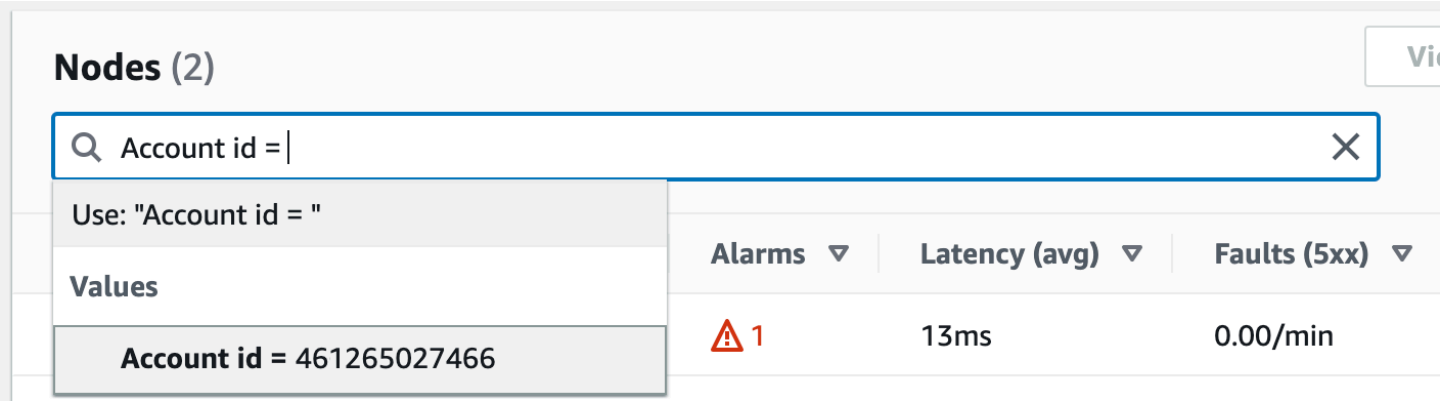
在 CloudWatch 主控台中，從左側導覽窗格的 [X-Ray 追蹤] 下選擇 [追蹤地圖]。根據預設，追蹤對映會顯示傳送追蹤至監視帳戶的所有來源帳戶的節點，以及監視帳戶本身的節點。在追蹤對應上，從左上角選擇「篩選器」，使用「帳戶」下拉式清單篩選追蹤對映。套用帳戶篩選器後，不符合目前篩選器之帳戶的服務節點會變成灰色。



當您選擇服務節點時，節點詳細資料窗格會包含服務的帳戶 ID 和標籤。



在追蹤對應的右上角，選擇 [清單檢視] 以查看服務節點清單。服務節點清單包括來自監視帳戶的服務以及所有已設定的來源帳戶。透過從「節點」篩選器中選擇「帳戶」標籤或「帳戶 ID」來篩選節點清單。

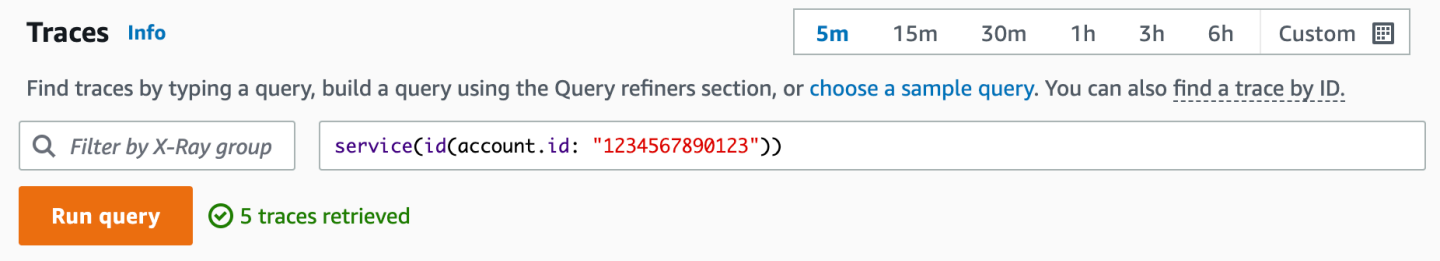


追蹤

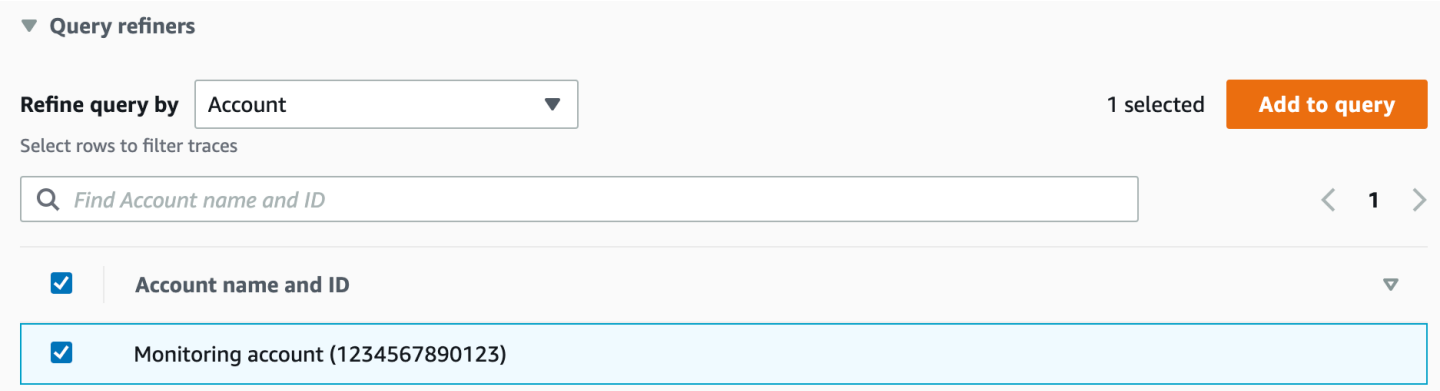
從監視帳戶開啟 CloudWatch 主控台，然後在左側導覽窗格中選擇 X-Ray 追蹤底下的 [追蹤]，檢視跨多個帳戶的追蹤詳細資料。您也可以在「X-Ray 追蹤對映」中選擇節點，然後從節點詳細資料窗格中選擇檢視追蹤，以開啟此頁面。

追蹤頁面支援按帳號 ID 進行查詢。若要開始使用，請輸入包含一或多個帳號 ID 的查詢。下列範例會查詢透過帳戶 ID X 或 Y 傳遞的追蹤：

```
service(id(account.id:"X")) OR service(id(account.id:"Y"))
```



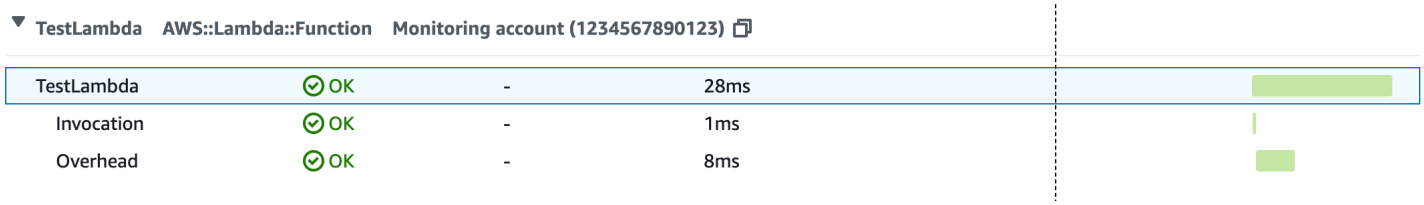
依「帳戶」精簡您的查詢。從清單中選取一或多個帳號，然後選擇 [新增至查詢]。



追蹤細節

從「追蹤」頁面底部的「追蹤」清單中選擇追蹤，以檢視追蹤的詳細資訊。隨即顯示「追蹤」詳細資訊，包括追蹤詳細資訊對應，其中包含來自追蹤所傳遞之所有帳戶的服務節點。選擇特定服務節點以查看其對應帳戶。

「節段時間表」區段會顯示時間表中每個節段的科目明細。



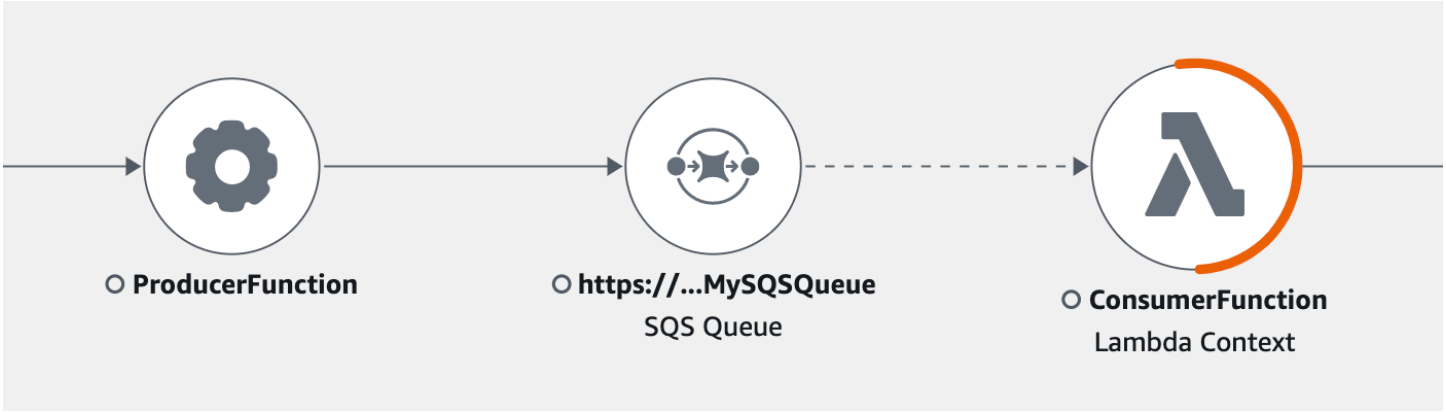
追蹤事件導向的應用程式

AWS X-Ray 支援使用 Amazon SQS 和. AWS Lambda使用主 CloudWatch 控台查看每個請求與 Amazon SQS 排入佇列並由一或多個 Lambda 函數處理時的連線檢視。來自上游訊息生產者的追蹤會自動連結至來自下游 Lambda 消費者節點的追蹤，以建立應用程式的 end-to-end 檢視。

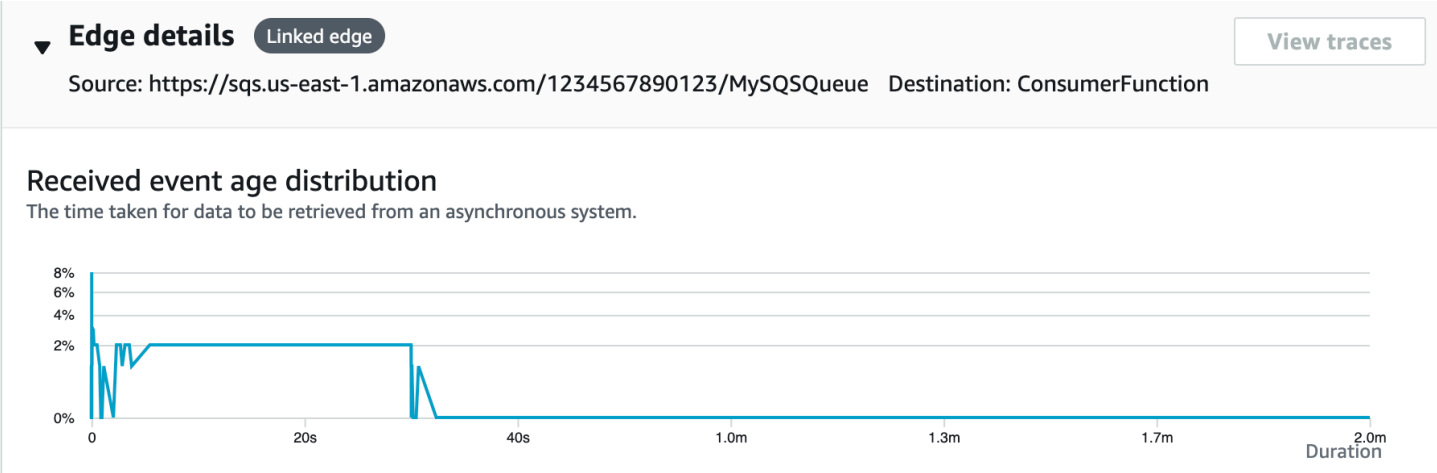
> **Note**
>
> 每個追蹤區段最多可連結至 20 條軌跡，而追蹤最多可包含 100 個連結。在某些情況下，連結其他追蹤可能會導致超過最大追蹤文件大小，進而導致可能不完整的追蹤。例如，啟用追蹤功能的 Lambda 函數會在單次叫用中將多個 SQS 訊息傳送至佇列時，就會發生這種情況。如果您遇到此問題，可以使用 X-Ray SDK 的緩解措施。如需詳細資訊，請參閱適用於 Java、Node.js、Python、圍棋或 .NET 的 X-Ray SDK。

在軌跡對應中檢視連結的軌跡

使用CloudWatch主控台內的「追蹤對應」頁面來檢視追蹤對應，其中包含來自訊息產生器的追蹤，這些追蹤來自 Lambda 取用者的追蹤。這些連結會以虛線邊緣顯示，可連接 Amazon SQS 節點和下游 Lambda 消費者節點。



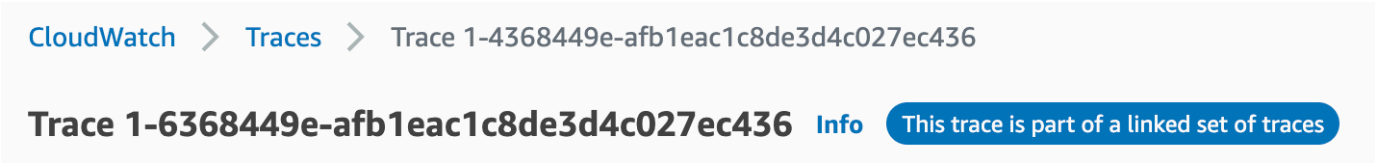
選取虛線邊緣以顯示接收到的事件年齡分佈圖，該分佈圖會對應消費者收到的事件存留時間分佈圖。每次收到事件時，都會計算年齡。



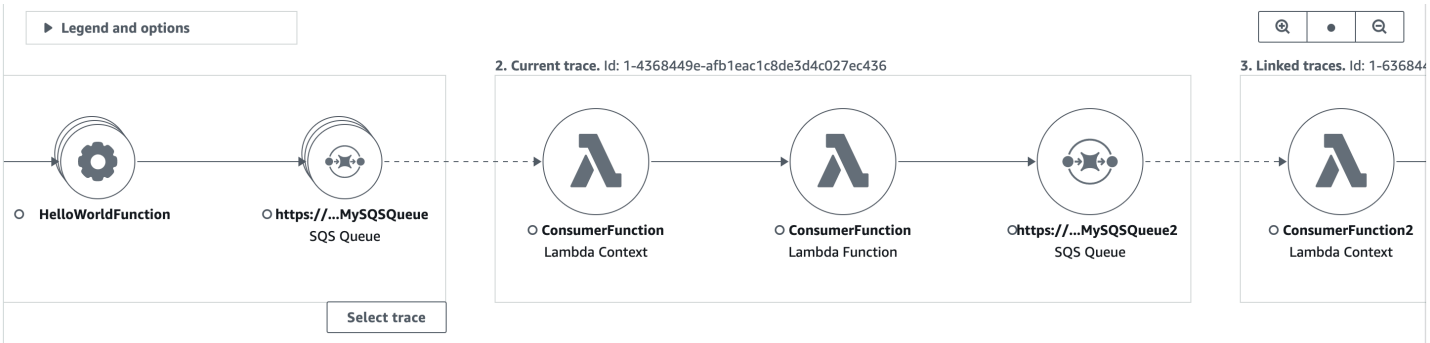
檢視連結的追蹤詳細

檢視從訊息生產者、Amazon SQS 佇列或 Lambda 取用者傳送的追蹤詳細資訊：

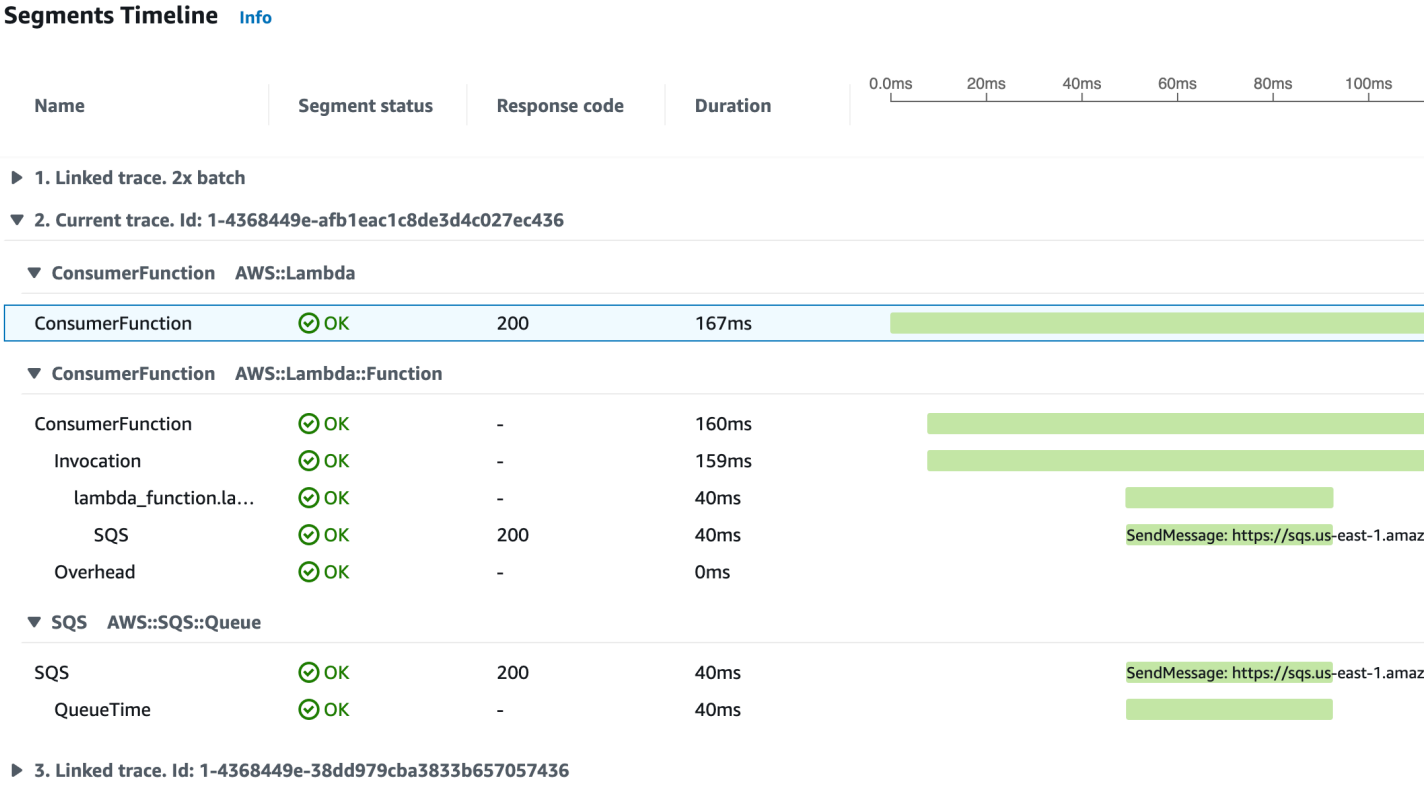
1. 使用追蹤對應來選取訊息產生者、Amazon SQS 或 Lambda 取用者節點。
2. 從節點詳細資料窗格中選擇檢視追蹤，以顯示追蹤清單。您也可以直接導覽至 CloudWatch 主控台內的 [追蹤] 頁面。
3. 從清單中選擇特定追蹤，以開啟追蹤詳細資訊頁面。當所選追蹤屬於連結追蹤集的一部分時，追蹤詳細資訊頁面會顯示訊息。



追蹤詳細資料對映會顯示目前追蹤，以及上游和下游連結的軌跡，每一條線都包含在指示每個追蹤邊界的方塊中。如果目前選取的繪線連結至多個上游或下游軌跡，則會堆疊上游或下游連結繪線內的節點，並顯示「選取繪線」(S elect trace) 按鈕。



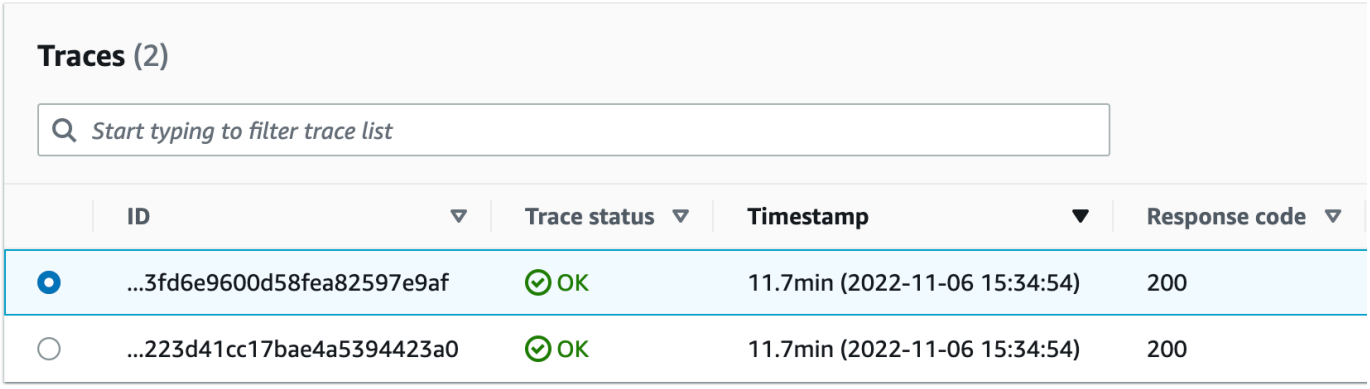
在繪線詳細資料對映下，會顯示追蹤區段的時間表，包括上游和下游連結的軌跡。如果有多條上游或下游連結的軌跡，則無法顯示其區段詳細資料。若要檢視一組連結繪線中單一繪線的區段詳細資訊，請如下所述選取單一繪線。



在一組連結的繪線中選取單一軌跡

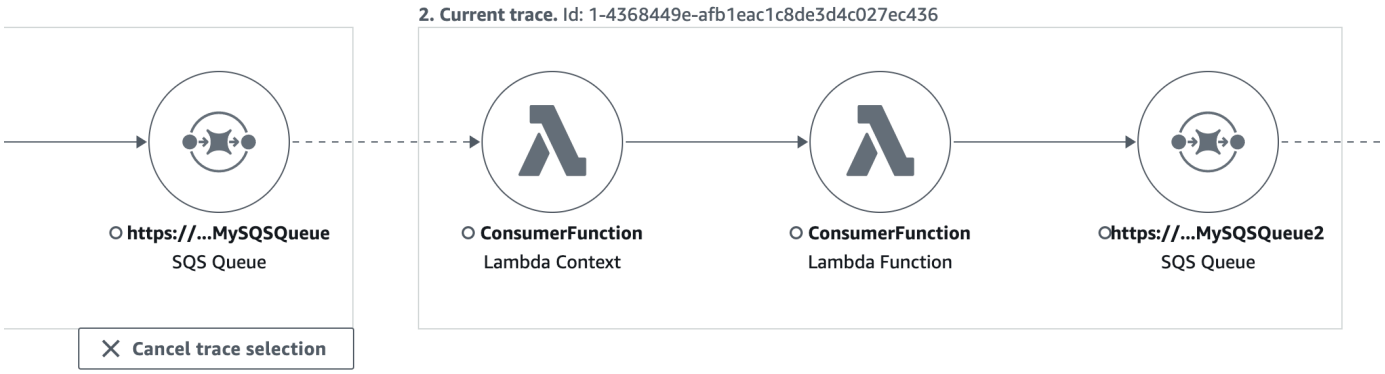
將連結的繪線集篩選為單一繪線，以查看時間軸中的區段詳細資訊。

1. 在繪線詳細資料地圖上，選擇連結繪線下方的「選取繪線」。將顯示軌跡清單。



2. 選取追蹤旁邊的圓鈕，即可在追蹤詳細資訊對映中檢視該追蹤。

3. 選擇「取消繪線選取」以檢視整組連結繪線。



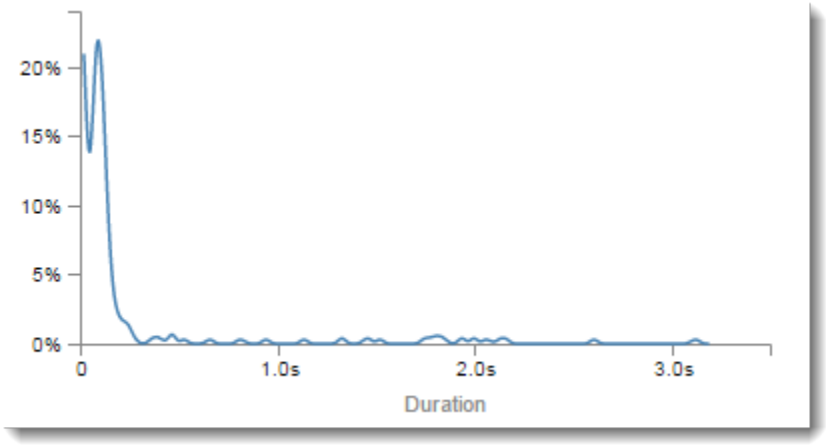
使用延遲直方圖

當您在 AWS X-Ray 追蹤對應上選取節點或邊緣時，X-Ray 主控台會顯示延遲分佈圖。

Latency (延遲)

延遲是指請求開始和完成之間的時間。長條圖中顯示了延遲分佈。它會在 x 軸上顯示持續時間，在 y 軸上顯示符合每個持續時間的請求百分比。

此長條圖顯示的服務可在 300 毫秒內完成大多數請求。少許百分比的請求花費最多 2 秒，以及幾個異常值花費更多時間。



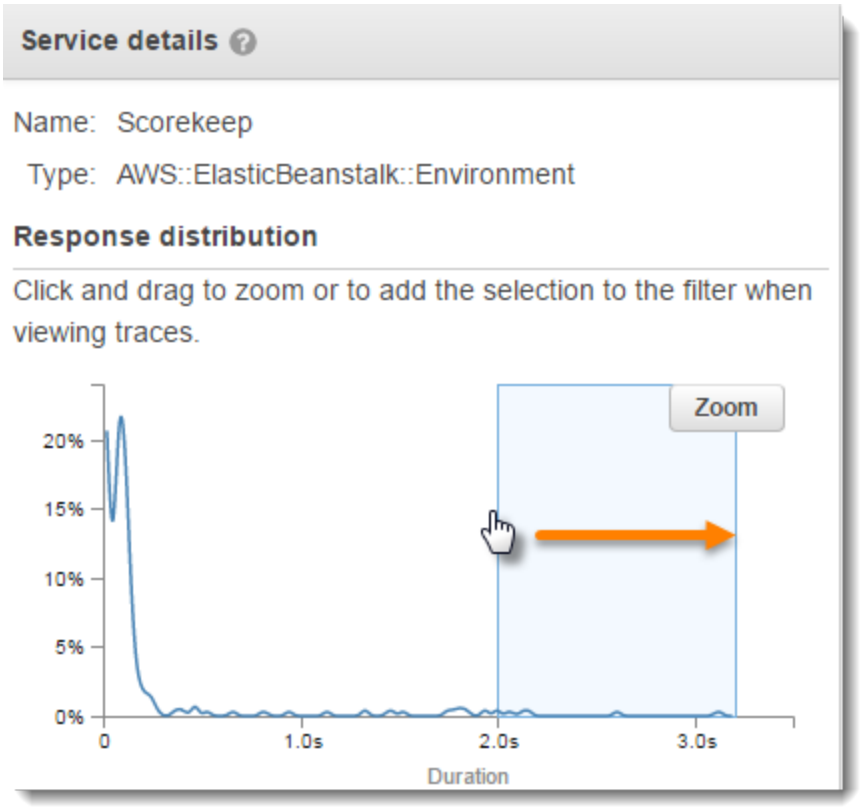
解譯服務詳細資訊

服務長條圖和邊緣長條圖可從服務或申請者的角度，提供延遲的視覺化呈現。

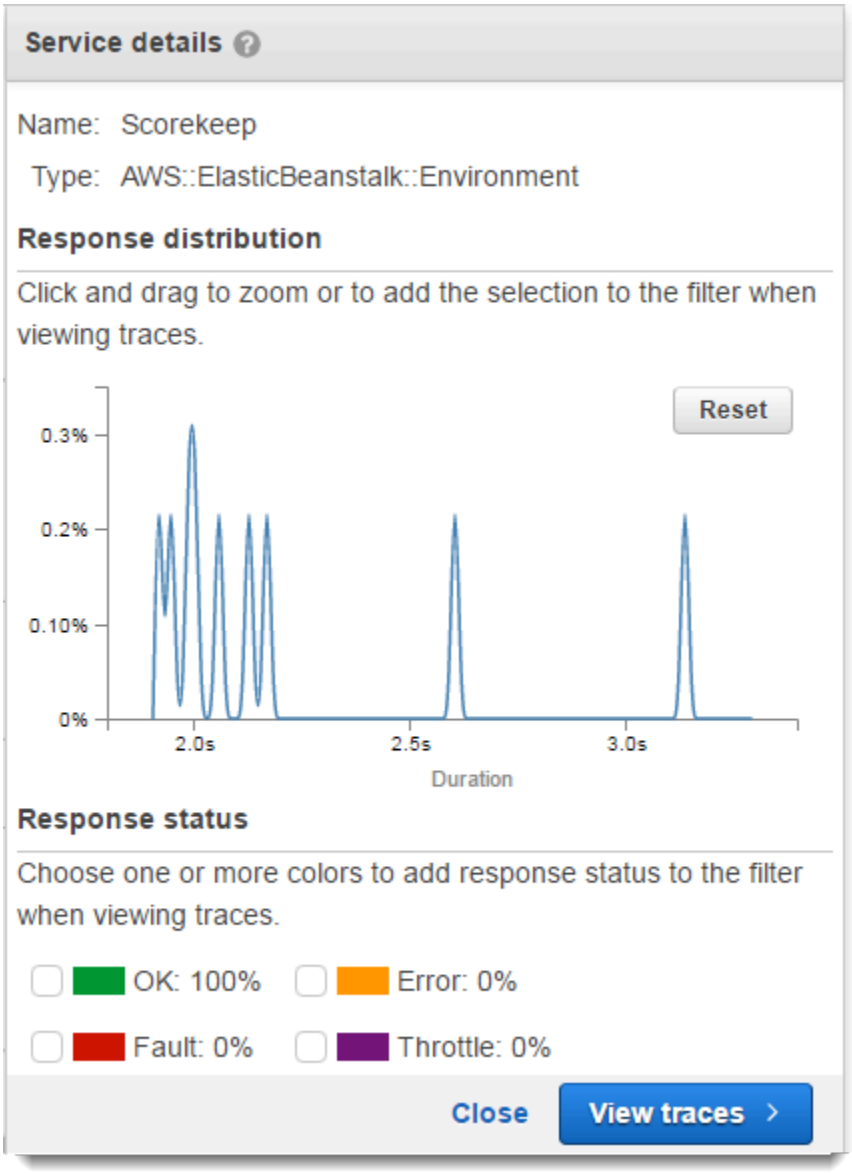
- 按一下圓圈以選擇服務節點。X-Ray 會針對服務提供的要求顯示長條圖。延遲是由服務記錄的延遲，且不包含任何服務和申請者之間的網路延遲。
- 按一下兩個服務之間邊緣的線條或箭頭尖端來選擇邊緣。X-Ray 會針對下游服務所提供之要求者的要求，顯示長條圖。這些延遲是由申請者記錄的延遲，且包含兩個服務之間的網路連線延遲。

若要解譯 Service details (服務詳細資訊) 長條圖，您可以尋找與長條圖中大多數值差異最大的值。您可將這些「異常值」視為長條圖中的峰值，並檢視特定區域的追蹤資料以調查發生的情況。

若要查看依延遲篩選的追蹤，請在長條圖上選取範圍。按一下您想要選擇的起點位置，並從左到右拖曳，將要包含在追蹤篩選條件中的延遲範圍反白顯示。



選取範圍之後，您可以選擇 Zoom (縮放)，以僅檢視部分長條圖並縮小您的選擇範圍。



一旦您設定好所需的焦點區域時，請選擇 View traces (檢視追蹤)。

使用 X-Ray 洞察

AWS X-Ray 持續分析帳戶中的追蹤資料，以識別應用程式中的緊急問題。當故障率超過預期範圍時，它會創建一個洞察力，以記錄問題並跟踪其影響，直到解決為止。透過深入解析，您可以：

- 識別應用程式問題發生的位置、問題的根本原因以及相關影響。深入解析提供的影響分析可讓您衍生問題的嚴重性和優先順序。
- 在問題隨時間變化時接收通知。洞察通知可以使用 Amazon EventBridge 與您的監控和警示解決方案整合。此整合可讓您根據問題的嚴重性傳送自動電子郵件或警示。

X-Ray 主控台可識別追蹤圖中有持續發生事件的節點。若要查看見解摘要，請選擇受影響的節點。您也可以從左側的導覽窗格中選擇 [深入解析]，以檢視和篩選深入解析。

X-Ray 會在偵測到服務對應的一或多個節點中的異常時建立洞察。此服務會使用統計模型來預測應用程式中服務的預期故障率。在前面的例子中，異常是從 AWS Elastic Beanstalk中增加故障。Elastic Beanstalk 伺服器遇到多個 API 呼叫逾時，造成下游節點異常。

在 X-Ray 主控台中啟用深入解

必須針對您想要使用深入解析功能的每個群組啟用深入解析。您可以從「群組」頁面啟用深入解析。

1. 開啟 X-Ray 主控台。
2. 選取現有群組或透過選擇建立群組建立新群組，然後選取啟用深入解析。如需有關在 X-Ray 主控台中設定群組的更多資訊，請參閱設定群組。
3. 在左側的導覽窗格中，選擇 [深入解析]，然後選擇要檢視的深入解析。

> **Note**
>
> X-Ray 使用 GetInsightSummaries GetInsight GetInsightEvents、和 GetInsightImpactGraph API 操作從見解中擷取資料。若要檢視見解，請使用 AWSXrayReadOnlyAccess IAM 受管政策，或將下列自訂政策新增至您的 IAM 角色：
>
> ```
> {
>    "Version": "2012-10-17",
>    "Statement": [
>        {
>            "Effect": "Allow",
>            "Action": [
>                "xray:GetInsightSummaries",
>                "xray:GetInsight",
>                "xray:GetInsightEvents",
>                "xray:GetInsightImpactGraph"
>            ],
>            "Resource": [
>                "*"
>            ]
>        }
>    ]
> }
> ```
>
> 如需詳細資訊，請參閱 如何與 IAM AWS X-Ray 搭配使用。

啟用見解通知

透過深入解析通知，系統會針對每個見解事件建立通知，例如深入解析建立、發生重大變更或關閉時。客戶可以透過 Amazon EventBridge 事件接收這些通知，並使用條件規則執行 SNS 通知、Lambda 叫用、將訊息發佈到 SQS 佇列或任何目標 EventBridge 支援等動作。我們會盡力發出見解通知，但不能保證。如需有關目標的詳細資訊，請參閱 Amazon EventBridge 目標。

您可以從「群組」頁面為任何啟用見解的群組啟用見解通知。

啟用 X-Ray 群組的通知

1. 開啟 X-Ray 主控台。

2. 選取現有群組或透過選擇建立群組建立新群組，確定已選取 [啟用深入解析]，然後選取 [啟用通知]。如需有關在 X-Ray 主控台中設定群組的更多資訊，請參閱設定群組。

若要設定 Amazon EventBridge 條件規則

1. 打開 Amazon EventBridge 控制台。
2. 導覽至左側導覽列中的「規則」，然後選擇「建立規則」。
3. 提供規則的名稱和說明。
4. 選擇 [事件模式]，然後選擇 [自訂模式]。提供包含`"source": [ "aws.xray" ]`和的模式`"detail-type": [ "AWS X-Ray Insight Update" ]`。以下是一些可能模式的範例。
   - 事件模式，以匹配所有來自 X-Ray 見解的傳入事件：

     ```
     {
     "source": [ "aws.xray" ],
     "detail-type": [ "AWS X-Ray Insight Update" ]
     }
     ```

   - 要與指定的和相符的事件模**`state`**式**`category`**：

     ```
     {
     "source": [ "aws.xray" ],
     "detail-type": [ "AWS X-Ray Insight Update" ],
     "detail": {
            "State": [ "ACTIVE" ],
            "Category": [ "FAULT" ]
       }
     }
     ```

5. 選取並設定當事件符合此規則時要呼叫的目標。
6. (選擇性) 提供標籤，以便更輕鬆地識別並選取此規則。
7. 選擇建立。

> **Note**
>
> X-Ray 洞察通知會將事件傳送至 Amazon EventBridge，而 Amazon 目前不支援客戶受管金鑰。如需詳細資訊，請參閱 AWS X-Ray 中的資料保護。

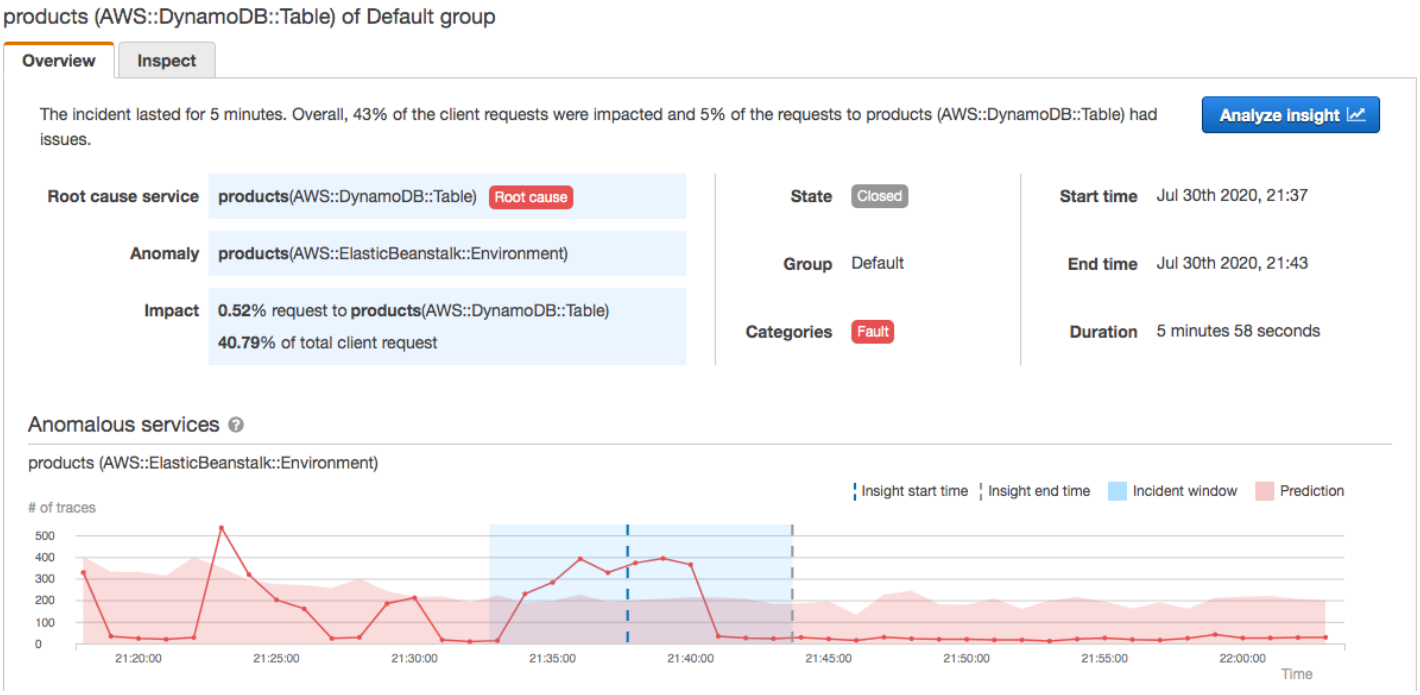
洞察概述

深入解析的概觀頁面會嘗試回答三個關鍵問題：

- 根本問題是什麼？
- 根本原因是什麼？
- 有什麼影響？

「異常服務」區段會顯示每個服務的時間表，說明事件期間的故障率變化。時間表會顯示實體頻帶上覆蓋故障的追蹤數目，根據記錄的流量指出預期的錯誤數目。洞察力的持續時間由「事件」窗口可視化。當 X-Ray 觀察到量度變得異常並在洞察活動時持續存在，事件窗口就會開始。

下列範例顯示造成事件的錯誤增加：

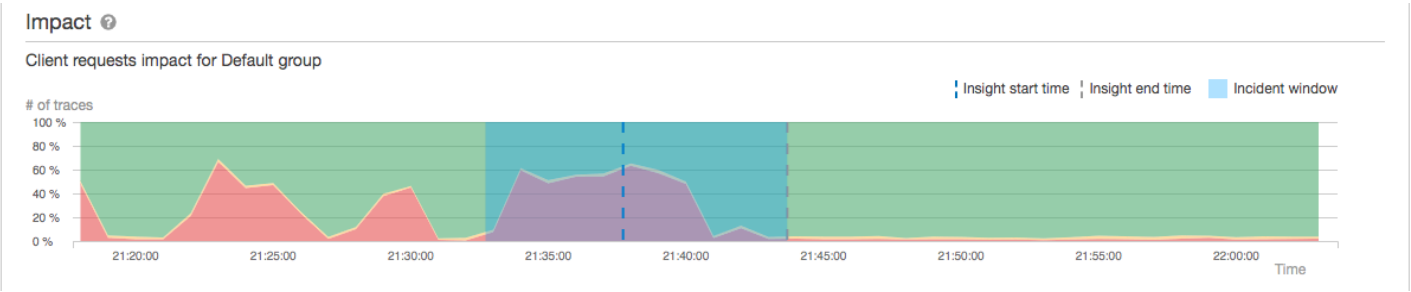


「根本原因」區段會顯示針對根本原因服務和受影響路徑的追蹤對應。您可以選取「根本原因」對映右上角的眼睛圖示，以隱藏未受影響的節點。根本原因服務是 X-Ray 識別異常的最遠下游節點。它可

以代表您已檢測的服務或您的服務使用已檢測的用戶端呼叫的外部服務。例如，如果您使用已檢測的 AWS SDK 用戶端呼叫 Amazon DynamoDB，DynamoDB 的錯誤增加會導致使用 DynamoDB 做為根本原因的洞察。

若要進一步調查根本原因，請選取根本原因圖表上的檢視根本原因詳細資訊。您可以使用「分析」頁面來調查根本原因和相關訊息。如需詳細資訊，請參閱 與 Analytics 主控台互動。

在地圖中繼續上游的錯誤可能會影響多個節點並導致多個異常。如果故障一路傳回提出要求的使用者，結果就是用戶端錯誤。這是跟踪映射的根節點的錯誤。「影響」圖表提供整個群組的用戶端體驗時間表。此體驗是根據下列狀態的百分比計算：「錯誤」、「錯誤」、「節流」和「正常」。



此範例顯示發生事件期間根節點發生錯誤的追蹤增加。下游服務中的事件並不總是與用戶端錯誤的增加相對應。

選擇「分析洞察」會在視窗中開啟 X-Ray Analytics 主控台，您可以在其中深入瞭解產生深入分析的追蹤集。如需詳細資訊，請參閱 與 Analytics 主控台互動。

了解影響

AWS X-Ray 衡量由持續問題引起的影響，作為生成見解和通知的一部分。影響的測量方式有兩種：

- 對 X-Ray 組的影響

- 對根本原因服務的影響

此影響是由指定時間段內失敗或導致錯誤的要求百分比決定。此影響分析可讓您根據特定情況推導問題的嚴重性和優先順序。除了深入解析通知之外，這項影響還可作為主控台體驗的一部分。

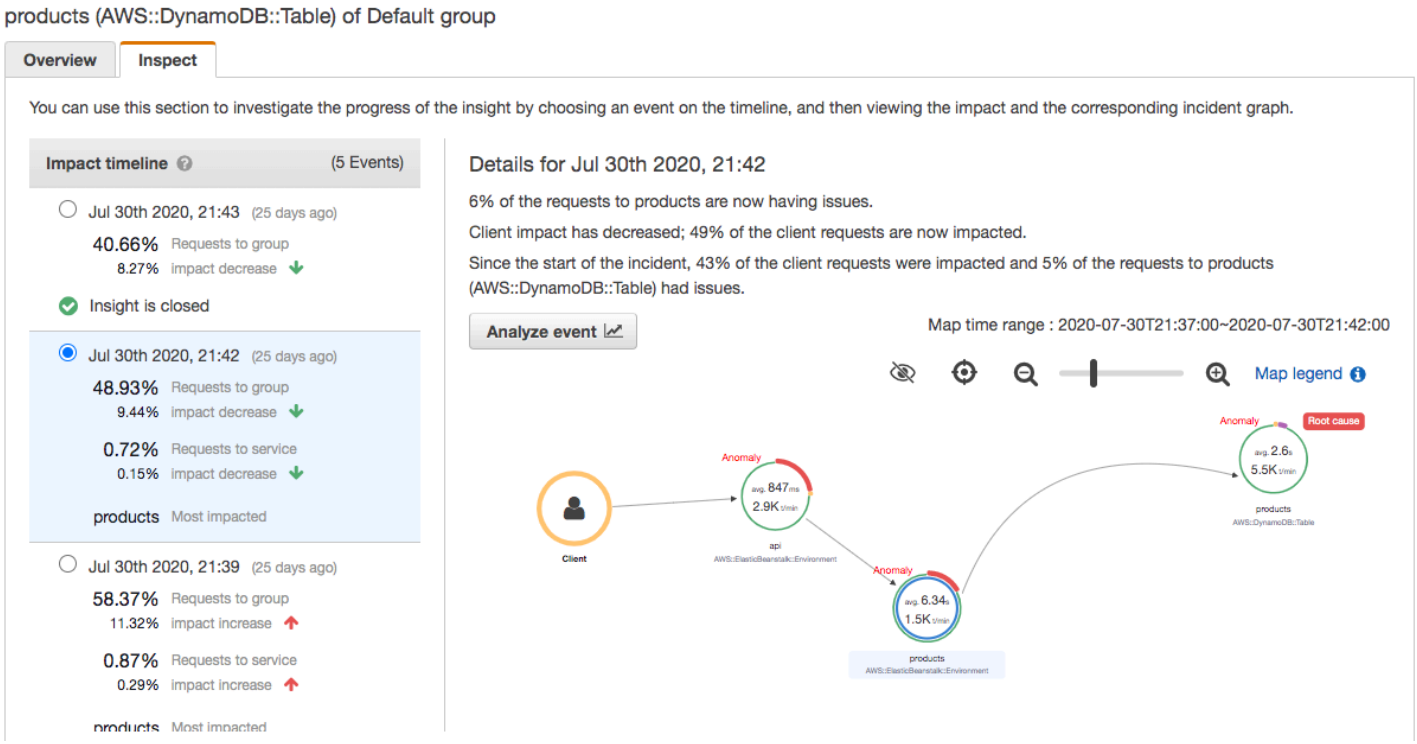
重複資料刪

AWS X-Ray 洞察解決跨多個微服務的重複問題。它會使用異常偵測來判斷問題根本原因的服務，判斷其他相關服務是否因為相同的根本原因而呈現異常行為，並將結果記錄為單一分析。

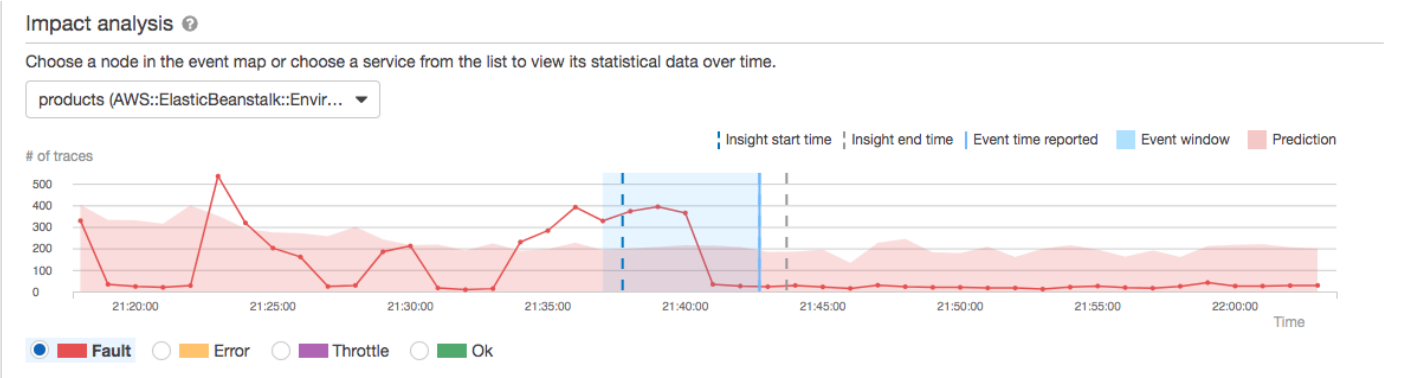
查看洞察力的進度

X-Ray 會定期重新評估見解，直到解決為止，並將每個重要的中間變更記錄為通知，並以 Amazon EventBridge 事件的形式傳送。這可讓您建立程序和工作流程，以判斷問題在一段時間內如何變更，並採取適當的動作，例如傳送電子郵件或使 EventBridge用與警示系統整合。

您可以在「檢查」頁面的「影響時間表」中檢閱事件事件。依預設，時間表會顯示受影響最嚴重的服務，直到您選擇不同的服務為止。



若要查看事件的追蹤對應和圖表，請從影響時間軸選擇它。追蹤對應會顯示應用程式中受事件影響的服務。在「影響分析」下，圖表會顯示所選節點和群組中從屬端的錯誤時間表。



若要深入查看事件所涉及的追蹤，請選擇「檢查」頁面上的「分析事件」。您可以使用「分析」頁面來精簡追蹤清單，並識別受影響的使用者。如需更多詳細資訊，請參閱 與 Analytics 主控台互動。

與 Analytics 主控台互動

AWS X-Ray Analytics 主控台是一種互動式工具，可用來解譯追蹤資料，以便快速瞭解應用程式及其基礎服務的執行情況。主控台可讓您透過互動的回應時間和時間序列圖表，探索、分析和視覺化追蹤。

在 Analytics 主控台中選取選項時，主控台會建構篩選條件以反映所有追蹤的所選子集。您可以按一下與目前追蹤集相關聯之指標和欄位的圖表和面板，使用愈益精細的篩選條件來精簡作用中的資料集。

主題

- 主控台功能
- 回應時間分佈
- 時間序列活動
- 工作流程範例
- 觀察服務圖中的故障
- 識別回應時間高峰
- 檢視有狀態碼標記的所有追蹤
- 檢視在子群組中並與使用者相關聯的所有項目
- 比較兩個具有不同篩選標準的追蹤集
- 識別感興趣的追蹤以及檢視其詳細資訊

主控台功能

X-Ray Analytics 主控台使用下列關鍵功能來分組、篩選、比較和量化追蹤資料。

功能

功能	描述
Groups (群組)	最初選取的群組是 `Default`。若要變更已擷取的群組，請從主要篩選條件表達式搜尋列右側的選單中，選取不同的群組。若要進一步了解群組，請參閱搭配使用篩選條件表達式與群組。
Retrieved traces (已擷取的追蹤)	根據預設，Analytics 主控台會根據所選群組中的所有追蹤產生圖表。已擷取的追蹤代表您工作集中的所有追蹤。您可在此圖標中找到追蹤計數。您套用到主要搜尋列的篩選條件表達式會精簡並更新擷取的追蹤。
Show in charts/Hide from charts (在圖表中顯示/隱藏)	切換比較作用中的群組和已擷取的追蹤。若要針對任何作用中的篩選條件比較群組的相關資料，請選擇 Show in charts (在圖表中顯示)。若要從圖表中移除此檢視，請選擇 Hide from charts (在圖表中隱藏)。
Filtered trace set A (篩選過的追蹤集 A)	透過與圖形和表格的互動，套用篩選器來建立已篩選追蹤集 A 的準則。 套用篩選器時，會在此拼貼中計算適用的繪線數目和擷取總計的繪線百分比。在 Filtered trace set A (篩選過的追蹤集 A) 圖標中，篩選條件填入為標籤，也可自圖標中移除。
Refine (精簡)	此功能會根據套用到追蹤集 A 的篩選條件，更新擷取的追蹤集。縮小擷取的追蹤集範圍會更新根據追蹤集 A 篩選條件擷取之所有追蹤的初步集合。已擷取追蹤的初步集合是群組中所有追蹤的抽樣子集。
Filtered trace set B (篩選過的追蹤集 B)	建立時，已篩選的追蹤集 B 是已篩選追蹤集 A 的複本。 若要比較這兩個追蹤集，請進行新的篩選選取項目，以套用至追蹤集 B，而追蹤集 A 會保持固定。套用篩選條件後，就會在此圖標內

功能	描述
	計算擷取自總量的適用追蹤數目與追蹤百分比。在 Filtered trace set B (篩選過的追蹤集 B) 圖標中，篩選條件填入為標籤，也可自圖標中移除。
Response time root cause entity paths (回應時間根本原因實體路徑)	已記錄實體路徑的表格。X-Ray 確定軌跡中的哪個路徑最有可能導致回應時間的原因。格式指出實體遇到的階層，以回應時間根本原因結束。使用這些資料列篩選重複出現的回應時間故障。如需自訂根本原因篩選條件以及透過 API 取得資料的詳細資訊，請參閱擷取和縮小範圍根本原因分析。
Delta (�)	當追蹤集 A 和追蹤集 B 都為作用中時，新增到指標表的欄。Delta 欄計算追蹤集 A 和追蹤集 B 之間的追蹤百分比差異。

回應時間分佈

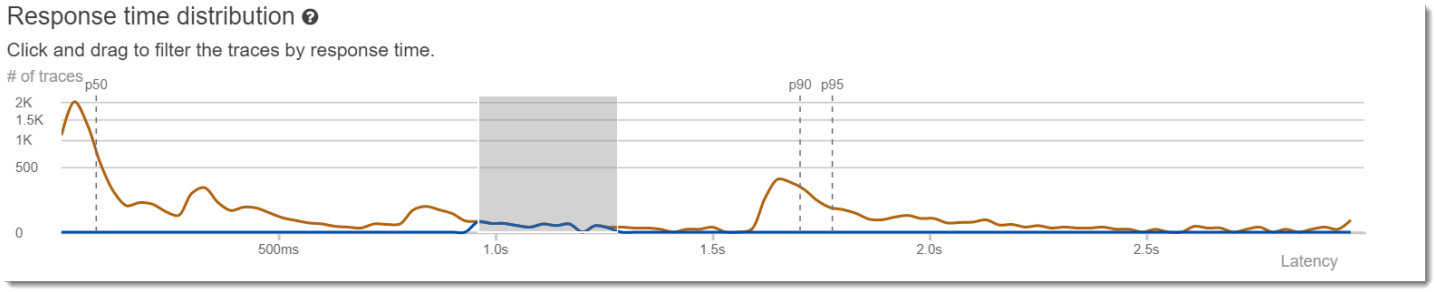
X-Ray Analytics 主控台會產生兩個主要圖表，以協助您視覺化追蹤：回應時間分佈和時間序列活動。本節和下節各提供一個範例，說明閱讀圖表的基本知識。

以下是與回應時間折線圖相關聯的顏色 (時間序列圖使用相同的顏色組合)：

- 群組中的所有痕跡 — 灰色
- 擷取的痕跡 — 橘色
- 已篩選的追蹤集 A — 綠色
- 已篩選的追蹤集 B — 藍色

Example — 響應時間分佈

回應時間分發是顯示特定回應時間追蹤數的圖表。按一下並拖曳，在回應時間分發內選取範圍。這會選取特定回應時間內所有追蹤的 `responseTime` 工作追蹤集，並建立其篩選條件。

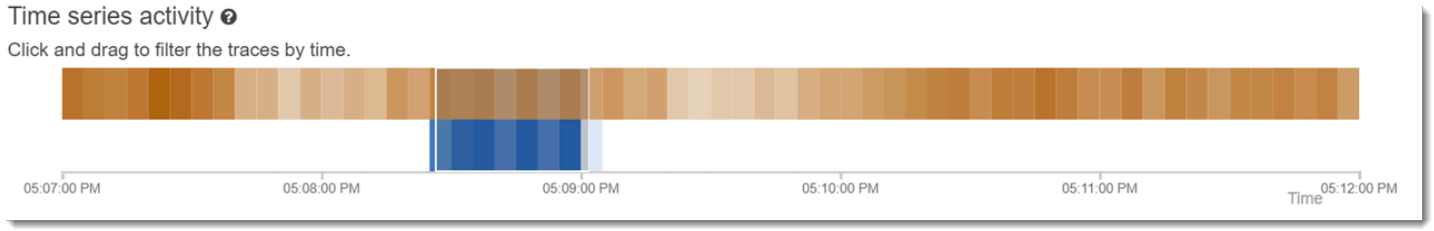


時間序列活動

時間序列活動圖會顯示特定期間的追蹤數目。顏色指標會鏡像回應時間分發的折線圖顏色。活動序列內的顏色區塊愈深愈滿，表示指定時間內的追蹤愈多。

Example — 時間序列活動

按一下並拖曳，在時間序列活動圖中選取範圍。這會選取特定時間範圍內所有追蹤的 `timerange` 工作追蹤集，並建立篩選條件。



工作流程範例

下列範例顯示 X-Ray 分析主控台的常見使用案例。每個範例都會示範主控台體驗的主要功能。因為是群組，所以範例會遵循基本的故障診斷工作流程。這些步驟會逐步解說如何先找出運作狀況不良節點，然後如何與 Analytics 主控台互動，以自動產生比較查詢。一旦透過查詢縮小範圍，您最終會看到特定追蹤的詳細資訊，以判斷是什麼損害伺服器的運作狀況。

觀察服務圖中的故障

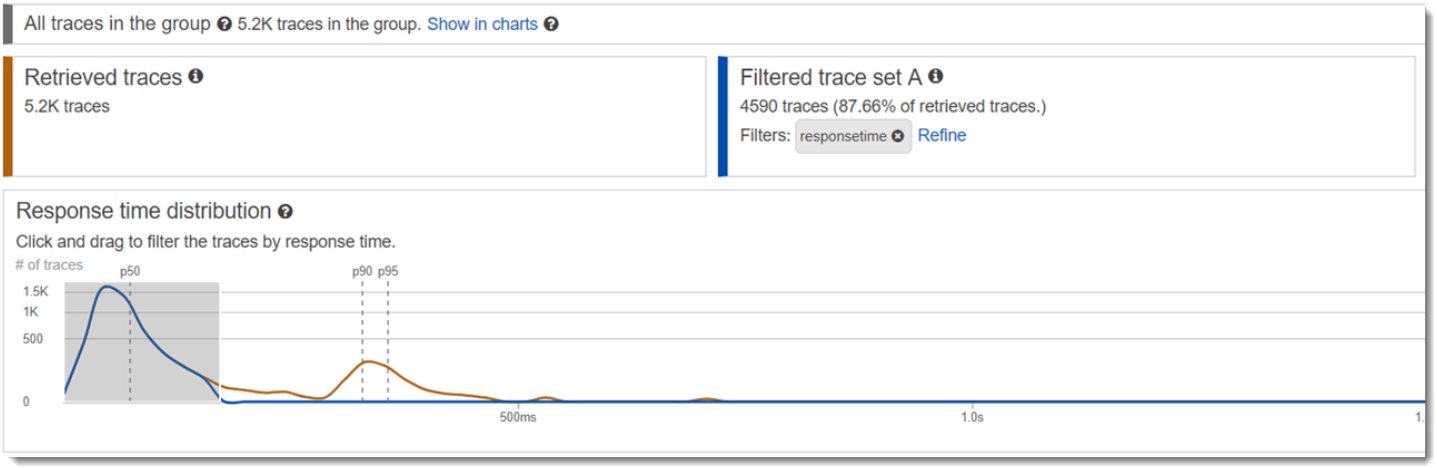
追蹤對應會根據成功呼叫與錯誤和錯誤的比例加上色來指示每個節點的健全狀況。當您在節點上看到紅色百分比時，表示有故障。使用 X-Ray 分析主控台對其進行調查。

若要取得有關如何讀取軌跡地圖的更多資訊，請參閱檢視軌跡貼圖。

識別回應時間高峰

使用回應時間分發，您可以觀察回應時間的高峰。只要選取回應時間的高峰，就會更新圖表下方的資料表，公開所有相關聯的指標，例如狀態碼。

按一下並拖曳時，X-Ray 會選取並建立篩選。它會以灰色陰影顯示在圖形線條的頂部。您現在可以沿著分佈左右拖曳陰影，更新選取範圍和篩選條件。



檢視有狀態碼標記的所有追蹤

您可以使用圖表下方的指標表，深入了解所選高峰的追蹤。按一下 HTTP STATUS CODE 表的資料列，即會自動建立工作資料集的篩選條件。例如，您可以檢視狀態碼為 500 的所有追蹤。這會在追蹤集圖標中建立名為 `http.status` 的篩選條件標籤。

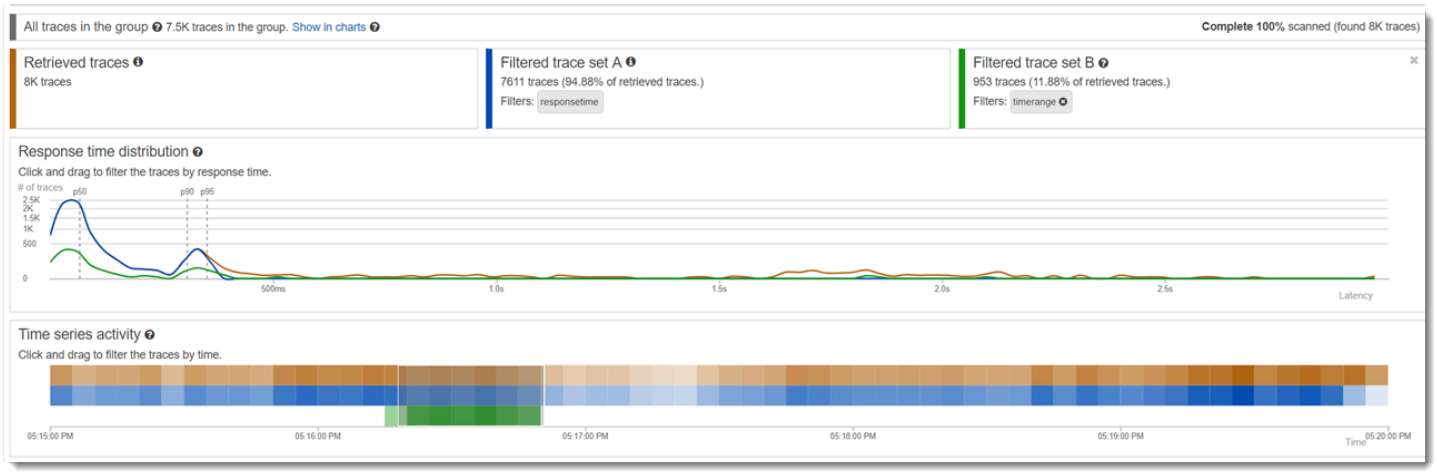
檢視在子群組中並與使用者相關聯的所有項目

深入了解以使用者、URL、回應時間根本原因或其他預先定義的屬性為基礎的錯誤集。例如，若要另行篩選狀態碼為 500 的追蹤集，請選取 USERS (使用者) 表中的資料列。這會讓追蹤集圖標出現兩個篩選條件標籤：之前已指定的`http.status` 以及 `user`。

比較兩個具有不同篩選標準的追蹤集

比較所有不同的使用者及其 POST 請求，尋找其他的差異和關聯性。套用您的第一組篩選條件。它們在回應時間分發中定義為藍色線條。然後選取 Compare (比較)。一開始，這會建立追蹤集 A 的篩選條件副本。

若要繼續，請定義一組新的篩選條件，套用到追蹤集 B。這第二組條件會以綠色線條表示。以下範例示範以藍綠顏色組合的不同線條。



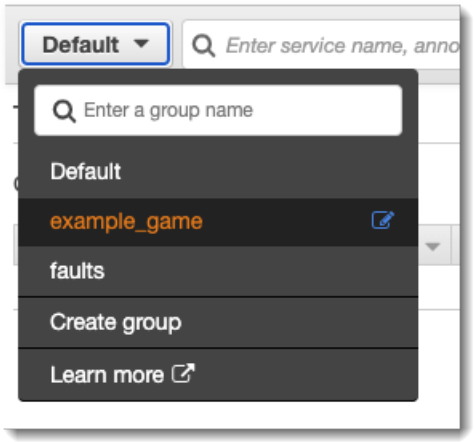
識別感興趣的追蹤以及檢視其詳細資訊

當您使用主控台篩選條件縮小範圍時，指標表下的追蹤清單就會變得更有意義。追蹤清單表將 URL、USER (使用者) 和 STATUS CODE (狀態碼) 的相關資訊結合到一份檢視中。如需更多詳情，請選取此表的資料列開啟追蹤的 detail (詳細資訊) 頁面，檢視其時間軸和原始資料。

設定群組

群組是一系列追蹤，為篩選條件表達式所定義。您可以使用群組產生其他服務圖表並提供 Amazon CloudWatch 指標。您可以使用主 AWS X-Ray 控台或 X-Ray API 來建立和管理服務的群組。本主題說明如何使用 X-Ray 主控台建立和管理群組。如需如何使用 X-Ray API 管理群組的相關資訊，請參閱群組。

您可以建立追蹤對應、追蹤或分析的追蹤群組。當您建立群組時，群組會在所有三個頁面上的群組下拉式功能表中以篩選條件的形式提供：「追蹤對應」、「追蹤」和「分析」。



群組會根據名稱或 Amazon Resource Name (ARN) 進行識別，且包含篩選條件表達式。此服務會比較傳入表達式的追蹤，並依序存放。如需如何建立篩選器運算式的詳細資訊，請參閱使用篩選運算式。

更新群組的篩選條件表達式不會變更已記錄的資料。更新僅適用於後續追蹤。這會導致圖表合併新舊表達式。若要避免這種情況，請刪除目前的群組並建立新群組。

> **Note**
>
> 群組的計費方式是根據符合篩選條件表達式的擷取追蹤。如需詳細資訊，請參閱 AWS X-Ray 定價。

主題

建立群組

> **Note**
>
> 您現在可以從 Amazon CloudWatch 主控台中設定 X-Ray 群組。您也可以繼續使用 X-Ray 控制台。

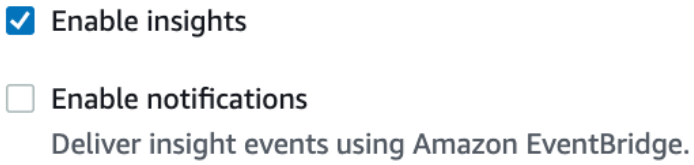
CloudWatch console

1. 請登入 AWS Management Console 並開啟 CloudWatch 主控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在左側導覽窗格中選擇 [設定]。
3. 在 X-Ray 軌跡區段中的群組下選擇檢視設定。
4. 選擇群組清單上方的 [建立群組]。

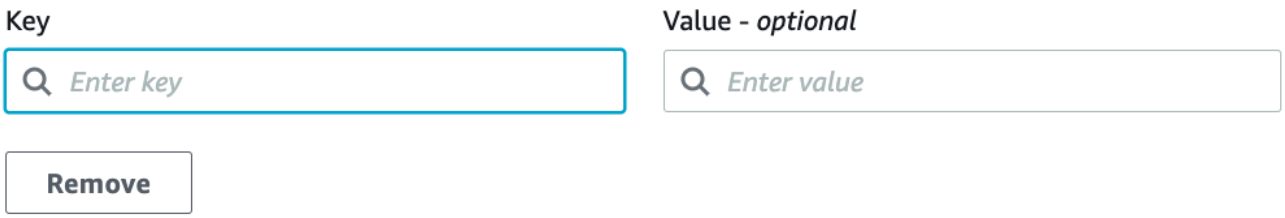
5. 在 [建立群組] 頁面上，輸入群組的名稱。群組名稱最多可包含 32 個字元，並包含英數字元和破折號。群組名稱區分大小寫。
6. 輸入篩選表示式。如需如何建立篩選器運算式的詳細資訊，請參閱使用篩選運算式。在下列範例中，群組會篩選來自服務的錯誤追蹤`api.example.com`。以及回應時間大於或等於 5 秒之服務的要求。

```
fault = true AND http.url CONTAINS "example/game" AND responsetime >= 5
```

7. 在深入解析中，啟用或停用群組的深入解析存取權。如需洞見的詳細資訊，請參閱使用 X-Ray 洞察。



8. 在「標籤」中，選擇「新增標籤」以輸入標籤關鍵字，並選擇性地輸入標籤值。依需要繼續加入其他標籤。標籤鍵必須是唯一的。若要刪除標記，請選擇每個標籤下方的「移除」。如需標籤的詳細資訊，請參閱標籤 X-Ray 取樣規則和羣組。



9. 選擇 Create group (建立群組)。

X-Ray console

1. 登入 AWS Management Console 並開啟 X-Ray 主控台，網址為 https://console.aws.amazon.com/xray/home。
2. 從左側導覽窗格的「群組」頁面，或從下列其中一個頁面上的群組功能表開啟「建立群組」頁面：「追蹤對應」、「追蹤」和「分析」。
3. 在 [建立群組] 頁面上，輸入群組的名稱。群組名稱最多可包含 32 個字元，並包含英數字元和破折號。群組名稱區分大小寫。

4. 輸入篩選表示式。如需如何建立篩選器運算式的詳細資訊，請參閱使用篩選運算式。在下列範例中，群組會篩選來自服務的錯誤追蹤`api.example.com`。以及回應時間大於或等於 5 秒之服務的要求。

```
fault = true AND http.url CONTAINS "example/game" AND responsetime >= 5
```

5. 在深入解析中，啟用或停用群組的深入解析存取權。如需洞見的詳細資訊，請參閱使用 X-Ray 洞察。

6. 在標籤中，輸入標籤鍵，並選擇性地輸入標籤值。當您加入標籤時，會出現一個新行供您輸入其他標籤。標籤鍵必須是唯一的。若要刪除標記，請選擇標記列末端的 X。如需標籤的詳細資訊，請參閱標籤 X-Ray 取樣規則和羣組。

7. 選擇 Create group (建立群組)。

套用群組

CloudWatch console

1. 請登入 AWS Management Console 並開啟 CloudWatch 主控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 從「X-Ray」追蹤下的導覽窗格開啟下列其中一個頁面：
   - 軌跡貼圖
   - 追蹤
3. 在「按 X-Ray 群組篩選」篩選條件中輸入群組名稱。頁面上顯示的資料會變更，以符合群組中設定的篩選器運算式。

X-Ray console

1. 登入 AWS Management Console 並開啟 X-Ray 主控台，網址為 https://console.aws.amazon.com/xray/home。
2. 從導覽窗格開啟下列其中一個頁面：
   - 軌跡貼圖
   - 追蹤
   - 分析
3. 在群組功能表上，選擇您在其中建立的群組the section called “建立群組”。頁面上顯示的資料會變更，以符合群組中設定的篩選器運算式。

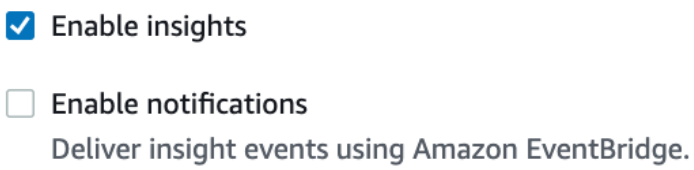
編輯群組

CloudWatch console

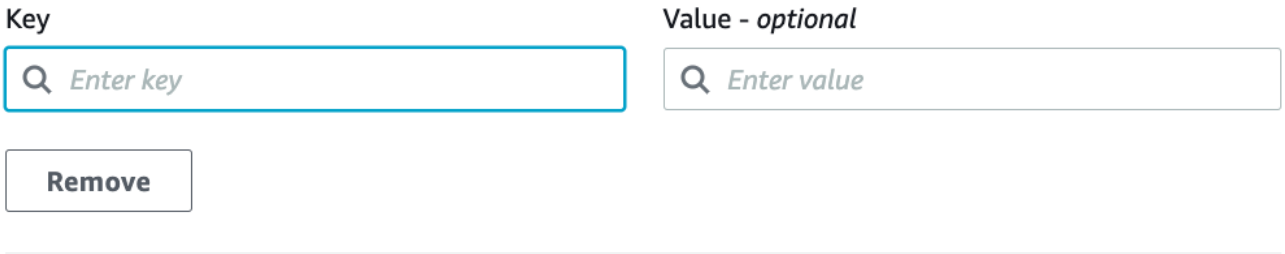
1. 請登入 AWS Management Console 並開啟 CloudWatch 主控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在左側導覽窗格中選擇 [設定]。
3. 在 X-Ray 軌跡區段中的群組下選擇檢視設定。
4. 從「群組」區段中選擇群組，然後選擇 「編輯」。
5. 雖然您無法重新命名群組，但您可以更新篩選器運算式。如需如何建立篩選器運算式的詳細資訊，請參閱使用篩選運算式。在下列範例中，群組篩選來自要求 URL 位址所包含之服務`api.example.com`的錯誤追蹤`example/game`，而要求的回應時間大於或等於 5 秒。

   ```
   fault = true AND http.url CONTAINS "example/game" AND responsetime >= 5
   ```

6. 在深入解析中，啟用或停用群組的深入解析存取權。如需洞見的詳細資訊，請參閱使用 X-Ray 洞察。

7. 在「標籤」中，選擇「新增標籤」以輸入標籤關鍵字，並選擇性地輸入標籤值。依需要繼續加入其他標籤。標籤鍵必須是唯一的。若要刪除標記，請選擇每個標籤下方的「移除」。如需標籤的詳細資訊，請參閱標籤 X-Ray 取樣規則和羣組。



8. 當您完成群組更新時，請選擇 [更新群組]。

X-Ray console

1. 登入 AWS Management Console 並開啟 X-Ray 主控台，網址為 https://console.aws.amazon.com/xray/home。
2. 執行下列任一項作業，開啟 [編輯群組] 頁面。

   a. 在 [群組] 頁面上，選擇要編輯群組的名稱。

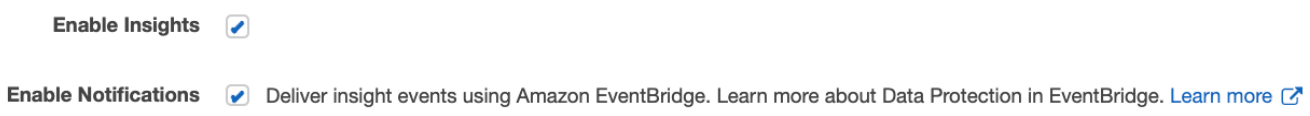
   b. 在下列其中一個頁面的群組功能表上，指向群組，然後選擇 [編輯]。

   - 軌跡貼圖
   - 追蹤
   - 分析

3. 雖然您無法重新命名群組，但您可以更新篩選器運算式。如需如何建立篩選器運算式的詳細資訊，請參閱使用篩選運算式。在下列範例中，群組篩選來自要求 URL 位址所包含之服務`api.example.com`的錯誤追蹤`example/game`，而要求的回應時間大於或等於 5 秒。

```
fault = true AND http.url CONTAINS "example/game" AND responsetime >= 5
```

4. 在深入解析中，啟用或停用群組的見解和深入解析通知。如需洞見的詳細資訊，請參閱使用 X-Ray 洞察。



5. 在標籤中，編輯標籤關鍵字和值。標籤鍵必須是唯一的。標籤值是選擇性的；您可以視需要刪除值。若要刪除標記，請選擇標記列末端的 X。如需標籤的詳細資訊，請參閱標籤 X-Ray 取樣規則和羣組。

6. 當您完成群組更新時，請選擇 [更新群組]。

複製群組

複製群組會建立具有現有群組之篩選運算式和標記的新群組。當您複製群組時，新群組的名稱與複製來源群組的名稱相同，並-clone附加在名稱之後。

CloudWatch console

1. 請登入 AWS Management Console 並開啟 CloudWatch 主控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在左側導覽窗格中選擇 [設定]。
3. 在 X-Ray 軌跡區段中的群組下選擇檢視設定。
4. 從「群組」區段中選擇群組，然後選擇「複製」。
5. *# [####] ###############* -clone選擇性地輸入群組的新名稱。群組名稱最多可包含 32 個字元，並包含英數字元和破折號。群組名稱區分大小寫。
6. 您可以保留現有群組中的篩選器運算式，也可以選擇性地輸入新的篩選運算式。如需如何建立篩選器運算式的詳細資訊，請參閱使用篩選運算式。在下列範例中，群組會篩選來自服務的錯誤追蹤api.example.com。以及回應時間大於或等於 5 秒之服務的要求。

```
service("api.example.com") { fault = true OR responsetime >= 5 }
```

7. 如果需要，請在標籤中編輯標籤關鍵字和值。標籤鍵必須是唯一的。標籤值是可選的；如果需要，您可以刪除值。若要刪除標記，請選擇標記列末端的 X。如需標籤的詳細資訊，請參閱標籤 X-Ray 取樣規則和羣組。
8. 選擇 Create group (建立群組)。

X-Ray console

1. 登入 AWS Management Console 並開啟 X-Ray 主控台，網址為 https://console.aws.amazon.com/xray/home。
2. 從左側導覽窗格開啟 [群組] 頁面，然後選擇要複製的群組名稱。
3. 從「動作」功能表選擇「複製群組」。
4. *# [####] ###############* `-clone`選擇性地輸入群組的新名稱。群組名稱最多可包含 32 個字元，並包含英數字元和破折號。群組名稱區分大小寫。
5. 您可以保留現有群組中的篩選器運算式，也可以選擇性地輸入新的篩選運算式。如需如何建立篩選器運算式的詳細資訊，請參閱使用篩選運算式。在下列範例中，群組會篩選來自服務的錯誤追蹤`api.example.com`。以及回應時間大於或等於 5 秒之服務的要求。

   ```
   service("api.example.com") { fault = true OR responsetime >= 5 }
   ```

6. 如果需要，請在標籤中編輯標籤關鍵字和值。標籤鍵必須是唯一的。標籤值是可選的；如果需要，您可以刪除值。若要刪除標記，請選擇標記列末端的 X。如需標籤的詳細資訊，請參閱標籤 X-Ray 取樣規則和羣組。
7. 選擇 Create group (建立群組)。

刪除群組

請依照本節中的步驟刪除群組。您無法刪除「預設」群組。

CloudWatch console

1. 請登入 AWS Management Console 並開啟 CloudWatch 主控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在左側導覽窗格中選擇 [設定]。
3. 在 X-Ray 軌跡區段中的群組下選擇檢視設定。
4. 從「群組」區段中選擇群組，然後選擇「刪除」。
5. 當系統提示您確認時，請選擇「刪除」。

X-Ray console

1. 登入 AWS Management Console 並開啟 X-Ray 主控台，網址為 https://console.aws.amazon.com/xray/home。

2. 從左側導覽窗格開啟 [群組] 頁面，然後選擇要刪除的群組名稱。
3. 在 [動作] 功能表上，選擇 [刪除群組]。
4. 當系統提示您確認時，請選擇「刪除」。

在 Amazon 中查看群組指標 CloudWatch

建立群組後，傳入的追蹤會在儲存在 X-Ray 服務中時，根據群組的篩選器運算式進行檢查。符合每個條件的追蹤數量指標 CloudWatch 每分鐘都會發佈到 Amazon。選擇 「編輯群組」 頁面的檢視測量結果，可開啟「測量結果」頁面的 CloudWatch主控台 如需有關如何使用指 CloudWatch 標的詳細資訊，請參閱 Amazon 使用 CloudWatch 者指南中的使用 Amazon 指 CloudWatch標。

CloudWatch console

1. 請登入 AWS Management Console 並開啟 CloudWatch 主控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在左側導覽窗格中選擇 [設定]。
3. 在 X-Ray 軌跡區段中的群組下選擇檢視設定。
4. 從「群組」區段中選擇群組，然後選擇 「編輯」。
5. 在 「編輯群組」 頁面上，選擇檢視測量結果。

   主 CloudWatch 控台「測量結果」頁面會在新標籤中開啟。

X-Ray console

1. 登入 AWS Management Console 並開啟 X-Ray 主控台，網址為 https://console.aws.amazon.com/xray/home。
2. 從左側導覽窗格開啟「群組」頁面，然後選擇您要檢視其量度的群組名稱。
3. 在 「編輯群組」 頁面上，選擇檢視測量結果。

   主 CloudWatch 控台「測量結果」頁面會在新標籤中開啟。

設定 取樣規則

您可以使用主 AWS X-Ray 控台來設定服務的取樣規則。X-Ray SDK 以及支援 AWS 服務 使用取樣配置的主動追蹤使用取樣規則來決定要記錄哪些要求。

主題

設定 取樣規則

您可以針對下列使用案例設定取樣：

- API Gateway 入口點 — API Gateway 支援取樣和主動追蹤。若要在 API 階段上啟用主動追蹤，請參閱 Amazon API Gateway 主動追蹤支援 AWS X-Ray。
- AWS AppSync— AWS AppSync 支持採樣和主動跟踪。若要啟用 AWS AppSync 要求的作用中追蹤，請參閱使用 AWS X-Ray 追蹤。
- 運算平台上的儀器 X-Ray 開發套件 — 使用 Amazon EC2、Amazon ECS 等運算平台時，如果應用程式已使用最新的 X-Ray 開發套件進行檢測 AWS Elastic Beanstalk，則支援取樣。

自訂抽樣規則

透過自訂取樣規則，您可以控制記錄的資料量。您也可以在不修改或重新部署程式碼的情況下修改取樣行為。取樣規則會告訴 X-Ray SDK 要針對一組準則記錄多少個要求。根據預設，X-Ray SDK 會每秒記錄第一個要求，以及任何其他要求的百分之五。每秒一個請求是儲槽。這可確保只要服務持續提供請求，每秒都會記錄至少一個追蹤。5% 是超過儲槽大小的額外請求抽樣「速率」。

您可以將 X-Ray SDK 設定為從您的程式碼中包含的 JSON 文件讀取取樣規則。但是，當您執行服務的多個執行個體時，每個執行個體都會獨立執行抽樣。這會導致抽樣請求的整體百分比增加，因為所有執行個體的儲槽都會加在一起。此外，若要更新本機取樣規則，您必須重新部署程式碼。

透過在 X-Ray 主控台中定義取樣規則，並將 SDK 設定為從 X-Ray 服務讀取規則，就可以避免這兩個問題。服務會管理每個規則的儲槽，並根據執行中的執行個體數，將配額指派給每個服務執行個體來平均分散儲槽。儲槽限制是根據您設定的規則所計算。由於規則是在服務中配置的，因此您無需進行其他部署即可管理規則。

> **Note**
>
> X-Ray 在應用採樣規則時採用盡最大努力的方法，在某些情況下，有效採樣率可能與配置的採樣規則不完全匹配。但是，隨著時間的推移，採樣的請求數量應該接近配置的百分比。

您現在可以從 Amazon CloudWatch 主控台設定 X-Ray 取樣規則。您也可以繼續使用 X-Ray 控制台。

CloudWatch console

若要在 CloudWatch 主控台中設定取樣規則

1. 請登入 AWS Management Console 並開啟 CloudWatch 主控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在左側導覽窗格中選擇 [設定]。
3. 在 X-Ray 軌跡區段中的取樣規則下選擇檢視設定。
4. 若要建立規則，請選擇 Create sampling rule (建立抽樣規則)。

   若要編輯規則，請選擇規則，然後選擇 「編輯」進行編輯。

   若要刪除規則，請選擇規則，然後選擇「刪除」將其刪除。

X-Ray console

在 X-Ray 主控台中設定取樣規則

1. 開啟 X-Ray 主控台。
2. 在左側導覽窗格中選擇 [取樣]。
3. 若要建立規則，請選擇 Create sampling rule (建立抽樣規則)。

   若要編輯規則，請選擇規則的名稱。

   若要刪除規則，請選擇規則並使用 Actions (動作) 功能表來刪除它。

抽樣規則選項

下列選項可用於每個規則。字串值可以使用萬用字元來以符合單一字元 (?) 或零或多個字元 (*)。

抽樣規則選項

- 規則名稱 (字串) — 規則的唯一名稱。
- 優先順序 (介於 1 到 9999 之間的整數) — 取樣規則的優先順序。服務會以遞增的優先順序來評估規則，並使用符合的第一個規則來決定取樣決策。
- 儲存器（非負整數）— 應用固定費率之前，每秒儀器匹配請求的固定數量。儲槽不會直接用於服務，而是集體套用至使用該規則的所有服務。
- 速率（0 到 100 之間的整數）— 儲存器用盡後，與儀器匹配請求的百分比。在主控台中設定取樣規則時，請選擇 0 到 100 之間的百分比。使用 JSON 文件在用戶端 SDK 中設定取樣規則時，請提供介於 0 到 1 之間的百分比值。
- 服務名稱 (字串) — 已檢測服務的名稱，當它顯示在追蹤對映中。
  - X-Ray SDK — 您在錄像機上配置的服務名稱。
  - Amazon API Gateway —*api-name*/*stage*.
- 服務類型 (字串) — 顯示在追蹤對映中的服務類型。對於 X-Ray SDK，請通過應用適當的插件來設置服務類型：
  - `AWS::ElasticBeanstalk::Environment`-一個 AWS Elastic Beanstalk 環境（插件）。
  - `AWS::EC2::Instance`-一個 Amazon EC2 實例（插件）。
  - `AWS::ECS::Container`— Amazon ECS 容器 (插件).
  - `AWS::APIGateway::Stage`— 一個 Amazon API Gateway 階段.
  - `AWS::AppSync::GraphQLAPI` -一個 AWS AppSync API 請求。
- 主機 (字串) — 來自 HTTP 主機標頭的主機名稱。
- HTTP 方法 (字串) — HTTP 要求的方法。
- URL 路徑 (字串) — 要求的 URL 路徑。
  - X-Ray SDK — HTTP 請求網址的路徑部分。
- 資源 ARN (字串) — 執行服務之 AWS 資源的 ARN。
  - X-Ray SDK — 不支援。軟體開發套件僅能使用 Resource ARN (資源 ARN) 設為 * 的規則。
  - Amazon API Gateway-階段 ARN。
- (選擇性) 屬性 (鍵和值) — 做出取樣決定時已知的區段屬性。
  - X-Ray SDK — 不支援。軟體開發套件會忽略指定屬性的規則。
  - Amazon API Gateway — 來自原始 HTTP 請求的標頭。

抽樣規則範例

Example -默認規則，沒有水庫和低速率

您可以修改預設規則的儲槽和速率。預設規則會套用至不符合任何其他規則的請求。

- 水庫：**0**
- 速率:**5**(**0.05**如果使用 JSON 文件設定)

Example — 調試規則以跟踪有問題的路由的所有請求

會暫時套用高優先順序的規則來進行除錯。

- 規則名稱：**DEBUG - history updates**
- 優先順序：**1**
- 水庫：**1**
- 速率:**100**(**1**如果使用 JSON 文件設定)
- 服務名稱：**Scorekeep**
- Service type (服務類型)：*****
- 主持人：*****
- HTTP 方法：**PUT**
- 網址路徑：**/history/***
- 資源庫：*****

Example -帖子的最低利率更高

- 規則名稱：**POST minimum**
- 優先順序：**100**
- 水庫：**10**
- 速率:**10**(**.1**如果使用 JSON 文件設定)
- 服務名稱：*****
- Service type (服務類型)：*****
- 主持人：*****

- HTTP 方法：**POST**
- 網址路徑：*****
- 資源庫：*****

將您的服務設定為使用抽樣規則

X-Ray SDK 需要額外的設定，才能使用您在主控台中設定的取樣規則。如需設定抽樣策略的詳細資訊，請參閱適用您語言的組態主題：

- Java：抽樣規則
- 前往：抽樣規則
- Node.js: 抽樣規則
- Python：抽樣規則
- 紅寶石：抽樣規則
- .NET：抽樣規則

如需 API Gateway 的資訊，請參閱Amazon API Gateway 主動追蹤支援 AWS X-Ray。

檢視抽樣結果

X-Ray 主控台取樣頁面會顯示有關您的服務如何使用每個取樣規則的詳細資訊。

Trend (趨勢) 資料行會顯示過去幾分鐘內使用規則的方式。每個資料行都會顯示 10 秒間的統計資料。

抽樣統計資料

- 符合規則總計：符合此規則的要求數目。此數量不包含本來會和此規則相符，但是卻先符合高優先順序規則的請求。
- 抽樣總數：記錄的請求數目。
- 以固定費率取樣：套用規則的固定費率所取樣的要求數目。
- 使用儲存器限制抽樣：使用 X-Ray 指派的配額取樣的要求數目。
- 從水庫借來的：通過從水庫借用抽樣的請求數量。服務第一次符合規則的要求時，尚未透過 X-Ray 指派配額。但是，如果儲存器至少為 1，則服務每秒借用一條追蹤，直到 X-Ray 指派配額為止。

如需抽樣統計資料和服務使用抽樣規則方式的詳細資訊，請參閱使用取樣規則搭配 X-Ray API。

後續步驟

您可以使用 X-Ray API 來管理取樣規則。透過 API，您可以透過編寫程式設計的方式在排程上建立及更新規則，或是回應警示或通知。如需說明及其他規則範例，請參閱使用 AWS X-Ray API 設定抽樣、分組和加密設定。

X-Ray SDK AWS 服務 也使用 X-Ray API 讀取取樣規則、報告取樣結果，以及取得取樣目標。當要求符合 X-Ray 尚未為服務指派配額的規則時，服務必須追蹤套用每個規則的頻率、根據優先順序評估規則，以及從儲存器借用。如需服務使用 API 進行抽樣的詳細資訊，請參閱使用取樣規則搭配 X-Ray API。

當 X-Ray SDK 呼叫取樣 API 時，它會使用 X-Ray 精靈做為代理伺服器。若您已使用 TCP 連接埠 2000，現在您可以設定精靈，在不同的連接埠執行代理。如需詳細資訊，請參閱 配置 AWS X-Ray 守護進程。

主控台深層連結

您可以使用路由和查詢深入連結至特定追蹤，或篩選的追蹤和追蹤對應檢視。

主控台頁面

- 歡迎頁面 — x ra/ 首頁 #/歡迎
- 開始使用 — x ra/ 首頁 #/開始
- 跟踪圖-x ray/家 # / 服務地圖
- 痕跡 — x ra/ 家 #/ 痕跡

追蹤

您可以為個別追蹤的時間表檢視、原始檢視及映射檢視建立連結。

追蹤時間軸 — `xray/home#/traces/`*`trace-id`*

原始追蹤資料 — `xray/home#/traces/`*`trace-id`*`/raw`

Example — 原始跟踪數據

```
https://console.aws.amazon.com/xray/home#/traces/1-57f5498f-d91047849216d0f2ea3b6442/
raw
```

篩選條件表達式

連結到追蹤的篩選清單。

已篩選的追蹤檢視 — xray/home#/traces?filter=*filter-expression*

Example -過濾器表達式

```
https://console.aws.amazon.com/xray/home#/traces?filter=service("api.amazon.com")
 { fault = true OR responsetime > 2.5 } AND annotation.foo = "bar"
```

Example -過濾器表達式（URL 編碼）

```
https://console.aws.amazon.com/xray/home#/traces?filter=service(%22api.amazon.com
%22)%20%7B%20fault%20%3D%20true%20OR%20responsetime%20%3E%202.5%20%7D%20AND
%20annotation.foo%20%3D%20%22bar%22
```

如需篩選條件表達式的詳細資訊，請參閱使用篩選運算式。

時間範圍

指定時間長度或開始及結束時間 (格式為 ISO8601)。時間範圍以 UTC 為單位，最長可達 6 小時。

時間長度 — xray/home#/*page*?timeRange=*range-in-minutes*

Example — 最後一小時的跟踪圖

```
https://console.aws.amazon.com/xray/home#/service-map?timeRange=PT1H
```

開始和結束時間 — xray/home#/*page*?timeRange=*start*~*end*

Example — 時間範圍精確到秒

```
https://console.aws.amazon.com/xray/home#/traces?
timeRange=2023-7-01T16:00:00~2023-7-01T22:00:00
```

Example — 時間範圍精確到分鐘

```
https://console.aws.amazon.com/xray/home#/traces?
timeRange=2023-7-01T16:00~2023-7-01T22:00
```

區域

指定 AWS 區域 要連結至該區域中的頁面。若您未指定區域，主控台會將您重新導向至最後一個前往的區域。

地區 — xray/home**?region=*region***#/*page*

Example — 美國西部 (俄勒岡州) 的追蹤地圖 (美國西部 -2)

```
https://console.aws.amazon.com/xray/home?region=us-west-2#/service-map
```

當您將 Region 與其他查詢參數加入時，Region 查詢會在雜湊之前進行，而 X 射線特定的查詢會在頁面名稱之後進行。

Example — 美國西部（俄勒岡州）最後一小時的跟踪地圖（美國西部 -2）

```
https://console.aws.amazon.com/xray/home?region=us-west-2#/service-map?timeRange=PT1H
```

合併

Example — 帶有持續時間過濾器的最近痕跡

```
https://console.aws.amazon.com/xray/home#/traces?timeRange=PT15M&filter=duration%20%3E
%3D%205%20AND%20duration%20%3C%3D%208
```

輸出

- 頁面 — 追蹤
- 時間範圍 — 最近 15 分鐘
- 過濾器-持續時間 >= 5 和持續時間 <= 8

AWS X-Ray 守護進

> **Note**
>
> 您現在可以使用 CloudWatch 代理程式從 Amazon EC2 執行個體和內部部署伺服器收集指標、日誌和追蹤。 CloudWatch 代理程式版本 1.300025.0 及更新版本可以從OpenTelemetry或 X-Ray 用戶端 SDK 收集追蹤，並將它們傳送至 X-Ray。使用 CloudWatch 代理程式代替發行 AWS 版 OpenTelemetry (ADOT) 收集器或 X-Ray 精靈收集追蹤，可協助您減少管理的代理程式數量。如需詳細資訊，請參閱 CloudWatch 使用指南中的CloudWatch 代理程式主題。

AWS X-Ray 精靈是一種軟體應用程式，可偵聽 UDP 連接埠 2000 上的流量、收集原始區段資料，並將其轉送至 API。 AWS X-Ray 該守護程序與 AWS X-Ray SDK 一起工作，並且必須運行，以便 SDK 發送的數據可以訪問 X-Ray 服務。X-Ray 守護程序是一個開源項目。您可以關注該項目並在以下位置提交問題並提取請求 GitHub：gi thub.com/aws/ aws-xray-daemon

開啟 AWS Lambda 並 AWS Elastic Beanstalk使用這些服務與 X-Ray 整合來執行守護程式。Lambda 會在為取樣要求叫用函數時自動執行常駐程式。在 Elastic Beanstalk 上，使用XRayEnabled組態選項在環境中的執行個體上執行精靈。如需詳細資訊，請參閱

若要在本機、內部部署或其他地方執行 X-Ray 精靈 AWS 服務，請下載並執行它，然後授與其上傳區段文件至 X-Ray 的權限。

下載精靈

您可以從 Amazon S3、Amazon ECR 或碼頭集線器下載常駐程式，然後在本機執行，或在啟動時將其安裝在 Amazon EC2 執行個體上。

Amazon S3

X-Ray 守護程序安裝程序和可執行

- Linux（可執行文件）-aws-xray-daemon-linux-3.x.zip（簽名）
- (轉速安裝程式) — aws-xray-daemon-3.x.rpm
- Linux (DEB 安裝程式) — aws-xray-daemon-3.x.deb
- （ARM64，可執行文件）-aws-xray-daemon-linux-arm64-3.x.zip（簽名）
- Linux (ARM64, 轉速安裝程式) — aws-xray-daemon-arm64-3.x.rpm

- Linux 系統 (ARM64, 安裝程式) — aws-xray-daemon-arm64-3.x.deb
- OS X（可執行文件）-aws-xray-daemon-macos-3.x.zip（簽名）
- 視窗（可執行文件）-aws-xray-daemon-windows-process-3.x.zip（SIG）
- 視窗 (服務) — aws-xray-daemon-windows-service-3.x.zip(SIG)

這些連結永遠指向最新的 3.x 版本的守護程式。若要下載特定版本，請將 3.x 取代為版本編號。例如 2.1.0。

X-Ray 資產會複製到每個支援區域中的儲存貯體。若要使用與您或您 AWS 資源最近的儲存貯體，請將上述連結中的區域取代為您的區域。

```
https://s3.us-west-2.amazonaws.com/aws-xray-assets.us-west-2/xray-daemon/aws-xray-daemon-3.x.rpm
```

Amazon ECR

從版本 3.2.0 開始，該守護進程可以在 Amazon EC R 上找到。在提取映像之前，您應該向 Amazon ECR 公共註冊表驗證 docker 客戶端。

執行下列命令來提取最新發行的 3.x 版本標籤：

```
docker pull public.ecr.aws/xray/aws-xray-daemon:3.x
```

先前版本或 Alpha 版本可以取代為alpha或3.x特定版本號碼來下載。不建議在生產環境中使用帶有 alpha 標籤的守護程序映像。

Docker Hub

該守護進程可以在碼頭集線器上找到。若要下載最新發行的 3.x 版本，請執行下列命令：

```
docker pull amazon/aws-xray-daemon:3.x
```

以前版本的守護程序可以通過替換所需3.x的版本來釋放。

驗證精靈存檔的簽章

GPG 簽章檔案會針對以 ZIP 存檔形式壓縮的精靈資產包含在其中。公有金鑰位於此處：aws-xray.gpg。

您可以使用公有金鑰來驗證精靈的 ZIP 存檔是否為原始狀態且未經修改。首先，透過 GnuPG 匯入公有金鑰。

匯入公有金鑰

1. 下載公開金鑰。

```
$ BUCKETURL=https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2
$ wget $BUCKETURL/xray-daemon/aws-xray.gpg
```

2. 將公開金鑰匯入至您的 keyring。

```
$ gpg --import aws-xray.gpg
gpg: /Users/me/.gnupg/trustdb.gpg: trustdb created
gpg: key 7BFE036BFE6157D3: public key "AWS X-Ray <aws-xray@amazon.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

使用匯入的金鑰驗證精靈 ZIP 存檔的簽章。

驗證存檔的簽章

1. 下載存檔及簽章檔案

```
$ BUCKETURL=https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2
$ wget $BUCKETURL/xray-daemon/aws-xray-daemon-linux-3.x.zip
$ wget $BUCKETURL/xray-daemon/aws-xray-daemon-linux-3.x.zip.sig
```

2. 執行 `gpg --verify` 以驗證簽章。

```
$ gpg --verify aws-xray-daemon-linux-3.x.zip.sig aws-xray-daemon-linux-3.x.zip
gpg: Signature made Wed 19 Apr 2017 05:06:31 AM UTC using RSA key ID FE6157D3
gpg: Good signature from "AWS X-Ray <aws-xray@amazon.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: EA6D 9271 FBF3 6990 277F  4B87 7BFE 036B FE61 57D3
```

請注意有關信任的警告。只有您或您信任的人員已進行簽署，金鑰才會被視為受信任。這不表示該簽章是無效，只是您尚未驗證該公開金鑰。

執行精靈

從命令列於本機執行精靈。使用 -o 選向來在本機模式中執行，以及 -n 來設定區域。

```
~/Downloads$ ./xray -o -n us-east-2
```

如需詳細的平台限定說明，請參閱以下主題：

- Linux (本機) — 在 Linux 上執行 X-Ray 精靈
- 視窗 (本機) — 在 Windows 上執行 X-Ray 精靈
- Elastic Beanstalk — 在上執行 X-Ray 協助程式AWS Elastic Beanstalk
- Amazon EC2 — 在 Amazon EC2 上執行 X-Ray 常設程式
- Amazon ECS — X-Ray Amazon

您可以使用命令列選項或組態檔來更進一步自訂精靈的行為。如需詳細資訊，請參閱 配置 AWS X-Ray 守護進程。

授予守護進程將數據發送到 X-Ray 的權限

X-Ray 精靈會使用 AWS SDK 將追蹤資料上傳至 X-Ray，並且需要具有權限的 AWS 認證才能執行此操作。

在 Amazon EC2 上，常駐程式會自動使用執行個體的執行個體設定檔角色。如需在本機執行常駐程式所需認證的相關資訊，請參閱在本機執行應用程式。

若您在超過一個位置指定登入資料 (登入資料檔案、執行個體描述檔，或是環境變數)，軟體開發套件提供者鏈會判斷使用的登入資料有哪些。如需有關向 SDK 提供認證的詳細資訊，請參閱《SDK to Go 開發人員指南》中的 < 指定認證 > 。AWS

精靈登入資料所屬的 IAM 角色或使用者必須具備代您將資料寫入服務的許可。

- 若要在 Amazon EC2 上使用精靈，請建立新的執行個體設定檔角色，或將受管政策新增至現有的執行個體設定檔角色。
- 若要在 Elastic Beanstalk 上使用常駐程式，請將受管理的原則新增至 Elastic Beanstalk 預設執行個體設定檔角色。
- 若要在本機執行協助程式，請參閱在本機執行應用程式。

如需詳細資訊，請參閱 的身分識別與存取管理 AWS X-Ray。

X-Ray 守护进程

精靈會輸出其目前組態和傳送給 AWS X-Ray區段的相關資訊。

```
2016-11-24T06:07:06Z [Info] Initializing AWS X-Ray daemon 2.1.0
2016-11-24T06:07:06Z [Info] Using memory limit of 49 MB
2016-11-24T06:07:06Z [Info] 313 segment buffers allocated
2016-11-24T06:07:08Z [Info] Successfully sent batch of 1 segments (0.123 seconds)
2016-11-24T06:07:09Z [Info] Successfully sent batch of 1 segments (0.006 seconds)
```

根據預設，精靈會將日誌輸出到 STDOUT。若您在背景執行精靈，請使用 `--log-file` 命令列選項或組態檔來設定日誌檔案路徑。您可以設定日誌層級及停用日誌輪換。如需說明，請參閱 配置 AWS X-Ray 守護進程。

在 Elastic Beanstalk 上，平台會設定守護程式記錄檔的位置。如需詳細資訊，請參閱 在上執行 X-Ray 協助程式AWS Elastic Beanstalk。

配置 AWS X-Ray 守護進程

您可以使用指令行選項或組態檔案來自訂 X-Ray 精靈的行為。大多數的選項皆可透過這兩種方法取得，但有些選項僅可透過組態檔取得，有些選項也僅能透過命令列使用。

若要開始使用，您唯一需要知道的選項是`-n`或`--region`，您用來設定精靈用來將追蹤資料傳送至 X-Ray 的區域。

```
~/xray-daemon$ ./xray -n us-east-2
```

如果您在本機執行常駐程式 (也就是不是在 Amazon EC2 上)，您可以新增略過檢查執行個體設定檔登入資料的`-o`選項，以便協助程式更快就緒。

```
~/xray-daemon$ ./xray -o -n us-east-2
```

其餘命令列選項可讓您設定記錄日誌、在不同連接埠進行接聽、限制精靈能使用的記憶體數量，或是取得角色來將追蹤資料傳送至不同帳戶。

您可以將組態檔案傳遞給精靈，以存取進階設定選項，並執行諸如將同時呼叫數限制為 X-Ray、停用記錄輪替，以及將流量傳送至 Proxy 等作業。

章節

支援的環境變數

X-Ray 精靈支援下列環境變數：

- AWS_REGION— 指定 AWS 區域X-Ray 服務端點。
- HTTPS_PROXY— 指定要透過上傳區段的精靈代理位址。這可以是 DNS 網域名稱，或是代理伺服器使用的 IP 地址和連接埠號碼。

使用命令列選項

在您於本機執行，或是使用使用者資料指令碼時將這些選項傳遞至精靈。

命令列選項

- -b, --bind — 偵聽不同 UDP 連接埠上的區段文件。

  ```
  --bind "127.0.0.1:3000"
  ```

  預設值 — 2000。
- -t，--bind-tcp— 在不同的 TCP 連接埠上接聽 X-Ray 服務的呼叫。

  ```
  -bind-tcp "127.0.0.1:3000"
  ```

  預設值 — 2000。
- -c，--config— 從指定的路徑載入組態檔案。

  ```
  --config "/home/ec2-user/xray-daemon.yaml"
  ```
- -f，--log-file— 將記錄輸出至指定的檔案路徑。

  ```
  --log-file "/var/log/xray-daemon.log"
  ```

- `-l`，`--log-level`-日誌級別，從最詳細到最小：開發，調試，信息，警告，錯誤，prod。

```
--log-level warn
```

  預設值 — prod
- `-m`，`--buffer-memory`— 變更緩衝區可以使用的記憶體容量 (至少 3)。

```
--buffer-memory 50
```

  預設 — 1% 的可用記憶體。
- `-o`，`--local-mode`— 不檢查 EC2 執行個體中繼資料。
- `-r`，`--role-arn`— 假設指定的 IAM 角色，將區段上傳至其他帳戶。

```
--role-arn "arn:aws:iam::123456789012:role/xray-cross-account"
```

- `-a`, `--resource-arn` — 執行守護程式之資源的 Amazon AWS 資源名稱 (ARN)。
- `-p`, `--proxy-address` — AWS X-Ray 透過代理將區段上傳至。必須指定代理伺服器的通訊協定。

```
--proxy-address "http://192.0.2.0:3000"
```

- `-n`、`--region` — 將區段傳送至特定區域的 X-Ray 服務。
- `-v`, `--version` — 顯示 AWS X-Ray 守護進程版本。
- `-h`，`--help`— 顯示說明畫面。

使用組態檔

您也可以使用 YAML 格式的檔案來設定精靈。使用 `-c` 選項來將組態檔傳遞至精靈。

```
~$ ./xray -c ~/xray-daemon.yaml
```

組態檔選項

- `TotalBufferSizeMB`— 緩衝區大小上限 (以 MB 為單位) (最小 3)。選擇 0 來使用主機記憶體的 1%。
- `Concurrency`— 上傳區段文件的同時 AWS X-Ray 呼叫數目上限。
- `Region`— 發送細分以在特定區域 AWS X-Ray 服務。

- Socket— 配置守護進程的綁定。
  - UDPAddress— 變更守護程式偵聽的連接埠。
  - TCPAddress— 在不同的 TCP 端口上偵聽對 X-Ray 服務的呼叫。
- Logging— 設定記錄行為。
  - LogRotation— 設定為停false用防護記錄輪換。
  - LogLevel— 將記錄層級從最詳細資訊變更為最小：dev、debug、info或prod、、warn、error、prod。預設值為prod，相當於info。
  - LogPath— 將日誌輸出到指定的文件路徑。
- LocalMode— 設定為可略true過檢查 EC2 執行個體中繼資料。
- ResourceARN— 執行守護程式之資源的 Amazon AWS 資源名稱 (ARN)。
- RoleARN— 假設指定的 IAM 角色將區段上傳至其他帳戶。
- ProxyAddress— 透 AWS X-Ray 過代理將區段上傳至。
- Endpoint— 變更精靈將區段文件傳送至的 X-Ray 服務端點。
- NoVerifySSL— 停用 TLS 憑證驗證。
- Version— 守護程式組態檔案格式版本。檔案格式版本為必填欄位。

Example xray-daemon.yaml

此組態檔會將精靈的接聽連接埠變更為 3000、關閉執行個體中繼資料檢查，設定上傳區段所要使用的角色，並變更區域與記錄日誌選項。

```
Socket:
  UDPAddress: "127.0.0.1:3000"
  TCPAddress: "127.0.0.1:3000"
Region: "us-west-2"
Logging:
  LogLevel: "warn"
  LogPath: "/var/log/xray-daemon.log"
LocalMode: true
RoleARN: "arn:aws:iam::123456789012:role/xray-cross-account"
Version: 2
```

在本機執行 X-Ray 精靈

您可以在 Linux、MacOS、Windows 或 Docker 容器中於本機執行 AWS X-Ray 精靈。執行常駐程式，以在開發和測試已檢測式應用程式轉送追蹤資料至 X-Ray。透過使用此處的說明，下載並解壓縮精靈。

在本機執行時，精靈可以從AWS SDK 認證檔案 (`.aws/credentials`在您的使用者目錄中) 或環境變數讀取認證。如需詳細資訊，請參閱授予守護進程將數據發送到 X-Ray 的權限。

精靈會在連接埠 2000 接聽 UDP 資料。您可以透過使用組態檔和命令列選項來變更連接埠及其他選項。如需詳細資訊，請參閱配置 AWS X-Ray 守護進程。

在 Linux 上執行 X-Ray 精靈

您可以從命令列執行精靈的可執行檔。使用 `-o` 選向來在本機模式中執行，以及 `-n` 來設定區域。

```
~/xray-daemon$ ./xray -o -n us-east-2
```

若要在背景執行精靈，請使用 &。

```
~/xray-daemon$ ./xray -o -n us-east-2 &
```

使用 `pkill` 終止在背景執行的精靈程序。

```
~$ pkill xray
```

在 Docker 容器執行 X-Ray 精靈

若要在 Docker 容器內於本機執行精靈，請將以下文字儲存到名為 `Dockerfile` 的檔案。在亞馬遜 ECR 上下載完整的示例圖片。如需詳細資訊，請參閱下載協助程式。

Example 碼頭檔案 — Amazon Linux

```
FROM amazonlinux
RUN yum install -y unzip
RUN curl -o daemon.zip https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/
xray-daemon/aws-xray-daemon-linux-3.x.zip
RUN unzip daemon.zip && cp xray /usr/bin/xray
ENTRYPOINT ["/usr/bin/xray", "-t", "0.0.0.0:2000", "-b", "0.0.0.0:2000"]
EXPOSE 2000/udp
```

```
EXPOSE 2000/tcp
```

使用 docker build 建置容器映像。

```
~/xray-daemon$ docker build -t xray-daemon .
```

使用 docker run 在容器中執行映像。

```
~/xray-daemon$ docker run \
      --attach STDOUT \
      -v ~/.aws/:/root/.aws/:ro \
      --net=host \
      -e AWS_REGION=us-east-2 \
      --name xray-daemon \
      -p 2000:2000/udp \
      xray-daemon -o
```

此命令使用下列選項：

- --attach STDOUT— 查看終端中守護進程的輸出。
- -v ~/.aws/:/root/.aws/:ro— 為容器提供.aws目錄的唯讀存取權，以便讀取您的AWS SDK認證。
- AWS_REGION=*us-east-2*— 設置AWS_REGION環境變量以告訴守護進程使用哪個區域。
- --net=host— 將容器連接到host網絡。主機網路上的容器可彼此互相通訊，而無須透過連接埠發佈。
- -p 2000:2000/udp— 將計算機上的 UDP 端口 2000 映射到容器上的相同端口。這並非位於相同網路上容器進行通訊的必要項目，但它可讓您從命令列將區段傳送至精靈，或是從沒有在 Docker 中執行的應用程式傳送。
- --name xray-daemon— 命名容器，xray-daemon而不是產生隨機名稱。
- -o(映像檔名稱之後) — 將選-o項附加至在容器內執行協助程式的進入點。此選項會告知常駐程式在本機模式下執行，以防止其嘗試讀取 Amazon EC2 執行個體中繼資料。

若要停止精靈，請使用 docker stop。若您變更 Dockerfile 並建置新的映像，您需要刪除現有的容器，才能使用相同名稱建立另一個容器。使用 docker rm 來刪除容器。

```
$ docker stop xray-daemon
$ docker rm xray-daemon
```

在 Windows 上執行 X-Ray 精靈

您可以從命令列執行精靈的可執行檔。使用 -o 選向來在本機模式中執行，以及 -n 來設定區域。

```
> .\xray_windows.exe -o -n us-east-2
```

使用 PowerShell 指令碼建立並執行協助程式的服務。

Example PowerShell 腳本-視窗

```
if ( Get-Service "AWSXRayDaemon" -ErrorAction SilentlyContinue ){
    sc.exe stop AWSXRayDaemon
    sc.exe delete AWSXRayDaemon
}
if ( Get-Item -path aws-xray-daemon -ErrorAction SilentlyContinue ) {
    Remove-Item -Recurse -Force aws-xray-daemon
}

$currentLocation = Get-Location
$zipFileName = "aws-xray-daemon-windows-service-3.x.zip"
$zipPath = "$currentLocation\$zipFileName"
$destPath = "$currentLocation\aws-xray-daemon"
$daemonPath = "$destPath\xray.exe"
$daemonLogPath = "C:\inetpub\wwwroot\xray-daemon.log"
$url = "https://s3.dualstack.us-west-2.amazonaws.com/aws-xray-assets.us-west-2/xray-
daemon/aws-xray-daemon-windows-service-3.x.zip"

Invoke-WebRequest -Uri $url -OutFile $zipPath
Add-Type -Assembly "System.IO.Compression.Filesystem"
[io.compression.zipfile]::ExtractToDirectory($zipPath, $destPath)

sc.exe create AWSXRayDaemon binPath= "$daemonPath -f $daemonLogPath"
sc.exe start AWSXRayDaemon
```

在 OS X-Ray 精靈

您可以從命令列執行精靈的可執行檔。使用 -o 選向來在本機模式中執行，以及 -n 來設定區域。

```
~/xray-daemon$ ./xray_mac -o -n us-east-2
```

若要在背景執行精靈，請使用 &。

```
~/xray-daemon$ ./xray_mac -o -n us-east-2 &
```

使用 nohup 來防止精靈在終端機關閉時終止。

```
~/xray-daemon$ nohup ./xray_mac &
```

在上執行 X-Ray 協助程式AWS Elastic Beanstalk

將跟蹤數據從應用程序中繼到AWS X-Ray，您可以在 Elastic Beanstalk 環境的 Amazon EC2 執行個體上執行 X-Ray 協助程式。如需支援的平台清單，請參設定AWS X-Ray除錯中的AWS Elastic Beanstalk開發人員指南。

> **Note**
>
> 精靈會使用您環境的執行個體描述檔以獲得許可。如需將許可新增至 Elastic Beanstalk 執行個體組態檔的說明，請參授予守護進程將數據發送到 X-Ray 的權限。

Elastic Beanstalk 平台提供組態選項，您可以設定自動執行精靈。您可以在您來源碼的組態檔中啟用精靈，或是在 Elastic Beanstalk 主控台中選擇選項來啟用。當您啟用組態選項時，精靈便會在執行個體上安裝並做為服務執行。

Elastic Beanstalk 平台上包含的版本可能並非最新版本。請參閱支援的平台主題以了解您平台組態可使用的精靈版本。

Elastic Beanstalk 不會在 Multicontainer Docker (Amazon ECS) 平台上提供 X-Ray 協助程式。

使用 Elastic Beanstalk X-Ray 精靈

使用主控台開啟 X-Ray 集成，或是使用組態檔，在您的應用程式來源碼中設定它。

在 Elastic Beanstalk 主控台中啟用 X-Ray 協助程式

1. 開啟Elastic Beanstalk 主控台。
2. 導覽至 。管理主控台適用於您的環境。
3. 選擇 Configuration (組態)。
4. 選擇 Software Settings (軟體設定)。
5. 針對 X-Ray daemon (X-Ray 精靈)，請選擇 Enabled (啟用)。

6. 選擇 Apply (套用)。

您可以在您的來源碼中包含組態檔，來在環境間攜帶您的組態。

Example .ebextensions/xray-daemon.config

```
option_settings:
  aws:elasticbeanstalk:xray:
    XRayEnabled: true
```

Elastic Beanstalk 會將組態檔傳遞至精靈，並將日誌輸出至標準位置。

Windows Server 平台

- 組態檔案–`C:\Program Files\Amazon\XRay\cfg.yaml`
- 日誌–`c:\Program Files\Amazon\XRay\logs\xray-service.log`

Linux 平台

- 組態檔案–`/etc/amazon/xray/cfg.yaml`
- 日誌–`/var/log/xray/xray.log`

Elastic Beanstalk 提供從AWS Management Console或命令列。您可以透過使用組態檔新增任務，來告知 Elastic Beanstalk 包含 X-Ray 協助程式日誌。

Example .ebextensions/xray-logs.config - Linux

```
files:
  "/opt/elasticbeanstalk/tasks/taillogs.d/xray-daemon.conf" :
    mode: "000644"
    owner: root
    group: root
    content: |
      /var/log/xray/xray.log
```

Example .ebextensions/xray-logs.config - Windows Server

```
files:
  "c:/Program Files/Amazon/ElasticBeanstalk/config/taillogs.d/xray-daemon.conf" :
```

```
    mode: "000644"
    owner: root
    group: root
    content: |
      c:\Progam Files\Amazon\XRay\logs\xray-service.log
```

請參閱從 Elastic Beanstalk 環境的 Amazon EC2 執行個體查看日誌中的AWS Elastic Beanstalk開發人員指南以了解詳細資訊。

手動下載 X-Ray 協助程式（高級）

如果 X-Ray 協助程式不適用於您的平台組態，您可以從 Amazon S3 下載它並使用組態檔執行它。

使用 Elastic Beanstalk 組態檔下載及執行精靈。

Example .ebextensions/xray.config - Linux

```
commands:
  01-stop-tracing:
    command: yum remove -y xray
    ignoreErrors: true
  02-copy-tracing:
    command: curl https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-
daemon/aws-xray-daemon-3.x.rpm -o /home/ec2-user/xray.rpm
  03-start-tracing:
    command: yum install -y /home/ec2-user/xray.rpm

files:
  "/opt/elasticbeanstalk/tasks/taillogs.d/xray-daemon.conf" :
    mode: "000644"
    owner: root
    group: root
    content: |
      /var/log/xray/xray.log
  "/etc/amazon/xray/cfg.yaml" :
    mode: "000644"
    owner: root
    group: root
    content: |
      Logging:
        LogLevel: "debug"
      Version: 2
```

Example .ebextensions/xray.config - Windows Server

```
container_commands:
  01-execute-config-script:
    command: Powershell.exe -ExecutionPolicy Bypass -File c:\\temp\\installDaemon.ps1
    waitAfterCompletion: 0

files:
  "c:/temp/installDaemon.ps1":
    content: |
      if ( Get-Service "AWSXRayDaemon" -ErrorAction SilentlyContinue ) {
          sc.exe stop AWSXRayDaemon
          sc.exe delete AWSXRayDaemon
      }

      $targetLocation = "C:\Program Files\Amazon\XRay"
      if ((Test-Path $targetLocation) -eq 0) {
          mkdir $targetLocation
      }

      $zipFileName = "aws-xray-daemon-windows-service-3.x.zip"
      $zipPath = "$targetLocation\$zipFileName"
      $destPath = "$targetLocation\aws-xray-daemon"
      if ((Test-Path $destPath) -eq 1) {
          Remove-Item -Recurse -Force $destPath
      }

      $daemonPath = "$destPath\xray.exe"
      $daemonLogPath = "$targetLocation\xray-daemon.log"
      $url = "https://s3.dualstack.us-west-2.amazonaws.com/aws-xray-assets.us-west-2/
xray-daemon/aws-xray-daemon-windows-service-3.x.zip"

      Invoke-WebRequest -Uri $url -OutFile $zipPath
      Add-Type -Assembly "System.IO.Compression.Filesystem"
      [io.compression.zipfile]::ExtractToDirectory($zipPath, $destPath)

      New-Service -Name "AWSXRayDaemon" -StartupType Automatic -BinaryPathName
 "`"$daemonPath`" -f `"$daemonLogPath`""
      sc.exe start AWSXRayDaemon
    encoding: plain
  "c:/Program Files/Amazon/ElasticBeanstalk/config/taillogs.d/xray-daemon.conf" :
    mode: "000644"
    owner: root
    group: root
```

```
    content: |
      C:\Program Files\Amazon\XRay\xray-daemon.log
```

這些範例也會將精靈的日誌檔新增至 Elastic Beanstalk 尾日誌任務，使其在您透過主控台或 Elastic Beanstalk 命令列界面 (EB CLI) 請求日誌時包含在其中。

在 Amazon EC2 上執行 X-Ray 常設程式

您可以在 Amazon EC2 上的下列作業系統中執行 X-Ray 常設程式：

- Amazon Linux
- Ubuntu
- Windows Server (2012 R2 和更新版本)

使用實例配置文件授予協助程式將追蹤資料上傳至 X-Ray 的許可。如需詳細資訊，請參閱 授予守護進程將數據發送到 X-Ray 的權限。

利用使用者資料指令碼，在您啟動執行個體時自動執行協助程式。

Example 使用者資料指令碼 - Linux

```
#!/bin/bash
curl https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-daemon/aws-xray-daemon-3.x.rpm -o /home/ec2-user/xray.rpm
yum install -y /home/ec2-user/xray.rpm
```

Example 使用者資料指令碼 - Windows Server

```
<powershell>
if ( Get-Service "AWSXRayDaemon" -ErrorAction SilentlyContinue ) {
    sc.exe stop AWSXRayDaemon
    sc.exe delete AWSXRayDaemon
}

$targetLocation = "C:\Program Files\Amazon\XRay"
if ((Test-Path $targetLocation) -eq 0) {
    mkdir $targetLocation
}

$zipFileName = "aws-xray-daemon-windows-service-3.x.zip"
```

```
$zipPath = "$targetLocation\$zipFileName"
$destPath = "$targetLocation\aws-xray-daemon"
if ((Test-Path $destPath) -eq 1) {
    Remove-Item -Recurse -Force $destPath
}

$daemonPath = "$destPath\xray.exe"
$daemonLogPath = "$targetLocation\xray-daemon.log"
$url = "https://s3.dualstack.us-west-2.amazonaws.com/aws-xray-assets.us-west-2/xray-
daemon/aws-xray-daemon-windows-service-3.x.zip"

Invoke-WebRequest -Uri $url -OutFile $zipPath
Add-Type -Assembly "System.IO.Compression.Filesystem"
[io.compression.zipfile]::ExtractToDirectory($zipPath, $destPath)

New-Service -Name "AWSXRayDaemon" -StartupType Automatic -BinaryPathName
 "`"$daemonPath`" -f `"$daemonLogPath`""
sc.exe start AWSXRayDaemon
</powershell>
```

X-Ray Amazon

在 Amazon ECS 中，建立執行 X-Ray 精靈的 Docker 映像，將其上傳到 Docker 映像儲存庫，然後將其部署到 Amazon ECS 叢集。您可以使用您任務定義檔案中的連接埠映射和網路模式設定，來允許應用程式與精靈容器通訊。

使用官方 Docker 映像檔

X-Ray 在 Amazon ECR 上提供 Docker 容器映像檔，您可以與應用程式一起部署。如需詳細資訊，請參閱下載協助程式。

Example 任務定義

```
    {
      "name": "xray-daemon",
      "image": "amazon/aws-xray-daemon",
      "cpu": 32,
      "memoryReservation": 256,
      "portMappings" : [
          {
              "hostPort": 0,
```

```
            "containerPort": 2000,
            "protocol": "udp"
        }
    ]
}
```

建立和建置 Docker 影像

用於自訂組態時，您可能需要定義自己的 Docker 影像。

將受管理的原則新增至您的任務角色，授予常駐程式將追蹤資料上傳至 X-Ray 的許可。如需詳細資訊，請參閱授予守護進程將數據發送到 X-Ray 的權限。

使用以下其中一個 Dockerfiles 建立執行精靈的映像。

Example Amazon

```
FROM amazonlinux
RUN yum install -y unzip
RUN curl -o daemon.zip https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/
xray-daemon/aws-xray-daemon-linux-3.x.zip
RUN unzip daemon.zip && cp xray /usr/bin/xray
ENTRYPOINT ["/usr/bin/xray", "-t", "0.0.0.0:2000", "-b", "0.0.0.0:2000"]
EXPOSE 2000/udp
EXPOSE 2000/tcp
```

> **Note**
>
> 指定要接聽多重容器環境迴路的繫結地址時，必須提供標記 `-t` 和 `-b`。

Example Dockerfile-Ubuntu 的

若是 Debian 的衍生產品，您還需要安裝憑證授權機構 (CA) 憑證，以避免下載安裝程式時發生問題。

```
FROM ubuntu:16.04
RUN apt-get update && apt-get install -y --force-yes --no-install-recommends apt-
transport-https curl ca-certificates wget && apt-get clean && apt-get autoremove && rm
 -rf /var/lib/apt/lists/*
RUN wget https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-daemon/aws-
xray-daemon-3.x.deb
```

```
RUN dpkg -i aws-xray-daemon-3.x.deb
ENTRYPOINT ["/usr/bin/xray", "--bind=0.0.0.0:2000", "--bind-tcp=0.0.0.0:2000"]
EXPOSE 2000/udp
EXPOSE 2000/tcp
```

在您的任務定義中，組態取決於您使用的聯網模式。預設值是橋接聯網，可在您的預設 VPC 中使用。在橋接網路中，設定AWS_XRAY_DAEMON_ADDRESS環境變數，告知 X-Ray SDK 要參考哪個容器連接埠，並設定主機連接埠。舉例來說，您可以發佈 UDP 連接埠 2000，然後建立您應用程式容器與精靈容器的連結。

Example 任務定義

```
    {
      "name": "xray-daemon",
      "image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/xray-daemon",
      "cpu": 32,
      "memoryReservation": 256,
      "portMappings" : [
          {
              "hostPort": 0,
              "containerPort": 2000,
              "protocol": "udp"
          }
       ]
    },
    {
      "name": "scorekeep-api",
      "image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/scorekeep-api",
      "cpu": 192,
      "memoryReservation": 512,
      "environment": [
          { "name" : "AWS_REGION", "value" : "us-east-2" },
          { "name" : "NOTIFICATION_TOPIC", "value" : "arn:aws:sns:us-
east-2:123456789012:scorekeep-notifications" },
          { "name" : "AWS_XRAY_DAEMON_ADDRESS", "value" : "xray-daemon:2000" }
      ],
      "portMappings" : [
          {
              "hostPort": 5000,
              "containerPort": 5000
          }
      ],
```

```
      "links": [
        "xray-daemon"
      ]
    }
```

若您在 VPC 私有子網路中執行您的叢集，您可以使用 awsvpc 網路模式來將彈性網路界面 (ENI) 連接到您的容器。這可讓您避免使用連結。忽略連接埠映射、連結，以及 AWS_XRAY_DAEMON_ADDRESS 環境變數中的主機連接埠。

Example VPC 任務定義

```
{
    "family": "scorekeep",
    "networkMode":"awsvpc",
    "containerDefinitions": [
        {
          "name": "xray-daemon",
          "image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/xray-daemon",
          "cpu": 32,
          "memoryReservation": 256,
          "portMappings" : [
              {
                  "containerPort": 2000,
                  "protocol": "udp"
              }
          ]
        },
        {
            "name": "scorekeep-api",
            "image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/scorekeep-api",
            "cpu": 192,
            "memoryReservation": 512,
            "environment": [
                { "name" : "AWS_REGION", "value" : "us-east-2" },
                { "name" : "NOTIFICATION_TOPIC", "value" : "arn:aws:sns:us-
east-2:123456789012:scorekeep-notifications" }
            ],
            "portMappings" : [
                {
                    "containerPort": 5000
                }
            ]
        }
```

```
    ]
}
```

在亞馬遜 ECS 主控台中設定命令列選項

命令列選項會覆寫影像組態檔中的任何衝突值。命令列選項通常用於本機測試，但也可以在設定環境變數或控制啟動程序時使用。

透過新增命令列選項，您將更新傳遞給容器的 Docker CMD。如需詳細資訊，請參閱 Docker 執行參考。

若要設定命令列選項

1. 開啟 Amazon，網址為 https://console.aws.amazon.com/ecs/ 開啟 Amazon
2. 從導覽列中選擇包含您任務定義的區域。
3. 在導覽窗格中，選擇 Task Definitions (任務定義)。
4. 在 Task Definitions (任務定義) 頁面中，選取要修訂之任務定義左側的方塊，並選擇 Create new revision (建立新修訂版)。
5. 在 Create new revision of Task Definition (建立任務定義的新修訂版) 頁面上，選取容器。
6. 在 ENVIRONMENT (環境) 區段中，將逗號分隔的命令列選項清單新增至 Command (命令) 欄位。
7. 選擇 Update (更新)。
8. 驗證資訊，然後選擇 Create (建立)。

下列範例顯示如何為 RoleARN 選項撰寫逗號分隔的命令列。此選RoleARN項會假設指定的 IAM 角色，將區段上傳至其他帳戶。

Example

```
--role-arn, arn:aws:iam::123456789012:role/xray-cross-account
```

若要深入瞭解 X-Ray 中可用的指令行選項，請參閱設定AWS X-Ray協助程式。

檢測您的應用程式 AWS X-Ray

檢測您的應用程式需要傳送追蹤資料，以及應用程式內的傳入和輸出要求以及其他事件，以及每個要求的中繼資料。根據您的特定需求，您可以選擇或組合幾種不同的儀器選項：

- auto 動檢測 — 透過組態變更、新增自動檢測代理程式或其他機制，以零程式碼變更來檢測您的應用程式。
- 程式庫檢測 — 進行最少的應用程式程式碼變更，以新增針對特定程式庫或架構 (例如 AWS SDK、Apache HTTP 用戶端或 SQL 用戶端) 的預先建置檢測。
- 手動檢測 — 在您要傳送追蹤資訊的每個位置，將檢測程式碼新增至應用程式。

有數個 SDK、代理程式和工具可用來檢測您的應用程式以進行 X-Ray 追蹤。

主題

使用發行 AWS 版檢測您的應用程式 OpenTelemetry

該發行 AWS 版 OpenTelemetry （ADOT）是基於雲原生計算基礎（CNCF）項目的 AWS 發行版。OpenTelemetry OpenTelemetry 提供一組開放原始碼 API、程式庫和代理程式，以收集分散式追蹤和指標。此工具組是上游 OpenTelemetry 元件的散佈版本，包括 SDK、自動檢測代理程式，以及經過測試、最佳化、保護和支援的收集器。 AWS

有了 ADOT，工程師只需一次測試應用程式，就可以將相關的指標和追蹤傳送到包括 Amazon 和 Amazon Ser CloudWatch vice 在內的多個 AWS 監控解決方案。 AWS X-Ray OpenSearch

將 X-Ray 與 ADOT 搭配使用需要兩個元件：啟用可與 X-Ray 搭配使用的 OpenTelemetry SDK，以及啟用可與 X-Ray 搭配使用的 OpenTelemetry 收集器發行AWS 版。 OpenTelemetry 有關使用發行版 AWS X-Ray 和其他 AWS 版本的更多信息 AWS 服務，請參閱發行AWS 版的 OpenTelemetry文檔。

如需有關語言支援和使用方式的詳細資訊，請參閱上 GitHub的AWS 可觀測性。

> **Note**
>
> 您現在可以使用 CloudWatch 代理程式從 Amazon EC2 執行個體和內部部署伺服器收集指標、日誌和追蹤。 CloudWatch 代理程式版本 1.300025.0 及更新版本可以從OpenTelemetry或 X-Ray 用戶端 SDK 收集追蹤，並將它們傳送至 X-Ray。使用 CloudWatch 代理程式代替發行 AWS 版 OpenTelemetry (ADOT) 收集器或 X-Ray 精靈收集追蹤，可協助您減少管理的代理程式數量。如需詳細資訊，請參閱 CloudWatch 使用指南中的CloudWatch 代理程式主題。

ADOT 包括以下內容：

- AWS 適用於圍棋的發行 OpenTelemetry 版
- AWS 爪哇發行版 OpenTelemetry
- AWS 發行版 OpenTelemetry JavaScript
- AWS Python 的 OpenTelemetry 發行版
- AWS 適用於. NET 的發行 OpenTelemetry 版

ADOT 目前包括對 Java 和 Python 的自動檢測支援。此外，亞朵透過 ADOT 管理的 AWS Lambda 層，透過 Java、Node.js 和 Python 執行階段，啟用 Lambda 函數及其下游請求的自動檢測。

適用於 Java 和 Go 的 ADOT 開發套件支援 X-Ray 集中式取樣規則。如果您需要其他語言的 X-Ray 取樣規則的支援，請考慮使用 AWS X-Ray SDK。

> **Note**
>
> 您現在可以發送 W3C 跟踪 ID 到 X-Ray。根據預設，使用建立的追蹤具 OpenTelemetry 有以 W3C 追蹤內容規格為基礎的追蹤 ID 格式。這與使用 X-Ray SDK 或透過與 X-Ray 整合的 AWS 服務所建立的追蹤 ID 格式不同。為了確保 X-Ray 接受 W3C 格式的追蹤 ID，您必須使用AWS X-Ray 匯出程式版本 0.86.0 或更新版本，該版本包含在 ADOT 收集器 0.34.0 及更新版本中。舊版匯出程式會驗證追蹤識別碼時間戳記，這可能會導致 W3C 追蹤識別碼遭到拒絕。

使用 SDK 檢測您的應用程式 AWS X-Ray

AWS X-Ray 包括一組特定於語言的 SDK，用於檢測您的應用程序以將跟踪發送到 X-Ray。每個 X-Ray SDK 都提供以下內容：

- 攔截程式，可新增至您的程式碼以追蹤傳入的 HTTP 請求
- 用於檢測應用程式用來呼叫其他 AWS SDK 用戶端的用戶端處理常式 AWS 服務
- 用於檢測對其他內部和外部 HTTP Web 服務的呼叫的 HTTP 用戶端

X-Ray SDK 也支援檢測對 SQL 資料庫的呼叫、自動 AWS SDK 用戶端檢測和其他功能。SDK 不會將追蹤資料直接傳送至 X-Ray，而是將 JSON 區段文件傳送至偵聽 UDP 流量的精靈程序。X-Ray 守護程序緩衝隊列中的段，並將其批量上傳到 X-Ray。

提供下列語言特定的 SDK：

- AWS X-Ray SDK for Go
- AWS X-Ray 適用於 Java 的開發套件
- AWS X-Ray Node.js 適用的開發套
- AWS X-Ray 適用於 Python 的 SDK
- AWS X-Ray SDK for .NET
- AWS X-Ray SDK for Ruby

X-Ray 目前包括對 Java 的自動檢測支援。

在 AWS 發行版 OpenTelemetry 和 X-Ray SDK 之間進行選擇

X-Ray 隨附的 SDK 是由 AWS提供的緊密集成儀器解決方案的一部分。發行 AWS 版 OpenTelemetry 是更廣泛的行業解決方案的一部分，其中 X-Ray 只是眾多跟踪解決方案之一。您可以使用任何一種方法在 X-Ray 中實現 end-to-end 跟踪，但是了解差異以確定最有用的方法非常重要。

OpenTelemetry 如果您需要以下內容，我們建議您使用 AWS Distro 檢測您的應用程序：

- 能夠將跟踪發送到多個不同的跟踪後端，而無需重新檢測代碼
- Support 由社區維護的每種語言的大量庫工具 OpenTelemetry
- 完全受管的 Lambda 層，可封裝收集遙測資料所需的一切，而不需要在使用 Java、Python 或 Node.js 時變更程式碼

> **Note**
>
> AWS 用於測試 Lambda 函數的發行版 OpenTelemetry 提供了更簡單的入門體驗。但是，由於靈活性 OpenTelemetry 提供，您的 Lambda 函數需要額外的記憶體，而呼叫可能會遇到冷啟動延遲增加的情況，這可能會導致額外的費用。如果您要針對低延遲進行最佳化，而且不需要 OpenTelemetry動態設定後端目標等進階功能，您可能需要使用 AWS X-Ray SDK 來檢測您的應用程式。

如果您需要以下條件，我們建議您選擇 X-Ray SDK 來檢測您的應用程式：

- 緊密整合的單一廠商解決方案
- 與 X-Ray 集中式取樣規則整合，包括能夠在使用 Node.js、Python、Ruby 或 .NET 時，從 X-Ray 主控台設定取樣規則，並在多個主機上自動使用這些規則

檢測您的應用程式 Go

有兩種方法可以檢測您的Go應用程序以將痕跡發送到 X-Ray：

- AWS 發行版 OpenTelemetry Go — 提供一組開放原始碼程式庫的 AWS 發行版 CloudWatch，AWS X-Ray可透過收集器的發行AWS 版，將相關的指標和追蹤傳送至包括 Amazon 和 Amazon OpenSearch 服務在內的多個 AWS 監控解決方案。 OpenTelemetry
- AWS X-Ray SDK 用於 Go — 一組程式庫，用於透過 X-Ray 守護程式產生追蹤並將其傳送至 X-Ray。

如需更多詳細資訊，請參閱 在 AWS 發行版 OpenTelemetry 和 X-Ray SDK 之間進行選擇。

AWS發行版的OpenTelemetry前往

隨著AWS發行版的OpenTelemetryGo，您可以一次檢測您的應用程序，並將相關的指標和跟踪發送到多個AWS監控解決方案，包括CloudWatch,AWS X-Ray和亞馬遜OpenSearch服務。使用 X 射線 AWS發行版的OpenTelemetry需要兩個組件：一個OpenTelemetrySDK已啟用以與 X 射線搭配使用，以及AWS發行版的OpenTelemetry收藏家已啟用以與 X 射線搭配使用。

若要開始使用，請參閱AWS發行版的OpenTelemetry去文檔。

有關使用的更多信息AWS發行版的OpenTelemetry與AWS X-Ray和其他AWS 服務，請參閱AWS發行版的OpenTelemetry或AWS發行版的OpenTelemetry文件。

如需語言支援和使用方式的詳細資訊，請參閱AWS上的可觀測性GitHub。

適用於 Go 的 AWS X-Ray 開發套件

適用於 Go 的 X-Ray 開發套件集合，其提供類別與方法，可用於產生及傳送追蹤資料至 X-Ray 守護程式。追蹤資料包含應用程式處理的傳入 HTTP 請求相關資訊，以及應用程式使用AWSSDK、HTTP 客戶端或 SQL 數據庫連接器。您也可以手動建立區段，並將除錯資訊新增至註釋和中繼資料中。

使用 `go get` 以從軟體開發套件的 GitHub 儲存庫下載軟體開發套件：

```
$ go get -u github.com/aws/aws-xray-sdk-go/...
```

若是 Web 應用程式，請先使用 `xray.Handler` 函數來追蹤傳入請求。訊息處理常式會為每個追蹤的請求建立區段，並在傳送回應時完成區段。當區段開啟時，您可以使用軟體開發套件用戶端的方法，將資訊新增到區段，並建立子區段以追蹤下游呼叫。軟體開發套件也會在區段為開啟時自動記錄應用程式擲回的例外狀況。

對於由分析應用程序或服務調用的 Lambda 函數，Lambda 讀取追蹤標頭並自動跟蹤採樣請求。對於其他函數，您可以設定 Lambda以抽樣和追蹤傳入的請求。無論哪種情況，Lambda 都會創建區段並將其提供給 X-Ray SDK。

> **Note**
>
> 在 Lambda 上，X-Ray 開發套件是可選的。如果您不在函數中使用它，則服務映射仍將包含 Lambda 服務的一個節點，每個 Lambda 函數都會包含一個節點。通過添加 SDK，您可以測試函數代碼，將子段添加到 Lambda 記錄的函數段。如需詳細資訊，請參閱 AWS Lambda 而且 AWS X-Ray。

接著，請呼叫 AWS 函數以包裝您的用戶端。此步驟可確保 X-Ray 檢測到對任何用戶端方法的呼叫。您也可以檢測對 SQL 資料庫的呼叫。

開始使用軟體開發套件之後，請設定配置記錄器和中間件。您可以新增外掛程式，以記錄執行應用程式所需的運算資源相關資料、定義抽樣規則以自訂抽樣行為，並設定日誌層級以在應用程式日誌中查看更多或更少的軟體開發套件資訊。

使用註釋與中繼資料，記錄應用程式所做的請求和工作等其他資訊。註釋是簡單的鍵/值對，系統會為其建立索引以用於篩選條件表達式，因此您可以搜尋包含特定資料的追蹤。元數據條目的限制性較小，並且可以記錄整個對象和數組-任何可以序列化為 JSON 的內容。

> **標註與中繼資料**
>
> 註釋和元數據是使用 X-Ray 開發套件添加至區段的任意文本。註釋會建立索引以與篩選條件表達式搭配使用。元數據不是索引的，但可以使用 X-Ray 控制台或 API 在原始數據段中查看。您授予 X-Ray 讀取訪問權限的任何人都可以查看此數據。

當程式碼中有很多經過檢測的用戶端時，單一請求區段可能包含大量子區段，每個使用經檢測用戶端進行的呼叫都有一個子區段。您可以將用戶端呼叫包裝在自訂子區段中，以組織和群組子區段。您可以為整個函數或任何部分的程式碼建立自訂子區段，並記錄子區段上的中繼資料和註釋，而不必寫入父區段上的所有項目。

要求

適用於 Go 的 X-Ray 開發套件需要 Go 1.9 或更高版本。

軟體開發套件在編譯和執行時間需仰賴下列程式庫：

- AWSSDK for Go 開發套件 1.10.0 或更新版本

軟體開發套件的 `README.md` 檔案中有宣告這些相依性。

參考文件

下載軟體開發套件之後，請本機建置和託管文件以在 Web 瀏覽器中檢視。

檢視參考文件

1. 導覽至 `$GOPATH/src/github.com/aws/aws-xray-sdk-go` (Linux 或 Mac) 目錄或 `%GOPATH%\src\github.com\aws\aws-xray-sdk-go` (Windows) 資料夾
2. 執行 godoc 命令。

   ```
   $ godoc -http=:6060
   ```

3. 在瀏覽器中開啟 `http://localhost:6060/pkg/github.com/aws/aws-xray-sdk-go/` 網址。

設定適用於 Go 的 X-Ray 的開發套件

您可以指定 X-Ray SDK 的配置，通過`Configure`使用`Config`對象調用或假設默認值。環境變數優先於 `Config` 值，而後者則優先於任何預設值。

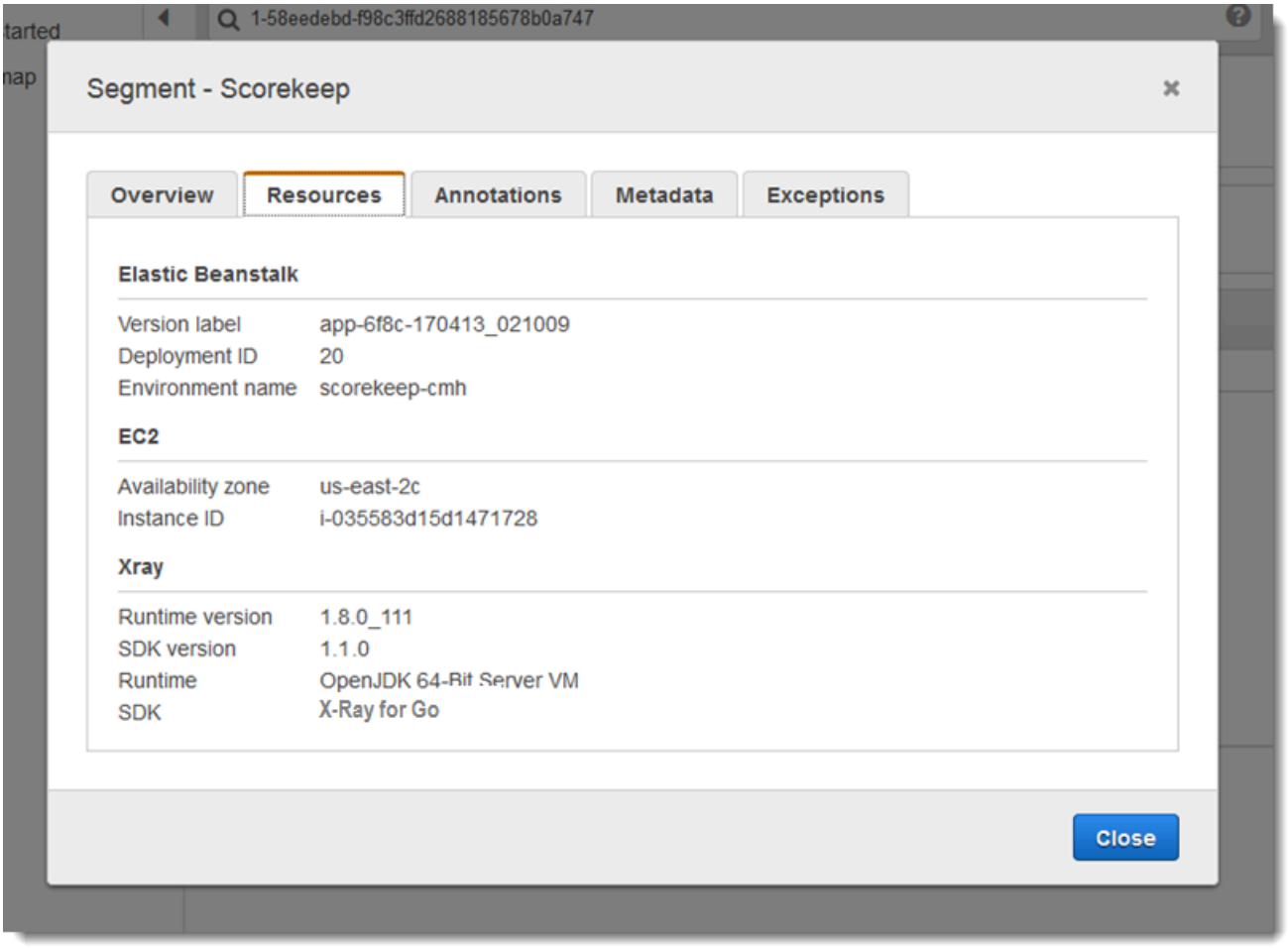
章節

服務外掛程式

用`plugins`於記錄託管應用程式之服務的相關資訊。

外掛程式

- Amazon EC2 — `EC2Plugin` 新增執行個體 ID、可用區域和 CloudWatch 日誌群組。
- Elastic Beanstalk — `ElasticBeanstalkPlugin` 新增環境名稱、版本標籤和部署 ID。
- Amazon ECS-`ECSPlugin` 添加容器 ID。



若要使用外掛程式，請匯入以下其中一個套件。

```
"github.com/aws/aws-xray-sdk-go/awsplugins/ec2"
"github.com/aws/aws-xray-sdk-go/awsplugins/ecs"
"github.com/aws/aws-xray-sdk-go/awsplugins/beanstalk"
```

每個外掛程式都有一個明確載入該外掛程式的 `Init()` 函數呼叫。

Example ec2.Init()

```
import (
 "os"

 "github.com/aws/aws-xray-sdk-go/awsplugins/ec2"
 "github.com/aws/aws-xray-sdk-go/xray"
)

func init() {
```

```
  // conditionally load plugin
  if os.Getenv("ENVIRONMENT") == "production" {
    ec2.Init()
  }

  xray.Configure(xray.Config{
    ServiceVersion: "1.2.3",
  })
}
```

SDK 也會使用外掛程式設定來設定區origin段上的欄位。這表示運行您的應用程序的 AWS 資源的類型。當您使用多個外掛程式時，SDK 會使用下列解析順序來判斷來源： ElasticBeanstalk > EKS > ECS > EC2。

抽樣規則

SDK 會使用您在 X-Ray 主控台中定義的取樣規則來決定要記錄的要求。預設規則會每秒追蹤第一個要求，而所有服務的任何其他要求的百分之五會傳送追蹤至 X-Ray。在 X-Ray 主控台中建立其他規則，以自訂為每個應用程式記錄的資料量。

SDK 會依定義規則的順序套用自訂規則。如果要求符合多個自訂規則，SDK 只會套用第一個規則。

> **Note**
>
> 如果 SDK 無法達到 X-Ray 以取得取樣規則，它會每秒還原為第一個要求的預設本機規則，而每台主機的任何其他要求的百分之五。如果主機沒有調用採樣 API 的權限，或者無法連接到 X-Ray 守護程序（作為 SDK 發出的 API 調用的 TCP 代理），則可能會發生這種情況。

您也可以將 SDK 設定為從 JSON 文件載入取樣規則。SDK 可以使用本機規則做為無法使用 X-Ray 取樣的情況的備份，或僅使用本機規則。

Example 採樣規則

```
{
  "version": 2,
  "rules": [
    {
      "description": "Player moves.",
      "host": "*",
      "http_method": "*",
      "url_path": "/api/move/*",
```

```
      "fixed_target": 0,
      "rate": 0.05
    }
  ],
  "default": {
    "fixed_target": 1,
    "rate": 0.1
  }
}
```

此範例定義了一個自訂規則和一個預設規則。自訂規則會套用百分之五的取樣率，而且沒有追蹤下限路徑的要求數目下/api/move/限。預設規則會每秒追蹤第一個要求，以及 10% 的額外要求。

在本機定義規則的缺點是，固定目標會由記錄器的每個執行個體獨立套用，而不是由 X-Ray 服務管理。當您部署更多主機時，固定費率會倍增，因此更難以控制記錄的資料量。

開啟時 AWS Lambda，您無法修改取樣率。如果您的函數是由已檢測的服務呼叫，則 Lambda 會記錄產生由該服務取樣之請求的呼叫。如果啟用主動追蹤且沒有追蹤標頭，Lambda 會做出取樣決策。

若要提供備份規則，請使用 `NewCentralizedStrategyWithFilePath` 指向本機抽樣 JSON 檔案。

Example 主要-本地抽樣規則

```
s, _ := sampling.NewCentralizedStrategyWithFilePath("sampling.json") // path to local
 sampling json
xray.Configure(xray.Config{SamplingStrategy: s})
```

若僅要使用本機規則，請使用 `NewLocalizedStrategyFromFilePath` 指向本機抽樣 JSON 檔案。

Example 主. 去-禁用採樣

```
s, _ := sampling.NewLocalizedStrategyFromFilePath("sampling.json") // path to local
 sampling json
xray.Configure(xray.Config{SamplingStrategy: s})
```

日誌

> **Note**
>
> 從版本 1.0.0-rc.10 開始已取代 `xray.Config{}` 欄位 `LogLevel` 和 `LogFormat`。

X-Ray 使用下列介面進行記錄。在 LogLevelInfo 和以上時，預設記錄器會寫入 stdout。

```
type Logger interface {
 Log(level LogLevel, msg fmt.Stringer)
}

const (
 LogLevelDebug LogLevel = iota + 1
 LogLevelInfo
 LogLevelWarn
 LogLevelError
)
```

Example 寫入 **io.Writer**

```
xray.SetLogger(xraylog.NewDefaultLogger(os.Stderr, xraylog.LogLevelError))
```

環境變數

您可以使用環境變數來設定 Go 的 X-Ray SDK。軟體開發套件支援以下變數。

- AWS_XRAY_CONTEXT_MISSING— 設定為當您RUNTIME_ERROR的檢測程式碼嘗試在沒有區段開啟時記錄資料時擲回例外狀況。

  有效值

  - RUNTIME_ERROR— 擲回執行階段例外狀況。
  - LOG_ERROR— 記錄錯誤並繼續 (預設值)。
  - IGNORE_ERROR— 忽略錯誤並繼續。

  當您嘗試在沒有要求開啟時執行的啟動程式碼中使用已檢測的用戶端，或在產生新執行緒的程式碼中使用已檢測的用戶端時，可能會發生與遺失區段或子區段相關的錯誤。
- AWS_XRAY_TRACING_NAME— 設定 SDK 用於區段的服務名稱。
- AWS_XRAY_DAEMON_ADDRESS— 設定 X-Ray 精靈監聽程式的主機和連接埠。依預設，SDK 會將追蹤資料傳送至127.0.0.1:2000。如果您已將協助程式設定為在不同的連接埠上接聽，或是在不同的主機上執行，請使用此變數。

環境變數會覆寫程式碼中所設的同等值。

使用設定

您還可以使用該Configure方法為 Go 配置 X-Ray SDK。 Configure需要一個引數，一個Config物件，包含下列選擇性欄位。

DaemonAddr

此字串指定 X-Ray 精靈監聽程式的主機和連接埠。如果未指定，X-Ray 會使用AWS_XRAY_DAEMON_ADDRESS環境變數的值。如果未設定該值，則會使用 "127.0.0.1:2000"。

ServiceVersion

此字串可指定服務的版本。如果未指定，X-Ray 會使用空字串 (「」)。

SamplingStrategy

此 SamplingStrategy 物件可指定要追蹤哪些應用程式呼叫。如果未指定，X-Ray 會使用 aLocalizedSamplingStrategy，它採用中所定義的策略xray/resources/DefaultSamplingRules.json。

StreamingStrategy

該StreamingStrategy對象指定當RequiresStreaming返回 true 是否流段。如果未指定，X-Ray 會在子區段數目大於 20 時使用串流取樣線段。DefaultStreamingStrategy

ExceptionFormattingStrategy

此 ExceptionFormattingStrategy 物件可指定您要如何處理各種例外狀況。如果未指定，X-Ray 會使DefaultExceptionFormattingStrategy用 a XrayError 類型error、錯誤訊息和堆疊追蹤。

使用適用於 Go 的 X-Ray 的開發套件來檢測傳入 HTTP 請求

您可以使用 X-Ray 開發套件來追蹤您應用程式在 Amazon EC2 中 EC2 執行個體上處理的傳入 HTTP 請求AWS Elastic Beanstalk或亞馬遜彈性雲服務器。

使用 xray.Handler 檢測傳入的 HTTP 請求。適用於 Go 的 X-Ray SDK for Go 實現標準 Go 庫http.Handler界面中xray.Handler類來攔截 Web 請求。xray.Handler 類別會使用請求的內容、剖析傳入標頭、視需要新增回應標頭，以將提供的 http.Handler 和 xray.Capture 包裝在一起並設定 HTTP 特定追蹤欄位。

當您使用此類別來處理 HTTP 請求和回應時，適用於 Go 的 X-Ray 的開發套件會為每個抽樣請求建立區段。此區段包括時間、方法，以及 HTTP 請求的處置方式。其他檢測會在此區段上建立子區段。

> **Note**
>
> 適用於AWS Lambda函數時，Lambda 會為每個抽樣請求建立區段。如需詳細資訊，請參閱 AWS Lambda 而且 AWS X-Ray。

下面的示例攔截在端口 8000 的請求，並返回「Hello!」 作為迴應. 它會建立 myApp 區段，並檢測透過任何應用程式的呼叫。

Example main.go

```
func main() {
  http.Handle("/", xray.Handler(xray.NewFixedSegmentNamer("MyApp"),
 http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
    w.Write([]byte("Hello!"))
  })))

  http.ListenAndServe(":8000", nil)
}
```

每個區段都有一個用於標識服務映射中的應用程序的名稱。可以靜態命名段，也可以將 SDK 配置為基於傳入請求中的主機標頭動態命名它。動態命名允許您根據請求中的域名對跟蹤進行分組，如果名稱與預期模式不匹配（例如，如果主機標頭是偽造的），則應用默認名稱。

> **轉發請求**
>
> 如果負載均衡器或其他中間機構將請求轉發給您的應用程序，X-Ray 會從X-Forwarded-For標頭，而不是來自 IP 數據包中的源 IP。為轉發請求記錄的客户端 IP 可以偽造，因此不應信任該 IP。

轉發請求時，SDK 會在段中設置一個額外的字段來指示這一點。如果區段包含字段x_forwarded_for設定為true，則客户端 IP 是從X-Forwarded-For標頭。

處理常式會使用 http 區塊為每個傳入的請求建立區段，其中包含以下資訊：

- HTTP method (HTTP 方法)— 獲取、張貼、放置、刪除等。
- 用户端地址— 傳送請求的用户端 IP 地址。
- 回應代碼— 已完成請求的 HTTP 回應代碼。

- Timing (時間點)— 開始時間 (收到請求) 和結束時間 (傳送回應)。
- 用户代理程式—`user-agent`從請求。
- 內容長度—`content-length`來自回應的。

設定區段命名策略

AWS X-Ray使用服務名稱來識別您的應用程序，並將其與其他應用程序、數據庫、外部 API 和AWS您應用程式使用的資源。當 X-Ray SDK 為傳入請求生成區段時，它會將應用程序的服務名稱記錄在欄位名稱。

X-Ray SDK 可以在 HTTP 請求標頭中的主機名後面命名段。但是，此標頭可能是偽造的，這可能會導致服務映射中出現意外的節點。要防止 SDK 由於使用偽造主機標頭的請求而錯誤地命名段，您必須為傳入請求指定默認名稱。

如果您的應用程序為多個域的請求提供服務，則可以將 SDK 配置為使用動態命名策略在段名稱中反映這一點。動態命名策略允許 SDK 對與預期模式匹配的請求使用主機名，並將默認名稱應用於不匹配的請求。

例如，您可能有單個應用程式來處理三個子域名的請求—`www.example.com`、`api.example.com`，以及`static.example.com`。您可以將動態命名策略與模式`*.example.com`來標識具有不同名稱的每個子域的段，從而在服務映射上生成三個服務節點。如果您的應用程序收到的主機名與模式不匹配的請求，則您將在服務映射上看到第四個節點，其中包含您指定的回退名稱。

若要為所有請求區段使用相同的名稱，請如上一節所述，在建立處理常式時指定您應用程式的名稱。

> **Note**
>
> 您可以使用 `AWS_XRAY_TRACING_NAME` 環境變數來覆寫您在程式碼中定義的預設服務名稱。

動態命名策略可定義主機名稱應相符的模式，以及如果 HTTP 請求中的主機名稱不符合模式時要使用的預設名稱。若要動態命名區段，請使用 `NewDynamicSegmentNamer` 來設定要符合的預設名稱和模式。

Example main.go

如果請求中的主機名稱符合 `*.example.com` 模式，請使用該主機名稱。否則，請使用 `MyApp`。

```
func main() {
  http.Handle("/", xray.Handler(xray.NewDynamicSegmentNamer("MyApp", "*.example.com"),
 http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
    w.Write([]byte("Hello!"))
  })))

  http.ListenAndServe(":8000", nil)
}
```

使用適用於 Go 的 X-Ray AWS SDK 追蹤 SDK 呼叫

當您的應用程式呼叫 AWS 服務 以儲存資料、寫入佇列或傳送通知時，Go 的 X-Ray SDK 會追蹤子區段中下游的呼叫。您在這些服務中存取的追蹤 AWS 服務 和資源 (例如，Amazon S3 儲存貯體或 Amazon SQS 佇列) 會在 X-Ray 主控台的追蹤對應上顯示為下游節點。

若要追蹤 AWS 開發套件用戶端，請使用 `xray.AWS()` 呼叫來包裝物件用戶端，如下範例所示。

Example main.go

```
var dynamo *dynamodb.DynamoDB
func main() {
  dynamo = dynamodb.New(session.Must(session.NewSession()))
  xray.AWS(dynamo.Client)
}
```

然後，當您使用 AWS SDK 客戶端時，請使用調用方法的`withContext`版本，並將其`context`從傳遞給處理常式的`http.Request`對象傳遞。

Example 主. 去- AWS SDK 調用

```
func listTablesWithContext(ctx context.Context) {
  output := dynamo.ListTablesWithContext(ctx, &dynamodb.ListTablesInput{})
  doSomething(output)
}
```

對於所有服務，您都可以在 X-Ray 控制台中看到調用的 API 的名稱。對於服務子集，X-Ray SDK 會將資訊新增至區段，以在服務對應中提供更多精細度。

例如，當您使用已檢測的 DynamoDB 用戶端進行呼叫時，SDK 會將資料表名稱新增至區段，以便針對以資料表為目標的呼叫。在主控台中，每個表格在服務對應中顯示為獨立節點，其中包含一般 DynamoDB 節點，用於未針對資料表的呼叫。

Example 呼叫 DynamoDB 以儲存項目的子區段

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

您存取具名資源時，對以下服務的呼叫會在服務地圖中建立額外節點。未針對特定資源的呼叫，則會建立服務的一般節點。

- Amazon DynamoDB — 表名
- Amazon 簡單存儲服務 — 存儲桶和密鑰名稱
- Amazon 簡單隊列服務-隊列名稱

使用適用於 Go 的 X-Ray 開發套件追蹤對下遊 HTTP Web 服務進行的呼叫

當您的應用程式呼叫微服務或公有 HTTP API 時，您可以使用 `xray.Client` 並以 Go 應用程式子區段的形式檢測這些呼叫，如下範例所示，其中 http-client 是 HTTP 用戶端。

用户端會建立所提供 HTTP 用户端的淺層複製，默認為`http.DefaultClient`, 與纏繞器包裹`xray.RoundTripper`。

Example

<caption>main.go — HTTP 用户端</caption>

```
myClient := xray.Client(http-client)
```

<caption>主 .go — 使用 ctxhttp 庫跟蹤下遊 HTTP 調用</caption>

下面的示例使用 ctxhttp 庫來測試傳出 HTTP 調用`xray.Client`。`ctx`可以從上遊調用傳遞。這可確保使用現有的區段上下文。例如，X-Ray 不允許在 Lambda 函數中創建新的段，因此應使用現有的 Lambda 段上下文。

```
resp, err := ctxhttp.Get(ctx, xray.Client(nil), url)
```

使用適用於 Go 的 X-Ray 開發套件追蹤 SQL 查詢

若要追蹤 PostgreSQL 或 MySQL 的 SQL 呼叫，請將 `sql.Open` 呼叫取代為 `xray.SQLContext`，如下範例所示。盡可能使用 URL，而非組態字串。

Example main.go

```
func main() {
  db, err := xray.SQLContext("postgres", "postgres://user:password@host:port/db")
  row, err := db.QueryRowContext(ctx, "SELECT 1") // Use as normal
}
```

使用適用於 Go 的 X-Ray 的開發套件生成自訂子區段

子段擴展追蹤的段，詳細介紹了為了服務請求而完成的工作。每次與分析客户端進行呼叫時，X-Ray SDK 都會記錄在子段中生成的信息。您可以創建其他子段來對其他子段進行分組、測量代碼部分的性能或記錄註釋和元數據。

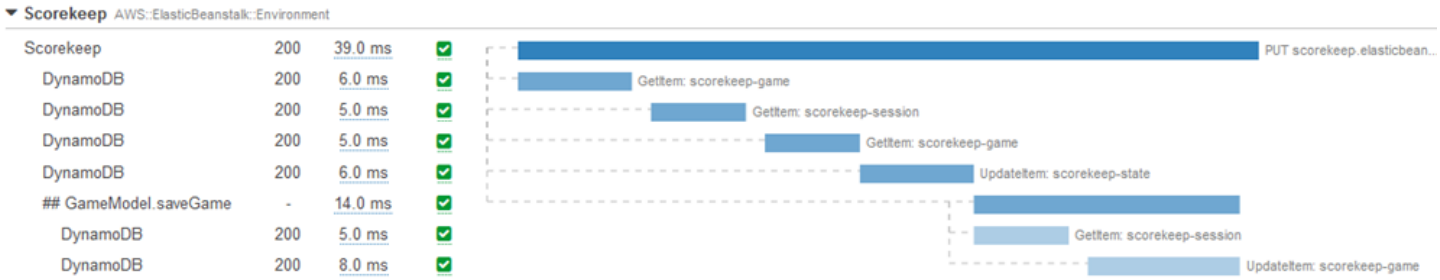
使用 `Capture` 方法來建立函數周圍的子區段。

Example main.go — 自訂子區段

```
func criticalSection(ctx context.Context) {
  //this is an example of a subsegment
  xray.Capture(ctx, "GameModel.saveGame", func(ctx1 context.Context) error {
    var err error

    section.Lock()
    result := someLockedResource.Go()
    section.Unlock()

    xray.AddMetadata(ctx1, "ResourceResult", result)
```

```
})
```

以下螢幕擷取畫面顯示 `saveGame` 子區段在 `Scorekeep` 應用程式追蹤中可能出現的方式。



使用適用於 Go 的 X-Ray SDK 將註釋和中繼資料新增至區段

您可以使用註釋和中繼資料來記錄有關請求、環境或應用程式的其他資訊。您可以將註釋和中繼資料新增至 X-Ray SDK 建立的區段，或新增至您建立的自訂子區段。

註釋是與字符串，數字或布爾值鍵-值對。註釋會編製索引以與篩選器運算式搭配使用 使用標記記錄您想要用來在主控台將追蹤分組的資料，或是在呼叫 `GetTraceSummaries` API 時使用標記。

中繼資料是索引鍵-值配對，可以具有任何類型的值 (包括物件和清單)，但不會編製索引以供篩選運算式使用。使用元數據記錄要存儲在跟踪中但不需要與搜索一起使用的其他數據。

除了註釋和中繼資料，您也可以在區段上記錄使用者 ID 字串。區段會將使用者 ID 記錄在單獨的欄位中，並建立索引以用於搜尋。

章節

- 使用適用於 Go 的 X-Ray SDK 錄製註釋
- 使用適用於 Go 的 X-Ray SDK 記錄中繼資料
- 使用適用於 Go 的 X-Ray SDK 記錄使用者 ID

使用適用於 Go 的 X-Ray SDK 錄製註釋

針對您想要建立索引以用於搜尋的區段，請使用註釋來記錄這些區段上的資訊。

註釋要求

- 按鍵 — X-Ray 註解的金鑰最多可包含 500 個英數字元。您不能使用底線符號 (_) 以外的空格或符號。
- 值 — X-Ray 註釋的值最多可包含 1,000 個 Unicode 字元。

- 註釋的數量 — 每個追蹤最多可以使用 50 個註釋。

若要記錄註釋，請使用包含要與區段建立關聯之中繼資料的字串來呼叫 AddAnnotation。

```
xray.AddAnnotation(key string, value interface{})
```

軟體開發套件會將標註以鍵/值對記錄在區段文件中的 annotations 物件內。若使用相同索引鍵呼叫 AddAnnotation 兩次，則會覆寫之前在相同區段上記錄的值。

若要尋找具有特定值之註釋的繪線，請在篩選器運算式中使用annotations.*key*關鍵字。

使用適用於 Go 的 X-Ray SDK 記錄中繼資料

針對您不想要建立索引以用於搜尋的區段，請使用中繼資料來記錄這些區段上的資訊。

若要記錄中繼資料，請使用包含要與區段建立關聯之中繼資料的字串來呼叫 AddMetadata。

```
xray.AddMetadata(key string, value interface{})
```

使用適用於 Go 的 X-Ray SDK 記錄使用者 ID

記錄請求區段上的使用者 ID 以識別傳送請求的使用者。

記錄使用者 ID

1. 從 AWSXRay 取得目前區段的參考。

   ```
   import (
     "context"
     "github.com/aws/aws-xray-sdk-go/xray"
   )

   mySegment := xray.GetSegment(context)
   ```

2. 使用傳送請求之使用者的字串 ID 呼叫 setUser。

   ```
   mySegment.User = "U12345"
   ```

若要尋找使用者 ID 的追蹤，請在篩選器運算式中使用user關鍵字。

檢測您的應用程式 Java

有兩種方法可以檢測您的Java應用程序以將痕跡發送到 X-Ray：

- AWS 發行版 OpenTelemetry Java — 提供一組開放原始碼程式庫的 AWS 發行版，可透過收集器的發行AWS 版，將相關的指標和追蹤傳送至包括 Amazon CloudWatch 和 Amazon OpenSearch 服務在內的多個 AWS 監控解決方案。 AWS X-Ray OpenTelemetry
- AWS X-Ray SDK 用於 Java — 一組程式庫，用於透過 X-Ray 守護程式產生追蹤並將其傳送至 X-Ray。

如需更多詳細資訊，請參閱 在 AWS 發行版 OpenTelemetry 和 X-Ray SDK 之間進行選擇。

AWS發行版OpenTelemetry爪哇

隨著AWS發行版OpenTelemetry（ADOT）Java，您可以一次檢測您的應用程序，並將相關的指標和跟踪發送給多個AWS監控解決方案，包括CloudWatch,AWS X-Ray和亞馬遜OpenSearch服務。使用 X 射線與 ADOT 需要兩個組成部分：OpenTelemetrySDK已啟用以與 X 射線搭配使用，以及AWS發行版OpenTelemetry收藏家已啟用以與 X 射線搭配使用。ADOT Java 包含自動檢測支援，可讓您的應用程式在不變更程式碼的情況下傳送追蹤。

若要開始使用，請參閱AWS發行版OpenTelemetry爪哇文件。

有關使用的更多信息AWS發行版OpenTelemetry與AWS X-Ray和其他AWS 服務，請參閱AWS發行版OpenTelemetry或AWS發行版OpenTelemetry文件。

如需語言支援和使用方式的詳細資訊，請參閱AWS上的可觀測性GitHub。

AWS X-Ray 適用於的 SDK Java

Java 的 X-Ray SDK 是一組適用於 Java Web 應用程式的程式庫，可提供產生追蹤資料並將追蹤資料傳送至 X-Ray 守護程式的類別和方法。追蹤資料包括應用程式所提供之傳入 HTTP 要求的相關資訊，以及應用程式使用 AWS SDK、HTTP 用戶端或 SQL 資料庫連接器對下游服務進行呼叫的相關資訊。您也可以手動建立區段，並將除錯資訊新增至註釋和中繼資料中。

適用於 Java 的 X-Ray SDK 是一個開源項目。您可以關注該項目並在以下位置提交問題並提取請求 GitHub：gi thub.com/aws/ aws-xray-sdk-java

首先，將 `AWSXRayServletFilter` 新增為 Servlet 篩選條件以追蹤傳入的請求。servlet 篩選條件會建立區段。當區段開啟時，您可以使用軟體開發套件用戶端的方法，將資訊新增到區段，並建立子區段以追蹤下游呼叫。軟體開發套件也會在區段為開啟時自動記錄應用程式擲回的例外狀況。

從 1.3 版開始，您可以使用 Spring 的切面導向程式設計 (AOP) 來檢測應用程式。這意味著您可以在應用程序運行時檢測應用程序 AWS，而無需向應用程序的運行時添加任何代碼。

接下來，使用適用於 Java 的 X-Ray SDK 通過在構建配置中包含 SDK 儀器子模塊來檢測 AWS SDK for Java 客戶端。每當您使用已檢測的用戶端對下游 AWS 服務 或資源進行呼叫時，SDK 都會在子區段中記錄有關呼叫的資訊。 AWS 服務 而您在服務中存取的資源會顯示為追蹤對映上的下游節點，以協助您識別個別連線上的錯誤和節流問題。

如果您不想對所有下游呼叫進行檢測 AWS 服務，則可以省略 Instrumentor 子模塊並選擇要檢測的客戶端。通過向 AWS SDK 服務客戶端添加一個`TracingHandler`來檢測單個客戶端。

Java 子模塊的其他 X-Ray SDK 為 HTTP Web API 和 SQL 數據庫的下游調用提供了檢測。您可以使用適用於 Java 版本的`HTTPClient`和 Apache HTTP 子模組`HTTPClientBuilder`中的 X-Ray SDK 來檢測阿帕奇 HTTP 用戶端。若要檢測 SQL 查詢，請將軟體開發套件的攔截程式新增至資料來源。

開始使用 SDK 之後，請透過設定記錄器和 servlet 篩選器來自訂其行為。您可以新增外掛程式，以記錄執行應用程式所需的運算資源相關資料、定義抽樣規則以自訂抽樣行為，並設定日誌層級以在應用程式日誌中查看更多或更少的軟體開發套件資訊。

使用註釋與中繼資料，記錄應用程式所做的請求和工作等其他資訊。註釋是簡單的鍵/值對，系統會為其建立索引以用於篩選條件表達式，因此您可以搜尋包含特定資料的追蹤。元數據條目的限制較低，可以記錄整個對象和數組-任何可以序列化為 JSON 的內容。

> **標註與中繼資料**
>
> 註釋和中繼資料是您使用 X-Ray SDK 新增至區段的任意文字。註釋會編製索引以與篩選器運算式搭配使用 中繼資料不會建立索引，但可以使用 X-Ray 主控台或 API 在原始區段中檢視。您授與 X-Ray 讀取權限的任何人都可以檢視此資料。

當程式碼中有許多經過檢測的用戶端時，單一請求區段可能包含大量子區段，每個使用經檢測用戶端進行的呼叫都有一個子區段。您可以將用戶端呼叫包裝在自訂子區段中，以組織和群組子區段。您可以為整個函數或任何部分的程式碼建立自訂子區段，並記錄子區段上的中繼資料和註釋，而不必寫入父區段上的所有項目。

子模組

您可以從 Maven 下載適用於 Java 的 X-Ray SDK。Java 的 X-Ray SDK 依使用案例分割為子模組，並附有用於版本管理的材料清單：

- aws-xray-recorder-sdk-core(必要) — 建立區段和傳輸區段的基本功能。包含 AWSXRayServletFilter 以檢測傳入的請求。
- aws-xray-recorder-sdk-aws-sdk— 通過添加跟踪 AWS SDK for Java 客戶端作為請求處理程序來 AWS 服務 調用與客戶端進行的儀器。
- aws-xray-recorder-sdk-aws-sdk-v2— 通過添加跟踪客戶端作為請求接收器來對 AWS 服務 AWS SDK for Java 2.2 和更高版本的客戶進行的儀器調用。
- aws-xray-recorder-sdk-aws-sdk-instrumentor— 同aws-xray-recorder-sdk-aws-sdk, 儀器自動所有 AWS SDK for Java 客戶端.
- aws-xray-recorder-sdk-aws-sdk-v2-instrumentor— 與aws-xray-recorder-sdk-aws-sdk-v2, 儀器所有 AWS SDK for Java 2.2 及更高版本的客戶自動.
- aws-xray-recorder-sdk-apache-http— 使用阿帕奇 HTTP 客戶端進行的儀器輸出 HTTP 調用。
- aws-xray-recorder-sdk-spring— 為 Spring AOP 框架應用程序提供攔截器。
- aws-xray-recorder-sdk-sql-postgres— 儀器對使用 JDBC 進行的 PostgreSQL 資料庫的呼出呼叫。
- aws-xray-recorder-sdk-sql-mysql— 儀器對使用 JDBC 進行的 MySQL 數據庫的出站呼叫。
- aws-xray-recorder-sdk-bom— 提供材料清單，您可以用來指定要用於所有子模組的版本。
- aws-xray-recorder-sdk-metrics— 從您收集的 X-Ray 段中發布未採樣的 Amazon CloudWatch 指標。

如果您使用 Maven 或 Gradle 來構建應用程序，請將適用於 Java 的 X-Ray SDK 添加到構建配置中。

如需 SDK 類別和方法的參考文件，請參閱 AWS X-Ray SDK 以取得 Java API 參考。

要求

適用於 Java 的 X-Ray SDK 需要 Java 8 或更高版本，SERVLET API 3， AWS SDK 和傑克遜。

軟體開發套件在編譯和執行時間需仰賴下列程式庫：

- AWS 適用於 1.11.398 或更新Java版本的開發套件
- Servlet API 3.1.0

軟體開發套件的 pom.xml 檔案中有宣告這些相依性。如果您使用 Maven 或 Gradle 來建置，則會自動包含這些相依性。

如果您使用包含在 Java 的 X-Ray SDK 中的程式庫，您必須使用隨附的版本。例如，如果您已在執行時間相依於 Jackson 並為了該相依性在部署中包含 JAR 檔案，則必須移除這些 JAR 檔案，因為軟體開發套件 JAR 包含自己的 Jackson 程式庫版本。

相依性管理

適用於 Java 的 X-Ray SDK 可從 Maven 獲得：

- 集團 — `com.amazonaws`
- Artifact — `aws-xray-recorder-sdk-bom`
- 版本：`2.11.0`

如果您使用 Maven 來建置應用程式，請在 `pom.xml` 檔案中，將軟體開發套件新增為依存項目。

Example pom.xml - 依存項目

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-xray-recorder-sdk-bom</artifactId>
      <version>2.11.0</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-core</artifactId>
  </dependency>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-apache-http</artifactId>
  </dependency>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-aws-sdk</artifactId>
  </dependency>
  <dependency>
    <groupId>com.amazonaws</groupId>
```

```
    <artifactId>aws-xray-recorder-sdk-aws-sdk-instrumentor</artifactId>
  </dependency>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-sql-postgres</artifactId>
  </dependency>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-sql-mysql</artifactId>
  </dependency>
</dependencies>
```

若是 Gradle，請在 `build.gradle` 檔案中，將軟體開發套件新增為編譯時間依存項目。

Example build.gradle - 相依性

```
dependencies {
  compile("org.springframework.boot:spring-boot-starter-web")
  testCompile("org.springframework.boot:spring-boot-starter-test")
  compile("com.amazonaws:aws-java-sdk-dynamodb")
  compile("com.amazonaws:aws-xray-recorder-sdk-core")
  compile("com.amazonaws:aws-xray-recorder-sdk-aws-sdk")
  compile("com.amazonaws:aws-xray-recorder-sdk-aws-sdk-instrumentor")
  compile("com.amazonaws:aws-xray-recorder-sdk-apache-http")
  compile("com.amazonaws:aws-xray-recorder-sdk-sql-postgres")
  compile("com.amazonaws:aws-xray-recorder-sdk-sql-mysql")
  testCompile("junit:junit:4.11")
}
dependencyManagement {
    imports {
        mavenBom('com.amazonaws:aws-java-sdk-bom:1.11.39')
        mavenBom('com.amazonaws:aws-xray-recorder-sdk-bom:2.11.0')
    }
}
```

如果您使用 Elastic Beanstalk 部署應用程式，則可以在每次部署時使用 Maven 或 Gradle 執行個體建置，而不必建置和上傳包含所有相依性的大型歸檔。如需使用 Gradle 的範例，請參閱範例應用程式。

AWS X-Ray適用於 Java 的自動測試代理

所以此AWS X-Ray適用於 Java 的自動檢測代理是一個跟蹤解決方案，可以用最少的開發工作來測試 Java Web 應用程序。代理啟用跟蹤基於服務器的應用程序以及使用受支持的框架和庫發出的代理的所

有下遊請求。這包括下遊 Apache HTTP 請求，AWSSDK 請求以及使用 JDBC 驅動程序進行的 SQL 查詢。代理將 X-Ray 上下文（包括所有活動段和子段）跨線程傳播。X-Ray SDK 的所有配置和多功能性仍可用於 Java 代理程序。選擇了合適的默認值，以確保代理以最小的努力工作。

X-Ray 代理解決方案最適合基於服務器的請求響應 Java Web 應用程序服務器。如果您的應用程序使用異步框架，或者沒有很好地建模為請求響應服務，則可能需要考慮使用 SDK 手動檢測。

X-Ray 代理是使用分佈式系統理解工具包或 DisCO 構建的。Disco 是一個開源框架，用於構建可用於分佈式系統的 Java 代理。雖然不需要瞭解迪斯科才能使用 X-Ray 代理，但您可以通過訪問GitHub上的主頁。X-Ray 劑也是完全開源的。要查看源代碼、做出貢獻或提出有關代理的問題，請訪問其GitHub 上的儲存庫。

範例應用程式

所以此eb-java-scoreep樣品應用程序適用於使用 X-Ray 劑進行儀器檢測。此分支不包含 servlet 過濾器或記錄器配置，因為這些功能是由代理完成的。若要在本地運行應用程序或使用AWS資源，請按範例應用程式的自述文件中的步驟進行操作。有關使用示例應用程序生成 X-Ray 跟蹤的說明請參見示例應用程序的教程。

入門

要在您自己的應用程序中開始使用 X-Ray 自動檢測 Java 代理，請按照下列步驟操作。

1. 在您的環境中運行 X-Ray 常駐程式。如需詳細資訊，請參閱「」X-Ray 常駐程式。
2. 下載代理的最新分佈。解壓縮檔案並記下其在文件系統中的位置。其內容應如下所示。

```
disco
### disco-java-agent.jar
### disco-plugins
    ### aws-xray-agent-plugin.jar
    ### disco-java-agent-aws-plugin.jar
    ### disco-java-agent-sql-plugin.jar
    ### disco-java-agent-web-plugin.jar
```

3. 修改應用程序的 JVM 參數以包含以下內容，從而啟用代理。確保`-javaagent`引數之前該`-jar`參數（如果適用）。修改 JVM 參數的過程取決於您用於啟動 Java 服務器的工具和框架。有關具體指南，請參閱服務器框架的文檔。

```
-javaagent:/<path-to-disco>/disco-java-agent.jar=pluginPath=/<path-to-disco>/disco-
plugins
```

4. 要指定應用程序名稱在 X-Ray 控制台上的顯示方式，請將AWS_XRAY_TRACING_NAME環境變數或com.amazonaws.xray.strategy.tracingName系統屬性。如果未提供名稱，則使用默認名稱。
5. 重新啟動服務器或容器。現在會跟蹤傳入請求及其下遊呼叫。如果您未看到預期結果，請參the section called “疑難排解”。

組態

X-Ray 代理由用户提供的外部 JSON 文件進行配置。默認情況下，此文件位於用户類路徑的根目錄（例如，在resources名為的目錄）xray-agent.json。您可以配置配置文件的自定義位置，方法是設置com.amazonaws.xray.configFile系統屬性設置為配置文件的絕對文件系統路徑。

下面會顯示範例組態檔案。

```
{
    "serviceName": "XRayInstrumentedService",
    "contextMissingStrategy": "LOG_ERROR",
    "daemonAddress": "127.0.0.1:2000",
    "tracingEnabled": true,
    "samplingStrategy": "CENTRAL",
    "traceIdInjectionPrefix": "prefix",
    "samplingRulesManifest": "/path/to/manifest",
    "awsServiceHandlerManifest": "/path/to/manifest",
    "awsSdkVersion": 2,
    "maxStackTraceLength": 50,
    "streamingThreshold": 100,
    "traceIdInjection": true,
    "pluginsEnabled": true,
    "collectSqlQueries": false
}
```

配置規範

下表介紹了每個屬性的有效值。屬性名稱區分大小寫，但它們的鍵不是。對於可由環境變量和系統屬性覆蓋的屬性，優先級的順序始終是環境變量，然後是系統屬性，然後是配置文件。請參環境變數，瞭解有關可覆蓋的屬性的信息。所有欄位都是選擇性的。

屬性名稱	類型	有效值	描述	環境變數	系統屬性	預設
serviceName	字串	任何字串	X-Ray 主控台中顯示的檢測服務的名稱。	查詢 _ 追蹤名稱	卓越亞馬遜策略追蹤名稱	X 射線儀器設備服務
上下文缺失策略	字串	日誌錯誤，忽略錯誤	當座席嘗試使用 X-Ray 段上下文但不存在時採取的操作。	無條件 _ 上下文缺失	卓越亞馬遜戰略上下文策略	日誌錯誤
守護門地址	字串	格式化的 IP 地址和端口，或 TCP 和 UDP 地址列表	代理程式用於與 X-Ray 常駐程式通信的地址。	任何定義 _ 守護門 _ 地址	登錄. 亞馬遜 X 射線發射器. 守護單位地址	127.0.0
可氣管	布林值	TRUE、FALSE	通過 X-Ray 代理啟用儀器。	啟用了 AWS_XRAY_ 跟蹤	亞馬遜有限公司氣候素	TRUE
採樣策略	字串	中央, 地方, 無, 全部	代理使用的採樣策略。ALL 捕獲所有請求，NONE 不捕獲任何請求。請參閱抽樣規則。	N/A	N/A	中央
跟蹤數據注入前綴	字串	任何字串	在日誌中注入跟蹤 ID	N/A	N/A	無（空白字串）

屬性名稱	類型	有效值	描述	環境變數	系統屬性	預設
			之前包括提供的前綴。			
採樣規則清單	字串	絕對文件路徑	自定義採樣規則文件的路徑，用作本地採樣策略的採樣規則源，或中央策略的回退規則。	N/A	N/A	默認採樣規則 .JSON
AWS 服務處理程序清單	字串	絕對文件路徑	自定義參數允許列表的路徑，該列表捕獲 AWSSDK 客戶端。	N/A	N/A	默認參數白名單 .JSON
awsSdkVersion	整數	1、2	版本AWSSDK for Java你正在使用。如果忽略`awsServiceHandlerManifest` 也沒有設置。	N/A	N/A	2
最大堆疊跟蹤長度	整數	非負數整數	要記錄在跟蹤中的堆棧跟蹤的最大行。	N/A	N/A	50

屬性名稱	類型	有效值	描述	環境變數	系統屬性	預設
流閾值	整數	非負數整數	至少這麼多子段關閉後，它們會被流式傳輸到帶外守護進程，以避免塊過大。	N/A	N/A	100
追蹤注射	布林值	TRUE、FALSE	啟用 X-Ray 跟蹤 ID 注入到日誌中，如果依賴關係和配置記錄日誌組態也會新增。否則，什麼也不做。	N/A	N/A	TRUE
插件	布林值	TRUE、FALSE	啟用用於記錄有關AWS環境。請參閱插件。	N/A	N/A	TRUE
collectSqlQueries	布林值	TRUE、FALSE	盡可能記錄 SQL 子區段中的 SQL 查詢字串。	N/A	N/A	FALSE

屬性名稱	類型	有效值	描述	環境變數	系統屬性	預設
上下文傳播	布林值	TRUE、FALSE	如果為 true，則自動在線程之間傳播 X-Ray 上下文。否則為、使用線程本地存儲上下文，並且需要在線程之間手動傳播。	N/A	N/A	TRUE

記錄日誌記錄

X-Ray 代理的日誌級別配置方式與適用於 Java 的 X-Ray SDK 相同。請參閱日誌，瞭解使用適用於 Java 的 X-Ray 開發套件設定記錄的詳細信息。

手動檢測

如果除了代理的自動測試外，還要執行手動檢測，請將 X-Ray SDK 作為依賴項添加到項目中。請注意，SDK 的自定義 servlet 過濾器追蹤傳入的請求與 X-Ray 劑不兼容。

> **Note**
>
> 您必須使用最新版本的 X-Ray SDK 在使用代理的同時執行手動檢測。

如果您正在使用 Maven 項目，請將以下依賴項添加到`pom.xml`file.

```
<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-core</artifactId>
    <version>2.11.0</version>
  </dependency>
```

```
  </dependencies>
```

如果您正在使用 Gradle 項目，請將以下依賴項添加到build.gradlefile.

```
implementation 'com.amazonaws:aws-xray-recorder-sdk-core:2.11.0'
```

您可以新增自訂子區段除此之外批註、元數據和用户 ID，就像使用普通 SDK 一樣。代理會自動跨線程傳播上下文，因此在使用多線程應用程序時，不需要傳播上下文的解決方法。

疑難排解

由於代理提供全自動儀器，因此在遇到問題時很難確定問題的根本原因。 如果 X-Ray 代理無法按預期工作，請查看以下問題和解決方案。X-Ray 代理和 SDK 使用雅加達共享日誌記錄 (JCL)。要查看日誌記錄輸出，請確保將 JCL 連接到日誌記錄後端的網橋位於類路徑上，如以下示例所示：log4j-jcl或者jcl-over-slf4j。

問題：我已經在我的應用程序上啟用了 Java 代理，但在 X-Ray 控制台上沒有看到任何內容

X-Ray 常駐程式是否在同一台計算機上運行？

如果沒有，請參閱X-Ray 常駐程式對其進行設置。

在應用程序日誌中，是否會看到類似「初始化 X-Ray 代理記錄器」的消息？

如果您已將代理正確添加到應用程序，則在應用程序啟動時，在開始接受請求之前，此消息將在 INFO 級別記錄。如果此消息不存在，則 Java 代理未與您的 Java 進程一起運行。請確保您已正確遵循所有設置步驟，沒有拼寫錯誤。

在您的應用程序日誌中，您是否看到幾個錯誤消息，其中包括「禁止AWS X-Ray上下文缺少異常」？

出現這些錯誤的原因是代理正在嘗試處理下遊請求，如AWSSDK 請求或 SQL 查詢，但代理無法自動創建段。如果您看到許多這些錯誤，則代理可能不是您使用案例的最佳工具，您可能希望考慮使用 X-Ray SDK 手動檢測。或者，您可以啟用 X-Ray 開發套件調試日誌查看發生上下文缺失異常的堆棧跟蹤。您可以用自定義段來包裝代碼的這些部分，這應該解決這些錯誤。如需使用自訂區段包裝下遊請求的範例，請參檢測啟動程式碼。

問題：我期望的一些細分不會出現在 X-Ray 控制台上

您的應用程序是否使用多線程？

如果您希望創建的某些段未出現在控制台中，則應用程序中的後台線程可能是原因。如果您的應用程序使用「觸發和忘記」的後台線程執行任務，Lambda 如使用AWSSDK 或定期輪詢某個 HTTP 終端節

點，這可能會在代理跨線程傳播上下文時混淆代理。要驗證這是您的問題，請啟用 X-Ray SDK 調試日誌並檢查如下消息：不發出命名的段，<NAME > 因為它是正在進行中的子段。要解決此問題，您可以嘗試在服務器返回之前加入後台線程，以確保記錄在後台線程中完成的所有工作。或者，您可以將代理的`contextPropagation`設定為`false`來禁用後台線程中的上下文傳播。如果你這樣做，你必須使用自定義段手動測試這些線程，或忽略它們產生的上下文缺少異常。

您是否設定取樣規則？

如果 X-Ray 控制台上出現看似隨機或意外的段落，或者您希望在控制台上的段不顯示，則可能會遇到採樣問題。X-Ray 代理使用 X-Ray 控制台中的規則，將集中採樣應用於其創建的所有段。默認規則為每秒 1 段，加上 5% 之後的段進行採樣。這意味着可能不會對使用代理快速創建的段進行採樣。要解決這個問題，您應該在 X-Ray 控制台上創建自定義採樣規則，以便對所需的段進行適當採樣。如需詳細資訊，請參閱「」抽樣。

設定適用於 Java 的 X-Ray SDK

適用於 Java 的 X-Ray SDK 包含一個名`AWSXRay`為提供全域記錄器的類別。這是可以用來檢測程式碼的 `TracingHandler`。您可以設定全域記錄器來自訂為傳入 HTTP 呼叫建立區段的 `AWSXRayServletFilter`。

章節

- 服務外掛程式
- 抽樣規則
- 日誌
- 區段接聽程式
- 環境變數
- 系統屬性

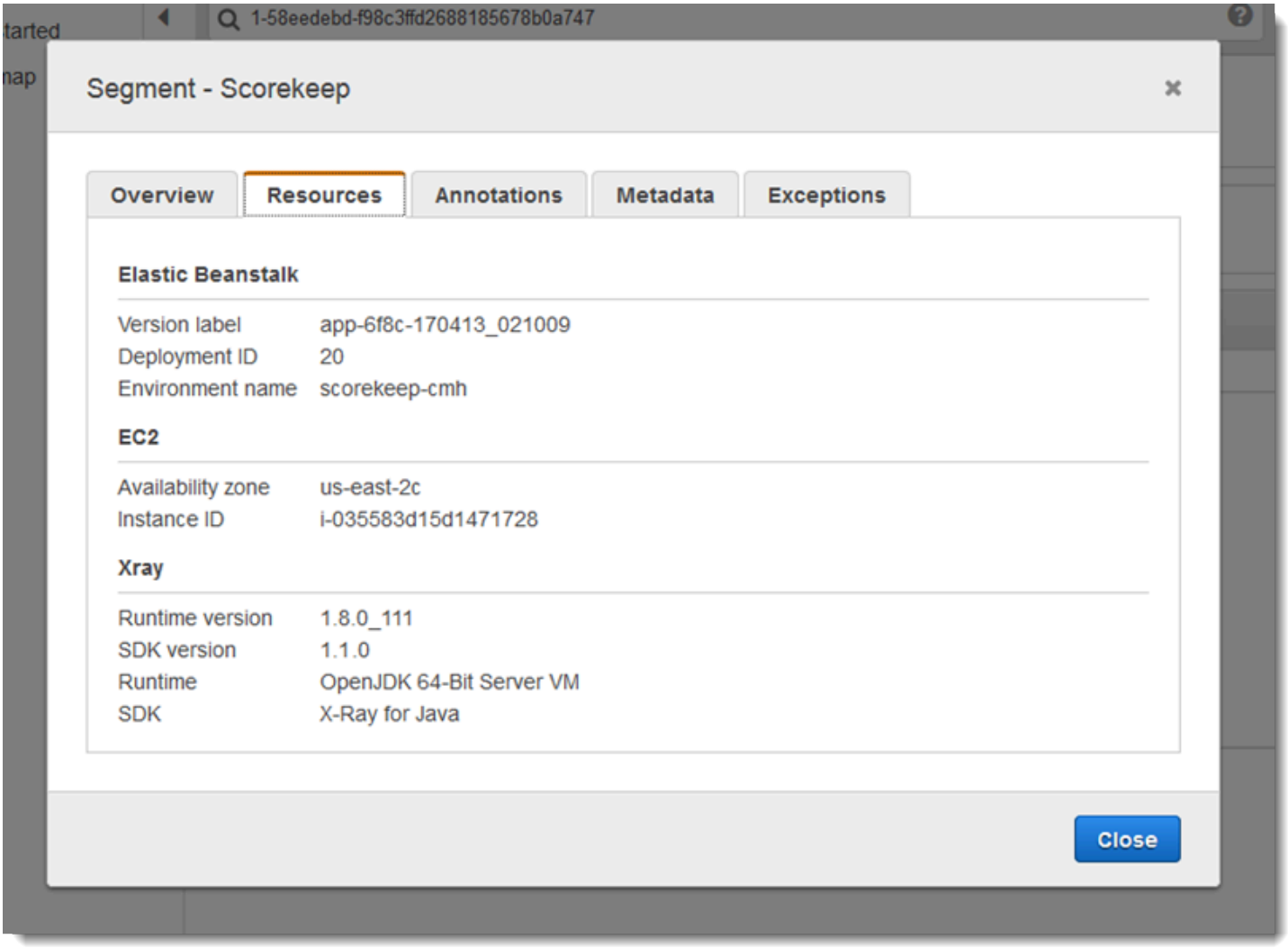
服務外掛程式

用`plugins`來記錄託管應用程式之服務的相關資訊。

外掛程式

- Amazon EC2 — `EC2Plugin` 新增執行個體 ID、可用區域和 CloudWatch 日誌群組。
- Elastic Beanstalk — `ElasticBeanstalkPlugin` 新增環境名稱、版本標籤和部署 ID。
- Amazon ECS-`ECSPlugin` 添加容器 ID。

- Amazon EKS — `EKSPlugin` 新增容器 ID、叢集名稱、網繭識別碼和 CloudWatch 日誌群組。



若要使用外掛程式，請在 `AWSXRayRecorderBuilder` 上呼叫 `withPlugin`。

Example src /主/爪/記分/ .java-記錄器 WebConfig

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.AWSXRayRecorderBuilder;
import com.amazonaws.xray.plugins.EC2Plugin;
import com.amazonaws.xray.plugins.ElasticBeanstalkPlugin;
import com.amazonaws.xray.strategy.sampling.LocalizedSamplingStrategy;

@Configuration
public class WebConfig {
...
```

```
  static {
    AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder.standard().withPlugin(new
 EC2Plugin()).withPlugin(new ElasticBeanstalkPlugin());

    URL ruleFile = WebConfig.class.getResource("/sampling-rules.json");
    builder.withSamplingStrategy(new LocalizedSamplingStrategy(ruleFile));

    AWSXRay.setGlobalRecorder(builder.build());
  }
}
```

SDK 也會使用外掛程式設定來設定區origin段上的欄位。這表示運行您的應用程序的 AWS 資源的類型。當您使用多個外掛程式時，SDK 會使用下列解析順序來判斷來源： ElasticBeanstalk > EKS > ECS > EC2。

抽樣規則

SDK 會使用您在 X-Ray 主控台中定義的取樣規則來決定要記錄的要求。預設規則會每秒追蹤第一個要求，而所有服務的任何其他要求的百分之五會傳送追蹤至 X-Ray。在 X-Ray 主控台中建立其他規則，以自訂為每個應用程式記錄的資料量。

SDK 會依定義規則的順序套用自訂規則。如果要求符合多個自訂規則，SDK 只會套用第一個規則。

> **Note**
>
> 如果 SDK 無法達到 X-Ray 以取得取樣規則，它會每秒還原為第一個要求的預設本機規則，而每台主機的任何其他要求的百分之五。如果主機沒有調用採樣 API 的權限，或者無法連接到 X-Ray 守護程序（作為 SDK 發出的 API 調用的 TCP 代理），則可能會發生這種情況。

您也可以將 SDK 設定為從 JSON 文件載入取樣規則。SDK 可以使用本機規則做為無法使用 X-Ray 取樣的情況的備份，或僅使用本機規則。

Example 採樣規則

```
{
  "version": 2,
  "rules": [
    {
      "description": "Player moves.",
      "host": "*",
      "http_method": "*",
```

```
      "url_path": "/api/move/*",
      "fixed_target": 0,
      "rate": 0.05
    }
  ],
  "default": {
    "fixed_target": 1,
    "rate": 0.1
  }
}
```

此範例定義了一個自訂規則和預設規則。自訂規則會套用百分之五的取樣率，而且沒有追蹤下限路徑的要求數目下/api/move/限。預設規則會每秒追蹤第一個要求，以及 10% 的額外要求。

在本機定義規則的缺點是，固定目標會由記錄器的每個執行個體獨立套用，而不是由 X-Ray 服務管理。當您部署更多主機時，固定費率會倍增，因此更難以控制記錄的資料量。

開啟時 AWS Lambda，您無法修改取樣率。如果您的函數是由已檢測的服務呼叫，則 Lambda 會記錄產生由該服務取樣之請求的呼叫。如果啟用主動追蹤且沒有追蹤標頭，Lambda 會做出取樣決策。

若要在 Spring 中提供備份規則，請在組態類別中使用 CentralizedSamplingStrategy 設定全域記錄器：

Example src/主/java/我的應用程序/. Java WebConfig-記錄器配置

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.AWSXRayRecorderBuilder;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;
import com.amazonaws.xray.plugins.EC2Plugin;
import com.amazonaws.xray.strategy.sampling.LocalizedSamplingStrategy;

@Configuration
public class WebConfig {

  static {
  AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder.standard().withPlugin(new
 EC2Plugin());

  URL ruleFile = WebConfig.class.getResource("/sampling-rules.json");
  builder.withSamplingStrategy(new CentralizedSamplingStrategy(ruleFile));

  AWSXRay.setGlobalRecorder(builder.build());
}
```

若是 Tomcat，請新增接聽程式以擴展 ServletContextListener，並在部署描述項中註冊接聽程式。

Example src/com/myapp/web/Startup.java

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.AWSXRayRecorderBuilder;
import com.amazonaws.xray.plugins.EC2Plugin;
import com.amazonaws.xray.strategy.sampling.LocalizedSamplingStrategy;

import java.net.URL;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

public class Startup implements ServletContextListener {

    @Override
    public void contextInitialized(ServletContextEvent event) {
        AWSXRayRecorderBuilder builder =
 AWSXRayRecorderBuilder.standard().withPlugin(new EC2Plugin());

        URL ruleFile = Startup.class.getResource("/sampling-rules.json");
        builder.withSamplingStrategy(new CentralizedSamplingStrategy(ruleFile));

        AWSXRay.setGlobalRecorder(builder.build());
    }

    @Override
    public void contextDestroyed(ServletContextEvent event) { }
}
```

Example WEB-INF/web.xml

```
...
  <listener>
    <listener-class>com.myapp.web.Startup</listener-class>
  </listener>
```

若僅要使用本機規則，請將 CentralizedSamplingStrategy 取代為 LocalizedSamplingStrategy。

```
builder.withSamplingStrategy(new LocalizedSamplingStrategy(ruleFile));
```

日誌

依預設，SDK 會將ERROR層級訊息輸出至您的應用程式記錄。您可以在 SDK 上啟用除錯層級記錄，將更詳細的記錄輸出至應用程式記錄檔。有效的記錄層級為DEBUGINFOWARN、ERROR、、和FATAL。FATAL日誌級別將所有日誌消息靜音，因為 SDK 不會在致命級別進行記錄。

Example application.properties

使用 `logging.level.com.amazonaws.xray` 屬性設定記錄日誌層級。

```
logging.level.com.amazonaws.xray = DEBUG
```

當您手動產生子區段時，可使用除錯日誌來識別問題，例如未結束的子區段。

將追蹤 ID 插入日誌

若要將您日誌陳述式公開給目前的追蹤 ID，您可以將此 ID 插入到映射的診斷內容 (MDC)。使用 `SegmentListener` 界面，會在區段生命週期事件期間從 X-Ray 記錄器呼叫方法。當區段或子區段開始時，合格的追蹤 ID 會透過金鑰 `AWS-XRAY-TRACE-ID` 注入至 MDC。當該區段結束時，該索引鍵即會從 MDC 中移除。這會公開正在使用的記錄程式庫追蹤 ID。當子區段結束時，其父 ID 會注入 MDC 中。

Example 完整的合格追蹤 ID

完整的合格 ID 會表示為 `TraceID@EntityID`

```
1-5df42873-011e96598b447dfca814c156@541b3365be3dafc3
```

此功能適用於搭配 Java 的 AWS X-Ray SDK 測試的 Java 應用程式，並支援下列記錄設定：

- SLF4J 前端 API 與 Logback 後端
- SLF4J 前端 API 與 Log4J2 後端
- Log4J2 前端 API 與 Log4J2 後端

請參閱下列標籤，以了解每個前端和每個後端的需求。

SLF4J Frontend

1. 將以下 Maven 相依性新增到您的專案。

   ```
   <dependency>
   ```

```
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-xray-recorder-sdk-slf4j</artifactId>
    <version>2.11.0</version>
</dependency>
```

2. 建置 AWSXRayRecorder 時包含 withSegmentListener 方法。這會新增 SegmentListener 類別，將新的追蹤 ID 自動插入 SLF4J MDC。

   SegmentListener 會採用選擇性字串做為參數來設定日誌陳述式的字首。您可以透過下列方式設定字首：

   - 無 — 使用預設AWS-XRAY-TRACE-ID首碼。
   - 空白 — 使用空字串 (例如"")。
   - 自訂 — 使用字串中定義的自訂前置詞。

   Example **AWSXRayRecorderBuilder** 陳述式

   ```
   AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
           .standard().withSegmentListener(new SLF4JSegmentListener("CUSTOM-
   PREFIX"));
   ```

Log4J2 front end

1. 將以下 Maven 相依性新增到您的專案。

   ```
   <dependency>
       <groupId>com.amazonaws</groupId>
       <artifactId>aws-xray-recorder-sdk-log4j</artifactId>
       <version>2.11.0</version>
   </dependency>
   ```

2. 建置 AWSXRayRecorder 時包含 withSegmentListener 方法。這會新增 SegmentListener 類別，將新的完整合格追蹤 ID 自動注入 SLF4J MDC。

   SegmentListener 會採用選擇性字串做為參數來設定日誌陳述式的字首。您可以透過下列方式設定字首：

   - 無 — 使用預設AWS-XRAY-TRACE-ID首碼。
   - 空白 — 使用空字串 (例如"") 並移除前置字元。

- 自訂 — 使用字串中定義的自訂前置詞。

Example **`AWSXRayRecorderBuilder`** 陳述式

```
AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
        .standard().withSegmentListener(new Log4JSegmentListener("CUSTOM-
PREFIX"));
```

Logback backend

若要將追蹤 ID 插入到日誌事件中，您必須修改記錄器的 `PatternLayout`，這會格式化每個記錄陳述式。

1. 尋找 `patternLayout` 的設定位置。您可透過編寫程式的方式，或透過 XML 組態檔案執行此作業。若要深入了解，請參閱 Logback 組態。
2. 將 `%X{AWS-XRAY-TRACE-ID}` 插入到 `patternLayout` 的任何位置，可將追蹤 ID 插入未來的記錄陳述式。`%X{}` 表示您正在使用 MDC 提供的索引鍵擷取值。若要 PatternLayouts 在登入中進一步了解，請參閱PatternLayout。

Log4J2 backend

1. 尋找 `patternLayout` 的設定位置。您可透過編寫程式的方式，或透過以 XML、JSON、YAML 或屬性格式編寫的組態檔案執行此作業。

   若要深入了解如何透過組態檔案設定 Log4J2，請參閱組態。

   若要深入了解如何透過編寫程式的方式設定 Log4J2，請參閱透過編寫程式方式的組態。
2. 將 `%X{AWS-XRAY-TRACE-ID}` 插入到 `PatternLayout` 的任何位置，可將追蹤 ID 插入未來的記錄陳述式。`%X{}` 表示您正在使用 MDC 提供的索引鍵擷取值。要了解有關 Log4J2 PatternLayouts 的更多信息，請參閱模式佈局。

插入追蹤 ID 範例

以下示範包含追蹤 ID 的修改後 `PatternLayout` 字串。追蹤 ID 會列印在執行緒名稱 (`%t`) 之後、日誌層級 (`%-5p`) 之前。

Example 插入 ID 的 **PatternLayout**

```
%d{HH:mm:ss.SSS} [%t] %X{AWS-XRAY-TRACE-ID} %-5p %m%n
```

AWS X-Ray 自動打印日誌語句中的密鑰和跟踪 ID，以便於解析。以下顯示使用修改後 PatternLayout 的日誌說明。

Example 插入 ID 的日誌說明

```
2019-09-10 18:58:30.844 [nio-5000-exec-4]  AWS-XRAY-TRACE-ID:
 1-5d77f256-19f12e4eaa02e3f76c78f46a@1ce7df03252d99e1 WARN 1 - Your logging message
 here
```

記錄訊息本身以 %m 模式包裝，在呼叫記錄器時設定。

區段接聽程式

區段接聽程式是攔截生命週期事件 (例如由 AWSXRayRecorder 產生的開始和結束區段) 的介面。區段接聽程式事件函數的實作可能是在透過 onBeginSubsegment 建立時，將相同的註釋新增至所有子區段、使用 afterEndSegment 將每個區段傳送到精靈後記錄訊息，或者透過 beforeEndSubsegment記錄由 SQL 攔截器傳送的查詢，以驗證子區段是否代表 SQL 查詢，如果是的話，則會新增額外的中繼資料。

若要查看完整的SegmentListener函數清單，請參閱適用於 Java API 的AWS X-Ray 記錄器 SDK 的文件。

下列範例顯示如何在建立 onBeginSubsegment 時將一致的註釋加入所有子區段，以及透過 afterEndSegment 在每個區段結尾列印記錄訊息。

Example MySegmentListener. 爪哇

```
import com.amazonaws.xray.entities.Segment;
import com.amazonaws.xray.entities.Subsegment;
import com.amazonaws.xray.listeners.SegmentListener;

public class MySegmentListener implements SegmentListener {
    .....

    @Override
    public void onBeginSubsegment(Subsegment subsegment) {
        subsegment.putAnnotation("annotationKey", "annotationValue");
    }
```

```
    @Override
    public void afterEndSegment(Segment segment) {
        // Be mindful not to mutate the segment
        logger.info("Segment with ID " + segment.getId());
    }
}
```

在建立 AWSXRayRecorder 時參考此自訂區段接聽程式。

Example AWSXRayRecorderBuilder 聲明

```
AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
        .standard().withSegmentListener(new MySegmentListener());
```

環境變數

您可以使用環境變數來設定適用於 Java 的 X-Ray SDK。軟體開發套件支援以下變數。

- AWS_XRAY_TRACING_NAME— 設定 SDK 用於區段的服務名稱。覆寫您在 servlet 篩選條件的區段命名策略中設定的服務名稱。
- AWS_XRAY_DAEMON_ADDRESS— 設定 X-Ray 精靈監聽程式的主機和連接埠。根據預設，SDK 會同時用127.0.0.1:2000於追蹤資料 (UDP) 和取樣 (TCP)。如果您已將協助程式設定為在不同的連接埠上接聽，或是在不同的主機上執行，請使用此變數。

  格式

  - 相同的連接埠 — *address*:*port*
  - 不同的端口-tcp:*address*:*port* udp:*address*:*port*
- AWS_XRAY_CONTEXT_MISSING— 設定為當您RUNTIME_ERROR的檢測程式碼嘗試在沒有區段開啟時記錄資料時擲回例外狀況。

  有效值

  - RUNTIME_ERROR— 擲回執行階段例外狀況。
  - LOG_ERROR— 記錄錯誤並繼續 (預設值)。
  - IGNORE_ERROR— 忽略錯誤並繼續。

  當您嘗試在沒有要求開啟時執行的啟動程式碼中使用已檢測的用戶端，或在產生新執行緒的程式碼中使用已檢測的用戶端時，可能會發生與遺失區段或子區段相關的錯誤。

環境變數會覆寫程式碼中所設的同等系統屬性和值。

系統屬性

您可以將系統屬性做為環境變數的 JVM 專用替代方案。開發套件支援以下屬性：

- `com.amazonaws.xray.strategy.tracingName`— 相當於`AWS_XRAY_TRACING_NAME`.
- `com.amazonaws.xray.emitters.daemonAddress`— 相當於`AWS_XRAY_DAEMON_ADDRESS`.
- `com.amazonaws.xray.strategy.contextMissingStrategy`— 相當於`AWS_XRAY_CONTEXT_MISSING`.

如果同時設定了系統屬性和同等環境變數，則會使用環境變數的值。兩種方法都會覆寫程式碼中所設的值。

使用適用於 Java 的 X-Ray SDK 追蹤傳入的要求

您可以使用 X-Ray 開發套件來追蹤應用程式在 Amazon EC2 或 Amazon ECS 的 EC2 執行個體上提供的傳入 HTTP 請求。 AWS Elastic Beanstalk

使用 `Filter` 檢測傳入的 HTTP 請求。當您將 X-Ray Servlet 篩選器新增至應用程式時，Java 適用的 X-Ray SDK 會為每個取樣要求建立一個區段。此區段包括時間、方法，以及 HTTP 請求的處置方式。其他檢測會在此區段上建立子區段。

> **Note**
>
> 對於 AWS Lambda 函數，Lambda 會為每個取樣的請求建立區段。如需詳細資訊，請參閱AWS Lambda 而且 AWS X-Ray。

每個區段都有一個名稱，可在服務對應中識別您的應用程式。區段可以以靜態方式命名，也可以設定 SDK 根據傳入要求中的主機標頭動態命名該區段。動態命名可讓您根據要求中的網域名稱對追蹤進行分組，並在名稱與預期的模式不符時套用預設名稱 (例如，如果主機標頭是偽造的)。

> **轉寄的要求**
>
> 如果負載平衡器或其他中介機構將要求轉寄至您的應用程式，X-Ray 會從要求中的`X-Forwarded-For`標頭取得用戶端 IP，而不是從 IP 封包中的來源 IP 取得。為轉寄要求所記錄的用戶端 IP 可以偽造，因此不應該受信任。

轉送請求時，SDK 會在區段中設定一個額外的欄位來指出這一點。如果區段包含`x_forwarded_for`設定為的欄位`true`，則會從 HTTP 要求中的`X-Forwarded-For`標頭取得用戶端 IP。

訊息處理常式會使用 `http` 區塊為每個傳入的請求建立區段，其中包含以下資訊：

- HTTP 方法-獲取，發布，把，刪除等。
- [用戶端位址] — 傳送要求的用戶端 IP 位址。
- 回應碼 — 已完成要求的 HTTP 回應碼。
- 時間 — 開始時間 (收到請求的時間) 和結束時間 (傳送回應的時間)。
- 使用者代理程式 — `user-agent` 來自請求的。
- 「內容長度」— 響應`content-length`中的內容。

章節

- 將追蹤篩選條件新增至應用程式 (Tomcat)
- 將追蹤篩選條件新增至應用程式 (Spring)
- 設定區段命名策略

將追蹤篩選條件新增至應用程式 (Tomcat)

若是 Tomcat，請將 `<filter>` 新增至您專案的 `web.xml` 檔案。使用 `fixedName` 參數，指定要套用至針對傳入請求建立之區段的服務名稱。

Example WEB-INF/web.xml - Tomcat

```
<filter>
  <filter-name>AWSXRayServletFilter</filter-name>
  <filter-class>com.amazonaws.xray.javax.servlet.AWSXRayServletFilter</filter-class>
  <init-param>
    <param-name>fixedName</param-name>
    <param-value>MyApp</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>AWSXRayServletFilter</filter-name>
  <url-pattern>*</url-pattern>
</filter-mapping>
```

將追蹤篩選條件新增至應用程式 (Spring)

若是 Spring，請將 `Filter` 新增至您的 `WebConfig` 類別。以字串形式將區段名稱傳遞給 `AWSXRayServletFilter` 建構函數。

Example SRC /主/爪/我的/WebConfig. Java-春天

```
package myapp;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Bean;
import javax.servlet.Filter;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;

@Configuration
public class WebConfig {

  @Bean
  public Filter TracingFilter() {
    return new AWSXRayServletFilter("Scorekeep");
  }
}
```

設定區段命名策略

AWS X-Ray 使用服務名稱來識別您的應用程式，並將其與應用程式使用的其他應用程式、資料庫、外部 API 和 AWS 資源區分開來。X-Ray SDK 為傳入要求產生區段時，會在區段的名稱欄位中記錄應用程式的服務名稱。

X-Ray SDK 可以在 HTTP 要求標頭中的主機名稱之後命名區段。但是，此標頭可能會被偽造，這可能會導致服務對應中出現非預期的節點。若要防止 SDK 因為具有偽造主機標頭的要求而不正確地命名區段，您必須為傳入要求指定預設名稱。

如果您的應用程式為多個網域提供要求，您可以將 SDK 設定為使用動態命名策略，在區段名稱中反映此問題。動態命名策略可讓 SDK 針對符合預期模式的要求使用主機名稱，並將預設名稱套用至不符合要求的要求。

例如，您可能有一個應用程式向三個子網域提供要求 — `www.example.com` `api.example.com`、和`static.example.com`。您可以將動態命名策略與模式搭配使用，`*.example.com`以識別具有不同名稱的每個子網域的區段，從而在服務對應上產生三個服務節點。如果您的應用程式收到的主機名稱與模式不相符的要求，您會在服務對應上看到第四個節點，其中包含您指定的後援名稱。

若要為所有請求區段使用相同的名稱，請如上一節所述，在初始化 servlet 篩選條件時指定您應用程式的名稱。這SegmentNamingStrategy.fixed()與通過調用並將其傳遞給AWSXRayServletFilter構造函數SegmentNamingStrategy來創建固定的效果相同。

> **Note**
>
> 您可以使用 AWS_XRAY_TRACING_NAME 環境變數來覆寫您在程式碼中定義的預設服務名稱。

動態命名策略可定義主機名稱應相符的模式，以及如果 HTTP 請求中的主機名稱不符合模式時要使用的預設名稱。若要在 Tomcat 中動態命名區段，請分別使用 dynamicNamingRecognizedHosts 和 dynamicNamingFallbackName 來定義模式和預設名稱。

Example WEB-INF/web.xml - 使用動態命名的 Servlet 篩選條件

```
<filter>
  <filter-name>AWSXRayServletFilter</filter-name>
  <filter-class>com.amazonaws.xray.javax.servlet.AWSXRayServletFilter</filter-class>
  <init-param>
    <param-name>dynamicNamingRecognizedHosts</param-name>
    <param-value>*.example.com</param-value>
  </init-param>
  <init-param>
    <param-name>dynamicNamingFallbackName</param-name>
    <param-value>MyApp</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>AWSXRayServletFilter</filter-name>
  <url-pattern>*</url-pattern>
</filter-mapping>
```

對於 Spring，通SegmentNamingStrategy過調用創建一個動態SegmentNamingStrategy.dynamic()，並將其傳遞給AWSXRayServletFilter構造函數。

Example src/主/java/我的應用程序/ .java-具有動態命名WebConfig的 SERVLET 過濾器

```
package myapp;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Bean;
import javax.servlet.Filter;
```

```
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;
import com.amazonaws.xray.strategy.SegmentNamingStrategy;

@Configuration
public class WebConfig {

  @Bean
  public Filter TracingFilter() {
    return new AWSXRayServletFilter(SegmentNamingStrategy.dynamic("MyApp",
 "*.example.com"));
  }
}
```

使用適用於 Java 的 X-Ray AWS SDK 追蹤 SDK 呼叫

當您的應用程式呼叫 AWS 服務 以儲存資料、寫入佇列或傳送通知時，Java 的 X-Ray SDK 會追蹤子區段中下游的呼叫。您在這些服務中存取的追蹤 AWS 服務 和資源 (例如，Amazon S3 儲存貯體或 Amazon SQS 佇列) 會在 X-Ray 主控台的追蹤對應上顯示為下游節點。

當您在組建中包含`aws-sdk`和`aws-sdk-instrumentor`子模組時，SDK for Java 的 X-Ray 開發套件會自動檢測所有 AWS 開發套件用戶端。如果您未包含 Instrumentor 子模組，您可以選擇檢測某些特定用戶端，而排除其他用戶端。

要檢測單個客戶端，請從構建中刪除`aws-sdk-instrumentor`子模塊，然後使用服務`XRayClient`的客戶端構建器在 AWS SDK 客戶端`TracingHandler`上添加一個。

例如，若要檢測 AmazonDynamoDB 用戶端，請將追蹤處理常式傳遞至 `AmazonDynamoDBClientBuilder`。

Example MyModel. Java DynamoDB 戶端

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.handlers.TracingHandler;

...
public class MyModel {
  private AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
        .withRegion(Regions.fromName(System.getenv("AWS_REGION")))
        .withRequestHandlers(new TracingHandler(AWSXRay.getGlobalRecorder()))
        .build();
...
```

對於所有服務，您都可以在 X-Ray 控制台中看到調用的 API 的名稱。對於服務子集，X-Ray SDK 會將資訊新增至區段，以在服務對應中提供更多精細度。

例如，當您使用已檢測的 DynamoDB 用戶端進行呼叫時，SDK 會將資料表名稱新增至區段，以便針對以資料表為目標的呼叫。在主控台中，每個表格在服務對應中顯示為獨立節點，其中包含一般 DynamoDB 節點，用於未針對資料表的呼叫。

Example 呼叫 DynamoDB 以儲存項目的子區段

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

您存取具名資源時，對以下服務的呼叫會在服務地圖中建立額外節點。未針對特定資源的呼叫，則會建立服務的一般節點。

- Amazon DynamoDB — 表名稱
- Amazon 簡單存儲服務 — 存儲桶和密鑰名稱
- Amazon 簡單隊列服務-隊列名稱

要 AWS 服務 使用 AWS SDK for Java 2.2 及更高版本檢測下游調用，您可以從構建配置中省略該`aws-xray-recorder-sdk-aws-sdk-v2-instrumentor`模塊。這時改成包含 `aws-xray-recorder-sdk-aws-sdk-v2` module，然後使用 `TracingInterceptor` 為其進行設定，檢測個別的用戶端。

Example AWS SDK for Java 2.2 及更高版本-跟踪攔截器

```
import com.amazonaws.xray.interceptors.TracingInterceptor;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
//...
public class MyModel {
private DynamoDbClient client = DynamoDbClient.builder()
.region(Region.US_WEST_2)
.overrideConfiguration(ClientOverrideConfiguration.builder()
.addExecutionInterceptor(new TracingInterceptor())
.build()
)
.build();
//...
```

使用適用於 Java 的 X-Ray SDK 追蹤對下游 HTTP 網路服務的呼叫

當您的應用程式呼叫微服務或公用 HTTP API 時，您可以使用 Java 版本的 X-Ray SDK `HttpClient` 來檢測這些呼叫，並將 API 作為下游服務新增至服務圖形。

適用於 Java 的 X-Ray SDK 包括`DefaultHttpClient`和可用於代替 Apache HttpComponents 等效用於檢測傳出 HTTP 呼叫的`HttpClientBuilder`類別。

- `com.amazonaws.xray.proxies.apache.http.DefaultHttpClient` - `org.apache.http.impl.client.DefaultHttpClient`
- `com.amazonaws.xray.proxies.apache.http.HttpClientBuilder` - `org.apache.http.impl.client.HttpClientBuilder`

這些程式庫位於 [aws-xray-recorder-sdk-apache-http] 子模組中。

您可以使用相當於檢測所有用戶端的 X-Ray 來取代現有的匯入陳述式，或者在初始化用戶端以檢測特定用戶端時使用完整名稱。

Example HttpClientBuilder

```
import com.fasterxml.jackson.databind.ObjectMapper;
import org.apache.http.HttpEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.CloseableHttpClient;
```

```
import org.apache.http.util.EntityUtils;
import com.amazonaws.xray.proxies.apache.http.HttpClientBuilder;
...
  public String randomName() throws IOException {
    CloseableHttpClient httpclient = HttpClientBuilder.create().build();
    HttpGet httpGet = new HttpGet("http://names.example.com/api/");
    CloseableHttpResponse response = httpclient.execute(httpGet);
    try {
      HttpEntity entity = response.getEntity();
      InputStream inputStream = entity.getContent();
      ObjectMapper mapper = new ObjectMapper();
      Map<String, String> jsonMap = mapper.readValue(inputStream, Map.class);
      String name = jsonMap.get("name");
      EntityUtils.consume(entity);
      return name;
    } finally {
      response.close();
    }
  }
```

當您檢測對下游 Web API 的調用時，Java 的 X-Ray SDK 會記錄一個子段，其中包含有關 HTTP 請求和響應的信息。X-Ray 會使用子區段來產生遠端 API 的推斷區段。

Example 下游 HTTP 呼叫的子區段

```
{
  "id": "004f72be19cddc2a",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "name": "names.example.com",
  "namespace": "remote",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  }
}
```

Example 下游 HTTP 呼叫的推斷區段

```
{
  "id": "168416dc2ea97781",
  "name": "names.example.com",
  "trace_id": "1-62be1272-1b71c4274f39f122afa64eab",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "parent_id": "004f72be19cddc2a",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  },
  "inferred": true
}
```

使用適用於 Java 的 X-Ray 開發套件追蹤 SQL 查詢

攔截器

通過將 Java JDBC 攔截器的 X-Ray SDK 添加到您的數據源配置中來檢測 SQL 數據庫查詢。

- PostgreSQL – `com.amazonaws.xray.sql.postgres.TracingInterceptor`
- MySQL – `com.amazonaws.xray.sql.mysql.TracingInterceptor`

這些攔截器分別位於 aws-xray-recorder-sql-postgres 和 aws-xray-recorder-sql-mysql 子模組中。他們會實作 `org.apache.tomcat.jdbc.pool.JdbcInterceptor`，並且與 Tomcat 連線集區相容。

> **Note**
>
> 基於安全性目的，SQL 攔截器不會在子區段內記錄 SQL 查詢本身。

針對 Spring，請在屬性檔案中新增攔截器，然後使用 Spring Boot 的 `DataSourceBuilder` 建置資料來源。

Example **`src/main/java/resources/application.properties`** - PostgreSQL JDBC 攔截器

```
spring.datasource.continue-on-error=true
spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto=create-drop
spring.datasource.jdbc-interceptors=com.amazonaws.xray.sql.postgres.TracingInterceptor
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQL94Dialect
```

Example **`src/main/java/myapp/WebConfig.java`** - 資料來源

```
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.jdbc.DataSourceBuilder;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

import javax.servlet.Filter;
import javax.sql.DataSource;
import java.net.URL;

@Configuration
@EnableAutoConfiguration
@EnableJpaRepositories("myapp")
public class RdsWebConfig {

  @Bean
  @ConfigurationProperties(prefix = "spring.datasource")
  public DataSource dataSource() {
      logger.info("Initializing PostgreSQL datasource");
      return DataSourceBuilder.create()
              .driverClassName("org.postgresql.Driver")
              .url("jdbc:postgresql://" + System.getenv("RDS_HOSTNAME") + ":" +
 System.getenv("RDS_PORT") + "/ebdb")
              .username(System.getenv("RDS_USERNAME"))
              .password(System.getenv("RDS_PASSWORD"))
              .build();
  }
...
```

```
}
```

對於 Tomcat，請`setJdbcInterceptors`使用 Java 類別的 X-Ray SDK 參考來呼叫 JDBC 資料來源。

Example **`src/main/myapp/model.java`** - 資料來源

```
import org.apache.tomcat.jdbc.pool.DataSource;
...
DataSource source = new DataSource();
source.setUrl(url);
source.setUsername(user);
source.setPassword(password);
source.setDriverClassName("com.mysql.jdbc.Driver");
source.setJdbcInterceptors("com.amazonaws.xray.sql.mysql.TracingInterceptor;");
```

Tomcat JDBC 資料來源程式庫包含在適用於 Java 的 X-Ray SDK 中，但您可以將其宣告為提供的相依性，以記錄您使用它。

Example **`pom.xml`** - JDBC 資料來源

```
<dependency>
  <groupId>org.apache.tomcat</groupId>
  <artifactId>tomcat-jdbc</artifactId>
  <version>8.0.36</version>
  <scope>provided</scope>
</dependency>
```

本機 SQL 跟踪裝飾器

- 添加[aws-xray-recorder-sdk-sql]到您的依賴關係。
- 裝飾您的數據庫數據源，連接或語句。

```
dataSource = TracingDataSource.decorate(dataSource)
connection = TracingConnection.decorate(connection)
statement = TracingStatement.decorateStatement(statement)
preparedStatement = TracingStatement.decoratePreparedStatement(preparedStatement,
 sql)
callableStatement = TracingStatement.decorateCallableStatement(callableStatement,
 sql)
```

使用適用於 Java 的 X-Ray SDK 產生自訂子區段

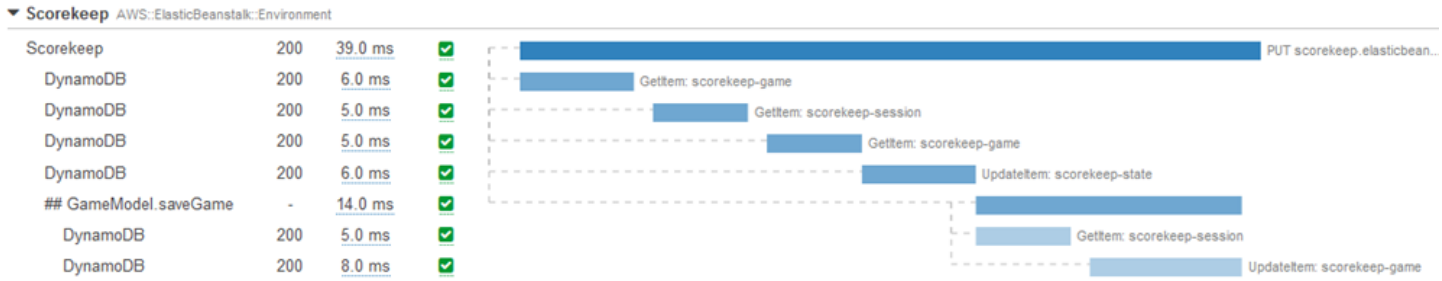
子段擴展了跟踪的段，其中包含有關完成工作的詳細信息，以滿足請求。每次您使用已檢測的用戶端進行呼叫時，X-Ray SDK 都會記錄在子區段中產生的資訊。您可以建立其他子區段來分組其他子區段、測量程式碼區段的效能，或是記錄註釋和中繼資料。

若要管理子區段，請使用 beginSubsegment 和 endSubsegment 方法。

Example GameModel.java-自定義子段

```
import com.amazonaws.xray.AWSXRay;
...
  public void saveGame(Game game) throws SessionNotFoundException {
    // wrap in subsegment
    Subsegment subsegment = AWSXRay.beginSubsegment("Save Game");
    try {
      // check session
      String sessionId = game.getSession();
      if (sessionModel.loadSession(sessionId) == null ) {
        throw new SessionNotFoundException(sessionId);
      }
      mapper.save(game);
    } catch (Exception e) {
      subsegment.addException(e);
      throw e;
    } finally {
      AWSXRay.endSubsegment();
    }
  }
```

在此範例中，子區段中的程式碼會以工作階段模型上的方法從 DynamoDB 載入遊戲的工作階段，並使用的 DynamoDB 對應程式來儲存遊戲。 AWS SDK for Java將此程式碼封裝在子區段中，會在主控台的追蹤檢視中呼叫Save Game子區段的 DynamoDB 子項。



如果子區段中的程式碼擲回已檢查的例外狀況，請將其包裝在 `try` 區塊中，並在 `finally` 區塊中呼叫 `AWSXRay.endSubsegment()`，以確保子區段一律關閉。如果子段未封閉，則無法完成父段，也不會將其傳送至 X-Ray。

對於不會拋出檢查異常的代碼，您可以將代碼`AWSXRay.CreateSubsegment`作為 Lambda 函數傳遞給。

Example 子區段 Lambda 函數

```
import com.amazonaws.xray.AWSXRay;

AWSXRay.createSubsegment("getMovies", (subsegment) -> {
    // function code
});
```

當您在區段或其他子區段內建立子區段時，Java 的 X-Ray SDK 會產生該區段的 ID，並記錄開始時間和結束時間。

Example 使用中繼資料的子區段

```
"subsegments": [{
  "id": "6f1605cd8a07cb70",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "Custom subsegment for UserModel.saveUser function",
  "metadata": {
    "debug": {
      "test": "Metadata string from UserModel.saveUser"
    }
  },
```

對於非同步和多執行緒程式設計，您必須手動將子區段傳遞給`endSubsegment()`方法，以確保其正確關閉，因為 X-Ray 環境可能會在非同步執行期間修改。如果非同步子區段在其父段關閉後關閉，此方法會自動將整個區段串流至 X-Ray 精靈。

Example 非同步子段

```

@GetMapping("/api")
public ResponseEntity<?> api() {
  CompletableFuture.runAsync(() -> {
      Subsegment subsegment = AWSXRay.beginSubsegment("Async Work");
```

```
        try {
            Thread.sleep(3000);
        } catch (InterruptedException e) {
            subsegment.addException(e);
            throw e;
        } finally {
            AWSXRay.endSubsegment(subsegment);
        }
    });
    return ResponseEntity.ok().build();
}
```

使用適用於 Java 的 X-Ray SDK 將註釋和中繼資料新增至區段

您可以使用註釋和中繼資料來記錄有關請求、環境或應用程式的其他資訊。您可以將註釋和中繼資料新增至 X-Ray SDK 建立的區段，或新增至您建立的自訂子區段。

註釋是與字符串，數字或布爾值鍵-值對。註釋會編製索引以與篩選器運算式搭配使用 使用標記記錄您想要用來在主控台將追蹤分組的資料，或是在呼叫 `GetTraceSummaries` API 時使用標記。

中繼資料是索引鍵-值配對，可以具有任何類型的值 (包括物件和清單)，但不會編製索引以供篩選運算式使用。使用元數據記錄要存儲在跟踪中但不需要與搜索一起使用的其他數據。

除了註釋和中繼資料，您也可以在區段上記錄使用者 ID 字串。區段會將使用者 ID 記錄在單獨的欄位中，並建立索引以用於搜尋。

章節

- 使用適用於 Java 的 X-Ray SDK 記錄註釋
- 使用適用於 Java 的 X-Ray SDK 記錄中繼資料
- 使用適用於 Java 的 X-Ray SDK 記錄使用者 ID

使用適用於 Java 的 X-Ray SDK 記錄註釋

針對您想要建立索引以用於搜尋的區段或子區段，請使用標註來記錄這些區段上的資訊。

註釋要求

- 按鍵 — X-Ray 註解的金鑰最多可包含 500 個英數字元。您不能使用底線符號 (_) 以外的空格或符號。
- 值 — X-Ray 註釋的值最多可包含 1,000 個 Unicode 字元。

- 註釋的數量 — 每個追蹤最多可以使用 50 個註釋。

記錄標註

1. 從 AWSXRay 取得目前區段或子區段的參考。

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Segment;
...
Segment document = AWSXRay.getCurrentSegment();
```

或

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Subsegment;
...
Subsegment document = AWSXRay.getCurrentSubsegment();
```

2. 使用字串索引鍵、布林值、數字或字串值，呼叫 putAnnotation。

```
document.putAnnotation("mykey", "my value");
```

軟體開發套件會將標註以鍵/值對記錄在區段文件中的 annotations 物件內。若使用相同鍵呼叫 putAnnotation 兩次，則會覆寫之前在相同區段或子區段上記錄的值。

若要尋找具有特定值之註釋的繪線，請在篩選器運算式中使用annotations.*key*關鍵字。

Example **src/main/java/scorekeep/GameModel.java**— 註釋和元數據

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Segment;
import com.amazonaws.xray.entities.Subsegment;
...
  public void saveGame(Game game) throws SessionNotFoundException {
    // wrap in subsegment
    Subsegment subsegment = AWSXRay.beginSubsegment("## GameModel.saveGame");
    try {
      // check session
      String sessionId = game.getSession();
      if (sessionModel.loadSession(sessionId) == null ) {
```

```
        throw new SessionNotFoundException(sessionId);
      }
      Segment segment = AWSXRay.getCurrentSegment();
      subsegment.putMetadata("resources", "game", game);
      segment.putAnnotation("gameid", game.getId());
      mapper.save(game);
    } catch (Exception e) {
      subsegment.addException(e);
      throw e;
    } finally {
      AWSXRay.endSubsegment();
    }
  }
```

使用適用於 Java 的 X-Ray SDK 記錄中繼資料

針對您不想要建立索引以用於搜尋的區段，請使用中繼資料來記錄這些區段或子區段上的資訊。中繼資料值可以是字串、數字、布林值，或可序列化為 JSON 物件或陣列的任何物件。

記錄中繼資料

1. 從 AWSXRay 取得目前區段或子區段的參考。

   ```
   import com.amazonaws.xray.AWSXRay;
   import com.amazonaws.xray.entities.Segment;
   ...
   Segment document = AWSXRay.getCurrentSegment();
   ```

   或

   ```
   import com.amazonaws.xray.AWSXRay;
   import com.amazonaws.xray.entities.Subsegment;
   ...
   Subsegment document = AWSXRay.getCurrentSubsegment();
   ```

2. 使用字串命名空間、字串索引鍵、布林值、數字、字串或物件值，呼叫 putMetadata。

   ```
   document.putMetadata("my namespace", "my key", "my value");
   ```

   或

   只使用鍵和值呼叫 putMetadata。

```
document.putMetadata("my key", "my value");
```

若您沒有指定命名空間，軟體開發套件會使用 default。若使用相同鍵呼叫 putMetadata 兩次，則會覆寫之前在相同區段或子區段上記錄的值。

Example **src/main/java/scorekeep/GameModel.java**— 註釋和元數據

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Segment;
import com.amazonaws.xray.entities.Subsegment;
...
  public void saveGame(Game game) throws SessionNotFoundException {
    // wrap in subsegment
    Subsegment subsegment = AWSXRay.beginSubsegment("## GameModel.saveGame");
    try {
      // check session
      String sessionId = game.getSession();
      if (sessionModel.loadSession(sessionId) == null ) {
        throw new SessionNotFoundException(sessionId);
      }
      Segment segment = AWSXRay.getCurrentSegment();
      subsegment.putMetadata("resources", "game", game);
      segment.putAnnotation("gameid", game.getId());
      mapper.save(game);
    } catch (Exception e) {
      subsegment.addException(e);
      throw e;
    } finally {
      AWSXRay.endSubsegment();
    }
  }
```

使用適用於 Java 的 X-Ray SDK 記錄使用者 ID

記錄請求區段上的使用者 ID 以識別傳送請求的使用者。

記錄使用者 ID

1. 從 AWSXRay 取得目前區段的參考。

```
import com.amazonaws.xray.AWSXRay;
```

```
import com.amazonaws.xray.entities.Segment;
...
Segment document = AWSXRay.getCurrentSegment();
```

2. 使用傳送請求之使用者的字串 ID 呼叫 `setUser`。

```
document.setUser("U12345");
```

您可以在控制器中呼叫 `setUser`，以在應用程式開始處理請求時馬上記錄使用者 ID。如果您只要使用區段來設定使用者 ID，可以將呼叫鏈結為單行。

Example src /主/爪/記分/. Java MoveController-用戶識別碼

```
import com.amazonaws.xray.AWSXRay;
...
  @RequestMapping(value="/{userId}", method=RequestMethod.POST)
  public Move newMove(@PathVariable String sessionId, @PathVariable String
 gameId, @PathVariable String userId, @RequestBody String move) throws
 SessionNotFoundException, GameNotFoundException, StateNotFoundException,
 RulesException {
    AWSXRay.getCurrentSegment().setUser(userId);
    return moveFactory.newMove(sessionId, gameId, userId, move);
  }
```

若要尋找使用者 ID 的追蹤，請在篩選運算式中使用`user`關鍵字。

AWS X-Ray 用於 Java 的 X-Ray SDK 的指標

本主題說明命 AWS X-Ray 名空間、測量結果和維度。您可以使用適用於 Java 的 X-Ray 開發套件，從收集到的 X-Ray 區段發佈未取樣的 Amazon CloudWatch 指標。這些指標衍生自區段的開始和結束時間，以及錯誤、故障和節流狀態標記。您可以使用這些追蹤指標，公開子區段內的重試和相依性問題。

CloudWatch 本質上是一個指標存儲庫。量度是中的基本概念 CloudWatch ，代表一組時間順序的資料點。您 (或 AWS 服務) 將量度資料點發佈至， CloudWatch 並擷取有關這些資料點的統計資料，做為一組排序的時間序列資料。

指標是由名稱、命名空間和一個或多個維度做唯一的定義。每個資料點都有時間戳記和可選的測量單位。當您請求統計資料時，傳回的資料流是藉由命名空間、指標名稱和維度做識別。

如需有關的詳細資訊 CloudWatch，請參閱 Amazon CloudWatch 使用者指南。

X-Ray CloudWatch 指標

ServiceMetrics/SDK 命名空間包含下列指標。

指標	統計資訊可用	描述	個單位
Latency	平均、最小、最大、計數	開始與結束時間之間的差異。平均、最小和最大皆描述操作延遲。計數描述呼叫計數。	毫秒
ErrorRate	平均數、總和	失敗原因為 4xx Client Error 狀態碼的請求速率，導致錯誤。	百分比
FaultRate	平均數、總和	失敗原因為 5xx Server Error 狀態碼的追蹤速率，導致故障。	百分比
ThrottleRate	平均數、總和	傳回 429 狀態碼的節流追蹤速率。這是 ErrorRate 指標的一部分。	百分比
OkRate	平均數、總和	產生 OK 狀態碼的追蹤請求速率。	百分比

X-Ray CloudWatch 尺寸

您可以使用下表中的維度，精簡為 X-Ray 檢測的 Java 應用程式傳回的量度。

維度	描述
ServiceType	如不清楚，即為服務的類型，例如，AWS::EC2::Instance 或 NONE。

維度	描述
ServiceName	服務的正式名稱。

啟用 X-Ray CloudWatch 指標

請使用下列程序，在已檢測的 Java 應用程式中啟用追蹤指標。

設定追蹤指標

1. 將 `aws-xray-recorder-sdk-metrics` 套件新增為 Maven 相依性。如需詳細資訊，請參閱適用於 Java 子模組的 X-Ray SDK。
2. 啟用新的 `MetricsSegmentListener()` 做為全域記錄器組建的一部分。

   Example src/com/myapp/web/Startup.java

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.AWSXRayRecorderBuilder;
import com.amazonaws.xray.plugins.EC2Plugin;
import com.amazonaws.xray.plugins.ElasticBeanstalkPlugin;
import com.amazonaws.xray.strategy.sampling.LocalizedSamplingStrategy;

@Configuration
public class WebConfig {
...
  static {
    AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder
                                        .standard()
                                        .withPlugin(new EC2Plugin())
                                        .withPlugin(new ElasticBeanstalkPlugin())
                                        .withSegmentListener(new
 MetricsSegmentListener());

    URL ruleFile = WebConfig.class.getResource("/sampling-rules.json");
    builder.withSamplingStrategy(new LocalizedSamplingStrategy(ruleFile));

    AWSXRay.setGlobalRecorder(builder.build());
  }
}
```

3. 部署 CloudWatch 代理程式以使用 Amazon 彈性運算雲端 (Amazon EC2)、Amazon Elastic Container Service (Amazon ECS) 或 Amazon Elastic Kubernetes Service (亞馬遜 EKS) 來收集指標：

   - 若要設定 Amazon EC2，請參閱在 Amazon EC2 上部署 CloudWatch 代理程式和 X-Ray 精靈。
   - 若要設定 Amazon ECS，請參閱在 Amazon ECS 上部署 CloudWatch 代理程式和 X-Ray 精靈。
   - 若要設定 Amazon EKS，請參閱在 Amazon EKS 上部署 CloudWatch 代理程式和 X-Ray 精靈。

4. 設定 SDK 以與 CloudWatch 代理程式通訊。依預設，SDK 會與位`127.0.0.1`址上的 CloudWatch 代理程式通訊。您可以將環境變數或 Java 屬性設為 `address:port`，以設定替代位址。

   Example 環境變數

   ```
   AWS_XRAY_METRICS_DAEMON_ADDRESS=address:port
   ```

   Example Java 屬性

   ```
   com.amazonaws.xray.metrics.daemonAddress=address:port
   ```

驗證組態

1. 請登入 AWS Management Console 並開啟 CloudWatch 主控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 開啟 Metrics (指標) 標籤，以觀察指標的湧入。
3. (選擇性) 在 CloudWatch 主控台的 [記錄] 索引標籤上，開啟記`ServiceMetricsSDK`錄群組。尋找符合主機指標的日誌資料流，並確認日誌訊息。

在多執行緒應用程式之間傳遞區段內容

當您在應用程式中建立新執行緒時，`AWSXRayRecorder` 不會維持目前區段或子區段實體的參考。如果您在新執行緒中使用已檢測的用戶端，SDK 會嘗試寫入不存在的區段，導致. SegmentNotFoundException

為了避免在開發過程中拋出異常，您可以使用告ContextMissingStrategy訴它記錄錯誤來配置記錄器。您可以使用程式碼來設定策略 SetContextMissingStrategy，或使用環境變數或系統屬性來設定對等選項。

其中一種處理錯誤的方法是透過在啟動執行緒時呼叫 beginSegment 來使用新區段，以及在關閉時呼叫 endSegment。若您要檢測並非針對回應 HTTP 請求而執行的程式碼，這種方法可以正常運作，就像在應用程式啟動時執行的程式碼。

若您使用多個執行緒來處理傳入請求，您可以將目前區段或子區段傳遞至新執行緒，並將它提供給全域記錄器。這可確保在記錄該請求的其餘資訊時，於新執行緒中記錄的資訊會與相同區段建立關聯。一旦段在新線程中可用，您可以使用該方法執行任何可運行的訪問該段的上下文。`segment.run(() -> { ... })`

如需範例，請參閱在工作者執行緒中使用受檢測用戶端。

使用 X-Ray 與異步編程

適用於 Java 的 X-Ray SDK 可以在非同步 Java 程式中使用 SegmentContextExecutors. SegmentContextExecutor 實現了 Executor 接口，這意味著它可以傳遞到一個 CompletableFuture. 這可確保任何非同步操作都將在其上下文中使用正確的段執行。

Example 例如 App.java：傳遞 SegmentContextExecutor 給 CompletableFuture

```
DynamoDbAsyncClient client = DynamoDbAsyncClient.create();

AWSXRay.beginSegment();

// ...

client.getItem(request).thenComposeAsync(response -> {
    // If we did not provide the segment context executor, this request would not be
 traced correctly.
    return client.getItem(request2);
}, SegmentContextExecutors.newSegmentContextExecutor());
```

AOP 與彈簧和適用於 Java 的 X-Ray SDK

本主題說明如何使用 X-Ray SDK 和 Spring 架構在不變更其核心邏輯的情況下檢測您的應用程式。這意味著現在有一種非侵入性的方法可以檢測在中 AWS遠程運行的應用程序。

啟用 Spring 中的 AOP

1. 設定 Spring
2. 將追蹤篩選器新增至您的應用程式
3. 對您的程式碼做註釋或實作界面
4. 啟用應用程式中的 X-Ray

設定 Spring

您可以使用 Maven 或 Gradle 將 Spring 設定為使用 AOP 檢測您的應用程式。

如果您使用 Maven 來建置應用程式，請在 `pom.xml` 檔案中，新增以下相依性。

```
<dependency>
     <groupId>com.amazonaws</groupId>
     <artifactId>aws-xray-recorder-sdk-spring</artifactId>
     <version>2.11.0</version>
</dependency>
```

若是 Gradle，請在 `build.gradle` 檔案中，新增以下依存性。

```
compile 'com.amazonaws:aws-xray-recorder-sdk-spring:2.11.0'
```

配置春季啟動

除了上一節中描述的 Spring 依賴項之外，如果您使用的是 Spring Boot，請添加以下依賴項（如果它尚未在類路徑中）。

釋界：

```
<dependency>
     <groupId>org.springframework.boot</groupId>
     <artifactId>spring-boot-starter-aop</artifactId>
     <version>2.5.2</version>
</dependency>
```

搖籃：

```
compile 'org.springframework.boot:spring-boot-starter-aop:2.5.2'
```

將追蹤篩選器新增至您的應用程式

添加一個Filter到你的WebConfig班級。以字串形式將區段名稱傳遞給 AWSXRayServletFilter 建構函數。如需有關追蹤篩選器和檢測傳入要求的詳細資訊，請參閱使用適用於 Java 的 X-Ray SDK 追蹤傳入的要求。

Example SRC /主/爪/我的/WebConfig. Java-春天

```
package myapp;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Bean;
import javax.servlet.Filter;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;

@Configuration
public class WebConfig {

  @Bean
  public Filter TracingFilter() {
    return new AWSXRayServletFilter("Scorekeep");
  }
}
```

雅加達 Support

6 年春季使用雅加達而不是 Javax 作為其企業版。為了支持這個新的命名空間，X-Ray 創建了一組存在於自己的雅加達名稱空間中的 parallel 類。

對於過濾器類別，請取javax代為jakarta。設定區段命名策略時，請在命名策略類別名稱jakarta之前新增，如下列範例所示：

```
package myapp;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Bean;
import jakarta.servlet.Filter;
import com.amazonaws.xray.jakarta.servlet.AWSXRayServletFilter;
import com.amazonaws.xray.strategy.jakarta.SegmentNamingStrategy;

@Configuration
public class WebConfig {
    @Bean
    public Filter TracingFilter() {
        return new AWSXRayServletFilter(SegmentNamingStrategy.dynamic("Scorekeep"));
```

```
    }
}
```

對您的程式碼做註釋或實作界面

您的類必須用@XRayEnabled註釋註釋，要么實現接XRayTraced口。這會通知 AOP 系統，針對 X-Ray 檢測包裝受影響的類別函數。

在應用程序中激活 X-Ray

若要在應用程式中啟用 X-Ray 追蹤，您的程式碼必須BaseAbstractXRayInterceptor藉由覆寫下列方法來擴充抽象類別。

- generateMetadata— 此函數允許自訂附加到目前函數追蹤的中繼資料。根據預設，執行函數的類別名稱會記錄到中繼資料中。如果您需要其他資訊，可以新增更多資料。
- xrayEnabledClasses此函數是空的，應該保持如此。可做為 pointcut 的主機，指示攔截程式有哪些包裝方法。指定哪些類別要使用 @XRayEnabled 做註釋以便追蹤，藉此定義 pointcut。以下 pointcut 陳述式會通知攔截程式包裝所有含 @XRayEnabled 註釋的控制器 Bean。

  ```
  @Pointcut(“@within(com.amazonaws.xray.spring.aop.XRayEnabled) && bean(*Controller)”)
  ```

如果您的BaseAbstractXRayInterceptor項目正在使用 Spring 數據 JPA，請考慮從擴展AbstractXRayInterceptor而不是.

範例

下面的代碼擴展了抽象類BaseAbstractXRayInterceptor。

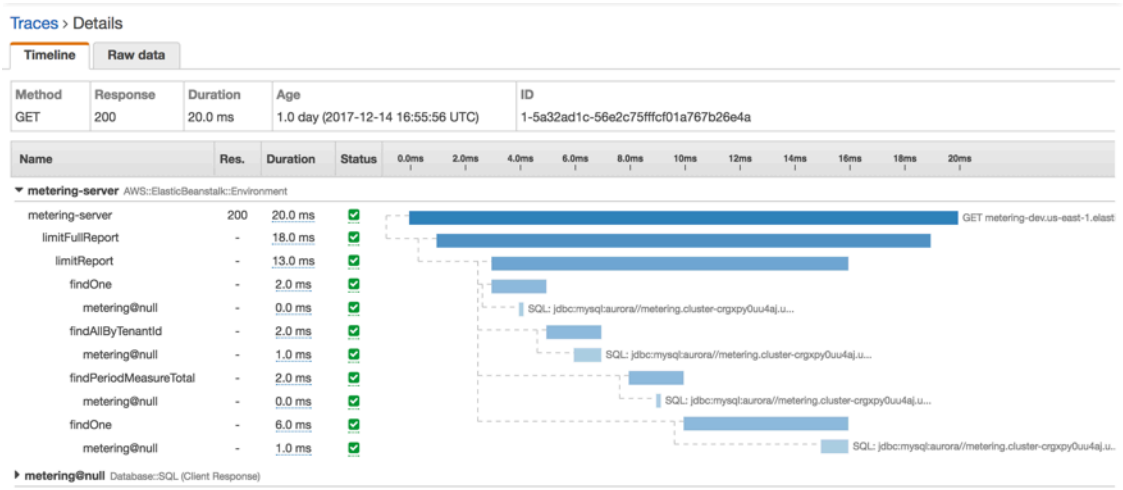
```
@Aspect
@Component
public class XRayInspector extends BaseAbstractXRayInterceptor {
    @Override
    protected Map<String, Map<String, Object>> generateMetadata(ProceedingJoinPoint
 proceedingJoinPoint, Subsegment subsegment) throws Exception {
        return super.generateMetadata(proceedingJoinPoint, subsegment);
    }

  @Override
  @Pointcut("@within(com.amazonaws.xray.spring.aop.XRayEnabled) && bean(*Controller)")
```

```
  public void xrayEnabledClasses() {}

}
```

下列程式碼為將由 X-Ray 檢測的類別。

```
@Service
@XRayEnabled
public class MyServiceImpl implements MyService {
    private final MyEntityRepository myEntityRepository;

    @Autowired
    public MyServiceImpl(MyEntityRepository myEntityRepository) {
        this.myEntityRepository = myEntityRepository;
    }

    @Transactional(readOnly = true)
    public List<MyEntity> getMyEntities(){
        try(Stream<MyEntity> entityStream = this.myEntityRepository.streamAll()){

            return entityStream.sorted().collect(Collectors.toList());
        }
    }
}
```

如果您已正確設定應用程式，則應該會看到應用程式的完整呼叫堆疊 (從控制器到服務呼叫)，如以下主控台的螢幕擷取畫面所示。



檢測您的應用程式 Node.js

有兩種方法可以檢測您的 Node.js 應用程序以將痕跡發送到 X-Ray：

- AWS 發行版 OpenTelemetry JavaScript — 提供一組開放原始碼程式庫的 AWS 發行版，可透過收集器的發行AWS 版，將相關的指標和追蹤傳送至包括 Amazon CloudWatch 和 Amazon OpenSearch 服務在內的多個 AWS 監控解決方案。 AWS X-Ray OpenTelemetry
- AWS X-Ray 適用於 Node.js 的 SDK — 一組程式庫，用於透過 X-Ray 守護程式產生追蹤並將其傳送至 X-Ray。

如需更多詳細資訊，請參閱 在 AWS 發行版 OpenTelemetry 和 X-Ray SDK 之間進行選擇。

AWS發行版OpenTelemetry JavaScript

隨著AWS發行版OpenTelemetry(行為)JavaScript，您可以一次檢測您的應用程序，並將相關的指標和跟踪發送到多個AWS監控解決方案，包括CloudWatch,AWS X-Ray和亞馬遜OpenSearch服務。使用X 射線AWS發行版OpenTelemetry需要兩個組成部分：OpenTelemetrySDK已啟用以與 X 射線搭配使用，以及AWS發行版OpenTelemetry收藏家已啟用以與 X 射線搭配使用。

若要開始使用，請參閱AWS發行版OpenTelemetry JavaScript文件。

> **Note**
>
> ADOTJavaScript支援所有伺服器端 Node.js 應用程式。ADOTJavaScript無法從瀏覽器用戶端將資料匯出至 X-Ray。

有關使用的更多信息AWS發行版OpenTelemetry與AWS X-Ray和其他AWS 服務，請參閱AWS發行版OpenTelemetry或AWS發行版OpenTelemetry文件。

如需語言支援和使用方式的詳細資訊，請參閱AWS上的可觀測性GitHub。

AWS 適用於 Node.js 的 X-Ray SDK

Node.js 適用的 X-Ray SDK 是快速 Web 應用程式和 Node.js Lambda 函數的程式庫，提供產生追蹤資料並將其傳送至 X-Ray 精靈的類別和方法。追蹤資料包括應用程式所提供之傳入 HTTP 要求的相關資訊，以及應用程式使用 AWS SDK 或 HTTP 用戶端對下游服務進行呼叫的相關資訊。

> **Note**
>
> 適用於 Node.js 的 X-Ray SDK 是一個開放原始碼專案。您可以關注該項目並在以下位置提交問題並提取請求 GitHub：gi thub.com/aws/ aws-xray-sdk-node

若您使用 Express，請在您的應用程式伺服器上從將軟體開發套件新增為中介軟體開始，來追蹤傳入請求。中介軟體會為每個追蹤的請求建立區段，並在傳送回應時完成區段。當區段開啟時，您可以使用軟體開發套件用戶端的方法，將資訊新增到區段，並建立子區段以追蹤下游呼叫。軟體開發套件也會在區段為開啟時自動記錄應用程式擲回的例外狀況。

對於經過測試的應用程式或服務呼叫的 Lambda 函數，Lambda 會讀取追蹤標頭並自動追蹤已取樣的請求。對於其他函數，您可以設定 Lambda 來取樣和追蹤傳入的請求。在任何一種情況下，Lambda 都會建立區段，並將其提供給 X-Ray SDK。

> **Note**
>
> 在 Lambda 上，X-Ray 開發套件是選用的。如果您沒有在函數中使用它，您的服務對應仍會包含 Lambda 服務的節點，每個 Lambda 函數有一個節點。透過新增 SDK，您可以檢測函數程式碼，將子區段新增至 Lambda 記錄的函數區段。如需詳細資訊，請參閱AWS Lambda 而且 AWS X-Ray。

接下來，使用適用於 Node.js 的 X-Ray AWS SDK 來檢測 Node.js 用戶端 JavaScript 中的 SDK。每當您使用已檢測的用戶端對下游 AWS 服務 或資源進行呼叫時，SDK 都會在子區段中記錄有關呼叫的資訊。 AWS 服務 而您在服務中存取的資源會顯示為追蹤對映上的下游節點，以協助您識別個別連線上的錯誤和節流問題。

Node.js 的 X-Ray SDK 也為 HTTP 網頁 API 和 SQL 查詢的下游呼叫提供了檢測。將您的 HTTP 用戶端包裝在軟體開發套件的擷取方法中，來記錄傳出 HTTP 呼叫的相關資訊。針對 SQL 用戶端，請使用您資料庫類型的擷取方法。

中介軟體會將抽樣規則套用到傳入請求，判斷要追蹤的請求。您可以設定 Node.js 的 X-Ray SDK，以調整取樣行為或記錄應用程式執行所在之運 AWS 算資源的相關資訊。

使用註釋與中繼資料，記錄應用程式所做的請求和工作等其他資訊。註釋是簡單的鍵/值對，系統會為其建立索引以用於篩選條件表達式，因此您可以搜尋包含特定資料的追蹤。元數據條目的限制較低，可以記錄整個對象和數組-任何可以序列化為 JSON 的內容。

> **標註與中繼資料**
>
> 註釋和中繼資料是您使用 X-Ray SDK 新增至區段的任意文字。註釋會編製索引以與篩選器運算式搭配使用 中繼資料不會建立索引，但可以使用 X-Ray 主控台或 API 在原始區段中檢視。您授與 X-Ray 讀取權限的任何人都可以檢視此資料。

當程式碼中有很多經過檢測的用戶端時，單一請求區段可能包含大量子區段，每個使用經檢測用戶端進行的呼叫都有一個子區段。您可以將用戶端呼叫包裝在自訂子區段中，以組織和群組子區段。您可以為整個函數或任何部分的程式碼建立自訂子區段，並記錄子區段上的中繼資料和註釋，而不必寫入父區段上的所有項目。

如需有關 SDK 類別和方法的參考文件，請參閱 Node.js API 參考的AWS X-Ray SDK。

要求

Node.js 的 X-Ray SDK 需要 Node.js 和下列程式庫：

- `atomic-batcher`— 1.0.2
- `cls-hooked`— 4.2.2
- `pkginfo`— 0.4.0
- `semver`—

軟體開發套件會在您使用 NPM 安裝它時提取這些程式庫。

若要追蹤 AWS SDK 用戶端，Node.js 的 X-Ray SDK 需要 Node.js JavaScript 中最低版本的 AWS SDK。

- `aws-sdk`— 2.7.15

相依性管理

Node.js 的 X-Ray SDK 可從故宮取得。

- Package — `aws-xray-sdk`

針對本機開發，請在您的專案目錄中使用 npm 安裝軟體開發套件。

```
~/nodejs-xray$ npm install aws-xray-sdk
aws-xray-sdk@3.3.3
  ### aws-xray-sdk-core@3.3.3
  # ### @aws-sdk/service-error-classification@3.15.0
  # ### @aws-sdk/types@3.15.0
  # ### @types/cls-hooked@4.3.3
  # # ### @types/node@15.3.0
  # ### atomic-batcher@1.0.2
  # ### cls-hooked@4.2.2
  # # ### async-hook-jl@1.7.6
  # # # ### stack-chain@1.3.7
  # # ### emitter-listener@1.1.2
  # #   ### shimmer@1.2.1
  # ### semver@5.7.1
  ### aws-xray-sdk-express@3.3.3
  ### aws-xray-sdk-mysql@3.3.3
  ### aws-xray-sdk-postgres@3.3.3
```

使用 `--save` 選項來在您應用程式的 `package.json` 中將軟體開發套件做為依存項目儲存。

```
~/nodejs-xray$ npm install aws-xray-sdk --save
aws-xray-sdk@3.3.3
```

如果您的應用程式有任何版本與 X-Ray SDK 相依性衝突的相依性，則會安裝這兩個版本以確保相容性。有關更多詳細信息，請參閱官方 NPM 文檔以了解依賴關係解決

Node.js 樣本

使用適用於 Node.js 的 AWS X-Ray SDK，以在請求通過 Node.js 應用程序時獲取請求的 end-to-end 視圖。

- Node.js 範例應用程式 (位於 GitHub.

設定適用於 Node.js 的 X-Ray SDK

您可以使用外掛程式為 Node.js 設定 X-Ray SDK，以包含應用程式執行之服務的相關資訊、修改預設取樣行為，或新增套用至特定路徑要求的取樣規則。

章節

- 服務外掛程式

- 抽樣規則
- 日誌
- X-Ray 守护进程
- 環境變數

服務外掛程式

用plugins於記錄託管應用程式之服務的相關資訊。

外掛程式

- Amazon EC2 — EC2Plugin 新增執行個體 ID、可用區域和 CloudWatch 日誌群組。
- Elastic Beanstalk — ElasticBeanstalkPlugin 新增環境名稱、版本標籤和部署 ID。
- Amazon ECS-ECSPlugin 添加容器 ID。

若要使用外掛程式，請使用config方法為 Node.js 用戶端設定 X-Ray SDK。

Example app.js - 外掛程式

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.config([AWSXRay.plugins.EC2Plugin,AWSXRay.plugins.ElasticBeanstalkPlugin]);
```

SDK 也會使用外掛程式設定來設定區origin段上的欄位。這表示運行您的應用程序的 AWS 資源的類型。當您使用多個外掛程式時，SDK 會使用下列解析順序來判斷來源： ElasticBeanstalk > EKS > ECS > EC2。

抽樣規則

SDK 會使用您在 X-Ray 主控台中定義的取樣規則來決定要記錄的要求。預設規則會每秒追蹤第一個要求，而所有服務的任何其他要求的百分之五會傳送追蹤至 X-Ray。在 X-Ray 主控台中建立其他規則，以自訂為每個應用程式記錄的資料量。

SDK 會依定義規則的順序套用自訂規則。如果要求符合多個自訂規則，SDK 只會套用第一個規則。

> **Note**
>
> 如果 SDK 無法達到 X-Ray 以取得取樣規則，它會每秒還原為第一個要求的預設本機規則，而每台主機的任何其他要求的百分之五。如果主機沒有調用採樣 API 的權限，或者無法連接到 X-Ray 守護程序（作為 SDK 發出的 API 調用的 TCP 代理），則可能會發生這種情況。

您也可以將 SDK 設定為從 JSON 文件載入取樣規則。SDK 可以使用本機規則做為無法使用 X-Ray 取樣的情況的備份，或僅使用本機規則。

Example 採樣規則

```
{
  "version": 2,
  "rules": [
    {
      "description": "Player moves.",
      "host": "*",
      "http_method": "*",
      "url_path": "/api/move/*",
      "fixed_target": 0,
      "rate": 0.05
    }
  ],
  "default": {
    "fixed_target": 1,
    "rate": 0.1
  }
}
```

此範例定義了一個自訂規則和一個預設規則。自訂規則會套用百分之五的取樣率，而且沒有追蹤下限路徑的要求數目下`/api/move/`限。預設規則會每秒追蹤第一個要求，以及 10% 的額外要求。

在本機定義規則的缺點是，固定目標會由記錄器的每個執行個體獨立套用，而不是由 X-Ray 服務管理。當您部署更多主機時，固定費率會倍增，因此更難以控制記錄的資料量。

開啟時 AWS Lambda，您無法修改取樣率。如果您的函數是由已檢測的服務呼叫，則 Lambda 會記錄產生由該服務取樣之請求的呼叫。如果啟用主動追蹤且沒有追蹤標頭，Lambda 會做出取樣決策。

若要設定備份規則，請告知 Node.js 的 X-Ray SDK 從檔案中載入取樣規則`setSamplingRules`。

Example app.js - 檔案的抽樣規則

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.middleware.setSamplingRules('sampling-rules.json');
```

您也可以在程式碼中定義規則，然後將其做為物件傳遞給 setSamplingRules。

Example app.js - 物件的抽樣規則

```
var AWSXRay = require('aws-xray-sdk');
var rules = {
  "rules": [ { "description": "Player moves.", "service_name": "*", "http_method": "*",
 "url_path": "/api/move/*", "fixed_target": 0, "rate": 0.05 } ],
  "default": { "fixed_target": 1, "rate": 0.1 },
  "version": 1
  }

AWSXRay.middleware.setSamplingRules(rules);
```

若要僅使用本機規則，請呼叫 disableCentralizedSampling。

```
AWSXRay.middleware.disableCentralizedSampling()
```

日誌

若要記錄軟體開發套件的輸出，請呼叫 AWSXRay.setLogger(logger)，其中 logger 是提供標準記錄日誌方法的物件 (warn、info 等)。

默認情況下，SDK 將使用控制台對象上的標準方法將錯誤消息記錄到控制台。內置記錄器的日誌級別可以通過使用AWS_XRAY_DEBUG_MODE或AWS_XRAY_LOG_LEVEL環境變量進行設置。如需有效記錄層級值的清單，請參閱環境變數。

如果您希望為日誌提供不同的格式或目的地，那麼您可以為 SDK 提供自己的記錄器接口實現，如下所示。任何實現此接口的對象都可以使用。這意味著許多日誌庫（例如溫斯頓）可以使用並直接傳遞給 SDK。

Example app.js - 日誌記錄

```
var AWSXRay = require('aws-xray-sdk');

// Create your own logger, or instantiate one using a library.
```

```
var logger = {
  error: (message, meta) => { /* logging code */ },
  warn: (message, meta) => { /* logging code */ },
  info: (message, meta) => { /* logging code */ },
  debug: (message, meta) => { /* logging code */ }
}

AWSXRay.setLogger(logger);
AWSXRay.config([AWSXRay.plugins.EC2Plugin]);
```

請在執行其他組態方法前呼叫 `setLogger`，確保擷取來自這些操作的輸出。

X-Ray 守护进程

如果 X-Ray 精靈在連接埠或主機上接聽`127.0.0.1:2000`，您可以將 Node.js 的 X-Ray SDK 設定為將追蹤資料傳送至不同的位址。

```
AWSXRay.setDaemonAddress('host:port');
```

您可以透過名稱或 IPv4 地址指定主機。

Example app.js - 精靈地址

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.setDaemonAddress('daemonhost:8082');
```

若您設定精靈針對 TCP 和 UDP 接聽不同連接埠，您可以在精靈地址設定中同時指定它們。

Example app.js - 位於不同連接埠的精靈地址

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.setDaemonAddress('tcp:daemonhost:8082 udp:daemonhost:8083');
```

您也可以透過使用 AWS_XRAY_DAEMON_ADDRESS 環境變數來設定精靈地址。

環境變數

您可以使用環境變數來設定 Node.js 的 X-Ray SDK。軟體開發套件支援以下變數。

- AWS_XRAY_CONTEXT_MISSING— 設定為當您RUNTIME_ERROR的檢測程式碼嘗試在沒有區段開啟時記錄資料時擲回例外狀況。

有效值

- RUNTIME_ERROR— 擲回執行階段例外狀況。
- LOG_ERROR— 記錄錯誤並繼續 (預設值)。
- IGNORE_ERROR— 忽略錯誤並繼續。

當您嘗試在沒有要求開啟時執行的啟動程式碼中使用已檢測的用戶端，或在產生新執行緒的程式碼中使用已檢測的用戶端時，可能會發生與遺失區段或子區段相關的錯誤。

- AWS_XRAY_DAEMON_ADDRESS— 設定 X-Ray 精靈監聽程式的主機和連接埠。根據預設，SDK 會同時用127.0.0.1:2000於追蹤資料 (UDP) 和取樣 (TCP)。如果您已將協助程式設定為在不同的連接埠上接聽，或是在不同的主機上執行，請使用此變數。

  格式

  - 相同的連接埠 — *address*:*port*
  - 不同的端口-tcp:*address*:*port* udp:*address*:*port*
- AWS_XRAY_DEBUG_MODE— 設定TRUE為將 SDK 設定為在debug層級將記錄檔輸出至主控台。
- AWS_XRAY_LOG_LEVEL — 設定預設記錄器的記錄層級。有效值為debug、info、warn、error、silent。當 AWS_XRAY_DEBUG_MODE 模式設定為 TRUE 時，會忽略此值。
- AWS_XRAY_TRACING_NAME— 設定 SDK 用於區段的服務名稱。覆寫您在 Express 中介軟體上設定的區段名稱。

使用適用於 Node.js 的 X-Ray SDK 追蹤傳入的要求

您可以使用 Node.js 專用的 X-Ray 開發套件來追蹤快速和重新啟動應用程式在 Amazon EC2 或 Amazon ECS 中的 EC2 執行個體上提供的傳入 HTTP 請求。 AWS Elastic Beanstalk

Node.js 的 X-Ray SDK 為使用快速和重新化框架的應用程序提供了中間件。當您將 X-Ray 中介軟體新增至應用程式時，Node.js 專用的 X-Ray SDK 會為每個取樣要求建立一個區段。此區段包括時間、方法，以及 HTTP 請求的處置方式。其他檢測會在此區段上建立子區段。

> **Note**
>
> 對於 AWS Lambda 函數，Lambda 會為每個取樣的請求建立區段。如需詳細資訊，請參閱AWS Lambda 而且 AWS X-Ray。

每個區段都有一個名稱，可在服務對應中識別您的應用程式。區段可以以靜態方式命名，也可以設定 SDK 根據傳入要求中的主機標頭動態命名區段。動態命名可讓您根據要求中的網域名稱對追蹤進行分組，並在名稱與預期的模式不符時套用預設名稱 (例如，如果主機標頭是偽造的)。

> **轉寄的要求**
>
> 如果負載平衡器或其他中介機構將要求轉寄至您的應用程式，X-Ray 會從要求中的`X-Forwarded-For`標頭取得用戶端 IP，而不是從 IP 封包中的來源 IP 取得。為轉寄要求所記錄的用戶端 IP 可以偽造，因此不應該受信任。

轉送請求時，SDK 會在區段中設定一個額外的欄位來指出這一點。如果區段包含`x_forwarded_for`設定為的欄位`true`，則會從 HTTP 要求中的`X-Forwarded-For`標頭取得用戶端 IP。

訊息處理常式會使用 `http` 區塊為每個傳入的請求建立區段，其中包含以下資訊：

- HTTP 方法-獲取，發布，看跌，刪除等
- [用戶端位址] — 傳送要求的用戶端 IP 位址。
- 回應碼 — 已完成要求的 HTTP 回應碼。
- 時間 — 開始時間 (收到請求的時間) 和結束時間 (傳送回應的時間)。
- 使用者代理程式 — `user-agent` 來自請求的。
- 「內容長度」— 響應`content-length`中的內容。

章節

- 使用 Express 追蹤傳入的請求
- 使用 Restify 追蹤傳入的請求
- 設定區段命名策略

使用 Express 追蹤傳入的請求

若要使用 Express 中介軟體，請初始化軟體開發套件用戶端，並使用油 `express.openSegment` 函數傳回的中介軟體，再定義您的路由。

Example app.js - Express

```
var app = express();
```

```
var AWSXRay = require('aws-xray-sdk');
app.use(AWSXRay.express.openSegment('MyApp'));

app.get('/', function (req, res) {
  res.render('index');
});

app.use(AWSXRay.express.closeSegment());
```

定義路由之後，請使用所示的`express.closeSegment`輸出來處理 Node.js 的 X-Ray SDK 所傳回的任何錯誤。

使用 Restify 追蹤傳入的請求

若要使用 Restify 中介軟體，請初始化軟體開發套件用戶端並執行 `enable`。將它您的 Restify 伺服器和區段名稱傳遞給它。

Example app.js - Restify

```
var AWSXRay = require('aws-xray-sdk');
var AWSXRayRestify = require('aws-xray-sdk-restify');

var restify = require('restify');
var server = restify.createServer();
AWSXRayRestify.enable(server, 'MyApp'));

server.get('/', function (req, res) {
  res.render('index');
});
```

設定區段命名策略

AWS X-Ray 使用服務名稱來識別您的應用程式，並將其與應用程式使用的其他應用程式、資料庫、外部 API 和 AWS 資源區分開來。X-Ray SDK 為傳入要求產生區段時，會在區段的名稱[欄位中記錄應用程式的服務名稱](#)。

X-Ray SDK 可以在 HTTP 要求標頭中的主機名稱之後命名區段。但是，此標頭可能會被偽造，這可能會導致服務對應中出現非預期的節點。若要避免 SDK 因為具有偽造主機標頭的要求而不正確地命名區段，您必須為傳入要求指定預設名稱。

如果您的應用程式為多個網域提供要求，您可以將 SDK 設定為使用動態命名策略，在區段名稱中反映此問題。動態命名策略可讓 SDK 針對符合預期模式的要求使用主機名稱，並將預設名稱套用至不符合要求的要求。

例如，您可能有一個應用程式向三個子網域提供要求 — `www.example.com` `api.example.com`、和`static.example.com`。您可以將動態命名策略與模式搭配使用，`*.example.com`以識別具有不同名稱的每個子網域的區段，從而在服務對應上產生三個服務節點。如果您的應用程式收到的主機名稱與模式不相符的要求，您會在服務對應上看到第四個節點，其中包含您指定的後援名稱。

若要為所有請求區段使用相同的名稱，請如上一節所述，在初始化中介軟體時指定您應用程式的名稱。

> **Note**
>
> 您可以使用 `AWS_XRAY_TRACING_NAME` 環境變數來覆寫您在程式碼中定義的預設服務名稱。

動態命名策略可定義主機名稱應相符的模式，以及如果 HTTP 請求中的主機名稱不符合模式時要使用的預設名稱。若要動態命名區段，請使用 `AWSXRay.middleware.enableDynamicNaming`。

Example app.js - 動態區段名稱

如果請求中的主機名稱符合 `*.example.com` 模式，請使用該主機名稱。否則，請使用 `MyApp`。

```
var app = express();

var AWSXRay = require('aws-xray-sdk');
app.use(AWSXRay.express.openSegment('MyApp'));
AWSXRay.middleware.enableDynamicNaming('*.example.com');

app.get('/', function (req, res) {
  res.render('index');
});

app.use(AWSXRay.express.closeSegment());
```

使用適用於 Node.js 的 X-Ray AWS SDK 追蹤 SDK 呼叫

當您的應用程式呼叫 AWS 服務 以儲存資料、寫入佇列或傳送通知時，Node.js 的 X-Ray SDK 會追蹤子區段中下游的呼叫。追蹤和您在這些服務中存取的資源 (例如，Amazon S3 儲存貯體或 Amazon SQS 佇列) 會在 X-Ray 主控 AWS 服務台的追蹤對應上顯示為下游節點。

您透過 AWS SDK for JavaScript V2 或 AWS SDK for JavaScript V3 建立的儀器 AWS SDK 用戶端。每個 AWS SDK 版本都提供不同的方法來檢測 AWS SDK 用戶端。

> **Note**
>
> 目前，與檢測 V2 用戶端相比，Node.js 的 AWS X-Ray SDK 會在檢測 AWS SDK for JavaScript V3 用戶端時傳回較少的區段資訊。例如，代表呼叫 DynamoDB 的子區段不會傳回資料表名稱。如果您在追蹤中需要此區段資訊，請考慮使用 AWS SDK for JavaScript V2。

AWS SDK for JavaScript V2

您可以通過在呼叫中包裝您的 `aws-sdk` require 語句來檢測所有 AWS SDK V2 客戶端`AWSXRay.captureAWS`。

Example app.js AWS 工具开发工具

```
const AWS = AWSXRay.captureAWS(require('aws-sdk'));
```

若要檢測個別用戶端，請將您的 AWS SDK 用戶端包裝在呼叫中`AWSXRay.captureAWSClient`。例如，若要檢測 AmazonDynamoDB 用戶端：

Example app.js-DynamoDB 端儀器

```
    const AWSXRay = require('aws-xray-sdk');
...
    const ddb = AWSXRay.captureAWSClient(new AWS.DynamoDB());
```

> **Warning**
>
> 請勿將 `captureAWS` 和 `captureAWSClient` 一起使用。這會導致重複的子區段。

如果您想要使TypeScript用 ECMAScript 模組 (ESM) 來載入JavaScript程式碼，請使用下列範例來匯入程式庫：

Example app.js AWS 工具开发工具

```
import * as AWS from 'aws-sdk';
import * as AWSXRay from 'aws-xray-sdk';
```

若要使用 ESM 檢測所有用 AWS 戶端，請使用下列程式碼：

Example app.js AWS 工具开发工具

```
import * as AWS from 'aws-sdk';
import * as AWSXRay from 'aws-xray-sdk';
const XRAY_AWS = AWSXRay.captureAWS(AWS);
const ddb = new XRAY_AWS.DynamoDB();
```

對於所有服務，您都可以在 X-Ray 控制台中看到調用的 API 的名稱。對於服務子集，X-Ray SDK 會將資訊新增至區段，以在服務對應中提供更多精細度。

例如，當您使用已檢測的 DynamoDB 用戶端進行呼叫時，SDK 會將資料表名稱新增至區段，以便針對以資料表為目標的呼叫。在主控台中，每個表格在服務對應中顯示為個別節點，其中包含一般 DynamoDB 節點，用於未針對資料表的呼叫。

Example 呼叫 DynamoDB 以儲存項目的子區段

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

您存取具名資源時，對以下服務的呼叫會在服務地圖中建立額外節點。未針對特定資源的呼叫，則會建立服務的一般節點。

- Amazon DynamoDB — 表名
- Amazon 簡單存儲服務 — 存儲桶和密鑰名稱

- Amazon 簡單隊列服務-隊列名稱

AWS SDK for JavaScript V3

AWS SDK for JavaScript V3 是模塊化的，所以你的代碼只加載它需要的模塊。因此，由於 V3 不支持該`captureAWS`方法，因此無法檢測所有 AWS SDK 客戶端。

如果您想要使 TypeScript 用 ECMAScript 模組 (ESM) 來載入 JavaScript程式碼，您可以使用下列範例來匯入程式庫：

```
import * as AWS from 'aws-sdk';
import * as AWSXRay from 'aws-xray-sdk';
```

使用該`AWSXRay.captureAWSv3Client`方法檢測每個 AWS SDK 客戶端。例如，若要檢測 `AmazonDynamoDB` 用戶端：

Example app.js-DynamoDB SDK 的客戶端儀器

```
    const AWSXRay = require('aws-xray-sdk');
    const { DynamoDBClient } = require("@aws-sdk/client-dynamodb");
...
    const ddb = AWSXRay.captureAWSv3Client(new DynamoDBClient({ region:
 "region" }));
```

使用 AWS SDK for JavaScript V3 時，目前不會傳回資料表名稱、儲存貯體和金鑰名稱或佇列名稱等中繼資料，因此追蹤對應不會像使用 AWS SDK for JavaScript V2 檢測 AWS SDK 用戶端時一樣包含每個已命名資源的離散節點。

Example 使用 V3 時，呼叫 DynamoDB 以儲存項目的子區段 AWS SDK for JavaScript

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
```

```
  },
  "aws": {
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

使用適用於 Node.js 的 X-Ray SDK 追蹤對下游 HTTP 網路服務的呼叫

當您的應用程式呼叫微服務或公用 HTTP API 時，您可以使用 Node.js 用戶端的 X-Ray SDK 來檢測這些呼叫，並將 API 新增至服務圖表做為下游服務。

將您`http`或`https`用戶端傳遞至 Node.js `captureHTTPs` 方法的 X-Ray SDK，以追蹤撥出電話。

> **Note**
>
> 使用協力廠商 HTTP 要求程式庫 (例如 Axios 或 Superagent) 的呼叫會透過 `captureHTTPsGlobal()` API 支援，並在使用原生 `http` 模組時，仍會追蹤這些呼叫。

Example app.js - HTTP 用戶端

```
var AWSXRay = require('aws-xray-sdk');
var http = AWSXRay.captureHTTPs(require('http'));
```

若要啟用所有 HTTP 用戶端上的追蹤，請在載入 `http` 前呼叫 `captureHTTPsGlobal`。

Example app.js - HTTP 用戶端 (全域)

```
var AWSXRay = require('aws-xray-sdk');
AWSXRay.captureHTTPsGlobal(require('http'));
var http = require('http');
```

當您檢測對下游 Web API 的呼叫時，Node.js 的 X-Ray SDK 會記錄一個子區段，其中包含 HTTP 要求和回應的相關資訊。X-Ray 會使用子區段來產生遠端 API 的推斷區段。

Example 下游 HTTP 呼叫的子區段

```
{
  "id": "004f72be19cddc2a",
  "start_time": 1484786387.131,
```

```
  "end_time": 1484786387.501,
  "name": "names.example.com",
  "namespace": "remote",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  }
}
```

Example 下游 HTTP 呼叫的推斷區段

```
{
  "id": "168416dc2ea97781",
  "name": "names.example.com",
  "trace_id": "1-62be1272-1b71c4274f39f122afa64eab",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "parent_id": "004f72be19cddc2a",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  },
  "inferred": true
}
```

使用適用於 Node.js 的 X-Ray 開發套件追蹤 SQL 查詢

檢測 SQL 資料庫查詢，方法是將 SQL 用戶端封裝在 Node.js 用戶端方法的對應 X-Ray SDK 中。

- PostgreSQL – `AWSXRay.capturePostgres()`

```
var AWSXRay = require('aws-xray-sdk');
var pg = AWSXRay.capturePostgres(require('pg'));
var client = new pg.Client();
```

- MySQL – `AWSXRay.captureMySQL()`

```
var AWSXRay = require('aws-xray-sdk');
var mysql = AWSXRay.captureMySQL(require('mysql'));
...
var connection = mysql.createConnection(config);
```

當您使用受檢測用戶端進行 SQL 查詢時，適用於 Node.js 的 X-Ray 開發套件會在子區段中記錄連線及查詢的相關資訊。

在 SQL 子區段中包含其他資料

您可以將其他資訊新增至針對 SQL 查詢產生的子區段，只要該子區段已對應至允許列出的 SQL 欄位即可。例如，若要在子區段中記錄已淨化的 SQL 查詢字串，您可以將其直接新增至子區段的 SQL 物件。

Example 將 SQL 指派給子區段

```
    const queryString = 'SELECT * FROM MyTable';
connection.query(queryString, ...);

// Retrieve the most recently created subsegment
const subs = AWSXRay.getSegment().subsegments;

if (subs & & subs.length > 0) {
  var sqlSub = subs[subs.length - 1];
  sqlSub.sql.sanitized_query = queryString;
}
```

如需允許列出 SQL 欄位的完整清單，請參閱AWS X-Ray 開發人員指南中的 SQL 查詢。

使用適用於 Node.js 的 X-Ray SDK 產生自訂子區段

子段擴展了跟踪的段，其中包含有關完成工作的詳細信息，以滿足請求。每次您使用已檢測的用戶端進行呼叫時，X-Ray SDK 都會記錄在子區段中產生的資訊。您可以建立其他子區段來分組其他子區段、測量程式碼區段的效能，或是記錄註釋和中繼資料。

自訂快速子區段

若要為呼叫下游服務的函數建立自訂子區段，請使用 `captureAsyncFunc` 函數。

Example app.js - 快速自訂子區段

```
var AWSXRay = require('aws-xray-sdk');

app.use(AWSXRay.express.openSegment('MyApp'));

app.get('/', function (req, res) {
  var host = 'api.example.com';

  AWSXRay.captureAsyncFunc('send', function(subsegment) {
    sendRequest(host, function() {
      console.log('rendering!');
      res.render('index');
      subsegment.close();
    });
  });
});

app.use(AWSXRay.express.closeSegment());

function sendRequest(host, cb) {
  var options = {
    host: host,
    path: '/',
  };

  var callback = function(response) {
    var str = '';

    response.on('data', function (chunk) {
      str += chunk;
    });
```

```
    response.on('end', function () {
      cb();
    });
  }

  http.request(options, callback).end();
};
```

在此範例中，應用程式會為針對 sendRequest 函數的呼叫建立名為 send 的自訂子區段。captureAsyncFunc 會在其發出的非同步呼叫完成時傳遞您必須在回撥函數中關閉的子區段。

針對同步函數，您可以使用 captureFunc 函數，自動在函數區塊完成執行後關閉子區段。

當您在區段或其他子區段內建立子區段時，Node.js 的 X-Ray SDK 會產生該區段的 ID，並記錄開始時間和結束時間。

Example 使用中繼資料的子區段

```
"subsegments": [{
  "id": "6f1605cd8a07cb70",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "Custom subsegment for UserModel.saveUser function",
  "metadata": {
    "debug": {
      "test": "Metadata string from UserModel.saveUser"
    }
  },
```

自訂 Lambda 子區段

SDK 設定為在 Lambda 中偵測到預留位置外觀區段時，會自動建立預留位置外觀區段。要創建一個基本的子部分，這將在 X-Ray 跟踪映射上創建一個AWS::Lambda::Function節點，調用並重新利用外觀段。如果您使用新 ID 手動建立新區段 (在共用追蹤 ID、父系 ID 和抽樣決策時)，您將能夠傳送新區段。

Example app.js - 手動自訂子區段

```
const segment = AWSXRay.getSegment(); //returns the facade segment
const subsegment = segment.addNewSubsegment('subseg');
...
subsegment.close();
```

```
//the segment is closed by the SDK automatically
```

使用適用於 Node.js 的 X-Ray SDK 將註釋和中繼資料新增至區段

您可以使用註釋和中繼資料來記錄有關請求、環境或應用程式的其他資訊。您可以將註釋和中繼資料新增至 X-Ray SDK 建立的區段，或新增至您建立的自訂子區段。

註釋是與字符串，數字或布爾值鍵-值對。註釋會編製索引以與篩選器運算式搭配使用 使用標記記錄您想要用來在主控台將追蹤分組的資料，或是在呼叫 `GetTraceSummaries` API 時使用標記。

中繼資料是索引鍵-值配對，可以具有任何類型的值 (包括物件和清單)，但不會編製索引以供篩選運算式使用。使用元數據記錄要存儲在跟踪中但不需要與搜索一起使用的其他數據。

除了註釋和中繼資料，您也可以在區段上記錄使用者 ID 字串。區段會將使用者 ID 記錄在單獨的欄位中，並建立索引以用於搜尋。

章節

- 使用適用於 Node.js 的 X-Ray SDK 錄製註釋
- 使用適用於 Node.js 的 X-Ray SDK 記錄中繼資料
- 使用適用於 Node.js 的 X-Ray SDK 記錄使用者 ID

使用適用於 Node.js 的 X-Ray SDK 錄製註釋

針對您想要建立索引以用於搜尋的區段或子區段，請使用標註來記錄這些區段上的資訊。

註釋要求

- 按鍵 — X-Ray 註解的金鑰最多可包含 500 個英數字元。您不能使用底線符號 (_) 以外的空格或符號。
- 值 — X-Ray 註釋的值最多可包含 1,000 個 Unicode 字元。
- 註釋的數量 — 每個追蹤最多可以使用 50 個註釋。

記錄標註

1. 取得目前區段或子區段的參考。

```
var AWSXRay = require('aws-xray-sdk');
...
var document = AWSXRay.getSegment();
```

2. 使用字串索引鍵、布林值、數字或字串值，呼叫 addAnnotation。

```
document.addAnnotation("mykey", "my value");
```

軟體開發套件會將標註以鍵/值對記錄在區段文件中的 annotations 物件內。若使用相同鍵呼叫 addAnnotation 兩次，則會覆寫之前在相同區段或子區段上記錄的值。

若要尋找具有特定值之註釋的繪線，請在篩選器運算式中使用annotations.*key*關鍵字。

Example app.js - 標註

```
var AWS = require('aws-sdk');
var AWSXRay = require('aws-xray-sdk');
var ddb = AWSXRay.captureAWSClient(new AWS.DynamoDB());
...
app.post('/signup', function(req, res) {
    var item = {
        'email': {'S': req.body.email},
        'name': {'S': req.body.name},
        'preview': {'S': req.body.previewAccess},
        'theme': {'S': req.body.theme}
    };

    var seg = AWSXRay.getSegment();
    seg.addAnnotation('theme', req.body.theme);

    ddb.putItem({
      'TableName': ddbTable,
      'Item': item,
      'Expected': { email: { Exists: false } }
  }, function(err, data) {
...
```

使用適用於 Node.js 的 X-Ray SDK 記錄中繼資料

針對您不想要建立索引以用於搜尋的區段，請使用中繼資料來記錄這些區段或子區段上的資訊。中繼資料值可以是字串、數字、布林值，或可序列化為 JSON 物件或陣列的任何其他物件。

記錄中繼資料

1. 取得目前區段或子區段的參考。

```
var AWSXRay = require('aws-xray-sdk');
...
var document = AWSXRay.getSegment();
```

2. 使用字串鍵、布林值、數字、字串或物件值，以及字串命名空間，呼叫 addMetadata。

```
document.addMetadata("my key", "my value", "my namespace");
```

或

只使用鍵和值呼叫 addMetadata。

```
document.addMetadata("my key", "my value");
```

若您沒有指定命名空間，軟體開發套件會使用 default。若使用相同鍵呼叫 addMetadata 兩次，則會覆寫之前在相同區段或子區段上記錄的值。

使用適用於 Node.js 的 X-Ray SDK 記錄使用者 ID

記錄請求區段上的使用者 ID 以識別傳送請求的使用者。此作業與 AWS Lambda 函數不相容，因為 Lambda 環境中的區段是不可變的。setUser 呼叫只可套用至區段，而非子區段。

記錄使用者 ID

1. 取得目前區段或子區段的參考。

```
var AWSXRay = require('aws-xray-sdk');
...
var document = AWSXRay.getSegment();
```

2. 使用傳送請求之使用者的字串 ID 呼叫 setUser()。

```
var user = 'john123';

AWSXRay.getSegment().setUser(user);
```

您可以呼叫 setUser，以在 Express 應用程式開始處理請求時馬上記錄使用者 ID。如果您只要使用區段來設定使用者 ID，可以將呼叫鏈結為單行。

Example app.js - 使用者 ID

```
var AWS = require('aws-sdk');
var AWSXRay = require('aws-xray-sdk');
var uuidv4 = require('uuid/v4');
var ddb = AWSXRay.captureAWSClient(new AWS.DynamoDB());
...
    app.post('/signup', function(req, res) {
    var userId = uuidv4();
    var item = {
        'userId': {'S': userId},
        'email': {'S': req.body.email},
        'name': {'S': req.body.name}
    };

    var seg = AWSXRay.getSegment().setUser(userId);

    ddb.putItem({
      'TableName': ddbTable,
      'Item': item,
      'Expected': { email: { Exists: false } }
  }, function(err, data) {
...
```

若要尋找使用者 ID 的追蹤，請在篩選運算式中使用`user`關鍵字。

檢測您的應用程式 Python

有兩種方法可以檢測您的Python應用程序以將痕跡發送到 X-Ray：

- AWS 發行版 OpenTelemetry Python — 提供一組開放原始碼程式庫的 AWS 發行版，可透過收集器的發行AWS 版，將相關的指標和追蹤傳送至包括 Amazon CloudWatch 和 Amazon OpenSearch 服務在內的多個 AWS 監控解決方案。 AWS X-Ray OpenTelemetry
- AWS X-Ray SDK 用於 Python — 一組程式庫，用於透過 X-Ray 守護程式產生追蹤並將其傳送至 X-Ray。

如需更多詳細資訊，請參閱 在 AWS 發行版 OpenTelemetry 和 X-Ray SDK 之間進行選擇。

AWS 發行版的 OpenTelemetry Python

使用適用於 OpenTelemetry (ADOT) 的發行 AWS 版Python，您可以對應用程式進行一次檢測，並將相關的指標和追蹤傳送到包括 Amazon 和 Amazon CloudWatch 服務在內的多個 AWS 監控解決方案。AWS X-Ray OpenSearch 將 X-Ray 與 ADOT 搭配使用需要兩個元件：啟用可與 X-Ray 搭配使用的 OpenTelemetry SDK，以及啟用可與 X-Ray 搭配使用的 OpenTelemetry收集器發行AWS 版。ADOT Python 包含自動檢測支援，可讓您的應用程式在不變更程式碼的情況下傳送追蹤。

要開始使用，請參閱AWS 發行版以獲取 OpenTelemetry Python文檔。

OpenTelemetry 有關使用發行 AWS 版 AWS X-Ray 和其他的更多信息 AWS 服務，請參閱AWS 發行版 OpenTelemetry或發行AWS 文檔。 OpenTelemetry

如需有關語言支援和使用方式的詳細資訊，請參閱上 GitHub的AWS 可觀測性。

AWS X-Ray 適用於的 SDK Python

Python 的 X-Ray SDK 是一個適用於 Python Web 應用程式的程式庫，提供產生追蹤資料並將其傳送至 X-Ray 精靈的類別和方法。追蹤資料包括應用程式所提供之傳入 HTTP 要求的相關資訊，以及應用程式使用 AWS SDK、HTTP 用戶端或 SQL 資料庫連接器對下游服務進行呼叫的相關資訊。您也可以手動建立區段，並將除錯資訊新增至註釋和中繼資料中。

您可以使用 `pip` 下載軟體開發套件。

```
$ pip install aws-xray-sdk
```

> **Note**
>
> 適用於 Python 的 X-Ray SDK 是一個開源項目。您可以關注該項目並在以下位置提交問題並提取請求 GitHub：gi thub.com/aws/ aws-xray-sdk-python

若您使用 Django 或 Flask，請從將軟體開發套件中介軟體新增到您的應用程式開始，來追蹤傳入請求。中介軟體會為每個追蹤的請求建立區段，並在傳送回應時完成區段。當區段開啟時，您可以使用軟體開發套件用戶端的方法，將資訊新增到區段，並建立子區段以追蹤下游呼叫。軟體開發套件也會在區段為開啟時自動記錄應用程式擲回的例外狀況。針對其他應用程式，您可以手動建立區段。

對於經過測試的應用程式或服務呼叫的 Lambda 函數，Lambda 會讀取追蹤標頭並自動追蹤已取樣的請求。對於其他函數，您可以設定 Lambda 來取樣和追蹤傳入的請求。在任何一種情況下，Lambda 都會建立區段，並將其提供給 X-Ray SDK。

> **Note**
>
> 在 Lambda 上，X-Ray 開發套件是選用的。如果您沒有在函數中使用它，您的服務對應仍會包含 Lambda 服務的節點，每個 Lambda 函數有一個節點。透過新增 SDK，您可以檢測函數程式碼，將子區段新增至 Lambda 記錄的函數區段。如需詳細資訊，請參閱AWS Lambda 而且 AWS X-Ray。

如工作程序需 Lambda 中檢測的Python函數範例，請參閱。

接下來，使用適用於 Python 的 X-Ray SDK，透過修補應用程式的程式庫來測量下游呼叫。軟體開發套件支援以下程式庫。

支援的程式庫

- `botocore`, `boto3` — 儀器 AWS SDK for Python (Boto) 客戶.
- `pynamodb`— 儀器版本的亞馬遜動態 B 用戶端。
- `aiobotocore`, `aioboto3` — 用於 Python 客戶端的工具非同步集成版本的 SDK。
- `requests`, `aiohttp` — 儀器高級 HTTP 客戶端。
- `httplib`, `http.client`— 儀器低級 HTTP 客戶端以及使用它們的更高級別庫。
- `sqlite3`— 儀器客戶端。
- `mysql-connector-python`— 儀器 MySQL 客戶端.
- `pg8000`— 儀器純 Python 接口。
- `psycopg2`— 儀器 PostgreSQL 配接器。
- `pymongo`— 儀器 MongoDB 的客戶端.
- `pymysql`— 儀器基於 PyMy SQL 的客戶端為 MySQL 和 MariaDB.

每當您的應用程式對 SQL 資料庫或其他 HTTP 服務進行呼叫時，SDK 會在子區段中記錄有關呼叫的資訊。 AWS AWS 服務 而您在服務中存取的資源會顯示為追蹤對映上的下游節點，以協助您識別個別連線上的錯誤和節流問題。

開始使用 SDK 之後，請透過設定記錄器和中介軟體來自訂其行為。您可以新增外掛程式，以記錄執行應用程式所需的運算資源相關資料、定義抽樣規則以自訂抽樣行為，並設定日誌層級以在應用程式日誌中查看更多或更少的軟體開發套件資訊。

使用註釋與中繼資料，記錄應用程式所做的請求和工作等其他資訊。註釋是簡單的鍵/值對，系統會為其建立索引以用於篩選條件表達式，因此您可以搜尋包含特定資料的追蹤。元數據條目的限制較低，可以記錄整個對象和數組-任何可以序列化為 JSON 的內容。

> **標註與中繼資料**
>
> 註釋和中繼資料是您使用 X-Ray SDK 新增至區段的任意文字。註釋會編製索引以與篩選器運算式搭配使用 中繼資料不會建立索引，但可以使用 X-Ray 主控台或 API 在原始區段中檢視。您授與 X-Ray 讀取權限的任何人都可以檢視此資料。

當程式碼中有很多經過檢測的用戶端時，單一請求區段可能包含大量子區段，每個使用經檢測用戶端進行的呼叫都有一個子區段。您可以將用戶端呼叫包裝在自訂子區段中，以組織和群組子區段。您可以為整個函數或一部分的程式碼建立自訂子區段。您接著可以在子區段上記錄中繼資料及標註，而非將所有項目寫入父區段。

如需 SDK 類別和方法的參考文件，請參閱 AWS X-Ray SDK 以取得 Python API 參考。

要求

適用於 Python 的 X-Ray SDK 支援以下語言和程式庫版本。

- Python — 2.7, 3.4, 和更高版本
- 賈戈-1.10 及更高版本
- 瓶-0.10 和更高版本
- 愛奧赫特普 — 2.3.0 和更高版本
- AWS SDK for Python (Boto)— 1.4.0 及更新版本
- 肉毒核心 — 1.5.0 及更高版本
- 枚舉-0.4.7 及更高版本，適用於Python版本 3.4.0 及更高版本
- 傑森泡菜 — 1.0.0 及更高版本
- 設置工具 — 40. 6.3 和更高版本
- 包裹 — 1. 11.0 和更新版本

相依性管理

適用於 Python 的 X-Ray SDK 可從中獲得`pip`。

- Package — `aws-xray-sdk`

在您的 `requirements.txt` 檔案中將軟體開發套件新增為依存項目。

Example requirements.txt

```
aws-xray-sdk==2.4.2
boto3==1.4.4
botocore==1.5.55
Django==1.11.3
```

如果您使用 Elastic Beanstalk 部署您的應用程式，Elastic Beanstalk 會自動安裝所有套件。`requirements.txt`

配置 X-Ray SDK

Python 的 X-Ray SDK 有一個名`xray_recorder`為提供全局記錄器的類。您可以設定全域記錄器來自訂為傳入 HTTP 呼叫建立區段的中介軟體。

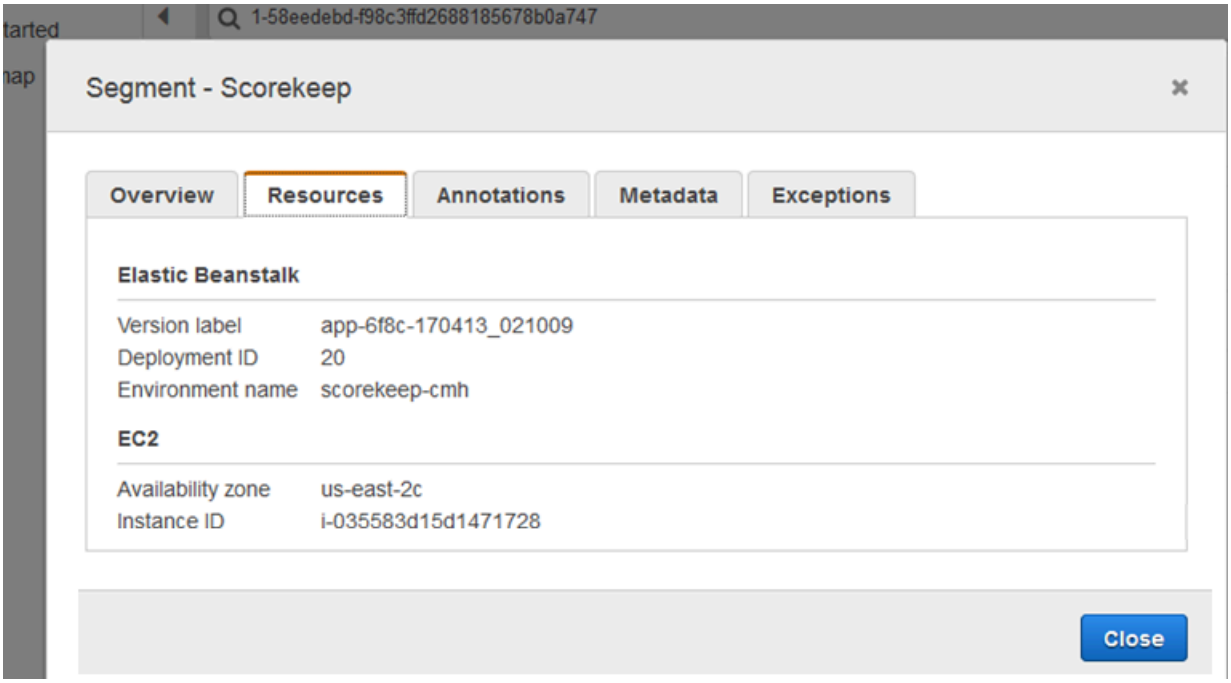
章節

服務外掛程式

用`plugins`於記錄託管應用程式之服務的相關資訊。

外掛程式

- Amazon EC2 —`EC2Plugin` 新增執行個體 ID、可用區域和 CloudWatch 日誌群組。
- Elastic Beanstalk —`ElasticBeanstalkPlugin` 新增環境名稱、版本標籤和部署 ID。
- 亞馬遜 ECS-`ECSPlugin` 添加容器 ID。



若要使用外掛程式，請在 `xray_recorder` 上呼叫 `configure`。

```
from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch_all

xray_recorder.configure(service='My app')
plugins = ('ElasticBeanstalkPlugin', 'EC2Plugin')
xray_recorder.configure(plugins=plugins)
patch_all()
```

> **Note**
>
> 由plugins於作為元組傳入，因此在指定單個插件,時請務必包含尾隨。例如：`plugins = ('EC2Plugin',)`

您也可以使用優先於程式碼中設定值的環境變數，來設定記錄器。

在修補程式庫前設定外掛程式來記錄下游呼叫。

SDK 也會使用外掛程式設定來設定區origin段上的欄位。這表示運行您的應用程序的AWS資源的類型。當您使用多個外掛程式時，SDK 會使用下列解析順序來判斷來源： ElasticBeanstalk > EKS > ECS > EC2。

抽樣規則

SDK 會使用您在 X-Ray 主控台中定義的取樣規則來決定要記錄的要求。預設規則會每秒追蹤第一個要求，而所有服務的任何其他要求的百分之五會傳送追蹤至 X-Ray。在 X-Ray 主控台中建立其他規則，以自訂為每個應用程式記錄的資料量。

SDK 會依定義順序套用自訂規則。如果要求符合多個自訂規則，SDK 只會套用第一個規則。

> **Note**
>
> 如果 SDK 無法達到 X-Ray 以取得取樣規則，它會每秒還原為第一個要求的預設本機規則，而每台主機的任何其他要求的百分之五。如果主機沒有調用採樣 API 的權限，或者無法連接到 X-Ray 守護程序（作為 SDK 發出的 API 調用的 TCP 代理），則可能會發生這種情況。

您也可以將 SDK 設定為從 JSON 文件載入取樣規則。SDK 可以使用本機規則做為無法使用 X-Ray 取樣的情況的備份，或僅使用本機規則。

Example 採樣規則

```
{
  "version": 2,
  "rules": [
    {
      "description": "Player moves.",
      "host": "*",
      "http_method": "*",
      "url_path": "/api/move/*",
      "fixed_target": 0,
      "rate": 0.05
    }
  ],
  "default": {
    "fixed_target": 1,
    "rate": 0.1
  }
}
```

此範例定義了一個自訂規則和一個預設規則。自訂規則會套用百分之五的取樣率，而且沒有追蹤下限路徑的要求數目下限`/api/move/`。預設取樣規則每秒 1 次請求和 10% 的額外請求。

在本機定義規則的缺點是，固定目標會由記錄器的每個執行個體獨立套用，而不是由 X-Ray 服務管理。當您部署更多主機時，固定費率會倍增，因此更難以控制記錄的資料量。

開啟時AWS Lambda，您無法修改取樣率。如果您的函數是由已檢測的服務呼叫，則 Lambda 會記錄產生由該服務取樣之請求的呼叫。如果啟用主動追蹤且沒有追蹤標頭，Lambda 會做出取樣決策。

若要設定備份抽樣規則，請呼叫 `xray_recorder.configure` (如以下範例所示)，其中 *`rules`* 為規則的字典或指向包含抽樣規則 JSON 檔案的絕對路徑。

```
xray_recorder.configure(sampling_rules=rules)
```

若僅要使用本機規則，請使用 `LocalSampler` 設定記錄器。

```
from aws_xray_sdk.core.sampling.local.sampler import LocalSampler
xray_recorder.configure(sampler=LocalSampler())
```

您也可以設定全域記錄器來停用所有傳入請求的抽樣和檢測。

Example main.py — 停用取樣

```
xray_recorder.configure(sampling=False)
```

日誌

SDK 使用具有預設WARNING記錄層級的 Python 內建`logging`模組。為 `aws_xray_sdk` 類別取得記錄器的參考，然後在其上呼叫 `setLevel` 來為程式庫及您其餘的應用程式設定不同日誌層級。

Example app.py — 日誌記錄

```
logging.basicConfig(level='WARNING')
logging.getLogger('aws_xray_sdk').setLevel(logging.ERROR)
```

當您手動產生子區段時，可使用除錯日誌來識別問題，例如未結束的子區段。

程式碼中的記錄器組態

可以從 `xray_recorder` 上的 `configure` 方法取得其他可用設定。

- `context_missing`— 設定為以`LOG_ERROR`避免在檢測過的程式碼嘗試在沒有區段開啟時記錄資料時擲回例外狀況。

- daemon_address— 設定 X-Ray 精靈監聽程式的主機和連接埠。
- service— 設定 SDK 用於區段的服務名稱。
- plugins— 記錄有關應用程序AWS資源的信息。
- sampling— 設定False為停用取樣。
- sampling_rules— 設定包含取樣規則的 JSON 檔案路徑。

Example main.py-禁用上下文缺少異常

```
from aws_xray_sdk.core import xray_recorder

xray_recorder.configure(context_missing='LOG_ERROR')
```

使用 Django 的記錄器組態

若您使用 Django 框架，您可以使用 Django settings.py 檔案來在全域記錄器上設定選項。

- AUTO_INSTRUMENT(僅限 Django) — 記錄內建資料庫和範本顯示作業的子區段。
- AWS_XRAY_CONTEXT_MISSING— 設定為以LOG_ERROR避免在檢測過的程式碼嘗試在沒有區段開啟時記錄資料時擲回例外狀況。
- AWS_XRAY_DAEMON_ADDRESS— 設定 X-Ray 精靈監聽程式的主機和連接埠。
- AWS_XRAY_TRACING_NAME— 設定 SDK 用於區段的服務名稱。
- PLUGINS— 記錄有關應用程序AWS資源的信息。
- SAMPLING— 設定False為停用取樣。
- SAMPLING_RULES— 設定包含取樣規則的 JSON 檔案路徑。

若要在 settings.py 中啟用記錄器組態，請將 Django 中介軟體新增至已安裝的應用程式清單。

Example settings.py — 已安裝應用程式

```
INSTALLED_APPS = [
    ...
    'django.contrib.sessions',
    'aws_xray_sdk.ext.django',
]
```

在名為 XRAY_RECORDER 的字典中設定可用設定。

Example settings.py — 已安裝應用程式

```
XRAY_RECORDER = {
    'AUTO_INSTRUMENT': True,
    'AWS_XRAY_CONTEXT_MISSING': 'LOG_ERROR',
    'AWS_XRAY_DAEMON_ADDRESS': '127.0.0.1:5000',
    'AWS_XRAY_TRACING_NAME': 'My application',
    'PLUGINS': ('ElasticBeanstalkPlugin', 'EC2Plugin', 'ECSPlugin'),
    'SAMPLING': False,
}
```

環境變數

您可以使用環境變數，配置 X-Ray SDK。軟體開發套件支援以下變數：

- AWS_XRAY_TRACING_NAME— 設定 SDK 用於區段的服務名稱。覆寫您以程式設計方式設定的服務名稱。
- AWS_XRAY_SDK_ENABLED— 設定為時false，會停用 SDK。軟體開發套件預設為啟用，除非環境變數設定為 false。
  - 停用時，全域記錄器會自動產生虛擬、不會傳送給協助程式的區段和子區段，且自動修補會停用。中介軟體是編寫做為透過全域記錄器的包裝函式。透過中介軟體產生的所有區段和子區段，也會成為虛擬區段和虛擬子區段。
  - 透過環境變數，或是透過與 aws_xray_sdk 程式庫內 global_sdk_config 物件的直接互動，設定 AWS_XRAY_SDK_ENABLED 的值。環境變數設定會覆寫這些互動。
- AWS_XRAY_DAEMON_ADDRESS— 設定 X-Ray 精靈監聽程式的主機和連接埠。根據預設，SDK 會同時用127.0.0.1:2000於追蹤資料 (UDP) 和取樣 (TCP)。如果您已將協助程式設定為在不同的連接埠上接聽，或是在不同的主機上執行，請使用此變數。

  格式

  - 相同的連接埠 —*address*:*port*
  - 不同的端口-tcp:*address*:*port* udp:*address*:*port*
- AWS_XRAY_CONTEXT_MISSING— 設定RUNTIME_ERROR為當您的檢測程式碼嘗試在沒有區段開啟時記錄資料時擲回例外狀況。

  有效值

  - RUNTIME_ERROR— 擲回執行階段例外狀況。
  - LOG_ERROR— 記錄錯誤並繼續 (預設值)。

- `IGNORE_ERROR`— 忽略錯誤並繼續。

  當您嘗試在沒有要求開啟時執行的啟動程式碼中使用已檢測的用戶端，或在產生新執行緒的程式碼中使用已檢測的用戶端時，可能會發生與遺失區段或子區段相關的錯誤。

環境變數會覆寫程式碼中所設的值。

使用適用於 Python 中間件的 X-Ray 開發套件之傳入請求

當您將中介軟體新增至應用程式並設定區段名稱時，Python 的 X-Ray SDK 會為每個取樣要求建立區段。此區段包括時間、方法，以及 HTTP 請求的處置方式。其他檢測會在此區段上建立子區段。

適用於 Python 的 X-Ray SDK 支援下列中間件來檢測傳入的 HTTP 請求：

- Django
- Flask
- Bottle

> **Note**
>
> 對於AWS Lambda函數，Lambda 會為每個取樣的請求建立區段。如需詳細資訊，請參閱 AWS Lambda 而且 AWS X-Ray。

如工作程序需在 Lambda 中檢測到的 Python 函數範例，請參閱。

針對指令碼或其他框架上的 Python 應用程式，您可以手動建立區段。

每個區段都有一個名稱，可在服務對應中識別您的應用程式。區段可以以靜態方式命名，也可以設定 SDK 根據傳入要求中的主機標頭動態命名該區段。動態命名可讓您根據要求中的網域名稱對追蹤進行分組，並在名稱與預期的模式不符時套用預設名稱 (例如，如果主機標頭是偽造的)。

> **轉寄的要求**
>
> 如果負載平衡器或其他中介機構將要求轉寄至您的應用程式，X-Ray 會從要求中的`X-Forwarded-For`標頭取得用戶端 IP，而不是從 IP 封包中的來源 IP 取得。為轉寄的要求所記錄的用戶端 IP 可以偽造，因此不應該受信任。

轉送請求時，SDK 會在區段中設定一個額外的欄位來指出這一點。如果區段包含`x_forwarded_for`設定為的欄位`true`，則會從 HTTP 要求中的`X-Forwarded-For`標頭取得用戶端 IP。

中介軟體會使用 `http` 區塊為每個傳入的請求建立區段，其中包含以下資訊：

- HTTP 方法-GET、POST、PUT、DELETE 等
- [用戶端地址]-傳送請求的用戶端 IP 地址。
- 回應碼 — 已完成要求的 HTTP 回應碼。
- 時間 — 開始時間 (收到請求的時間) 和結束時間 (傳送回應的時間)。
- 使用者代理程式 —`user-agent` 來自請求的。
- 「內容長度」— 響應`content-length`中的內容。

章節

- 將中介軟體新增至應用程式 (Django)
- 將中介軟體新增至應用程式 (Flask)
- 將中介軟體新增至應用程式 (Bottle)
- 手動檢測 Python 程式碼
- 設定區段命名策略

將中介軟體新增至應用程式 (Django)

將中介軟體新增至您 `settings.py` 檔案中的 `MIDDLEWARE` 清單。X-Ray 中介軟體應為您 `settings.py` 檔案中的第一行，確保會記錄在其他中介軟體中失敗的請求。

Example settings.py-適用於蟒蛇中間件的 X-Ray SDK

```
MIDDLEWARE = [
    'aws_xray_sdk.ext.django.middleware.XRayMiddleware',
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware'
]
```

將 X-Ray SDK Django 應用程序添加到`settings.py`文件中的`INSTALLED_APPS`列表中。這將允許在應用程序啟動期間配置 X-Ray 記錄器。

Example settings.py-適用於蟒蛇應用程序的 X-Ray SDK

```
INSTALLED_APPS = [
    'aws_xray_sdk.ext.django',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

在您的 settings.py 檔案中設定區段名稱。

Example settings.py — 區段名稱

```
XRAY_RECORDER = {
    'AWS_XRAY_TRACING_NAME': 'My application',
    'PLUGINS': ('EC2Plugin',),
}
```

這告訴 X-Ray 記錄器以默認採樣率跟踪 Django 應用程序提供的請求。您可以在您的 Django 設定檔中設定記錄器來套用自訂抽樣規則或變更其他設定。

> **Note**
>
> 由`plugins`於作為元組傳入，因此在指定單個插件,時請務必包含尾隨。例如：`plugins = ('EC2Plugin',)`

將中介軟體新增至應用程式 (Flask)

若要檢測您的 Flask 應用程式，請先在 `xray_recorder` 上設定區段名稱。然後，使用 `XRayMiddleware` 函數來在程式碼中修補您的 Flask 應用程式。

Example app.py

```
from aws_xray_sdk.core import xray_recorder
```

```
from aws_xray_sdk.ext.flask.middleware import XRayMiddleware

app = Flask(__name__)

xray_recorder.configure(service='My application')
XRayMiddleware(app, xray_recorder)
```

這告訴 X-Ray 記錄器以默認採樣率跟踪 Flask 應用程序提供的請求。您可以在程式碼中設定記錄器來套用自訂抽樣規則或變更其他設定。

將中介軟體新增至應用程式 (Bottle)

若要檢測您的 Bottle 應用程式，請先在 `xray_recorder` 上設定區段名稱。然後，使用 `XRayMiddleware` 函數來在程式碼中修補您的 Bottle 應用程式。

Example app.py

```
from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.ext.bottle.middleware import XRayMiddleware

app = Bottle()

xray_recorder.configure(service='fallback_name', dynamic_naming='My application')
app.install(XRayMiddleware(xray_recorder))
```

這告訴 X-Ray 記錄器以默認採樣率跟踪 Bottle 應用程序提供的請求。您可以在程式碼中設定記錄器來套用自訂抽樣規則或變更其他設定。

手動檢測 Python 程式碼

若您並未使用 Django 或 Flask，您可以手動建立區段。您可以為每個傳入要求建立區段，或在已修補的 HTTP 或AWS SDK 用戶端周圍建立區段，以提供錄製程式新增子區段的內容。

Example main.py — 手動儀表

```
from aws_xray_sdk.core import xray_recorder

# Start a segment
segment = xray_recorder.begin_segment('segment_name')
# Start a subsegment
subsegment = xray_recorder.begin_subsegment('subsegment_name')
```

```
# Add metadata and annotations
segment.put_metadata('key', dict, 'namespace')
subsegment.put_annotation('key', 'value')

# Close the subsegment and segment
xray_recorder.end_subsegment()
xray_recorder.end_segment()
```

設定區段命名策略

AWS X-Ray使用服務名稱來識別您的應用程式，並將其與應用程式使用的其他應用程式、資料庫、外部 API 和AWS資源區分開來。X-Ray SDK 為傳入要求產生區段時，會在區段的名稱欄位中記錄應用程式的服務名稱。

X-Ray SDK 可以在 HTTP 要求標頭中的主機名稱之後命名區段。但是，此標頭可能會被偽造，這可能會導致服務對應中出現非預期的節點。若要避免 SDK 因為具有偽造主機標頭的要求而不正確地命名區段，您必須為傳入要求指定預設名稱。

如果您的應用程式為多個網域提供要求，您可以將 SDK 設定為使用動態命名策略，在區段名稱中反映此問題。動態命名策略可讓 SDK 針對符合預期模式的要求使用主機名稱，並將預設名稱套用至不符合要求的要求。

例如，您可能有一個應用程式向三個子網域提供要求 —www.example.comapi.example.com、和static.example.com。您可以將動態命名策略與模式搭配使用，*.example.com以識別具有不同名稱的每個子網域的區段，從而在服務對應上產生三個服務節點。如果您的應用程式收到的主機名稱與模式不相符的要求，您會在服務對應上看到第四個節點，其中包含您指定的後援名稱。

若要為所有請求區段使用相同的名稱，請如上一節所述，在設定記錄器時指定您應用程式的名稱。

動態命名策略可定義主機名稱應相符的模式，以及如果 HTTP 請求中的主機名稱不符合模式時要使用的預設名稱。若要在 Django 中動態命名區段，請將 DYNAMIC_NAMING 設定新增到您的 settings.py 檔案。

Example settings.py-動態命名

```
XRAY_RECORDER = {
    'AUTO_INSTRUMENT': True,
    'AWS_XRAY_TRACING_NAME': 'My application',
    'DYNAMIC_NAMING': '*.example.com',
    'PLUGINS': ('ElasticBeanstalkPlugin', 'EC2Plugin')
}
```

您可以在模式中使用 '*' 來比對任何字串，或是 '?' 來比對任何單一字元。針對 Flask，請在程式碼中設定記錄器。

Example main.py — 區段名稱

```
from aws_xray_sdk.core import xray_recorder
xray_recorder.configure(service='My application')
xray_recorder.configure(dynamic_naming='*.example.com')
```

> **Note**
>
> 您可以使用 AWS_XRAY_TRACING_NAME 環境變數來覆寫您在程式碼中定義的預設服務名稱。

修補程式庫來檢測下游呼叫

若要測量下游呼叫，請使用 Python 專用的 X-Ray SDK 來修補應用程式所使用的程式庫。適用於 Python 的 X 射線 SDK 可以修補以下庫。

支援的程式庫

- botocore,boto3— 儀器AWS SDK for Python (Boto)客戶端。
- pynamodb— 儀器版本的亞馬遜動態 B 用戶端。
- aiobotocore,aioboto3— 儀器異步-適用於 Python 客戶端的 SDK 的集成版本。
- requests,aiohttp— 儀器高級 HTTP 客戶端。
- httplib,http.client— 儀器低級 HTTP 客戶端和使用它們的更高級別庫。
- sqlite3— 儀器客戶端。
- mysql-connector-python— 儀器 MySQL 客戶端.
- pg8000— 儀器純蟒接口。
- psycopg2— 儀器資料庫配接器。
- pymongo— 儀器蒙古數據庫的客戶端.
- pymysql— 儀器PyMy基於 SQL 的客戶端，適用於 MySQL 和瑪麗亞德。

當您使用已修補的程式庫時，Python 的 X-Ray SDK 會建立呼叫的子區段，並記錄來自要求和回應的資訊。區段必須透過軟體開發套件中介軟體或 AWS Lambda 供軟體開發套件使用，以建立子區段。

> **Note**
>
> 若您使用 SQLAlchemy ORM，您可以透過匯入 SQLAlchemy 工作階段和查詢類別的軟體開發套件版本，來檢測您的 SQL 查詢。如需說明，請參閱使用 SQLAlchemy ORM。

若要修補所有可用程式庫，請使用 `aws_xray_sdk.core` 中的 `patch_all` 函數。有些程式庫 (例如 `httplib` 和 `urllib`) 可能需要呼叫 `patch_all(double_patch=True)` 以啟用雙重修補。

Example main.py — 修補所有支援的程式庫

```
import boto3
import botocore
import requests
import sqlite3

from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch_all

patch_all()
```

若要修補單一程式庫，請以該程式庫名稱的元組來呼叫 `patch`。若要執行這個作業，您將需要提供單一個元素清單。

Example main.py — 修補程式特定程式庫

```
import boto3
import botocore
import requests
import mysql-connector-python

from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch

libraries = ['botocore']
patch(libraries)
```

> **Note**
>
> 在某些情況下，您用來修補程式庫的鍵會與程式庫名稱不相符。有些鍵會做為一或多個程式庫的別名。

程式庫別名

- httplib—httplib和http.client
- mysql – mysql-connector-python

追蹤非同步工作的內容

對於asyncio整合式程式庫，或至為非同步函數建立子區段時，您還必須使用異步上下文為 Python 配置 X 射線 SDK。匯入AsyncContext類並將其實例傳遞給 X 射線記錄器。

> **Note**
>
> Web 架構支援程式庫，例如 AIOHTTP，將不會經由 aws_xray_sdk.core.patcher 模組獲得處理。它們不會出現在支援程式庫的 patcher 目錄中。

Example main.py — 安全修補程式 3

```
import asyncio
import aioboto3
import requests

from aws_xray_sdk.core.async_context import AsyncContext
from aws_xray_sdk.core import xray_recorder
xray_recorder.configure(service='my_service', context=AsyncContext())
from aws_xray_sdk.core import patch

libraries = (['aioboto3'])
patch(libraries)
```

使用適用於 Python 的 X-Ray AWS SDK 追蹤 SDK 呼叫

當您的應用程式呼叫 AWS 服務 以儲存資料、寫入佇列或傳送通知時，Python 的 X-Ray SDK 會追蹤子區段中下游的呼叫。您在這些服務中存取的追蹤 AWS 服務 和資源 (例如，Amazon S3 儲存貯體或 Amazon SQS 佇列) 會在 X-Ray 主控台的追蹤對應上顯示為下游節點。

當您修補程式botocore庫時，適用於 Python 的 X-Ray AWS SDK 會自動檢測所有 SDK 用戶端。您無法檢測個別用戶端。

對於所有服務，您可以在 X-Ray 控制台中看到調用的 API 的名稱。對於服務子集，X-Ray SDK 會將資訊新增至區段，以在服務對應中提供更多精細度。

例如，當您使用已檢測的 DynamoDB 用戶端進行呼叫時，SDK 會將資料表名稱新增至區段，以便針對以資料表為目標的呼叫。在主控台中，每個表格在服務對應中顯示為獨立節點，其中包含一般 DynamoDB 節點，用於未針對資料表的呼叫。

Example 呼叫 DynamoDB 以儲存項目的子區段

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

您存取具名資源時，對以下服務的呼叫會在服務地圖中建立額外節點。未針對特定資源的呼叫，則會建立服務的一般節點。

- Amazon DynamoDB — 表名稱
- Amazon 簡單存儲服務 — 存儲桶和密鑰名稱
- Amazon 簡單隊列服務-隊列名稱

使用適用於 Python 的 X-Ray 開發套件追蹤對下游 HTTP Web 服務進行的呼叫

當您的應用程式呼叫微服務或公有 HTTP API 時，您可以使用適用於 Python 的 X-Ray 開發套件來檢測這些呼叫，並將 API 做為下游服務新增到服務圖表。

若要檢測 HTTP 用戶端，請修補您用來進行傳出呼叫的程式庫。若您使用 `requests` 或 Python 內建的 HTTP 用戶端，只需進行這些作業。針對 `aiohttp`，請同時使用非同步內容設定記錄器。

若您使用 `aiohttp` 3 的用戶端 API，您也需要使用軟體開發套件提供的追蹤組態執行個體設定 `ClientSession` 的部分。

Example **`aiohttp`** 3 用戶端 API

```
from aws_xray_sdk.ext.aiohttp.client import aws_xray_trace_config

async def foo():
    trace_config = aws_xray_trace_config()
    async with ClientSession(loop=loop, trace_configs=[trace_config]) as session:
        async with session.get(url) as resp
            await resp.read()
```

當您檢測對下游 Web API 進行的呼叫時，適用於 Python 的 X-Ray 開發套件會記錄包含 HTTP 請求及回應相關資訊的子區段。X-Ray 會使用子區段來產生遠端 API 的推斷區段。

Example 下游 HTTP 呼叫的子區段

```
{
  "id": "004f72be19cddc2a",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "name": "names.example.com",
  "namespace": "remote",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  }
}
```

Example 下游 HTTP 呼叫的推斷區段

```
{
```

```
  "id": "168416dc2ea97781",
  "name": "names.example.com",
  "trace_id": "1-62be1272-1b71c4274f39f122afa64eab",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "parent_id": "004f72be19cddc2a",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  },
  "inferred": true
}
```

使用適用於 Python 的 X-Ray 開發套件

子段擴展追蹤的段，詳細介紹了為了服務請求而完成的工作。每次與分析客户端進行呼叫時，X-Ray SDK 都會記錄在子段中生成的信息。您可以創建其他子段來對其他子段進行分組、測量代碼部分的性能或記錄註釋和元數據。

若要管理子區段，請使用 `begin_subsegment` 和 `end_subsegment` 方法。

Example main.py — 自訂子區段

```
from aws_xray_sdk.core import xray_recorder

subsegment = xray_recorder.begin_subsegment('annotations')
subsegment.put_annotation('id', 12345)
xray_recorder.end_subsegment()
```

若要建立同步函數的子區段，請使用 `@xray_recorder.capture` 裝飾項目。您可以將子區段的名稱傳遞給擷取函數，或是不傳遞它來使用函數名稱。

Example main.py — 函數子區段

```
from aws_xray_sdk.core import xray_recorder
```

```
@xray_recorder.capture('## create_user')
def create_user():
...
```

針對非同步函數，請使用 @xray_recorder.capture_async 裝飾項目，並將非同步內容傳遞給記錄器。

Example main.py 非同步函數子區段

```
from aws_xray_sdk.core.async_context import AsyncContext
from aws_xray_sdk.core import xray_recorder
xray_recorder.configure(service='my_service', context=AsyncContext())

@xray_recorder.capture_async('## create_user')
async def create_user():
    ...

async def main():
    await myfunc()
```

當您在某區段或其他子區段內建立子區段時，適用於 Python 的 X-Ray 開發套件會為其產生一個 ID，並記錄開始時間和結束時間。

Example 使用中繼資料的子區段

```
"subsegments": [{
  "id": "6f1605cd8a07cb70",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "Custom subsegment for UserModel.saveUser function",
  "metadata": {
    "debug": {
      "test": "Metadata string from UserModel.saveUser"
    }
  },
```

使用適用於 Python 的 X-Ray SDK 將註釋和中繼資料新增至區段

您可以使用註釋和中繼資料來記錄有關請求、環境或應用程式的其他資訊。您可以將註釋和中繼資料新增至 X-Ray SDK 建立的區段，或新增至您建立的自訂子區段。

註釋是與字符串，數字或布爾值鍵-值對。註釋會編製索引以與篩選器運算式搭配使用 使用標記記錄您想要用來在主控台將追蹤分組的資料，或是在呼叫 `GetTraceSummaries` API 時使用標記。

中繼資料是索引鍵-值配對，可以具有任何類型的值 (包括物件和清單)，但不會編製索引以供篩選運算式使用。使用元數據記錄要存儲在跟踪中但不需要與搜索一起使用的其他數據。

除了註釋和中繼資料，您也可以在區段上記錄使用者 ID 字串。區段會將使用者 ID 記錄在單獨的欄位中，並建立索引以用於搜尋。

章節

- 使用適用於 Python 的 X-Ray SDK 記錄註釋
- 使用適用於 Python 的 X-Ray SDK 記錄元數據
- 使用 X-Ray SDK 記錄用戶 ID

使用適用於 Python 的 X-Ray SDK 記錄註釋

針對您想要建立索引以用於搜尋的區段或子區段，請使用標註來記錄這些區段上的資訊。

註釋要求

- 按鍵 — X-Ray 註解的金鑰最多可包含 500 個英數字元。您不能使用底線符號 (_) 以外的空格或符號。
- 值 — X-Ray 註釋的值最多可包含 1,000 個 Unicode 字元。
- 註釋的數量 — 每個追蹤最多可以使用 50 個註釋。

記錄標註

1. 從 `xray_recorder` 取得目前區段或子區段的參考。

```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_segment()
```

或

```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_subsegment()
```

2. 使用字串索引鍵、布林值、數字或字串值，呼叫 put_annotation。

```
document.put_annotation("mykey", "my value");
```

或者，您可以使用 xray_recorder 上的 put_annotation 方法。此方法會在目前的子區段或區段 (若沒有任何開啟的子區段) 上記錄標註。

```
xray_recorder.put_annotation("mykey", "my value");
```

軟體開發套件會將標註以鍵/值對記錄在區段文件中的 annotations 物件內。若使用相同鍵呼叫 put_annotation 兩次，則會覆寫之前在相同區段或子區段上記錄的值。

若要尋找具有特定值之註釋的繪線，請在篩選器運算式中使用annotations.*key*關鍵字。

使用適用於 Python 的 X-Ray SDK 記錄元數據

針對您不想要建立索引以用於搜尋的區段，請使用中繼資料來記錄這些區段或子區段上的資訊。中繼資料值可以是字串、數字、布林值，或可序列化為 JSON 物件或陣列的任何物件。

記錄中繼資料

1. 從 xray_recorder 取得目前區段或子區段的參考。

```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_segment()
```

或

```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_subsegment()
```

2. 使用字串鍵、布林值、數字、字串或物件值，以及字串命名空間，呼叫 put_metadata。

```
document.put_metadata("my key", "my value", "my namespace");
```

或

只使用鍵和值呼叫 put_metadata。

```
document.put_metadata("my key", "my value");
```

或者，您可以使用 xray_recorder 上的 put_metadata 方法。此方法會在目前的子區段或區段 (若沒有任何開啟的子區段) 上記錄中繼資料。

```
xray_recorder.put_metadata("my key", "my value");
```

若您沒有指定命名空間，軟體開發套件會使用 default。若使用相同鍵呼叫 put_metadata 兩次，則會覆寫之前在相同區段或子區段上記錄的值。

使用 X-Ray SDK 記錄用戶 ID

記錄請求區段上的使用者 ID 以識別傳送請求的使用者。

記錄使用者 ID

1. 從 xray_recorder 取得目前區段的參考。

```
from aws_xray_sdk.core import xray_recorder
...
document = xray_recorder.current_segment()
```

2. 使用傳送請求之使用者的字串 ID 呼叫 setUser。

```
document.set_user("U12345");
```

您可以在控制器中呼叫 set_user，以在應用程式開始處理請求時馬上記錄使用者 ID。

若要尋找使用者 ID 的追蹤，請在篩選運算式中使用user關鍵字。

檢測部署在無伺服器環境中的 Web 框架

適用於 Python 的 AWS X-Ray SDK 支援檢測無伺服器應用程式中部署的 Web 架構。無伺服器解決方案是雲端原生架構，能讓您將更多的操作責任轉移到 AWS，提高您的敏捷度和創新能力。

無伺服器架構是一種軟體應用程式模型，可讓您建置和執行應用程式和服務，而完全不用考慮伺服器。這種做法可免除基礎設施管理工作，例如伺服器或叢集佈建、修補、作業系統維護和容量佈建。您可以為幾乎任何應用程式類型或後端服務建立無伺服器解決方案，並為您包辦執行和擴展高可用性應用程式所需的一切工作。

本教程將向您展示如何 AWS X-Ray 在部署到無服務器環境中的 Web 框架（例如 Flask 或 Django）上自動進行檢測。應用程式的 X-Ray 檢測可讓您檢視從 Amazon API Gateway 開始透過您的 AWS Lambda 功能進行的所有下游呼叫，以及應用程式進行的撥出呼叫。

適用於 Python 的 X-Ray 開發套件支援下列 Python 應用程式架構：

- Flask 版本 0.8 或更新版本
- Django 版本 1.0 或更新版本

本教學課程開發一個範例無伺服器應用程式，該應用程式部署至 Lambda 並由 API Gateway 叫用。本教學課程使用 Zappa 將應用程式自動部署到 Lambda，並設定 API Gateway 端點。

必要條件

- Zappa
- Python — 版本 2.7 或 3.6.
- AWS CLI— 驗證您的帳戶 AWS CLI 已配置，並且您將 AWS 區域 在其中部署應用程序。
- Pip
- Virtualenv

步驟 1：建立環境

在此步驟中，您將建立使用 `virtualenv` 來主控應用程式的虛擬環境。

1. 使用 AWS CLI，建立應用程式的目錄。然後，切換至新目錄。

```
mkdir serverless_application
cd serverless_application
```

2. 接下來，在您的新目錄中建立虛擬環境。使用下列命令，將其啟動。

```
# Create our virtual environment
virtualenv serverless_env

# Activate it
source serverless_env/bin/activate
```

3. 安裝 X-Ray, 燒瓶, Zappa, 和請求庫到您的環境.

```
# Install X-Ray, Flask, Zappa, and Requests into your environment
pip install aws-xray-sdk flask zappa requests
```

4. 將應用程式程式碼新增到 serverless_application 目錄。在這個範例中，我們可以建置出 Flasks 的 Hello World 範例。

   在 serverless_application 目錄中，建立名為 my_app.py 的檔案。然後，使用文字編輯器來新增下列命令。這個應用程式會檢測請求程式庫、修補 Flask 應用程式的中介軟體，並開啟端點 '/'。

```
# Import the X-Ray modules
from aws_xray_sdk.ext.flask.middleware import XRayMiddleware
from aws_xray_sdk.core import patcher, xray_recorder
from flask import Flask
import requests

# Patch the requests module to enable automatic instrumentation
patcher.patch(('requests',))

app = Flask(__name__)

# Configure the X-Ray recorder to generate segments with our service name
xray_recorder.configure(service='My First Serverless App')

# Instrument the Flask application
XRayMiddleware(app, xray_recorder)

@app.route('/')
def hello_world():
    resp = requests.get("https://aws.amazon.com")
    return 'Hello, World: %s' % resp.url
```

步驟 2：建立和部署 Zappa 環境

在此步驟中，您將使用 Zappa 自動設定 API Gateway 端點，然後部署到 Lambda。

1. 從 serverless_application 目錄當中起始 Zappa。在這個範例中，我們使用的是預設設定，但如果您有自訂偏好設定，則 Zappa 會顯示組態指示。

```
zappa init
```

```
What do you want to call this environment (default 'dev'): dev
...
What do you want to call your bucket? (default 'zappa-*******'): zappa-*******
...
...
It looks like this is a Flask application.
What's the modular path to your app's function?
This will likely be something like 'your_module.app'.
We discovered: my_app.app
Where is your app's function? (default 'my_app.app'): my_app.app
...
Would you like to deploy this application globally? (default 'n') [y/n/
(p)rimary]: n
```

2. 啟用 X-Ray。開啟 zappa_settings.json 檔案，並確認其類似此範例。

```
{
    "dev": {
        "app_function": "my_app.app",
        "aws_region": "us-west-2",
        "profile_name": "default",
        "project_name": "serverless-exam",
        "runtime": "python2.7",
        "s3_bucket": "zappa-*********"
    }
}
```

3. 將 "xray_tracing": true 新增做為組態檔的項目：

```
{
    "dev": {
        "app_function": "my_app.app",
        "aws_region": "us-west-2",
        "profile_name": "default",
        "project_name": "serverless-exam",
        "runtime": "python2.7",
        "s3_bucket": "zappa-*********",
        "xray_tracing": true
    }
}
```

4. 部署應用程式。這會自動設定 API Gateway 端點，並將您的程式碼上傳至 Lambda。

```
zappa deploy
```

```
...
Deploying API Gateway..
Deployment complete!: https://**********.execute-api.us-west-2.amazonaws.com/dev
```

步驟 3：啟用 API Gateway 的 X-Ray 追蹤

在此步驟中，您將與 API Gateway 主控台互動，以啟用 X-Ray 追蹤。

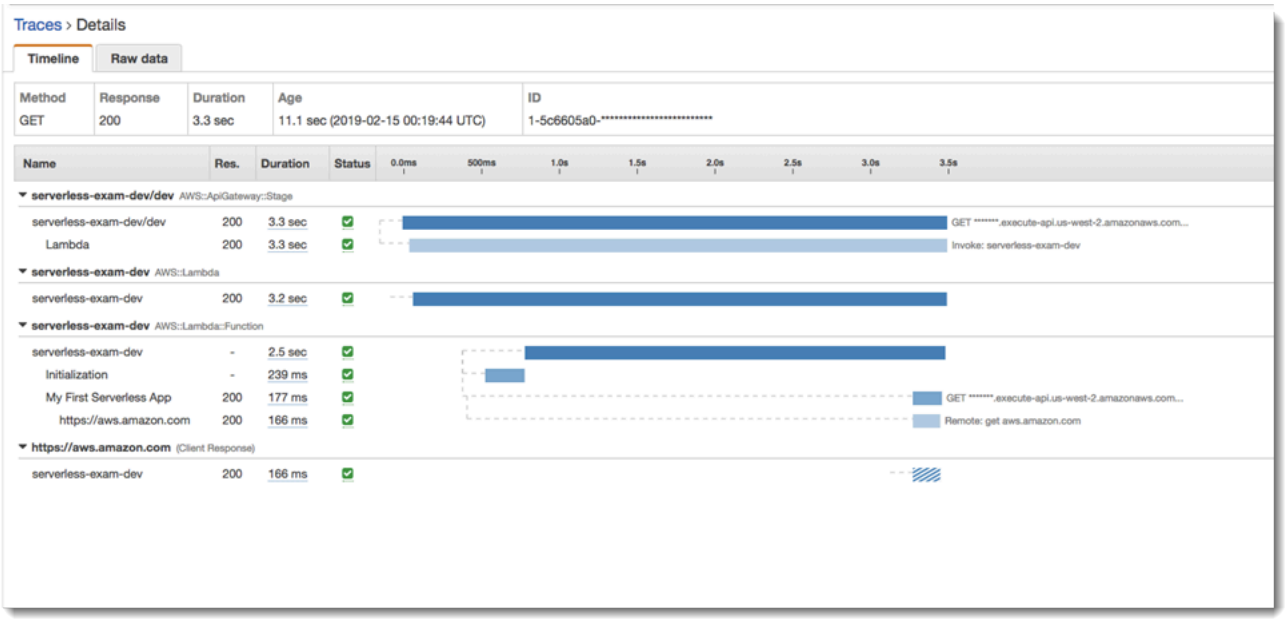
1. 登入 AWS Management Console 並開啟 API Gateway 主控台，網址為 https://console.aws.amazon.com/apigateway/。
2. 尋找新產生的 API。該項目看起來應該會類似 `serverless-exam-dev`。
3. 選擇 Stages (階段)。
4. 選擇您的部署階段名稱。預設值為 `dev`。
5. 在 Logs/Tracing (日誌/追蹤) 標籤中，選取 Enable X-Ray Tracing (啟用 X-Ray 追蹤) 方塊。
6. 選擇 Save Changes (儲存變更)。
7. 在您的瀏覽器中存取端點。如果您使用的是 `Hello World` 應用程式範例，這時應該顯示如下。

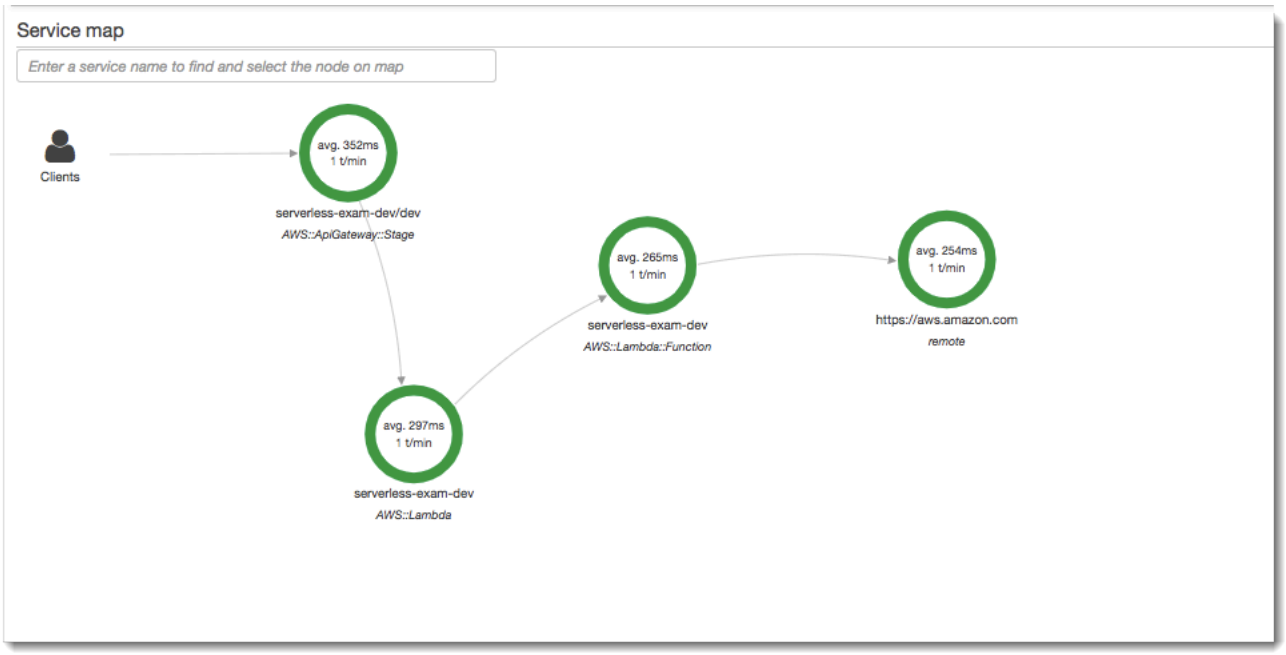
```
"Hello, World: https://aws.amazon.com/"
```

步驟 4：檢視已建立的追蹤

在此步驟中，您將與 X-Ray 控制台進行交互，以查看示例應用程序創建的跟踪。如需追蹤分析的詳細演練，請參閱檢視服務映射。

1. 登入 AWS Management Console 並開啟 X-Ray 主控台，網址為 https://console.aws.amazon.com/xray/home。
2. 檢視 API Gateway、Lambda 函數和 Lambda 容器所產生的區段。
3. 在 Lambda 函數區段下，檢視名為`My First Serverless App`的子區段。它的後面是一個名為 `https://aws.amazon.com` 的第二個子區段。
4. 在初始化期間，Lambda 也可能會產生名為`initialization`的第三個子區段。





步驟 5：清除

務必終止您不再使用的資源，避免累積超出預期的費用。在此教學示範中，像是 Zappa 等工具會簡化無伺服器解決方案的重新部署。

若要從 Lambda、API Gateway 和 Amazon S3 移除您的應用程式，請在專案目錄中使用 AWS CLI.

```
zappa undeploy dev
```

後續步驟

通過添加 AWS 客戶端並使用 X-Ray 對其進行檢測，為應用程序添加更多功能。在上進一步了解無伺服器運算選項。 AWS

檢測您的應用程式 .NET

有兩種方法可以檢測您的.NET應用程序以將痕跡發送到 X-Ray：

- AWS 發行版 OpenTelemetry .NET — 提供一組開放原始碼程式庫的 AWS 發行版 CloudWatch，AWS X-Ray可透過收集器的發行AWS 版，將相關的指標和追蹤傳送至包括 Amazon 和 Amazon OpenSearch 服務在內的多個 AWS 監控解決方案。 OpenTelemetry
- AWS X-Ray SDK 用於 .NET — 一組程式庫，用於透過 X-Ray 守護程式產生追蹤並將其傳送至 X-Ray。

如需更多詳細資訊，請參閱 在 AWS 發行版 OpenTelemetry 和 X-Ray SDK 之間進行選擇。

AWS 發行版的 OpenTelemetry .NET

使用 AWS Distro OpenTelemetry .NET，您可以對應用程序進行一次檢測，並將相關的指標和跟踪發送到包括 Amazon 和 Amazon CloudWatch 服務在內的多個 AWS 監控解決方案。 AWS X-Ray OpenSearch 使用 X-Ray 與 AWS 發行版 OpenTelemetry 需要兩個組件：啟用與 X-Ray 一起使用的 OpenTelemetry SDK，以及啟用與 X-Ray 一起使用的發行AWS 版。 OpenTelemetry

要開始使用，請參閱AWS 發行版以獲取 OpenTelemetry .NET文檔。

OpenTelemetry 有關使用發行 AWS 版 AWS X-Ray 和其他的更多信息 AWS 服務，請參閱AWS 發行版 OpenTelemetry或發行AWS 文檔。 OpenTelemetry

如需有關語言支援和使用方式的詳細資訊，請參閱上 GitHub的AWS 可觀測性。

AWS X-Ray SDK for .NET

適用於 .NET 的 X-Ray SDK 是用於檢測 C# .NET Web 應用程式、.NET 核心 Web 應用程式和 .NET 核心功能的程式庫。 AWS Lambda它提供了用於生成和發送跟踪數據到 X-Ray 守護進程的類和方法。這包括應用程式所提供之傳入要求的相關資訊，以及應用程式對下游 AWS 服務、HTTP Web API 和 SQL 資料庫進行呼叫的相關資訊。

> **Note**
>
> 適用於 .NET 的 X-Ray SDK 是一個開放原始碼專案。您可以關注該項目並在以下位置提交問題並提取請求 GitHub：gi thub.com/aws/ aws-xray-sdk-dotnet

針對 Web 應用程式，請先將訊息處理常式新增至 Web 組態以追蹤傳入的請求。訊息處理常式會為每個追蹤的請求建立區段，並在傳送回應時完成區段。當區段開啟時，您可以使用軟體開發套件用戶端的方法，將資訊新增到區段，並建立子區段以追蹤下游呼叫。軟體開發套件也會在區段為開啟時自動記錄應用程式擲回的例外狀況。

對於經過測試的應用程式或服務呼叫的 Lambda 函數，Lambda 會讀取追蹤標頭並自動追蹤已取樣的請求。對於其他函數，您可以設定 Lambda 來取樣和追蹤傳入的請求。在任何一種情況下，Lambda 都會建立區段，並將其提供給 X-Ray SDK。

> **Note**
>
> 在 Lambda 上，X-Ray 開發套件是選用的。如果您沒有在函數中使用它，您的服務對應仍會包含 Lambda 服務的節點，每個 Lambda 函數有一個節點。透過新增 SDK，您可以檢測函數程式碼，將子區段新增至 Lambda 記錄的函數區段。如需更多資訊，請參閱AWS Lambda 而且 AWS X-Ray。

接下來，使用適用於 .NET 的 X-Ray SDK 來檢測您的 AWS SDK for .NET 客戶端。每當您使用已檢測的用戶端對下游 AWS 服務 或資源進行呼叫時，SDK 都會在子區段中記錄有關呼叫的資訊。 AWS 服務和您在服務中存取的資源會顯示為追蹤對映上的下游節點，以協助您識別個別連線上的錯誤和節流問題。

適用於 .NET 的 X-Ray SDK 也提供了對 HTTP Web API 和 SQL 資料庫的下游呼叫的檢測。適用於 `System.Net.HttpWebRequest` 的 `GetResponseTraced` 延伸方法可追蹤傳出的 HTTP 呼叫。您可以使用 .NET 版本的 X-Ray SDK `SqlCommand` 來檢測 SQL 查詢。

開始使用 SDK 之後，請設定記錄器和訊息處理常式來自訂其行為。您可以新增外掛程式，以記錄執行應用程式所需的運算資源相關資料、定義抽樣規則以自訂抽樣行為，並設定日誌層級以在應用程式日誌中查看更多或更少的軟體開發套件資訊。

使用註釋與中繼資料，記錄應用程式所做的請求和工作等其他資訊。註釋是簡單的鍵/值對，系統會為其建立索引以用於篩選條件表達式，因此您可以搜尋包含特定資料的追蹤。元數據條目的限制較低，可以記錄整個對象和數組-任何可以序列化為 JSON 的內容。

> **標註與中繼資料**
>
> 註釋和中繼資料是您使用 X-Ray SDK 新增至區段的任意文字。註釋會編製索引以與篩選器運算式搭配使用 中繼資料不會建立索引，但可以使用 X-Ray 主控台或 API 在原始區段中檢視。您授與 X-Ray 讀取權限的任何人都可以檢視此資料。

當程式碼中有許多經過檢測的用戶端時，單一請求區段可能包含大量子區段，每個使用經檢測用戶端進行的呼叫都有一個子區段。您可以將用戶端呼叫包裝在自訂子區段中，以組織和群組子區段。您可以為整個函數或任何部分的程式碼建立自訂子區段，並記錄子區段上的中繼資料和註釋，而不必寫入父區段上的所有項目。

如需軟體開發套件之類別和方法的參考文件，請參閱以下項目：

- AWS X-Ray 適用於 .NET API 參考的 SDK
- AWS X-Ray 適用於 .NET 核心 API 參考的 SDK

相同套件可同時支援 .NET 和 .NET Core，但使用的類別不同。除非類別是 .NET Core 專屬，否則本章範例都會連結至 .NET API 參考。

要求

適用於 .NET 的 X-Ray SDK 需要 .NET 框架 4.5 或更高版本和 AWS SDK for .NET.

若是 .NET Core 應用程式和函數，則軟體開發套件需要 .NET Core 2.0 或更新版本。

將 .NET 的 X-Ray SDK 新增至您的應用程式

用於 NuGet 將適用於 .NET 的 X-Ray SDK 新增至您的應用程式。

若要在 NuGet 套件管理員中安裝適用於 .NET 的 X-Ray SDK

1. 選擇 [工具]、[NuGet P NuGet ackage 件管理員]、[管理方案的套件
2. 搜尋 AWSXRayRecorder。
3. 選擇套件，然後選擇 Install (安裝)。

相依性管理

適用於 .NET 的 X-Ray SDK 可從 Nuget 取得。使用軟件包管理器安裝 SDK：

```
Install-Package AWSXRayRecorder -Version 2.10.1
```

AWSXRayRecorder v2.10.1nuget 套件具有下列相依性：

NET 框架 4.5

```
AWSXRayRecorder (2.10.1)
|
|-- AWSXRayRecorder.Core (>= 2.10.1)
|   |-- AWSSDK.Core (>= 3.3.25.1)
|
|-- AWSXRayRecorder.Handlers.AspNet (>= 2.7.3)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|
|-- AWSXRayRecorder.Handlers.AwsSdk (>= 2.8.3)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|
|-- AWSXRayRecorder.Handlers.EntityFramework (>= 1.1.1)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|   |-- EntityFramework (>= 6.2.0)
|
|-- AWSXRayRecorder.Handlers.SqlServer (>= 2.7.3)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|
|-- AWSXRayRecorder.Handlers.System.Net (>= 2.7.3)
    |-- AWSXRayRecorder.Core (>= 2.10.1)
```

NET 框架

```
AWSXRayRecorder (2.10.1)
|
|-- AWSXRayRecorder.Core (>= 2.10.1)
|   |-- AWSSDK.Core (>= 3.3.25.1)
|   |-- Microsoft.AspNetCore.Http (>= 2.0.0)
|   |-- Microsoft.Extensions.Configuration (>= 2.0.0)
|   |-- System.Net.Http (>= 4.3.4)
|
|-- AWSXRayRecorder.Handlers.AspNetCore (>= 2.7.3)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
```

```
|   |-- Microsoft.AspNetCore.Http.Extensions (>= 2.0.0)
|   |-- Microsoft.AspNetCore.Mvc.Abstractions (>= 2.0.0)
|
|-- AWSXRayRecorder.Handlers.AwsSdk (>= 2.8.3)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|
|-- AWSXRayRecorder.Handlers.EntityFramework (>= 1.1.1)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|   |-- Microsoft.EntityFrameworkCore.Relational (>= 3.1.0)
|
|-- AWSXRayRecorder.Handlers.SqlServer (>= 2.7.3)
|   |-- AWSXRayRecorder.Core (>= 2.10.1)
|   |-- System.Data.SqlClient (>= 4.4.0)
|
|-- AWSXRayRecorder.Handlers.System.Net (>= 2.7.3)
    |-- AWSXRayRecorder.Core (>= 2.10.1)
```

有關依賴管理的更多詳細信息，請參閱微軟關於 Nuget 依賴關係和 Nuget 依賴解決方案的文檔。

配置適用於 .NET 的 X-Ray SDK

您可以使用外掛程式為 .NET 設定 X-Ray SDK，以包含應用程式執行之服務的相關資訊、修改預設取樣行為，或新增套用至特定路徑要求的取樣規則。

針對 .NET web 應用程式，請將鍵新增至您 `Web.config` 檔案中的 `appSettings` 區段。

Example Web.config

```
<configuration>
  <appSettings>
    <add key="AWSXRayPlugins" value="EC2Plugin"/>
    <add key="SamplingRuleManifest" value="sampling-rules.json"/>
  </appSettings>
</configuration>
```

針對 .NET Core，請建立名為 `appsettings.json` 的檔案，其中帶有名為 `XRay` 的最上層鍵。

Example .NET appsettings.json

```
{
  "XRay": {
```

```
    "AWSXRayPlugins": "EC2Plugin",
    "SamplingRuleManifest": "sampling-rules.json"
  }
}
```

然後，在您的應用程式程式碼中，建立設定物件，並使用它來初始化 X-Ray 記錄器。請在初始化記錄器前執行此作業。

Example . NET 核心 Program.cs-記錄器配置

```
using Amazon.XRay.Recorder.Core;
...
AWSXRayRecorder.InitializeInstance(configuration);
```

如果您要檢測 .NET Core Web 應用程式，您也可以在設定訊息處理常式時將組態物件傳遞給UseXRay方法。對於 Lambda 函數，請使用如上所示的InitializeInstance方法。

如需 .NET 核心設定 API 的詳細資訊，請參閱設定 ASP.NET 核心應用程式。

章節

- 外掛程式
- 抽樣規則
- 記錄日誌 (.NET)
- 記錄日誌 (.NET Core)
- 環境變數

外掛程式

使用外掛程式來新增託管您應用程式的服務相關資料。

外掛程式

- Amazon EC2 —EC2Plugin 新增執行個體 ID、可用區域和 CloudWatch 日誌群組。
- Elastic Beanstalk —ElasticBeanstalkPlugin 新增環境名稱、版本標籤和部署 ID。
- 亞馬遜 ECS-ECSPlugin 添加容器 ID。

若要使用外掛程式，請新增AWSXRayPlugins設定，以配置適用於 .NET 用戶端的 X-Ray 開發套件。若要將多個外掛程式套用至您的應用程式，請在相同設定中指定它們，並以逗號分隔。

Example Web.config - 外掛程式

```
<configuration>
  <appSettings>
    <add key="AWSXRayPlugins" value="EC2Plugin,ElasticBeanstalkPlugin"/>
  </appSettings>
</configuration>
```

Example . NET 核心應用程序設置. JSON-插件

```
{
  "XRay": {
    "AWSXRayPlugins": "EC2Plugin,ElasticBeanstalkPlugin"
  }
}
```

抽樣規則

SDK 會使用您在 X-Ray 主控台中定義的取樣規則來決定要記錄的要求。預設規則會每秒追蹤第一個要求，而所有服務的任何其他要求的百分之五會傳送追蹤至 X-Ray。在 X-Ray 主控台中建立其他規則，以自訂為每個應用程式記錄的資料量。

SDK 會依定義順序套用自訂規則。如果要求符合多個自訂規則，SDK 只會套用第一個規則。

> **Note**
>
> 如果 SDK 無法達到 X-Ray 以取得取樣規則，它會每秒還原為第一個要求的預設本機規則，而每台主機的任何其他要求的百分之五。如果主機沒有調用採樣 API 的權限，或者無法連接到 X-Ray 守護程序（作為 SDK 發出的 API 調用的 TCP 代理），則可能會發生這種情況。

您也可以將 SDK 設定為從 JSON 文件載入取樣規則。SDK 可以使用本機規則做為無法使用 X-Ray 取樣的情況的備份，或僅使用本機規則。

Example 採樣規則

```
{
  "version": 2,
  "rules": [
    {
```

```
      "description": "Player moves.",
      "host": "*",
      "http_method": "*",
      "url_path": "/api/move/*",
      "fixed_target": 0,
      "rate": 0.05
    }
  ],
  "default": {
    "fixed_target": 1,
    "rate": 0.1
  }
}
```

此範例定義了一個自訂規則和一個預設規則。自訂規則會套用百分之五的取樣率，而且沒有追蹤下限路徑的要求數目下限/api/move/。預設規則會每秒追蹤第一次請求和 10% 的額外請求。

在本機定義規則的缺點是，固定目標會由記錄器的每個執行個體獨立套用，而不是由 X-Ray 服務管理。當您部署更多主機時，固定費率會倍增，因此更難以控制記錄的資料量。

開啟時AWS Lambda，您無法修改取樣率。如果您的函數是由已檢測的服務呼叫，則 Lambda 會記錄產生由該服務取樣之請求的呼叫。如果啟用主動追蹤且沒有追蹤標頭，Lambda 會做出取樣決策。

若要設定備份規則，請告知 .NET 的 X-Ray SDK 從具有該SamplingRuleManifest設定的檔案載入取樣規則。

Example .NET Web.config - 抽樣規則

```
<configuration>
  <appSettings>
    <add key="SamplingRuleManifest" value="sampling-rules.json"/>
  </appSettings>
</configuration>
```

Example .json 核心應用程序設置-採樣規則

```
{
  "XRay": {
    "SamplingRuleManifest": "sampling-rules.json"
  }
}
```

若僅要使用本機規則，請使用 `LocalizedSamplingStrategy` 建置記錄器。若您已設定備份規則，請移除該組態。

Example . NET-本地抽樣規則

```
var recorder = new AWSXRayRecorderBuilder().WithSamplingStrategy(new
 LocalizedSamplingStrategy("samplingrules.json")).Build();
AWSXRayRecorder.InitializeInstance(recorder: recorder);
```

Example .NET 核心 Program.cs-本地抽樣規則

```
var recorder = new AWSXRayRecorderBuilder().WithSamplingStrategy(new
 LocalizedSamplingStrategy("sampling-rules.json")).Build();
AWSXRayRecorder.InitializeInstance(configuration,recorder);
```

記錄日誌 (.NET)

適用於 .NET 的 X-Ray SDK 使用與 AWS SDK for .NET. 若已配置應用程式以記錄AWS SDK for .NET 輸出，則適用於適用於 .NET 的 X-Ray 開發套件所進行之輸出。

若要設定記錄日誌，請將名為 `aws` 的組態區段新增至您的 `App.config` 檔案或 `Web.config` 檔案。

Example Web.config - 記錄

```
...
<configuration>
  <configSections>
    <section name="aws" type="Amazon.AWSSection, AWSSDK.Core"/>
  </configSections>
  <aws>
    <logging logTo="Log4Net"/>
  </aws>
</configuration>
```

如需詳細資訊，請參閱AWS SDK for .NET開發人員指南中的設定AWS SDK for .NET應用程式。

記錄日誌 (.NET Core)

適用於 .NET 的 X-Ray SDK 使用與 AWS SDK for .NET. 若要設定 .NET Core 應用程式的記錄，請將記錄選項傳遞給方`AWSXRayRecorder.RegisterLogger`法。

例如，若要使用 log4net，請建立定義記錄器、輸出格式及檔案位置的組態檔。

Example .NET Core log4net.config

```
<?xml version="1.0" encoding="utf-8" ?>
<log4net>
  <appender name="FileAppender" type="log4net.Appender.FileAppender,log4net">
    <file value="c:\logs\sdk-log.txt" />
    <layout type="log4net.Layout.PatternLayout">
      <conversionPattern value="%date [%thread] %level %logger - %message%newline" />
    </layout>
  </appender>
  <logger name="Amazon">
    <level value="DEBUG" />
    <appender-ref ref="FileAppender" />
  </logger>
</log4net>
```

然後建立記錄器並在您的程式程式碼中套用組態。

Example . NET 核心 Program.cs-日誌記錄

```
using log4net;
using Amazon.XRay.Recorder.Core;

class Program
{
  private static ILog log;
  static Program()
  {
    var logRepository = LogManager.GetRepository(Assembly.GetEntryAssembly());
    XmlConfigurator.Configure(logRepository, new FileInfo("log4net.config"));
    log = LogManager.GetLogger(typeof(Program));
    AWSXRayRecorder.RegisterLogger(LoggingOptions.Log4Net);
  }
  static void Main(string[] args)
  {
  ...
  }
}
```

如需有關設定 log4net 的詳細資訊，請參閱登入 .apache.org 的設定。

環境變數

您可以使用環境變數，配置適用於 .NET 的 X-Ray 開發套件。軟體開發套件支援以下變數。

- `AWS_XRAY_TRACING_NAME`— 設定 SDK 用於區段的服務名稱。覆寫您在 servlet 篩選條件的區段命名策略中設定的服務名稱。
- `AWS_XRAY_DAEMON_ADDRESS`— 設定 X-Ray 精靈監聽程式的主機和連接埠。根據預設，SDK 會同時用`127.0.0.1:2000`於追蹤資料 (UDP) 和取樣 (TCP)。如果您已將協助程式設定為在不同的連接埠上接聽，或是在不同的主機上執行，請使用此變數。

  格式

  - 相同的連接埠 —*`address`*`:`*`port`*
  - 不同的端口`-tcp:`*`address`*`:`*`port`* `udp:`*`address`*`:`*`port`*
- `AWS_XRAY_CONTEXT_MISSING`— 設定`RUNTIME_ERROR`為當您的檢測程式碼嘗試在沒有區段開啟時記錄資料時擲回例外狀況。

  有效值

  - `RUNTIME_ERROR`— 擲回執行階段例外狀況。
  - `LOG_ERROR`— 記錄錯誤並繼續 (預設值)。
  - `IGNORE_ERROR`— 忽略錯誤並繼續。

  當您嘗試在沒有要求開啟時執行的啟動程式碼中使用已檢測的用戶端，或在產生新執行緒的程式碼中使用已檢測的用戶端時，可能會發生與遺失區段或子區段相關的錯誤。

使用 X-Ray SDK 檢測傳入 HTTP 請求

您可以使用 X-Ray SDK 來追蹤您應用程式在 Amazon EC2 中 EC2 執行個體上處理的傳入 HTTP 請求，AWS Elastic Beanstalk或亞馬遜彈性雲服務器。

使用訊息處理常式檢測傳入的 HTTP 請求。當您新增 X-Ray 訊息處理常式新增至應用程式時，.NET X-Ray SDK 會為每個抽樣請求建立區段。此區段包括時間、方法，以及 HTTP 請求的處置方式。其他檢測會在此區段上建立子區段。

> **Note**
>
> 適用於AWS Lambda函數時，Lambda 會為每個抽樣請求建立區段。如需詳細資訊，請參閱 AWS Lambda 而且 AWS X-Ray。

每個區段都有一個用於標識服務映射中的應用程序的名稱。可以靜態命名段，也可以將 SDK 配置為基於傳入請求中的主機標頭動態命名它。動態命名允許您根據請求中的域名對跟蹤進行分組，如果名稱與預期模式不匹配（例如，如果主機標頭是偽造的），則應用默認名稱。

> **轉發請求**
>
> 如果負載均衡器或其他中間機構將請求轉發給您的應用程序，X-Ray 會從`X-Forwarded-For`標頭，而不是來自 IP 數據包中的源 IP。為轉發請求記錄的客户端 IP 可以偽造，因此不應信任該 IP。

訊息處理常式會使用 `http` 區塊為每個傳入的請求建立區段，其中包含以下資訊：

- HTTP method (HTTP 方法)— 獲取、張貼、刪除等。
- 用户端地址— 傳送請求的用户端 IP 地址。
- 回應代碼— 已完成請求的 HTTP 回應代碼。
- Timing (時間點)— 開始時間 (收到請求) 和結束時間 (傳送回應)。
- 用户代理程式—`user-agent`從請求。
- 內容長度—`content-length`從回應。

章節

檢測傳入的請求 (.NET)

若要檢測您應用程式處理的請求，請在您 `global.asax` 檔案的 `Init` 方法中呼叫 `RegisterXRay`。

Example global.asax - 訊息處理常式

```
using System.Web.Http;
using Amazon.XRay.Recorder.Handlers.AspNet;

namespace SampleEBWebApplication
{
  public class MvcApplication : System.Web.HttpApplication
```

```
  {
    public override void Init()
    {
      base.Init();
      AWSXRayASPNET.RegisterXRay(this, "MyApp");
    }
  }
}
```

檢測傳入的請求 (.NET Core)

若要檢測您應用程式處理的請求，請致電UseXRay方法之前的任何其他中間件Configure方法作為理想情況下 X-Ray 中間件應該是處理請求的第一箇中間件，最後一箇中間件應該是處理管道中響應的最後一箇中間件。

> **Note**
>
> 對於 .NET Core 2.0，如果您有UseExceptionHandler方法，請確保調用UseXRay之後UseExceptionHandler方法來確保記錄異常。

Example Startup.cs

<caption>.NET Core 2.1 and above</caption>

```
using Microsoft.AspNetCore.Builder;

public void Configure(IApplicationBuilder app, IHostingEnvironment env)
  {
    app.UseXRay("MyApp");
    // additional middleware
    ...
  }
```

<caption>.NET Core 2.0</caption>

```
using Microsoft.AspNetCore.Builder;

public void Configure(IApplicationBuilder app, IHostingEnvironment env)
  {
    app.UseExceptionHandler("/Error");
    app.UseXRay("MyApp");
```

```
    // additional middleware
    ...
  }
```

UseXRay 方法也可以接收組態物件做為第二個引數。

```
app.UseXRay("MyApp", configuration);
```

設定區段命名策略

AWS X-Ray使用服務名稱來識別您的應用程序，並將其與其他應用程序、數據庫、外部 API 和AWS您應用程式使用的資源。當 X-Ray SDK 為傳入請求生成段時，它會將應用程序的服務名稱記錄在區段的名稱欄位。

X-Ray SDK 可以在 HTTP 請求標頭中的主機名後面命名段。但是，此標頭可能是偽造的，這可能會導致服務映射中出現意外的節點。要防止 SDK 由於使用偽造主機標頭的請求而錯誤地命名段，您必須為傳入請求指定默認名稱。

如果您的應用程序為多個域的請求提供服務，則可以將 SDK 配置為使用動態命名策略在段名稱中反映這一點。動態命名策略允許 SDK 對與預期模式匹配的請求使用主機名，並將默認名稱應用於不匹配的請求。

例如，您可能有單個應用程式處理三個子域名的請求 —www.example.com、api.example.com，和static.example.com。您可以將動態命名策略與模式一起使用*.example.com來標識具有不同名稱的每個子域的段，從而在服務映射上生成三個服務節點。如果您的應用程序收到的主機名與模式不匹配的請求，則您將在服務映射上看到第四個節點，其中包含您指定的回退名稱。

若要為所有請求區段使用相同的名稱，請如上一節所述，在初始化訊息處理常式時指定您應用程式的名稱。這與建立 FixedSegmentNamingStrategy 並將其傳遞給 RegisterXRay 方法有相同的效果。

```
AWSXRayASPNET.RegisterXRay(this, new FixedSegmentNamingStrategy("MyApp"));
```

> **Note**
>
> 您可以使用 AWS_XRAY_TRACING_NAME 環境變數來覆寫您在程式碼中定義的預設服務名稱。

動態命名策略可定義主機名稱應相符的模式，以及如果 HTTP 請求中的主機名稱不符合模式時要使用的預設名稱。若要動態命名區段，請建立 DynamicSegmentNamingStrategy 並將其傳遞至 RegisterXRay 方法。

```
AWSXRayASPNET.RegisterXRay(this, new DynamicSegmentNamingStrategy("MyApp",
 "*.example.com"));
```

使用適用於 .NET 的 X-Ray AWS SDK 追蹤 SDK 呼叫

當您的應用程式呼叫 AWS 服務 以儲存資料、寫入佇列或傳送通知時，.NET 的 X-Ray SDK 會追蹤子區段中下游的呼叫。您在這些服務中存取的追蹤 AWS 服務 和資源 (例如，Amazon S3 儲存貯體或 Amazon SQS 佇列) 會在 X-Ray 主控台的追蹤對應上顯示為下游節點。

您可以在創建`RegisterXRayForAllServices`之前通過致電來檢測所有 AWS SDK for .NET 客戶。

Example SampleController.cs-DynamoDB 端儀器

```
using Amazon;
using Amazon.Util;
using Amazon.DynamoDBv2;
using Amazon.DynamoDBv2.DocumentModel;
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.AwsSdk;

namespace SampleEBWebApplication.Controllers
{
  public class SampleController : ApiController
  {
    AWSSDKHandler.RegisterXRayForAllServices();
    private static readonly Lazy<AmazonDynamoDBClient> LazyDdbClient = new
 Lazy<AmazonDynamoDBClient>(() =>
    {
      var client = new AmazonDynamoDBClient(EC2InstanceMetadata.Region ??
 RegionEndpoint.USEast1);
      return client;
    });
```

若要針對某些特定服務檢測用戶端，請呼叫 `RegisterXRay`，而不是 `RegisterXRayForAllServices`。將醒目提示的文字取代為服務的用戶端界面名稱。

```
AWSSDKHandler.RegisterXRay<IAmazonDynamoDB>()
```

對於所有服務，您都可以在 X-Ray 控制台中看到調用的 API 的名稱。對於服務子集，X-Ray SDK 會將資訊新增至區段，以在服務對應中提供更多精細度。

例如，當您使用已檢測的 DynamoDB 用戶端進行呼叫時，SDK 會將資料表名稱新增至區段，以便針對以資料表為目標的呼叫。在主控台中，每個表格在服務對應中顯示為獨立節點，其中包含一般 DynamoDB 節點，用於未針對資料表的呼叫。

Example 呼叫 DynamoDB 以儲存項目的子區段

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

您存取具名資源時，對以下服務的呼叫會在服務地圖中建立額外節點。未針對特定資源的呼叫，則會建立服務的一般節點。

- Amazon DynamoDB — 表名
- Amazon 簡單存儲服務 — 存儲桶和密鑰名稱
- Amazon 簡單隊列服務-隊列名稱

使用 X-Ray SDK 進行的 X-Ray SDK 來追蹤對下游 HTTP Web 服務進行的呼叫

當您的應用程式呼叫微服務或公有 HTTP API 時，您可以使用 X-Ray SDK 進行的 X-Ray SDK`GetResponseTraced`的延伸方法`System.Net.HttpWebRequest`來檢測這些呼叫，並將 API 做為下游服務新增到服務圖表。

Example HttpWebRequest

```
using System.Net;
```

```
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.System.Net;

private void MakeHttpRequest()
{
  HttpWebRequest request = (HttpWebRequest)WebRequest.Create("http://names.example.com/
api");
  request.GetResponseTraced();
}
```

針對非同步呼叫，請使用 GetAsyncResponseTraced。

```
request.GetAsyncResponseTraced();
```

若您使用 system.net.http.httpclient，請使用 HttpClientXRayTracingHandler 委派處理常式來記錄呼叫。

Example HttpClient

```
using System.Net.Http;
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.System.Net;

private void MakeHttpRequest()
{
  var httpClient = new HttpClient(new HttpClientXRayTracingHandler(new
 HttpClientHandler()));
  httpClient.GetAsync(URL);
}
```

當您檢測對下游 Web API 進行的呼叫時，The X-Ray SDK for .NET 的 X-Ray SDK) 會記錄附帶 HTTP 請求和回應相關資訊。X-Ray 會使用子區段來產生 API 的推論區段。

Example 下游 HTTP 呼叫的子區段

```
{
  "id": "004f72be19cddc2a",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "name": "names.example.com",
  "namespace": "remote",
  "http": {
```

```
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  }
}
```

Example 下游 HTTP 呼叫的推斷區段

```
{
  "id": "168416dc2ea97781",
  "name": "names.example.com",
  "trace_id": "1-62be1272-1b71c4274f39f122afa64eab",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "parent_id": "004f72be19cddc2a",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  },
  "inferred": true
}
```

使用適用於 .NET 的 X-Ray 開發套件追蹤 SQL 查詢

適用於 .NET 的 X-Ray 軟體開發套件為`System.Data.SqlClient.SqlCommand`, 名為`TraceableSqlCommand`，可用來代替`SqlCommand`。您可以使用 `TraceableSqlCommand` 類別來初始化 SQL 命令。

使用同步和非同步方法追蹤 SQL 查詢

以下範例說明如何使用 `TraceableSqlCommand` 來以同步和非同步方式自動追蹤 SQL Server 查詢。

Example **Controller.cs** - SQL 用戶端檢測 (同步)

```
using Amazon;
using Amazon.Util;
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.SqlServer;

private void QuerySql(int id)
{
  var connectionString = ConfigurationManager.AppSettings["RDS_CONNECTION_STRING"];
  using (var sqlConnection = new SqlConnection(connectionString))
  using (var sqlCommand = new TraceableSqlCommand("SELECT " + id, sqlConnection))
  {
    sqlCommand.Connection.Open();
    sqlCommand.ExecuteNonQuery();
  }
}
```

您可以使用 ExecuteReaderAsync 方法，以非同步方式執行查詢。

Example **Controller.cs** - SQL 用戶端檢測 (非同步)

```
using Amazon;
using Amazon.Util;
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.SqlServer;
private void QuerySql(int id)
{
  var connectionString = ConfigurationManager.AppSettings["RDS_CONNECTION_STRING"];
  using (var sqlConnection = new SqlConnection(connectionString))
  using (var sqlCommand = new TraceableSqlCommand("SELECT " + id, sqlConnection))
  {
    await sqlCommand.ExecuteReaderAsync();
  }
}
```

收集對 SQL Server 執行的 SQL 查詢

您可以啟用 SqlCommand.CommandText 的擷取，做為 SQL 查詢所建立子區段的一部分。SqlCommand.CommandText 會在子區段 JSON 中顯示為欄位 sanitized_query。基於安全考量，此功能預設為停用。

> **Note**
>
> 如果您在 SQL 查詢中包含純文字形式的敏感資訊，請勿啟用收集功能。

您可以透過兩種方式啟用收集 SQL 查詢：

- 針對您的應用程式，在全域組態中將 `CollectSqlQueries` 屬性設定為 `true`。
- 將 `TraceableSqlCommand` 執行個體中的 `collectSqlQueries` 參數設為 `true`，以收集執行個體內的呼叫。

啟用全域 CollectSqlQueries 屬性

以下範例說明如何啟用適用於 .NET 和 .NET Core 的 `CollectSqlQueries` 屬性。

.NET

若要在 .NET 中將您應用程式全域組態的 `CollectSqlQueries` 屬性設為 `true`，請修改您 `App.config` 或 `Web.config` 檔案的 `appsettings`，如下所示。

Example **`App.config`或`Web.config`**— 全域啟用 SQL 查詢收集

```
<configuration>
<appSettings>
    <add key="CollectSqlQueries" value="true">
</appSettings>
</configuration>
```

.NET Core

若要設置`CollectSqlQueries`屬性設置為`true`在 .NET Core 中您應用程式全域組態，請修改您`appsettings.json`文件，X-Ray 圖所示。

Example **`appsettings.json`**— 全域啟用 SQL 查詢收集

```
{
  "XRay": {
    "CollectSqlQueries":"true"
  }
}
```

啟用 collectSqlQueries 參數

您可以將 `TraceableSqlCommand` 執行個體中的 `collectSqlQueries` 參數設為 `true`，以收集使用該執行個體進行之 SQL Server 查詢的 SQL 查詢文字。將參數設為 `false` 會停用 `TraceableSqlCommand` 執行個體的 `CollectSqlQuery` 功能。

> **Note**
>
> `TraceableSqlCommand` 執行個體中的 `collectSqlQueries` 值會覆寫 `CollectSqlQueries` 屬性之全域組態中設定的值。

Example 範例**`Controller.cs`**— 啟用執行個體的 SQL 查詢收集

```
using Amazon;
using Amazon.Util;
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.SqlServer;

private void QuerySql(int id)
{
  var connectionString = ConfigurationManager.AppSettings["RDS_CONNECTION_STRING"];
  using (var sqlConnection = new SqlConnection(connectionString))
  using (var command = new TraceableSqlCommand("SELECT " + id, sqlConnection,
 collectSqlQueries: true))
  {
    command.ExecuteNonQuery();
  }
}
```

建立其他子區段

子段擴展追蹤的段，詳細介紹了為了服務請求而完成的工作。每次與分析客户端進行呼叫時，X-Ray SDK 都會記錄在子段中生成的信息。您可以創建其他子段來對其他子段進行分組、測量代碼部分的性能或記錄註釋和元數據。

若要管理子區段，請使用 `BeginSubsegment` 和 `EndSubsegment` 方法。執行 `try` 區塊之子區段中的任何工作，並使用 `AddException` 追蹤例外狀況。呼叫 `finally` 區塊中的 `EndSubsegment`，以確保子區段結束。

Example Controller.cs — 自訂子區段

```
AWSXRayRecorder.Instance.BeginSubsegment("custom method");
try
{
  DoWork();
}
catch (Exception e)
{
  AWSXRayRecorder.Instance.AddException(e);
}
finally
{
  AWSXRayRecorder.Instance.EndSubsegment();
}
```

當您在某區段或其他子區段內建立子區段時，SDK for .NET 的 X-Ray 開發套件將為其產生一個 ID，並記錄開始時間和結束時間。

Example 使用中繼資料的子區段

```
"subsegments": [{
  "id": "6f1605cd8a07cb70",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "Custom subsegment for UserModel.saveUser function",
  "metadata": {
    "debug": {
      "test": "Metadata string from UserModel.saveUser"
    }
  },
```

使用適用於 .NET 的 X-Ray SDK 將註釋和中繼資料新增至區段

您可以使用註釋和中繼資料來記錄有關請求、環境或應用程式的其他資訊。您可以將註釋和中繼資料新增至 X-Ray SDK 建立的區段，或新增至您建立的自訂子區段。

註釋是與字符串，數字或布爾值鍵-值對。註釋會編製索引以與篩選器運算式搭配使用 使用標記記錄您想要用來在主控台將追蹤分組的資料，或是在呼叫 `GetTraceSummaries` API 時使用標記。

中繼資料是索引鍵-值配對，可以具有任何類型的值 (包括物件和清單)，但不會編製索引以供篩選運算式使用。使用元數據記錄要存儲在跟踪中但不需要與搜索一起使用的其他數據。

章節

使用適用於 .NET 的 X-Ray SDK 錄製註釋

針對您想要建立索引以用於搜尋的區段或子區段，請使用標註來記錄這些區段上的資訊。

X-Ray 中的所有註記都需要以下內容：

註釋要求

- 按鍵 — X-Ray 註解的金鑰最多可包含 500 個英數字元。您不能使用底線符號 (_) 以外的空格或符號。
- 值 — X-Ray 註釋的值最多可包含 1,000 個 Unicode 字元。
- 註釋的數量 — 每個追蹤最多可以使用 50 個註釋。

若要記錄 AWS Lambda 函數外部的註釋

1. 取得 `AWSXRayRecorder` 的執行個體。

```
using Amazon.XRay.Recorder.Core;
...
AWSXRayRecorder recorder = AWSXRayRecorder.Instance;
```

2. 使用字串鍵、布林值、Int32、Int64、雙精確度浮點數或字串值，呼叫 `addAnnotation`。

```
recorder.AddAnnotation("mykey", "my value");
```

若要在 AWS Lambda 函數內記錄註釋

Lambda 函數中的區段和子區段都是由 Lambda 執行階段環境管理。如果要將註釋新增至 Lambda 函數內的區段或子區段，則必須執行下列動作：

1. 在 Lambda 函數內建立區段或子區段。
2. 將註釋加入至段或子段。
3. 結束段或子段。

下列程式碼範例示範如何將註解新增至 Lambda 函數內的子區段：

```
#Create the subsegment
AWSXRayRecorder.Instance.BeginSubsegment("custom method");
#Add an annotation
AWSXRayRecorder.Instance.AddAnnotation("My", "Annotation");
try
{
  YourProcess(); #Your function
}
catch (Exception e)
{
  AWSXRayRecorder.Instance.AddException(e);
}
finally #End the subsegment
{
  AWSXRayRecorder.Instance.EndSubsegment();
}
```

X-Ray SDK 會將註釋記錄為區段文件中annotations物件中的索引鍵值配對。使用相同索引鍵呼叫addAnnotation作業兩次，會覆寫相同區段或子區段上先前記錄的值。

若要尋找具有特定值之註釋的繪線，請在篩選器運算式中使用annotations.*key*關鍵字。

使用適用於 .NET 的 X-Ray SDK 記錄元數據

使用中繼資料來記錄您不需要編製索引以便在搜尋中使用的區段或子區段的資訊。元數據值可以是字符串，數字，布爾值或任何其他可以序列化為 JSON 對象或數組的對象。

記錄中繼資料

1. 取得的執行個體AWSXRayRecorder，如下列程式碼範例所示：

   ```
   using Amazon.XRay.Recorder.Core;
   ...
   AWSXRayRecorder recorder = AWSXRayRecorder.Instance;
   ```

2. AddMetadata使用字串命名空間、字串索引鍵和物件值呼叫，如下列程式碼範例所示：

   ```
   recorder.AddMetadata("my namespace", "my key", "my value");
   ```

   您也可以只使用索引鍵和值組來呼叫AddMetadata作業，如下列程式碼範例所示：

```
recorder.AddMetadata("my key", "my value");
```

如果您未指定命名空間的值，X-Ray SDK 會使用`default`。使用相同索引鍵呼叫`AddMetadata`作業兩次，會覆寫相同區段或子區段上先前記錄的值。

使用 Ruby 檢測您的應用程式

有兩種方法可以檢測 Ruby 應用程式，將追蹤傳送至 X-Ray：

- AWS OpenTelemetry Ruby 發行版 — 提供一組開放原始碼程式庫的 AWS 發行版，可透過收集器的發行AWS 版，將相關的指標和追蹤傳送至包括 Amazon CloudWatch 和 Amazon OpenSearch 服務在內的多個 AWS 監控解決方案。 AWS X-Ray OpenTelemetry
- AWS X-Ray SDK for Ruby — 一組程式庫，用於透過 X-Ray 守護程式產生追蹤並將其傳送至 X-Ray。

如需更多詳細資訊，請參閱 在 AWS 發行版 OpenTelemetry 和 X-Ray SDK 之間進行選擇。

AWS發行版的OpenTelemetry紅寶石

與AWS發行版的OpenTelemetry(ADOT) Ruby，您可以一次檢測您的應用程式，並將相關的指標和追蹤傳送給多個AWS監控解決方案，包括CloudWatch,AWS X-Ray和亞馬遜OpenSearch服務。使用 X 射線與 ADOT 需要兩個組成部分：OpenTelemetrySDK已啟用與 X 射線搭配使用，以及AWS發行版的OpenTelemetry收藏家已啟用以與 X 射線搭配使用。

若要開始使用，請參閱AWS發行版的OpenTelemetry紅寶石文檔。

有關使用的更多信息AWS發行版的OpenTelemetry與AWS X-Ray和其他AWS 服務，請參閱AWS發行版的OpenTelemetry或AWS發行版的OpenTelemetry文件。

如需語言支援和使用方式的詳細資訊，請參閱AWS上的可觀測性GitHub。

AWS X-Ray SDK for Ruby

X-Ray SDK 是 Ruby Web 應用程式的程式庫，提供產生追蹤資料並將其傳送至 X-Ray 精靈的類別和方法。追蹤資料包括應用程式所提供之傳入 HTTP 要求的相關資訊，以及應用程式使用 AWS SDK、HTTP 用戶端或使用中記錄用戶端對下游服務進行呼叫的相關資訊。您也可以手動建立區段，並將除錯資訊新增至註釋和中繼資料中。

您可以透過將它新增至您的 gemfile 並執行 `bundle install`，來下載軟體開發套件。

Example Gemfile

```
gem 'aws-sdk'
```

如果您使用 Rails，請先新增 X-Ray SDK 中介軟體來追蹤傳入的要求。請求篩選條件會建立區段。當區段開啟時，您可以使用軟體開發套件用戶端的方法，將資訊新增到區段，並建立子區段以追蹤下游呼叫。軟體開發套件也會在區段為開啟時自動記錄應用程式擲回的例外狀況。針對非 Rails 應用程式，您可以手動建立區段。

接下來，使用 X-Ray SDK 來測量您的 AWS SDK for Ruby、HTTP 和 SQL 用戶端，方法是將記錄器設定為修補相關的程式庫。每當您使用已檢測的用戶端對下游 AWS 服務 或資源進行呼叫時，SDK 都會在子區段中記錄有關呼叫的資訊。 AWS 服務 而您在服務中存取的資源會顯示為追蹤對映上的下游節點，以協助您識別個別連線上的錯誤和節流問題。

開始使用軟體開發套件之後，請設定記錄器以自訂其行為。您可以新增外掛程式，以記錄執行應用程式所需的運算資源相關資料、定義抽樣規則以自訂抽樣行為，並提供記錄器以在應用程式日誌中查看更多或更少的軟體開發套件資訊。

使用註釋與中繼資料，記錄應用程式所做的請求和工作等其他資訊。註釋是簡單的鍵/值對，系統會為其建立索引以用於篩選條件表達式，因此您可以搜尋包含特定資料的追蹤。元數據條目的限制較低，可以記錄整個對象和數組-任何可以序列化為 JSON 的內容。

> **標註與中繼資料**
>
> 註釋和中繼資料是您使用 X-Ray SDK 新增至區段的任意文字。註釋會編製索引以與篩選器運算式搭配使用 中繼資料不會建立索引，但可以使用 X-Ray 主控台或 API 在原始區段中檢視。您授與 X-Ray 讀取權限的任何人都可以檢視此資料。

當程式碼中有很多經過檢測的用戶端時，單一請求區段可能包含大量子區段，每個使用經檢測用戶端進行的呼叫都有一個子區段。您可以將用戶端呼叫包裝在自訂子區段中，以組織和群組子區段。您可以為整個函數或任何部分的程式碼建立自訂子區段，並記錄子區段上的中繼資料和註釋，而不必寫入父區段上的所有項目。

如需 SDK 類別和方法的參考文件，請參閱 Ruby API 參考的AWS X-Ray SDK。

要求

X-Ray SDK 需要 Ruby 2.3 或更新版本，並且與下列程式庫相容：

- AWS SDK for Ruby 版本 3.0 或更高版本
- Rails 版本 5.1 或更新版本

配置 Ruby 的 X-Ray 開發套件

Ruby 的 X-Ray 開發套件具有名為的類別`XRay.recorder`，提供全域記錄程式。您可以設定全域記錄器來自訂為傳入 HTTP 呼叫建立區段的中介軟體。

章節

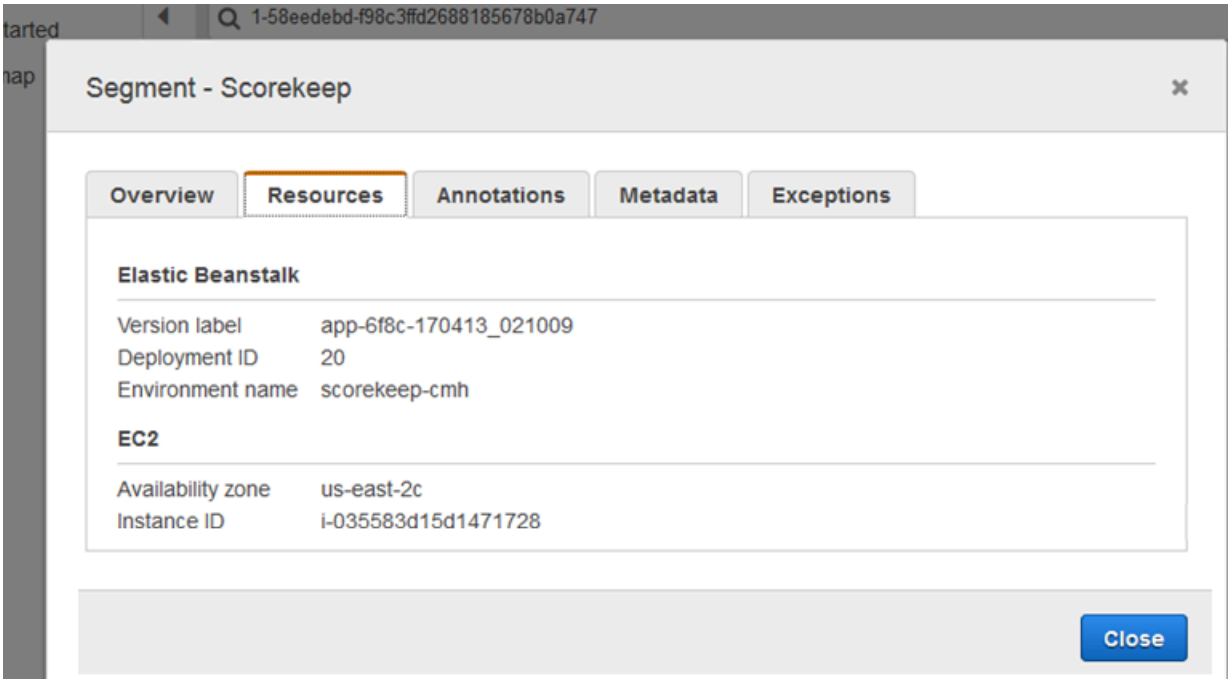
- 服務外掛程式
- 抽樣規則
- 日誌
- 程式碼中的記錄器組態
- 使用 Rails 的記錄器組態
- 環境變數

服務外掛程式

用`plugins`於記錄託管應用程式之服務的相關資訊。

外掛程式

- Amazon EC2-`ec2` 新增執行個體 ID 和可用區域。
- Elastic Beanstalk —`elastic_beanstalk` 新增環境名稱、版本標籤和部署 ID。
- 亞馬遜 ECS-`ecs` 添加容器 ID。



若要使用外掛程式，請在您傳遞至記錄器的組態物件中指定它。

Example 主 .rb-外掛程式組態

```
my_plugins = %I[ec2 elastic_beanstalk]

config = {
  plugins: my_plugins,
  name: 'my app',
}

XRay.recorder.configure(config)
```

您也可以使用優先於程式碼中設定值的環境變數，來設定記錄器。

SDK 也會使用外掛程式設定來設定區origin段上的欄位。這表示運行您的應用程序的AWS資源的類型。當您使用多個外掛程式時，SDK 會使用下列解析順序來判斷來源：ElasticBeanstalk> EKS > ECS > EC2。

抽樣規則

SDK 會使用您在 X-Ray 主控台中定義的取樣規則來決定要記錄的要求。預設規則會每秒追蹤第一個要求，而所有服務的任何其他要求的百分之五會傳送追蹤至 X-Ray。在 X-Ray 主控台中建立其他規則，以自訂為每個應用程式記錄的資料量。

SDK 會依定義順序套用自訂規則。如果要求符合多個自訂規則，SDK 只會套用第一個規則。

> **Note**
>
> 如果 SDK 無法達到 X-Ray 以取得取樣規則，它會每秒還原為第一個要求的預設本機規則，而每台主機的任何其他要求的百分之五。如果主機沒有調用採樣 API 的權限，或者無法連接到 X-Ray 守護程序（作為 SDK 發出的 API 調用的 TCP 代理），則可能會發生這種情況。

您也可以將 SDK 設定為從 JSON 文件載入取樣規則。SDK 可以使用本機規則做為無法使用 X-Ray 取樣的情況的備份，或僅使用本機規則。

Example 採樣規則

```
{
  "version": 2,
  "rules": [
    {
      "description": "Player moves.",
      "host": "*",
      "http_method": "*",
      "url_path": "/api/move/*",
      "fixed_target": 0,
      "rate": 0.05
    }
  ],
  "default": {
    "fixed_target": 1,
    "rate": 0.1
  }
}
```

此範例定義了一個自訂規則和一個預設規則。自訂規則會套用百分之五的取樣率，而且沒有追蹤下限路徑的要求數目下限`/api/move/`。預設取樣規則每秒 1 次請求和 10% 的額外請求。

在本機定義規則的缺點是，固定目標會由記錄器的每個執行個體獨立套用，而不是由 X-Ray 服務管理。當您部署更多主機時，固定費率會倍增，因此更難以控制記錄的資料量。

若要設定備份規則，請在您傳遞至記錄器的組態物件中定義文件的雜湊。

Example 主 .rb-Backup 規則配置

```
require 'aws-xray-sdk'
my_sampling_rules =  {
  version: 1,
  default: {
    fixed_target: 1,
    rate: 0.1
  }
}
config = {
  sampling_rules: my_sampling_rules,
  name: 'my app',
}
XRay.recorder.configure(config)
```

若要單獨存放抽樣規則，請在單獨的檔案中定義雜湊，然後要求 (require) 檔案來將其提取到您的應用程式中。

Example config/sampling-rules.rb

```
my_sampling_rules =  {
  version: 1,
  default: {
    fixed_target: 1,
    rate: 0.1
  }
}
```

Example 主 .rb-從一個文件的採樣規則

```
require 'aws-xray-sdk'
require 'config/sampling-rules.rb'

config = {
  sampling_rules: my_sampling_rules,
  name: 'my app',
}
XRay.recorder.configure(config)
```

若要僅使用本機規則，請要求 (require) 抽樣規則並設定 `LocalSampler`。

Example 主 .rb — 本地規則抽樣

```
require 'aws-xray-sdk'
require 'aws-xray-sdk/sampling/local/sampler'

config = {
  sampler: LocalSampler.new,
  name: 'my app',
}
XRay.recorder.configure(config)
```

您也可以設定全域記錄器來停用所有傳入請求的抽樣和檢測。

Example 主 .rb-禁用採樣

```
require 'aws-xray-sdk'
config = {
  sampling: false,
  name: 'my app',
}
XRay.recorder.configure(config)
```

日誌

根據預設，記錄器會將資訊層級的事件輸出到 `$stdout`。您可以透過在傳遞至記錄器的組態物件中定義 logger 來自定記錄日誌。

Example 主 .rb-日誌記錄

```
require 'aws-xray-sdk'
config = {
  logger: my_logger,
  name: 'my app',
}
XRay.recorder.configure(config)
```

當您手動產生子區段時，可使用除錯日誌來識別問題，例如未結束的子區段。

程式碼中的記錄器組態

可以從 `XRay.recorder` 上的 `configure` 方法取得其他可用設定。

- context_missing— 設定為以LOG_ERROR避免在檢測過的程式碼嘗試在沒有區段開啟時記錄資料時擲回例外狀況。
- daemon_address— 設定 X-Ray 精靈監聽程式的主機和連接埠。
- name— 設定 SDK 用於區段的服務名稱。
- naming_pattern— 設定網域名稱模式以使用動態命名。
- plugins— 使用插件記錄有關應用程序AWS資源的信息。
- sampling— 設定false為停用取樣。
- sampling_rules-設置包含您的採樣規則的哈希值。

Example main.rb-禁用上下文缺少異常

```
require 'aws-xray-sdk'
config = {
  context_missing: 'LOG_ERROR'
}

XRay.recorder.configure(config)
```

使用 Rails 的記錄器組態

若您使用 Rails 框架，您可以在位於 app_root/initializers 之下的 Ruby 檔案中設定全域記錄器上的選項。X-Ray SDK 支援一個額外的設定金鑰，以便與 Rails 搭配使用。

- active_record— 設定為記錄true使用中記錄資料庫交易的子區段。

在名為 Rails.application.config.xray 的組態物件中設定可用設定。

Example config/initializers/aws_xray.rb

```
Rails.application.config.xray = {
  name: 'my app',
  patch: %I[net_http aws_sdk],
  active_record: true
}
```

環境變數

您可以使用環境變數，為 Ruby 配置 X-Ray SDK。軟體開發套件支援以下變數：

- AWS_XRAY_TRACING_NAME— 設定 SDK 用於區段的服務名稱。覆寫您在 servlet 篩選條件的區段命名策略中設定的服務名稱。
- AWS_XRAY_DAEMON_ADDRESS— 設定 X-Ray 精靈監聽程式的主機和連接埠。依預設，SDK 會將追蹤資料傳送至127.0.0.1:2000。如果您已將協助程式設定為在不同的連接埠上接聽，或是在不同的主機上執行，請使用此變數。
- AWS_XRAY_CONTEXT_MISSING— 設定RUNTIME_ERROR為當您的檢測程式碼嘗試在沒有區段開啟時記錄資料時擲回例外狀況。

  有效值

  - RUNTIME_ERROR— 擲回執行階段例外狀況。
  - LOG_ERROR— 記錄錯誤並繼續 (預設值)。
  - IGNORE_ERROR— 忽略錯誤並繼續。

  當您嘗試在沒有要求開啟時執行的啟動程式碼中使用已檢測的用戶端，或在產生新執行緒的程式碼中使用已檢測的用戶端時，可能會發生與遺失區段或子區段相關的錯誤。

環境變數會覆寫程式碼中所設的值。

使用 Ruby 中介軟體的 X-Ray SDK 追蹤傳入的要求

您可以使用 X-Ray 開發套件來追蹤應用程式在 Amazon EC2 或 Amazon ECS 的 EC2 執行個體上提供的傳入 HTTP 請求。 AWS Elastic Beanstalk

若您使用 Rails，請使用 Rails 中介軟體來檢測傳入的 HTTP 請求。當您將中介軟體新增至應用程式並設定區段名稱時，Ruby 的 X-Ray SDK 會為每個取樣要求建立區段。任何由額外檢測建立的區段都會成為子區段，位於提供 HTTP 請求及回應資訊的請求層級區段之下。此資訊包括時間、方法，以及請求的處置方式。

每個區段都有一個名稱，可在服務對應中識別您的應用程式。區段可以以靜態方式命名，也可以設定 SDK 根據傳入要求中的主機標頭動態命名區段。動態命名可讓您根據要求中的網域名稱對追蹤進行分組，並在名稱與預期的模式不符時套用預設名稱 (例如，如果主機標頭是偽造的)。

> **轉寄的要求**
>
> 如果負載平衡器或其他中介機構將要求轉寄至您的應用程式，X-Ray 會從要求中的X-Forwarded-For標頭取得用戶端 IP，而不是從 IP 封包中的來源 IP 取得。為轉寄要求所記錄的用戶端 IP 可以偽造，因此不應該受信任。

轉送請求時，SDK 會在區段中設定一個額外的欄位來指出這一點。如果區段包含`x_forwarded_for`設定為的欄位`true`，則會從 HTTP 要求中的`X-Forwarded-For`標頭取得用戶端 IP。

中介軟體會使用 `http` 區塊為每個傳入的請求建立區段，其中包含以下資訊：

- HTTP 方法-獲取，發布，把，刪除等。
- [用戶端位址] — 傳送要求的用戶端 IP 位址。
- 回應碼 — 已完成要求的 HTTP 回應碼。
- 時間 — 開始時間 (收到請求的時間) 和結束時間 (傳送回應的時間)。
- 使用者代理程式 — `user-agent` 來自請求的。
- 「內容長度」— 響應`content-length`中的內容。

使用 Rails 中介軟體

若要使用中介軟體，請更新您的 gemfile 以包含必要的 railtie。

Example Gemfile - rails

```
gem 'aws-xray-sdk', require: ['aws-xray-sdk/facets/rails/railtie']
```

若要使用中介軟體，您還必須使用代表追蹤對映中應用程式的名稱來設定記錄器。

Example config/initializers/aws_xray.rb

```
Rails.application.config.xray = {
  name: 'my app'
}
```

手動檢測程式碼

若您沒有使用 Rails，請手動建立區段。您可以為每個傳入要求建立區段，或在已修補的 HTTP 或 AWS SDK 用戶端周圍建立區段，以提供錄製程式新增子區段的內容。

```
# Start a segment
segment = XRay.recorder.begin_segment 'my_service'
# Start a subsegment
subsegment = XRay.recorder.begin_subsegment 'outbound_call', namespace: 'remote'
```

```
# Add metadata or annotation here if necessary
my_annotations = {
  k1: 'v1',
  k2: 1024
}
segment.annotations.update my_annotations

# Add metadata to default namespace
subsegment.metadata[:k1] = 'v1'

# Set user for the segment (subsegment is not supported)
segment.user = 'my_name'

# End segment/subsegment
XRay.recorder.end_subsegment
XRay.recorder.end_segment
```

設定區段命名策略

AWS X-Ray 使用服務名稱來識別您的應用程式，並將其與應用程式使用的其他應用程式、資料庫、外部 API 和 AWS 資源區分開來。X-Ray SDK 為傳入要求產生區段時，會在區段的名稱[欄位中記錄應用程式的服務名稱](#)。

X-Ray SDK 可以在 HTTP 要求標頭中的主機名稱之後命名區段。但是，此標頭可能會被偽造，這可能會導致服務對應中出現非預期的節點。若要避免 SDK 因為具有偽造主機標頭的要求而不正確地命名區段，您必須為傳入要求指定預設名稱。

如果您的應用程式為多個網域提供要求，您可以將 SDK 設定為使用動態命名策略，在區段名稱中反映此問題。動態命名策略可讓 SDK 針對符合預期模式的要求使用主機名稱，並將預設名稱套用至不符合要求的要求。

例如，您可能有一個應用程式向三個子網域提供要求 — `www.example.com` `api.example.com`、和`static.example.com`。您可以將動態命名策略與模式搭配使用，`*.example.com`以識別具有不同名稱的每個子網域的區段，從而在服務對應上產生三個服務節點。如果您的應用程式收到的主機名稱與模式不符的要求，您會在服務對應上看到第四個節點，其中包含您指定的後援名稱。

若要為所有請求區段使用相同的名稱，請如[上一節](#)所述，在設定記錄器時指定您應用程式的名稱。

動態命名策略可定義主機名稱應相符的模式，以及如果 HTTP 請求中的主機名稱不符合模式時要使用的預設名稱。若要動態命名區段，請在組態雜湊中指定命名模式。

Example 主 .rb-動態命名

```
config = {
  naming_pattern: '*mydomain*',
  name: 'my app',
}

XRay.recorder.configure(config)
```

您可以在模式中使用 '*' 來比對任何字串，或是 '?' 來比對任何單一字元。

> **Note**
>
> 您可以使用 AWS_XRAY_TRACING_NAME 環境變數來覆寫您在程式碼中定義的預設服務名稱。

修補程式庫來檢測下游呼叫

若要測量下游呼叫，請使用 Ruby 的 X-Ray SDK 來修補應用程式所使用的程式庫。Ruby 的 X 射線 SDK 可以修補下列程式庫。

支援的程式庫

- net/http— 儀器 HTTP 客戶端。
- aws-sdk— 儀器AWS SDK for Ruby客戶端。

當您使用已修補的程式庫時，Ruby 的 X-Ray SDK 會建立呼叫的子區段，並記錄來自要求和回應的資訊。區段必須透過軟體開發套件中介軟體或呼叫 XRay.recorder.begin_segment 供軟體開發套件使用，以建立子區段。

若要修補程式庫，請在傳遞給 X 射線記錄器的組態物件中指定資源庫。

Example 主 .rb — 補丁庫

```
require 'aws-xray-sdk'

config = {
  name: 'my app',
  patch: %I[net_http aws_sdk]
}
```

```
XRay.recorder.configure(config)
```

使用紅寶石的 X-Ray AWS SDK 跟踪 SDK 調用

當您的應用程式呼叫 AWS 服務 以儲存資料、寫入佇列或傳送通知時，Ruby 的 X-Ray SDK 會追蹤子區段中下游的呼叫。您在這些服務中存取的追蹤 AWS 服務 和資源 (例如，Amazon S3 儲存貯體或 Amazon SQS 佇列) 會在 X-Ray 主控台的追蹤對應上顯示為下游節點。

Ruby 的 X-Ray SDK 會在您修補程式aws-sdk庫時自動檢測所有 AWS SDK 用戶端。您無法檢測個別用戶端。

對於所有服務，您都可以在 X-Ray 控制台中看到調用的 API 的名稱。對於服務子集，X-Ray SDK 會將資訊新增至區段，以在服務對應中提供更多精細度。

例如，當您使用已檢測的 DynamoDB 用戶端進行呼叫時，SDK 會將資料表名稱新增至區段，以便針對以資料表為目標的呼叫。在主控台中，每個表格在服務對應中顯示為獨立節點，其中包含一般 DynamoDB 節點，用於未針對資料表的呼叫。

Example 呼叫 DynamoDB 以儲存項目的子區段

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

您存取具名資源時，對以下服務的呼叫會在服務地圖中建立額外節點。未針對特定資源的呼叫，則會建立服務的一般節點。

- Amazon DynamoDB — 表名
- Amazon 簡單存儲服務 — 存儲桶和密鑰名稱
- Amazon 簡單隊列服務-隊列名稱

使用 X-Ray 開發套件產生自定義子區段

子段擴展追蹤的段，詳細介紹了為了服務請求而完成的工作。每次與分析客户端進行呼叫時，X-Ray SDK 都會記錄在子段中生成的信息。您可以創建其他子段來對其他子段進行分組、測量代碼部分的性能或記錄註釋和元數據。

若要管理子區段，請使用 begin_subsegment 和 end_subsegment 方法。

```
subsegment = XRay.recorder.begin_subsegment name: 'annotations', namespace: 'remote'
my_annotations = { id: 12345 }
subsegment.annotations.update my_annotations
XRay.recorder.end_subsegment
```

若要建立函數的子區段，請將它包裝在對 XRay.recorder.capture 進行的呼叫中。

```
XRay.recorder.capture('name_for_subsegment') do |subsegment|
  resp = myfunc() # myfunc is your function
  subsegment.annotations.update k1: 'v1'
  resp
end
```

當您在某區段或其他子區段內建立子區段時，則會為其產生一個 ID，並記錄開始時間和結束時間。

Example 使用中繼資料的子區段

```
"subsegments": [{
  "id": "6f1605cd8a07cb70",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "Custom subsegment for UserModel.saveUser function",
  "metadata": {
    "debug": {
      "test": "Metadata string from UserModel.saveUser"
    }
  },
```

使用 Ruby 專用的 X-Ray SDK 將註釋和中繼資料新增至區段

您可以使用註釋和中繼資料來記錄有關請求、環境或應用程式的其他資訊。您可以將註釋和中繼資料新增至 X-Ray SDK 建立的區段，或新增至您建立的自訂子區段。

註釋是與字符串，數字或布爾值鍵-值對。註釋會編製索引以與篩選器運算式搭配使用 使用標記記錄您想要用來在主控台將追蹤分組的資料，或是在呼叫 `GetTraceSummaries` API 時使用標記。

中繼資料是索引鍵-值配對，可以具有任何類型的值 (包括物件和清單)，但不會編製索引以供篩選運算式使用。使用元數據記錄要存儲在跟踪中但不需要與搜索一起使用的其他數據。

除了註釋和中繼資料，您也可以在區段上記錄使用者 ID 字串。區段會將使用者 ID 記錄在單獨的欄位中，並建立索引以用於搜尋。

章節

- 使用適用於 Ruby 的 X-Ray SDK 記錄註釋
- 使用適用於 Ruby 的 X-Ray SDK 記錄中繼資料
- 使用 Ruby 的 X-Ray SDK 記錄使用者 ID

使用適用於 Ruby 的 X-Ray SDK 記錄註釋

針對您想要建立索引以用於搜尋的區段或子區段，請使用標註來記錄這些區段上的資訊。

註釋要求

- 按鍵 — X-Ray 註解的金鑰最多可包含 500 個英數字元。您不能使用底線符號 (_) 以外的空格或符號。
- 值 — X-Ray 註釋的值最多可包含 1,000 個 Unicode 字元。
- 註釋的數量 — 每個追蹤最多可以使用 50 個註釋。

記錄標註

1. 從 `xray_recorder` 取得目前區段或子區段的參考。

   ```
   require 'aws-xray-sdk'
   ...
   document = XRay.recorder.current_segment
   ```

   或

```
require 'aws-xray-sdk'
...
document = XRay.recorder.current_subsegment
```

2. 使用雜湊值呼叫 update。

```
my_annotations = { id: 12345 }
document.annotations.update my_annotations
```

軟體開發套件會將標註以鍵/值對記錄在區段文件中的 annotations 物件內。若使用相同鍵呼叫 add_annotations 兩次，則會覆寫之前在相同區段或子區段上記錄的值。

若要尋找具有特定值之註釋的繪線，請在篩選器運算式中使用annotations.*key*關鍵字。

使用適用於 Ruby 的 X-Ray SDK 記錄中繼資料

針對您不想要建立索引以用於搜尋的區段，請使用中繼資料來記錄這些區段或子區段上的資訊。中繼資料值可以是字串、數字、布林值，或可序列化為 JSON 物件或陣列的任何物件。

記錄中繼資料

1. 從 xray_recorder 取得目前區段或子區段的參考。

```
require 'aws-xray-sdk'
...
document = XRay.recorder.current_segment
```

或

```
require 'aws-xray-sdk'
...
document = XRay.recorder.current_subsegment
```

2. 使用字串鍵、布林值、數字、字串或物件值，以及字串命名空間，呼叫 metadata。

```
my_metadata = {
  my_namespace: {
    key: 'value'
  }
}
```

```
subsegment.metadata my_metadata
```

若使用相同鍵呼叫 metadata 兩次，則會覆寫之前在相同區段或子區段上記錄的值。

使用 Ruby 的 X-Ray SDK 記錄使用者 ID

記錄請求區段上的使用者 ID 以識別傳送請求的使用者。

記錄使用者 ID

1. 從 xray_recorder 取得目前區段的參考。

```
require 'aws-xray-sdk'
...
document = XRay.recorder.current_segment
```

2. 將區段上的使用者欄位設為傳送請求的使用者字串 ID。

```
segment.user = 'U12345'
```

您可以在控制器中設定使用者，以在應用程式開始處理請求時馬上記錄使用者 ID。

若要尋找使用者 ID 的追蹤，請在篩選器運算式中使用user關鍵字。

AWS X-Ray 與其他整合 AWS 服務

許多 AWS 服務 提供不同層級的 X-Ray 整合，包括取樣和新增標頭至傳入的要求、執行 X-Ray 精靈，以及自動將追蹤資料傳送至 X-Ray。與 X-Ray 集成可以包括以下內容：

- 主動式儀器 — 樣本和儀器傳入請求
- 被動式儀器 — 已被其他服務取樣的儀器請求
- 要求追蹤 — 將追蹤標頭新增至所有傳入要求，並將其傳播至下游
- 工具 — 執行 X-Ray 精靈以接收來自 X-Ray SDK 的區段

> **Note**
>
> X-Ray SDK 包含用 AWS 服務於進行其他整合的外掛程式。例如，您可以使用適用於 Java Elastic Beanstalk 的 X-Ray SDK 來新增執行應用程式之 Elastic Beanstalk 環境的相關資訊，包括環境名稱和識別碼。

以下是一些與 X-Ray 集成的示例： AWS 服務

- AWS 適用於 ADOT 的發 OpenTelemetry 行版 — 借助 ADOT，工程師可以對其應用程序進行一次檢測，並將相關指標和跟踪發送到多個 AWS 監控解決方案，包括 Amazon CloudWatch AWS X-Ray，Amazon OpenSearch 服務和 Prometheus 的 Amazon 託管服務。
- AWS Lambda-所有運行時傳入請求的主動和被動儀器。 AWS Lambda 將兩個節點添加到跟踪映射中，一個用於 AWS Lambda 服務，另一個用於該功能。當您啟用檢測時， AWS Lambda 也會在 Java 和 Node.js 執行階段上執行 X-Ray 精靈，以與 X-Ray SDK 搭配使用。
- Amazon API Gateway — 主動和被動儀器。API Gateway 會使用取樣規則來決定要記錄哪些要求，並將閘道階段的節點新增至服務對應。
- AWS Elastic Beanstalk— 工具。Elastic Beanstalk 包括以下平台上的 X-Ray 守護進程：
  - Java SE-2.3.0 及更高版本的配置
  - 湯姆貓-2.4.0 及更高版本的配置
  - Node.js — 3.2.0 及更新版本的組態
  - 視窗服務器-除了 Windows 服務器核心以外的所有配置已經發布十二月九日, 2016

  您可以使用彈性 Beanstalk 控制台告訴 Elastic Beanstalk 在這些平台上運行守護程序，或使用命名空間中的`XRayEnabled`選項。`aws:elasticbeanstalk:xray`

- Elastic Load Balancing — 應用程式負載平衡器上的要求追蹤。應用程式負載平衡器會先將追蹤 ID 新增至要求標頭，再將追蹤 ID 傳送至目標群組。
- Amazon EventBridge-被動儀器。如果事件發佈至 EventBridge的服務已使用 X-Ray SDK 進行檢測，則事件目標將會收到追蹤標頭，並且可以繼續傳播原始追蹤識別碼。
- Amazon 簡單通知服務 — 被動儀器。如果 Amazon SNS 發行者使用 X-Ray SDK 追蹤其用戶端，訂閱者可以擷取追蹤標頭，並繼續使用相同追蹤 ID 從發行者傳播原始追蹤。
- Amazon 簡單隊列服務 — 被動儀器。如果服務使用 X-Ray SDK 追蹤請求，Amazon SQS 可以傳送追蹤標頭，並使用一致的追蹤 ID 繼續將原始追蹤從寄件者傳播到取用者。

請從下列主題中選擇，以探索整合的完整組合 AWS 服務。

主題

- AWS發行版OpenTelemetry和AWS X-Ray
- Amazon API Gateway 主動追蹤支援 AWS X-Ray
- Amazon EC2 和 AWS App Mesh
- AWS應用程序亞軍和 X 射
- AWS AppSync 與 AWS X-Ray
- 記錄 X-Ray API 呼叫 AWS CloudTrail
- CloudWatch 與 X-Ray 整合
- 跟踪 X-Ray 加密配置更改AWS Config
- Amazon Elastic Compute Cloud 和AWS X-Ray
- AWS Elastic Beanstalk 與 AWS X-Ray
- Elastic Load Balancing AWS X-Ray
- Amazon EventBridge 和 AWS X-Ray
- AWS Lambda 而且 AWS X-Ray
- Amazon SNS 和 AWS X-Ray
- AWS Step Functions 與 AWS X-Ray
- Amazon SQS 和 AWS X-Ray
- Amazon S3 和 AWS X-Ray

AWS發行版OpenTelemetry和AWS X-Ray

使用AWS發行版OpenTelemetry（ADOT）收集指標和跟踪並將其發送到AWS X-Ray和其他監控解決方案，例如亞馬遜CloudWatch, 亞馬遜OpenSearch適用於普羅米修斯的服務和亞馬遜託管服務。

AWS Distro for OpenTelemetry

該AWS發行版OpenTelemetry（亞多）是一個AWS以雲端原生運算基礎 (CNCF) 為基礎的散佈OpenTelemetry項目。OpenTelemetry提供一組開放原始碼 API、程式庫和代理程式，以收集分散式追蹤和指標。這個工具包是上游的發行版OpenTelemetry元件包括 SDK、自動檢測代理程式和收集器，這些元件經過測試、最佳化、保護和支援AWS。

使用 ADOT，工程師可以對其應用進行一次檢測，並將相關的指標和追蹤傳送給多個AWS監控解決方案，包括CloudWatch,AWS X-Ray, 亞馬遜OpenSearch適用於普羅米修斯的服務和亞馬遜託管服務。

ADOT 集成了越來越多的AWS 服務簡化發送跟踪和指標以監控解決方案，例如 X 射線。與 ADOT 整合的服務範例包括：

- AWS Lambda—AWS適用於 ADOT 的受管 Lambda 層可提供plug-and-play透過自動測試 Lambda 函數並封裝，進行使用者體驗OpenTelemetry連同out-of-the-box組態AWS Lambda和 X 射線在一個易於設置的層。使用者可以啟用和停用OpenTelemetry為他們的 Lambda 函數而不更改代碼。如需詳細資訊，請參閱AWS發行版OpenTelemetry拉姆達
- 亞馬遜彈性容器服務 (ECS)— 使用從亞馬遜 ECS 應用程序收集指標和跟踪AWS發行版OpenTelemetry收集器，發送到 X 射線和其他監測解決方案。如需詳細資訊，請參閱收集應用程式追蹤資在亞馬遜 ECS 開發人員指南。
- AWS應用亞軍— 應用程序運行器支持使用將跟踪發送到 X 射線AWS發行版OpenTelemetry(行為). 使用 ADOT SDK 為您的容器化應用程式收集追蹤資料，並使用 X-Ray 分析並深入瞭解您的儀器應用程式。如需詳細資訊，請參閱AWS應用程序亞軍和 X 射。

有關的更多信息AWS發行版OpenTelemetry，包括與其他整合AWS 服務，請參閱AWS發行版OpenTelemetry文件。

如需檢測您的應用程式的詳細資訊AWS發行版OpenTelemetry和 X 射線，請參閱檢測您的應用程式AWS發行版OpenTelemetry。

Amazon API Gateway 主動追蹤支援 AWS X-Ray

當使用者請求透過 Amazon API Gateway 送至基礎服務時，您可以使用 X-Ray 來追蹤和分析使用者請求。API Gateway 支援所有 API Gateway 端點類型的 X-Ray 追蹤：區域、邊緣最佳化和私有。您可以在所有可用 X-Ray 的 AWS 區域 地方使用 X-Ray 與 Amazon API Gateway。如需詳細資訊，請參閱 Amazon API Gateway 開發人員指南 AWS X-Ray中的追蹤 API 閘道 API 執行。

> **Note**
>
> X-Ray 僅支援透過 API Gateway 追蹤其餘 API。

Amazon API Gateway 提供的主動追蹤支援 AWS X-Ray。在 API 階段啟用主動追蹤，以取樣傳入的請求並將追蹤傳送至 X-Ray。

啟用 API 階段的主動追蹤

1. 在以下網址開啟 API Gateway 主控台：https://console.aws.amazon.com/apigateway/。
2. 選擇一個 API。
3. 選擇一個階段。
4. 在「記錄/追蹤」標籤上，選擇「啟用 X-Ray 追蹤」，然後選擇「儲存變更」。
5. 在左側導覽窗格中，選擇 Resources (資源)。
6. 若要使用新設定重新部署 API，請選擇 [動作] 下拉式清單，然後選擇 [部署 API]。

API Gateway 會使用您在 X-Ray 主控台中定義的取樣規則來決定要記錄的要求。您可以建立僅套用至 API 的規則，或僅套用至包含特定標頭的要求。API Gateway 會在區段的屬性中記錄標頭，以及有關階段和要求的詳細資訊。如需詳細資訊，請參閱 設定 取樣規則。

> **Note**
>
> 使用 API Gateway HTTP 整合追蹤 REST API 時，會將每個區段的服務名稱設定為從 API Gateway 到 HTTP 整合端點的要求 URL 路徑，從而在每個唯一 URL 路徑的 X-Ray 追蹤對應上產生一個服務節點。大量 URL 路徑可能會導致追蹤對應超過 10,000 個節點的限制，進而導致錯誤。

若要盡量減少 API Gateway 建立的服務節點數目，請考慮在 URL 查詢字串內或透過 POST 在要求內文中傳遞參數。這兩種方法都可以確保參數不是 URL 路徑的一部分，這可能會導致較少的不同 URL 路徑和服務節點。

對於所有傳入的請求，API Gateway 會將追蹤標頭新增至還沒有的連入 HTTP 要求。

```
X-Amzn-Trace-Id: Root=1-5759e988-bd862e3fe1be46a994272793
```

X-Ray 軌跡 ID 格式

X-Ray `trace_id` 由三個用連字符分隔的數字組成。例如 `1-58406520-a006649127e371903a2de979`。其中包含：

- 版本號碼，也就是1。
- 原始請求的時間在 Unix 紀元時間使用 8 個十六進制數字。

  例如，2016 年 12 月 1 日上午 10:00 PST (以紀元時間表示) 為`1480615200`秒或十六進`58406520`位數字。
- 追蹤的全域唯一 96 位元識別碼，以 24 個十六進位數字顯示。

即使停用主動追蹤，如果請求是來自已抽樣請求並已開始追蹤的服務，階段仍會記錄區段。例如，經過檢測的 Web 應用程式可以使用 HTTP 用戶端呼叫 API Gateway API。當您使用 X-Ray SDK 檢測 HTTP 用戶端時，它會在包含取樣決策的傳出要求中新增追蹤標頭。API Gateway 讀取追蹤標頭，並為取樣要求建立區段。

如果您使用 API Gateway 為您的 API 產生 Java SDK，您可以透過在用戶端建置器中新增要求處理常式來檢測 SDK 用戶端，方法與您手動檢測 AWS SDK 用戶端的方式相同。如需說明，請參閱 使用適用於 Java 的 X-Ray AWS SDK 追蹤 SDK 呼叫。

Amazon EC2 和 AWS App Mesh

AWS X-Ray 與整合AWS App Mesh以管理微服務的特使代理。App Mesh 提供了一個版本的 Envoy，您可以將其配置為將跟踪數據發送到在相同任務或網繭的容器中運行的 X-Ray 守護程序。X-Ray 支持跟踪與以下 App Mesh 兼容的服務：

- Amazon Elastic Container Service (Amazon ECS)

- Amazon Elastic Kubernetes Service (Amazon EKS)
- Amazon Elastic Compute Cloud (Amazon EC2)

使用以下指示，了解如何透過 App Mesh 來啟用 X-Ray 追蹤。

若要設定 Envoy 代理伺服器以將資料傳送至 X-Ray，請在其容器定義中設定`ENABLE_ENVOY_XRAY_TRACING`環境變數。

> **Note**
>
> Envoy 的 App Mesh 版本目前不會根據配置的取樣規則傳送追蹤。取而代之的是，特使 1.16.3 或更新版本使用 5% 的固定取樣率，而使用 1.16.3 之前的特使版本採樣率為 50%。

Example Amazon ECS 的特使容器定義

```
{
      "name": "envoy",
      "image": "public.ecr.aws/appmesh/aws-appmesh-envoy:envoy-version",
      "essential": true,
      "environment": [
        {
          "name": "APPMESH_VIRTUAL_NODE_NAME",
          "value": "mesh/myMesh/virtualNode/myNode"
        },
        {
          "name": "ENABLE_ENVOY_XRAY_TRACING",
          "value": "1"
        }
      ],
      "healthCheck": {
        "command": [
          "CMD-SHELL",
          "curl -s http://localhost:9901/server_info | cut -d' ' -f3 | grep -q live"
        ],
        "startPeriod": 10,
        "interval": 5,
        "timeout": 2,
        "retries": 3
      }
```

> **Note**
>
> 要了解有關可用的特使地區地址的更多信息，請參閱 AWS App Mesh 用戶指南中的 Envoy 圖像。

如需在容器中執行 X-Ray 精靈的詳細資訊，請參閱X-Ray Amazon。對於包含服務網格、微服務、Envoy Proxy 和 X-Ray 精靈的範例應用程式，請在 App Mesh 範例儲存庫中部署範例 GitHub。`colorapp`

進一步了解

- 開始使用 AWS App Mesh
- 開始使用 AWS App Mesh 和 Amazon ECS

AWS應用程序亞軍和 X 射

AWS應用程序亞軍是一個AWS 服務提供快速、簡單且符合成本效益的方式，可從原始程式碼或容器映像直接部署至可擴充且安全的 Web 應用程式AWS 雲端。您不需要學習新技術、決定要使用哪種運算服務，也無需了解如何佈建和設定 AWS。請參閱什麼是AWS應用亞軍以取得更多資訊。

AWS應用程序運行器通過與集成將跟踪發送到 X 射線AWS發行版OpenTelemetry(行為). 使用 ADOT SDK 為您的容器化應用程式收集追蹤資料，並使用 X-Ray 分析並深入瞭解您的儀器應用程式。如需詳細資訊，請參閱使用 X 射線追蹤您的應用程式執行程式。

AWS AppSync 與 AWS X-Ray

您可以啟用並追蹤AWSAppSync。如需詳細資訊，請參閱「」追蹤AWSX-Ray如需指示。

X-RayAWSAppSync API，一個AWSIdentity and Access Management服務連結角色會利用適當的許可，在您的帳戶中自動建立服務。這可讓AWSAppSync 以安全的方式將追蹤傳送到 X-Ray。

記錄 X-Ray API 呼叫 AWS CloudTrail

AWS X-Ray 與AWS CloudTrail提供使用者、角色或 AWS 服務. CloudTrail 將 X-Ray 的所有 API 呼叫擷取為事件。擷取的呼叫包括來自 X-Ray 主控台的呼叫，以及對 X-Ray API 作業的程式碼呼叫。使用收集的資訊 CloudTrail，您可以判斷向 X-Ray 發出的要求、提出要求的 IP 位址、提出要求的時間，以及其他詳細資訊。

每一筆事件或日誌項目都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是使用根使用者還是使用者憑證提出。
- 請求是否代表 IAM 身分中心使用者提出。

- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 該請求是否由另一項 AWS 服務服務提出。

CloudTrail 在您創建帳戶 AWS 帳戶 時處於活動狀態，並且您自動可以訪問 CloudTrail 事件歷史記錄。 CloudTrail 事件歷史記錄提供了過去 90 天的記錄管理事件的可查看，可搜索，可下載和不可變的記錄。 AWS 區域若要取得更多資訊，請參閱《使用指南》中的〈AWS CloudTrail 使用 CloudTrail 事件歷程〉。查看活動歷史記錄不 CloudTrail收取任何費用。

如需過 AWS 帳戶 去 90 天內持續的事件記錄，請建立追蹤或 CloudTrailLake 事件資料存放區。

CloudTrail 小徑

追蹤可 CloudTrail 將日誌檔交付到 Amazon S3 儲存貯體。使用建立的所有系統線 AWS Management Console 都是多區域。您可以使用建立單一區域或多區域系統線。 AWS CLI建議您建立多區域追蹤，因為您會擷取帳戶 AWS 區域 中的所有活動。如果您建立單一區域追蹤，則只能檢視追蹤記錄中的 AWS 區域事件。如需有關追蹤的詳細資訊，請參閱《AWS CloudTrail 使用指南》中的「為您的建立追蹤」 AWS 帳戶和「為組織建立追蹤」。

您可以透 CloudTrail 過建立追蹤，免費將一份正在進行的管理事件副本傳遞到 Amazon S3 儲存貯體，但是需要支付 Amazon S3 儲存費用。如需有關 CloudTrail 定價的詳細資訊，請參閱AWS CloudTrail 定價。如需 Amazon S3 定價的相關資訊，請參閱 Amazon S3 定價。

CloudTrail 湖泊事件資料存放區

CloudTrail L@@ ake 可讓您針對事件執行 SQL 型查詢。 CloudTrail 湖將基於行的 JSON 格式的現有事件轉換為 Apache ORC 格式。ORC 是一種單欄式儲存格式，針對快速擷取資料進行了最佳化。系統會將事件彙總到事件資料存放區中，事件資料存放區是事件的不可變集合，其依據為您透過套用進階事件選取器選取的條件。套用於事件資料存放區的選取器控制哪些事件持續存在並可供您查詢。若要取得有關 CloudTrail Lake 的更多資訊，請參閱使用指南中的〈AWS CloudTrail 使用 AWS CloudTrail Lake 〉。

CloudTrail Lake 事件資料存放區和查詢會產生費用。建立事件資料存放區時，您可以選擇要用於事件資料存放區的定價選項。此定價選項將決定擷取和儲存事件的成本，以及事件資料存放區的預設和最長保留期。如需有關 CloudTrail 定價的詳細資訊，請參閱AWS CloudTrail 定價。

主題

- X-Ray 管理事件 CloudTrail
- X-Ray 資料事件 CloudTrail

- X-Ray 事件範例

X-Ray 管理事件 CloudTrail

AWS X-Ray 與整合 AWS CloudTrail 以記錄使用者、角色或 X-Ray AWS 服務 中所做的 API 動作。您可以使 CloudTrail 用即時監控 X-Ray API 請求，並將日誌存放在 Amazon S3、Amazon CloudWatch 日誌和 Amazon CloudWatch 事件中。X-Ray 支援將下列動作記錄為記 CloudTrail 錄檔中的事件：

支援的 API 動作

- PutEncryptionConfig
- GetEncryptionConfig
- CreateGroup
- UpdateGroup
- DeleteGroup
- GetGroup
- GetGroups
- GetInsight
- GetInsightEvents
- GetInsightImpactGraph
- GetInsightSummaries
- GetSamplingStatisticSummaries

X-Ray 資料事件 CloudTrail

資料事件提供在資源上或在資源中執行的資源作業的相關資訊 (例如 PutTraceSegments，這些作業會將區段文件上載至 X-Ray)。

這些也稱為資料平面操作。資料事件通常是大量資料的活動。依預設， CloudTrail 不會記錄資料事件。 CloudTrail 事件歷史記錄不會記錄數據事件。

資料事件需支付額外的費用。如需有關 CloudTrail 定價的詳細資訊，請參閱AWS CloudTrail 定價。

您可以使用 CloudTrail 主控台或 CloudTrail API 作業記錄 X-Ray 資源類型的資料事件。 AWS CLI有關如何記錄資料事件的詳細資訊，請參閱AWS CloudTrail 使用《使用指南》 AWS Command Line Interface中的記錄資料事件 AWS Management Console和記錄資料事件。

下表列出您可以記錄其資料事件的 X-Ray 資源類型。[資料事件類型 (主控台)] 欄顯示可從主控台的 [資料事件類型 CloudTrail] 清單中選擇的值。resource .type 值欄會顯示**`resources.type`**值，您可以在使用或 API 設定進階事件選取器時指定這個值。 AWS CLI CloudTrail 記錄到資料 CloudTrail欄中的資料 API 會顯示 CloudTrail 針對資源類型記錄的 API 呼叫。

資料事件類型 (主控台)	resources.type 值	記錄到的資料 API CloudTrail
X-Ray 軌跡	`AWS::XRay::Trace`	• PutTraceSegments • GetTraceSummaries • GetTraceGraph • GetServiceGraph • BatchGetTraces • GetTimeSeriesServiceStatistics • PutTelemetryRecords • GetSamplingTargets

您可以設定進階事件選取器來篩選`eventName`和`readOnly`欄位，以僅記錄對您很重要的事件。但是，您無法透過新增`resources.ARN`欄位選取器來選取事件，因為 X-Ray 軌跡沒有 ARN。如需這些欄位的詳細資訊，請參閱 AWS CloudTrail API 參考AdvancedFieldSelector中的。以下是如何執行`put-event-selectors` AWS CLI 命令以記錄 CloudTrail 追蹤上的資料事件的範例。您必須在中執行指令，或指定在其中建立軌跡的「區域」；否則，作業會傳回`InvalidHomeRegionException`例外狀況。

```
aws cloudtrail put-event-selectors --trail-name myTrail --advanced-event-selectors \
'{
   "AdvancedEventSelectors": [
      {
         "FieldSelectors": [
            { "Field": "eventCategory", "Equals": ["Data"] },
            { "Field": "resources.type", "Equals": ["AWS::XRay::Trace"] },
            { "Field": "eventName", "Equals":
 ["PutTraceSegments","GetSamplingTargets"] }
         ],
         "Name": "Log X-Ray PutTraceSegments and GetSamplingTargets data events"
      }
   ]
```

```
}'
```

X-Ray 事件範例

管理事件的例子，**GetEncryptionConfig**

以下是中 X-Ray GetEncryptionConfig 記錄項目的範例 CloudTrail。

Example

```
{
    "eventVersion"=>"1.05",
    "userIdentity"=>{
        "type"=>"AssumedRole",
        "principalId"=>"AROAJVHBZWD3DN6CI2MHM:MyName",
        "arn"=>"arn:aws:sts::123456789012:assumed-role/MyRole/MyName",
        "accountId"=>"123456789012",
        "accessKeyId"=>"AKIAIOSFODNN7EXAMPLE",
        "sessionContext"=>{
            "attributes"=>{
                "mfaAuthenticated"=>"false",
                "creationDate"=>"2023-7-01T00:24:36Z"
            },
            "sessionIssuer"=>{
                "type"=>"Role",
                "principalId"=>"AROAJVHBZWD3DN6CI2MHM",
                "arn"=>"arn:aws:iam::123456789012:role/MyRole",
                "accountId"=>"123456789012",
                "userName"=>"MyRole"
            }
        }
    },
    "eventTime"=>"2023-7-01T00:24:36Z",
    "eventSource"=>"xray.amazonaws.com",
    "eventName"=>"GetEncryptionConfig",
    "awsRegion"=>"us-east-2",
    "sourceIPAddress"=>"33.255.33.255",
    "userAgent"=>"aws-sdk-ruby2/2.11.19 ruby/2.3.1 x86_64-linux",
    "requestParameters"=>nil,
    "responseElements"=>nil,
    "requestID"=>"3fda699a-32e7-4c20-37af-edc2be5acbdb",
    "eventID"=>"039c3d45-6baa-11e3-2f3e-e5a036343c9f",
    "eventType"=>"AwsApiCall",
```

```
    "recipientAccountId"=>"123456789012"
}
```

資料事件範例，**PutTraceSegments**

以下是中 X-Ray PutTraceSegments 資料事件記錄項目的範例 CloudTrail。

Example

```
{
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAWYXPW54Y4NEXAMPLE:i-0dzz2ac111c83zz0z",
    "arn": "arn:aws:sts::012345678910:assumed-role/my-service-role/
i-0dzz2ac111c83zz0z",
    "accountId": "012345678910",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAWYXPW54Y4NEXAMPLE",
        "arn": "arn:aws:iam::012345678910:role/service-role/my-service-role",
        "accountId": "012345678910",
        "userName": "my-service-role"
      },
      "attributes": {
        "creationDate": "2024-01-22T17:34:11Z",
        "mfaAuthenticated": "false"
      },
      "ec2RoleDelivery": "2.0"
    }
  },
  "eventTime": "2024-01-22T18:22:05Z",
  "eventSource": "xray.amazonaws.com",
  "eventName": "PutTraceSegments",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "198.51.100.0",
  "userAgent": "aws-sdk-ruby3/3.190.0 md/internal ua/2.0 api/xray#1.0.0 os/linux md/
x86_64 lang/ruby#2.7.8 md/2.7.8 cfg/retry-mode#legacy",
  "requestParameters": {
    "traceSegmentDocuments": [
      "trace_id:1-00zzz24z-EXAMPLE4f4e41754c77d0000",
```

```
      "trace_id:1-00zzz24z-EXAMPLE4f4e41754c77d0000",
      "trace_id:1-00zzz24z-EXAMPLE4f4e41754c77d0001",
      "trace_id:1-00zzz24z-EXAMPLE4f4e41754c77d0002"
    ]
  },
  "responseElements": {
    "unprocessedTraceSegments": []
  },
  "requestID": "5zzzzz64-acbd-46ff-z544-451a3ebcb2f8",
  "eventID": "4zz51z7z-77f9-44zz-9bd7-6c8327740f2e",
  "readOnly": false,
  "resources": [
    {
      "type": "AWS::XRay::Trace"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": false,
  "recipientAccountId": "012345678910",
  "eventCategory": "Data",
  "tlsDetails": {
    "tlsVersion": "TLSv1.2",
    "cipherSuite": "ZZZZZ-RSA-AAA128-GCM-SHA256",
    "clientProvidedHostHeader": "example.us-west-2.xray.cloudwatch.aws.dev"
  }
}
```

CloudWatch 與 X-Ray 整合

AWS X-Ray 與「應用CloudWatch 程式訊號」、「 CloudWatch RUM」和「 CloudWatch Synthetics」整合，讓您更輕鬆地監控應用程式的健康狀態。讓應用程式信號的應用程式能夠監視和疑難排解服務、用戶端頁面、Synthetics 金花園和服務相依性的作業健康狀態。

透過將 CloudWatch 指標、記錄和 X-Ray 追蹤相關聯，X-Ray 追蹤地圖可提供服務 end-to-end 檢視，協助您快速找出效能瓶頸並識別受影響的使用者。

使用 CloudWatch RUM，您可以執行真實的使用者監視，以近乎即時的方式從實際使用者工作階段收集和檢視 Web 應用程式效能的用戶端資料。使用 AWS X-Ray 和 CloudWatch RUM，您可以從應用程式的最終使用者開始到下游 AWS 受管理的服務，分析和偵錯要求路徑。這可協助您識別影響使用者的延遲趨勢和錯誤。

主題

CloudWatch 朗姆酒和 AWS X-Ray

使用 Amazon CloudWatch RUM，您可以執行真實的使用者監控，以近乎即時的方式從實際使用者工作階段收集和檢視 Web 應用程式效能的用戶端資料。使用 AWS X-Ray 和 CloudWatch RUM，您可以從應用程式的最終使用者開始，透過下游 AWS 受管理的服務來分析和偵錯要求路徑。這可協助您識別影響使用者的延遲趨勢和錯誤。

在您開啟使用者工作階段的 X-Ray 追蹤之後，R CloudWatch UM 會將 X-Ray 追蹤標頭新增至允許的 HTTP 要求，並記錄允許的 HTTP 要求的 X-Ray 區段。然後，您可以在 X-Ray 和 CloudWatch 主控台 (包括 X-Ray 追蹤圖) 中查看這些使用者工作階段的追蹤和區段。

> **Note**
>
> CloudWatch RUM 不會與 X-Ray 取樣規則整合。而是在將應用程式設定為使用 CloudWatch RUM 時，請選擇取樣百分比。從 CloudWatch RUM 傳送的追蹤可能會產生額外費用。如需詳細資訊，請參閱 AWS X-Ray 定價。

默認情況下，從 CloudWatch RUM 發送的客戶端跟踪不連接到服務器端跟踪。若要將用戶端追蹤與伺服器端追蹤連線，請設定 R CloudWatch UM Web 用戶端，將 X-Ray 追蹤標頭新增至這些 HTTP 要求。

> **Warning**
>
> 設定 R CloudWatch UM 網路用戶端以將 X-Ray 追蹤標頭新增至 HTTP 要求，可能會導致跨來源資源共用 (CORS) 失敗。為了避免這種情況，請將 `X-Amzn-Trace-Id` HTTP 標頭添加到下游服務的 CORS 配置上允許的標頭列表中。如果您使用 API Gateway 作為下游，請參閱為 REST API 資源啟用 CORS。強烈建議您在生產環境中新增用戶端 X-Ray 追蹤標頭之前，先測試您的應用程式。如需詳細資訊，請參閱 CloudWatch RUM 網路用戶端文件。

如需有關中的實際使用者監視的詳細資訊 CloudWatch，請參閱使用 CloudWatch RUM。若要設定應用程式使用 R CloudWatch UM，包括使用 X-Ray 追蹤使用者工作階段，請參閱設定應用程式以使用 R CloudWatch UM。

使用 X-R CloudWatch ay 調試合成纖維金絲雀

CloudWatch Synthetics 是一項全受管服務，可讓您使用每天 24 小時執行一次的指令碼加那群監視端點和 API。

您可以自訂 Canary 指令碼，以檢查下列項目中的變更：

- 可用性
- Latency (延遲)
- 交易
- 中斷或失效的連結
- S tep-by-step 任務完成
- 頁面載入錯誤
- UI 資產的載入延遲
- 複雜的精靈流程
- 應用程式中的簽出流程

Canary 會沿著相同的路線，執行與客戶相同的動作和行為，持續驗證客戶的體驗。

若要深入了解如何設定 Synthetics 測試，請參閱使用 Synthetics 建立及管理 Canary。

下列範例會示範您 Synthetics Canary 引起的偵錯問題常見使用案例。每個範例都會示範使用追蹤對應或 X-Ray Analytics 主控台進行偵錯的關鍵策略。

如需如何讀取追蹤對應並與之互動的詳細資訊，請參閱檢視服務對應。

如需如何讀取 X-Ray 分析主控台並與之互動的詳細資訊，請參閱與 AWS X-Ray 分析主控台互動。

主題

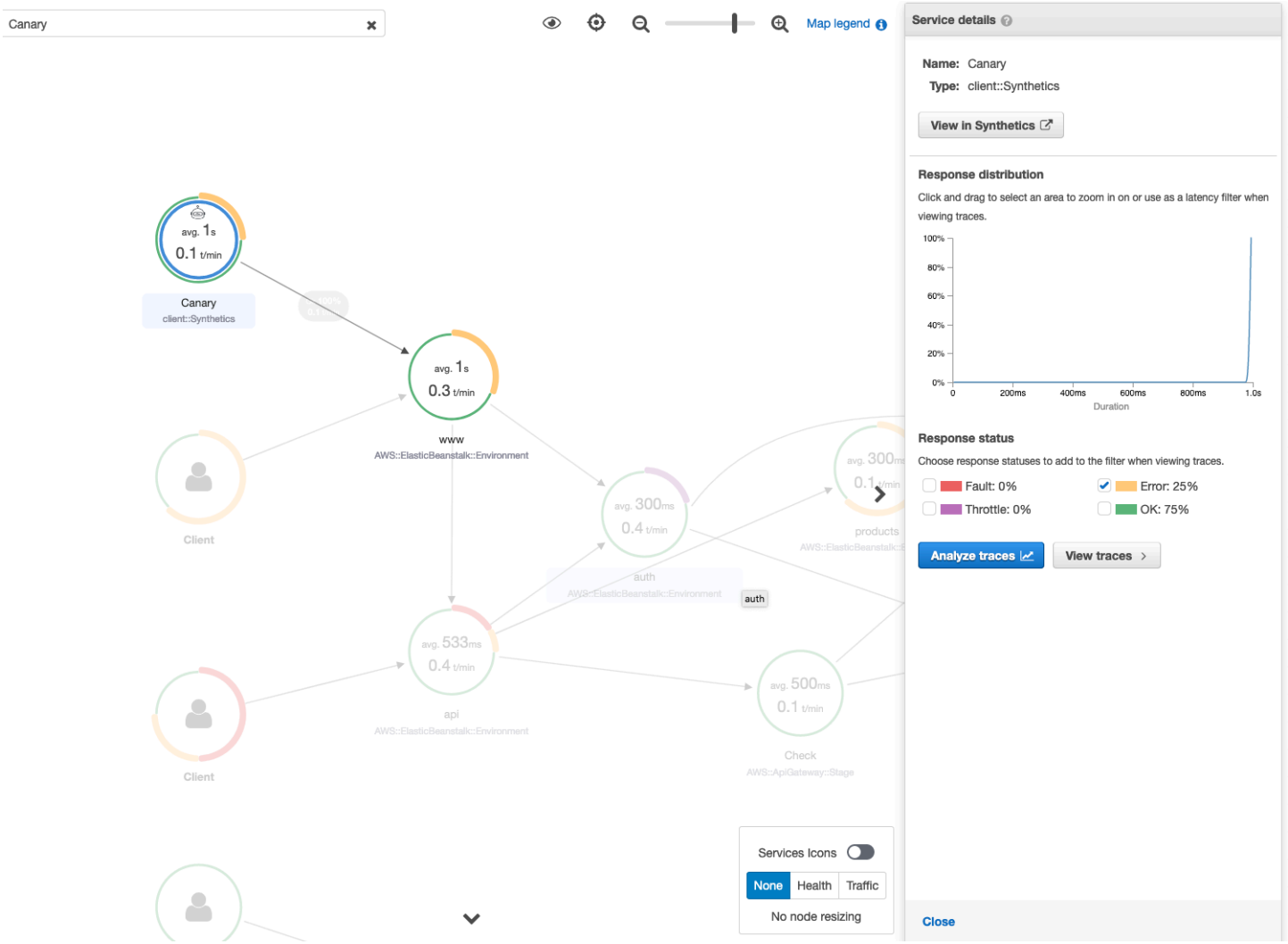
- 在跟踪圖中查看錯誤報告增加的金絲雀
- 針對個別追蹤使用追蹤詳細資訊對應，以詳細檢視每個要求
- 判斷上游和下游服務中持續性失敗的根本原因
- 識別效能瓶頸和趨勢

- 比較變更前後的延遲及錯誤率或容錯率
- 決定所有 API 和 URL 的必要 Canary 涵蓋範圍
- 使用群組專注於 Synthetics 測試

在跟踪圖中查看錯誤報告增加的金絲雀

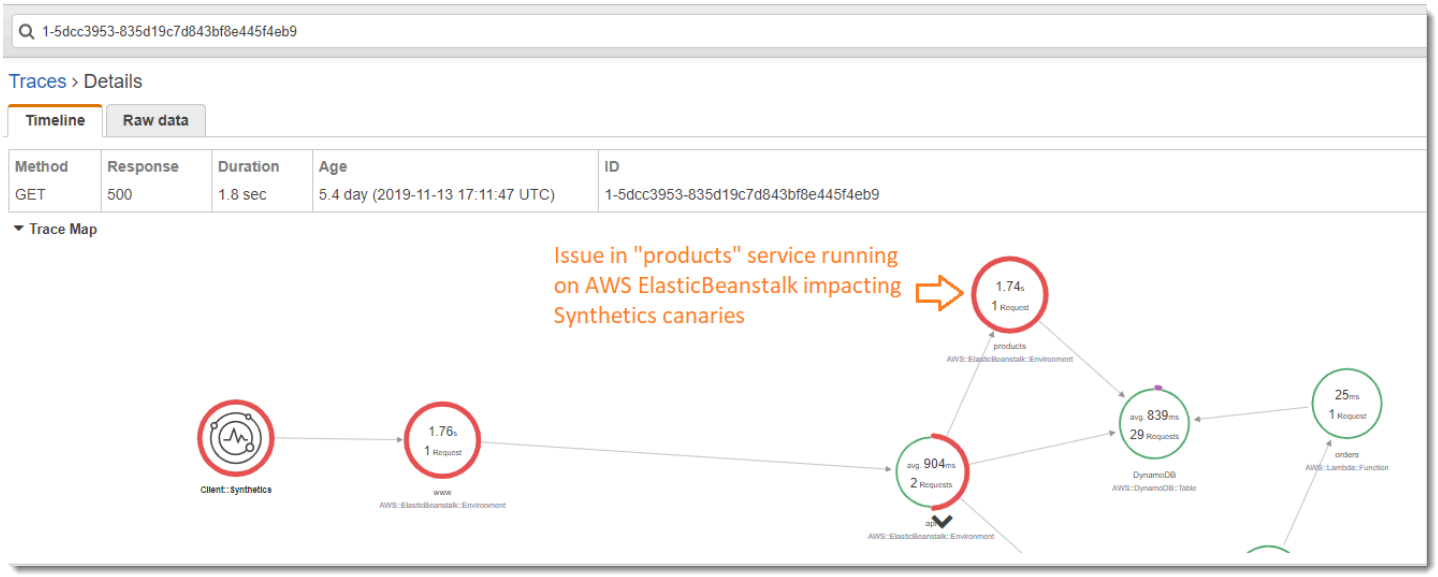
要查看哪些金絲雀在 X-Ray 跟踪地圖中錯誤，故障，節流率或緩慢的響應時間增加，您可以使用過濾器突出顯示 Synthetics 測試用戶端節點。`Client::Synthetic`按一下節點可顯示整個要求的回應時間分佈。按一下兩個節點之間的邊緣會顯示有關傳送該連線之要求的詳細資料。您也可以在追蹤對應中檢視相關下游服務的「遠端」推斷節點。

當您單擊 Synthetics 節點時，側面板上有一個在 Synthetics 中查看按鈕，可將您重定向到 Synthetics 控制台，您可以在其中檢查金絲雀詳細信息。



針對個別追蹤使用追蹤詳細資訊對應，以詳細檢視每個要求

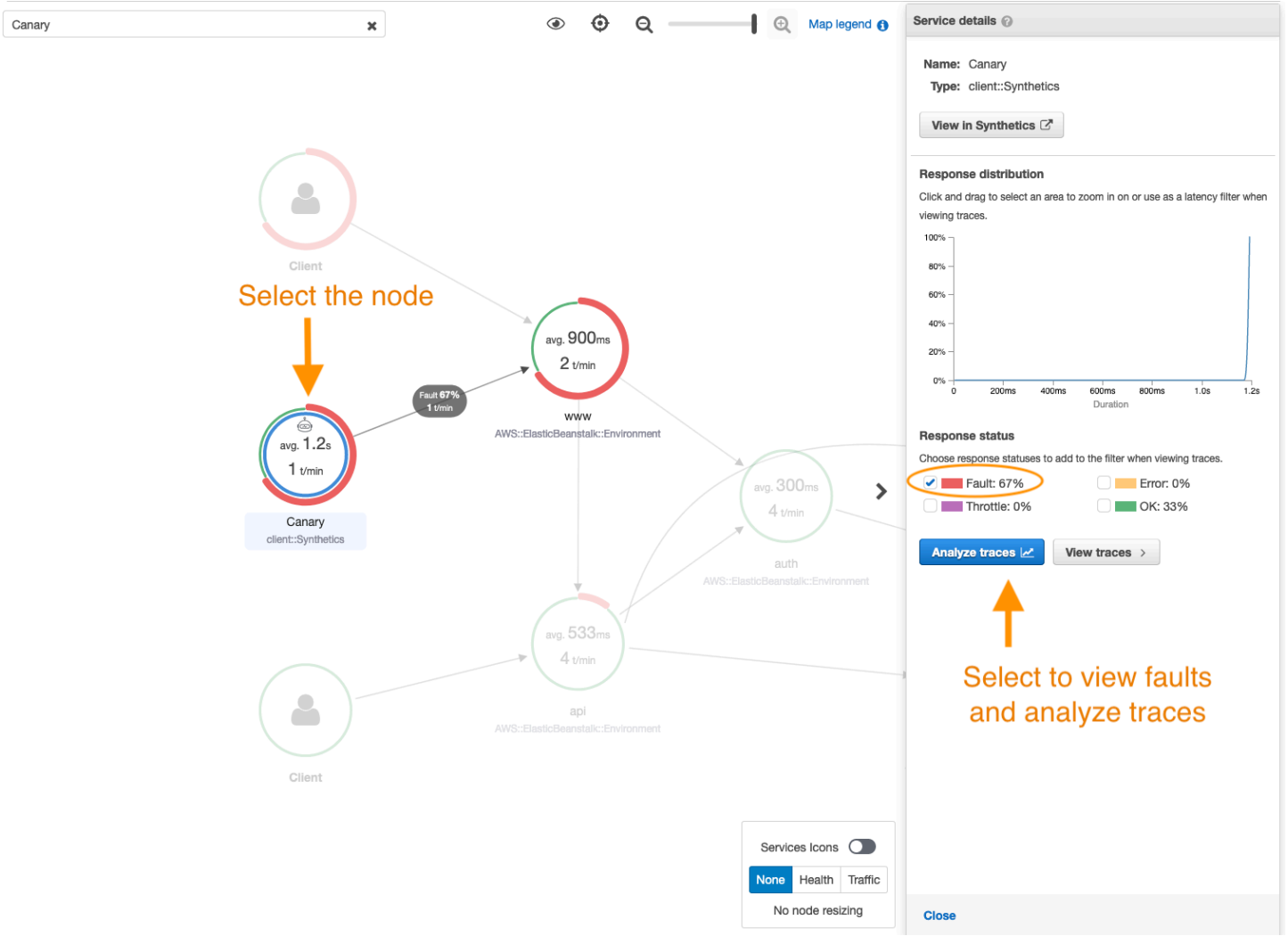
若要判斷哪個服務會造成最多延遲或造成錯誤，請在追蹤對映中選取追蹤來呼叫追蹤詳細資料對應。個別追蹤詳細資料對應會顯示單一要求的 end-to-end 路徑。使用此項目可了解呼叫的服務，並視覺化上游和下游服務。

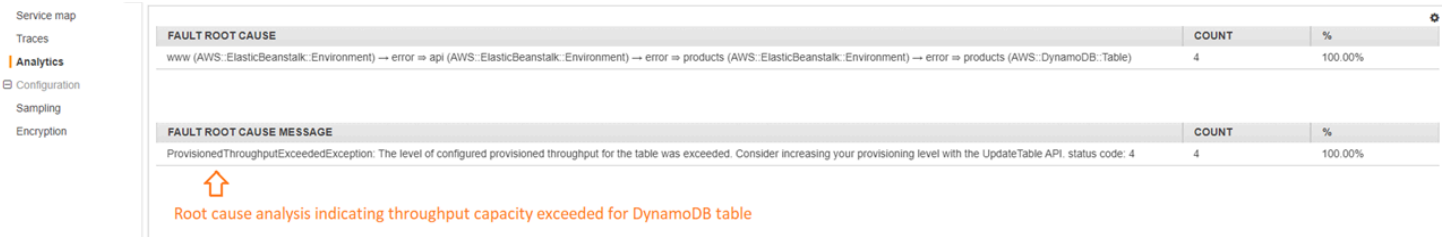


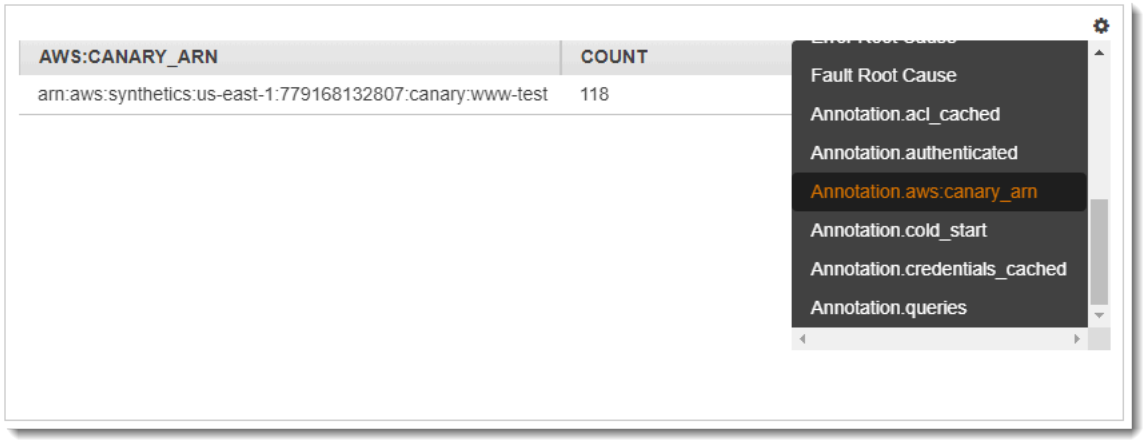
判斷上游和下游服務中持續性失敗的根本原因

當您收到 Synthetics 料初期測試失敗的 CloudWatch 警示後，請使用 X-Ray 中追蹤資料的統計模型，在 X-Ray Analytics 主控台中判斷問題的可能根本原因。在 Analytics 主控台中，「回應時間根本原因」表格會顯示記錄的實體路徑。X-Ray 確定軌跡中的哪個路徑最有可能導致回應時間的原因。格式指出實體遇到的階層，以回應時間根本原因結束。

下列範例顯示在 API Gateway 上執行之 API「XXX」的 Synthetics 測試因 Amazon DynamoDB 表格中的輸送量容量例外而失敗。

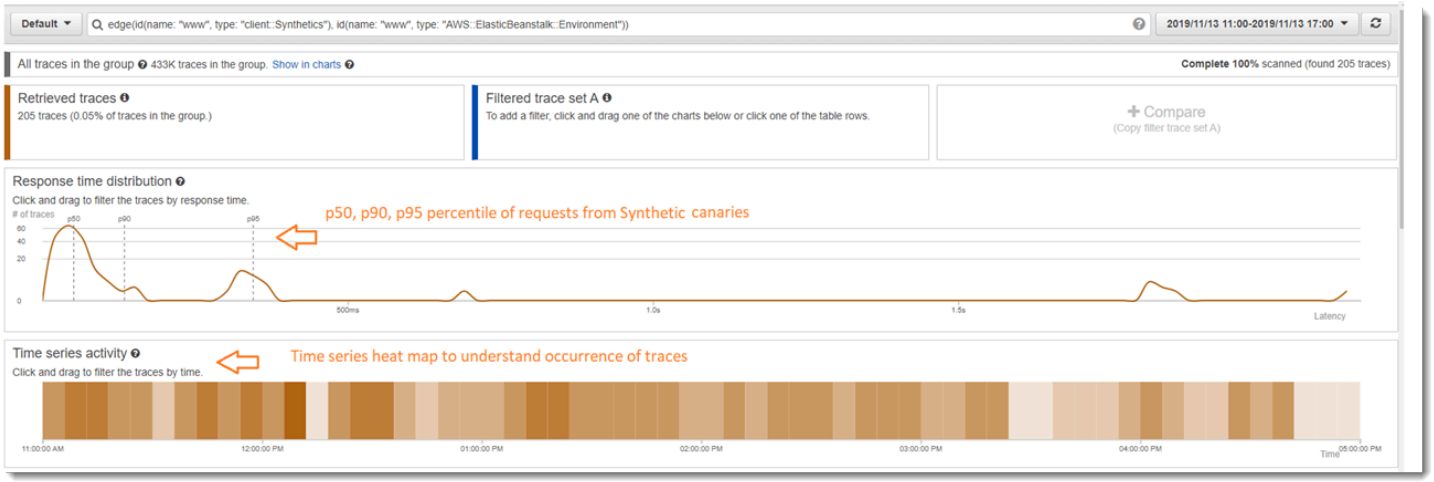

Canary
Map legend
Service details
Name: Canary
Type: client::Synthetics
View in Synthetics
Response distribution
Click and drag to select an area to zoom in on or use as a latency filter when viewing traces.
100%
80%
60%
40%
20%
0%
0
200ms
400ms
600ms
800ms
1.0s
1.2s
Duration
Response status
Choose response statuses to add to the filter when viewing traces.
Fault: 67%
Error: 0%
Throttle: 0%
OK: 33%
Analyze traces
View traces
Select to view faults and analyze traces
Client
Select the node
avg. 900ms
2 t/min
Fault 67%
1 t/min
www
AWS::ElasticBeanstalk::Environment
avg. 1.2s
1 t/min
Canary
client::Synthetics
avg. 300ms
4 t/min
auth
AWS::ElasticBeanstalk::Environment
avg. 533ms
4 t/min
api
AWS::ElasticBeanstalk::Environment
Client
Services Icons
None
Health
Traffic
No node resizing
Close


Service map
Traces
Analytics
Configuration
Sampling
Encryption
FAULT ROOT CAUSE
COUNT
%
www (AWS::ElasticBeanstalk::Environment) ⟶ error ⇒ api (AWS::ElasticBeanstalk::Environment) ⟶ error ⇒ products (AWS::ElasticBeanstalk::Environment) ⟶ error ⇒ products (AWS::DynamoDB::Table)
4
100.00%
FAULT ROOT CAUSE MESSAGE
COUNT
%
ProvisionedThroughputExceededException: The level of configured provisioned throughput for the table was exceeded. Consider increasing your provisioning level with the UpdateTable API. status code: 4
4
100.00%
Root cause analysis indicating throughput capacity exceeded for DynamoDB table



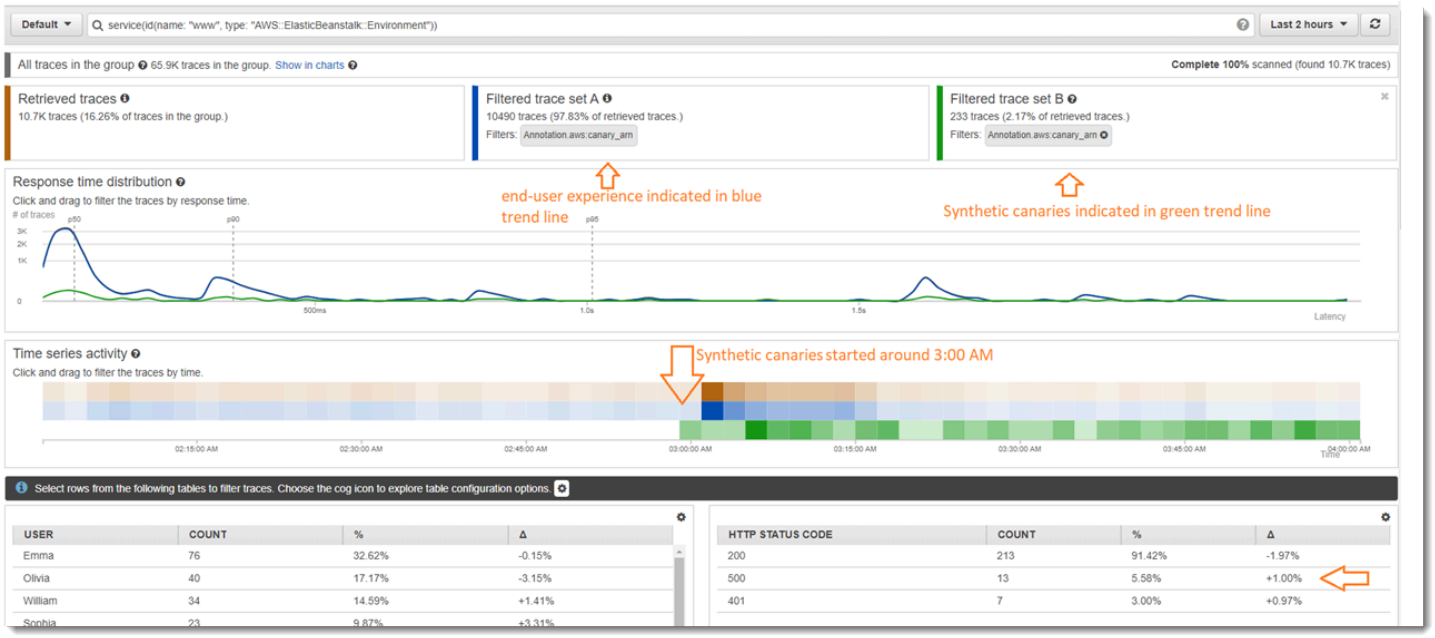
識別效能瓶頸和趨勢

您可以使用來自 Synthetics Canary 的連續流量來檢視端點隨時間的效能趨勢，以填入一段時間內的追蹤詳細資料對映。



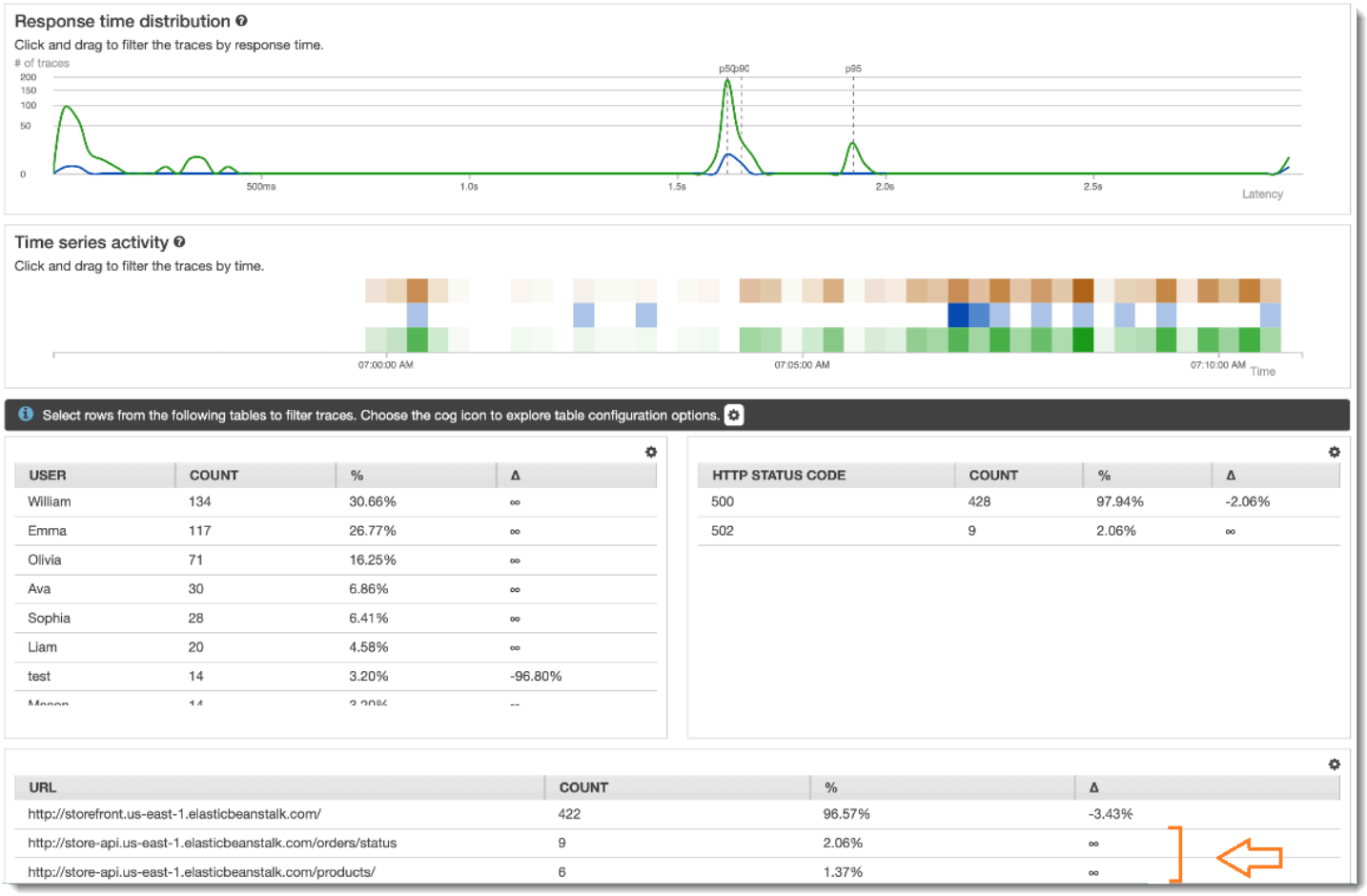
比較變更前後的延遲及錯誤率或容錯率

Pinpoint 出發生變更的時間，將該變更與加那群加那利群島所發生的問題增加相關聯。使用 X-Ray Analytics 主控台將前後時間範圍定義為不同的追蹤集，從而在回應時間分佈中建立視覺差異化。



決定所有 API 和 URL 的必要 Canary 涵蓋範圍

利用 X-Ray Analytics 比較 Canary 和使用者的經歷。下面的 UI 中，藍色趨勢線表示 Canary，綠色趨勢線代表使用者。您也可以看到三個 URL 中有兩個沒有 Canary 測試。

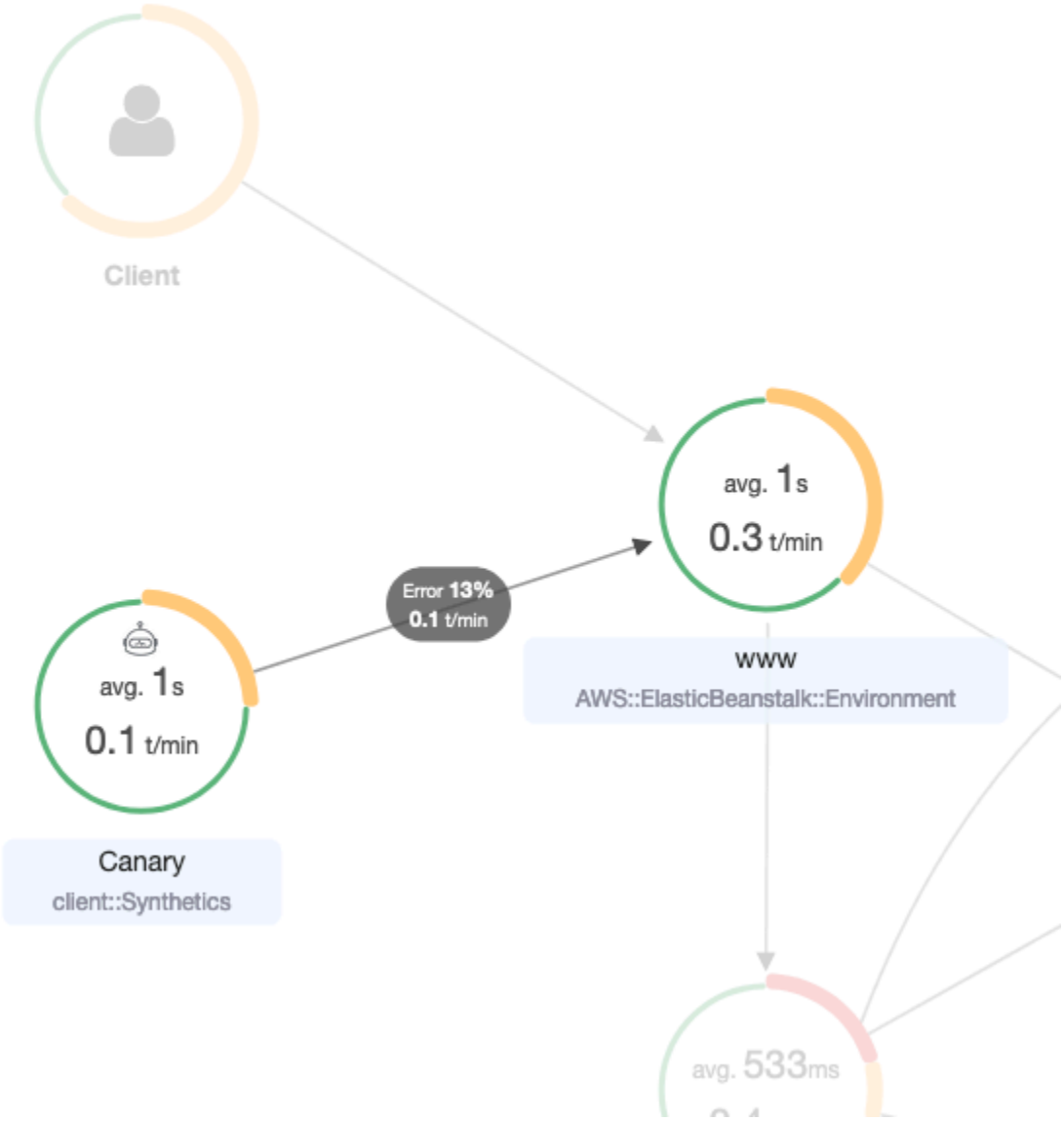


使用群組專注於 Synthetics 測試

您可以使用濾鏡運算式建立 X-Ray 群組，以專注於特定的工作流程集，例如執行於應用程式「www」的 Synthetics 測試。 AWS Elastic Beanstalk使用複雜的關鍵字，`service()`並`edge()`通過服務和邊緣進行過濾。

Example 群組篩選條件表達式

```
"edge(id(name: "www", type: "client::Synthetics"), id(name: "www", type:
 "AWS::ElasticBeanstalk::Environment"))"
```



跟踪 X-Ray 加密配置更改AWS Config

AWS X-Ray與整合AWS Config以記錄對 X-Ray 加密資源所做的組態變更。您可以用AWS Config來清查 X-Ray 加密資源、稽核 X-Ray 組態歷程記錄，以及根據資源變更傳送通知。

AWS Config支援將下列 X-Ray 加密資源變更記錄為事件：

- 組態變更 — 變更或新增加密金鑰，或回復為預設的 X-Ray 加密設定。

使用下列指示來瞭解如何在 X-Ray 和之間建立基本連接AWS Config。

建立 Lambda 函數觸發

您必須先具備自訂 AWS Lambda 函數的 ARN，之後才能產生自訂 AWS Config 規則。按照這些指示，使用 Node.js 來建立基本函數，以根據 `XrayEncryptionConfig` 資源的狀態將合規或未合規的值傳回給 AWS Config。

若要使用 AWS::XrayEncryptionConfig 變更觸發器建立 Lambda 函數

1. 開啟 Lambda 主控台。選擇 Create function (建立函數)。
2. 選擇藍圖，然後篩選藍圖程式庫以取得config-rule-change-triggered藍圖。按一下藍圖名稱中的連結，或選擇 Configure (設定) 以繼續。
3. 定義下列欄位以設定藍圖：
   - 針對 Name (名稱)，輸入名稱。
   - 在 Role (角色) 中，選擇 Create new role from template(s) (從範本建立新角色)。
   - 在 Role name (角色名稱) 中，輸入名稱。
   - 針對 Policy templates (政策範本)，選擇AWS Config Rules permissions (&CC; 規則許可)。
4. 選擇 Create function (建立函數) 以在 AWS Lambda 主控台中建立並顯示您的函數。
5. 編輯您的函數程式碼，將 `AWS::EC2::Instance` 取代為 `AWS::XrayEncryptionConfig`。您也可以更新描述欄位，以反映這個變更。

   預設程式碼

```
    if (configurationItem.resourceType !== 'AWS::EC2::Instance') {
        return 'NOT_APPLICABLE';
    } else if (ruleParameters.desiredInstanceType ===
configurationItem.configuration.instanceType) {
        return 'COMPLIANT';
    }
        return 'NON_COMPLIANT';
```

   更新的程式碼

```
    if (configurationItem.resourceType !== 'AWS::XRay::EncryptionConfig') {
        return 'NOT_APPLICABLE';
    } else if (ruleParameters.desiredInstanceType ===
configurationItem.configuration.instanceType) {
        return 'COMPLIANT';
    }
```

```
        return 'NON_COMPLIANT';
```

6. 將以下內容添加到 IAM 中的執行角色以訪問 X-Ray。這些許 X-Ray。未能提供適當資源的存取權，將導致在評估與規則關聯的 Lambda 函數AWS Config時產生超出範圍的訊息。

```
{
    "Sid": "Stmt1529350291539",
    "Action": [
        "xray:GetEncryptionConfig"
    ],
    "Effect": "Allow",
    "Resource": "*"
 }
```

建立適用於 X-Ray 的自訂 AWS Config 規則

建立 Lambda 函數時，請記下函數的 ARN，然後前往主AWS Config控台建立自訂規則。

建立 X-RayAWS Config 規則

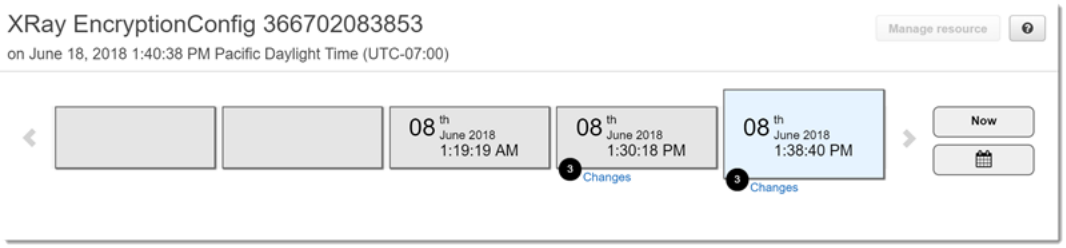
1. 開啟主AWS Config控台的 [規則] 頁面。
2. 選擇 Add rule (新增規則)，然後選擇 Add custom rule (新增自訂規則)。
3. 在AWS Lambda函數 ARN 中，插入與您要使用的 Lambda 函數相關聯的 ARN。
4. 選擇要設定的觸發類型：
   - 配置變更 — 當配置中符合規則範圍的任何資源變更時AWS Config觸發評估。AWS Config 傳送組態項目變更通知之後，即會執行評估。
   - 定期 — 以您選擇的頻率AWS Config執行規則評估 (例如，每 24 小時)。
5. 對於「資源類型」，EncryptionConfig在「X-Ray」區段中選擇。
6. 選擇 Save (儲存)。

AWS Config 主控台會立即開始評估規則的合規性。系統需要幾分鐘的時間來完成評估。

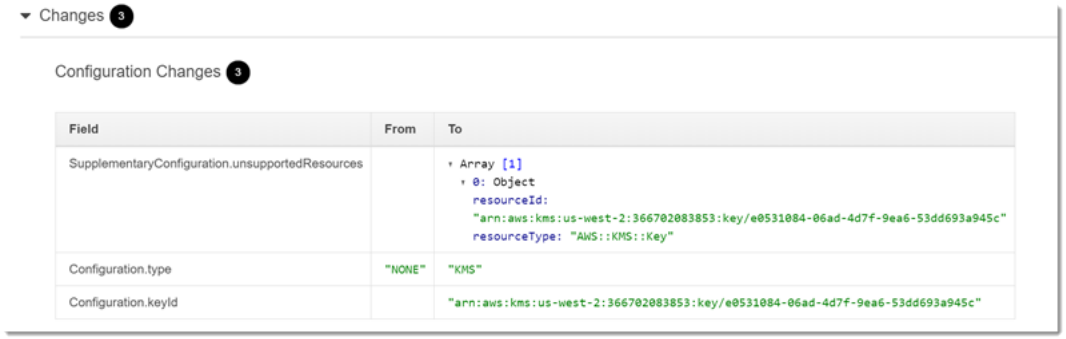
現在此規則為合規，AWS Config 即可開始編譯稽核歷史記錄。AWS Config 會以時間軸的形式記錄資源的變更。針對事件時間軸中的每個變更，AWS Config 會產生一份變更前/後的資料表，以顯示加密金鑰的 JSON 表示法中有哪些變更。與相關的兩個欄位變更 EncryptionConfig 為`Configuration.type`和`Configuration.keyID`。

範例結果

以下是 AWS Config 時間軸的範例，其顯示特定日期和時間所做的變更。



下列為 AWS Config 變更項目的範例。變更前/後格式可說明有哪些變更。此範例顯示預設 X-Ray 加密設定已變更為已定義的加密金鑰。



Amazon SNS 通知

如需組態變更的通知，請將 Amazon SNS 通知設定AWS Config為。如需詳細資訊，請參閱透過電子郵件監控 AWS Config 資源變更。

Amazon Elastic Compute Cloud 和AWS X-Ray

您可以使用使用者資 X-Ray 指令碼，在 Amazon EC2 執行個體上執行協助程式。如需說明，請參閱在 Amazon EC2 上執行 X-Ray 常設程式。

使用執行個體配置文件授予協助程式將追蹤資料上傳至 X-Ray 的許可。如需詳細資訊，請參閱授予守護進程將數據發送到 X-Ray 的權限。

AWS Elastic Beanstalk 與 AWS X-Ray

AWS Elastic Beanstalk平台包括 X-Ray 常設程式。您可以執行精靈通過在 Elastic Beanstalk 控台或使用配置文件設定選項。

在 Java SE 平台中，您可以使用 Buildfile 檔案搭配執行個體上的 Maven 或 Gradle 來建置應用程式。適用於 Java 的 X-Ray 開發套件和AWS SDK for JavaMaven 提供，因此您可以僅部署應用程式的程式碼並在執行個體上建置，以避免綁定與上傳所有相依性。

您可以使用 Elastic Beanstalk 環境屬性來設定 X-Ray 開發套件。Elastic Beanstalk 用來將環境屬性傳遞給應用程式的方法依平台而異。依據您的平台，使用 X-Ray 開發套件的環境變數或系統屬性。

- Node.js 平台— 使用環境變數
- Java SE 平台— 使用環境變數
- Tomcat 平台— 使用系統屬性

如需詳細資訊，請參閱「」設定AWS X-Ray除錯中的AWS Elastic Beanstalk開發人員指南。

Elastic Load Balancing AWS X-Ray

Elastic Load Balancing 應用程式負載平衡器會將追蹤識別碼新增至名為`X-Amzn-Trace-Id`的標頭中的傳入 HTTP 要求。

```
X-Amzn-Trace-Id: Root=1-5759e988-bd862e3fe1be46a994272793
```

X-Ray 軌跡 ID 格式

X-Ray `trace_id` 由三個用連字符分隔的數字組成。例如 `1-58406520-a006649127e371903a2de979`。其中包含：

- 版本號碼，也就是`1`。
- 原始請求的時間在 Unix 紀元時間使用 8 個十六進制數字。

  例如，2016 年 12 月 1 日上午 10:00 PST (以紀元時間表示) 為`1480615200`秒或十六進`58406520`位數字。
- 追蹤的全域唯一 96 位元識別碼，以 24 個十六進位數字顯示。

負載平衡器不會將資料傳送至 X-Ray，也不會顯示為服務對應上的節點。

如需詳細資訊，請參閱 Elastic Load Balancing 開發人員指南中的應用程式負載平衡器的要求追蹤。

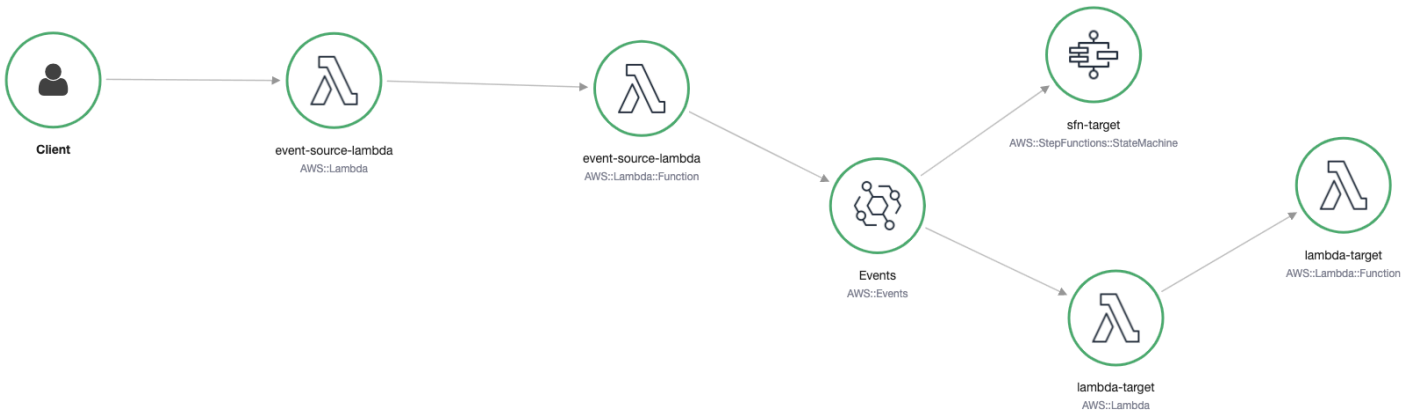
Amazon EventBridge 和 AWS X-Ray

AWS X-Ray 與 Amazon 集成 EventBridge 以跟踪傳遞的事件 EventBridge。如果使用 X-Ray SDK 檢測的服務將事件傳送至 EventBridge，追蹤前後關聯會傳播至追蹤標頭內的下游事件目標。X-Ray SDK 會自動拾取追蹤標頭，並將其套用至任何後續檢測。這種連續性可讓使用者在整個下游服務中追蹤、分析和偵錯，並提供更完整的系統檢視。

若要取得更多資訊，請參閱EventBridge 使用指南中的 EventBridge X-Ray 整合。

檢視 X-Ray 服務圖上的來源和目標

X-Ray 追蹤對映會顯示連接來源和目標服務的 EventBridge 事件節點，如下列範例所示：



將追蹤內容傳播至事件目標

X-Ray SDK 可讓 EventBridge 事件來源將追蹤內容傳播至下游事件目標。下列語言特定範例示範 EventBridge 從啟用主動追蹤的 Lambda 函數呼叫：

Java

為 X-Ray 添加必要的依賴關係：

- AWS X-Ray 適用於 Java 的開發套件
- AWS X-Ray 適用於 Java 的記錄器 SDK

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
```

```
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.services.eventbridge.AmazonEventBridge;
import com.amazonaws.services.eventbridge.AmazonEventBridgeClientBuilder;
import com.amazonaws.services.eventbridge.model.PutEventsRequest;
import com.amazonaws.services.eventbridge.model.PutEventsRequestEntry;
import com.amazonaws.services.eventbridge.model.PutEventsResult;
import com.amazonaws.services.eventbridge.model.PutEventsResultEntry;
import com.amazonaws.xray.handlers.TracingHandler;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.lang.StringBuilder;
import java.util.Map;
import java.util.List;
import java.util.Date;
import java.util.Collections;

/*
   Add the necessary dependencies for XRay:
   https://mvnrepository.com/artifact/com.amazonaws/aws-java-sdk-xray
   https://mvnrepository.com/artifact/com.amazonaws/aws-xray-recorder-sdk-aws-sdk
*/
public class Handler implements RequestHandler<SQSEvent, String>{
  private static final Logger logger = LoggerFactory.getLogger(Handler.class);

  /*
    build EventBridge client
  */
  private static final AmazonEventBridge eventsClient =
 AmazonEventBridgeClientBuilder
          .standard()
          // instrument the EventBridge client with the XRay Tracing Handler.
          // the AWSXRay globalRecorder will retrieve the tracing-context
          // from the lambda function and inject it into the HTTP header.
          // be sure to enable 'active tracing' on the lambda function.
          .withRequestHandlers(new TracingHandler(AWSXRay.getGlobalRecorder()))
          .build();

  @Override
  public String handleRequest(SQSEvent event, Context context)
  {
    PutEventsRequestEntry putEventsRequestEntry0 = new PutEventsRequestEntry();
```

```
    putEventsRequestEntry0.setTime(new Date());
    putEventsRequestEntry0.setSource("my-lambda-function");
    putEventsRequestEntry0.setDetailType("my-lambda-event");
    putEventsRequestEntry0.setDetail("{\"lambda-source\":\"sqs\"}");
    PutEventsRequest putEventsRequest = new PutEventsRequest();
    putEventsRequest.setEntries(Collections.singletonList(putEventsRequestEntry0));
    // send the event(s) to EventBridge
    PutEventsResult putEventsResult = eventsClient.putEvents(putEventsRequest);
    try {
      logger.info("Put Events Result: {}", putEventsResult);
    } catch(Exception e) {
      e.getStackTrace();
    }
    return "success";
  }
}
```

Python

將以下依賴項添加到您的 requirements.txt 文件中：

```
aws-xray-sdk==2.4.3
```

```
import boto3
from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch_all

# apply the XRay handler to all clients.
patch_all()

client = boto3.client('events')

def lambda_handler(event, context):
    response = client.put_events(
        Entries=[
            {
                'Source': 'foo',
                'DetailType': 'foo',
                'Detail': '{\"foo\": \"foo\"}'
            },
        ]
    )
```

```
    return response
```

Go

```
package main

import (
  "context"
  "github.com/aws/aws-lambda-go/lambda"
  "github.com/aws/aws-lambda-go/events"
  "github.com/aws/aws-sdk-go/aws/session"
  "github.com/aws/aws-xray-sdk-go/xray"
  "github.com/aws/aws-sdk-go/service/eventbridge"
  "fmt"
)

var client = eventbridge.New(session.New())


func main() {
 //Wrap the eventbridge client in the AWS XRay tracer
  xray.AWS(client.Client)
  lambda.Start(handleRequest)
}

func handleRequest(ctx context.Context, event events.SQSEvent) (string, error) {
  _, err := callEventBridge(ctx)
  if err != nil {
    return "ERROR", err
  }
  return "success", nil
}


func callEventBridge(ctx context.Context) (string, error) {
    entries := make([]*eventbridge.PutEventsRequestEntry, 1)
    detail := "{ \"foo\": \"foo\"}"
    detailType := "foo"
    source := "foo"
    entries[0] = &eventbridge.PutEventsRequestEntry{
        Detail: &detail,
        DetailType: &detailType,
        Source: &source,
```

```
    }

  input := &eventbridge.PutEventsInput{
     Entries: entries,
  }

  // Example sending a request using the PutEventsRequest method.
  resp, err := client.PutEventsWithContext(ctx, input)

  success := "yes"
  if err == nil { // resp is now filled
      success = "no"
      fmt.Println(resp)
  }
  return success, err
}
```

Node.js

```
const AWSXRay = require('aws-xray-sdk')
//Wrap the aws-sdk client in the AWS XRay tracer
const AWS = AWSXRay.captureAWS(require('aws-sdk'))
const eventBridge = new AWS.EventBridge()

exports.handler = async (event) => {

  let myDetail = { "name": "Alice" }

  const myEvent = {
    Entries: [{
      Detail: JSON.stringify({ myDetail }),
      DetailType: 'myDetailType',
      Source: 'myApplication',
      Time: new Date
    }]
  }

  // Send to EventBridge
  const result = await eventBridge.putEvents(myEvent).promise()

  // Log the result
  console.log('Result: ', JSON.stringify(result, null, 2))
```

```
}
```

C#

將以下 X-Ray 軟件包添加到 C # 依賴項中：

```
<PackageReference Include="AWSXRayRecorder.Core" Version="2.6.2" />
<PackageReference Include="AWSXRayRecorder.Handlers.AwsSdk" Version="2.7.2" />
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Amazon;
using Amazon.Util;
using Amazon.Lambda;
using Amazon.Lambda.Model;
using Amazon.Lambda.Core;
using Amazon.EventBridge;
using Amazon.EventBridge.Model;
using Amazon.Lambda.SQSEvents;
using Amazon.XRay.Recorder.Core;
using Amazon.XRay.Recorder.Handlers.AwsSdk;
using Newtonsoft.Json;
using Newtonsoft.Json.Serialization;

[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.Json.JsonSerializer))]

namespace blankCsharp
{
  public class Function
  {
    private static AmazonEventBridgeClient eventClient;

    static Function() {
      initialize();
    }

    static async void initialize() {
      //Wrap the AWS SDK clients in the AWS XRay tracer
      AWSSDKHandler.RegisterXRayForAllServices();
      eventClient = new AmazonEventBridgeClient();
```

```
    }

    public async Task<PutEventsResponse> FunctionHandler(SQSEvent invocationEvent,
 ILambdaContext context)
    {
      PutEventsResponse response;
      try
      {
        response = await callEventBridge();
      }
      catch (AmazonLambdaException ex)
      {
        throw ex;
      }

      return response;
    }

    public static async Task<PutEventsResponse> callEventBridge()
    {
      var request = new PutEventsRequest();
      var entry = new PutEventsRequestEntry();
      entry.DetailType = "foo";
      entry.Source = "foo";
      entry.Detail = "{\"instance_id\":\"A\"}";
      List<PutEventsRequestEntry> entries = new List<PutEventsRequestEntry>();
      entries.Add(entry);
      request.Entries = entries;
      var response = await eventClient.PutEventsAsync(request);
      return response;
    }
  }
}
```

AWS Lambda 而且 AWS X-Ray

您可以使用 AWS X-Ray 來跟踪您的 AWS Lambda 功能。Lambda 會執行 X-Ray 精靈，並記錄一個區段，其中包含有關叫用和執行函數的詳細資訊。對於進一步的檢測，您可以將 X-Ray SDK 與您的功能捆綁在一起，以記錄撥出電話並添加註釋和元數據。

如果您的 Lambda 函數是由另一個已檢測的服務呼叫，Lambda 會追蹤已經取樣的請求，而不需要任何其他設定。上游服務可以是已測試的 Web 應用程式或其他 Lambda 函數。您的服務可以使用已檢測的 AWS SDK 用戶端直接叫用函數，或透過使用已檢測的 HTTP 用戶端呼叫 API Gateway API。

AWS X-Ray 支援使用 AWS Lambda 和 Amazon SQS 追蹤事件導向的應用程式。使用主 CloudWatch 控台查看每個請求與 Amazon SQS 排入佇列並由下游 Lambda 函數進行處理時的連線檢視。來自上游訊息生產者的追蹤會自動連結至來自下游 Lambda 消費者節點的追蹤，以建立應用程式的 end-to-end 檢視。如需詳細資訊，請參閱追蹤事件導向應用程式。

> **Note**
>
> 如果您已啟用下游 Lambda 函數的追蹤，則還必須啟用根 Lambda 函數的追蹤，以呼叫下游函數，以便下游函數產生追蹤。

如果您的 Lambda 函數按排程執行，或是由未經測試的服務叫用，您可以設定 Lambda 使用作用中追蹤來取樣和記錄叫用。

若要在 AWS Lambda 功能上設定 X-Ray 整合

1. 開啟 AWS Lambda 主控台。
2. 從左側導覽列選取「功能」。
3. 選擇函數。
4. 在 [設定] 索引標籤上，向下捲動至 [其他監視工具] 卡。您還可以通過選擇左側導航窗格中的監視和操作工具來找到此卡。
5. 選擇 Edit (編輯)。
6. 在 AWS X-Ray 下，啟用 Active tracing (主動追蹤)。

在具有對應 X-Ray SDK 的執行階段中，Lambda 也會執行 X-Ray 精靈。

Lambda 上的 X-Ray 軟件包

- 適用於 Go 的 X-Ray SDK — 移至 1.7 及更新版本的執行階段
- 適用於 Java 的 X-Ray SDK-Java 8 運行時
- 適用於 Node.js — Node.js 4.3 和更新的執行階段的 X-Ray 開發套件
- 適用於 Python 的 X-Ray 開發套件 — Python 2.7、Python 3.6 和更新的執行階段
- 適用於 .NET 核心 2.0 和更新的執行階段的 X-Ray SDK

若要在 Lambda 上使用 X-Ray SDK，請在每次建立新版本時將其與函數程式碼搭配使用。您可以使用與測量在其他服務上執行的應用程式相同的方法來檢測 Lambda 函數。主要差別是，您無法使用軟體開發套件來檢測傳入的請求、制定抽樣決策及建立區段。

檢測 Lambda 函數和 Web 應用程式之間的另一個區別是，您的函數程式碼無法修改 Lambda 建立並傳送至 X-Ray 的區段。您可以建立子區段並記錄註釋和中繼資料，但您無法新增註釋和中繼資料到父區段。

若要取得更多資訊，請參閱AWS Lambda 開發人員指南中的使用 AWS X-Ray

Amazon SNS 和 AWS X-Ray

您可以 AWS X-Ray 搭配 Amazon Simple Notification Service (Amazon SNS) 使用，在請求透過 SNS 主題傳送至受 SNS 支援的訂閱服務時，追蹤和分析請求。搭配 Amazon SNS 使用 X-Ray 追蹤來分析訊息及其後端服務中的延遲情況，例如請求在一個主題中花費多長時間，以及將訊息傳遞到每個主題的訂閱所需的時間。Amazon SNS 支援標準和 FIFO 主題的 X-Ray 追蹤。

如果您從已使用 X-Ray 檢測的服務發佈到 Amazon SNS 主題，Amazon SNS 會將追蹤內容從發佈者傳送給訂閱者。此外，您可以開啟作用中追蹤，將 Amazon SNS 訂閱的區段資料傳送至 X-Ray，以便從檢測到的 SNS 用戶端發佈的訊息。使用 Amazon SNS 主控台或使用 Amazon SNS API 或 CLI，開啟 Amazon SNS 主題的使用中追蹤功能。如需檢測 SNS 用戶端的詳細資訊，請參閱檢測您的應用程式。

設定 Amazon SNS 作用中追蹤

您可以使用 Amazon SNS 主控台或 AWS CLI 或開發套件來設定 Amazon SNS 作用中追蹤。

當您使用 Amazon SNS 主控台時，Amazon SNS 會嘗試建立必要的許可，讓 SNS 呼叫 X-Ray。如果您沒有足夠的權限修改 X-Ray 資源策略，則可以拒絕嘗試。如需有關這些許可的詳細資訊，請參閱 Amazon SNS 中的身分識別和存取管理和 Amazon SNS 存取控制範例案例 (英文)，在 Amazon 簡單通知服務開發人員指南。如需有關使用 Amazon SNS 主控台開啟作用中追蹤的詳細資訊，請參閱 Amazon 簡單通知服務開發人員指南中的啟用 Amazon SNS 上的主動追蹤主題。

使用 AWS CLI 或 SDK 開啟使用中追蹤時，您必須使用以資源為基礎的原則手動設定權限。用於使用`PutResourcePolicy`必要的資源型政策設定 X-Ray，以允許 Amazon SNS 將追蹤傳送至 X-Ray。

Example 適用於 Amazon SNS 主動追蹤的 X-Ray 資源型政策範例

此範例政策文件指定 Amazon SNS 將追蹤資料傳送至 X-Ray 所需的許可：

```
{
    Version: "2012-10-17",
    Statement: [
      {
        Sid: "SNSAccess",
        Effect: Allow,
        Principal: {
          Service: "sns.amazonaws.com",
        },
        Action: [
          "xray:PutTraceSegments",
          "xray:GetSamplingRules",
          "xray:GetSamplingTargets"
        ],
        Resource: "*",
        Condition: {
          StringEquals: {
            "aws:SourceAccount": "account-id"
          },
          StringLike: {
            "aws:SourceArn": "arn:partition:sns:region:account-id:topic-name"
          }
        }
      }
    ]
  }
```

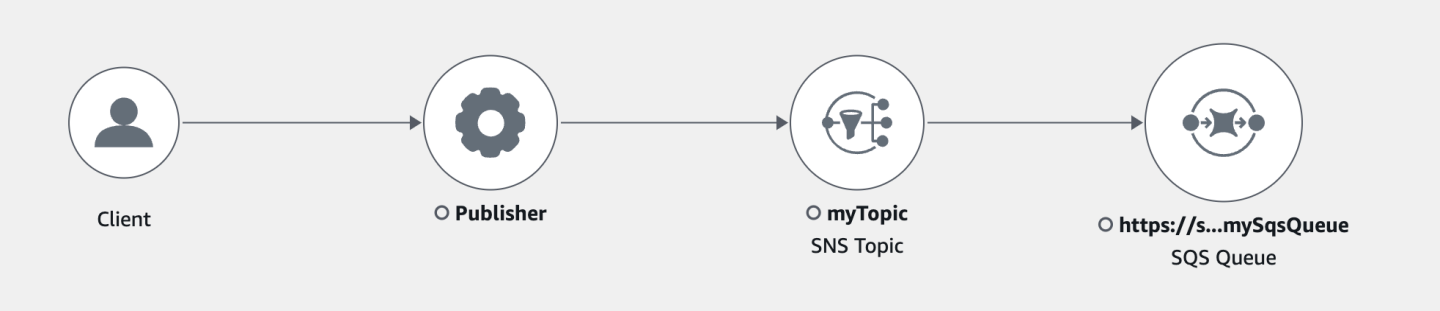
使用 CLI 建立以資源為基礎的政策，授與 Amazon SNS 將追蹤資料傳送至 X-Ray 的許可：

```
aws xray put-resource-policy --policy-name MyResourcePolicy --policy-document
 '{ "Version": "2012-10-17", "Statement": [ { "Sid": "SNSAccess", "Effect": "Allow",
 "Principal": { "Service": "sns.amazonaws.com" }, "Action": [ "xray:PutTraceSegments",
 "xray:GetSamplingRules", "xray:GetSamplingTargets" ], "Resource": "*",
 "Condition": { "StringEquals": { "aws:SourceAccount": "account-id" }, "StringLike":
 { "aws:SourceArn": "arn:partition:sns:region:account-id:topic-name" } } } ] }'
```

若要使用這些範例，請將*partitionregion*、*account-id*、和取代為您*topic-name*的特定 AWS 分割區、區域、帳戶 ID 和 Amazon SNS 主題名稱。若要授予所有 Amazon SNS 主題將追蹤資料傳送至 X-Ray 的權限，請將主題名稱取代為*。

在 X-Ray 主控台中檢視 Amazon SNS 發行者和訂閱者追蹤

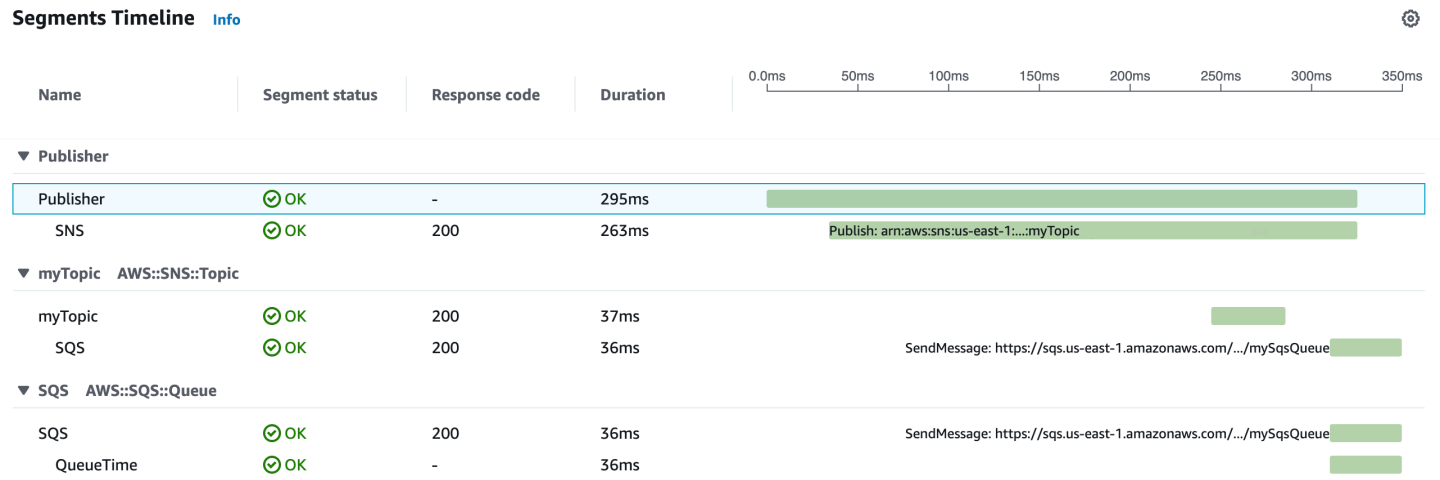
使用 X-Ray 主控台檢視追蹤地圖和追蹤詳細資料，以顯示 Amazon SNS 發佈者和訂閱者的連線檢視。針對某個主題開啟 Amazon SNS 主動追蹤時，X-Ray 追蹤對應和追蹤詳細資料對映會顯示 Amazon SNS 發行者、Amazon SNS 主題和下游訂閱者的連線節點：



選擇橫跨 Amazon SNS 發佈者和訂閱者的追蹤後，X-Ray 追蹤詳細資料頁面會顯示追蹤詳細資料對映和區段時間表。

Example Amazon SNS 發佈者和訂閱者的時間表範例

此範例顯示一個時間表，其中包含將訊息傳送至 Amazon SNS 主題 (由 Amazon SQS 訂閱者處理) 的 Amazon SNS 發行者。

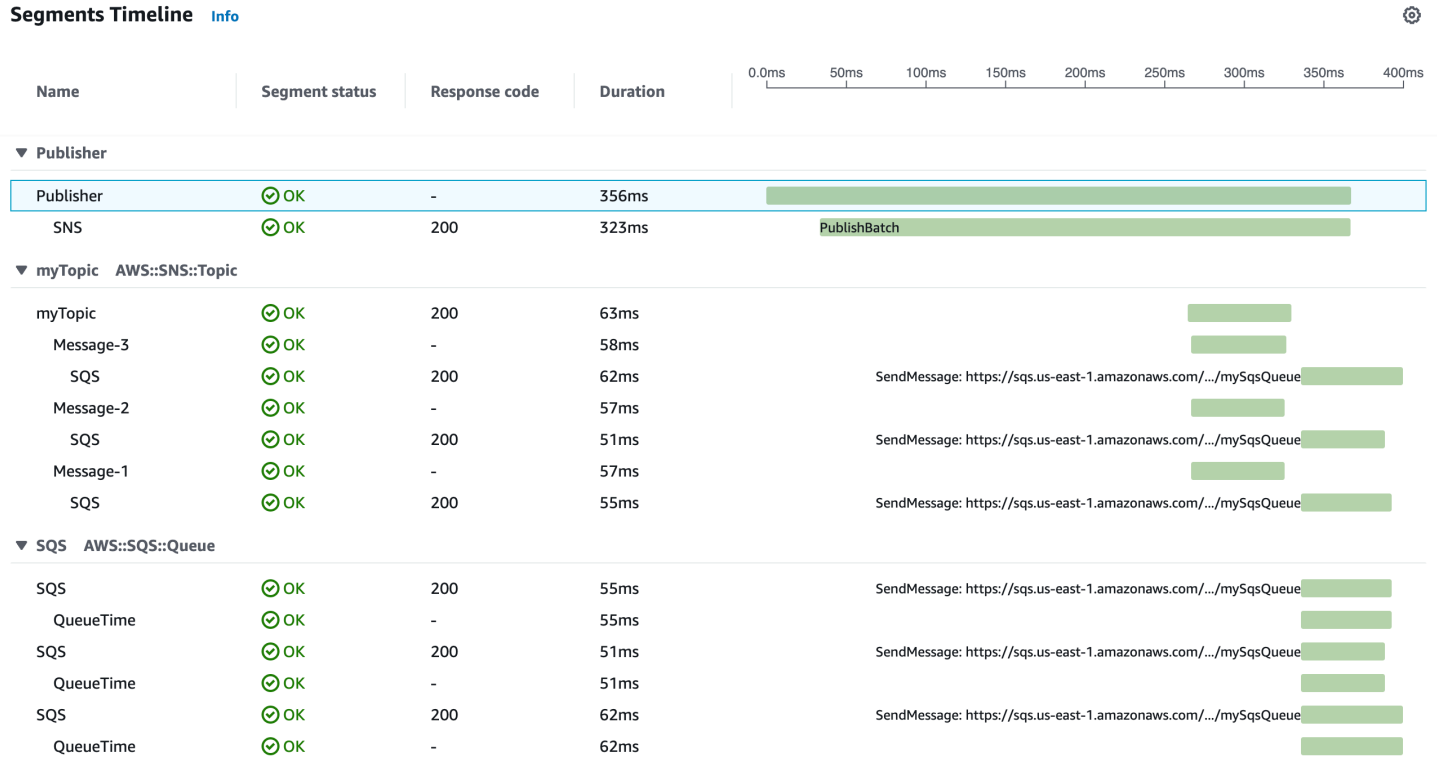


上述範例時間表提供有關 Amazon SNS 訊息流程的詳細資訊：

- SNS 區段代表來自用戶端 `Publish` API 呼叫的往返持續時間。
- MyTopic 區段代表 Amazon SNS 對發佈請求的回應延遲。
- SQS 子區段代表 Amazon SNS 將訊息發佈到 Amazon SQS 佇列所需的往返時間。
- MyTopic 區段和 SQS 子區段之間的時間表示訊息在 Amazon SNS 系統中花費的時間。

Example 包含批次處理 Amazon SNS 訊息的時間軸範例

如果在單一追蹤中批次處理多個 Amazon SNS 訊息，則區段時間表會顯示代表處理之每則訊息的區段。



AWS Step Functions 與 AWS X-Ray

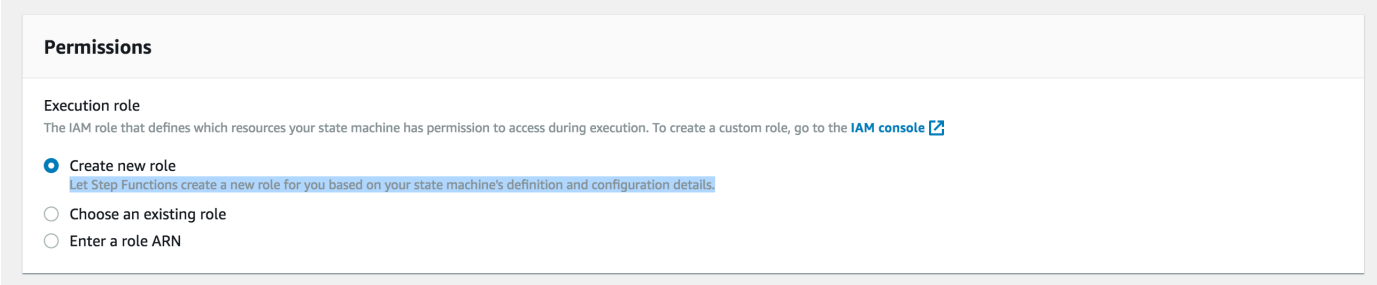
AWS X-Ray與 整合AWS Step Functions來跟蹤和分析 Step Functions 的請求。您可以直觀地顯示狀態機器、識別效能瓶頸，以及疑難排解導致錯誤的請求。如需詳細資訊，請參閱「」AWS X-Ray和 Step Functions中的AWS Step Functions開發人員指南。

創建新狀態機時啟用 X-Ray 跟蹤

1. 打開 Step Functions 控制台，位於https://console.aws.amazon.com/states/。
2. 選擇建立狀態機器。
3. 在定義狀態機器頁面上，選擇使用程式碼片段編寫或者從模板開始。如果選擇運行示例項目，則無法建立在 CX-Ray 追蹤。相反、啟用 X-Ray 追蹤建立您的狀態機器。
4. 選擇 Next (下一步)。
5. 在指定詳細資訊頁面上，配置狀態機器。
6. 選擇啟用 X-Ray 追蹤。

在現有狀態機器中啟用 X-Ray 追蹤

1. 在「Step Functions」控制台中，選擇要為其啟用跟蹤的狀態機。
2. 選擇 Edit (編輯)。
3. 選擇啟用 X-Ray 追蹤。
4. （可選）自動為狀態機生成新角色以包含 X-Ray 權限，方法是選擇建立新角色從「權限」窗口。



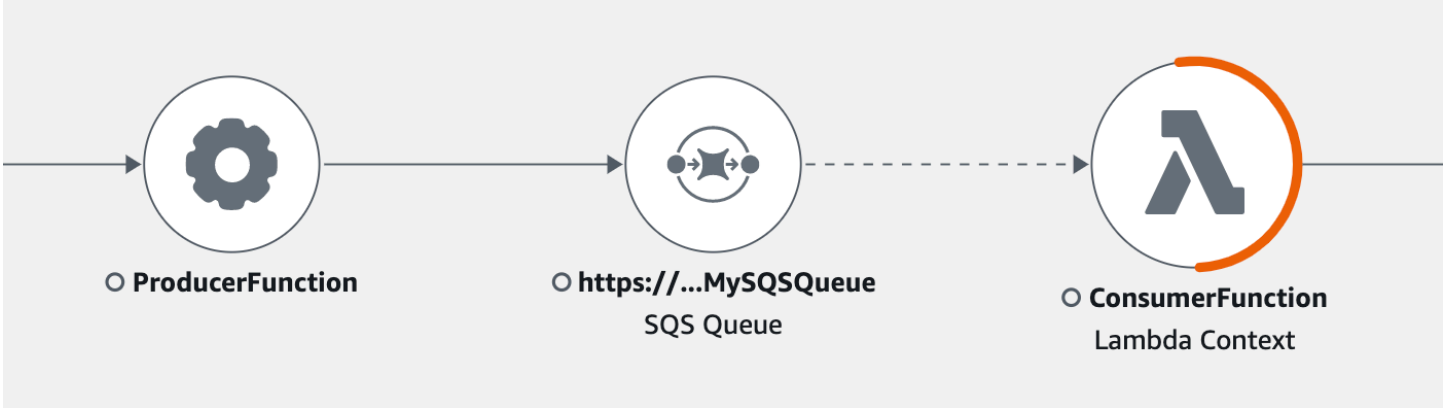
5. 選擇 Save (儲存)。

> **Note**
>
> 當您建立新的狀態機器時，它的 如果對請求進行採樣並且在上遊服務（如 Amazon API Gateway 或AWS Lambda。對於未通過控制台配置的任何現有狀態機，例如通過AWS CloudFormation範本,檢查您是否擁有授予足夠權限以啟用 X-Ray 跟蹤的 IAM 策略。

Amazon SQS 和 AWS X-Ray

AWS X-Ray 與 Amazon Simple Queue Service (Amazon SQS) 整合，以追蹤透過 Amazon SQS 佇列傳遞的訊息。如果服務使用 X-Ray SDK 追蹤請求，Amazon SQS 可以傳送追蹤標頭，並使用一致的追蹤 ID 繼續將原始追蹤從寄件者傳播到取用者。追蹤連續性可讓使用者追蹤、分析和偵錯整個下游服務。

AWS X-Ray 支援使用 Amazon SQS 和. AWS Lambda使用主 CloudWatch 控台查看每個請求與 Amazon SQS 排入佇列並由下游 Lambda 函數進行處理時的連線檢視。來自上游訊息生產者的追蹤會自動連結至來自下游 Lambda 消費者節點的追蹤，以建立應用程式的 end-to-end 檢視。如需詳細資訊，請參閱追蹤事件導向應用程式。



Amazon SQS 支援下列追蹤標頭儀器：

- 預設 HTTP 標頭 — 當您透過 SDK 呼叫 Amazon SQS 時，X-Ray 開發套件會自動將追蹤標頭填入為 HTTP 標頭。 AWS 預設追蹤標頭由 `X-Amzn-Trace-Id` 攜帶，並對應至 `SendMessage` 或 `SendMessageBatch` 請求中包含的所有訊息。若要進一步了解預設 HTTP 標頭，請參閱 追蹤標頭。
- **`AWSTraceHeader`**系統屬性 — Amazon SQS 保留的訊息系統屬性，用來攜帶 X-Ray 追蹤標頭及佇列中的訊息。`AWSTraceHeader` `AWSTraceHeader`即使沒有透過 X-Ray SDK 進行自動檢測，例如建立新語言的追蹤 SDK 時，仍可使用。同時設定這兩個標頭檢測時，訊息系統屬性會覆寫 HTTP 追蹤標頭。

在 Amazon EC2 上執行時，Amazon SQS 支援一次處理一則訊息。這適用於在現場部署主機上執行，以及使用容器服務 (例如 AWS Fargate Amazon ECS 或 AWS App Mesh) 時。

追蹤標頭會從 Amazon SQS 訊息大小和訊息屬性配額中排除。啟用 X-Ray 追蹤不會超過您的 Amazon SQS 配額。若要進一步了解 AWS 配額，請參閱 Amazon SQS 配額。

傳送 HTTP 追蹤標頭

Amazon SQS 中的寄件者元件可透過`SendMessageBatch`或`SendMessage`呼叫自動傳送追蹤標頭。檢測 AWS SDK 用戶端時，可透過 X-Ray SDK 支援的所有語言自動追蹤這些用戶端。您在這些服務中存取的追蹤 AWS 服務 和資源 (例如，Amazon S3 儲存貯體或 Amazon SQS 佇列) 會在 X-Ray 主控台的追蹤對應上顯示為下游節點。

若要瞭解如何使用您慣用的語言追蹤 AWS SDK 呼叫，請參閱支援的 SDK 中的下列主題：

- Go – 使用適用於 Go 的 X-Ray AWS SDK 追蹤 SDK 呼叫
- 爪哇 — 使用適用於 Java 的 X-Ray AWS SDK 追蹤 SDK 呼叫

- Node.js – 使用適用於 Node.js 的 X-Ray AWS SDK 追蹤 SDK 呼叫
- Python – 使用適用於 Python 的 X-Ray AWS SDK 追蹤 SDK 呼叫
- Ruby – 使用紅寶石的 X-Ray AWS SDK 跟踪 SDK 調用
- .NET – 使用適用於 .NET 的 X-Ray AWS SDK 追蹤 SDK 呼叫

擷取追蹤標頭和復原追蹤內容

如果您使用 Lambda 下游消費者，追蹤內容傳播是自動的。若要繼續與其他 Amazon SQS 取用者進行內容傳播，您必須手動測量接收器元件的遞交。

復原追蹤內容有三個主要步驟：

- 透過呼叫 `ReceiveMessage`API，從`AWSTraceHeader`屬性的佇列接收訊息。
- 從屬性擷取追蹤標頭。
- 從標頭復原追蹤 ID。選擇性地將更多指標新增至區段。

以下是使用適用於 Java 的 X-Ray SDK 編寫的示例實現。

Example ：擷取追蹤標頭和復原追蹤內容

```
// Receive the message from the queue, specifying the "AWSTraceHeader"
ReceiveMessageRequest receiveMessageRequest = new ReceiveMessageRequest()
        .withQueueUrl(QUEUE_URL)
        .withAttributeNames("AWSTraceHeader");
List<Message> messages = sqs.receiveMessage(receiveMessageRequest).getMessages();

if (!messages.isEmpty()) {
    Message message = messages.get(0);

    // Retrieve the trace header from the AWSTraceHeader message system attribute
    String traceHeaderStr = message.getAttributes().get("AWSTraceHeader");
    if (traceHeaderStr != null) {
        TraceHeader traceHeader = TraceHeader.fromString(traceHeaderStr);

        // Recover the trace context from the trace header
        Segment segment = AWSXRay.getCurrentSegment();
        segment.setTraceId(traceHeader.getRootTraceId());
        segment.setParentId(traceHeader.getParentId());

 segment.setSampled(traceHeader.getSampled().equals(TraceHeader.SampleDecision.SAMPLED));
```

```
    }
}
```

Amazon S3 和 AWS X-Ray

AWS X-Ray 與 Amazon S3 整合以追蹤上游請求以更新應用程式的 S3 儲存貯體。如果服務使用 X-Ray 開發套件追蹤請求，Amazon S3 可以將追蹤標頭傳送給下游事件訂閱者 AWS Lambda，例如 Amazon SQS 和 Amazon SNS。X-Ray 可為 Amazon S3 事件通知啟用追蹤訊息。

您可以使用 X-Ray 追蹤對應來檢視 Amazon S3 與應用程式使用的其他服務之間的連線。您也可以使用主控台來檢視指標，例如平均延遲和失敗率。如需 X-Ray 主控台的詳細資訊，請參閱AWS X-Ray 控制台。

Amazon S3 支持默認的 http 標頭儀器。當您透過開發套件呼叫 Amazon S3 時，X-Ray SDK 會自動將追蹤標頭填入為 HTTP 標頭。 AWS 預設的追蹤標頭由攜帶`X-Amzn-Trace-Id`。若要進一步瞭解關於追蹤標頭的資訊，請參閱概念頁追蹤標頭上的。Amazon S3 追蹤內容傳播支援下列訂閱者：Lambda、SQS 和 SNS。由於 SQS 和 SNS 本身不會發出區段資料，因此當 S3 觸發時，它們不會出現在追蹤或追蹤對映中，即使它們會將追蹤標頭傳播至下游服務。

設定 Amazon S3 事件通知

使用 Amazon S3 通知功能，您會在儲存貯體中發生特定事件時收到通知。然後，這些通知可以傳播到應用程式中的下列目的地：

- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Queue Service (Amazon SQS)
- AWS Lambda

如需支援事件的清單，請參閱 Amazon S3 開發人員指南中的支援事件類型。

Amazon SNS 和 Amazon SQS

若要將通知發佈到 SNS 主題或 SQS 佇列，您必須先授與 Amazon S3 許可。若要授與這些權限，請將 AWS Identity and Access Management (IAM) 政策附加至目的地 SNS 主題或 SQS 佇列。若要進一步了解所需的 IAM 政策，請參閱授與將訊息發佈到 SNS 主題或 SQS 佇列的權限。

如需將 SNS 和 SQS 與 X-Ray 整合的資訊，請參閱Amazon SNS 和 AWS X-Ray和Amazon SQS 和 AWS X-Ray。

AWS Lambda

當您使用 Amazon S3 主控台在 S3 儲存貯體上為 Lambda 函數設定事件通知時，主控台會在 Lambda 函數上設定必要的許可，以便 Amazon S3 具有從儲存貯體叫用函數的許可。如需詳細資訊，請參閱如何啟用和設定 S3 儲存貯體的事件通知？ 在 Amazon 簡單儲存服務主控台使用者指南中。

您也可以從授與 Amazon S3 許可以叫 AWS Lambda 用您的 Lambda 函數。如需詳細資訊，請參閱 AWS Lambda 開發人員指南中的教學課程：搭配 Amazon S3 使用 AWS Lambda。

如需有關整合 Lambda 與 X-Ray 的詳細資訊，請參閱在 AWS Lambda 中檢測 Java 程式碼。

建立 X 射線資源AWS CloudFormation

AWS X-Ray 已與 AWS CloudFormation 整合，這項服務可協助您建立 AWS 資源的模型和設定，以減少建立和管理資源和基礎設施的時間。您可以建立一個範本，描述所有您想要的 AWS 資源，AWS CloudFormation 就會為您佈建和設定那些資源。

當您使用AWS CloudFormation，您可以重複使用範本以一致且重複地設定 X-Ray 資源。只需描述一次您的資源，即可在多個 AWS 帳戶 帳戶與區域內重複佈建相同資源。

X 光及AWS CloudFormation模板

要為 X-Ray 和相關服務提供和配置資源，您必須了解AWS CloudFormation模板。範本是以 JSON 或 YAML 格式化的文本檔案。而您亦可以透過這些範本的說明，了解欲在 AWS CloudFormation 堆疊中佈建的資源。如果您不熟悉 JSON 或 YAML，您可以使用 AWS CloudFormation Designer 協助您開始使用 AWS CloudFormation 範本。如需詳細資訊，請參閱 AWS CloudFormation 使用者指南中的什麼是 AWS CloudFormation Designer？。

X 射線支援建立`AWS::XRay::Group`和`AWS::XRay::SamplingRule`資源AWS CloudFormation。如需詳細資訊，包括 JSON 和 YAML 範本的範例，請參閱X 射線資源類型參考在AWS CloudFormation使用者指南。

進一步了解 AWS CloudFormation

若要進一步了解 AWS CloudFormation，請參閱下列資源：

- AWS CloudFormation
- 《AWS CloudFormation 使用者指南》https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html
- AWS CloudFormation API 參考
- 《AWS CloudFormation 命令列介面使用者指南》https://docs.aws.amazon.com/cloudformation-cli/latest/userguide/what-is-cloudformation-cli.html

標籤 X-Ray 取樣規則和羣組

標籤是您可用來辨識和整理 AWS 資源的文字或短句。您可以將多個標籤新增至每個資源。每個標籤都包含一個索引鍵和您定義的選用值。例如，標籤鍵可能是**domain**，並且標籤值可能是**example.com**。您可以根據新增的標籤搜尋和篩選資源。如需標籤的方法的詳細資訊，請參標記AWS資源中的AWS一般參考。

您可以使用標籤在 CloudFront 分發上強制使用以標籤為基礎的許可。如需詳細資訊，請參閱「」控制存取AWS使用資源標籤。

> **Note**
>
> Tag Editor和AWS資源群組目前不支持 X-Ray 資源。您可以使用新增和管理標籤AWS X-Ray控制台或 API。

您可以使用 X-Ray 主控台、API、AWS CLI、軟件開發工具包和AWS Tools for Windows PowerShell。如需詳細資訊，請參閱下列文件：

- X-Ray API — 請參閱AWS X-RayAPI 參考：
  - ListTagsForResource
  - 創建抽樣規則
  - CreateGroup
  - TagResource
  - UntagResource
- AWS CLI— 請參xray中的AWS CLI命令參考
- 開發套件 – 請參閱 AWS 說明文件頁面上適用的開發套件說明文件

> **Note**
>
> 如果無法在 X-Ray 資源上添加或更改標籤，或者無法添加具有特定標記的資源，則可能沒有執行此操作的權限。要請求訪問權限，請聯繫AWS企業中的用户管理員X-Ray 中的許可。

主題

標籤限制

下列限制適用於標籤。

- 每一資源最多標籤數 – 50
- 金鑰長度上限 - 128 個 Unicode 字元
- 數值長度上限 - 256 個 Unicode 字元
- 金鑰與值的有效值 – a-z、A-Z、0-9、空格和下列字元：_ . : / = + - 及 @
- 標籤鍵與值皆區分大小寫。
- 請勿使用aws:作為鍵的前綴；它是保留給AWS使用。

> **Note**
>
> 您無法編輯或刪除系統標籤。

在主控台中管理標籤

您可以在創建 X-Ray 組或採樣規則時添加可選標記。稍後也可以在控制台中更改或刪除標籤。

下列程序說明如何新增、編輯和刪除在 X-Ray 主控台適用於組的標籤和採樣規則。

主題

將標籤新增到新的羣組 (主控台)

創建新的 X-Ray 組時，您可以在建立羣組(憑證已建立！) 頁面上的名稱有些許差異。

1. 登入AWS Management Console並開啟 X-Ray 主控台https://console.aws.amazon.com/xray/home。
2. 在導覽窗格中，展開組態，然後選擇Groups (群組)。
3. 選擇 Create group (建立群組)。
4. 在建立羣組頁面上，為組指定名稱和篩選器表達式。如需有關這些屬性的詳細資訊，請參閱設定群組。
5. In標籤中，輸入標籤索引鍵和選用的標籤值。例如，您可以在**Stage**，並且標籤值**Production**，表示此組用於生產用途。添加標籤時，將顯示一個新行，以便您添加另一個標籤（如果需要）。請參閱標籤限制，瞭解標籤的限制。
6. 當您完成新增標籤的作業時，請選擇建立羣組。

將標籤新增至新的採樣規則 (主控台)

創建新的 X-Ray 採樣規則時，可以在建立抽樣規則(憑證已建立！) 頁面上的名稱有些許差異。

1. 登入AWS Management Console並開啟 X-Ray 主控台https://console.aws.amazon.com/xray/home。
2. 在導覽窗格中，展開組態，然後選擇抽樣。
3. 選擇建立抽樣規則。
4. 在建立抽樣規則頁面上，指定名稱、優先級、限制、匹配條件和匹配屬性。如需有關這些屬性的詳細資訊，請參閱設定 取樣規則。
5. In標籤中，輸入標籤索引鍵和選用的標籤值。例如，您可以在**Stage**，並且標籤值**Production**，以指示此抽樣規則用於生產使用。添加標籤時，將顯示一個新行，以便您添加另一個標籤（如果需要）。請參閱標籤限制，瞭解標籤的限制。
6. 當您完成新增標籤的作業時，請選擇建立抽樣規則。

編輯或刪除組的標籤 (主控台)

您可以更改或刪除 X-Ray 組上的編輯羣組(憑證已建立！) 頁面上的名稱有些許差異。

1. 登入AWS Management Console並開啟 X-Ray 主控台https://console.aws.amazon.com/xray/home。
2. 在導覽窗格中，展開組態，然後選擇Groups (群組)。
3. 在 中Groups (群組)表中，選擇羣組名稱。
4. 在編輯羣組頁面上的標籤中，編輯標籤鍵和值。您不能有重複的標籤鍵。標籤值是可選的；如果需要，您可以刪除值。如需其他屬性的詳細資訊，請參編輯羣組頁面上，請參閱設定群組。請參閱標籤限制，瞭解標籤的限制。
5. 欲刪除標籤，請選擇X在標籤的右側。
6. 當您完成編輯或刪除標籤的作業時，請選擇更新羣組。

編輯或刪除抽樣規則的標籤 (主控台)

您可以更改或刪除 X-Ray 採樣規則上的標記，編輯抽樣規則(憑證已建立！) 頁面上的名稱有些許差異。

1. 登入AWS Management Console並開啟 X-Ray 主控台https://console.aws.amazon.com/xray/home。
2. 在導覽窗格中，展開組態，然後選擇抽樣。
3. 在 中抽樣規則表中，選擇採樣規則的名稱。
4. In標籤中，編輯標籤鍵和值。您不能有重複的標籤鍵。標籤值是可選的；如果需要，您可以刪除值。如需其他屬性的詳細資訊，請參編輯抽樣規則頁面上，請參閱設定 取樣規則。請參閱標籤限制，瞭解標籤的限制。
5. 欲刪除標籤，請選擇X在標籤的右側。
6. 當您完成編輯或刪除標籤的作業時，請選擇更新抽樣規則。

管理標籤AWS CLI

您可以在建立 X-Ray 組或採樣規則時新增標籤。您也可以使用AWS CLI來建立和管理標籤。要更新現有組或取樣規則上的標籤，請使用AWS X-Ray主控台或TagResource或者UntagResourceAPI。

主題

- 將標籤添加到新的 X-Ray 組或採樣規則 (CLI)
- 新增標籤到現有資源 (CLI)
- 列出資源上的標籤 (CLI)

- 刪除資源上的標籤 (CLI)

將標籤添加到新的 X-Ray 組或採樣規則 (CLI)

要在創建新的 X-Ray 組或採樣規則時添加可選標記，請使用以下命令之一。

- 若要將標籤新增至新增至新組，請執行下列命令，將*group_name*使用您的羣組名稱，*###*使用您的服務端點，*key_name*使用標籤鍵，並且可以選擇*#*使用標籤值。如需如何建立組的詳細資訊，請參群組。

```
aws xray create-group \
   --group-name "group_name" \
   --filter-expression "service(\"mydomain.com\") {fault OR error}" \
   --tags [{"Key": "key_name","Value": "value"},{"Key": "key_name","Value": "value"}]
```

以下是範例。

```
aws xray create-group \
   --group-name "AdminGroup" \
   --filter-expression "service(\"mydomain.com\") {fault OR error}" \
   --tags [{"Key": "Stage","Value": "Prod"},{"Key": "Department","Value": "QA"}]
```

- 若要將標籤新增至新採樣規則，請執行下列命令，將*key_name*使用標籤鍵，並且可以選擇*#*使用標籤值。此命令指定`--sampling-rule`參數作為 JSON 檔案。如需如何建立抽樣規則的詳細資訊，請參抽樣規則。

```
aws xray create-sampling-rule \
   --cli-input-json file://file_name.json
```

下列為 JSON 檔案的內容*#### .json*指定的`--cli-input-json`參數。

```
{
    "SamplingRule": {
        "RuleName": "rule_name",
        "RuleARN": "string",
        "ResourceARN": "string",
        "Priority": integer,
        "FixedRate": double,
        "ReservoirSize": integer,
        "ServiceName": "string",
```

```
        "ServiceType": "string",
        "Host": "string",
        "HTTPMethod": "string",
        "URLPath": "string",
        "Version": integer,
        "Attributes": {"attribute_name": "value","attribute_name": "value"...}
    }
    "Tags": [
           {
               "Key":"key_name",
               "Value":"value"
           },
           {
               "Key":"key_name",
               "Value":"value"
           }
          ]
}
```

下列是範例命令。

```
aws xray create-sampling-rule \
   --cli-input-json file://9000-base-scorekeep.json
```

下列為範例的內容9000-base-scorekeep.json文件指定--cli-input-json參數。

```
{
    "SamplingRule": {
        "RuleName": "base-scorekeep",
        "ResourceARN": "*",
        "Priority": 9000,
        "FixedRate": 0.1,
        "ReservoirSize": 5,
        "ServiceName": "Scorekeep",
        "ServiceType": "*",
        "Host": "*",
        "HTTPMethod": "*",
        "URLPath": "*",
        "Version": 1
    }
    "Tags": [
           {
```

```
                "Key":"Stage",
                "Value":"Prod"
            },
            {
                "Key":"Department",
                "Value":"QA"
            }
           ]
}
```

新增標籤到現有資源 (CLI)

您可以執行`tag-resource`命令將標籤添加到現有的 X-Ray 組或採樣規則此方法可能比通過運行`update-group`或者`update-sampling-rule`。

要向組或採樣規則添加標籤，請運行以下命令，將 ARN 替換為資源的 ARN，並指定要添加的標籤的鍵和可選值。

```
aws xray tag-resource \
   --resource-arn "ARN" \
   --tag-keys [{"Key":"key_name","Value":"value"}, {"Key":"key_name","Value":"value"}]
```

以下是範例。

```
aws xray tag-resource \
   --resource-arn "arn:aws:xray:us-east-2:01234567890:group/AdminGroup" \
   --tag-keys [{"Key": "Stage","Value": "Prod"},{"Key": "Department","Value": "QA"}]
```

列出資源上的標籤 (CLI)

您可以執行`list-tags-for-resource`命令列出 X-Ray 組或採樣規則的標籤。

要列出與組或取樣規則相關聯的標籤，請運行以下命令，將 ARN 替換為資源的 ARN。

```
aws xray list-tags-for-resource \
   --resource-arn "ARN"
```

以下是範例。

```
aws xray list-tags-for-resource \
```

```
   --resource-arn "arn:aws:xray:us-east-2:01234567890:group/AdminGroup"
```

刪除資源上的標籤 (CLI)

您可以執行`untag-resource`命令將 X-Ray 組或採樣規則移除標籤。

要從組或取樣規則中刪除標籤，請運行以下命令，將 ARN 替換為資源的 ARN，並指定要刪除的標籤的鍵。

您只能刪除整個標籤，使用`untag-resource`命令。要刪除標記值，請使用 X-Ray 控制台，或者刪除標籤並添加具有相同鍵但值不同或空值的新標記。

```
aws xray untag-resource \
   --resource-arn "ARN" \
   --tag-keys ["key_name","key_name"]
```

以下是範例。

```
aws xray untag-resource \
   --resource-arn "arn:aws:xray:us-east-2:01234567890:group/group_name" \
   --tag-keys ["Stage","Department"]
```

根據標籤控制對 X-Ray 資源的存取

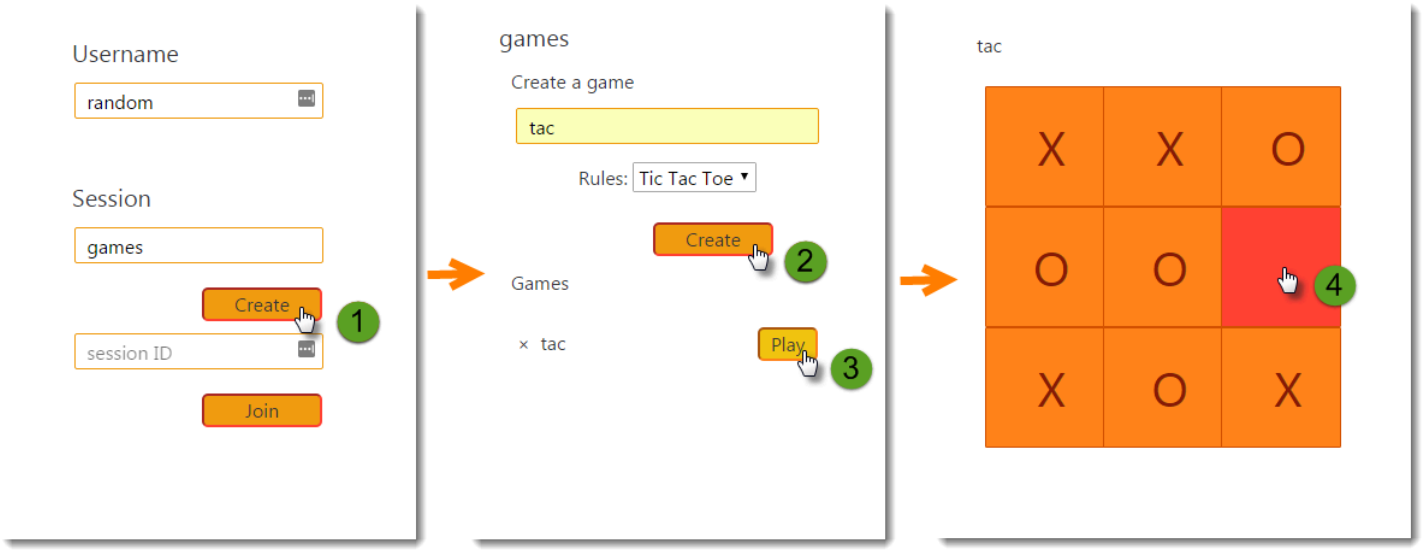
您可以將標籤附加到 X-Ray 組或採樣規則，或將請求中的標籤傳遞給 X-Ray。若要根據標籤控制存取，請使用 `xray:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，在政策的條件元素中，提供標籤資訊。若要進一步了解這些條件鍵，請參控制 存取AWS使用資源標籤。

若要檢視身分型政策範例，以根據該資源上的標籤來限制存取資源，請參閱 根據標籤管理 X-Ray 群組和取樣規則的存取。

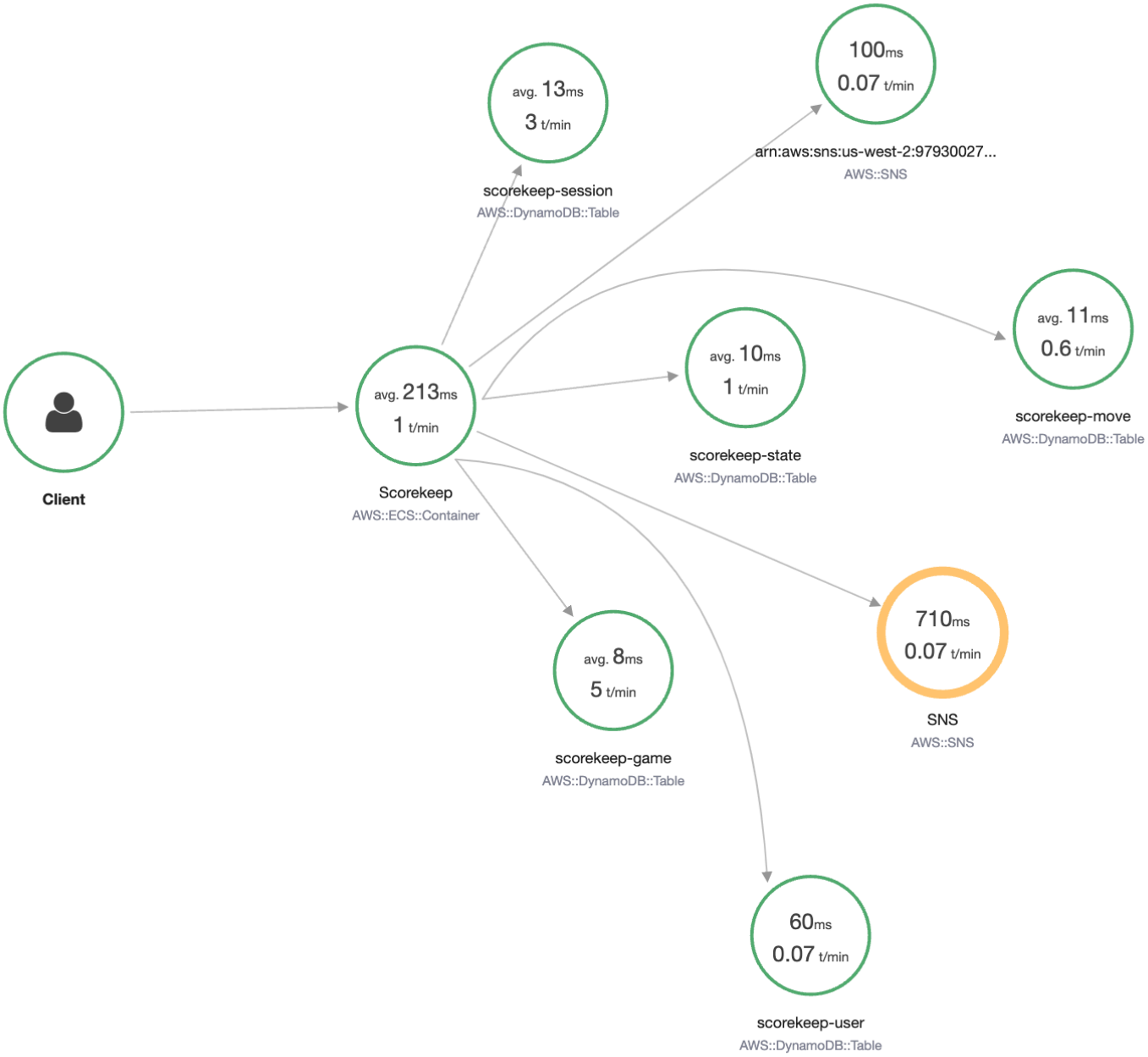
AWS X-Ray 範例應用

上 GitHub提供的 AWS X-Ray eb-java-scorekeep範例應用程式會顯示如何使用 AWS X-Ray SDK 來檢測傳入的 HTTP 呼叫、DynamoDB SDK 用戶端和 HTTP 用戶端。範例應用程式可用 AWS CloudFormation 來建立 DynamoDB 表格、在執行個體上編譯 Java 程式碼，以及執行 X-Ray 精靈，無需任何其他設定。

請參閱 S corekeep 教學課程，以使用或開始安裝和使用已檢測的範例應用程式 AWS Management Console 。 AWS CLI



此範例包括前端網頁應用程式、它呼叫的 API，以及用來儲存資料的 DynamoDB 資料表。具有過濾器，插件和儀表 AWS SDK 客戶端的基本檢測顯示在項目的`xray-gettingstarted`分支中。這次您在入門教學中部署的項目。由於此分支僅包含基本項目，您可以將它針對 `master` 分支進行 diff，來快速了解基本項目。



範例應用程式會在這些檔案中示範基本檢測：

- HTTP 要求篩選器 — `WebConfig.java`
- AWS SDK 用戶端檢測 — `build.gradle`

該應用程序的`xray`分支包括使用 HttpClient，註釋，SQL 查詢，自定義子段，儀表AWS Lambda函數以及檢測初始化代碼和腳本。

為了支援使用者登入和在瀏覽器中 AWS SDK for JavaScript 使用，該分支機`xray-cognito`構新增了 Amazon Cognito 以支援使用者身份驗證和授權。使用從 Amazon Cognito 擷取的登入資料，Web 應用程式也會將追蹤資料傳送至 X-Ray，以記錄從用戶端的角度來看請求資訊。瀏覽器用戶端會在追蹤對映上顯示為自己的節點，並記錄其他資訊，包括使用者正在檢視的頁面 URL 和使用者的 ID。

最後，該`xray-worker`分支新增了一個經過檢測的 Python Lambda 函數，該函數可以獨立執行，並處理來自 Amazon SQS 佇列的項目。Scorekeep 會在每次遊戲結束時新增項目到佇列中。由 CloudWatch 事件觸發的 Lambda 工作者每隔幾分鐘就會從佇列中提取項目，然後處理這些項目，以將遊戲記錄存放在 Amazon S3 中進行分析。

主題

開始使用記分範例應用程式

本教學課程使用 Scorekeep 範例應用程式的`xray-gettingstarted`分支，該應用程式用 AWS CloudFormation 於建立和設定在 Amazon ECS 上執行範例應用程式和 X-Ray 精靈的資源。此應用程式會使用 Spring 架構來實作 JSON 網頁 API，並將資料保留 AWS SDK for Java 至 Amazon DynamoDB。應用程式中的 Servlet 篩選器會檢測應用程式提供的所有傳入請求，以及 AWS SDK 用戶端上的要求處理常式會向 DynamoDB 進行下游呼叫。

您可以使用 AWS Management Console 或遵循本自學課程 AWS CLI。

章節

必要條件

本教學課程用 AWS CloudFormation 於建立和設定執行範例應用程式和 X-Ray 精靈的資源。在教學課程中安裝和執行需要下列先決條件：

1. 如果您使用具有有限許可的 IAM 使用者，請在 IAM 主控台中新增以下使用者政策：
   - `AWSCloudFormationFullAccess`— 訪問和使用 CloudFormation
   - `AmazonS3FullAccess`-將模板文件上傳到 CloudFormation使用 AWS Management Console
   - `IAMFullAccess`— 創建 Amazon ECS 和 Amazon EC2 實例角色
   - `AmazonEC2FullAccess`— 創建 Amazon EC2 資源
   - `AmazonDynamoDBFullAccess`— 建立動 DynamoDB 料表
   - `AmazonECS_FullAccess`— 創建 Amazon ECS 資源
   - `AmazonSNSFullAccess`— 創建 Amazon SNS 主題
   - `AWSXrayReadOnlyAccess`— 允許在 X-Ray 控制台中查看跟踪圖和痕跡
2. 若要使用執行教學課程 AWS CLI，請安裝 CLI 2.7.9 版或更新版本，並使用上一個步驟中的使用者設定 CLI。使用使用者進行設定時，請確定已 AWS CLI 設定區域。如果未設定區域，您將需要附加`--region` *`AWS-REGION`*至每個 CLI 命令。
3. 請確定已安裝 Git，以便複製範例應用程式存放庫。
4. 使用下列程式碼範例來複製 Scorekeep 儲存庫的`xray-gettingstarted`分支：

```
git clone https://github.com/aws-samples/eb-java-scorekeep.git xray-scorekeep -b
 xray-gettingstarted
```

使用下列方式安裝記分應用程式 CloudFormation

AWS Management Console

使用安裝範例應用程式 AWS Management Console

1. 開啟 CloudFormation 主控台
2. 選擇 [建立堆疊]，然後從下拉式功能表中選擇 [使用新資源]。
3. 在 Specify template (指定範本) 區段中，選擇 Upload a template file (上傳範本檔案)。
4. 選擇選擇文件，導航到克隆 git repo 時創建的文件`xray-scorekeep/cloudformation`夾，然後選擇該`cf-resources.yaml`文件。
5. 選擇 Next (下一步) 繼續。
6. 輸`scorekeep`入到堆棧名稱文本框，然後選擇下一頁在頁面底部繼續。請注意，本教程的其餘部分假定堆棧被命名為`scorekeep`。
7. 捲動至 [設定堆疊選項] 頁面底部，然後選擇 [下一步] 繼續。
8. 捲動至 [檢閱] 頁面底部，選擇 CloudFormation 可能建立具有自訂名稱的 IAM 資源的確認核取方塊，然後選擇 [建立堆疊]。
9. CloudFormation 堆棧現在正在創建。堆棧狀態將`CREATE_IN_PROGRESS`持續約五分鐘，然後再更改為`CREATE_COMPLETE`。狀態會定期重新整理，或者您可以重新整理頁面。

AWS CLI

使用安裝範例應用程式 AWS CLI

1. 導覽至您先前在教學課程中複製的`xray-scorekeep`儲存庫`cloudformation`資料夾：

```
cd xray-scorekeep/cloudformation/
```

2. 輸入下列 AWS CLI 指令以建立 CloudFormation 堆疊：

```
aws cloudformation create-stack --stack-name scorekeep --capabilities
 "CAPABILITY_NAMED_IAM" --template-body file://cf-resources.yaml
```

3. 等到 CloudFormation 堆棧狀態為`CREATE_COMPLETE`，這將需要大約五分鐘。使用以下 AWS CLI 命令來檢查狀態：

```
aws cloudformation describe-stacks --stack-name scorekeep --query
 "Stacks[0].StackStatus"
```

產生追蹤資料

範例應用程式包含前端 Web 應用程式。使用 Web 應用程式產生 API 的流量，並將追蹤資料傳送至 X-Ray。首先，使用 AWS Management Console 或來擷取 Web 應用程式 URL AWS CLI：

AWS Management Console

使用尋找應用程式 URL AWS Management Console

1. 開啟 CloudFormation 主控台
2. 從清單中選擇`scorekeep`堆疊。
3. 選擇`scorekeep`堆疊頁面上的 [輸出] 索引標籤，然後選擇 `LoadBalancerUrl` URL 連結以開啟 Web 應用程式。

AWS CLI

使用尋找應用程式 URL AWS CLI

1. 使用以下命令來顯示 Web 應用程序的 URL：

```
aws cloudformation describe-stacks --stack-name scorekeep --query
 "Stacks[0].Outputs[0].OutputValue"
```

2. 複製此 URL 並在瀏覽器中開啟，以顯示記分網頁應用程式。

使用 Web 應用程式產生追蹤資料

1. 選擇 Create (建立) 來產生使用者及工作階段。
2. 輸入 game name (遊戲名稱)、將 Rules (規則) 設為 Tic Tac Toe (井字遊戲)，然後選擇 Create (建立)。
3. 選擇 Play (遊玩) 來啟動遊戲。
4. 選擇一個磚來進行移動並變更遊戲狀態。

這些步驟中的每個步驟都會產生 API 的 HTTP 請求，並對 DynamoDB 產生下游呼叫，以讀取和寫入使用者、工作階段、遊戲、移動和狀態資料。

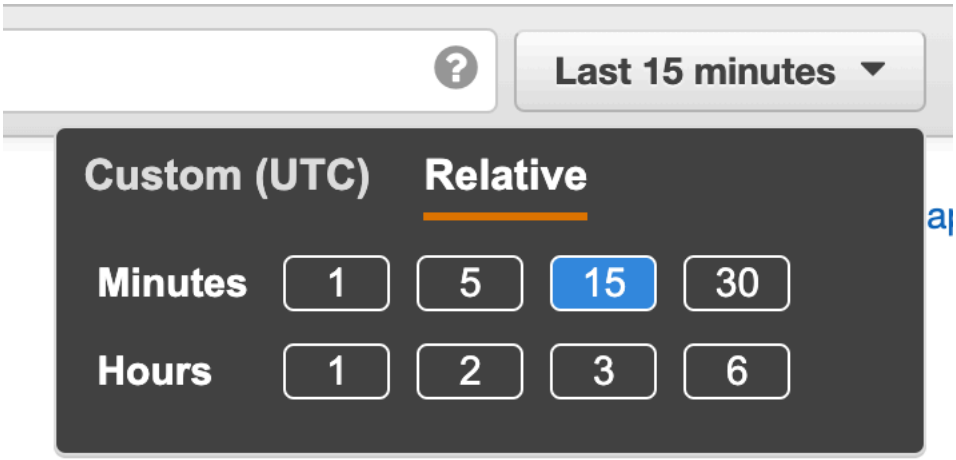
在「」中檢視軌跡圖 AWS Management Console

您可以在 X-Ray 和 CloudWatch 主控台中查看範例應用程式產生的軌跡圖和軌跡。

X-Ray console

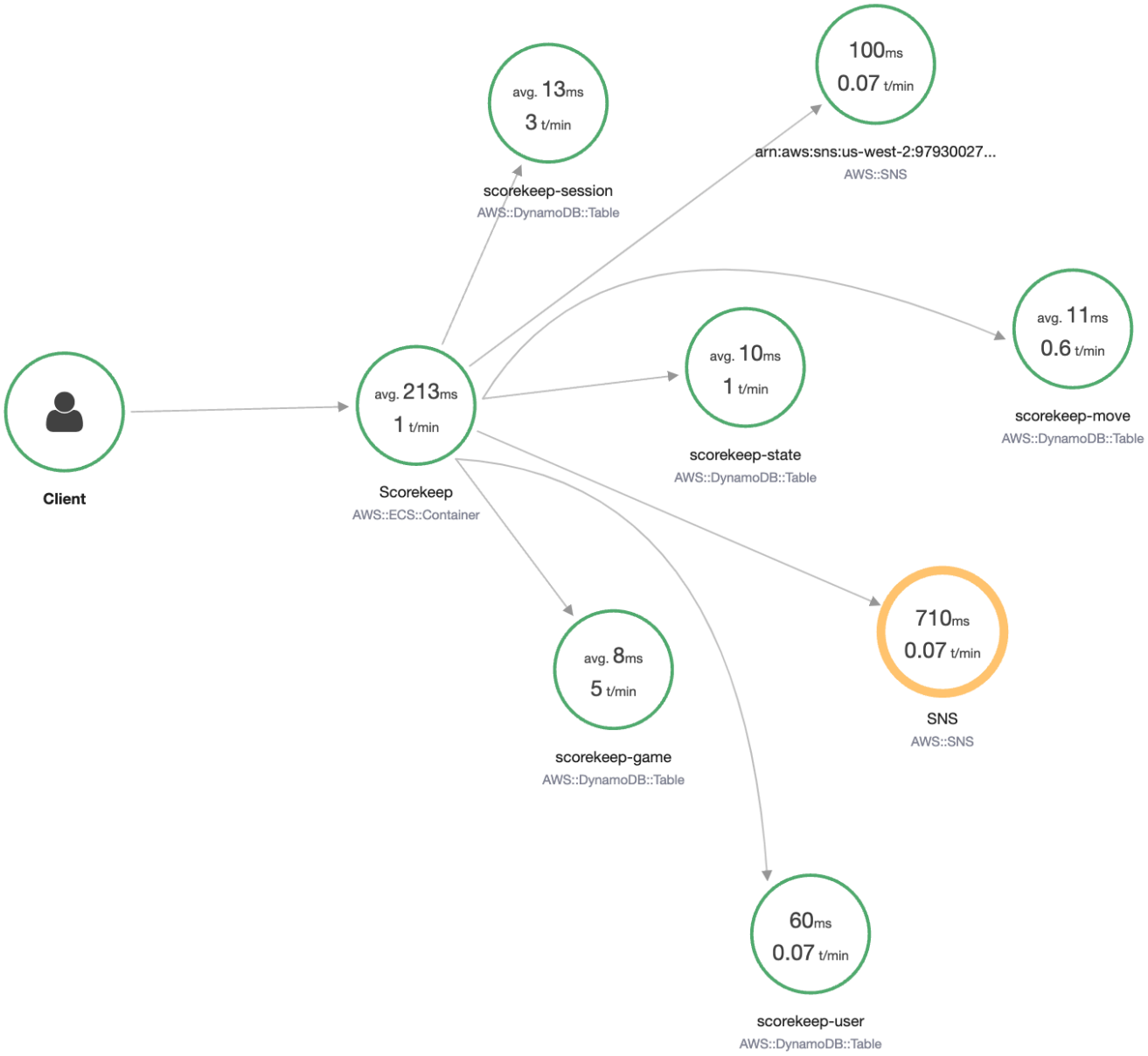
使用 X-Ray 控制台

1. 開啟 X-Ray 主控台的追蹤地圖頁面。
2. 主控台會顯示 X-Ray 從應用程式傳送的追蹤資料所產生的服務圖表。請務必視需要調整追蹤對應的時間週期，以確保它會顯示自您第一次啟動 Web 應用程式以來的所有追蹤。



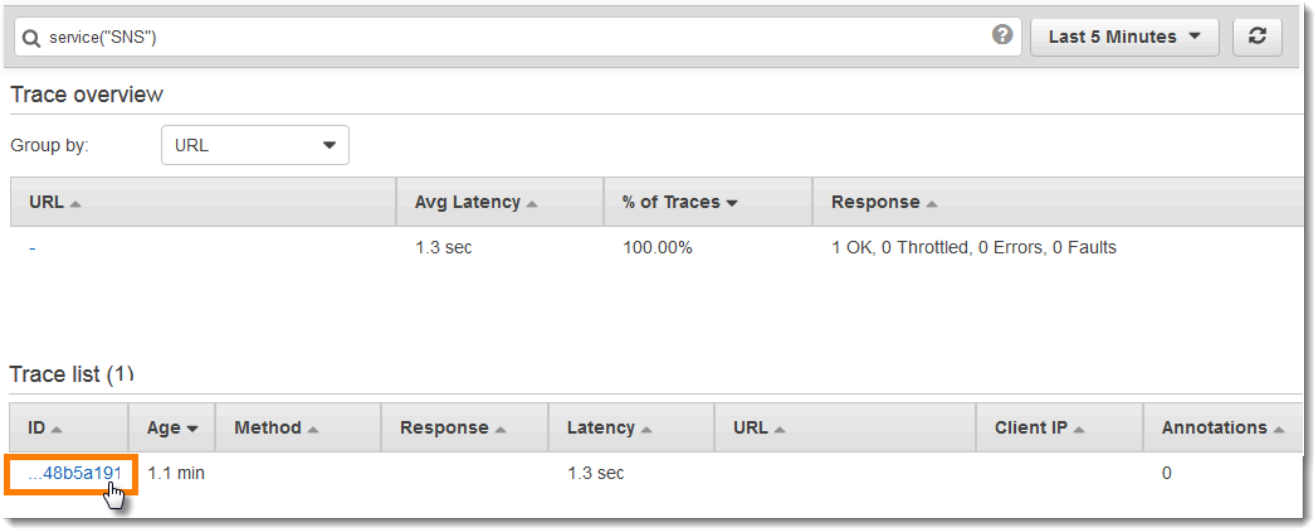
追蹤對應會顯示網路應用程式用戶端、在 Amazon ECS 中執行的 API，以及應用程式使用的每個 DynamoDB 表格。每個向應用程式發出的請求都會在命中 API 時受到追蹤、產生向下游服務發出的請求並完成，其數量上限為可設定的每秒請求數量上限。

您可以在服務圖表中選擇任何節點來檢視產生流量至該節點的請求追蹤。目前，Amazon SNS 節點為黃色。向下切入來了解原因。

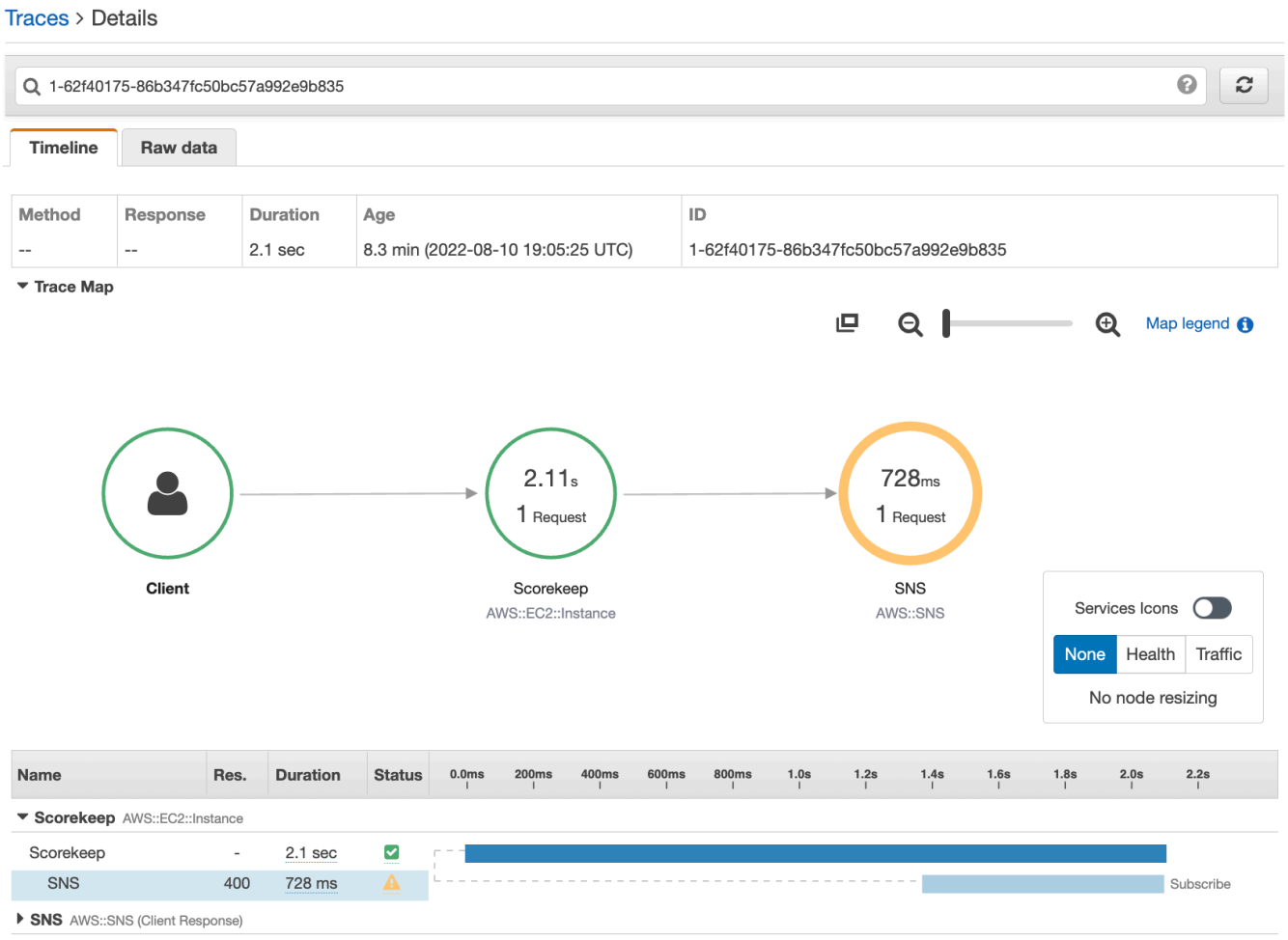


尋找錯誤原因

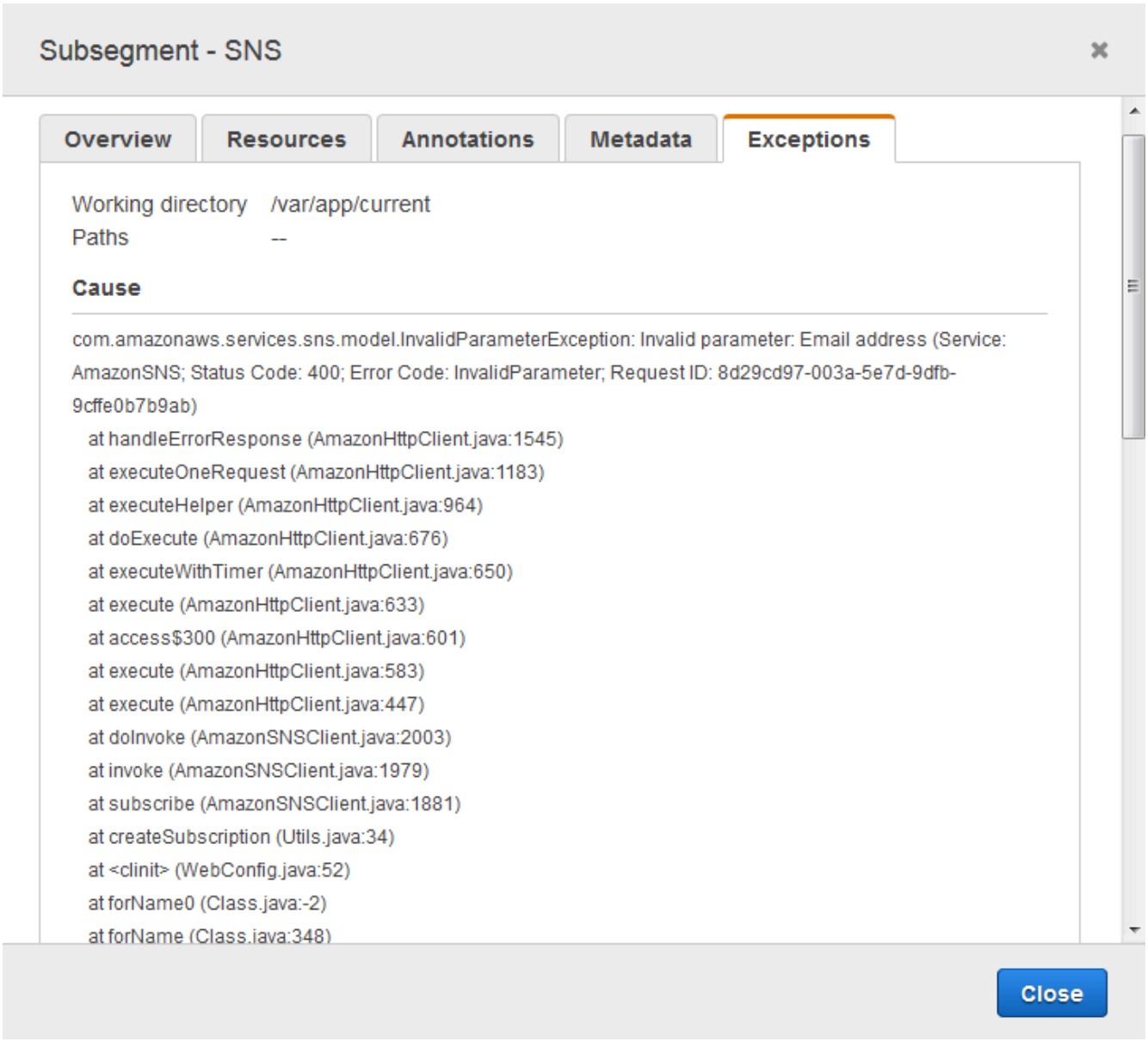
1. 選擇名為 SNS 的節點。此時會顯示節點詳細資料面板。
2. 選擇 View traces (檢視追蹤) 以存取 Trace overview (追蹤概觀) 畫面。
3. 從 Trace list (追蹤清單) 中選擇追蹤。此追蹤不具有方法或 URL，因為它是在啟動期間，而非回應傳入請求時記錄。



4. 在頁面底部的 Amazon SNS 區段中選擇錯誤狀態圖示，以開啟 SNS 子區段的「例外」頁面。



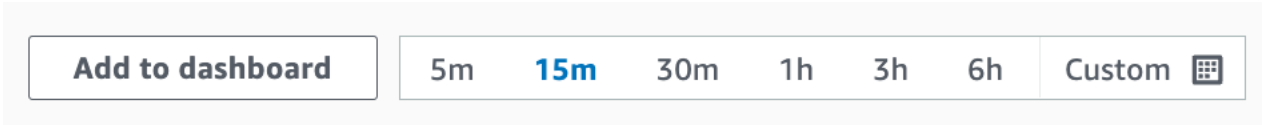
5. X-Ray SDK 會自動擷取已檢測的 AWS SDK 用戶端所擲回的例外狀況，並記錄堆疊追蹤。



CloudWatch console

使用主 CloudWatch 控台

1. 開啟主控台的 X-Ray 追蹤對應頁 CloudWatch 面。
2. 主控台會顯示 X-Ray 從應用程式傳送的追蹤資料所產生的服務圖表。請務必視需要調整追蹤對應的時間週期，以確保它會顯示自您第一次啟動 Web 應用程式以來的所有追蹤。



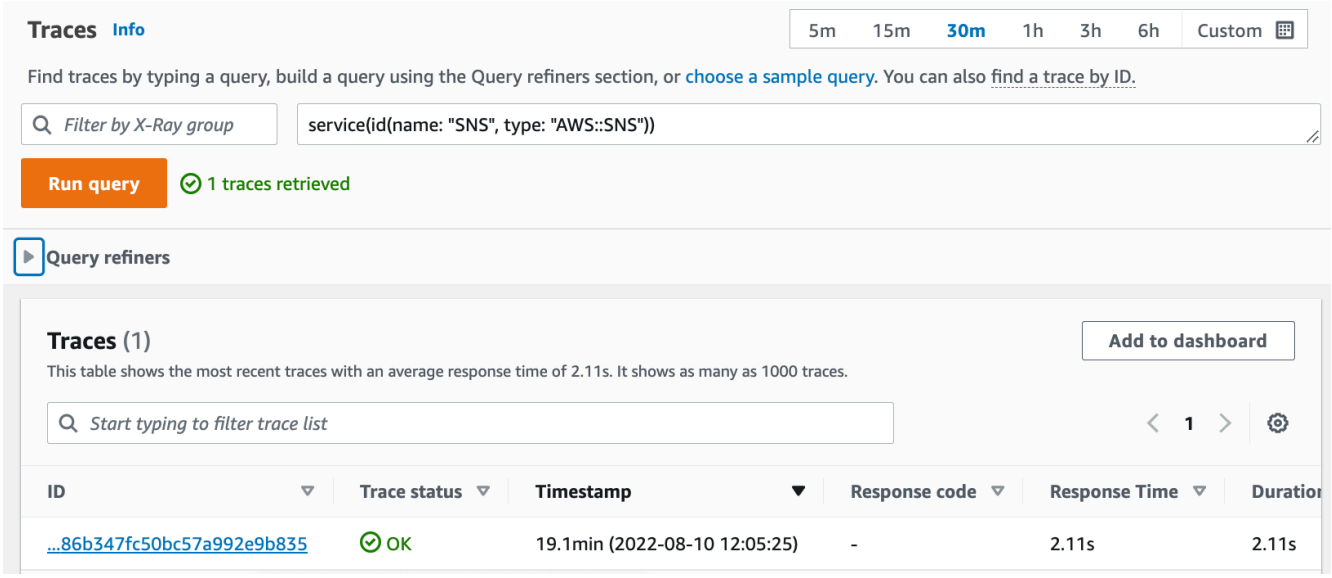
追蹤對應會顯示網路應用程式用戶端、在 Amazon EC2 中執行的 API，以及應用程式使用的每個 DynamoDB 表格。每個向應用程式發出的請求都會在命中 API 時受到追蹤、產生向下游服務發出的請求並完成，其數量上限為可設定的每秒請求數量上限。

您可以在服務圖表中選擇任何節點來檢視產生流量至該節點的請求追蹤。目前，Amazon SNS 節點是橙色的。向下切入來了解原因。

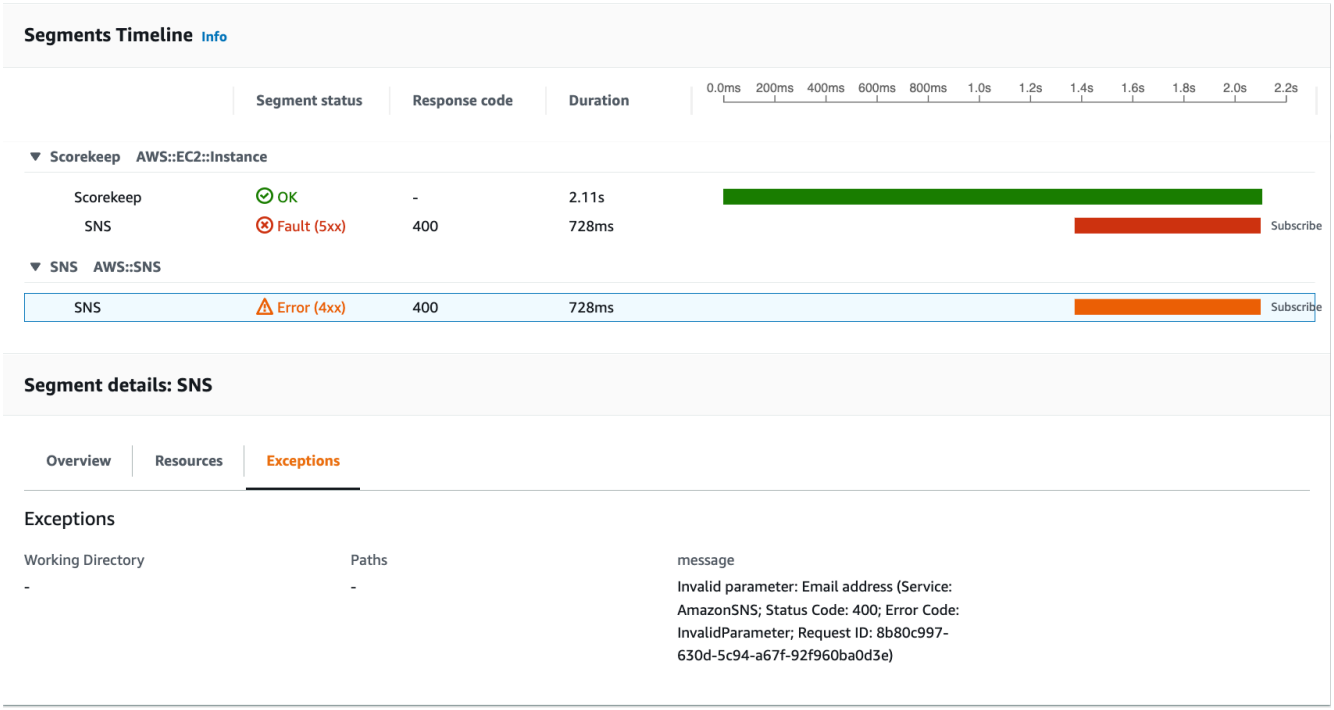

scorekeep-state
DynamoDB Table
scorekeep-game
DynamoDB Table
scorekeep-move
DynamoDB Table
Client
Scorekeep
ECS Container
SNS
SNS
scorekeep-session
DynamoDB Table

尋找錯誤原因

1. 選擇名為 SNS 的節點。SNS 節點詳細資訊面板顯示在地圖下方。
2. 選擇檢視追蹤來存取「追蹤」頁面。
3. 添加頁面底部，從跟踪列表中選擇跟踪。此追蹤不具有方法或 URL，因為它是在啟動期間，而非回應傳入請求時記錄。



4. 選擇區段時間表底部的 Amazon SNS 子區段，然後選擇 SNS 子區段的「例外」索引標籤以檢視例外詳細資訊。



原因指出在 WebConfig 類別中對 createSubscription 發出的呼叫內所提供的電子郵件地址無效。在下一節中，我們將修正此問題。

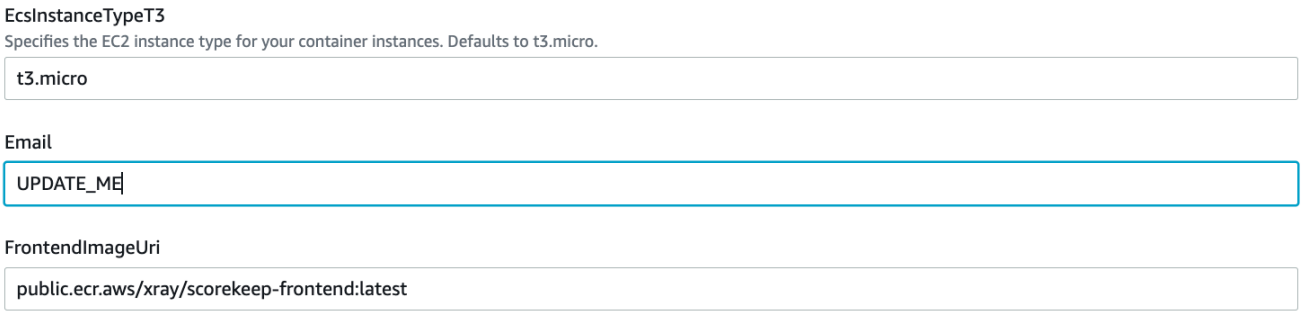
設定 Amazon SNS 通知

記分保持使用者完成遊戲時，會使用 Amazon SNS 傳送通知。當應用程式啟動時，它會嘗試為 CloudFormation 堆疊參數中定義的電子郵件地址建立訂閱。該呼叫目前失敗。設定通知電子郵件以啟用通知，並解決追蹤對映中反白顯示的失敗項目。

AWS Management Console

若要使用設定 Amazon SNS 通知 AWS Management Console

1. 開啟 CloudFormation 主控台
2. 選擇清單中scorekeep堆疊名稱旁邊的圓形按鈕，然後選擇 [更新]。
3. 確定已選取 [使用目前的範本]，然後按 [更新堆疊] 頁面上的 [下一步]。
4. 在清單中找到「電子郵件」參數，並以有效的電子郵件地址取代預設值。



5. 向下捲動到頁面底部並選擇 Next (下一步)。
6. 捲動至 [檢閱] 頁面底部，選擇 CloudFormation 可能建立具有自訂名稱的 IAM 資源的確認核取方塊，然後選擇 [更新堆疊]。
7. CloudFormation 堆棧現在正在更新。堆棧狀態將UPDATE_IN_PROGRESS持續約五分鐘，然後再更改為UPDATE_COMPLETE。狀態會定期重新整理，或者您可以重新整理頁面。

AWS CLI

若要使用設定 Amazon SNS 通知 AWS CLI

1. 導覽至您先前建立的xray-scorekeep/cloudformation/資料夾，然後在文字編輯器中開啟cf-resources.yaml檔案。
2. 在「電子郵件」參數中尋找Default值，並將其從 *UPDATE_ME* 變更為有效的電子郵件地址。

```
Parameters:
  Email:
    Type: String
    Default: UPDATE_ME # <- change to a valid abc@def.xyz email address
```

3. 從cloudformation資料夾中，使用下列 AWS CLI 命令更新 CloudFormation 堆疊：

```
aws cloudformation update-stack --stack-name scorekeep --capabilities
 "CAPABILITY_NAMED_IAM" --template-body file://cf-resources.yaml
```

4. 等到 CloudFormation 堆棧狀態為止UPDATE_COMPLETE，這將需要幾分鐘的時間。使用以下 AWS CLI 命令來檢查狀態：

```
aws cloudformation describe-stacks --stack-name scorekeep --query
 "Stacks[0].StackStatus"
```

更新完成時，Scorekeep 會重新啟動並建立 SNS 主題訂閱。檢查您的電子郵件以確認訂閱，在您完成遊戲時查看更新。開啟追蹤對應，確認對 SNS 的呼叫不再失敗。

探索範例應用程式

範例應用程式是 Java 中的 HTTP 網頁 API，其設定為使用適用於 Java 的 X-Ray SDK。當您使用 CloudFormation 範本部署應用程式時，它會建立 DynamoDB 表、Amazon ECS 叢集，以及在 ECS 上執行記分所需的其他服務。ECS 的任務定義檔案是透過 CloudFormation建立的。此檔案定義 ECS 叢集中每項工作所使用的容器映像。這些圖像是從官方 X-Ray 公共 ECR 獲得的。得分保持 API 容器映像具有使用搖籃編譯的 API。Scorekeep 前端容器的容器映像使用 nginx 代理服務器為前端提供服務。此伺服器會將要求路由傳送至以 /api 開頭的路徑至 API。

為了檢測傳入 HTTP 請求，應用程式會新增軟體開發套件提供的 TracingFilter。

Example src /主/爪/記分/. Java-服務程序過濾器 WebConfig

```
import javax.servlet.Filter;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;
...

@Configuration
public class WebConfig {

  @Bean
```

```
  public Filter TracingFilter() {
    return new AWSXRayServletFilter("Scorekeep");
  }
...
```

此篩選條件會傳送應用程式處理的所有傳入請求相關追蹤資料，包括請求 URL、方法、回應狀態、開始時間及結束時間。

此應用程式也會使用. AWS SDK for Java為了測量這些調用，應用程序只需將 AWS SDK 相關的子模塊作為依賴關係，Java 的 X-Ray SDK 會自動檢測所有 AWS SDK 客戶端。

該應用程序Docker使用與Gradle Docker Image和Scorekeep API Dockerfile文件實例構建源代碼，以運行 Gradle 在其中生成的可執行 JAR。ENTRYPOINT

Example 使用碼頭通過搖籃碼頭圖像構建

```
docker run --rm -v /PATH/TO/SCOREKEEP_REPO/home/gradle/project -w /home/gradle/project
 gradle:4.3 gradle build
```

Example 碼頭檔入口點

```
ENTRYPOINT [ "sh", "-c", "java -Dserver.port=5000 -jar scorekeep-api-1.0.0.jar" ]
```

build.gradle 檔案會在編譯期間，透過將其宣告為依存項目，來從 Maven 下載軟體開發套件子模組。

Example build.gradle -- 相依性

```
...
dependencies {
    compile("org.springframework.boot:spring-boot-starter-web")
    testCompile('org.springframework.boot:spring-boot-starter-test')
    compile('com.amazonaws:aws-java-sdk-dynamodb')
    compile("com.amazonaws:aws-xray-recorder-sdk-core")
    compile("com.amazonaws:aws-xray-recorder-sdk-aws-sdk")
    compile("com.amazonaws:aws-xray-recorder-sdk-aws-sdk-instrumentor")
    ...
}
dependencyManagement {
    imports {
        mavenBom("com.amazonaws:aws-java-sdk-bom:1.11.67")
        mavenBom("com.amazonaws:aws-xray-recorder-sdk-bom:2.11.0")
```

```
    }
}
```

核心、 AWS SDK 和 AWS SDK 儀器子模組都是自動檢測使用 SDK 進行的任何下游呼叫所需的全部功能。 AWS

若要將原始區段資料轉送至 X-Ray API，X-Ray 精靈需要監聽 UDP 連接埠 2000 上的流量。為此，應用程序將 X-Ray 守護程序在一個容器中運行，該容器與 ECS 上的 Scorekeep 應用程序一起部署為附屬容器。如需詳細資訊，請查看 X-Ray 精靈主題。

Example ECS 任務定義中的 X-Ray 精靈容器定義

```
...
Resources:
  ScorekeepTaskDefinition:
    Type: AWS::ECS::TaskDefinition
    Properties:
      ContainerDefinitions:
      ...

      - Cpu: '256'
        Essential: true
        Image: amazon/aws-xray-daemon
        MemoryReservation: '128'
        Name: xray-daemon
        PortMappings:
          - ContainerPort: '2000'
            HostPort: '2000'
            Protocol: udp
      ...
```

Java 的 X-Ray SDK 提供了一個名為的類AWSXRay，它提供了全局記錄器TracingHandler，您可以使用它來檢測代碼。您可以設定全域記錄器來自訂為傳入 HTTP 呼叫建立區段的AWSXRayServletFilter。範例包括 WebConfig 類別中的靜態區塊，該區塊會使用外掛程式和抽樣規則設定全域記錄器。

Example src /主/爪/記分/ .java-記錄器 WebConfig

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.AWSXRayRecorderBuilder;
import com.amazonaws.xray.javax.servlet.AWSXRayServletFilter;
import com.amazonaws.xray.plugins.ECSPlugin;
```

```
import com.amazonaws.xray.plugins.EC2Plugin;
import com.amazonaws.xray.strategy.sampling.LocalizedSamplingStrategy;
...

@Configuration
public class WebConfig {
  ...

  static {
    AWSXRayRecorderBuilder builder = AWSXRayRecorderBuilder.standard().withPlugin(new
 ECSPlugin()).withPlugin(new EC2Plugin());

    URL ruleFile = WebConfig.class.getResource("/sampling-rules.json");
    builder.withSamplingStrategy(new LocalizedSamplingStrategy(ruleFile));

    AWSXRay.setGlobalRecorder(builder.build());
    ...

  }
}
```

此範例使用建立器從名為 `sampling-rules.json` 的檔案載入抽樣規則。抽樣規則會判斷軟體開發套件記錄傳入請求區段的速率。

Example src/main/java/resources/sampling-rules.json

```
{
  "version": 1,
  "rules": [
    {
      "description": "Resource creation.",
      "service_name": "*",
      "http_method": "POST",
      "url_path": "/api/*",
      "fixed_target": 1,
      "rate": 1.0
    },
    {
      "description": "Session polling.",
      "service_name": "*",
      "http_method": "GET",
      "url_path": "/api/session/*",
      "fixed_target": 0,
```

```
      "rate": 0.05
    },
    {
      "description": "Game polling.",
      "service_name": "*",
      "http_method": "GET",
      "url_path": "/api/game/*/*",
      "fixed_target": 0,
      "rate": 0.05
    },
    {
      "description": "State polling.",
      "service_name": "*",
      "http_method": "GET",
      "url_path": "/api/state/*/*/*",
      "fixed_target": 0,
      "rate": 0.05
    }
  ],
  "default": {
    "fixed_target": 1,
    "rate": 0.1
  }
}
```

抽樣規則檔案會定義四個自訂抽樣規則及預設規則。針對每個傳入請求，軟體開發套件會依照定義規則的順序評估自訂規則。軟體開發套件會套用第一個與請求方法、路徑及服務名稱相符的規則。針對 Scorekeep，第一個規則會透過以 1.0 的速率每秒套用一個請求的單一固定目標，或是在滿足固定目標之後套用 100% 的請求，來追補所有 POST 請求 (資源建立呼叫)。

其他三個自訂規則會在沒有指向工作階段、遊戲和狀態讀取 (GET 請求) 固定目標的情況下套用 5% 的速率 這會將前端為了確保內容處於最新狀態，每幾秒鐘自動發出定期呼叫的追蹤數量降至最低。針對所有其他請求，檔案會定義每秒單一請求的預設速率及 10% 的速率。

範例應用程式也會示範如何使用進階功能 (例如手動軟體開發套件用戶端檢測、建立額外子區段及傳出 HTTP 呼叫)。如需詳細資訊，請參閱 AWS X-Ray 範例應用。

選用：最低權限政策

Scorekeep ECS 容器會使用完整存取原則 (例如和) 存取資源。`AmazonSNSFullAccess` `AmazonDynamoDBFullAccess`使用完整存取原則不是生產應用程式的最佳作法。下列範例會更新

DynamoDB 身分與存取權管理政策，以改善應用程式的安全性。若要進一步了解 IAM 政策中的安全最佳實務，請參閱 AWS X-Ray 的身分識別與存取管理。

Example CF-資源管理模板 ECS 定義 TaskRole

```
ECSTaskRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          -
            Effect: "Allow"
            Principal:
              Service:
                - "ecs-tasks.amazonaws.com"
            Action:
              - "sts:AssumeRole"
      ManagedPolicyArns:
        - "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess"
        - "arn:aws:iam::aws:policy/AmazonSNSFullAccess"
        - "arn:aws:iam::aws:policy/AWSXrayFullAccess"
      RoleName: "scorekeepRole"
```

若要更新您的原則，請先識別 DynamoDB 資源的 ARN。然後，您可以在自訂 IAM 政策中使用 ARN。最後，您將該政策套用至您的執行個體設定檔。

若要識別動態資源的 ARN，請執行下列動作：

1. 開啟 DynamoDB 主控台。
2. 從左側導覽列選擇 [表格]。
3. 選擇任一項`scorekeep-*`來顯示表格詳細資訊頁面。
4. 在 [概觀] 索引標籤下，選擇 [其他資訊] 以展開區段並檢視 Amazon 資源名稱 (ARN)。複製這個值。
5. 將 ARN 插入下列 IAM 政策，並以您的特定區域`AWS_REGION`和帳戶 ID 取代和`AWS_ACCOUNT_ID`值。這個新策略只允許指定的處理行動，而不允許任何動作的`AmazonDynamoDBFullAccess`策略。

Example

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ScorekeepDynamoDB",
            "Effect": "Allow",
            "Action": [
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:Scan",
                "dynamodb:Query"
            ],
            "Resource": "arn:aws:dynamodb:<AWS_REGION>:<AWS_ACCOUNT_ID>:table/
scorekeep-*"
        }
    ]
}
```

應用程式建立的資料表遵循一致的命名慣例。您可以使用此`scorekeep-*`格式來表示所有「計分保留」表格。

變更您的 IAM 政策

1. 從 IAM 主控台開啟記分任務角色 (記分項目)。
2. 選擇`AmazonDynamoDBFullAccess`策略旁邊的核取方塊，然後選擇 [移除] 以移除此原則。
3. 選擇 [新增權限]，然後選取 [附加原則]，最後選取 [建立原則
4. 選擇 JSON 選項卡並粘貼到上面創建的策略中。
5. 選擇下一步:頁面底部的標籤。
6. 選擇「下一步」：頁面底部的「檢閱」。
7. 對於名稱，指定策略的名稱。
8. 選擇頁面底部的 [建立原則]。
9. 將新建立的原則附加至`scorekeepRole`角色。附加的原則可能需要幾分鐘的時間才會生效。

如果您已將新原則附加至scorekeepRole角色，則必須先將其中斷連結，然後再刪除 CloudFormation 堆疊，因為此連結的原則會封鎖堆疊遭到刪除。可透過刪除原則來自動卸離原則。

移除您的自訂 IAM 政策

1. 開啟 IAM 主控台。
2. 從左側導覽列選擇「策略」。
3. 搜尋您先前在本節中建立的自訂原則名稱，然後選擇原則名稱旁邊的圓鈕將其反白顯示。
4. 選擇動作下拉式清單，然後選擇刪除。
5. 輸入自訂原則的名稱，然後選擇 [刪除] 以確認刪除。這會自動將原則與scorekeepRole角色中斷連結。

清除

請依照下列步驟刪除「記分」應用程式資源：

> **Note**
>
> 如果您使用本教學課程的前一節建立並附加自訂原則，則必須先從中移除原則，scorekeepRole然後再刪除 CloudFormation 堆疊。

AWS Management Console

使用刪除範例應用程式 AWS Management Console

1. 開啟 CloudFormation 主控台
2. 選擇清單中scorekeep堆疊名稱旁邊的圓形按鈕，然後選擇 [刪除]。
3. CloudFormation 堆棧現在被刪除。堆疊狀態會持續幾DELETE_IN_PROGRESS分鐘，直到刪除所有資源為止。狀態會定期重新整理，或者您可以重新整理頁面。

AWS CLI

使用刪除範例應用程式 AWS CLI

1. 輸入下列 AWS CLI 指令以刪除 CloudFormation 堆疊：

```
aws cloudformation delete-stack --stack-name scorekeep
```

2. 等到 CloudFormation 堆棧不再存在，這將需要大約五分鐘。使用以下 AWS CLI 命令來檢查狀態：

```
aws cloudformation describe-stacks --stack-name scorekeep --query
 "Stacks[0].StackStatus"
```

後續步驟

在下一章中了解有關 X-Ray 的更多信息，AWS X-Ray 概念.

要檢測您自己的應用程序，請進一步了解 Java 的 X-Ray SDK 或其他 X-Ray SDK 之一：

- 適用於 Java 的 X-Ray SDK — AWS X-Ray 適用於的 SDK Java
- 適用於 Node.js 的 X-Ray SDK — AWS 適用於 Node.js 的 X-Ray SDK
- 適用於. NET 的 X-Ray SDK — AWS X-Ray SDK for .NET

若要在本機或上執行 X-Ray 精靈 AWS，請參閱AWS X-Ray 守護進。

若要參閱上的範例應用程式 GitHub，請參閱eb-java-scorekeep。

手動檢測 AWS SDK 用戶端

當您在構建依賴項中包含 AWS SDK 儀器子模塊時，SDK for Java 的 X-Ray AWS SDK 會自動檢測所有 SDK 客戶端。

若要停用自動用戶端檢測，您可以移除 Instrumentor 子模組。這可讓您手動檢測一些特定用戶端而忽略其他用戶端，或使用不同用戶端上的不同追蹤處理常式。

為了說明對檢測特定 AWS SDK 用戶端的支援，應用程式會將追蹤處理常式傳遞給`AmazonDynamoDBClientBuilder`使用者、遊戲和工作階段模型中的要求處理常式。此程式碼變更會告知 SDK 使用這些用戶端來檢測對 DynamoDB 的所有呼叫。

Example **`src/main/java/scorekeep/SessionModel.java`**— 手動 AWS SDK 客戶端檢測

```
import com.amazonaws.xray.AWSXRay;
```

```
import com.amazonaws.xray.handlers.TracingHandler;

public class SessionModel {
  private AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
        .withRegion(Constants.REGION)
        .withRequestHandlers(new TracingHandler(AWSXRay.getGlobalRecorder()))
        .build();
  private DynamoDBMapper mapper = new DynamoDBMapper(client);
```

如果您從項目依賴項中刪除 AWS SDK Instrumentor 子模塊，則只有手動檢測的 AWS SDK 客戶端會顯示在跟踪映射中。

建立其他子區段

在使用者模型類別中，應用程式會手動建立子區段以分組 saveUser 函數內發出的所有下游呼叫，並新增中繼資料。

Example **src/main/java/scorekeep/UserModel.java** - 自訂子區段

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Subsegment;
...
    public void saveUser(User user) {
    // Wrap in subsegment
    Subsegment subsegment = AWSXRay.beginSubsegment("## UserModel.saveUser");
    try {
      mapper.save(user);
    } catch (Exception e) {
      subsegment.addException(e);
      throw e;
    } finally {
      AWSXRay.endSubsegment();
    }
  }
```

記錄標註、中繼資料及使用者 ID

在遊戲模型類別中，每次將遊戲儲存到 DynamoDB 時，應用程式都會在中繼資料區塊中記錄Game物件。應用程式還會另外在標註中記錄遊戲 ID，用於篩選條件表達式。

Example **src/main/java/scorekeep/GameModel.java**— 註釋和元數據

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.entities.Segment;
import com.amazonaws.xray.entities.Subsegment;
...
  public void saveGame(Game game) throws SessionNotFoundException {
    // wrap in subsegment
    Subsegment subsegment = AWSXRay.beginSubsegment("## GameModel.saveGame");
    try {
      // check session
      String sessionId = game.getSession();
      if (sessionModel.loadSession(sessionId) == null ) {
        throw new SessionNotFoundException(sessionId);
      }
      Segment segment = AWSXRay.getCurrentSegment();
      subsegment.putMetadata("resources", "game", game);
      segment.putAnnotation("gameid", game.getId());
      mapper.save(game);
    } catch (Exception e) {
      subsegment.addException(e);
      throw e;
    } finally {
      AWSXRay.endSubsegment();
    }
  }
```

在移動控制器中，應用程序使用記錄用戶 ID `setUser`。區段會將使用者 ID 記錄在單獨的欄位中，並建立索引以用於搜尋。

Example src /主/爪/記分/. Java MoveController-用戶識別碼

```
import com.amazonaws.xray.AWSXRay;
...
  @RequestMapping(value="/{userId}", method=RequestMethod.POST)
  public Move newMove(@PathVariable String sessionId, @PathVariable String
 gameId, @PathVariable String userId, @RequestBody String move) throws
 SessionNotFoundException, GameNotFoundException, StateNotFoundException,
 RulesException {
    AWSXRay.getCurrentSegment().setUser(userId);
    return moveFactory.newMove(sessionId, gameId, userId, move);
  }
```

檢測傳出的 HTTP 呼叫

使用者工廠類別會顯示應用程式如何使用 Java 版本的 X-Ray SDK HTTPClientBuilder 來檢測傳出的 HTTP 呼叫。

Example **src/main/java/scorekeep/UserFactory.java**— 客戶端儀器

```
import com.amazonaws.xray.proxies.apache.http.HttpClientBuilder;

  public String randomName() throws IOException {
    CloseableHttpClient httpclient = HttpClientBuilder.create().build();
    HttpGet httpGet = new HttpGet("http://uinames.com/api/");
    CloseableHttpResponse response = httpclient.execute(httpGet);
    try {
      HttpEntity entity = response.getEntity();
      InputStream inputStream = entity.getContent();
      ObjectMapper mapper = new ObjectMapper();
      Map<String, String> jsonMap = mapper.readValue(inputStream, Map.class);
      String name = jsonMap.get("name");
      EntityUtils.consume(entity);
      return name;
    } finally {
      response.close();
    }
  }
```

若您目前使用 org.apache.http.impl.client.HttpClientBuilder，您可以直接使用一個用於 com.amazonaws.xray.proxies.apache.http.HttpClientBuilder 的項目將該類別的匯入陳述式切換出去。

檢測 PostgreSQL 資料庫的呼叫

所以此application-pgsql.properties檔案會將 X-Ray PostgreSQL 追蹤攔截程式新增至RdsWebConfig.java。

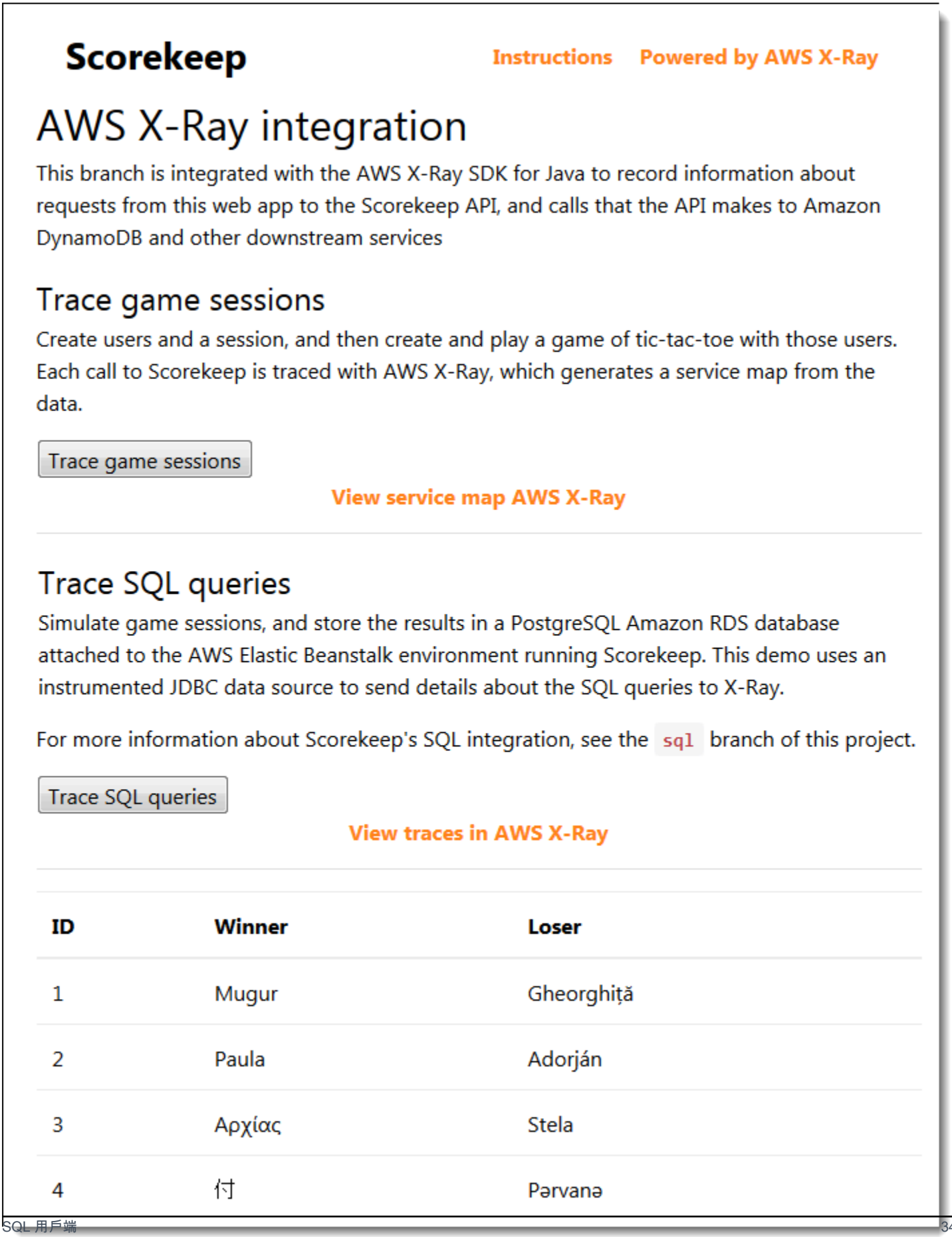
Example **application-pgsql.properties**— PostgreSQL 資料庫檢測

```
spring.datasource.continue-on-error=true
spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto=create-drop
```

```
spring.datasource.jdbc-interceptors=com.amazonaws.xray.sql.postgres.TracingInterceptor
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQL94Dialect
```

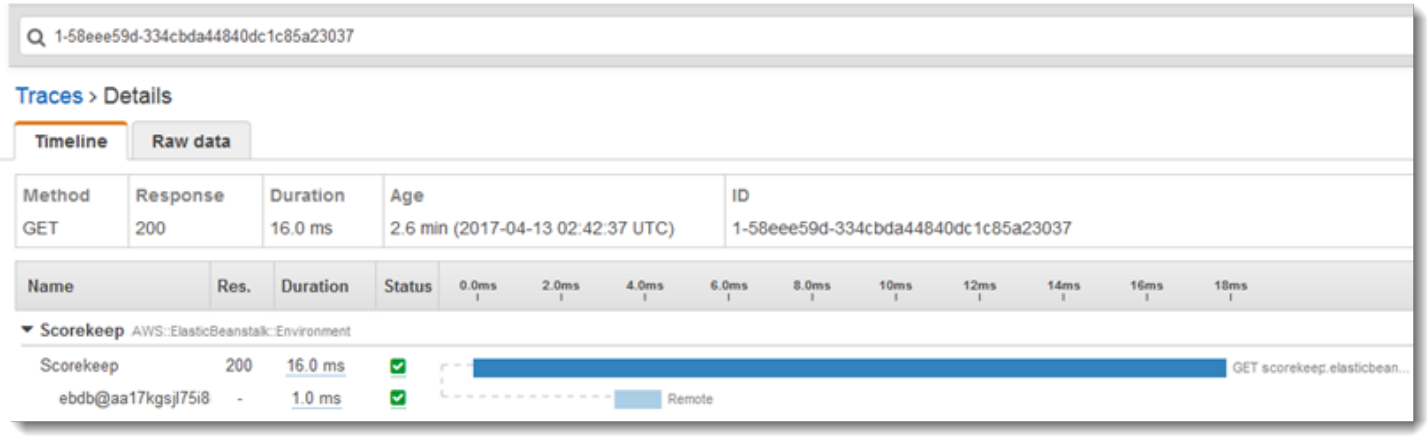
> **Note**
>
> 請參閱使用彈性手柄配置數據庫中的AWS Elastic Beanstalk開發人員指南，瞭解有關如何將 PostgreSQL 數據庫添加到應用程序環境的詳細信息。

中的 X-Ray 演示頁面`xray`分支包含一個示範，其會使用經檢測的資料來源產生追蹤，並顯示其產生的 SQL 查詢資訊。導覽至執行中應用程式的 `/#/xray` 路徑，或選擇導覽列中的 Powered by AWS X-Ray (採用 &xraylong; 技術)，以查看示範頁面。


Scorekeep
Instructions
Powered by AWS X-Ray
AWS X-Ray integration
This branch is integrated with the AWS X-Ray SDK for Java to record information about requests from this web app to the Scorekeep API, and calls that the API makes to Amazon DynamoDB and other downstream services
Trace game sessions
Create users and a session, and then create and play a game of tic-tac-toe with those users. Each call to Scorekeep is traced with AWS X-Ray, which generates a service map from the data.
Trace game sessions
View service map AWS X-Ray
Trace SQL queries
Simulate game sessions, and store the results in a PostgreSQL Amazon RDS database attached to the AWS Elastic Beanstalk environment running Scorekeep. This demo uses an instrumented JDBC data source to send details about the SQL queries to X-Ray.
For more information about Scorekeep's SQL integration, see the sql branch of this project.
Trace SQL queries
View traces in AWS X-Ray
ID
Winner
Loser
1
Mugur
Gheorghiță
2
Paula
Adorján
3
Αρχίας
Stela
4
付
Pərvanə

選擇 Trace SQL queries (追蹤 SQL 查詢) 以模擬遊戲工作階段，並將結果存放在連接的資料庫中。然後，選擇查看追蹤AWSX-Ray，以查看命中 API 的`/api/history`路由。

從清單中選擇其中一個追蹤以查看時間軸，包括 SQL 查詢。



儀表 AWS Lambda 功能

記分保持使用兩個 AWS Lambda 功能。第一個是來自 `lambda` 分支的 Node.js 函數，會為新使用者產生隨機名稱。當使用者在未輸入名稱的情況下建立工作階段時，應用程式會使用 AWS SDK for Java 呼叫名為 `random-name` 的函數。適用於 Java 的 X-Ray SDK 會記錄子區段中呼叫 Lambda 的相關資訊，就像使用已檢測的 AWS SDK 用戶端進行的任何其他呼叫一樣。

> **Note**
>
> 執行 `random-name` Lambda 函數需要在 Elastic Beanstalk 環境之外建立其他資源。如需詳細資訊和指示，請參閱讀我檔案：AWS Lambda 整合。

第二個函數 (`scorekeep-worker`) 是一種 Python 函數，會在 Scorekeep API 之外獨立執行。遊戲結束時，API 會將工作階段 ID 及遊戲 ID 寫入 SQS 佇列。工作者函數會從佇列讀取項目，並呼叫 Scorekeep API 來建構每個遊戲工作階段的完整記錄，以儲存在 Amazon S3 中。

Scorekeep 包括用於創建這兩個功能的 AWS CloudFormation 模板和腳本。因為您需要將 X-Ray SDK 與函數代碼捆綁在一起，因此模板創建了沒有任何代碼的函數。部署 Scorekeep 時，包含在 `.ebextensions` 資料夾中的組態檔會建立包含軟體開發套件的來源套件，並使用 AWS Command Line Interface更新函數程式碼及組態。

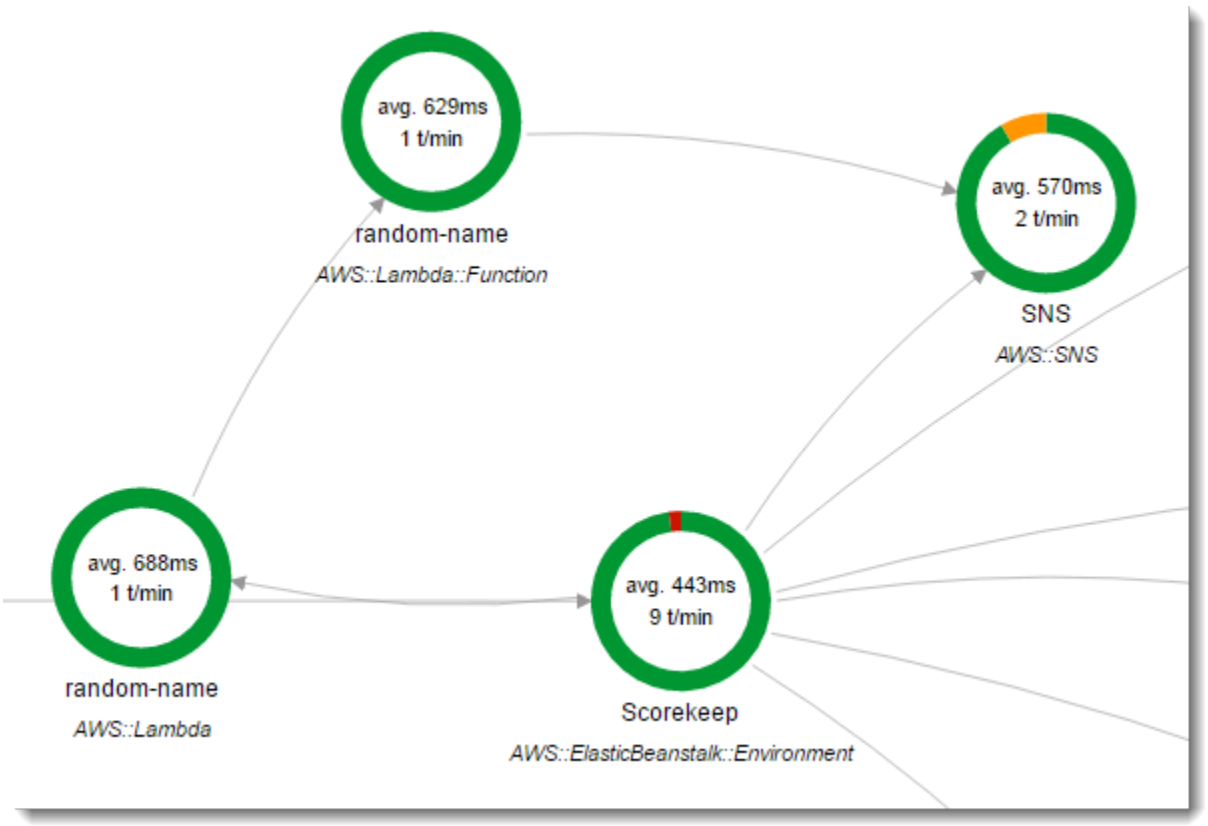
函數

隨機名稱

Scorekeep 會在使用者未登入或未指定任何使用者名稱啟動遊戲工作階段時呼叫隨機名稱函數。Lambda 處理呼叫時`random-name`，會讀取追蹤標頭，其中包含由 X-Ray SDK for Java 撰寫的追蹤識別碼和取樣決策。

對於每個取樣的請求，Lambda 都會執行 X-Ray 精靈並寫入兩個區段。第一個區段會記錄呼叫函數之 Lambda 的相關資訊。此區段包含與記分守記錄的子區段相同的資訊，但從 Lambda 的角度來看。第二個區段則代表函數進行的工作。

Lambda 會透過函數內容，將函數區段傳遞至 X-Ray SDK。檢測 Lambda 函數時，不會使用 SDK 為傳入請求建立區段。Lambda 提供區段，您可以使用 SDK 來檢測用戶端並撰寫子區段。



`random-name` 函數實作於 Node.js 中。它使用 Node.js JavaScript 中的 SDK 透過 Amazon SNS 傳送通知，而 Node.js 的 X-Ray 開發套件則使用 SDK 來檢測 AWS SDK 用戶端。為寫入標註，函數會使

用 AWSXRay.captureFunc 建立子區段，然後在受檢測函數中寫入標註。在 Lambda 中，您不能將註釋直接寫入函數段，只能將註釋寫入您建立的子區段。

Example **function/index.js** -- 隨機名稱 Lambda 函數

```
var AWSXRay = require('aws-xray-sdk-core');
var AWS = AWSXRay.captureAWS(require('aws-sdk'));

AWS.config.update({region: process.env.AWS_REGION});
var Chance = require('chance');

var myFunction = function(event, context, callback) {
  var sns = new AWS.SNS();
  var chance = new Chance();
  var userid = event.userid;
  var name = chance.first();

  AWSXRay.captureFunc('annotations', function(subsegment){
    subsegment.addAnnotation('Name', name);
    subsegment.addAnnotation('UserID', event.userid);
  });

  // Notify
  var params = {
    Message: 'Created randon name "' + name + '"" for user "' + userid + '".',
    Subject: 'New user: ' + name,
    TopicArn: process.env.TOPIC_ARN
  };
  sns.publish(params, function(err, data) {
    if (err) {
      console.log(err, err.stack);
      callback(err);
    }
    else {
      console.log(data);
      callback(null, {"name": name});
    }
  });
};

exports.handler = myFunction;
```

此函數會在您將簡易應用程式部署到 Elastic Beanstalk 時自動建立。該`xray`分支包括用於創建空白 Lambda 函數的腳本。`.ebextensions`資料夾中的組態檔案會在部署`npm install`期間建置函數套件，然後使用 AWS CLI 更新 Lambda 函數。

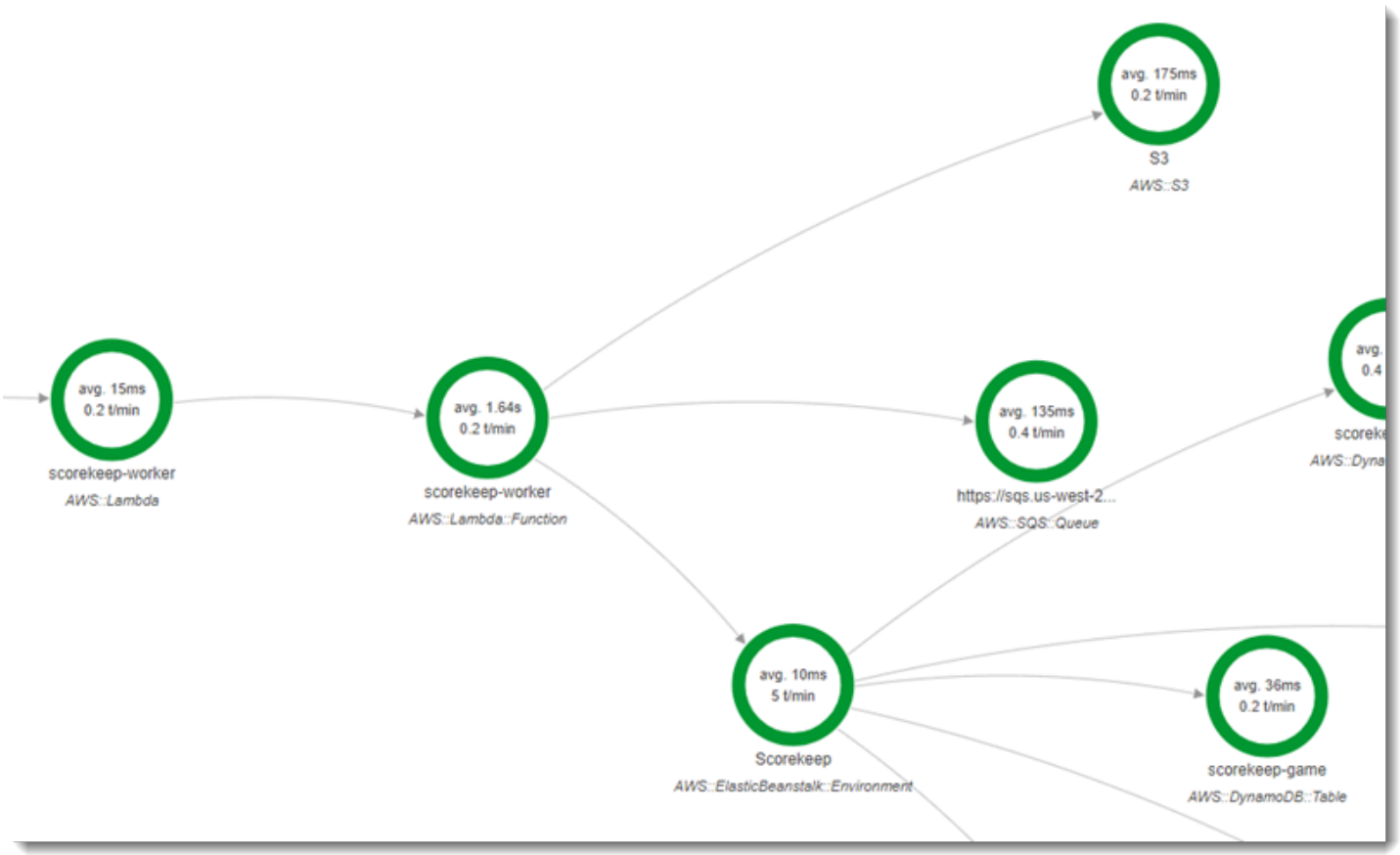
工作程序

受檢測工作者函數會在自己的分支 (`xray-worker`) 中提供，因為除非您先建立工作者函數及相關資源，否則它便無法執行。如需說明，請參閱分支讀我檔案。

該功能由捆綁的 Amazon CloudWatch 事件每 5 分鐘觸發一次。當它執行時，函數會從 Scorekeep 管理的 Amazon SQS 佇列中提取項目。每個訊息都包含完整遊戲的相關資訊。

工作者會從遊戲記錄參考的其他資料表提取遊戲記錄及文件。例如，DynamoDB 中的遊戲記錄包含在遊戲期間執行的動作清單。清單不包含移動本身，但包含存放在單獨資料表中移動的 ID。

工作階段及狀態也會以參考存放。這可避免遊戲資料表中的項目過大，但需要額外的呼叫才能取得所有遊戲相關資訊。工作者會解除參照所有這些項目，並在 Amazon S3 中將遊戲的完整記錄建構為單一文件。當您想要對資料進行分析時，可以使用 Amazon Athena 直接在 Amazon S3 中對其執行查詢，而無需執行大量讀取資料遷移即可將資料從 DynamoDB 中取出。



工作者函數在其位於 AWS Lambda內的組態中已啟用主動式追蹤。與隨機名稱函數不同，Worker 不會從已檢測的應用程式接收要求，因此 AWS Lambda 不會收到追蹤標頭。透過使用中追蹤，Lambda 會建立追蹤 ID 並做出取樣決策。

適用於 Python 的 X-Ray 開發套件只是函數頂端的幾行，可匯入 SDK 並執行其`patch_all`函數來修補它用來呼叫 Amazon SQS AWS SDK for Python (Boto) 和 Amazon S3 的 HTTP 用戶端和 HTTP 用戶端。工作者呼叫 Scorekeep API 時，軟體開發套件會將追蹤標頭新增至請求，來追蹤透過 API 進行的呼叫。

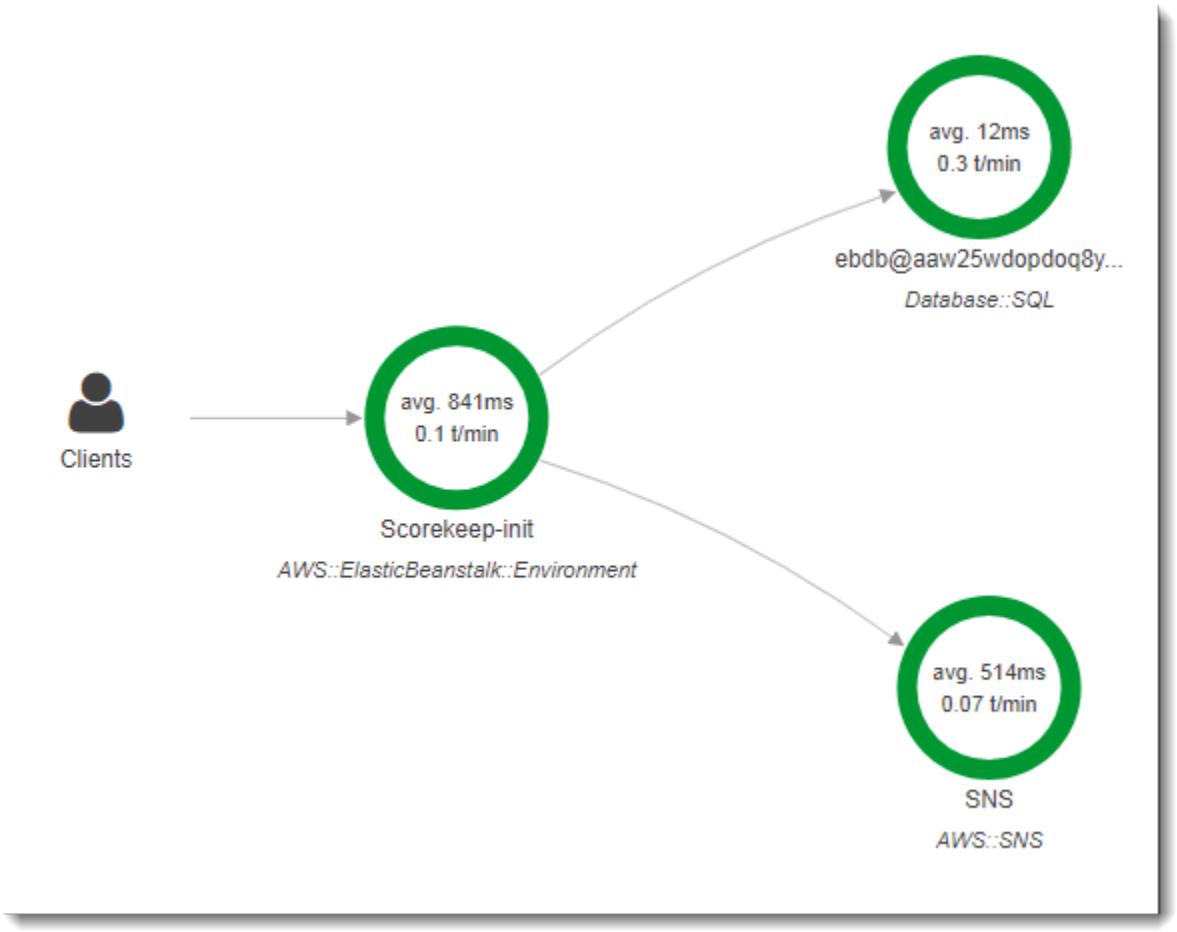
Example **_lambda/scorekeep-worker/scorekeep-worker.py** -- 工作者 Lambda 函數

```
import os
import boto3
import json
import requests
import time
from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch_all

patch_all()
queue_url = os.environ['WORKER_QUEUE']

def lambda_handler(event, context):
    # Create SQS client
    sqs = boto3.client('sqs')
    s3client = boto3.client('s3')

    # Receive message from SQS queue
    response = sqs.receive_message(
        QueueUrl=queue_url,
        AttributeNames=[
            'SentTimestamp'
        ],
        MaxNumberOfMessages=1,
        MessageAttributeNames=[
            'All'
        ],
        VisibilityTimeout=0,
        WaitTimeSeconds=0
    )
   ...
```

檢測啟動程式碼

適用於 Java 的 X-Ray 開發套件會自動為傳入的請求建立區段。只要請求是在範圍內，您即可使用經檢測的用戶端並記錄子區段，而不會出現問題。如果您嘗試在啟動程式碼中使用經檢測的用戶端，則會收到 SegmentNotFoundException。

啟動程式碼會在 Web 應用程式的標準請求/回應流程之外執行，因此您需要手動建立區段以檢測起始程式碼。Scorekeep 會在其 `WebConfig` 檔案中顯示啟動程式碼的檢測。起始程式碼會在啟動期間呼叫 SQL 資料庫和 Amazon SNS。



預設的`WebConfig`類別會建立通知的 Amazon SNS 訂。為了在使用 Amazon SNS 用户端時提供 X-Ray 開發套件寫入的區段，Scorement 會在使用 Amazon SNS 用户端時呼叫`beginSegment`和`endSegment`在全局記錄器上。

Example **`src/main/java/scorekeep/WebConfig.java`**— 檢測AWS起始程式碼中經開發套件用

```
AWSXRay.beginSegment("Scorekeep-init");
```

```
if ( System.getenv("NOTIFICATION_EMAIL") != null ){
  try { Sns.createSubscription(); }
  catch (Exception e ) {
    logger.warn("Failed to create subscription for email "+
 System.getenv("NOTIFICATION_EMAIL"));
  }
}
AWSXRay.endSegment();
```

InRdsWebConfig在連線 Amazon RDS 資料庫時 Scoreep 所使用的中，組態也會為 SQL 用户端建立區段，以讓 Hibernate 在啟動期間套用資料庫結構描述時使用。

Example **src/main/java/scorekeep/RdsWebConfig.java**— 起始程式碼中經檢測的 SQL 資料庫用户端

```
@PostConstruct
public void schemaExport() {
  EntityManagerFactoryImpl entityManagerFactoryImpl = (EntityManagerFactoryImpl)
 localContainerEntityManagerFactoryBean.getNativeEntityManagerFactory();
  SessionFactoryImplementor sessionFactoryImplementor =
 entityManagerFactoryImpl.getSessionFactory();
  StandardServiceRegistry standardServiceRegistry =
 sessionFactoryImplementor.getSessionFactoryOptions().getServiceRegistry();
  MetadataSources metadataSources = new MetadataSources(new
 BootstrapServiceRegistryBuilder().build());
  metadataSources.addAnnotatedClass(GameHistory.class);
  MetadataImplementor metadataImplementor = (MetadataImplementor)
 metadataSources.buildMetadata(standardServiceRegistry);
  SchemaExport schemaExport = new SchemaExport(standardServiceRegistry,
 metadataImplementor);

  AWSXRay.beginSegment("Scorekeep-init");
  schemaExport.create(true, true);
  AWSXRay.endSegment();
}
```

SchemaExport 會自動執行，並使用 SQL 用戶端。由於用戶端已經過檢測，因此 Scorekeep 必須覆寫預設的實作，並提供呼叫用戶端時軟體開發套件使用的區段。

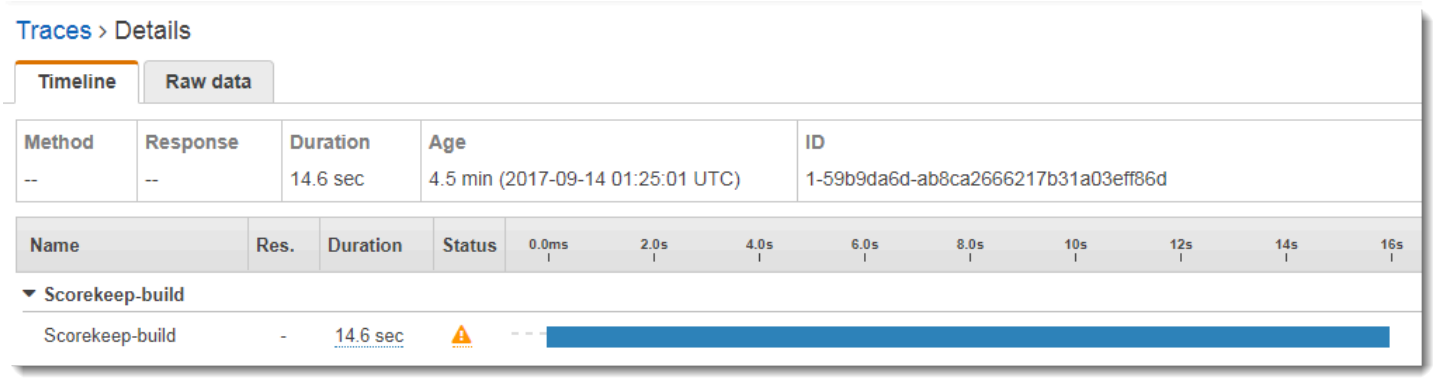
檢測指令碼

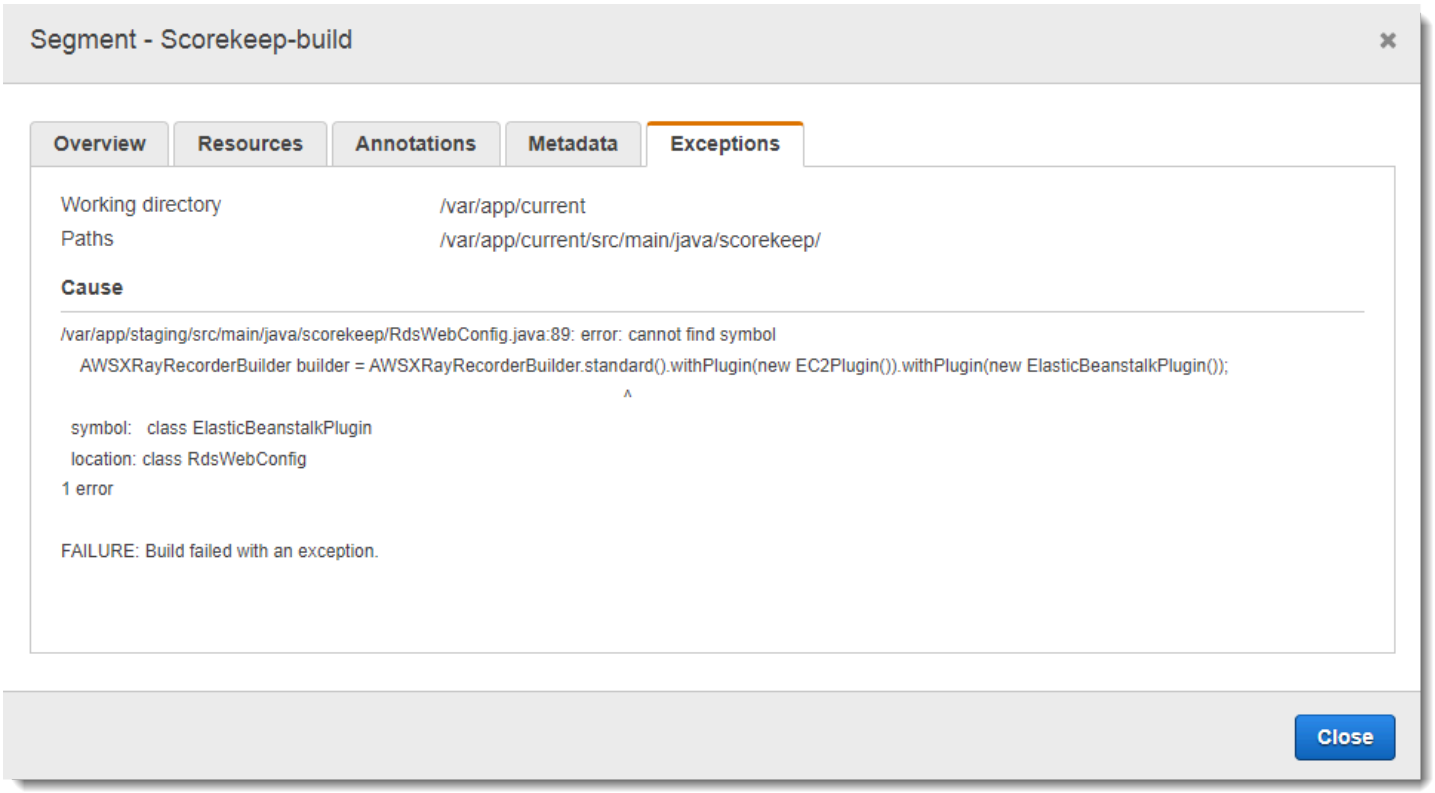
您也可以檢測不屬於您應用程式部分的程式碼。當 X-Ray 精靈執行時，它會將接收到的任何區段轉送至 X-Ray，即使這些區段不是由 X-Ray SDK 產生也一樣。Scorekeep 會使用自己的指令碼檢測在部署期間編譯應用程式的建置。

Example **bin/build.sh**— 已檢測的建置指令碼

```
SEGMENT=$(python bin/xray_start.py)
gradle build --quiet --stacktrace &> /var/log/gradle.log; GRADLE_RETURN=$?
if (( GRADLE_RETURN != 0 )); then
  echo "Gradle failed with exit status $GRADLE_RETURN" >&2
  python bin/xray_error.py "$SEGMENT" "$(cat /var/log/gradle.log)"
  exit 1
fi
python bin/xray_success.py "$SEGMENT"
```

xray_start.py、xray_error.py 和 xray_success.py 是簡易的 Python 指令碼，可建構區段物件，將它們轉換成 JSON 文件，並透過 UDP 傳送到精靈。如果 Gradle 構建失敗，則可以通過單擊 X-Ray 控制台跟踪映射中的記分保持構建節點來找到錯誤消息。





檢測 Web 應用程式用戶端

在[xray-cognito]分支機構中，Scorekeep 使用 Amazon Cognito 讓使用者能夠建立帳戶並使用該帳戶登入，以便從 Amazon Cognito 使用者集區擷取其使用者資訊。當使用者登入時，Scorekeep 會使用 Amazon Cognito 身分集區取得臨時 AWS 登入資料，以便與. AWS SDK for JavaScript

Identity Pool 是設定為可讓登入的使用者將追蹤資料寫入 AWS X-Ray。Web 應用程式會使用這些登入資料來記錄登入使用者的 ID、瀏覽器路徑和用戶端對 Scorekeep API 的呼叫檢視。

大部分工作都會在名稱為 `xray` 的服務類別中完成。此服務類別提供產生必要識別碼、建立進行中區段、完成區段，以及將區段文件傳送至 X-Ray API 的方法。

Example **public/xray.js**— 記錄和上傳細分

```
...
  service.beginSegment = function() {
    var segment = {};
    var traceId = '1-' + service.getHexTime() + '-' + service.getHexId(24);

    var id = service.getHexId(16);
    var startTime = service.getEpochTime();
```

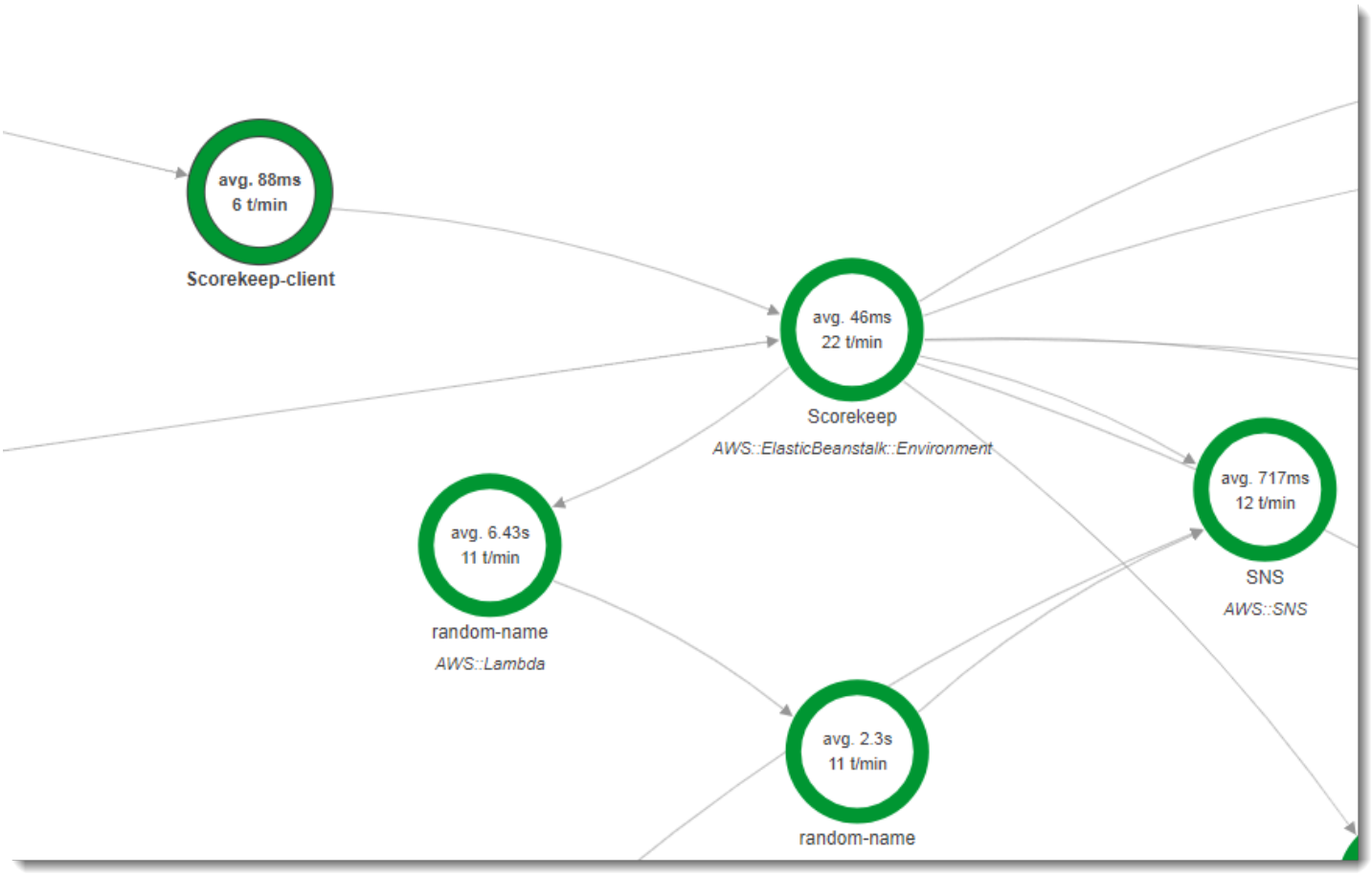
```
  segment.trace_id = traceId;
  segment.id = id;
  segment.start_time = startTime;
  segment.name = 'Scorekeep-client';
  segment.in_progress = true;
  segment.user =  sessionStorage['userid'];
  segment.http = {
    request: {
      url: window.location.href
    }
  };

  var documents = [];
  documents[0] = JSON.stringify(segment);
  service.putDocuments(documents);
  return segment;
}

service.endSegment = function(segment) {
  var endTime = service.getEpochTime();
  segment.end_time = endTime;
  segment.in_progress = false;
  var documents = [];
  documents[0] = JSON.stringify(segment);
  service.putDocuments(documents);
}

service.putDocuments = function(documents) {
  var xray = new AWS.XRay();
  var params = {
    TraceSegmentDocuments: documents
  };
  xray.putTraceSegments(params, function(err, data) {
    if (err) {
      console.log(err, err.stack);
    } else {
      console.log(data);
    }
  })
}
```

您可在資源服務的標頭和 `transformResponse` 函數中呼叫這些方法，而 Web 應用程式會使用這些資源服務來呼叫 Scorekeep API。若要將用戶端區段包含在與 API 產生的區段相同的追蹤中，Web 應用程式必須在 X-Ray SDK 可讀取的追蹤標頭 (`X-Amzn-Trace-Id`) 中包含追蹤 ID 和區段 ID。當檢測到的 Java 應用程式收到含有此標頭的要求時，Java 的 X-Ray SDK 會使用相同的追蹤識別碼，並使 Web 應用程式用戶端的區段成為其區段的父項。

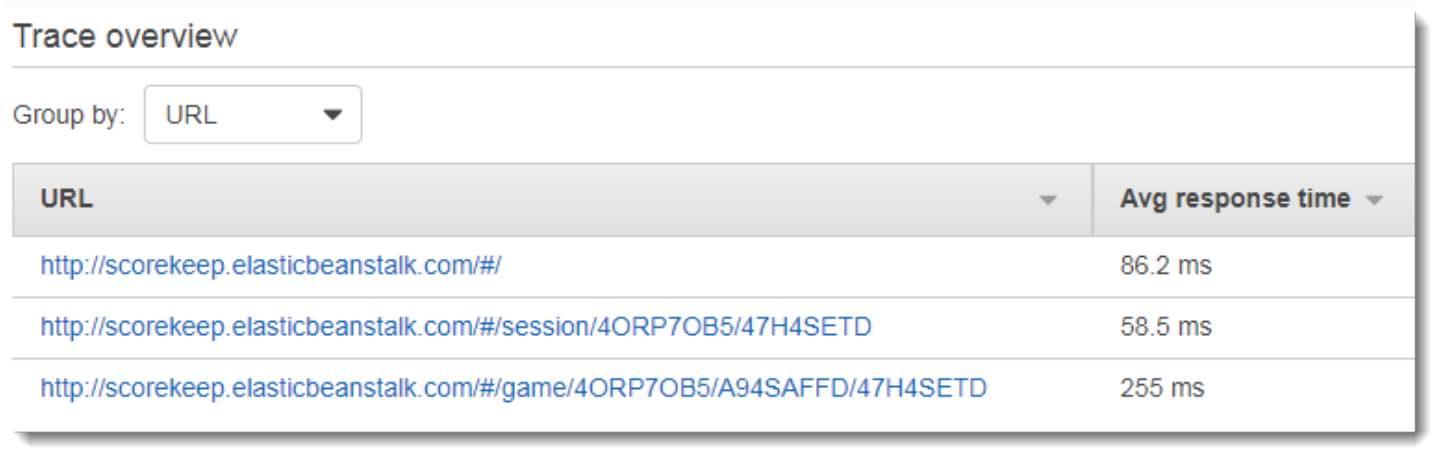
Example **public/app/services.js**— 記錄角度資源調用的段和寫入跟踪標題

```
var module = angular.module('scorekeep');
module.factory('SessionService', function($resource, api, XRay) {
  return $resource(api + 'session/:id', { id: '@_id' }, {
    segment: {},
    get: {
      method: 'GET',
      headers: {
        'X-Amzn-Trace-Id': function(config) {
          segment = XRay.beginSegment();
          return XRay.getTraceHeader(segment);
        }
      },
      transformResponse: function(data) {
        XRay.endSegment(segment);
        return angular.fromJson(data);
      },
    },
...
```

產生的追蹤對應包含 Web 應用程式用戶端的節點。



如果追蹤包括 Web 應用程式的區段，即會顯示使用者在瀏覽器中看到的 URL (路徑開頭為 `/#/`)。未使用用戶端檢測時，您只會取得 Web 應用程式呼叫之 API 資源的 URL (路徑開頭為 `/api/`)。



在工作者執行緒中使用受檢測用戶端

Scoreis 會使用工作者執行事，在使用者贏得遊戲時發佈通知到 Amazon SNS。發佈通知所花費的時間比其餘合併的請求操作更長，但不會影響用戶端或使用者。因此，若要改善回應時間，以非同步方式執行任務是一種好方法。

不過，適用於 Java 的 X-Ray 開發套件並不知道在執行建立時哪個區段為作用中。因此，當您嘗試在執行緒內使用經檢測的 AWS SDK for Java 用戶端時，會擲回 `SegmentNotFoundException` 而導致執行緒當機。

Example web-1.error.log

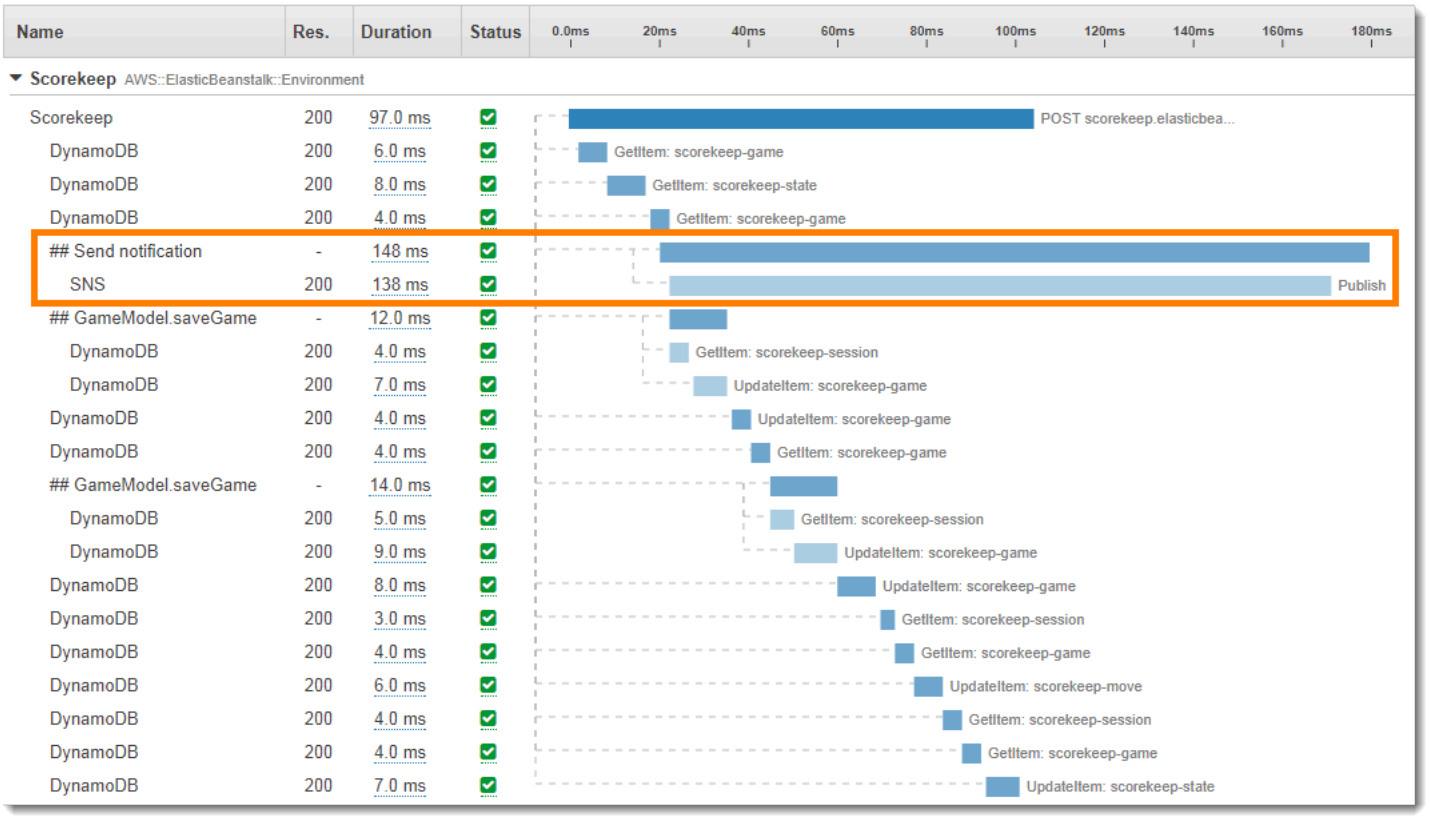
```
Exception in thread "Thread-2" com.amazonaws.xray.exceptions.SegmentNotFoundException:
 Failed to begin subsegment named 'AmazonSNS': segment cannot be found.
        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at
 sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
        at
 sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.ja
...
```

為瞭解決此問題，應用程序使用`GetTraceEntity`來獲取對主線程中線段的引用，`Entity.run()`來安全地運行工作線程代碼並訪問區段的上下文。

Example **src/main/java/scorekeep/MoveFactory.java**— 將追蹤內容傳遞至工作者執行

```
import com.amazonaws.xray.AWSXRay;
import com.amazonaws.xray.AWSXRayRecorder;
import com.amazonaws.xray.entities.Entity;
import com.amazonaws.xray.entities.Segment;
import com.amazonaws.xray.entities.Subsegment;
...
      Entity segment = recorder.getTraceEntity();
      Thread comm = new Thread() {
        public void run() {
         segment.run(() -> {
            Subsegment subsegment = AWSXRay.beginSubsegment("## Send notification");
            Sns.sendNotification("Scorekeep game completed", "Winner: " + userId);
            AWSXRay.endSubsegment();
          }
        }
```

由於在呼叫之前已解析請求，因此應用程式會為執行問題建立獨立的子區段。這可防止 X-Ray 開發套件在記錄 Amazon SNS 的回應之前結束區段。如果 Amazon SNS corenow 解析請求時沒有任何子區段開啟，可能會遺失的回應。



如需多執行緒的詳細資訊，請參閱[在多執行緒應用程式之間傳遞區段內容]。

疑難排 AWS X-Ray

本主題列出使用 X-Ray API、主控台或 SDK 時可能遇到的常見錯誤和問題。如果您發現未列在此處的問題，您可以使用此頁面上的 Feedback (意見回饋) 按鈕來報告。

章節

X-Ray 軌跡圖和跟踪詳細信息頁

如果您在使用 X-Ray 追蹤圖和追蹤詳細資料頁面時遇到問題，以下各節可以提供協助：

我沒有看到我的所有 CloudWatch 日誌

如何設定記錄檔，使其顯示在 X-Ray 追蹤對應和追蹤詳細資料頁面中，視服務而定。

- API Gateway 日誌只會在 API Gateway 中開啟記錄日誌時出現。

並非所有服務對應節點都支援檢視關聯的記錄檔。檢視下列節點類型的記錄檔：

- Lambda 背景
- Lambda 函數
- API Gateway 階段
- Amazon ECS 叢集
- Amazon ECS 實例
- Amazon ECS 服務
- Amazon ECS 任務
- Amazon EKS 叢集
- Amazon EKS 命名空間
- Amazon EKS 節點

- Amazon EKS 吊艙
- Amazon EKS 服務

我在 X-Ray 軌跡地圖上看不到所有警報

如果與該節點相關聯的任何警示處於 ALARM 狀態，X-Ray 追蹤對應只會顯示節點的警示圖示。

追蹤對應會使用下列邏輯將警示與節點相關聯：

- 如果節點代表 AWS 服務，則與該服務相關聯的命名空間的所有警示都會與該節點相關聯。例如，類型`AWS::Kinesis`的節點會與以 CloudWatch 命名空間中度量為基礎的所有警示連結`AWS/Kinesis`。
- 如果節點代表資 AWS 源，則會連結該特定資源上的警示。例如，名稱為「MyTable」`AWS::DynamoDB::Table` 的類型節點會連結至以具有命名空間之量度為基礎`AWS/DynamoDB`且`TableName`維度設定為的所有警示`MyTable`。
- 如果節點是未知類型 (由名稱周圍的虛線外框表示)，則不會有任何警示與該節點建立關聯。

我在軌跡地圖上看不到一些 AWS 資源

並非每個 AWS 資源都由專用節點表示。對於 AWS 服務的所有請求，某些服務會以單一節點表示。下列資源類型會以每個資源一個節點的方式顯示：

- `AWS::DynamoDB::Table`
- `AWS::Lambda::Function`

  Lambda 函數由兩個節點表示，一個用於 Lambda 容器，另一個用於函數。這有助於識別 Lambda 函數的冷啟動問題。Lambda 容器節點與警示和儀表板建立關聯的方式同 Lambda 函數節點一樣。
- `AWS::ApiGateway::Stage`
- `AWS::SQS::Queue`
- `AWS::SNS::Topic`

跟踪映射上的節點太多

請使用 X-Ray 群組，以將您的映射分成多個映射。如需詳細資訊，請參閱[搭配群組使用篩選條件表達式](#)。

適用於 Java 的 X-Ray SDK

錯誤：線程「線程 1」中的異常。 SegmentNotFoundException: 無法開始名為 'AmazonSNS' 的子區段：找不到區段。

此錯誤表示 X-Ray SDK 嘗試記錄撥出呼叫 AWS，但找不到開啟的區段。發生此問題的可能情況如下：

- 未設定 Servlet 篩選器 — X-Ray SDK 會使用名`AWSXRayServletFilter`為篩選器的傳入要求建立區段。設定 servlet 篩選條件來檢測傳入請求。
- 您在 servlet 程式碼之外使用已檢測的用戶端 — 如果您使用已檢測的用戶端在啟動程式碼或其他未回應傳入要求而執行的程式碼中進行呼叫，則必須手動建立區段。如需範例，請參閱 檢測啟動程式碼。
- 您正在工作執行緒中使用已檢測的用戶端 — 當您建立新執行緒時，X-Ray 記錄器會遺失對開啟區段的參考。您可以使用 `getTraceEntity` 和 `setTraceEntity` 方法來取得目前區段或子區段的參考 (`Entity`)，並將其傳遞給執行緒內部的記錄器。如需範例，請參閱在工作者執行緒中使用受檢測用戶端。

適用於 Node.js 的 X-Ray SDK

問題：「CLS 並未使用 Sequelize」

將 Node.js 命名空間的 X-Ray SDK 傳遞給使用該方法的封存。`cls`

```
var AWSXRay = require('aws-xray-sdk');
const Sequelize = require('sequelize');
Sequelize.cls = AWSXRay.getNamespace();
const sequelize = new Sequelize(...);
```

問題：「CLS 並未使用 Bluebird」

使用 `cls-bluebird` 來使 Bluebird 使用 CLS。

```
var AWSXRay = require('aws-xray-sdk');
var Promise = require('bluebird');
var clsBluebird = require('cls-bluebird');
clsBluebird(AWSXRay.getNamespace());
```

X-Ray 守護進程

問題：「精靈使用錯誤的登入資料」

常駐程式會使用 AWS SDK 載入認證。若您使用多個方法提供登入資料，便會使用優先順序最高的方法。如需詳細資訊，請參閱「執行精靈」。

中的安全性 AWS X-Ray

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，該架構專為滿足對安全性最敏感的組織的需求而打造。

安全是 AWS 與您之間共同承擔的責任。共同責任模型 將此描述為雲端的安全和雲端內的安全：

- 雲端的安全性 — AWS 負責保護 AWS 服務 中執行的基礎架構 AWS 雲端。 AWS 還為您提供可以安全使用的服務。第三方稽核人員定期檢測及驗證安全的效率也是我們 AWS 合規計劃的一部分。要了解適用於 X-Ray 的合規計劃，請參閱合規計劃範圍AWS 服務 中的。
- 雲端中的安全性 — 您的責任取決於您使用的資料。 AWS 服務 您也必須對資料敏感度、組織要求，以及適用法律和法規等其他因素負責。

本文件將協助您瞭解如何在使用 X-Ray 時套用共同責任模型。下列主題說明如何設定 X-Ray 以符合安全性和合規性目標。您還將學習如何使用其他可 AWS 服務 以幫助您監控和保護 X-Ray 資源的方法。

主題

- AWS X-Ray 中的資料保護
- 的身分識別與存取管理 AWS X-Ray
- 符合性驗證 AWS X-Ray
- AWS X-Ray 中的恢復能力
- AWS X-Ray 中的基礎設施安全

AWS X-Ray 中的資料保護

AWS X-Ray 一律會加密追蹤和相關靜態資料。當您需要針對合規性或內部需求稽核和停用加密金鑰時，您可以將 X-Ray 設定為使用 AWS Key Management Service (AWS KMS) 金鑰來加密資料。

X-Ray 提供了一個AWS 受管金鑰命名`aws/xray`。當您只想要稽核 AWS CloudTrail 中的金鑰使用情況而不需要管理金鑰本身時，請使用此金鑰。當您需要管理金鑰的存取權限或設定金鑰輪替時，您可以建立客戶管理的金鑰。

當您變更加密設定時，X-Ray 會花費一些時間來產生和傳播資料金鑰。雖然會處理新的金鑰，但 X-Ray 可能會使用新設定和舊設定的組合來加密資料。當您變更加密設定時，並不會重新加密現有資料。

> **Note**
>
> AWS KMSX-Ray 使用 KMS 金鑰加密或解密追蹤資料時收費。
>
> - 默認加密-免費。
> - AWS 受管金鑰— 支付金鑰使用費用。
> - 客戶管理的金鑰 — 支付金鑰儲存和使用費用。
>
> 詳情請參閱AWS Key Management Service定價。

> **Note**
>
> X-Ray 洞察通知會將事件傳送至 AmazonEventBridge，而 Amazon 目前不支援客戶受管金鑰。如需詳細資訊，請參閱 Amazon 中的資料保護EventBridge。

您必須擁有客戶管理金鑰的使用者層級存取權，才能將 X-Ray 設定為使用該金鑰，然後檢視加密的追蹤。如需詳細資訊，請參閱 用於加密的使用者許可。

CloudWatch console

將 X-Ray 設定為使用 KMS 金鑰使用CloudWatch主控台進行加密

1. 請登入AWS Management Console並開啟CloudWatch主控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在左側導覽窗格中選擇 [設定]。
3. 在 X-Ray 軌跡區段中，選擇加密下的檢視設定。
4. 在「加密組態」區段中選擇 「編輯」。
5. 選擇 [使用 KMS 金鑰]。
6. 從下拉式選單中選擇金鑰：
   - aws/X 射線 — 使用. AWS 受管金鑰
   - 金鑰別名 — 在您的帳戶中使用客戶管理的金鑰。
   - 手動輸入金鑰 ARN — 在不同的帳戶中使用客戶受管金鑰。在顯示欄位中，輸入金鑰的完整 Amazon Resource Name (ARN)。

7. 選擇更新加密。

X-Ray console

將 X-Ray 設定為使用 KMS 金鑰使用 X-Ray 主控台進行加密

1. 開啟 X-Ray 主控台。
2. 選擇 Encryption (加密)。
3. 選擇 [使用 KMS 金鑰]。
4. 從下拉式選單中選擇金鑰：
   - aws/X 射線 — 使用. AWS 受管金鑰
   - 金鑰別名 — 在您的帳戶中使用客戶管理的金鑰。
   - 手動輸入金鑰 ARN — 在不同的帳戶中使用客戶受管金鑰。在顯示欄位中，輸入金鑰的完整 Amazon Resource Name (ARN)。
5. 選擇 Apply (套用)。

> **Note**
>
> X-Ray 不支援非對稱 KMS 金鑰。

如果 X-Ray 無法存取您的加密金鑰，就會停止儲存資料。如果您的使用者無法存取 KMS 金鑰，或停用目前使用中的金鑰，就會發生這種情況。發生這種情況時，X-Ray 會在導覽列中顯示通知。

若要使用 X-Ray API 設定加密設定，請參閱使用 AWS X-Ray API 設定抽樣、分組和加密設定。

的身分識別與存取管理 AWS X-Ray

AWS Identity and Access Management (IAM) 可協助系統管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM 管理員控制哪些人可以驗證 (登入) 和授權 (具有權限) 以使用 X-Ray 資源。IAM 是您可以使用的 AWS 服務 ，無需額外付費。

主題

- 物件
- 使用身分驗證

- 使用政策管理存取權
- 如何與 IAM AWS X-Ray 搭配使用
- AWS X-Ray 以識別為基礎的原則範例
- 對 AWS X-Ray 身分與存取進行疑難排解

物件

您使用 AWS Identity and Access Management (IAM) 的方式會有所不同，具體取決於您在 X-Ray 中所做的工作。

服務使用者 — 如果您使用 X-Ray 服務執行工作，則管理員會為您提供所需的認證和權限。當您使用更多 X-Ray 特徵執行工作時，您可能需要其他權限。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果無法在 X-Ray 中存取特徵，請參閱對 AWS X-Ray 身分與存取進行疑難排解。

服務管理員 — 如果您負責公司的 X-Ray 資源，您可能可以完全使用 X-Ray。您的工作就是確定服務使用者應存取哪些 X-Ray 功能和資源。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要深入瞭解貴公司如何搭配 X-Ray 使用 IAM，請參閱如何與 IAM AWS X-Ray 搭配使用。

IAM 管理員 — 如果您是 IAM 管理員，可能需要瞭解如何撰寫政策以管理 X-Ray 存取權的詳細資訊。若要檢視可在 IAM 中使用的 X-Ray 身分型政策範例，請參閱。AWS X-Ray 以識別為基礎的原則範例

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以 IAM 使用者身分或假設 IAM 角色進行驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM 身分中心) 使用者、貴公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料都是聯合身分識別的範例。當您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以加密方式簽署您的要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的簽署 AWS API 請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。若要進一步了解，請參閱《AWS IAM Identity Center 使用者指南》中的多重要素驗證和《IAM 使用者指南》中的在 AWS中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常作業。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱《IAM 使用者指南》中的需要根使用者憑證的任務。

IAM 使用者和群組

IAM 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。建議您盡可能依賴暫時性憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#rotate-credentials中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分登入。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的過程變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。若要進一步了解，請參閱《IAM 使用者指南》中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 內部具有特定許可的身分。它類似 IAM 使用者，但不與特定的人員相關聯。您可以切換角色，在中暫時擔任 IAM 角色。 AWS Management Console 您可以透過呼叫 AWS CLI 或 AWS API 作業或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 – 若要向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需聯合角色的相關資訊，請參閱《IAM 使用者指南》中的為第三方身分供應商建立角色。如果您使用 IAM Identity Center，則

需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱《AWS IAM Identity Center 使用者指南》中的許可集。

- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的委託人) 存取您帳戶的資源。角色是授予跨帳户存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要了解跨帳戶存取角色和資源型政策間的差異，請參閱《IAM 使用者指南》中的 IAM 角色與資源類型政策的差異。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
  - 服務角色：服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可給 AWS 服務。
  - 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 — 您可以使用 IAM 角色來管理在 EC2 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。這是在 EC2 執行個體內存放存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並提供給其所有應用程式，請建立連接至執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得臨時性憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

若要了解是否要使用 IAM 角色或 IAM 使用者，請參閱《IAM 使用者指南》中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色

工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以 JSON 文件的形式儲存在中。如需 JSON 政策文件結構和內容的相關資訊，請參閱《IAM 使用者指南》中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以連接到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。如需瞭解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些委託人 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 和 Amazon VPC 是支援 ACL 的服務範例。 AWS WAF若要進一步了解 ACL，請參閱《Amazon Simple Storage Service 開發人員指南》中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限：許可界限是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限的限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可邊界的相關資訊，請參閱《IAM 使用者指南》中的 IAM 實體許可邊界。
- 服務控制策略（SCP）— SCP 是 JSON 策略，用於指定中組織或組織單位（OU）的最大權限。AWS Organizations AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 限制成員帳戶中實體的權限，包括每個 AWS 帳戶根使用者帳戶。如需 Organizations 和 SCP 的相關資訊，請參閱《AWS Organizations 使用者指南》中的 SCP 運作方式。
- 工作階段政策：工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合身分使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱《IAM 使用者指南》中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。要了解如何在涉及多個政策類型時 AWS 確定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

如何與 IAM AWS X-Ray 搭配使用

在您使用 IAM 管理 X-Ray 的存取權限之前，您應該瞭解哪些 IAM 功能可用於 X-Ray。若要深入瞭解 X-Ray 和其他人如何 AWS 服務 使用 IAM AWS 服務 ，請參閱 IAM 使用者指南中的使用 IAM。

您可以使用 AWS Identity and Access Management (IAM) 將 X-Ray 許可授與帳戶中的使用者和運算資源。無論使用者使用哪個用戶端 (主控台、 AWS SDK AWS CLI)，IAM 都能在 API 層級控制對 X-Ray 服務的存取，以統一強制執行許可。

若要使用 X-Ray 主控台檢視軌跡地圖和區段，您只需要讀取權限。若要啟用主控台存取，請將 `AWSXrayReadOnlyAccess` 受管政策新增到您的 IAM 使用者。

若要進行本機開發和測試，請建立具有讀取和寫入權限的 IAM 角色。假設角色並儲存角色的暫時認證。您可以將這些認證與 X-Ray 精靈 AWS CLI、和 AWS SDK 搭配使用。 AWS CLI如需詳細資訊，請參閱搭配使用暫時安全登入資料。

若要將已檢測的應用程式部署到 AWS，請建立具有寫入權限的 IAM 角色，並將其指派給執行應用程式的資源。 AWSXRayDaemonWriteAccess包括上傳追蹤的權限，以及一些讀取權限，以及支援使用取樣規則。

讀取和寫入政策不包含設定加密金鑰設定及抽樣規則的許可。使用 AWSXrayFullAccess 存取這些設定，或是在自訂政策中新增組態 API。針對使用您所建立客戶受管金鑰的加密和解密，您也需要使用金鑰的許可。

主題

- 基於 X-Ray 身份的政策
- X-Ray 資源型政策
- 基於 X-Ray 標籤的授權
- 於本機執行您的應用程式
- 執行您的應用程式 AWS
- 用於加密的使用者許可

基於 X-Ray 身份的政策

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。X-Ray 支援特定動作、資源和條件鍵。若要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 JSON 政策元素參考。

動作

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON 政策的 Action 元素描述您可以用來允許或拒絕政策中存取的動作。原則動作通常與關聯的 AWS API 作業具有相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些操作需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授與執行相關聯操作的許可。

X-Ray 中的原則動作會在動作之前使用下列前置詞：xray:例如，若要授與某人使用 X-Ray GetGroup API 作業擷取群組資源詳細資料的權限，您可以將該xray:GetGroup動作包含在他們的政

策中。政策陳述式必須包含 Action 或 NotAction 元素。X-Ray 會定義自己的一組動作，說明您可以使用此服務執行的工作。

若要在單一陳述式中指定多個動作，請用逗號分隔，如下所示：

```
"Action": [
      "xray:action1",
      "xray:action2"
```

您也可以使用萬用字元 (*) 來指定多個動作。例如，若要指定開頭是 Get 文字的所有動作，請包含以下動作：

```
"Action": "xray:Get*"
```

若要查看 X-Ray 動作清單，請參閱《IAM 使用者指南》 AWS X-Ray中的 「定義的動作」。

資源

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Resource JSON 政策元素可指定要套用動作的物件。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出作業)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

您可以使用 IAM 政策來控制對資源的存取。針對支援資源層級許可的動作，您可以使用 Amazon Resource Name (ARN) 來識別要套用政策的資源。

所有 X-Ray 動作都可以在 IAM 政策中使用，授予或拒絕使用者使用該動作的權限。不過，並非所有 X-Ray 動作都支援資源層級權限，這可讓您指定可以執行動作的資源。

對於不支援資源層級許可的動作，您必須使用 "*" 做為資源。

下列 X-Ray 動作支援資源層級權限：

- CreateGroup

- GetGroup
- UpdateGroup
- DeleteGroup
- CreateSamplingRule
- UpdateSamplingRule
- DeleteSamplingRule

以下是 CreateGroup 動作的身分型許可政策範例。此範例示範使用與有唯一 ID 之群組名稱 local-users 有關的 ARN 做為萬用字元。群組建立時產生的唯一 ID，所以無法在政策中事先預測。使用 GetGroup、UpdateGroup 或 DeleteGroup 時，您可以將此定義為萬用字元或確切的 ARN，包括 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:CreateGroup"
            ],
            "Resource": [
                "arn:aws:xray:eu-west-1:123456789012:group/local-users/*"
            ]
        }
    ]
}
```

以下是 CreateSamplingRule 動作的身分型許可政策範例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:CreateSamplingRule"
            ],
            "Resource": [
                "arn:aws:xray:eu-west-1:123456789012:sampling-rule/base-scorekeep"
```

```
            ]
        }
    ]
}
```

> **Note**
>
> 依其名稱定義之取樣規則的 ARN。不像群組 ARN，取樣規則沒有唯一產生的 ID。

若要查看 X-Ray 資源類型及其 ARN 的清單，請參閱《IAM 使用者指南》 AWS X-Ray中的「定義資源」。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱 AWS X-Ray定義的動作。

條件索引鍵

X-Ray 不提供任何服務特定的條件金鑰，但它確實支援使用某些全域條件金鑰。若要查看所有 AWS 全域條件金鑰，請參閱 IAM 使用者指南中的AWS 全域條件內容金鑰。

範例

若要檢視 X-Ray 身分型原則的範例，請參閱。AWS X-Ray 以識別為基礎的原則範例

X-Ray 資源型政策

X-Ray 支援以資源為基礎的政策，用於目前和 future 的 AWS 服務 整合，例如 Amazon SNS 主動追蹤。基於 X-Ray 資源的策略可以由其他 AWS Management Console S，或通過 AWS SDK 或 CLI 進行更新。例如，Amazon SNS 主控台嘗試自動設定以資源為基礎的政策，以將追蹤傳送至 X-Ray。下列政策文件提供手動設定 X-Ray 資源型政策的範例。

Example 適用於 Amazon SNS 主動追蹤的 X-Ray 資源型政策範例

此範例政策文件指定 Amazon SNS 將追蹤資料傳送至 X-Ray 所需的許可：

```
{
    Version: "2012-10-17",
    Statement: [
      {
        Sid: "SNSAccess",
        Effect: Allow,
        Principal: {
          Service: "sns.amazonaws.com",
```

```
        },
        Action: [
          "xray:PutTraceSegments",
          "xray:GetSamplingRules",
          "xray:GetSamplingTargets"
        ],
        Resource: "*",
        Condition: {
          StringEquals: {
            "aws:SourceAccount": "account-id"
          },
          StringLike: {
            "aws:SourceArn": "arn:partition:sns:region:account-id:topic-name"
          }
        }
      }
    ]
  }
```

使用 CLI 建立以資源為基礎的政策，授與 Amazon SNS 將追蹤資料傳送至 X-Ray 的許可：

```
aws xray put-resource-policy --policy-name MyResourcePolicy --policy-document
 '{ "Version": "2012-10-17", "Statement": [ { "Sid": "SNSAccess", "Effect": "Allow",
 "Principal": { "Service": "sns.amazonaws.com" }, "Action": [ "xray:PutTraceSegments",
 "xray:GetSamplingRules", "xray:GetSamplingTargets" ], "Resource": "*",
 "Condition": { "StringEquals": { "aws:SourceAccount": "account-id" }, "StringLike":
 { "aws:SourceArn": "arn:partition:sns:region:account-id:topic-name" } } } ] }'
```

若要使用這些範例，請將*partition*、*region*、*account-id*、和取代為您*topic-name*的特定 AWS 分割區、區域、帳戶 ID 和 Amazon SNS 主題名稱。若要授予所有 Amazon SNS 主題將追蹤資料傳送至 X-Ray 的權限，請將主題名稱取代為*。

基於 X-Ray 標籤的授權

您可以將標籤附加到 X-Ray 群組或取樣規則，或將請求中的標籤傳遞給 X-Ray。若要根據標籤控制存取，請使用 `xray:ResourceTag/`*key-name*、`aws:RequestTag/`*key-name* 或 `aws:TagKeys` 條件金鑰，在政策的條件元素中，提供標籤資訊。若要取得有關為 X-Ray 資源貼標籤的更多資訊標籤 X-Ray 取樣規則和羣組，

若要檢視身分型原則範例，以根據該資源上的標籤來限制存取資源，請參閱根據標籤管理 X-Ray 群組和取樣規則的存取。

於本機執行您的應用程式

已檢測的應用程式會將追蹤資料傳送至 X-Ray 精靈。常駐程式會緩衝區段文件，並將它們批次上傳至 X-Ray 服務。精靈需要寫入權限，才能將追蹤資料和遙測上傳至 X-Ray 服務。

當您在本機執行精靈時，請建立 IAM 角色、擔任該角色並將臨時登入資料儲存在環境變數中，或儲存在使用者資料夾.aws中名為的資料夾中的檔案中。credentials AWS CLI如需詳細資訊，請參閱搭配使用暫時安全登入資料。

Example ~/.aws/credentials

```
[default]
aws_access_key_id={access key ID}
aws_secret_access_key={access key}
aws_session_token={AWS session token}
```

如果您已經設定了與 AWS SDK 搭配使用的認證 AWS CLI，或者精靈可以使用這些認證。若有多個可用的描述檔，精靈會使用預設描述檔。

執行您的應用程式 AWS

在上執行應用程式時 AWS，請使用角色將權限授與執行精靈的 Amazon EC2 執行個體或 Lambda 函數。

- 亞馬遜彈性運算雲端 (Amazon EC2) — 建立 IAM 角色，並將其作為執行個體設定檔連接到 EC2 執行個體。
- Amazon Elastic Container Service (Amazon ECS) — 建立 IAM 角色，並將其作為容器執行個體 IAM 角色附加到容器執行個體。
- AWS Elastic Beanstalk （彈性魔豆莖）— Elastic Beanstalk 在其默認實例配置文件中包含 X-Ray 權限。您可以使用預設執行個體描述檔，或是將寫入許可新增到自訂執行個體描述檔。
- AWS Lambda (Lambda) — 將寫入權限新增至函數的執行角色。

建立與 X-Ray 配合使用的角色

1. 開啟 IAM 主控台。
2. 選擇角色。
3. 選擇 Create New Role (建立新角色)。

4. 在 Role Name (角色名稱) 中，輸入 **xray-application**。選擇 Next Step (後續步驟)。
5. 針對 Role Type (角色類型)，選擇 Amazon EC2。
6. 附加下列受管理政策，讓您的應用程式存取權限 AWS 服務：

   - AWSXRayDaemonWriteAccess— 授予 X-Ray 守護程序上傳追蹤資料的權限。

   如果您的應用程式使用 AWS SDK 存取其他服務，請新增授與這些服務存取權的政策。
7. 選擇 Next Step (後續步驟)。
8. 選擇建立角色。

用於加密的使用者許可

X-Ray 預設會加密所有追蹤資料，您可以將其設定為使用您管理的金鑰。如果您選擇 AWS Key Management Service 客戶管理的金鑰，則必須確保金鑰的存取政策允許您授與 X-Ray 的權限，以使用該金鑰進行加密。您帳戶中的其他使用者也需要存取金鑰，才能在 X-Ray 主控台中檢視加密的追蹤資料。

對於客戶管理的金鑰，請使用允許執行下列動作的存取原則來設定金鑰：

- 在 X-Ray 中設定金鑰的使用者有權呼叫`kms:CreateGrant`和。`kms:DescribeKey`
- 能夠存取加密追蹤資料的使用者必須具備呼叫 `kms:Decrypt` 的許可。

當您在 IAM 主控台的金鑰設定區段中將使用者新增至金鑰使用者群組時，他們會擁有這兩項作業的權限。只需要在金鑰原則上設定權限，因此您不需要使用者、群組或角色的任何 AWS KMS 權限。如需詳細資訊，請參閱AWS KMS 開發人員指南中的使用金鑰原則。

對於預設加密，或者如果您選擇 AWS 託管 CMK (`aws/xray`)，權限取決於誰可以存取 X-Ray API。任何擁有 `PutEncryptionConfig` 存取權的人員 (包含在 `AWSXrayFullAccess` 中) 都可以變更加密組態。若要防止使用者變更加密金鑰，請不要給予他們使用 `PutEncryptionConfig` 的許可。

AWS X-Ray 以識別為基礎的原則範例

依預設，使用者和角色沒有建立或修改 X-Ray 資源的權限。他們也無法使用 AWS Management Console AWS CLI、或 AWS API 執行工作。管理員必須建立 IAM 政策，授與使用者和角色在指定資源上執行特定 API 操作所需的許可。管理員接著必須將這些政策連接至需要這些許可的使用者或群組。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的在 JSON 索引標籤上建立政策。

主題

政策最佳實務

以身分識別為基礎的政策會決定某人是否可以建立、存取或刪除您帳戶中的 X-Ray 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管原則並邁向最低權限權限 — 若要開始將權限授與使用者和工作負載，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們可用在您的 AWS 帳戶. 建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需詳細資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低權限許可：設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需如何使用 IAM 套用許可的詳細資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。您也可以使用條件來授與對服務動作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您編寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM Access Analyzer 政策驗證。
- 需要多因素身份驗證 (MFA) — 如果您的案例需要 IAM 使用者或根使用者 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的設定 MFA 保護的 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

使用 X-Ray 控制台

若要存取 AWS X-Ray 主控台，您必須擁有最少一組權限。這些權限必須允許您列出並檢視您的 AWS 帳戶. 如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體（使用者或角色）而言，主控台就無法如預期運作。

若要確保這些實體仍可使用 X-Ray 主控台，請將AWSXRayReadOnlyAccess AWS 受管理的原則附加至實體。此政策在適用於 X-Ray 的 IAM 受管政策中有更詳細的說明。如需詳細資訊，請參閱《IAM 使用者指南》中的新增許可到使用者。

您不需要為僅對 AWS CLI 或 AWS API 進行呼叫的使用者允許最低主控台權限。反之，只需允許存取符合您嘗試執行之 API 操作的動作就可以了。

允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此原則包含在主控台上或以程式設計方式使用 AWS CLI 或 AWS API 完成此動作的權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
```

```
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

根據標籤管理 X-Ray 群組和取樣規則的存取

您可以使用身分型原則中的條件，根據標記控制對 X-Ray 群組的存取和取樣規則。下列範例原則可用來拒絕使用者角色具有建立、刪除或更新群組的權`stage:prod`限`stage:preprod`。如需標籤 X-Ray 取樣規則和群組的更多資訊，請參閱〈〉標籤 X-Ray 取樣規則和羣組。

若要拒絕使用者存取建立、更新或刪除具有標記的群組`stage:preprod`，`stage:prod`或者，請為使用者指定具有類似下列原則的角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllXRay",
            "Effect": "Allow",
            "Action": "xray:*",
            "Resource": "*"
        },
        {
            "Sid": "DenyCreateGroupWithStage",
            "Effect": "Deny",
            "Action": [
                "xray:CreateGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/stage": [
                        "preprod",
                        "prod"
                    ]
                }
```

```
            }
        },
        {
            "Sid": "DenyUpdateGroupWithStage",
            "Effect": "Deny",
            "Action": [
                "xray:UpdateGroup",
                "xray:DeleteGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stage": [
                        "preprod",
                        "prod"
                    ]
                }
            }
        }
    ]
}
```

若要拒絕建立取樣規則，請使用aws:RequestTag來指示無法在建立要求中傳遞的標籤。若要拒絕更新或刪除取樣規則，請使aws:ResourceTag用根據這些資源上的標籤來拒絕動作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllXRay",
            "Effect": "Allow",
            "Action": "xray:*",
            "Resource": "*"
        },
        {
            "Sid": "DenyCreateSamplingRuleWithStage",
            "Effect": "Deny",
            "Action": "xray:CreateSamplingRule",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/stage": [
                        "preprod",
```

```
                        "prod"
                    ]
                }
            }
        },
        {
            "Sid": "DenyUpdateSamplingRuleWithStage",
            "Effect": "Deny",
            "Action": [
                "xray:UpdateSamplingRule",
                "xray:DeleteSamplingRule"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stage": [
                        "preprod",
                        "prod"
                    ]
                }
            }
        }
    ]
}
```

您可以將這些策略附加到您帳戶中的使用者 (或將其合併到單一策略中，然後附加策略)。若要讓使用者變更群組或取樣規則，群組或取樣規則不得加上標記`stage=prepod`或`stage=prod`。條件標籤鍵 `Stage` 符合 `Stage` 和 `stage`，因為條件索引鍵名稱不區分大小寫。如需有關條件區塊的詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。

具有已附加下列策略之角色的使用者無法將標籤新增`role:admin`至資源，也無法從與其`role:admin`相關聯的資源中移除標籤。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllXRay",
            "Effect": "Allow",
            "Action": "xray:*",
            "Resource": "*"
        },
        {
```

```
            "Sid": "DenyRequestTagAdmin",
            "Effect": "Deny",
            "Action": "xray:TagResource",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/role": "admin"
                }
            }
        },
        {
            "Sid": "DenyResourceTagAdmin",
            "Effect": "Deny",
            "Action": "xray:UntagResource",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/role": "admin"
                }
            }
        }
    ]
}
```

適用於 X-Ray 的 IAM 受管政策

為了簡化授與許可，IAM 支援每個服務的受管政策。當服務發行新的 API 時，可以使用新的權限來更新這些受管理的策略。 AWS X-Ray 針對唯讀、唯寫和管理員使用案例提供受管理的原則。

- `AWSXrayReadOnlyAccess`— 讀取使用 X-Ray 控制台或 AWS SDK 從 X-Ray API 獲取跟踪數據，跟踪地圖，見解和 X-Ray 配置的權限。 AWS CLI包括可觀察性存取管理員 (OAM) `oam:ListSinks` 和`oam:ListAttachedSinks`權限，可讓主控台檢視從來源帳戶共用的追蹤，做為CloudWatch 跨帳戶觀察性的一部分。`BatchGetTraceSummaryById`和 `GetDistinctTraceGraphs` API 動作不打算由您的程式碼呼叫，也不會包含在 AWS CLI 和 AWS SDK 中。

  ```
  {
      "Version": "2012-10-17",
      "Statement": [
          {
              "Effect": "Allow",
              "Action": [
  ```

```
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets",
                "xray:GetSamplingStatisticSummaries",
                "xray:BatchGetTraces",
                "xray:BatchGetTraceSummaryById",
                "xray:GetDistinctTraceGraphs",
                "xray:GetServiceGraph",
                "xray:GetTraceGraph",
                "xray:GetTraceSummaries",
                "xray:GetGroups",
                "xray:GetGroup",
                "xray:ListTagsForResource",
                "xray:ListResourcePolicies",
                "xray:GetTimeSeriesServiceStatistics",
                "xray:GetInsightSummaries",
                "xray:GetInsight",
                "xray:GetInsightEvents",
                "xray:GetInsightImpactGraph",
                "oam:ListSinks"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:ListAttachedLinks"
            ],
            "Resource": "arn:aws:oam:*:*:sink/*"
        }

}
```

- `AWSXRayDaemonWriteAccess`— 使用 X-Ray 精靈或 AWS SDK 將區段文件和遙測上傳至 X-Ray API 的寫入權限。 AWS CLI包含用於取得抽樣規則及報告抽樣結果的讀取許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets",
                "xray:GetSamplingStatisticSummaries"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

- `AWSXrayCrossAccountSharingConfiguration`— 授予建立、管理和檢視可觀察性存取管理員連結的權限，以便在帳戶之間共用 X-Ray 資源。用於啟用來源和監視CloudWatch 帳戶之間的跨帳戶觀察能力。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:Link",
                "oam:ListLinks"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:DeleteLink",
                "oam:GetLink",
                "oam:TagResource"
            ],
            "Resource": "arn:aws:oam:*:*:link/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:CreateLink",
                "oam:UpdateLink"
            ],
```

```
            "Resource": [
                "arn:aws:oam:*:*:link/*",
                "arn:aws:oam:*:*:sink/*"
            ]
        }
    ]

}
```

- `AWSXrayFullAccess`— 使用所有 X-Ray API 的權限，包括讀取權限、寫入權限，以及設定加密金鑰設定和取樣規則的權限。包括可觀察性存取管理員 (OAM) `oam:ListSinks` 和`oam:ListAttachedSinks`權限，可讓主控台檢視從來源帳戶共用的追蹤，做為CloudWatch 跨帳戶觀察性的一部分。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:*",
                "oam:ListSinks"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:ListAttachedLinks"
            ],
            "Resource": "arn:aws:oam:*:*:sink/*"
        }
    ]
}
```

新增受管政策連接到 IAM 使用者、群組或角色

1. 開啟 IAM 主控台。
2. 開啟與您的執行個體描述檔、IAM 使用者或 IAM 群組關聯的角色。

3. 在 Permissions (許可) 下，關聯受管政策。

AWS 受管理政策的 X-Ray 更新

檢視 X-Ray AWS 受管原則更新的詳細資料，因為此服務開始追蹤這些變更。如需有關此頁面變更的自動警示，請訂閱「X-Ray 文件歷史記錄」頁面上的 RSS 摘要。

變更	描述	日期
X-Ray 的 IAM 受管政策 — 新增了新`AWSXrayCrossAccountSharingConfiguration` 的`AWSXrayReadOnlyAccess` 和更新的`AWSXrayFullAccess` 政策。	X-Ray 新增了可觀測性存取管理員 (OAM) 權限`oam:ListSinks` 和這些原則，`oam:ListAttachedSinks` 以允許主控台檢視從來源帳戶共用的追蹤，做為CloudWatch 跨帳戶觀察性的一部分。	2022 年 11 月 27 日
X-Ray 的 IAM 受管政策 — `AWSXrayReadOnlyAccess` 政策更新。	X-Ray 添加了一個 API 操作，`ListResourcePolicies` .	2022 年 11 月 15 日
使用 X-Ray 主控台 — 更新至`AWSXrayReadOnlyAccess` 原則	X-Ray 添加了兩個新的 API 操作，`BatchGetTraceSummaryById` 和`GetDistinctTraceGraphs` . 您的程式碼不會呼叫這些動作。因此，這些 API 動作不會包含在 AWS CLI 和 AWS SDK 中。	2022 年 11 月 11 日

在 IAM 政策中指定資源

您可以使用 IAM 政策來控制對資源的存取。針對支援資源層級許可的動作，您可以使用 Amazon Resource Name (ARN) 來識別要套用政策的資源。

所有 X-Ray 動作都可以在 IAM 政策中使用，授予或拒絕使用者使用該動作的權限。不過，並非所有 X-Ray 動作都支援資源層級權限，這可讓您指定可以執行動作的資源。

對於不支援資源層級許可的動作，您必須使用 "*" 做為資源。

下列 X-Ray 動作支援資源層級權限：

- `CreateGroup`
- `GetGroup`
- `UpdateGroup`
- `DeleteGroup`
- `CreateSamplingRule`
- `UpdateSamplingRule`
- `DeleteSamplingRule`

以下是 `CreateGroup` 動作的身分型許可政策範例。此範例示範使用與有唯一 ID 之群組名稱 `local-users` 有關的 ARN 做為萬用字元。群組建立時產生的唯一 ID，所以無法在政策中事先預測。使用 `GetGroup`、`UpdateGroup` 或 `DeleteGroup` 時，您可以將此定義為萬用字元或確切的 ARN，包括 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:CreateGroup"
            ],
            "Resource": [
                "arn:aws:xray:eu-west-1:123456789012:group/local-users/*"
            ]
        }
    ]
}
```

以下是 `CreateSamplingRule` 動作的身分型許可政策範例。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:CreateSamplingRule"
            ],
            "Resource": [
                "arn:aws:xray:eu-west-1:123456789012:sampling-rule/base-scorekeep"
            ]
        }
    ]
}
```

> **Note**
>
> 依其名稱定義之取樣規則的 ARN。不像群組 ARN，取樣規則沒有唯一產生的 ID。

對 AWS X-Ray 身分與存取進行疑難排解

使用下列資訊可協助您診斷並修正使用 X-Ray 和 IAM 時可能會遇到的常見問題。

主題

- 我沒有在 X 射線中執行動作的權限
- 我沒有授權執行 iam：PassRole
- 我是管理員，並希望允許其他人訪問 X-Ray
- 我想讓我以外的人AWS 帳戶訪問我的 X 射線資源

我沒有在 X 射線中執行動作的權限

若 AWS Management Console 告知您並未獲得執行動作的授權，您必須聯絡您的管理員以取得協助。您的管理員是為您提供登入憑證的人員。

下面的例子發生錯誤時`mateojackson`用戶嘗試使用控制台查看有關採樣規則的詳細信息，但沒有`xray:GetSamplingRules`權限。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to
 perform: xray:GetSamplingRules on resource: arn:${Partition}:xray:${Region}:
${Account}:sampling-rule/${SamplingRuleName}
```

在此情況下，Mateo 請求管理員更新他的政策，以允許他使用 `xray:GetSamplingRules` 動作來存取取樣規則資源。

我沒有授權執行 iam：PassRole

如果您收到錯誤訊息，表示您未獲授權執行`iam:PassRole`動作時，您的政策必須更新，以允許您將角色傳遞給 X-Ray。

有些 AWS 服務 允許您傳遞現有的角色至該服務，而無須建立新的服務角色或服務連結角色。若要執行此作業，您必須擁有將角色傳遞至該服務的許可。

當 IAM 使用者名為時，就會發生下列範例錯誤`marymajor`嘗試使用控制台在 X 射線中執行動作。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 `iam:PassRole` 動作。

如需任何協助，請聯絡您的 AWS 管理員。您的管理員提供您的登入憑證。

我是管理員，並希望允許其他人訪問 X-Ray

若要允許其他人存取 X-Ray，您必須為需要存取的人員或應用程式建立 IAM 實體 (使用者或角色)。他們將使用該實體的憑證來存取 AWS。然後，您必須將策略附加到實體，以便在 X-Ray 中授予他們正確的權限。

若要立即開始使用，請參閱《IAM 使用者指南》中的建立您的第一個 IAM 委派使用者及群組。

我想讓我以外的人AWS 帳戶訪問我的 X 射線資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任對象取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您資源的許可。

若要進一步了解，請參閱以下內容：

- 若要瞭解 X-Ray 是否支援這些功能，請參閱如何與 IAM AWS X-Ray 搭配使用。
- 若要了解如何存取您擁有的所有 AWS 帳戶 所提供的資源，請參閱《IAM 使用者指南》中的將存取權提供給您所擁有的另一個 AWS 帳戶 中的 IAM 使用者。

- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《IAM 使用者指南》中的將存取權提供給第三方擁有的 AWS 帳戶。
- 若要了解如何透過聯合身分提供存取權，請參閱《IAM 使用者指南》中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 若要了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 角色與資源型政策的差異。

AWS X-Ray 中的記錄和監控

監控是維護您 AWS 解決方案之可靠性、可用性和效能的重要部分。您應該從AWS解決方案，以便在出現多點故障時更輕鬆的調試。AWSX-Ray 提供多種工具，讓您監控 X-Ray 資源並回應潛在事件：

AWS CloudTrail 日誌

AWS X-Ray與 整合AWS CloudTrail以記錄由使用者、角色或AWSX-Ray 服務。您可以使用 CloudTrail 即時監控 X-Ray API 請求，並將日誌存放在 Amazon S3、Amazon CloudWatch Logs 以及 Amazon CloudWatch Events 中。如需詳細資訊，請參閱 記錄 X-Ray API 呼叫 AWS CloudTrail。

AWS Config 追蹤

AWS X-Ray與 整合AWS Config以記錄 X-Ray 加密資源所做的組態變更。您可以使用AWS Config 以清查 X-Ray 加密資源，審核 X-Ray 配置歷史記錄，並根據資源變更發送通知。如需詳細資訊，請參閱 跟踪 X-Ray 加密配置更改AWS Config。

Amazon CloudWatch 監控

您可以使用適用於 Java 的 X-Ray 開發套件從收集的 X-Ray 區段發佈未採樣的 Amazon CloudWatch 指標。這些指標衍生自區段的開始和結束時間，以及錯誤、故障和節流狀態標記。您可以使用這些追蹤指標，公開子區段內的重試和相依性問題。如需詳細資訊，請參閱AWS X-Ray用於 Java 的 X-Ray SDK 的指標。

符合性驗證 AWS X-Ray

若要瞭解 AWS 服務 是否屬於特定規範遵循方案的範圍內，請參閱AWS 服務 遵循規範計劃方案中的，並選擇您感興趣的合規方案。如需一般資訊，請參閱AWS 規範計劃AWS。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載中的報告中的 AWS Artifact。

您在使用時的合規責任取決 AWS 服務 於資料的敏感性、公司的合規目標以及適用的法律和法規。AWS 提供下列資源以協助遵循法規：

- 安全性與合規性快速入門指南 — 這些部署指南討論架構考量，並提供部署以安全性和合規性 AWS 為重點的基準環境的步驟。
- 在 Amazon Web Services 上架構 HIPAA 安全性與合規性 — 本白皮書說明公司如何使用建立符合 HIPAA 資格的應 AWS 用程式。

> **Note**
>
> 並非所有人 AWS 服務 都符合 HIPAA 資格。如需詳細資訊，請參閱 HIPAA 資格服務參照。

- AWS 合規資源AWS — 此工作簿和指南集合可能適用於您的產業和所在地。
- AWS 客戶合規指南 — 透過合規的角度瞭解共同的責任模式。這份指南總結了在多個架構 (包括美國國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO)) 中，保 AWS 服務 護指引並對應至安全控制的最佳實務。
- 使用AWS Config 開發人員指南中的規則評估資源 — 此 AWS Config 服務會評估您的資源組態符合內部實務、產業準則和法規的程度。
- AWS Security Hub— 這 AWS 服務 提供了內部安全狀態的全面視圖 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱 Security Hub controls reference。
- AWS Audit Manager— 這 AWS 服務 有助於您持續稽核您的 AWS 使用情況，以簡化您管理風險的方式，以及遵守法規和業界標準的方式。

AWS X-Ray 中的恢復能力

AWS 全球基礎架構是以 AWS 區域 與可用區域為中心建置的。AWS 區域 提供多個分開且隔離的實際可用區域，並以具備低延遲、高輸送量和高度備援特性的聯網相互連結。透過可用區域，您所設計與操作的應用程式和資料庫，就能夠在可用區域之間自動容錯移轉，而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域 與可用區域的詳細資訊，請參閱AWS全球基礎架構。

AWS X-Ray 中的基礎設施安全

作為一種受管服務，AWS X-Ray 受 AWS 全域網路安全的保護。如需有關 AWS 安全服務以及 AWS 如何保護基礎設施的詳細資訊，請參閱 AWS 雲端安全。若要使用基礎設施安全性的最佳實務來設計您的 AWS 環境，請參閱安全性支柱 AWS 架構良好的框架中的基礎設施保護。

您使用AWS發布的 API 調用通過網絡訪問 X 射線。用戶端必須支援下列項目：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密 (PFS) 的密碼套件，例如 DHE (Ephemeral Diffie-Hellman) 或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統 (如 Java 7 和更新版本) 大多會支援這些模式。

此外，請求必須使用存取索引鍵 ID 和與 IAM 主體相關聯的私密存取索引鍵來簽署。或者，您可以使用 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

使用 AWS X-Ray 搭配 VPC 端點

如果您使用亞馬遜虛擬私有雲（亞馬遜 VPC）託管AWS資源，您可以在 VPC 和 X-Ray 之間建立私人連接。這可讓 Amazon VPC 中的資源與 X-Ray 服務進行通訊，而無需透過公用網際網路。

亞馬遜 VPC 是一個AWS 服務您可以使用它來啟動AWS您定義的虛擬網路中的資源。您可利用 VPC 來控制您的網路設定，例如 IP 地址範圍、子網路、路由表和網路閘道。要將 VPC 連接到 X 射線，請定義VPC 端點介面。端點提供可靠、可擴充的 X-Ray 連線，無需網際網路閘道、網路位址轉譯 (NAT) 執行個體或 VPN 連線。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的什麼是 Amazon VPC。

介面 VPC 端點由AWS PrivateLink，一個AWS技術，使之間的私人通信AWS 服務通過使用具有私有 IP 地址的彈性網絡接口。如需詳細資訊，請參閱新增 —AWS PrivateLink為了AWS 服務博客文章和開始使用在亞馬遜 VPC 用戶指南。

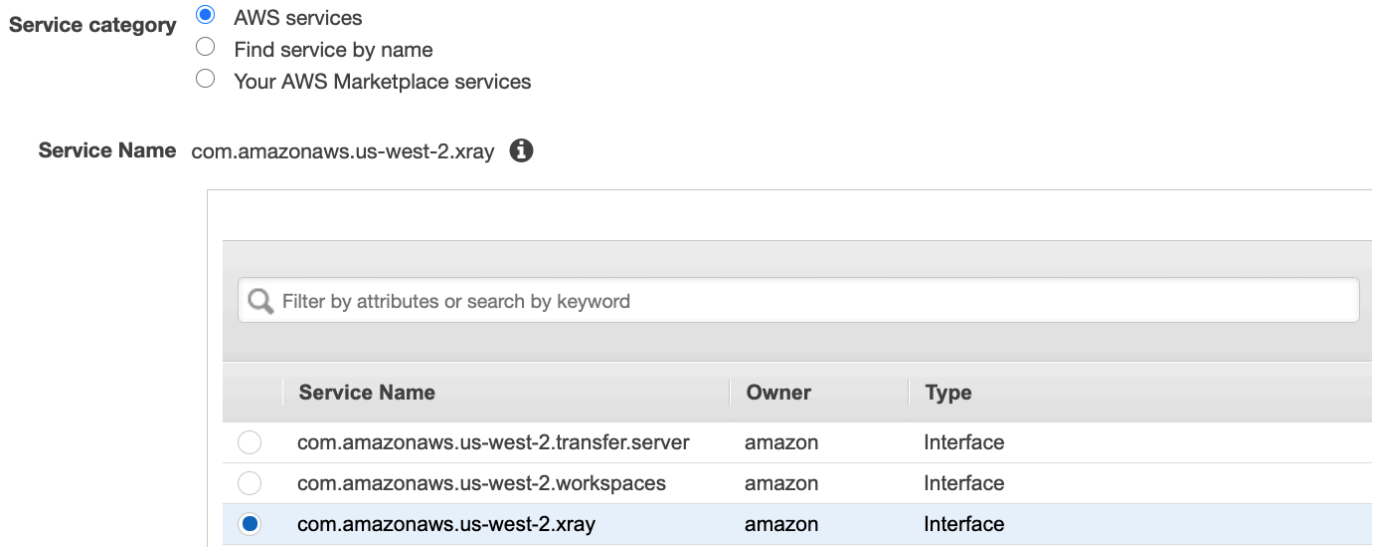
為了確保您可以在您選擇的 X 射線中創建 VPC 端點AWS 區域，請參閱支援的區域。

建立 X 射線的 VPC 端點

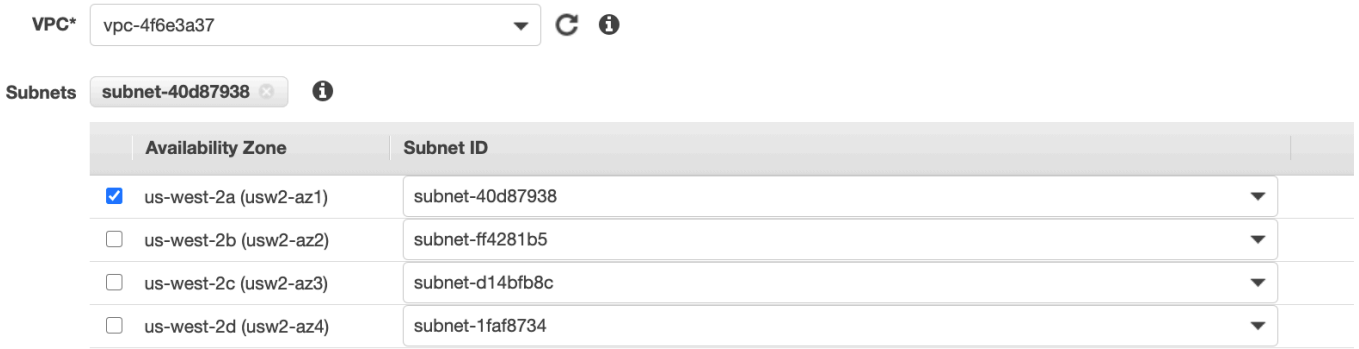
若要開始將 X-Ray 與 VPC 搭配使用，請為 X 射線建立介面 VPC 端點。

1. 在 https://console.aws.amazon.com/vpc/ 開啟 Amazon VPC 主控台。
2. 導覽至端點在導航窗格中並選擇建立端點。

3. 搜尋並選取AWS X-Ray服務：com.amazonaws.*region*.xray。

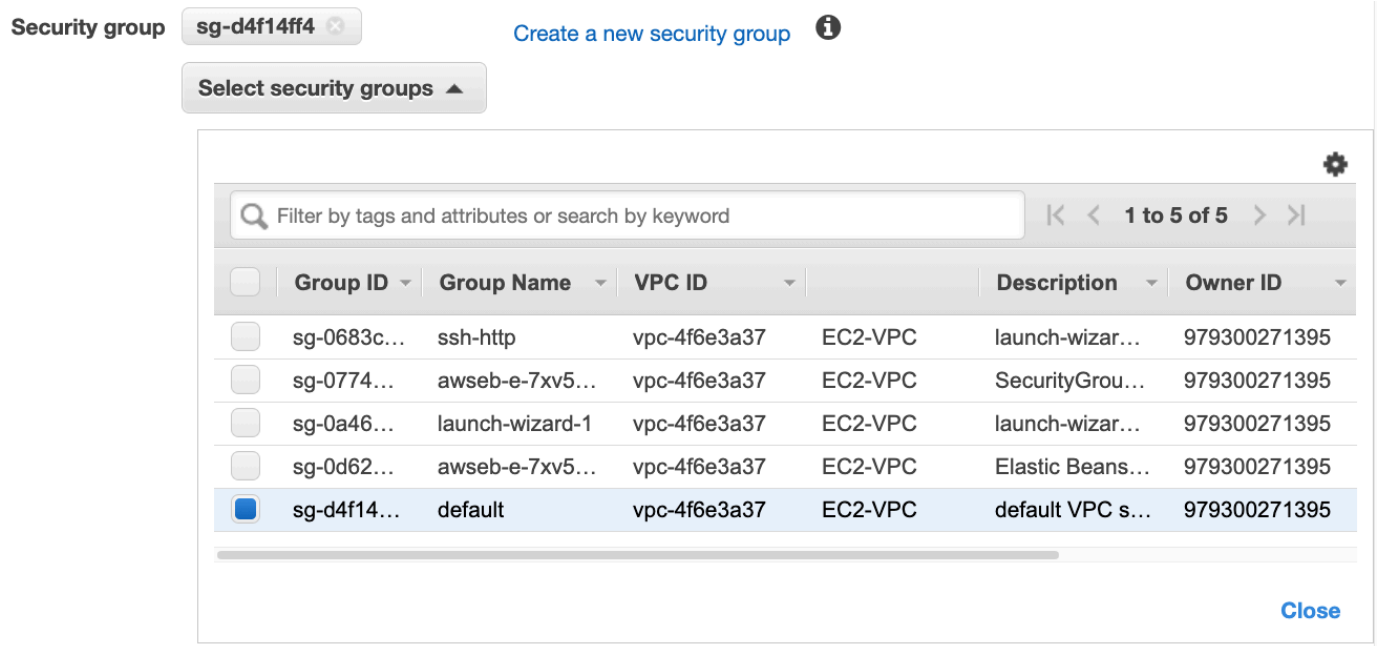


4. 選取所需的 VPC，然後在 VPC 中選取要使用介面端點的子網路。端點網路介面會建立在選取的子網路中。您可以指定不同可用區域中的多個子網路 (服務所支援)，協助確保界面端點在可用區域失敗的狀況下保有彈性。如果這樣做，則會在您指定的每個子網路中建立一個介面網路介面。



5. (選擇性) 端點預設為啟用私人 DNS，因此您可以使用其預設 DNS 主機名稱向 X-Ray 發出要求。您可以選擇禁用它。

6. 指定要與端點網路界面建立關聯的安全群組。



7. (選擇性) 指定自訂原則以控制存取 X-Ray 服務的權限。默認情況下，允許完全訪問。

控制對 X-Ray VPC 端點的存取

當您建立或修改端點時，VPC 端點政策是您連接至端點的 IAM 資源政策。如果您未在建立端點時連接政策，Amazon VPC 會以預設政策連接以允許完整存取服務。端點政策不會覆寫或取代 IAM 使用者政策或服務特定的政策。這個另行區分的政策會控制從端點到所指定之服務的存取。端點政策必須以 JSON 格式撰寫。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的使用 VPC 端點控制服務的存取。

VPC 端點原則可讓您控制各種 X-Ray 動作的權限。例如，您可以建立僅允許的策略PutTraceSegment並拒絕所有其他行動。這會限制 VPC 中的工作負載和服務僅傳送追蹤資料至 X-Ray，並拒絕任何其他動作，例如擷取資料、變更加密設定或建立/更新群組。

以下是 X-Ray 的端點策略範例。此原則可讓使用者透過 VPC 連線至 X-Ray，將區段資料傳送至 X-Ray，並防止他們執行其他 X 射線動作。

```
{"Statement": [
    {"Sid": "Allow PutTraceSegments",
      "Principal": "*",
      "Action": [
        "xray:PutTraceSegments"
      ],
      "Effect": "Allow",
```

```
      "Resource": "*"
    }
  ]
}
```

編輯 X 射線的 VPC 端點原則

1. 在 https://console.aws.amazon.com/vpc/ 開啟 Amazon VPC 主控台。
2. 在導覽窗格中選擇 Endpoints (端點)。
3. 如果尚未建立 X-Ray 的端點，請依照中的步驟執行建立 X 射線的 VPC 端點。
4. 選擇COM. 亞馬遜。##.xray端點，然後選擇政策標籤。
5. 選擇 Edit Policy (編輯政策)，然後進行變更。

支援的區域

X-Ray 目前在下列情況下支援 VPC 端點AWS 區域:

- 美國東部 (俄亥俄)
- 美國東部 (維吉尼亞北部)
- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 非洲 (開普敦)
- 亞太區域 (香港)
- 亞太區域 (孟買)
- 亞太區域 (大阪)
- 亞太區域 (首爾)
- 亞太區域 (新加坡)
- 亞太區域 (雪梨)
- 亞太區域 (東京)
- 加拿大 (中部)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)

- 歐洲 (米蘭)
- 歐洲 (巴黎)
- 歐洲 (斯德哥爾摩)
- 中東 (巴林)
- 南美洲 (聖保羅)
- AWS GovCloud(美國東部)
- AWS GovCloud(美國西部)

AWS X-Ray API

X-Ray API 可透過 AWS SDK 或直接透過 HTTPS 存取所有 X-Ray 功能。 AWS Command Line InterfaceX-Ray API 參考會記錄每個 API 動作的輸入參數，以及它們傳回的欄位和資料類型。

您可以使用 AWS SDK 來開發使用 X-Ray API 的程式。X-Ray 控制台和 X-Ray 守護程序都使用 AWS SDK 與 X-Ray 進行通信。每種語言的 AWS SDK 都有對應至 X-Ray API 動作和類型之類別和方法的參考文件。

AWS SDK 參考資料

- 爪哇 — AWS SDK for Java
- JavaScript – AWS SDK for JavaScript
- .NET – AWS SDK for .NET
- Ruby – AWS SDK for Ruby
- 前往 — AWS SDK for Go
- PHP – AWS SDK for PHP
- Python – AWS SDK for Python (Boto)

這 AWS Command Line Interface 是一個命令列工具，它使用適用於 Python 的 SDK 來呼叫 AWS API。當您第一次學習 AWS API 時， AWS CLI 提供了一種簡單的方法來探索可用的參數，並以 JSON 或文本形式查看服務輸出。

如需有關子指AWS CLI 令的詳細資訊，請參閱命`aws xray`令參考。

主題

- 使用AWS X-Ray使用指令碼的 APIAWSCLI
- 傳送追蹤資料至 AWS X-Ray
- 從中獲取數據 AWS X-Ray
- 使用 AWS X-Ray API 設定抽樣、分組和加密設定
- 使用取樣規則搭配 X-Ray API
- AWS X-Ray 區段文件

使用AWS X-Ray使用指令碼的 APIAWSCLI

該AWSCLI 可讓您直接存取 X-Ray 服務，並使用 X-Ray 主控台用來擷取服務圖表和原始追蹤資料的相同 API。範例應用程式包含指令碼，說明如何搭配使用這些 APIAWSCLI。

先決條件

此教學使用 Scorekeep 範例應用程式和隨附的指令碼，以產生追蹤資料和服務地圖。按照入門教學中的指示啟動應用程式。

本教學課程使用AWS CLI顯示 X 射線 API 的基本用途。該AWSCLI,適用於視窗、Linux 和作業系統，為所有人提供對公共 API 的命令行訪問AWS 服務。

> **Note**
>
> 您必須驗證 AWS CLI 是否設定為建立 Scorekeep 範例應用程式的同一個區域。

隨附的指令碼 (用來測試範例應用程式) 會使用 cURL，將流量傳送到 API 和 `jq` 以剖析輸出。您可以下載`jq`可執行來源斯泰多兰·吉图比奥，以及`curl`可執行來源https://curl.haxx.se/download.html。大多數的 Linux 和 OS X 安裝都包括 cURL。

產生追蹤資料

當遊戲進行中時，Web 應用程式會每隔幾秒鐘持續產生對 API 的流量，但只會產生一種類型的請求。使用 `test-api.sh` 指令碼來執行端對端案例，並在您測試 API 時產生更多元化的追蹤資料。

使用 **`test-api.sh`** 指令碼

1. 開啟 Elastic Beanstalk 主控台。
2. 導覽至管理主控台適用於您的環境。
3. 複製頁面標頭的環境 URL。
4. 開啟 `bin/test-api.sh` 並將 API 的值取代為您環境的 URL。

   ```
   #!/bin/bash
   API=scorekeep.9hbtbm23t2.us-west-2.elasticbeanstalk.com/api
   ```

5. 執行指令碼來產生對 API 的流量。

   ```
   ~/debugger-tutorial$ ./bin/test-api.sh
   ```

```
Creating users,
session,
game,
configuring game,
playing game,
ending game,
game complete.
{"id":"MTBP8BAS","session":"HUF6IT64","name":"tic-tac-toe-test","users":
["QFF3HBGM","KL6JR98D"],"rules":"102","startTime":1476314241,"endTime":1476314245,"states":
["JQVLEOM2","D67QLPIC","VF9BM9NC","OEAA6GK9","2A705O73","1U2LFTLJ","HUKIDD70","BAN1C8FI","G
["BS8F8LQ","4MTTSPKP","463OETES","SVEBCL3N","N7CQ1GHP","O84ONEPD","EG4BPROQ","V4BLIDJ3","9R
```

使用 X 射線 API

該AWSCLI 為 X-Ray 提供的所有 API 動作提供命令，包括GetServiceGraph和GetTraceSummaries。如需所使用的所有支援動作和資料類型詳細資訊，請參閱 AWS X-Ray API 參考。

Example bin/service-graph.sh

```
EPOCH=$(date +%s)
aws xray get-service-graph --start-time $(($EPOCH-600)) --end-time $EPOCH
```

指令碼會擷取最後 10 分鐘的服務圖表。

```
~/eb-java-scorekeep$ ./bin/service-graph.sh | less
{
    "StartTime": 1479068648.0,
    "Services": [
        {
            "StartTime": 1479068648.0,
            "ReferenceId": 0,
            "State": "unknown",
            "EndTime": 1479068651.0,
            "Type": "client",
            "Edges": [
                {
                    "StartTime": 1479068648.0,
                    "ReferenceId": 1,
                    "SummaryStatistics": {
                        "ErrorStatistics": {
```

```
                            "ThrottleCount": 0,
                            "TotalCount": 0,
                            "OtherCount": 0
                        },
                        "FaultStatistics": {
                            "TotalCount": 0,
                            "OtherCount": 0
                        },
                        "TotalCount": 2,
                        "OkCount": 2,
                        "TotalResponseTime": 0.054000139236450195
                    },
                    "EndTime": 1479068651.0,
                    "Aliases": []
                }
            ]
        },
        {
            "StartTime": 1479068648.0,
            "Names": [
                "scorekeep.elasticbeanstalk.com"
            ],
            "ReferenceId": 1,
            "State": "active",
            "EndTime": 1479068651.0,
            "Root": true,
            "Name": "scorekeep.elasticbeanstalk.com",
...
```

Example bin/trace-urls.sh

```
EPOCH=$(date +%s)
aws xray get-trace-summaries --start-time $(($EPOCH-120)) --end-time $(($EPOCH-60)) --
query 'TraceSummaries[*].Http.HttpURL'
```

指令碼會擷取前一分鐘和兩個分鐘之間產生的追蹤 URL。

```
~/eb-java-scorekeep$ ./bin/trace-urls.sh
[
    "http://scorekeep.elasticbeanstalk.com/api/game/6Q0UE1DG/5FGLM9U3/
endtime/1479069438",
    "http://scorekeep.elasticbeanstalk.com/api/session/KH4341QH",
    "http://scorekeep.elasticbeanstalk.com/api/game/GLQBJ3K5/153AHDIA",
```

```
    "http://scorekeep.elasticbeanstalk.com/api/game/VPDL672J/G2V41HM6/
endtime/1479069466"
]
```

Example bin/full-traces.sh

```
EPOCH=$(date +%s)
TRACEIDS=$(aws xray get-trace-summaries --start-time $(($EPOCH-120)) --end-time
 $(($EPOCH-60)) --query 'TraceSummaries[*].Id' --output text)
aws xray batch-get-traces --trace-ids $TRACEIDS --query 'Traces[*]'
```

指令碼會擷取前一分鐘和兩個分鐘之間產生的完整追蹤。

```
~/eb-java-scorekeep$ ./bin/full-traces.sh | less
[
    {
        "Segments": [
            {
                "Id": "3f212bc237bafd5d",
                "Document": "{\"id\":\"3f212bc237bafd5d\",\"name\":\"DynamoDB\",
\"trace_id\":\"1-5828d9f2-a90669393f4343211bc1cf75\",\"start_time\":1.479072242459E9,
\"end_time\":1.479072242477E9,\"parent_id\":\"72a08dcf87991ca9\",\"http\":
{\"response\":{\"content_length\":60,\"status\":200}},\"inferred\":true,\"aws\":
{\"consistent_read\":false,\"table_name\":\"scorekeep-session-xray\",\"operation\":
\"GetItem\",\"request_id\":\"QAKE0S8DD0LJM245KAOPMA746BVV4KQNSO5AEMVJF66Q9ASUAAJG\",
\"resource_names\":[\"scorekeep-session-xray\"]},\"origin\":\"AWS::DynamoDB::Table\"}"
            },
            {
                "Id": "309e355f1148347f",
                "Document": "{\"id\":\"309e355f1148347f\",\"name\":\"DynamoDB\",
\"trace_id\":\"1-5828d9f2-a90669393f4343211bc1cf75\",\"start_time\":1.479072242477E9,
\"end_time\":1.479072242494E9,\"parent_id\":\"37f14ef837f00022\",\"http\":
{\"response\":{\"content_length\":606,\"status\":200}},\"inferred\":true,\"aws\":
{\"table_name\":\"scorekeep-game-xray\",\"operation\":\"UpdateItem\",\"request_id
\":\"388GEROC4PCA6D59ED3CTI5EEJVV4KQNSO5AEMVJF66Q9ASUAAJG\",\"resource_names\":
[\"scorekeep-game-xray\"]},\"origin\":\"AWS::DynamoDB::Table\"}"
            }
        ],
        "Id": "1-5828d9f2-a90669393f4343211bc1cf75",
        "Duration": 0.05099987983703613
    }
...
```

清除

終止您的彈性豆莖環境，以關閉 Amazon EC2 執行個體、DynamoDB 表和其他資源。

若要終止您的 Elastic Beanstalk 環境

1. 開啟 Elastic Beanstalk 主控台。
2. 導覽至管理主控台適用於您的環境。
3. 選擇 Actions (動作)。
4. 選擇 Terminate Environment (終止環境)。
5. 選擇 Terminate (終止)。

追蹤資料會在 30 天後自動從 X-Ray 中刪除。

傳送追蹤資料至 AWS X-Ray

您可以將追蹤資料以區段文件的形式傳送至 X-Ray。區段文件是一種 JSON 格式字串，包含您應用程式在處理請求時執行工作的相關資訊。您的應用程式可記錄其在區段中自行執行的工作相關資料，或是在子區段中使用下游服務和資源的工作相關資料。

區段會記錄您應用程式執行工作的相關資訊。區段最少會記錄其在任務上耗費的時間、名稱及兩個 ID。追蹤 ID 會在請求於服務間傳送時追蹤請求。區段 ID 會追蹤單一服務為請求完成的工作。

Example 最小的完成區段

```
{
  "name" : "Scorekeep",
  "id" : "70de5b6f19ff9a0a",
  "start_time" : 1.478293361271E9,
  "trace_id" : "1-581cf771-a006649127e371903a2de979",
  "end_time" : 1.478293361449E9
}
```

接收到請求時，您可以傳送正在進行中的區段做為預留位置，直到完成請求。

Example 進行中的區段

```
{
```

```
  "name" : "Scorekeep",
  "id" : "70de5b6f19ff9a0b",
  "start_time" : 1.478293361271E9,
  "trace_id" : "1-581cf771-a006649127e371903a2de979",
  “in_progress”: true
}
```

您可以使用 X 射線精靈直接PutTraceSegments、或透過 X-Ray 精靈將區段傳送至 X-Ray。

大多數應用程式都會使用 AWS SDK 呼叫其他服務或存取資源。記錄子區段中下游呼叫的相關資訊。X-Ray 使用子區段來識別不傳送區段的下游服務，並在服務圖上為其建立項目。

子區段可內嵌在完整區段文件中，或是分別傳送。分別傳送子區段，以非同步方式追蹤長時間執行要求的下游呼叫，或避免超過區段文件大小上限 (64 kB)。

Example 子區段

子區段具備 subsegment 的 type，以及可識別父區段的 parent_id。

```
{
  "name" : "www2.example.com",
  "id" : "70de5b6f19ff9a0c",
  "start_time" : 1.478293361271E9,
  "trace_id" : "1-581cf771-a006649127e371903a2de979"
  “end_time” : 1.478293361449E9,
  “type” : “subsegment”,
  “parent_id” : “70de5b6f19ff9a0b”
}
```

如需您可以包含在區段和子區段中欄位和值的詳細資訊，請參閱 AWS X-Ray 區段文件。

章節

- 產生追蹤 ID
- 使用 PutTraceSegments
- 將區段文件傳送至 X-Ray 精靈

產生追蹤 ID

若要將資料傳送至 X-Ray，您必須為每個要求產生唯一的追蹤 ID。

X-Ray 軌跡 ID 格式

X-Ray `trace_id` 由三個用連字符分隔的數字組成。例如 `1-58406520-a006649127e371903a2de979`。其中包含：

- 版本號碼，也就是1。
- 原始請求的時間在 Unix 紀元時間使用 8 個十六進制數字。

  例如，2016 年 12 月 1 日上午 10:00 PST (以紀元時間表示) 為1480615200秒或十六進58406520位數字。
- 追蹤的全域唯一 96 位元識別碼，以 24 個十六進位數字顯示。

> **Note**
>
> X-Ray 現在支援使用以 OpenTelemetry 及符合 W3C 追蹤上下文規格的任何其他架構建立的追蹤 ID。傳送至 X-Ray 時，W3C 追蹤識別碼必須格式化為 X-Ray 追蹤 ID 格式。例如，W3C 追蹤識別碼`4efaaf4d1e8720b39541901950019ee5`應格式化為傳送至 X-Ray `1-4efaaf4d-1e8720b39541901950019ee5` 時。X-Ray 跟踪 ID 包括 Unix 紀元時間中的原始請求時間戳，但是在以 X-Ray 格式發送 W3C 跟踪 ID 時，這不是必需的。

您可以撰寫指令碼來產生 X-Ray 追蹤 ID 以進行測試。以下是兩個範例。

Python

```
import time
import os
import binascii

START_TIME = time.time()
HEX=hex(int(START_TIME))[2:]
TRACE_ID="1-{}-{}".format(HEX, binascii.hexlify(os.urandom(12)).decode('utf-8'))
```

Bash

```
START_TIME=$(date +%s)
HEX_TIME=$(printf '%x\n' $START_TIME)
GUID=$(dd if=/dev/random bs=12 count=1 2>/dev/null | od -An -tx1 | tr -d ' \t\n')
TRACE_ID="1-$HEX_TIME-$GUID"
```

如需建立追蹤 ID 並將區段傳送至 X-Ray 精靈的指令碼，請參閱 Scorekeep 範例應用程式。

- Python – xray_start.py
- 巴什 — xray_start.sh

使用 PutTraceSegments

您可以使用 PutTraceSegments API 上傳區段文件。API 具備單一參數 (TraceSegmentDocuments)，會接受 JSON 區段文件清單。

使用 AWS CLI，使用aws xray put-trace-segments命令將區段文件直接傳送到 X-Ray。

```
$ DOC='{"trace_id": "1-5960082b-ab52431b496add878434aa25", "id": "6226467e3f845502",
 "start_time": 1498082657.37518, "end_time": 1498082695.4042, "name":
 "test.elasticbeanstalk.com"}'
$ aws xray put-trace-segments --trace-segment-documents "$DOC"
{
    "UnprocessedTraceSegments": []
}
```

> **Note**
>
> Windows 命令處理器和 Windows PowerShell 對於 JSON 字符串中引用和轉義引號有不同的要求。如需詳細資訊，請參閱《 AWS CLI 使用指南》中的引用字串。

輸出會列出任何處理失敗的區段。例如，若追蹤 ID 內的日期為距離現在太遠的過去日期，您可能會看到如下的錯誤。

```
{
    "UnprocessedTraceSegments": [
        {
            "ErrorCode": "InvalidTraceId",
            "Message": "Invalid segment. ErrorCode: InvalidTraceId",
            "Id": "6226467e3f845502"
        }
    ]
}
```

您可以同時傳遞多個區段文件，並以空格區隔。

```
$ aws xray put-trace-segments --trace-segment-documents "$DOC1" "$DOC2"
```

將區段文件傳送至 X-Ray 精靈

您可以將區段和子區段傳送至 X-Ray API，而不是將區段文件傳送至 X-Ray API，而是將區段和子區段進行緩衝，並分批上傳至 X-Ray API。X-Ray SDK 會將區段文件傳送至精靈，以避免 AWS 直接呼叫。

> **Note**
>
> 如需執行精靈的說明，請參閱在本機執行 X-Ray 精靈。

透過 UDP 連接埠 2000 以 JSON 傳送區段，並在前面加上精靈標頭 (`{"format": "json", "version": 1}\n`)

```
{"format": "json", "version": 1}\n{"trace_id": "1-5759e988-bd862e3fe1be46a994272793",
 "id": "defdfd9912dc5a56", "start_time": 1461096053.37518, "end_time": 1461096053.4042,
 "name": "test.elasticbeanstalk.com"}
```

在 Linux 上，您可以從 Bash 終端機將區段文件傳送到精靈。將標頭和區段文件儲存到文字檔，並使用 `cat` 將它輸送到 `/dev/udp`。

```
$ cat segment.txt > /dev/udp/127.0.0.1/2000
```

Example segment.txt

```
{"format": "json", "version": 1}
{"trace_id": "1-594aed87-ad72e26896b3f9d3a27054bb", "id": "6226467e3f845502",
 "start_time": 1498082657.37518, "end_time": 1498082695.4042, "name":
 "test.elasticbeanstalk.com"}
```

檢查精靈記錄檔，確認是否已將區段傳送至 X-Ray。

```
2017-07-07T01:57:24Z [Debug] processor: sending partial batch
2017-07-07T01:57:24Z [Debug] processor: segment batch size: 1. capacity: 50
2017-07-07T01:57:24Z [Info] Successfully sent batch of 1 segments (0.020 seconds)
```

從中獲取數據 AWS X-Ray

AWS X-Ray 處理您傳送給它的追蹤資料，以便以 JSON 格式產生完整追蹤、追蹤摘要和服務圖形。您可以使用 AWS CLI 直接從 API 擷取產生的資料。

章節

擷取服務圖表

您可以使用 `GetServiceGraph` API 擷取 JSON 服務圖表。API 需要開始時間及結束時間。您可以從 Linux 終端機使用 `date` 命令來計算該時間。

```
$ date +%s
1499394617
```

`date +%s` 會印出日期 (秒)。使用此數字做為結束時間，並減去時間來取得開始時間。

Example 擷取最後 10 分鐘服務圖表的指令碼

```
EPOCH=$(date +%s)
aws xray get-service-graph --start-time $(($EPOCH-600)) --end-time $EPOCH
```

下列範例顯示包含 4 個節點的服務圖表，其中包括一個用戶端節點、EC2 執行個體、一個 DynamoDB 表和一個 Amazon SNS 主題。

Example GetServiceGraph 輸出

```
{
    "Services": [
        {
            "ReferenceId": 0,
            "Name": "xray-sample.elasticbeanstalk.com",
            "Names": [
                "xray-sample.elasticbeanstalk.com"
            ],
```

```
            "Type": "client",
            "State": "unknown",
            "StartTime": 1528317567.0,
            "EndTime": 1528317589.0,
            "Edges": [
                {
                    "ReferenceId": 2,
                    "StartTime": 1528317567.0,
                    "EndTime": 1528317589.0,
                    "SummaryStatistics": {
                        "OkCount": 3,
                        "ErrorStatistics": {
                            "ThrottleCount": 0,
                            "OtherCount": 1,
                            "TotalCount": 1
                        },
                        "FaultStatistics": {
                            "OtherCount": 0,
                            "TotalCount": 0
                        },
                        "TotalCount": 4,
                        "TotalResponseTime": 0.273
                    },
                    "ResponseTimeHistogram": [
                        {
                            "Value": 0.005,
                            "Count": 1
                        },
                        {
                            "Value": 0.015,
                            "Count": 1
                        },
                        {
                            "Value": 0.157,
                            "Count": 1
                        },
                        {
                            "Value": 0.096,
                            "Count": 1
                        }
                    ],
                    "Aliases": []
                }
            ]
```

```
        },
        {
            "ReferenceId": 1,
            "Name": "awseb-e-dixzws4s9p-stack-StartupSignupsTable-4IMSMHAYX2BA",
            "Names": [
                "awseb-e-dixzws4s9p-stack-StartupSignupsTable-4IMSMHAYX2BA"
            ],
            "Type": "AWS::DynamoDB::Table",
            "State": "unknown",
            "StartTime": 1528317583.0,
            "EndTime": 1528317589.0,
            "Edges": [],
            "SummaryStatistics": {
                "OkCount": 2,
                "ErrorStatistics": {
                    "ThrottleCount": 0,
                    "OtherCount": 0,
                    "TotalCount": 0
                },
                "FaultStatistics": {
                    "OtherCount": 0,
                    "TotalCount": 0
                },
                "TotalCount": 2,
                "TotalResponseTime": 0.12
            },
            "DurationHistogram": [
                {
                    "Value": 0.076,
                    "Count": 1
                },
                {
                    "Value": 0.044,
                    "Count": 1
                }
            ],
            "ResponseTimeHistogram": [
                {
                    "Value": 0.076,
                    "Count": 1
                },
                {
                    "Value": 0.044,
                    "Count": 1
```

```
                }
            ]
        },
        {
            "ReferenceId": 2,
            "Name": "xray-sample.elasticbeanstalk.com",
            "Names": [
                "xray-sample.elasticbeanstalk.com"
            ],
            "Root": true,
            "Type": "AWS::EC2::Instance",
            "State": "active",
            "StartTime": 1528317567.0,
            "EndTime": 1528317589.0,
            "Edges": [
                {
                    "ReferenceId": 1,
                    "StartTime": 1528317567.0,
                    "EndTime": 1528317589.0,
                    "SummaryStatistics": {
                        "OkCount": 2,
                        "ErrorStatistics": {
                            "ThrottleCount": 0,
                            "OtherCount": 0,
                            "TotalCount": 0
                        },
                        "FaultStatistics": {
                            "OtherCount": 0,
                            "TotalCount": 0
                        },
                        "TotalCount": 2,
                        "TotalResponseTime": 0.12
                    },
                    "ResponseTimeHistogram": [
                        {
                            "Value": 0.076,
                            "Count": 1
                        },
                        {
                            "Value": 0.044,
                            "Count": 1
                        }
                    ],
                    "Aliases": []
```

```
            },
            {
                "ReferenceId": 3,
                "StartTime": 1528317567.0,
                "EndTime": 1528317589.0,
                "SummaryStatistics": {
                    "OkCount": 2,
                    "ErrorStatistics": {
                        "ThrottleCount": 0,
                        "OtherCount": 0,
                        "TotalCount": 0
                    },
                    "FaultStatistics": {
                        "OtherCount": 0,
                        "TotalCount": 0
                    },
                    "TotalCount": 2,
                    "TotalResponseTime": 0.125
                },
                "ResponseTimeHistogram": [
                    {
                        "Value": 0.049,
                        "Count": 1
                    },
                    {
                        "Value": 0.076,
                        "Count": 1
                    }
                ],
                "Aliases": []
            }
        ],
        "SummaryStatistics": {
            "OkCount": 3,
            "ErrorStatistics": {
                "ThrottleCount": 0,
                "OtherCount": 1,
                "TotalCount": 1
            },
            "FaultStatistics": {
                "OtherCount": 0,
                "TotalCount": 0
            },
            "TotalCount": 4,
```

```
                "TotalResponseTime": 0.273
            },
            "DurationHistogram": [
                {
                    "Value": 0.005,
                    "Count": 1
                },
                {
                    "Value": 0.015,
                    "Count": 1
                },
                {
                    "Value": 0.157,
                    "Count": 1
                },
                {
                    "Value": 0.096,
                    "Count": 1
                }
            ],
            "ResponseTimeHistogram": [
                {
                    "Value": 0.005,
                    "Count": 1
                },
                {
                    "Value": 0.015,
                    "Count": 1
                },
                {
                    "Value": 0.157,
                    "Count": 1
                },
                {
                    "Value": 0.096,
                    "Count": 1
                }
            ]
        },
        {
            "ReferenceId": 3,
            "Name": "SNS",
            "Names": [
                "SNS"
```

```
            ],
            "Type": "AWS::SNS",
            "State": "unknown",
            "StartTime": 1528317583.0,
            "EndTime": 1528317589.0,
            "Edges": [],
            "SummaryStatistics": {
                "OkCount": 2,
                "ErrorStatistics": {
                    "ThrottleCount": 0,
                    "OtherCount": 0,
                    "TotalCount": 0
                },
                "FaultStatistics": {
                    "OtherCount": 0,
                    "TotalCount": 0
                },
                "TotalCount": 2,
                "TotalResponseTime": 0.125
            },
            "DurationHistogram": [
                {
                    "Value": 0.049,
                    "Count": 1
                },
                {
                    "Value": 0.076,
                    "Count": 1
                }
            ],
            "ResponseTimeHistogram": [
                {
                    "Value": 0.049,
                    "Count": 1
                },
                {
                    "Value": 0.076,
                    "Count": 1
                }
            ]
        }
    ]
}
```

根據群組擷取服務圖表

若要根據群組的內容呼叫服務圖形，請包含 groupName 或 groupARN。以下範例為服務圖表呼叫名為 Example1 的群組。

Example 根據名稱為 Example1 群組擷取服務圖表的指令碼

```
aws xray get-service-graph --group-name "Example1"
```

擷取追蹤

您可以使用 GetTraceSummaries API 取得追蹤摘要清單。追蹤摘要包含您可以用來識別要完整下載追蹤的資訊，包含標註、請求和回應資訊，以及 ID。

呼叫 aws xray get-trace-summaries 時有兩個 TimeRangeType 標記可用：

- TraceId— 預設GetTraceSummaries搜尋使用 TracEid 時間，並傳回在計算[start_time, end_time)範圍內啟動的追蹤。此時間戳記範圍是根據中的時間戳記編碼計算的 TraceId，或者可以手動定義。
- 事件時間 — 若要搜尋一段時間內發生的事件， AWS X-Ray 允許使用事件時間戳記搜尋追蹤。事件時間會傳回 [start_time, end_time) 範圍內作用中的追蹤，無論追蹤由何時開始。

使用 aws xray get-trace-summaries 命令來取得追蹤摘要清單。下列命令會使用預設 TraceId 時間取得過去 1 到 2 分鐘之間的追蹤摘要清單。

Example 取得追蹤摘要的指令碼

```
EPOCH=$(date +%s)
aws xray get-trace-summaries --start-time $(($EPOCH-120)) --end-time $(($EPOCH-60))
```

Example GetTraceSummaries 輸出

```
{
    "TraceSummaries": [
        {
            "HasError": false,
            "Http": {
                "HttpStatus": 200,
                "ClientIp": "205.255.255.183",
                "HttpURL": "http://scorekeep.elasticbeanstalk.com/api/session",
```

```
                "UserAgent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36",
                "HttpMethod": "POST"
            },
            "Users": [],
            "HasFault": false,
            "Annotations": {},
            "ResponseTime": 0.084,
            "Duration": 0.084,
            "Id": "1-59602606-a43a1ac52fc7ee0eea12a82c",
            "HasThrottle": false
        },
        {
            "HasError": false,
            "Http": {
                "HttpStatus": 200,
                "ClientIp": "205.255.255.183",
                "HttpURL": "http://scorekeep.elasticbeanstalk.com/api/user",
                "UserAgent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36",
                "HttpMethod": "POST"
            },
            "Users": [
                {
                    "UserName": "5M388M1E"
                }
            ],
            "HasFault": false,
            "Annotations": {
                "UserID": [
                    {
                        "AnnotationValue": {
                            "StringValue": "5M388M1E"
                        }
                    }
                ],
                "Name": [
                    {
                        "AnnotationValue": {
                            "StringValue": "Ola"
                        }
                    }
                ]
            },
```

```
            "ResponseTime": 3.232,
            "Duration": 3.232,
            "Id": "1-59602603-23fc5b688855d396af79b496",
            "HasThrottle": false
        }
    ],
    "ApproximateTime": 1499473304.0,
    "TracesProcessedCount": 2
}
```

使用來自輸出的追蹤 ID 來透過 BatchGetTraces API 擷取完整追蹤。

Example BatchGetTraces 命令

```
$ aws xray batch-get-traces --trace-ids 1-596025b4-7170afe49f7aa708b1dd4a6b
```

Example BatchGetTraces 輸出

```
{
    "Traces": [
        {
            "Duration": 3.232,
            "Segments": [
                {
                    "Document": "{\"id\":\"1fb07842d944e714\",\"name\":
\"random-name\",\"start_time\":1.499473411677E9,\"end_time\":1.499473414572E9,
\"parent_id\":\"0c544c1b1bbff948\",\"http\":{\"response\":{\"status\":200}},
\"aws\":{\"request_id\":\"ac086670-6373-11e7-a174-f31b3397f190\"},\"trace_id\":
\"1-59602603-23fc5b688855d396af79b496\",\"origin\":\"AWS::Lambda\",\"resource_arn\":
\"arn:aws:lambda:us-west-2:123456789012:function:random-name\"}",
                    "Id": "1fb07842d944e714"
                },
                {
                    "Document": "{\"id\":\"194fcc8747581230\",\"name\":\"Scorekeep
\",\"start_time\":1.499473411562E9,\"end_time\":1.499473414794E9,\"http\":{\"request
\":{\"url\":\"http://scorekeep.elasticbeanstalk.com/api/user\",\"method\":\"POST\",
\"user_agent\":\"Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/59.0.3071.115 Safari/537.36\",\"client_ip\":\"205.251.233.183\"},
\"response\":{\"status\":200}},\"aws\":{\"elastic_beanstalk\":{\"version_label\":\"app-
abb9-170708_002045\",\"deployment_id\":406,\"environment_name\":\"scorekeep-dev\"},
\"ec2\":{\"availability_zone\":\"us-west-2c\",\"instance_id\":\"i-0cd9e448944061b4a
\"},\"xray\":{\"sdk_version\":\"1.1.2\",\"sdk\":\"X-Ray for Java\"}},\"service
\":{},\"trace_id\":\"1-59602603-23fc5b688855d396af79b496\",\"user\":\"5M388M1E
```

\",\"origin\":\"AWS::ElasticBeanstalk::Environment\",\"subsegments\":[{\"id\":
\"0c544c1b1bbff948\",\"name\":\"Lambda\",\"start_time\":1.499473411629E9,\"end_time
\":1.499473414572E9,\"http\":{\"response\":{\"status\":200,\"content_length\":14}},
\"aws\":{\"log_type\":\"None\",\"status_code\":200,\"function_name\":\"random-name
\",\"invocation_type\":\"RequestResponse\",\"operation\":\"Invoke\",\"request_id
\":\"ac086670-6373-11e7-a174-f31b3397f190\",\"resource_names\":[\"random-name\"]},
\"namespace\":\"aws\"},{\"id\":\"071684f2e555e571\",\"name\":\"## UserModel.saveUser
\",\"start_time\":1.499473414581E9,\"end_time\":1.499473414769E9,\"metadata\":{\"debug
\":{\"test\":\"Metadata string from UserModel.saveUser\"}},\"subsegments\":[{\"id\":
\"4cd3f10b76c624b4\",\"name\":\"DynamoDB\",\"start_time\":1.49947341469E9,\"end_time
\":1.499473414769E9,\"http\":{\"response\":{\"status\":200,\"content_length\":57}},
\"aws\":{\"table_name\":\"scorekeep-user\",\"operation\":\"UpdateItem\",\"request_id
\":\"MFQ8CGJ3JTDDVVVASUAAJGQ6NJ82F738BOB4KQNSO5AEMVJF66Q9\",\"resource_names\":
[\"scorekeep-user\"]},\"namespace\":\"aws\"}]}]}",
                    "Id": "194fcc8747581230"
                },
                {
                    "Document": "{\"id\":\"00f91aa01f4984fd\",\"name\":
\"random-name\",\"start_time\":1.49947341283E9,\"end_time\":1.49947341457E9,
\"parent_id\":\"1fb07842d944e714\",\"aws\":{\"function_arn\":\"arn:aws:lambda:us-
west-2:123456789012:function:random-name\",\"resource_names\":[\"random-name\"],
\"account_id\":\"123456789012\"},\"trace_id\":\"1-59602603-23fc5b688855d396af79b496\",
\"origin\":\"AWS::Lambda::Function\",\"subsegments\":[{\"id\":\"e6d2fe619f827804\",
\"name\":\"annotations\",\"start_time\":1.499473413012E9,\"end_time\":1.499473413069E9,
\"annotations\":{\"UserID\":\"5M388M1E\",\"Name\":\"Ola\"}},{\"id\":\"b29b548af4d54a0f
\",\"name\":\"SNS\",\"start_time\":1.499473413112E9,\"end_time\":1.499473414071E9,
\"http\":{\"response\":{\"status\":200}},\"aws\":{\"operation\":\"Publish\",
\"region\":\"us-west-2\",\"request_id\":\"a2137970-f6fc-5029-83e8-28aadeb99198\",
\"retries\":0,\"topic_arn\":\"arn:aws:sns:us-west-2:123456789012:awseb-e-
ruag3jyweb-stack-NotificationTopic-6B829NT9V5O9\"},\"namespace\":\"aws\"},{\"id\":
\"2279c0030c955e52\",\"name\":\"Initialization\",\"start_time\":1.499473412064E9,
\"end_time\":1.499473412819E9,\"aws\":{\"function_arn\":\"arn:aws:lambda:us-
west-2:123456789012:function:random-name\"}}]}",
                    "Id": "00f91aa01f4984fd"
                },
                {
                    "Document": "{\"id\":\"17ba309b32c7fbaf\",\"name\":
\"DynamoDB\",\"start_time\":1.49947341469E9,\"end_time\":1.499473414769E9,
\"parent_id\":\"4cd3f10b76c624b4\",\"inferred\":true,\"http\":{\"response
\":{\"status\":200,\"content_length\":57}},\"aws\":{\"table_name
\":\"scorekeep-user\",\"operation\":\"UpdateItem\",\"request_id\":
\"MFQ8CGJ3JTDDVVVASUAAJGQ6NJ82F738BOB4KQNSO5AEMVJF66Q9\",\"resource_names\":
[\"scorekeep-user\"]},\"trace_id\":\"1-59602603-23fc5b688855d396af79b496\",\"origin\":
\"AWS::DynamoDB::Table\"}",

```
                    "Id": "17ba309b32c7fbaf"
                },
                {
                    "Document": "{\"id\":\"1ee3c4a523f89ca5\",\"name\":\"SNS
\",\"start_time\":1.499473413112E9,\"end_time\":1.499473414071E9,\"parent_id\":
\"b29b548af4d54a0f\",\"inferred\":true,\"http\":{\"response\":{\"status\":200}},\"aws
\":{\"operation\":\"Publish\",\"region\":\"us-west-2\",\"request_id\":\"a2137970-
f6fc-5029-83e8-28aadeb99198\",\"retries\":0,\"topic_arn\":\"arn:aws:sns:us-
west-2:123456789012:awseb-e-ruag3jyweb-stack-NotificationTopic-6B829NT9V5O9\"},
\"trace_id\":\"1-59602603-23fc5b688855d396af79b496\",\"origin\":\"AWS::SNS\"}",
                    "Id": "1ee3c4a523f89ca5"
                }
            ],
            "Id": "1-59602603-23fc5b688855d396af79b496"
        }
    ],
    "UnprocessedTraceIds": []
}
```

完整追蹤包含每個區段的文件，該文件是從所有使用相同追蹤 ID 接收到的區段文件編譯而成。這些文件並不代表應用程式傳送至 X-Ray 的資料。相反，它們代表 X-Ray 服務生成的處理文檔。X-Ray 會編譯應用程式傳送的區段文件，並移除不符合區段文件結構描述的資料，以建立完整的追蹤文件。

X-Ray 也會針對未傳送區段本身的服務的下游呼叫建立推斷的區段。例如，當您使用已檢測的用戶端呼叫 DynamoDB 時，X-Ray SDK 會記錄一個子區段，其中包含有關呼叫的詳細資訊。但是，DynamoDB 不會傳送對應的區段。X-Ray 會使用子區段中的資訊建立推斷的區段，以代表追蹤對應中的 DynamoDB 資源，並將其新增至追蹤文件。

要從 API 獲取多個跟踪，您需要一個跟踪 ID 的列表，您可以`get-trace-summaries`使用AWS CLI查詢從的輸出中提取該列表。將清單重新導向`batch-get-traces`至的輸入，以取得特定時段的完整追蹤。

Example 取得一分鐘期間完整追蹤的指令碼

```
EPOCH=$(date +%s)
TRACEIDS=$(aws xray get-trace-summaries --start-time $(($EPOCH-120)) --end-time
 $(($EPOCH-60)) --query 'TraceSummaries[*].Id' --output text)
aws xray batch-get-traces --trace-ids $TRACEIDS --query 'Traces[*]'
```

擷取和精簡根本原因分析

使用 GetTraceSummaries API 產生追蹤摘要時，部分追蹤摘要可以 JSON 格式重複使用，以根據根本原因建立精簡的篩選器運算式。如需精簡步驟的演練，請參閱以下範例。

Example 範例 GetTraceSummaries 輸出-回應時間根本原因區段

```
{
  "Services": [
    {
      "Name": "GetWeatherData",
      "Names": ["GetWeatherData"],
      "AccountId": 123456789012,
      "Type": null,
      "Inferred": false,
      "EntityPath": [
        {
          "Name": "GetWeatherData",
          "Coverage": 1.0,
          'Remote": false
        },
        {
          "Name": "get_temperature",
          "Coverage": 0.8,
          "Remote": false
        }
      ]
    },
    {
      "Name": "GetTemperature",
      "Names": ["GetTemperature"],
      "AccountId": 123456789012,
      "Type": null,
      "Inferred": false,
      "EntityPath": [
        {
          "Name": "GetTemperature",
          "Coverage": 0.7,
          "Remote": false
        }
      ]
    }
  ]
```

```
}
```

透過編輯和忽略上述輸出，可利用此 JSON 篩選條件符合根本原因的實體。針對出現在 JSON 中的每一個欄位，任何待選項目必須完全相符，否則不會傳回追蹤。已移除的欄位會成為萬用字元值，與篩選條件表達式查詢結構相容的格式。

Example 重新格式化回應時間根本原因

```
{
  "Services": [
    {
      "Name": "GetWeatherData",
      "EntityPath": [
        {
          "Name": "GetWeatherData"
        },
        {
          "Name": "get_temperature"
        }
      ]
    },
    {
      "Name": "GetTemperature",
      "EntityPath": [
        {
          "Name": "GetTemperature"
        }
      ]
    }
  ]
}
```

然後，此 JSON 會透過呼叫 `rootcause.json = #[{}]`，做為篩選條件表達式的一部分使用。如需查詢篩選條件表達式的詳細資訊，請參閱篩選條件表達式一章。

Example 範例 JSON 篩選條件

```
rootcause.json = #[{ "Services": [ { "Name": "GetWeatherData", "EntityPath": [{ "Name":
 "GetWeatherData" }, { "Name": "get_temperature" } ] }, { "Name": "GetTemperature",
 "EntityPath": [ { "Name": "GetTemperature" } ] } ] }]
```

使用 AWS X-Ray API 設定抽樣、分組和加密設定

AWS X-Ray 提供可設定抽樣規則、群組規則和加密設定的 API。

章節

加密設定

使用PutEncryptionConfig來指定AWS Key Management Service(AWS KMS) 密鑰以用於加密。

> **Note**
>
> X-Ray 不支援非對稱 KMS 密鑰。

```
$ aws xray put-encryption-config --type KMS --key-id alias/aws/xray
{
    "EncryptionConfig": {
        "KeyId": "arn:aws:kms:us-east-2:123456789012:key/c234g4e8-39e9-4gb0-84e2-
b0ea215cbba5",
        "Status": "UPDATING",
        "Type": "KMS"
    }
}
```

針對金鑰 ID，您可以使用別名 (如範例中所示)、金鑰 ID 或 Amazon Resource Name (ARN)。

使用 GetEncryptionConfig 以取得目前的組態。當 X-Ray 完成應用設定時，狀態會從UPDATING至ACTIVE。

```
$ aws xray get-encryption-config
{
    "EncryptionConfig": {
        "KeyId": "arn:aws:kms:us-east-2:123456789012:key/c234g4e8-39e9-4gb0-84e2-
b0ea215cbba5",
        "Status": "ACTIVE",
```

```
        "Type": "KMS"
    }
}
```

若要停止使用 KMS 密鑰並使用默認加密，請將加密類型設為NONE。

```
$ aws xray put-encryption-config --type NONE
{
    "EncryptionConfig": {
        "Status": "UPDATING",
        "Type": "NONE"
    }
}
```

抽樣規則

您可以管理抽樣規則在您的帳戶中使用 X-Ray API。如需新增和管理標籤的詳細資訊，請參。標籤 X-Ray 取樣規則和羣組。

使用 GetSamplingRules 取得所有抽樣規則。

```
$ aws xray get-sampling-rules
{
    "SamplingRuleRecords": [
        {
            "SamplingRule": {
                "RuleName": "Default",
                "RuleARN": "arn:aws:xray:us-east-2:123456789012:sampling-rule/Default",
                "ResourceARN": "*",
                "Priority": 10000,
                "FixedRate": 0.05,
                "ReservoirSize": 1,
                "ServiceName": "*",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "*",
                "URLPath": "*",
                "Version": 1,
                "Attributes": {}
            },
            "CreatedAt": 0.0,
            "ModifiedAt": 1529959993.0
        }
```

```
    ]
}
```

預設規則會套用至不符合其他規則的所有請求。這是優先順序最低的規則，且無法刪除。不過，您可以使用 UpdateSamplingRule 變更速率和儲槽大小。

Example 的 API 輸入**UpdateSamplingRule**— 10000-scorekee.json

```
{
    "SamplingRuleUpdate": {
        "RuleName": "Default",
        "FixedRate": 0.01,
        "ReservoirSize": 0
    }
}
```

以下範例使用之前的檔案做為輸入，將預設規則變更為 1%、無儲槽。標籤是選擇性的。如果選擇添加標籤，則需要標籤鍵，標籤值是可選的。要從採樣規則中刪除現有標籤，請使用UntagResource

```
$ aws xray update-sampling-rule --cli-input-json file://1000-default.json --tags
 [{"Key": "key_name","Value": "value"},{"Key": "key_name","Value": "value"}]
{
    "SamplingRuleRecords": [
        {
            "SamplingRule": {
                "RuleName": "Default",
                "RuleARN": "arn:aws:xray:us-east-2:123456789012:sampling-rule/Default",
                "ResourceARN": "*",
                "Priority": 10000,
                "FixedRate": 0.01,
                "ReservoirSize": 0,
                "ServiceName": "*",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "*",
                "URLPath": "*",
                "Version": 1,
                "Attributes": {}
            },
            "CreatedAt": 0.0,
            "ModifiedAt": 1529959993.0
        },
```

使用 CreateSamplingRule 建立額外的抽樣規則。當您建立規則時，大多數規則欄位都必須填寫。以下範例將建立兩個規則。第一個規則會設定 Scorekeep 範例應用程式的基本速率。此規則適用於所有 API 提供但不符合更高優先順序之規則的請求。

Example 的 API 輸入**UpdateSamplingRule**— 9000-scorekeep.json

```
{
    "SamplingRule": {
        "RuleName": "base-scorekeep",
        "ResourceARN": "*",
        "Priority": 9000,
        "FixedRate": 0.1,
        "ReservoirSize": 5,
        "ServiceName": "Scorekeep",
        "ServiceType": "*",
        "Host": "*",
        "HTTPMethod": "*",
        "URLPath": "*",
        "Version": 1
    }
}
```

第二個規則也會套用至 Scorekeep，但它的優先順序更高且更具體。此規則會設定非常低的抽樣速率以輪詢請求。這些是用戶端每隔幾秒發出的 GET 請求，以檢查遊戲狀態的變更。

Example 的 API 輸入**UpdateSamplingRule**— 5000-polling-scorekeep.json

```
{
    "SamplingRule": {
        "RuleName": "polling-scorekeep",
        "ResourceARN": "*",
        "Priority": 5000,
        "FixedRate": 0.003,
        "ReservoirSize": 0,
        "ServiceName": "Scorekeep",
        "ServiceType": "*",
        "Host": "*",
        "HTTPMethod": "GET",
        "URLPath": "/api/state/*",
        "Version": 1
    }
}
```

標籤是選擇性的。如果選擇添加標籤，則需要標籤鍵，標籤值是可選的。

```
$ aws xray create-sampling-rule --cli-input-json file://5000-polling-scorekeep.json --
tags [{"Key": "key_name","Value": "value"},{"Key": "key_name","Value": "value"}]
{
    "SamplingRuleRecord": {
        "SamplingRule": {
            "RuleName": "polling-scorekeep",
            "RuleARN": "arn:aws:xray:us-east-1:123456789012:sampling-rule/polling-
scorekeep",
            "ResourceARN": "*",
            "Priority": 5000,
            "FixedRate": 0.003,
            "ReservoirSize": 0,
            "ServiceName": "Scorekeep",
            "ServiceType": "*",
            "Host": "*",
            "HTTPMethod": "GET",
            "URLPath": "/api/state/*",
            "Version": 1,
            "Attributes": {}
        },
        "CreatedAt": 1530574399.0,
        "ModifiedAt": 1530574399.0
    }
}
$ aws xray create-sampling-rule --cli-input-json file://9000-base-scorekeep.json
{
    "SamplingRuleRecord": {
        "SamplingRule": {
            "RuleName": "base-scorekeep",
            "RuleARN": "arn:aws:xray:us-east-1:123456789012:sampling-rule/base-
scorekeep",
            "ResourceARN": "*",
            "Priority": 9000,
            "FixedRate": 0.1,
            "ReservoirSize": 5,
            "ServiceName": "Scorekeep",
            "ServiceType": "*",
            "Host": "*",
            "HTTPMethod": "*",
            "URLPath": "*",
            "Version": 1,
            "Attributes": {}
```

```
        },
        "CreatedAt": 1530574410.0,
        "ModifiedAt": 1530574410.0
    }
}
```

若要刪除抽樣規則，請使用 DeleteSamplingRule。

```
$ aws xray delete-sampling-rule --rule-name polling-scorekeep
{
    "SamplingRuleRecord": {
        "SamplingRule": {
            "RuleName": "polling-scorekeep",
            "RuleARN": "arn:aws:xray:us-east-1:123456789012:sampling-rule/polling-
scorekeep",
            "ResourceARN": "*",
            "Priority": 5000,
            "FixedRate": 0.003,
            "ReservoirSize": 0,
            "ServiceName": "Scorekeep",
            "ServiceType": "*",
            "Host": "*",
            "HTTPMethod": "GET",
            "URLPath": "/api/state/*",
            "Version": 1,
            "Attributes": {}
        },
        "CreatedAt": 1530574399.0,
        "ModifiedAt": 1530574399.0
    }
}
```

群組

您可以使用 X-Ray API 管理您的帳戶中的羣組。群組是一系列追蹤，為篩選條件表達式所定義。您可以使用羣組生成其他服務圖表並提供 Amazon CloudWatch 指標。請參閱從中獲取數據 AWS X-Ray以進一步了解透過 X-Ray API 使用服務圖表及指標的詳細資訊。如需組的詳細資訊，請參。設定群組。如需新增和管理標籤的詳細資訊，請參。標籤 X-Ray 取樣規則和羣組。

使用 CreateGroup 建立群組 標籤是選擇性的。如果選擇添加標籤，則需要標籤鍵，標籤值是可選的。

```
$ aws xray create-group --group-name "TestGroup" --filter-expression
 "service(\"example.com\") {fault}" --tags [{"Key": "key_name","Value": "value"},
{"Key": "key_name","Value": "value"}]
{
    "GroupName": "TestGroup",
    "GroupARN": "arn:aws:xray:us-east-2:123456789012:group/TestGroup/UniqueID",
    "FilterExpression": "service(\"example.com\") {fault OR error}"
}
```

使用 GetGroups 取得所有現有群組。

```
$ aws xray get-groups
{
    "Groups": [
        {
            "GroupName": "TestGroup",
            "GroupARN": "arn:aws:xray:us-east-2:123456789012:group/TestGroup/UniqueID",
            "FilterExpression": "service(\"example.com\") {fault OR error}"
        },
  {
            "GroupName": "TestGroup2",
            "GroupARN": "arn:aws:xray:us-east-2:123456789012:group/TestGroup2/
UniqueID",
            "FilterExpression": "responsetime > 2"
        }
    ],
 "NextToken": "tokenstring"
}
```

使用 UpdateGroup 更新群組 標籤是選擇性的。如果選擇添加標籤，則需要標籤鍵，標籤值是可選的。若要從組中移除現有標籤，請使用UntagResource。

```
$ aws xray update-group --group-name "TestGroup" --group-arn "arn:aws:xray:us-
east-2:123456789012:group/TestGroup/UniqueID" --filter-expression
 "service(\"example.com\") {fault OR error}" --tags [{"Key": "Stage","Value": "Prod"},
{"Key": "Department","Value": "QA"}]
{
    "GroupName": "TestGroup",
    "GroupARN": "arn:aws:xray:us-east-2:123456789012:group/TestGroup/UniqueID",
    "FilterExpression": "service(\"example.com\") {fault OR error}"
}
```

用 DeleteGroup 刪除群組。

```
$ aws xray delete-group --group-name "TestGroup" --group-arn "arn:aws:xray:us-
east-2:123456789012:group/TestGroup/UniqueID"
    {
    }
```

使用取樣規則搭配 X-Ray API

所以此AWS X-Ray開發套件會使用 X-Ray API 取得抽樣規則、報告抽樣結果及取得配額。您可以使用這些 API 來更進一步了解抽樣規則的運作方式，或是使用 X-Ray 開發套件不支援的語言實作抽樣。

從使用 GetSamplingRules 取得所有抽樣規則開始。

```
$ aws xray get-sampling-rules
{
    "SamplingRuleRecords": [
        {
            "SamplingRule": {
                "RuleName": "Default",
                "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/Default",
                "ResourceARN": "*",
                "Priority": 10000,
                "FixedRate": 0.01,
                "ReservoirSize": 0,
                "ServiceName": "*",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "*",
                "URLPath": "*",
                "Version": 1,
                "Attributes": {}
            },
            "CreatedAt": 0.0,
            "ModifiedAt": 1530558121.0
        },
        {
            "SamplingRule": {
                "RuleName": "base-scorekeep",
                "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/base-scorekeep",
                "ResourceARN": "*",
                "Priority": 9000,
```

```
                "FixedRate": 0.1,
                "ReservoirSize": 2,
                "ServiceName": "Scorekeep",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "*",
                "URLPath": "*",
                "Version": 1,
                "Attributes": {}
            },
            "CreatedAt": 1530573954.0,
            "ModifiedAt": 1530920505.0
        },
        {
            "SamplingRule": {
                "RuleName": "polling-scorekeep",
                "RuleARN": "arn:aws:xray:us-east-1::sampling-rule/polling-scorekeep",
                "ResourceARN": "*",
                "Priority": 5000,
                "FixedRate": 0.003,
                "ReservoirSize": 0,
                "ServiceName": "Scorekeep",
                "ServiceType": "*",
                "Host": "*",
                "HTTPMethod": "GET",
                "URLPath": "/api/state/*",
                "Version": 1,
                "Attributes": {}
            },
            "CreatedAt": 1530918163.0,
            "ModifiedAt": 1530918163.0
        }
    ]
}
```

輸出會包含預設規則和自訂規則。若您尚未建立抽樣規則，請參閱抽樣規則。

依照遞增優先順序排序，針對傳入請求評估規則。當規則相符時，使用固定速率和儲槽大小來進行抽樣決策。記錄抽樣請求並忽略 (針對追蹤用途) 未抽樣請求。在完成抽樣決策後停止評估規則。

規則儲槽大小是在套用固定速率前，每秒要記錄的追蹤目標數。儲槽會累積套用到所有服務，因此您無法直接使用它。但是，若該值並非零，您便可以從儲槽每秒借用一個追蹤，直到 X-Ray 指派配額為

止。在接收配額前，記錄每秒的第一個請求，然後將固定速率套用到其他請求。固定速率是介於 0 和 1.00 (100%) 間的十進位數。

以下範例會顯示對 GetSamplingTargets 發出的呼叫，其中包含在過去 10 秒間進行的抽樣決策詳細資訊。

```
$ aws xray get-sampling-targets --sampling-statistics-documents '[
    {
        "RuleName": "base-scorekeep",
        "ClientID": "ABCDEF1234567890ABCDEF10",
        "Timestamp": "2018-07-07T00:20:06",
        "RequestCount": 110,
        "SampledCount": 20,
        "BorrowCount": 10
    },
    {
        "RuleName": "polling-scorekeep",
        "ClientID": "ABCDEF1234567890ABCDEF10",
        "Timestamp": "2018-07-07T00:20:06",
        "RequestCount": 10500,
        "SampledCount": 31,
        "BorrowCount": 0
    }
]'
{
    "SamplingTargetDocuments": [
        {
            "RuleName": "base-scorekeep",
            "FixedRate": 0.1,
            "ReservoirQuota": 2,
            "ReservoirQuotaTTL": 1530923107.0,
            "Interval": 10
        },
        {
            "RuleName": "polling-scorekeep",
            "FixedRate": 0.003,
            "ReservoirQuota": 0,
            "ReservoirQuotaTTL": 1530923107.0,
            "Interval": 10
        }
    ],
    "LastRuleModification": 1530920505.0,
    "UnprocessedStatistics": []
```

```
}
```

X-Ray 的回應包含要使用的配額，而非從儲槽借用。在此範例中，服務在 10 秒間從儲槽借用 10 個追蹤，並將 10% 的固定速率套用到其他 100 個請求，其結果合計為 20 個抽樣請求。配額會五分鐘內有效 (以存續期間表示)，或是直到指派新的配額。X-Ray 也可以指派比預設值更長的報告間隔 (雖然在此處並未執行此作業)。

> **Note**
>
> 您第一次進行呼叫時，來自 X-Ray 的回應可能不會包含配額。繼續從儲存借用，直到您獲得指派配額。

回應中的其他兩個欄位可能會指出輸入的問題。針對您最後一次呼叫 `GetSamplingRules` 檢查 `LastRuleModification`。若其較新，請取得規則的新複本。`UnprocessedStatistics` 可包含錯誤，指出規則已遭到刪除、輸入中的統計文件過舊，或是許可錯誤。

AWS X-Ray 區段文件

追蹤區段是您應用程式所處理請求的 JSON 表示。追蹤區段會記錄原始要求的相關資訊、應用程式在本機執行之工作的相關資訊，以及子區段，其中包含應用程式對 AWS 資源、HTTP API 和 SQL 資料庫進行之下游呼叫的相關資訊。

區段文件會將有關區段的資訊傳達至 X-Ray。區段文件最多可達 64 kB，且包含具有子區段的整個區段、指出要求正在進行的區段片段，或是個別傳送的單一子區段。您可以使用 `PutTraceSegments`API 將區段文件直接傳送至 X-Ray。

X-Ray 會編譯和處理區段文件，以產生可查詢的追蹤摘要和完整追蹤，您可以分別使用`GetTraceSummaries`和 `BatchGetTraces`API 存取這些摘要。除了您傳送至 X-Ray 的區段和子區段之外，服務還會使用子區段中的資訊來產生推斷的區段，並將其新增至完整追蹤。推斷的區段代表追蹤對映中的下游服務和資源。

X-Ray 提供區段文件的 JSON 結構定義。您可以在此處下載架構：xray-segmentdocument-schema-v1.0.0。以下章節會更詳細的說明結構描述中所列出的欄位和物件。

區段欄位的子集會由 X-Ray 編製索引，以便與篩選運算式搭配使用。例如，如果您將區段上的`user`欄位設定為唯一識別碼，您可以在 X-Ray 主控台或使用 `GetTraceSummaries` API 搜尋與特定使用者相關聯的區段。如需詳細資訊，請參閱 使用篩選運算式。

當您使用 X-Ray SDK 檢測應用程式時，SDK 會為您產生區段文件。SDK 不會將區段文件直接傳送至 X-Ray，而是透過本機 UDP 連接埠將它們傳送至 X-Ray 精靈。如需詳細資訊，請參閱 將區段文件傳送至 X-Ray 精靈。

章節

區段欄位

區段會記錄您應用程式所處理請求的相關追蹤資訊。區段最少會記錄請求的名稱、ID、開始時間、追蹤 ID 及結束時間。

Example 最小的完成區段

```
{
  "name" : "example.com",
  "id" : "70de5b6f19ff9a0a",
  "start_time" : 1.478293361271E9,
  "trace_id" : "1-581cf771-a006649127e371903a2de979",
  "end_time" : 1.478293361449E9
}
```

以下欄位為區段的必要項目或條件式必要項目。

> **Note**
>
> 除非另有說明，否則值必須為字串 (最多 250 個字元)。

必要的區段欄位

- name— 處理要求之服務的邏輯名稱，最多 200 個字元。例如，您應用程式的名稱或網域名稱。名稱可包含 Unicode 字母、數字、空格和下列符號：_、.、:、/、%、&、#、=、+、\、-、@
- id— 區段的 64 位元識別碼，在相同軌跡中的區段中唯一，以 16 個十六進位數字表示。
- trace_id— 連接源自單一用戶端要求的所有區段和子區段的唯一識別碼。

  X-Ray 軌跡 ID 格式

  X-Ray trace_id 由三個用連字符分隔的數字組成。例如 1-58406520-a006649127e371903a2de979。其中包含：

  - 版本號碼，也就是1。
  - 原始請求的時間在 Unix 紀元時間使用 8 個十六進制數字。

    例如，2016 年 12 月 1 日上午 10:00 PST (以紀元時間表示) 為1480615200秒或十六進58406520位數字。
  - 追蹤的全域唯一 96 位元識別碼，以 24 個十六進位數字顯示。

> **Note**
>
> X-Ray 現在支援使用以 OpenTelemetry 及符合 W3C 追蹤上下文規格的任何其他架構建立的追蹤 ID。傳送至 X-Ray 時，W3C 追蹤識別碼必須格式化為 X-Ray 追蹤 ID 格式。例如，W3C 追蹤識別碼4efaaf4d1e8720b39541901950019ee5應格式化為傳送至 X-Ray 1-4efaaf4d-1e8720b39541901950019ee5 時。X-Ray 跟踪 ID 包括 Unix 紀元時間中的原始請求時間戳，但是當以 X-Ray 格式發送 W3C 跟踪 ID 時，這不是必需的。

> **追蹤 ID 安全**
>
> 您可在回應標頭中取得追蹤 ID。使用安全的隨機演算法產生追蹤 ID，以確保攻擊者無法計算未來的追蹤 ID，並使用這些 ID 傳送請求到您的應用程式。

- start_time— 數字，表示區段建立的時間，以紀元時間為單位的浮點秒數。例如 1480615200.010 或 1.480615200010E9。視需要使用任意小數位數。建議盡可能使用毫秒解析度。
- end_time-數字，即段被關閉的時間。例如 1480615200.090 或 1.480615200090E9。指定 end_time 或 in_progress。

- in_progress— 布林值，設定為true而非指定end_time以記錄區段已啟動但未完成。應用程式收到的請求需要很長時間時，請傳送進行中區段，以追蹤請求的接收情況。回應已傳送時，請傳送完成區段以覆寫進行中的區段。針對每個請求，請只傳送一個完成區段，以及一或零個進行中區段。

> **服務名稱**
>
> 區段name應與產生區段之服務的網域名稱或邏輯名稱相符。但是，這不是強制執行的。具有PutTraceSegments可傳送任何名稱之區段之權限的任何應用程式。

以下欄位是區段的選擇性欄位。

選擇性區段欄位

- service— 包含應用程式相關資訊的物件。
  - version— 字串，用來識別提供要求的應用程式版本。
- user— 識別傳送要求之使用者的字串。
- origin— 執行應用程式的 AWS 資源類型。

  支援值

  - AWS::EC2::Instance— 一個 Amazon EC2 實例。
  - AWS::ECS::Container— Amazon ECS 容器.
  - AWS::ElasticBeanstalk::Environment— Elastic Beanstalk 環境。

  當有多個值適用您的應用程式時，請使用最具指定性的。例如，多容器泊塢視窗 Elastic Beanstalk 環境會在 Amazon ECS 容器上執行您的應用程式，然後在 Amazon EC2 執行個體上執行。在此案例中，您會將來源設為 AWS::ElasticBeanstalk::Environment，因為環境是其他兩個資源的父系。
- parent_id— 您指定的子區段 ID 是否來自已檢測的應用程式的要求。X-Ray SDK 會將父子區段識別碼新增至追蹤標頭，以供下游 HTTP 呼叫使用。在巢狀子區段的情況下，子區段可以有一個區段或子區段作為其父項。
- http— 包含原始 HTTP 要求相關資訊的http物件。
- aws— aws含有應用程式提供要求之 AWS 資源相關資訊的物件。
- error、throttlefault、和 cause — 錯誤欄位，指出發生錯誤，其中包含造成錯誤之例外狀況的相關資訊。

- annotations— 具有您希望 X-Ray 索引以進行搜尋的索引鍵值配對的annotations物件。
- metadata— metadata含有您要儲存在區段中的任何其他資料的物件。
- subsegments— subsegment物件陣列。

子區段

您可以建立子區段，以記錄您使用 AWS SDK 進行的呼叫 AWS 服務 和資源、呼叫內部或外部 HTTP Web API 或 SQL 資料庫查詢。您也可以建立子區段，除錯或標註您應用程式中的程式碼區塊。子區段可包含其他子區段，因此記錄內部函數呼叫中繼資料的自訂子區段可包含其他自訂子區段及下游呼叫的子區段。

子區段會從呼叫該呼叫的服務的角度記錄下游呼叫。X-Ray 使用子區段來識別不傳送區段的下游服務，並在服務圖上為其建立項目。

子區段可內嵌在完整區段文件中，或是分別傳送。將子區段分別傳送，來非同步追蹤長時間執行請求的下游呼叫，或是避免超過區段文件大小上限。

Example 具有內嵌子區段的區段

獨立子區段具備 subsegment 的 type，以及可識別父區段的 parent_id。

```
{
  "trace_id"   : "1-5759e988-bd862e3fe1be46a994272793",
  "id"         : "defdfd9912dc5a56",
  "start_time" : 1461096053.37518,
  "end_time"   : 1461096053.4042,
  "name"       : "www.example.com",
  "http"       : {
    "request"  : {
      "url"        : "https://www.example.com/health",
      "method"     : "GET",
      "user_agent" : "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6)
 AppleWebKit/601.7.7",
      "client_ip"  : "11.0.3.111"
    },
    "response" : {
      "status"         : 200,
      "content_length" : 86
    }
  },
  "subsegments" : [
```

```
    {
      "id"         : "53995c3f42cd8ad8",
      "name"       : "api.example.com",
      "start_time" : 1461096053.37769,
      "end_time"   : 1461096053.40379,
      "namespace"  : "remote",
      "http"       : {
        "request"  : {
          "url"    : "https://api.example.com/health",
          "method" : "POST",
          "traced" : true
        },
        "response" : {
          "status"         : 200,
          "content_length" : 861
        }
      }
    }
  ]
}
```

對於長時間執行的請求，您可以傳送進行中的區段，通知 X-Ray 已收到請求，然後在完成原始請求之前分別傳送子區段以追蹤它們。

Example 進行中的區段

```
{
  "name" : "example.com",
  "id" : "70de5b6f19ff9a0b",
  "start_time" : 1.478293361271E9,
  "trace_id" : "1-581cf771-a006649127e371903a2de979",
  "in_progress": true
}
```

Example 獨立子區段

獨立子區段具備 subsegment 的 type、trace_id 及 parent_id，可識別父區段。

```
{
  "name" : "api.example.com",
  "id" : "53995c3f42cd8ad8",
  "start_time" : 1.478293361271E9,
  "end_time" : 1.478293361449E9,
```

```
  "type" : "subsegment",
  "trace_id" : "1-581cf771-a006649127e371903a2de979"
  "parent_id" : "defdfd9912dc5a56",
  "namespace"  : "remote",
  "http"       : {
      "request"  : {
          "url"    : "https://api.example.com/health",
          "method" : "POST",
          "traced" : true
      },
      "response" : {
          "status"         : 200,
          "content_length" : 861
      }
  }
}
```

當請求完成時，透過使用 end_time 重新傳送它來關閉區段。完整區段會覆寫正在進行中的區段。

您也可以為觸發非同步工作流程的已完成請求分別傳送子區段。例如，web API 可能會在開始使用者請求的工作前立即傳回 OK 200 回應。您可以在傳送回應後立即將完整區段傳送至 X-Ray，接著再傳送子區段以供稍後完成的工作。與區段相同，您也可以傳送子區段片段來記錄子區段已啟動，然後在完成下游呼叫時使用完整的子區段覆寫它。

以下欄位為子區段的必要項目或條件式必要項目。

> **Note**
>
> 除非另有說明，否則值必須為字串 (最多 250 個字元)。

必要的子區段欄位

- id— 子區段的 64 位元識別碼，在相同軌跡中的區段中唯一，以 16 個十六進位數字表示。
- name— 子區段的邏輯名稱。針對下游呼叫，請在呼叫資源或服務後命名子區段。針對自訂子區段，請在其檢測的程式碼 (例如函數名稱) 之後命名子區段。
- start_time— 數字，即創建子段的時間，以時代時間為單位的浮點秒，精確到毫秒。例如 1480615200.010 或 1.480615200010E9。
- end_time-數字，即子段關閉的時間。例如 1480615200.090 或 1.480615200090E9。指定 end_time 或 in_progress。

- in_progress— 設定為true而非指定以記錄子區段已啟動但未完成的布林值。end_time針對每個下游請求，請只傳送一個完成子區段，以及一或零個進行中子區段。
- trace_id— 子區段父區段的追蹤 ID。僅在分別傳送子區段時才為必要項目。

  X-Ray 軌跡 ID 格式

  X-Ray trace_id 由三個用連字符分隔的數字組成。例如 1-58406520-a006649127e371903a2de979。其中包含：

  - 版本號碼，也就是1。
  - 原始請求的時間在 Unix 紀元時間使用 8 個十六進制數字。

    例如，2016 年 12 月 1 日上午 10:00 PST (以紀元時間表示) 為1480615200秒或十六進58406520位數字。
  - 追蹤的全域唯一 96 位元識別碼，以 24 個十六進位數字顯示。

  > **Note**
  >
  > X-Ray 現在支援使用以 OpenTelemetry 及符合 W3C 追蹤上下文規格的任何其他架構建立的追蹤 ID。傳送至 X-Ray 時，W3C 追蹤識別碼必須格式化為 X-Ray 追蹤 ID 格式。例如，W3C 追蹤識別碼4efaaf4d1e8720b39541901950019ee5應格式化為傳送至 X-Ray 1-4efaaf4d-1e8720b39541901950019ee5 時。X-Ray 跟踪 ID 包括 Unix 紀元時間中的原始請求時間戳，但是當以 X-Ray 格式發送 W3C 跟踪 ID 時，這不是必需的。

- parent_id— 子區段父項區段的區段 ID。僅在分別傳送子區段時才為必要項目。在巢狀子區段的情況下，子區段可以有一個區段或子區段作為其父項。
- type—subsegment. 僅在分別傳送子區段時才需要。

以下欄位是子區段的選擇性欄位。

選擇性子區段欄位

- namespace— aws 適用於 AWS 開發套件呼叫；remote適用於其他下游呼叫。
- http— 包http含傳出 HTTP 呼叫相關資訊的物件。
- aws— 包含應用程式呼叫之下游 AWS 資源相關資訊的aws物件。
- error、throttlefault、和 cause — 錯誤欄位，指出發生錯誤，其中包含造成錯誤之例外狀況的相關資訊。

- annotations— 具有您希望 X-Ray 索引以進行搜尋的索引鍵值配對的annotations物件。
- metadata— metadata含有您要儲存在區段中的任何其他資料的物件。
- subsegments— subsegment物件陣列。
- precursor_ids— 子區段 ID 陣列，可識別在此子區段之前完成的相同父項的子區段。

HTTP 請求資料

使用 HTTP 區塊來記錄您應用程式所處理的 HTTP 請求相關詳細資訊 (位於區段中)，或是您應用程式對下游 HTTP API 所發出請求的相關詳細資訊 (位於子區段中)。此物件中大部分的欄位都會映射到 HTTP 請求和回應中找到的資訊。

http

所有欄位都是選擇性的。

- request— 有關請求的資訊。
  - method— 請求方法。例如 GET。
  - url— 請求的完整 URL，從請求的協議，主機名和路徑編譯。
  - user_agent— 來自請求者用戶端的使用者代理程式字串。
  - client_ip— 要求者的 IP 位址。可從 IP 封包的 Source Address 擷取，或是針對轉送的請求，則從 X-Forwarded-For 標頭擷取。
  - x_forwarded_for— (僅限區段) 布林值，表示client_ip已從X-Forwarded-For標頭讀取且不可靠，因為它可能已偽造。
  - traced— (僅限子區段) 布林值，表示下游呼叫是至另一個追蹤服務。如果將此欄位設定為true，X-Ray 會將追蹤視為中斷，直到下游服務上傳符合包含此區塊id之子區段的區段為止。parent_id
- response— 有關響應的信息。
  - status— 整數，表示回應的 HTTP 狀態。
  - content_length— 整數，指出以位元組為單位的回應主體長度。

當您檢測對下游 Web API 的呼叫時，請記錄一個子區段，其中包含 HTTP 要求和回應的相關資訊。X-Ray 會使用子區段來產生遠端 API 的推斷區段。

Example 在 Amazon EC2 上執行之應用程式提供的 HTTP 呼叫區段

```
{
  "id": "6b55dcc497934f1a",
  "start_time": 1484789387.126,
  "end_time": 1484789387.535,
  "trace_id": "1-5880168b-fd5158284b67678a3bb5a78c",
  "name": "www.example.com",
  "origin": "AWS::EC2::Instance",
  "aws": {
    "ec2": {
      "availability_zone": "us-west-2c",
      "instance_id": "i-0b5a4678fc325bg98"
    },
    "xray": {
        "sdk_version": "2.11.0 for Java"
    },
  },
  "http": {
    "request": {
      "method": "POST",
      "client_ip": "78.255.233.48",
      "url": "http://www.example.com/api/user",
      "user_agent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:45.0) Gecko/20100101
 Firefox/45.0",
      "x_forwarded_for": true
    },
    "response": {
      "status": 200
    }
  }
```

Example 下游 HTTP 呼叫的子區段

```
{
  "id": "004f72be19cddc2a",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "name": "names.example.com",
  "namespace": "remote",
  "http": {
    "request": {
      "method": "GET",
```

```
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  }
}
```

Example 下游 HTTP 呼叫的推斷區段

```
{
  "id": "168416dc2ea97781",
  "name": "names.example.com",
  "trace_id": "1-62be1272-1b71c4274f39f122afa64eab",
  "start_time": 1484786387.131,
  "end_time": 1484786387.501,
  "parent_id": "004f72be19cddc2a",
  "http": {
    "request": {
      "method": "GET",
      "url": "https://names.example.com/"
    },
    "response": {
      "content_length": -1,
      "status": 200
    }
  },
  "inferred": true
}
```

註釋

區段和子區段可包含一個annotations物件，其中包含 X-Ray 索引的一個或多個欄位，以便與篩選運算式搭配使用。欄位可以有字串、數字或布林值 (沒有物件或陣列)。X-Ray 索引每個軌跡最多 50 個註釋。

Example HTTP 呼叫區段與標註

```
{
  "id": "6b55dcc497932f1a",
  "start_time": 1484789187.126,
```

```
  "end_time": 1484789187.535,
  "trace_id": "1-5880168b-fd515828bs07678a3bb5a78c",
  "name": "www.example.com",
  "origin": "AWS::EC2::Instance",
  "aws": {
    "ec2": {
      "availability_zone": "us-west-2c",
      "instance_id": "i-0b5a4678fc325bg98"
    },
    "xray": {
        "sdk_version": "2.11.0 for Java"
    },
  },
  "annotations": {
    "customer_category" : 124,
    "zip_code" : 98101,
    "country" : "United States",
    "internal" : false
  },
  "http": {
    "request": {
      "method": "POST",
      "client_ip": "78.255.233.48",
      "url": "http://www.example.com/api/user",
      "user_agent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:45.0) Gecko/20100101
Firefox/45.0",
      "x_forwarded_for": true
    },
    "response": {
      "status": 200
    }
  }
```

鍵必須為英數字元，才能使用篩選條件。允許使用底線。不允許使用其他符號和空格。

中繼資料

區段和子區段可以包含一或多個欄位的metadata物件，其中包含任何類型的值，包括物件和陣列。X-Ray 不會為中繼資料建立索引，而且值可以是任何大小，只要區段文件不超過最大大小 (64 kB) 即可。您可以在 BatchGetTraces API 傳回的完整區段文件中檢視中繼資料。以開頭的欄位金鑰 (debug在下列範例中) AWS. 會保留給 AWS提供的 SDK 和用戶端使用。

Example 使用中繼資料的自訂子區段

```
{
  "id": "0e58d2918e9038e8",
  "start_time": 1484789387.502,
  "end_time": 1484789387.534,
  "name": "## UserModel.saveUser",
  "metadata": {
    "debug": {
      "test": "Metadata string from UserModel.saveUser"
    }
  },
  "subsegments": [
    {
      "id": "0f910026178b71eb",
      "start_time": 1484789387.502,
      "end_time": 1484789387.534,
      "name": "DynamoDB",
      "namespace": "aws",
      "http": {
        "response": {
          "content_length": 58,
          "status": 200
        }
      },
      "aws": {
        "table_name": "scorekeep-user",
        "operation": "UpdateItem",
        "request_id": "3AIENM5J4ELQ3SPODHKBIRVIC3VV4KQNSO5AEMVJF66Q9ASUAAJG",
        "resource_names": [
          "scorekeep-user"
        ]
      }
    }
  ]
}
```

AWS 資源資料

針對區段，aws 物件包含您應用程式於其上執行的資源相關資訊。可將多個欄位套用至單一資源。例如，在 Elastic Beanstalk 上的多容器 Docker 環境中執行的應用程式可能具有關於 Amazon EC2 執行個體、在執行個體上執行的 Amazon ECS 容器，以及 Elastic Beanstalk 環境本身的相關資訊。

aws (區段)

所有欄位都是選擇性的。

- account_id— 如果您的應用程式將區段傳送至其他區段 AWS 帳戶，請記錄執行應用程式的帳戶 ID。
- cloudwatch_logs— 描述單一 CloudWatch 記錄群組的物件陣列。
  - log_group—「記 CloudWatch 錄群組」名稱。
  - arn— 記 CloudWatch 錄群組 ARN。
- ec2— 有關 Amazon EC2 執行個體的資訊。
  - instance_id— EC2 執行個體的執行個體識別碼。
  - instance_size— EC2 執行個體的類型。
  - ami_id— Amazon 機器圖像 ID.
  - availability_zone— 執行執行個體所在的可用區域。
- ecs— 有關 Amazon ECS 容器的信息。
  - container— 容器的主機名稱。
  - container_id— 容器的完整容器 ID。
  - container_arn— 容器執行個體的 ARN。
- eks— 有關 Amazon EKS 集群的信息。
  - pod— 您的 EKS 網繭的主機名稱。
  - cluster_name— EKS 叢集名稱。
  - container_id— 容器的完整容器 ID。
- elastic_beanstalk— 有關 Elastic Beanstalk 環境的信息。您可以在最新的 Elastic Beanstalk 平台/var/elasticbeanstalk/xray/environment.conf上命名的文件中找到此信息。
  - environment_name - 環境名稱。
  - version_label— 目前部署至提供要求之執行個體的應用程式版本名稱。
  - deployment_id— 代表上次成功部署至提供要求之執行個體的識別碼的數字。
- xray— 有關所使用儀器類型和版本的中繼資料。
  - auto_instrumentation— 布林值，指出是否使用自動分析 (例如，Java 代理程式)。
  - sdk_version— 所使用的 SDK 或代理程式版本。
  - sdk— SDK 的類型。

Example AWS 阻止與插件

```
"aws":{
   "elastic_beanstalk":{
      "version_label":"app-5a56-170119_190650-stage-170119_190650",
      "deployment_id":32,
      "environment_name":"scorekeep"
   },
   "ec2":{
      "availability_zone":"us-west-2c",
      "instance_id":"i-075ad396f12bc325a",
      "ami_id":
   },
   "cloudwatch_logs":[
      {
         "log_group":"my-cw-log-group",
         "arn":"arn:aws:logs:us-west-2:012345678912:log-group:my-cw-log-group"
      }
   ],
   "xray":{
      "auto_instrumentation":false,
      "sdk":"X-Ray for Java",
      "sdk_version":"2.8.0"
   }
}
```

對於子區段，請記錄應用程式存取的 AWS 服務 和資源的相關資訊。X-Ray 會使用此資訊來建立表示服務對映中下游服務的推論區段。

aws (子區段)

所有欄位都是選擇性的。

- `operation`— 針對 AWS 服務 或資源叫用的 API 動作名稱。
- `account_id`— 如果您的應用程式存取不同帳戶中的資源，或將區段傳送至其他帳戶，請記錄擁有應用程式存取之 AWS 資源的帳號 ID。
- `region`— 如果資源位於與您的應用程式不同的區域，請記錄該區域。例如 `us-west-2`。
- `request_id`— 要求的唯一識別碼。
- `queue_url`— 對於 Amazon SQS 佇列上的操作，則為佇列的 URL。
- `table_name`— 對於 DynamoDB 資料表上的作業，則為資料表的名稱。

Example 呼叫 DynamoDB 以儲存項目的子區段

```
{
  "id": "24756640c0d0978a",
  "start_time": 1.480305974194E9,
  "end_time": 1.4803059742E9,
  "name": "DynamoDB",
  "namespace": "aws",
  "http": {
    "response": {
      "content_length": 60,
      "status": 200
    }
  },
  "aws": {
    "table_name": "scorekeep-user",
    "operation": "UpdateItem",
    "request_id": "UBQNSO5AEM8T4FDA4RQDEB94OVTDRVV4K4HIRGVJF66Q9ASUAAJG",
  }
}
```

錯誤和例外狀況

當發生錯誤時，您可以記錄錯誤及其產生異常的相關詳細資訊。在您應用程式將錯誤傳回使用者時於區段中記錄錯誤，或是在下游呼叫傳回錯誤時於子區段中記錄錯誤。

錯誤類型

將一或多個以下欄位設為 `true` 來指出發生錯誤。若錯誤發生複合，則可套用多個類型。例如，來自下游呼叫的 `429 Too Many Requests` 錯誤可能會造成您的應用程式傳回 `500 Internal Server Error`；在這種情況下，即會套用三種類型。

- `error`— 布林值，表示發生用戶端錯誤 (回應狀態碼為 4XX 用戶端錯誤)。
- `throttle`— 布林值，表示要求已限制 (回應狀態碼為 429 要求太多)。
- `fault`— 布林值，表示發生伺服器錯誤 (回應狀態碼為 5XX 伺服器錯誤)。

透過在區段或子區段中包含一個原因 (cause) 物件來指出錯誤的原因。

`cause`

原因可以是 16 字元的異常 ID，或是具備以下欄位的物件：

- working_directory— 發生異常時工作目錄的完整路徑。
- paths— 發生例外狀況時所使用的程式庫或模組的路徑陣列。
- exceptions— 例外狀況物件的陣列。

在一或多個異常 (exception) 物件中包含錯誤的詳細資訊。

exception

所有欄位都是選擇性的。

- id— 例外狀況的 64 位元識別碼，在相同追蹤中的區段中是唯一的，以 16 個十六進位數字表示。
- message— 例外狀況訊息。
- type— 例外狀況類型。
- remote— 布林值，表示例外狀況是由下游服務傳回的錯誤所造成。
- truncated— 整數，表示從省略的堆疊影格數目stack。
- skipped— 整數，表示在此例外狀況與其子系之間略過的例外狀況數目，也就是它所造成的例外狀況。
- cause— 例外父項的例外狀況識別碼，也就是造成此例外狀況的例外狀況。
- stack— stackFrame 物件的陣列。

若可用的話，在 stackFrame 物件中記錄呼叫堆疊的相關資訊。

stackFrame

所有欄位都是選擇性的。

- path— 檔案的相對路徑。
- line— 檔案中的行。
- label— 函數或方法名稱。

SQL 查詢

您可以建立為您應用程式對 SQL 資料庫發出的查詢建立子區段。

sql

所有欄位都是選擇性的。

- connection_string— 對於不使用 URL 連接字串的 SQL Server 或其他資料庫連線，請記錄連接字串 (不包括密碼)。
- url— 對於使用 URL 連線字串的資料庫連線，請記錄 URL，但不包括密碼。
- sanitized_query— 資料庫查詢，其中任何使用者提供的值會移除或取代為預留位置。
- database_type— 資料庫引擎的名稱。
- database_version— 資料庫引擎的版本號碼。
- driver_version— 應用程式使用的資料庫引擎驅動程式的名稱和版本號碼。
- user— 資料庫使用者名稱。
- preparation— call 如果查詢使用了PreparedCall；statement如果查詢使用PreparedStatement.

Example 使用 SQL 查詢的子區段

```
{
  "id": "3fd8634e78ca9560",
  "start_time": 1484872218.696,
  "end_time": 1484872218.697,
  "name": "ebdb@aawijb5u25wdoy.cpamxznpdoq8.us-west-2.rds.amazonaws.com",
  "namespace": "remote",
  "sql" : {
    "url": "jdbc:postgresql://aawijb5u25wdoy.cpamxznpdoq8.us-
west-2.rds.amazonaws.com:5432/ebdb",
    "preparation": "statement",
    "database_type": "PostgreSQL",
    "database_version": "9.5.4",
    "driver_version": "PostgreSQL 9.4.1211.jre7",
    "user" : "dbuser",
    "sanitized_query" : "SELECT  *  FROM  customers  WHERE  customer_id=?;"
  }
}
```

的文件歷史記錄 AWS X-Ray

下表說明的文件的重要變更 AWS X-Ray。如需有關此文件更新的通知，您可以訂閱 RSS 訂閱源。

最新文件更新：2023 年 2 月 8 日

變更	描述	日期
已新增的功能	X-Ray 現在會記錄資料事件 `PutTraceSegments` ，包括`GetTraceSummaries` 、和`BatchGetTraces` 目標 AWS CloudTrail。X-Ray 現在也會將`GetSamplingStatisticSummaries` 管理事件記錄到 CloudTrail。如需詳細資訊，X-Ray 參閱使用 AWS CloudTrail.	2024年3月7日
已新增的功能	X-Ray 現在支持通過 OpenTelemetry 或符合 W3C 跟踪上下文規範的任何其他框架創建的跟踪 ID。如需詳細資訊，請參閱將追蹤資料傳送至 X-Ray。	2023 年 10 月 25 日
已新增的功能	您現在可以設定 Amazon SNS 作用中追蹤，讓您能夠在請求在傳送 Amazon SNS 主題時追蹤和分析這些請求。如需詳細資訊，請參閱 Amazon SNS 和 AWS X-Ray.	2023 年 2 月 8 日
更新了 Node.js 主題的 X-Ray SDK	已新增使用 AWS SDK for JavaScript V3 檢測用戶端的詳細資料。如需詳細資訊，請參閱使用適用於 Node.js 的 X-	2023 年 2 月 7 日

	Ray AWS SDK 追蹤 SDK 呼叫。	
更新 IAM 受管政策詳細資料	已新增 IAM 權限AWSXRayReadOnlyAccess ，以便跨帳戶觀察AWSXRayFullAccess 和AWSXrayCrossAccountSharingConfiguration 受管政策。如需詳細資訊，請參閱 X-Ray 的 IAM 受管政策。	2023 年 2 月 7 日
已新增的功能	AWS X-Ray 現在支援跨帳戶觀察能力，讓您能夠監控和疑難排解跨多個帳戶的應用程式，並進行疑難排解。 AWS 區域 如需詳細資訊，請參閱跨帳戶追蹤。	2022 年 11 月 27 日
已新增的功能	您現在可以檢視訊息生產者、Amazon SQS 佇列和取用者之間的連結追蹤，提供從事件導向應用程式傳送追蹤的連線檢視。如需詳細資訊，請參閱追蹤事件導向應用程式。	2022 年 11 月 20 日
更新 IAM 受管政策詳細資料	已新增 IAM 權限，可將資源政策列出至AWSXRayReadOnlyAccess 受管政策。如需詳細資訊，請參閱 X-Ray 的 IAM 受管政策。	2022 年 11 月 15 日
更新 IAM 主控台許可和受管政策詳細資料	X-Ray 主控台使用的 IAM 許可集已更新，以及AWSXRayReadOnlyAccess 受管政策的說明。如需詳細資訊，請參閱使用 X-Ray 主控台。	2022 年 11 月 11 日

為紅寶 AWS 石添加發行 OpenTelemetry 版	AWS 適用於 OpenTelemetry (ADOT) 的發行版提供一組開放原始碼 API、程式庫和代理程式，以收集分散式追蹤和指標。ADOT Ruby 可讓您針對 X-Ray 和其他追蹤後端的 Ruby 應用程式進行檢測。如需詳細資訊，請參閱 OpenTelemetry Ruby 的AWS 發行版。	2022 年 2 月 7 日
已新增的功能	您現在可以從 CloudWatch 主控台檢視追蹤並設定 X-Ray。如需詳細資訊，請參閱 X-Ray 主控台。	2022 年 1 月 24 日
綜合 CloudWatch 朗姆酒	使用 AWS X-Ray 和 CloudWatch RUM，您可以從應用程式的最終使用者開始，透過下游 AWS 受管理的服務來分析和偵錯要求路徑。如需詳細資訊，請參閱 CloudWatch RUM 和 AWS X-Ray.	2021 年 12 月 3 日
整合式 AWS 發行版 OpenTelemetry	適用於 OpenTelemetry (ADOT) 的發行 AWS 版提供一組開放原始碼 API、程式庫和代理程式，以收集分散式追蹤和指標。ADOT 可讓您檢測應用程式的 X-Ray 和其他追蹤後端。如需詳細資訊，請參閱檢測您的應用程式。	2021 年 9 月 23 日

已新增的功能	AWS X-Ray 現在與 Amazon 虛擬私有雲整合，讓 Amazon VPC 中的資源無需透過公用網際網路即可與 X-Ray 服務進行通訊。如需詳細資訊，請參閱AWS X-Ray 搭配 VPC 端點使用。	2021 年 5 月 20 日
已新增的功能	AWS X-Ray 現在與整合 AWS CloudFormation，讓您能夠佈建和設定 X-Ray 資源。如需詳細資訊，請參閱使用建立 X-Ray 資源 CloudFormation。	2021 年 5 月 6 日
已新增的功能	AWS X-Ray 現在與 Amazon 集成 EventBridge 以跟踪傳遞的事件 EventBridge。這為用戶提供了他們的系統的更完整的視圖。有關更多信息，請參閱 Amazon EventBridge 和 AWS X-Ray.	2021 年 3 月 2 日
將守護進程添加到 ECR	該守護程序現在可以從 Amazon ECR 下載。如需詳細資訊，請參閱下載協助程式。	2021 年 3 月 1 日
已新增的功能	AWS X-Ray 現在支援傳送給 Amazon 的洞察相關通知 EventBridge。這使您可以對使用的洞察採取自動操作 EventBridge。如需詳細資訊，請參閱見解通知。	2020 年 10 月 15 日
新增可下載守護進程	AWS X-Ray 引入了 ARM64 的支持守護進程。如需詳細資訊，請參閱AWS X-Ray 守護進程 ws	2020 年 10 月 1 日

已新增的功能	AWS X-Ray 現在支持與 Amazon CloudWatch Synthetics 的主動集成。這可讓您查看有關 Synthetics 料初期測試用戶端節點的詳細資料，例如回應時間和狀態。您也可以根據 Synthetics 測試用戶端節點中的資訊，在 Analytics 主控台中進行分析。如需詳細資訊，請參閱使用 X-Ray 對 CloudWatch 合成材料金絲雀除錯。	2020 年 9 月 24 日
已新增的功能	AWS X-Ray 現在支援 end-to-end 追蹤 AWS Step Functions . 您可以視覺化狀態機器的元件、識別效能瓶頸，以及疑難排解導致錯誤的要求。如需詳細資訊，請參閱AWS Step Functions 和 AWS X-Ray。	2020 年 9 月 14 日
已新增的功能	AWS X-Ray 引入見解，持續分析帳戶中的追蹤資料，以識別應用程式中的緊急問題。見解會記錄事件並追蹤事件影響直到解決為止。如需詳細資訊，請參閱在AWS X-Ray 主控台中使用深入解析	2020 年 9 月 3 日

已新增的功能	AWS X-Ray 引入 Java 自動檢測代理程式，可讓客戶收集追蹤資料，而不必修改現有的 Java 應用程式。您現在可以追蹤基於 Java Web 和 servlet 的應用程式，只需最少的組態變更，也不需要變更程式碼。如需詳細資訊，請參閱 Java 的 AWS X-Ray 自動檢測代理程式。	2020 年 9 月 3 日
已新增的功能	AWS X-Ray 已將新的「群組」頁面新增至 X-Ray 主控台，以協助簡化追蹤群組的建立和管理作業。如需詳細資訊，請參閱在 X-Ray 主控台中設定群組。	2020 年 8 月 24 日
已新增的功能	AWS X-Ray 現在可讓您將標籤新增至群組和取樣規則。您也可以根據標籤來控制群組和取樣規則的存取。如需詳細資訊，請參閱為 X-Ray 取樣規則和群組加上標籤和管理 X-Ray 群組的存取權限和根據標籤的取樣規則。	2020 年 8 月 24 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。