aws

使用者指南

Amazon CloudWatch

Amazon CloudWatch: 使用者指南

Table of Contents

什麼是 Amazon CloudWatch？ 1
存取 CloudWatch 1
相關 AWS 服務 1
如何 CloudWatch 工作 2
概念 3
命名空間 3
指標 3
維度 5
解析度 6
統計資料 7
單位 7
期間 7
聚合 8
百分位數 8
警示 9
帳單與成本 10
資源 10
開始設定 11
註冊一個 AWS 帳戶 11
建立具有管理權限的使用者 11
登錄到 Amazon 控 CloudWatch 制台 12
設定 AWS CLI 13
開始使用 14
查看預先建置的跨服務儀表板 19
移除服務以避免出現在跨服務儀表板中 20
查看單 AWS 一服務的預先建置儀表板 20
查看資源群組的預先建置儀表板 22
CloudWatch 帳單和成本 23
使用 CloudWatch Cost Explorer 分析成本和用量資料 23
視覺化和分析 CloudWatch 成本和使用情況資料 23
使用 s 和 Athena 分析 CloudWatch 成本和 AWS Cost and Usage Report用量資料 27
與 AWS Cost and Usage Report s 和 Athena 分析成本和使用情況資料 28
最佳化和減少成本的最佳實務 31
CloudWatch 度量 31

CloudWatch 警報 38
CloudWatch 日誌 40
儀表板 44
建立儀表板 45
CloudWatch 跨帳戶可觀察儀表板 46
跨帳戶跨區域儀表板 47
透過 AWS Management Console建立和使用的跨帳戶跨區域儀表板 47
以程式設計方式建立跨帳戶跨區域儀表板 48
使用儀表板變數建立彈性儀表板 51
儀表板變數的類型 51
教學課程：建立以函數名稱為變數的 Lambda 儀表板 52
教學課程：建立使用規則運算式模式在區域之間切換的儀表板 53
將變數複製至另一個儀表板 54
在 CloudWatch 儀表板上建立和使用小器具 55
新增或移除圖形 55
在 CloudWatch 儀表板上手動繪製指標 58
編輯圖形 59
將資源管理器小部件添加到 CloudWatch 儀表板 66
新增或移除線條小工具 68
新增或移除數字小工具 69
新增或移除量測計小工具 70
將自訂小器具新增至 CloudWatch 儀表板 72
新增或移除文字小工具 82
新增或移除警示小工具 83
新增或移除資料表小工具 84
連結和取消連結圖形 87
共用儀表板 88
共享儀表板所需的許可 89
授予與您共享儀表板之人員的許可 90
與特定使用者共享單一儀表板 91
公開共享單一儀表板 92
使用 SSO 共用帳戶中的所有 CloudWatch 儀表板 93
設定 CloudWatch 儀表板共用的 SSO 93
查看共享的儀表板數目 94
查看要共享哪些儀表板 95
停止共享一個或多個儀表板 95

檢閱共享儀表板許可並變更許可範圍 96
允許您共享的對象查看複合警示 97
允許您與其共享的對象查看日誌表小工具 98
允許您與其共享的對象查看自訂小工具 99
使用即時資料 101
檢視動畫儀表板 102
將儀表板新增至我的最愛清單 102
變更期間覆寫設定或重新整理間隔 103
變更時間範圍或時區格式 104
指標 107
基本監控和詳細監控 107
使用指標洞察查詢您的 CloudWatch 指標 109
建置查詢 110
查詢元件和語法 111
在 Metrics Insights 查詢上建立警示 120
使用 Metrics Insights 查詢搭配指標數學 123
使用自然語言產生和更新 CloudWatch 指標見解查詢 124
SQL 推斷 127
範例查詢 128
Metrics Insights 限制 136
Metrics Insights 字彙 137
對 Metrics Insights 進行疑難排解 137
使用指標瀏覽器依其標籤和屬性監控資源 138
CloudWatch 指標總管的代理程式組態 140
使用指標串流 140
設定指標串流 142
可供串流的統計數字 152
指標串流操作與維護 153
使用指標監控您的 CloudWatch 指標串流 154
CloudWatch 與 Firehose 之間的信任 155
指標串流輸出格式 156
故障診斷 184
檢視可用的指標 185
搜尋可用的指標 188
建立指標圖形 189
將指標圖形化 190

將兩個圖形合併為一個 195
使用動態標籤 196
修改圖形的時間範圍或時區格式 198
放大圖形 201
修改圖形的 Y 軸 202
從圖形的指標建立警示 203
使用異常偵測 205
異常偵測的運作方式 207
指標數學上的異常偵測 207
使用指標數學 208
將數學運算式新增至 CloudWatch 圖表 209
指標數學語法和函數 210
使用 IF 表達式 238
指標數學上的異常偵測 241
在圖形中使用搜尋運算式 242
搜尋表達式語法 242
搜尋表達式範例 248
使用搜尋表達式建立圖形 250
取得指標的統計資訊 253
CloudWatch 統計定義 253
取得特定資源的統計資料 256
跨資源彙總統計資料 261
依據 Auto Scaling 群組彙總統計資料 263
依 AMI 彙總統計資料 265
發佈 自訂指標 267
高解析度指標 267
使用維度 268
發佈單一資料點 269
發佈統計資料集 270
發佈零值 270
停止發佈指標 271
警示 272
指標警示狀態 273
評估警示 273
警示動作 274
Lambda 警示動作 275

設定警示如何處理遺失資料 279
資料遺失時評估警示狀態的方式 280
高解析度警示 284
數學運算式的警示 284
以百分位數為基礎的警示和低資料範例 284
CloudWatch 警報的常見功能 284
服務的警報建 AWS 議 285
尋找並建立建議的警示 286
建議的警示 287
針對指標的警示 367
建立以靜態閾值為基礎的警示 367
根據指標數學運算式建立警示 369
根據 Metrics Insights 查詢建立警示 372
根據連線的資料來源建立警示 372
根據異常偵測建立警示 375
修改異常偵測模型 378
刪除異常偵測模型 379
針對日誌的警示 379
根據日誌群組指標篩選條件建立警示 379
合併警示 381
建立複合警示 383
抑制複合警示動作 386
針對警示變更採取行動 393
在警示變更時通知使用者 394
警報事件和 EventBridge 399
管理警示 412
編輯或刪除 CloudWatch 鬧鐘 412
隱藏 Auto Scaling 鬧鐘 414
警示使用案例和範例 414
建立帳單警示 414
建立 CPU 使用量警示 418
建立負載平衡器延遲警示 420
建立儲存體輸送量警示 422
從資料庫建立 Performance Insights 計 AWS 數器指標的警示 424
建立警示以停止、終止、重新啟動或復原 EC2 執行個體 426
警報和標記 433

Application Signals 435
Application Signals 所需的許可 438
啟用和管理 Application Signals 的許可 438
正在運作的 Application Signals 443
啟用 Application Signals 446
Application Signals 支援的系統 446
OpenTelemetry 相容性考量 447
在 Amazon EKS 叢集上啟用 Application Signals 449
使用自訂設定在其他平台上啟用 Application Signals 458
疑難排解 Application Signals 安裝 475
設定 Application Signals 479
服務水準目標 (SLO) 483
SLO 概念 484
建立 SLO 486
檢視和分類 SLO 狀態 488
編輯現有 SLO 489
刪除 SLO 490
監控應用程式的運作狀態 490
使用「服務」頁面檢視您的服務 491
檢視詳細的服務資訊 494
使用服務對應檢視您的應用程式拓撲 506
範例：解決操作運作狀態問題 526
收集的標準應用程式指標 530
收集的維度與維度組合 530
使用綜合監控 534
必要的角色和許可 536
建立 Canary 550
群組 646
在本地測試金絲雀 647
對失敗的 Canary 進行故障診斷 667
Canary 指令碼的範本程式碼 675
Canary 和 X-Ray 追蹤 681
在 VPC 上執行 Canary 682
加密 Canary 成品 683
檢視 Canary 統計資料和詳細資訊 685
CloudWatch 加那利群島發布的指標 687

編輯或刪除 Canary 690
啟動、停止、刪除或更新多個 Canary 的執行階段 691
用 Amazon 監控金絲雀事件 EventBridge 692
CloudWatch 明顯地執行發射和 A/B 實驗 696
使用 Evidently 的 IAM 政策 698
建立專案、功能、啟動和實驗 699
管理功能、啟動和實驗 718
將程式碼新增至應用程式 723
專案資料儲存 725
Evidently 如何計算結果 727
在儀表板中檢視啟動結果 729
在儀表板中檢視實驗結果 730
如何 CloudWatch 明顯地收集和存儲數據 731
使用服務連結角色 732
CloudWatch 顯然配額 734
教學：使用 Evidently 範例應用程式進行 A/B 測試 735
使用 CloudWatch 朗姆酒 744
使用 RUM 的 IAM CloudWatch 政策 747
設定應用程式以使用 CloudWatch RUM 748
設定 R CloudWatch UM 網頁用戶端 757
區域化 758
使用頁面群組 759
指定自訂中繼資料 759
傳送自訂事件 765
檢視 R CloudWatch UM 儀表板 768
CloudWatch 您可以使用 CloudWatch RUM 收集的指標 770
CloudWatchRUM 的資料保護和資料隱私 779
CloudWatch RUM 網頁用戶端收集的資訊 780
管理使用 CloudWatch RUM 的應用程式 812
CloudWatch 朗姆酒配額 813
故障診斷 813
網路監控 815
使用網路監視器 815
支援地區 817
定價 818
元件 819

互聯網天氣圖 821
網路監視器的運作方式 822
使用案例 828
互聯網監控跨帳戶觀察 829
開始使用 829
使用 CLI 的範例 842
網路監視器儀表板 852
使用工具探索資料 860
建立 警示 878
EventBridge 整合 879
故障診斷錯誤 879
資料保護和資料隱私權 880
身分和存取權管理 881
配額 891
使用 Network Monitor 892
Network Monitor 主要功能 892
術語和要素 892
限制和要求 893
Network Monitor 如何運作 893
區域可用性 895
建立 Network Monitor 897
使用監視器和探查 901
Network Monitor 儀表板 909
配額 914
安全 914
身分識別和存取管理 916
定價 933
基礎設施監控 935
Container Insights 935
Container Insights 搭配 Amazon EKS 的增強可觀測性 936
支援平台 937
CloudWatch 代理容器映像 937
支援地區 937
設定 Container Insights 939
檢視 Container Insights 指標 993
Container Insights 收集的指標 997

效能日誌參考 1075
Container Insights Prometheus 指標監控 1109
與 Application Insights 整合 1232
在 Container Insights 中查看 Amazon ECS 生命週期事件 1232
針對容器洞見進行故障診斷 1234
建立您自己的 CloudWatch 代理碼頭形象 1237
在容器中部署其他 CloudWatch 代理程式功能 1237
Lambda Insights 1238
開始使用 Lambda Insights 1238
檢視您的 Lambda Insights 指標 1294
與 Application Insights 整合 1295
Lambda Insights 收集的指標 1295
故障診斷和已知問題 1298
範例遙測事件 1299
使用貢獻者洞察分析高基數資料 1301
建立 Contributor Insights 規則 1302
Contributor Insights 規則語法 1306
範例規則 1310
檢視 Contributor Insights 報告 1314
繪製規則產生的指標 1315
使用 Contributor Insights 內建規則 1318
利用應用程式洞察偵測常見的 CloudWatch 應用 1318
什麼是 Amazon CloudWatch 應用程序洞察？ 1319
Application Insights 的運作方式 1328
開始使用 1342
Application Insights 跨帳戶觀察 1372
使用元件組態 1373
使用 CloudFormation範本 1443
教學課程：設定 SAP ASE 的監控 1456
教學課程：設定 SAP HANA 的監控 1464
教學課程：設定 SAP 的監督 NetWeaver 1479
檢視和故障診斷 Application Insights 1495
支援的日誌和指標 1499
使用資源運作狀態檢視 1592
必要條件 1592
CloudWatch 跨帳戶可觀察性 1595

連結監控帳戶與來源帳戶 1597
必要的許可 1597
設定概觀 1601
步驟 1：設定監控帳戶 1602
步驟 2：(選擇性) 下載 AWS CloudFormation 範本或網址 1603
步驟 3：連結來源帳戶 1603
管理監控帳戶和來源帳戶 1607
將更多來源帳戶連結至現有監控帳戶 1607
移除監控帳戶與來源帳戶之間的連結 1608
檢視監控帳戶的相關資訊 1609
從其他資料來源中查詢指標 1610
管理資料來源的存取 1611
使用精靈連線至預先建立的資料來源 1611
Amazon Managed Service for Prometheus 1613
Amazon OpenSearch 服務 1613
Amazon RDS for PostgreSQL 和 Amazon RDS for MySQL 1614
Amazon S3 CSV 檔案 1615
Microsoft Azure Monitor 1616
Prometheus 1617
可用更新的通知 1618
建立資料來源的自訂連接器 1618
使用範本 1619
從頭開始建立自訂資料來源 1620
使用自訂資料來源 1625
如何將引數傳遞給 Lambda 函數 1625
刪除資料來源的連接器 1626
使用 CloudWatch 代理程式收集指標、記錄和追蹤 1628
安裝 CloudWatch 代理程式 1630
使用命令列安裝 CloudWatch 代理程式 1631
使用系統管理員安裝 CloudWatch 代 Systems Manager 程式 1651
使用在新執行個體上安裝 CloudWatch代理程式 AWS CloudFormation 1669
CloudWatch 代理程式認證偏 1675
驗證代 CloudWatch 理程式套件的簽章 1676
建立 CloudWatch 代理程式組態檔 1685
使用精靈建立 CloudWatch 代理程式組態檔 1686
手動建立或編輯 CloudWatch 代理程式組態檔 1692

使用 Amazon CloudWatch 可觀測 EKS 附加元件安裝 CloudWatch 代理程式 1778
選項 1：在工作節點上使用 IAM 許可進行安裝 1779
選項 2：使用 IAM 服務帳戶角色進行安裝 1781
(選用) 額外組態 1782
故障診斷 1785
CloudWatch 代理程式收集的測量結果 1786
CloudWatch 代理程式在 Windows 伺服器執行處理上收集的測量 1787
CloudWatch 代理程式在 Linux 和 macOS 執行個體上收集的指標 1787
記憶體指標定義 1800
CloudWatch代理程式的常見案例 1802
以不同的使用者身分執行 CloudWatch 代理程式 1803
CloudWatch 代理程式處理稀疏記錄檔的方式 1805
將自訂維度新增至 CloudWatch 代理程式收集的指標 1805
多個 CloudWatch 代理程式組態檔 1806
彙總或彙總代理程式收集的 CloudWatch 測量結果 1808
使用 CloudWatch 代理程式收集高解析度量 1809
將指標、日誌和追蹤傳送到不同帳戶 1810
統一 CloudWatch 代理程式與舊版記錄代理程式之間的時間戳 CloudWatch 記差 1812
疑難排解 CloudWatch 代理 1813
CloudWatch 代理程式命令列參數 1813
使用執行命令安裝 CloudWatch 代理程式失敗 1813
CloudWatch 代理程式無法啟動 1814
確認 CloudWatch 代理程式正在執行 1814
CloudWatch 代理程式無法啟動，錯誤提到 Amazon EC2 區域 1815
CloudWatch 代理程式無法在 Windows 伺服器上啟動 1815
指標在哪裡？ 1816
CloudWatch 代理程式需要很長時間才能在容器中執行，或記錄躍點限制錯誤 1816
我已更新代理程式設定，但在 CloudWatch 主控台中看不到新的指標或記錄檔 1817
CloudWatch 代理程式檔案和位置 1817
尋找 CloudWatch 代理程式版本的資訊 1819
CloudWatch代理程式產生的記錄檔 1820
停止並重新啟動 CloudWatch 代理程式 1821
在日誌中內嵌指標 1822
使用內嵌指標格式發布日誌 1822
使用用戶端程式庫 1823
規格：內嵌指標格式 1824

使用 PutLogEvents API 傳送手動建立的內嵌指標格式記錄 1832
使用 CloudWatch 代理程式傳送內嵌的量度格式記錄 1834
將嵌入式度量格式與 AWS 發行版搭配使用 OpenTelemetry 1841
在主控台中檢視您的指標和日誌 1841
在以內嵌指標格式建立的指標上設定警示 1842
發佈指 CloudWatch 標的服務 1843
AWS 使用量度 1859
視覺化您的服務配額和設定警示 1859
AWS API 使用量指標 1861
CloudWatch 使用量度 1869
CloudWatch 教程 1871
案例：監控預估費用 1871
步驟 1：啟用帳單提醒 1872
步驟 2：建立帳單警示 1872
步驟 3：檢查警示狀態 1874
步驟 4：編輯帳單警示 1874
步驟 5：刪除帳單警示 1875
案例：發佈指標 1875
步驟 1：定義資料組態 1875
步驟 2：將量度新增至 CloudWatch 1876
步驟 3：從中獲取統計信息 CloudWatch 1877
步驟 4：使用主控台檢視圖形 1878
使用 AWS 軟體開發套件 1879
程式碼範例 1881
動作 1887
DeleteAlarms 1887
DeleteAnomalyDetector 1896
DeleteDashboards 1899
DescribeAlarmHistory 1902
DescribeAlarms 1906
DescribeAlarmsForMetric 1912
DescribeAnomalyDetectors 1925
DisableAlarmActions 1928
EnableAlarmActions 1939
GetDashboard 1949
GetMetricData 1950

GetMetricStatistics 1955
GetMetricWidgetImage 1964
ListDashboards 1969
ListMetrics 1971
PutAnomalyDetector 1986
PutDashboard 1989
PutMetricAlarm 1995
PutMetricData 2009
案例 2023
開始使用警示 2023
開始使用指標、儀表板和警示 2026
管理指標和警示 2100
跨服務範例 2108
監 DynamoDB 效能 2109
安全 2110
資料保護 2110
傳輸中加密 2111
身分與存取管理 2111
物件 2112
使用身分驗證 2112
使用政策管理存取權 2115
Amazon 如何與 IAM 合 CloudWatch 作 2117
身分型政策範例 2123
故障診斷 2127
CloudWatch 儀表板權限更新 2129
AWS 的管理 (預先定義) 策略 CloudWatch 2130
客戶受管政策範例 2155
政策更新 2157
使用條件鍵限制對 CloudWatch 命名空間的訪問 2171
使用條件索引鍵限制 Contributor Insights 使用者存取日誌群組 2172
使用條件索引鍵限制警示動作 2174
使用服務連結角色 2175
針 CloudWatch 對 RUM 使用服務連結角色 2185
針對 Application Insights 使用服務連結角色 2190
AWS 應用程式深入解析的受管 2201
Amazon CloudWatch 許可參考 2209

法規遵循驗證 2223
恢復能力 2224
基礎架構安全 2224
網路隔離 2224
AWS Security Hub 2225
介面 VPC 端點 2225
CloudWatch 2225
CloudWatch Synthetics 2227
Synthetics Canary 的安全考量 2229
使用安全連線 2229
Canary 命名考量 2230
Canary 程式碼中的秘密和敏感資訊 2230
許可考量 2230
堆疊追蹤和異常情況訊息 2231
以較小範圍限制您的 IAM 角色 2231
敏感資料修訂 2231
使用 AWS CloudTrail記錄 API 呼叫 2233
CloudWatch 中的資訊 CloudTrail 2234
範例： CloudWatch 記錄檔項目 2235
CloudWatch 互聯網監控 CloudTrail 2237
範例： CloudWatch 網際網路監視器記錄檔項目 2238
CloudWatch Synthetics 資訊 CloudTrail 2240
範例： CloudWatch Synthetics 件記錄檔項目 2240
標記您的 CloudWatch 資源 2245
支援的資源 CloudWatch 2245
管理標籤 2246
標籤命名和使用慣例 2246
Grafana 整合 2247
跨帳戶跨 CloudWatch 區域主控台 2248
啟用跨帳戶跨區域功能 2248
(選擇性) 整合 AWS Organizations 2252
故障診斷 2252
使用跨帳戶後停用和清除 2253
Service Quotas 2255
文件歷史紀錄 2262
..... mmcclxxxviii

什麼是 Amazon CloudWatch？

Amazon CloudWatch 監控您的 Amazon Web Services（AWS）資源和您實時運行 AWS 的應用程序。您可以用 CloudWatch 來收集和追蹤指標，這些指標是您可以針對資源和應用程式測量的變數。

CloudWatch 首頁會自動顯示您使用之每項 AWS 服務的測量結果。您還可以建立自訂儀表板，以顯示自訂應用程式的相關指標，以及顯示您選擇的自訂指標集合。

您可以建立警示來監控指標，並於超過閾值時傳送通知，或自動變更您所監控的資源。例如，您可以監控 CPU 用量和 Amazon EC2 執行個體的磁碟讀取和寫入，然後使用該資料，判斷您是否應該啟動其他的執行個體來處理增加的負載。您也可以使用此資料來停止較不常使用的執行個體，以節省成本。

有了 CloudWatch，您就可以掌握整個系統的資源使用率、應用程式效能和營運狀態。

存取 CloudWatch

您可以使 CloudWatch 用下列任何一種方法存取：

- Amazon CloudWatch 控制台 — https://console.aws.amazon.com/cloudwatch/
- AWS CLI — 如需詳細資訊，請參閱AWS Command Line Interface 使用者指南 AWS Command Line Interface中的〈進行設定〉。
- CloudWatch API — 如需詳細資訊，請參閱 Amazon CloudWatch API 參考資料。
- AWS 開發套件 — 如需詳細資訊，請參閱 Amazon Web Services 的工具。

相關 AWS 服務

以下服務與 Amazon 一起使用 CloudWatch：

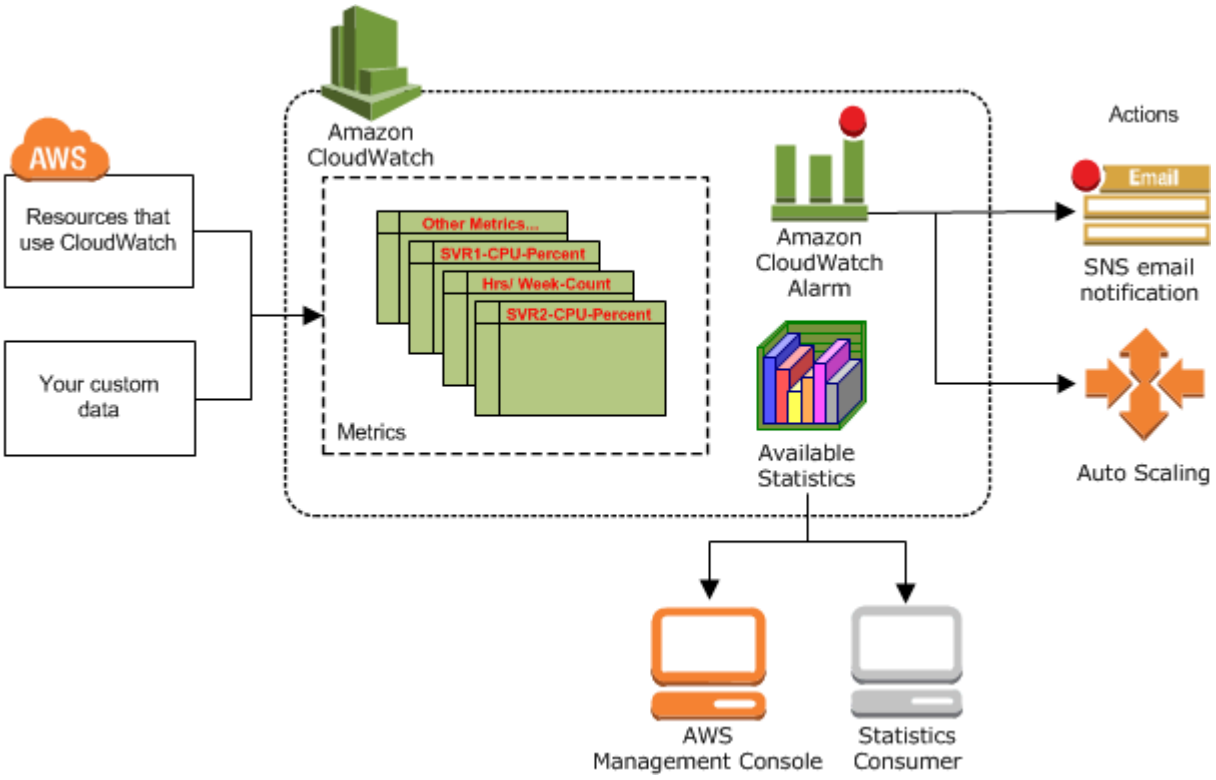
- Amazon Simple Notification Service (Amazon SNS) 會協調和管理消息傳遞或傳送到訂閱端點或客戶端。當達到警示閾值時，您可以使用 Amazon SNS CloudWatch 來傳送訊息。如需詳細資訊，請參閱 設定 Amazon SNS 通知。
- Amazon EC2 Auto Scaling 讓您能夠根據使用者定義的政策、運作狀態檢查和排程，自動啟動或終止 Amazon EC2 執行個體。您可以將 CloudWatch 警示與 Amazon EC2 Auto Scaling 搭配使用，根據需求擴展 EC2 執行個體。如需詳細資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的動態擴展。
- AWS CloudTrail可讓您監控帳戶對 Amazon CloudWatch API 進行的呼叫，包括 AWS Management Console AWS CLI、和其他服務的呼叫。開啟 CloudTrail 記錄時， CloudWatch 將日誌檔寫入您在

設定時指定的 Amazon S3 儲存貯體 CloudTrail。如需詳細資訊，請參閱 記錄 Amazon CloudWatch API 呼叫 AWS CloudTrail。

- AWS Identity and Access Management (IAM) 是一種 Web 服務，可協助您安全地控制使用者對 AWS 資源的存取。使用 IAM 控制誰可以使用您的 AWS 資源 (身分驗證)，以及他們可以透過何種方式使用哪些資源 (授權)。如需詳細資訊，請參閱 Amazon 的身份和訪問管理 CloudWatch。

Amazon 如何 CloudWatch 工作

Amazon 基本上 CloudWatch 是一個指標存儲庫。 AWS 服務 (例如 Amazon EC2) 將指標放入儲存庫，您可以根據這些指標擷取統計資料。如果您將自己的自訂指標放至儲存庫，您也可以對這些指標擷取統計資料。



您可以使用指標來計算統計資料，然後在CloudWatch 主控台中以圖形方式呈現資料。如需產生指標並將其傳送至其他 AWS 資源的詳細資訊 CloudWatch，請參閱AWS 發佈指 CloudWatch標的服務。

您可以設定警示動作，來在滿足特定條件時停止、開始或終止 Amazon EC2 執行個體。此外，您可以建立警示，代您啟動 Amazon EC2 Auto Scaling 和 Amazon Simple Notification Service (Amazon SNS) 動作。如需建立 CloudWatch警示的詳細資訊，請參閱警示。

AWS 雲計算資源存放在高可用性的數據中心設施中。為了提供額外的可擴展性和可靠性，每個資料中心設施位於特定的地理區域 (稱為區域)。每個區域設計為與其他區域完全隔開，以達到最大可能的

故障隔離和穩定性。指標會分別儲存在「區域」中，但您可以使用 CloudWatch 跨區域功能彙總來自不同區域的統計資料。如需詳細資訊，請參閱《Amazon Web Services 一般參考》中的「跨帳戶跨 CloudWatch 區域主控台」和「區域與端點」。

Amazon CloudWatch 概念

以下術語和概念是您對 Amazon 的理解和使用的核心 CloudWatch：

- 命名空間
- 指標
- 維度
- 解析度
- 統計資料
- 百分位數
- 警示

如需 CloudWatch 指標、警示、API 要求和警示電子郵件通知的服務配額的相關資訊，請參閱CloudWatch 服務配額。

命名空間

命名空間是 CloudWatch 指標的容器。指標在不同的命名空間中相互隔離，以便不同應用程式的指標不會被錯誤地彙總到相同的統計資訊中。

沒有預設的命名空間。您必須為發佈到的每個資料點指定命名空間 CloudWatch。您可以在建立指標名稱時指定命名空間。這些名稱必須包含有效的 ASCII 字元，且等於或少於 255 個字元。可能的字元包括：英數字元 (0-9A-ZA-Z)、句點 (.)、連字號 (-)、底線 (_)、正斜線 (/)、雜湊 (#)、冒號 (:) 和空格字元。命名空間必須包含至少一個非空格字元。

命 AWS 名空間通常使用下列命名慣例：`AWS/service`。例如，Amazon EC2 使用 `AWS/EC2` 命名空間。如需命 AWS 名空間的清單，請參閱AWS 發佈指 CloudWatch標的服務。

指標

度量標準是中的基本概念 CloudWatch。量度代表發佈至 CloudWatch的一組時間順序的資料點。您可以將指標視為要監控的變數，且資料點代表該變數隨著時間的值。例如，特定 EC2 執行個體的 CPU

用量是 Amazon EC2 提供的指標。資料點本身可以來自任何您從其中收集資料的應用程式或企業活動。

依預設，許多 AWS 服務為資源 (例如 Amazon EC2 執行個體、Amazon EBS 磁碟區和 Amazon RDS 資料庫執行個體) 免費提供指標。您也可以啟用詳細監控一些資源，例如您的 Amazon EC2 執行個體，或發佈自己的應用程式指標。對於自訂指標，您可以任何順序以及您所選的任何費率新增資料點。您可以一組時間序列資料的形式來擷取這些資料點的相關統計資料。

指標只存在於已在其中建立之的區域中。指標無法刪除，但他們會在沒有將資料匯入其中的 15 個月後到期。久於 15 個月的會輪流到期，隨著新資料點加入，久於 15 個月的資料會被刪除。

指標是由名稱、命名空間和零或多個維度做唯一的定義。指標中的每個資料點都有時間戳記和 (選用) 的測量單位。您可以從 CloudWatch 任何測量結果擷取統計資料。

如需更多詳細資訊，請參閱「檢視可用的指標」和「發佈 自訂指標」。

時間戳記

每個指標資料點都必須和時間戳記建立關聯。時間戳記最多可達過去兩週和高達未來兩小時。如果您未提供時間戳記，則 CloudWatch 會根據接收到資料點的時間為您建立時間戳記。

時間戳記是 `dateTime` 物件，內含完整的日期加上小時、分鐘和秒 (例如，2016-10-31T23:59:59 Z)。如需更多詳細資訊，請參閱 dateTime。雖然沒有要求，我們建議您使用國際標準時間 (UTC)。當您從中檢索統計信息時 CloudWatch，所有時間都是 UTC。

CloudWatch 警報會根據 UTC 的目前時間來檢查指標。使用目前 UTC 時間以外 CloudWatch的時間戳記傳送至的自訂量度可能會導致警示顯示「資料不足」狀態或導致警示延遲。

指標保留

CloudWatch 如下所示保留測量結果資料：

- 含少於 60 秒期間的資料點可供使用 3 小時。這些資料點為高解析自訂指標。
- 含少於 60 秒期間 (1 分鐘) 的資料點可供使用 15 天。
- 含少於 300 秒期間 (5 分鐘) 的資料點可供使用 63 天
- 含少於 3600 秒期間 (1 小時) 的資料點可供使用 455 天 (15 個月)。

原先使用較短期間發佈的資料點會一起彙總以供長期儲存。例如，如果您使用 1 分鐘的期間收集資料，資料會以 1 分鐘的解析度維持可供使用 15 天。在 15 天候，此資料仍可供使用，但會進行彙總並

以僅 5 分鐘的解析度可供擷取。在 63 天候，此資料會進一步進行彙總並以僅 1 小時的解析度可供使用。

> **Note**
>
> 過去兩週內沒有任何新資料點的指標不會顯示在主控台中。當您在主控台的 All metrics (所有指標) 索引標籤的搜尋方塊中輸入公制名稱或維度名稱時，它們也不會顯示，而且在 list-metrics 命令的結果中不會傳回這些名稱。擷取這些度量的最佳方式是使用中的get-metric-data或get-metric-statistics命令 AWS CLI。

維度

維度是一組名稱值對，是指標身分的一部分。您可以在一個指標中指派最多 30 個維度。

每個指標都有特定的特性 (會描述該特性)，您可以將這些維度視為這些特性的類別。維度可協助您設計統計資料計劃的結構。由於維度是指標唯一識別碼的一部分，當您將唯一名稱/值組新增至其中一個指標時，您建立的是該指標的新變化。

AWS 傳送資料以 CloudWatch 將維度附加至每個量度的服務。您可以使用維度來篩選 CloudWatch 傳回的結果。例如，您可以獲得特定 EC2 執行個體的統計資料，方法是在搜尋指標時指定 `InstanceId` 維度。

對於某些 AWS 服務 (例如 Amazon EC2) 產生的指標， CloudWatch 可以跨維度彙總資料。例如，如果您在`AWS/EC2`命名空間中搜尋測量結果，但未指定任何維度，則會 CloudWatch 彙總指定測量結果的所有資料，以建立您要求的統計資料。 CloudWatch 不會針對您的自訂指標跨維度彙總。

維度組合

CloudWatch 即使量度具有相同的量度名稱，也會將每個唯一維度組合視為個別量度。您可以使用具體發佈的維度組合來只擷取統計資料。當您擷取統計資料，請指定用於命名空間的相同值、指標名稱，以及建立指標時使用的維度參數。您也可以指定用於 CloudWatch 彙總的開始和結束時間。

例如，假設您發佈在命名DataCenterMetric 空間 ServerStats 中命名的四個具有下列屬性的不同度量：

```
Dimensions: Server=Prod, Domain=Frankfurt, Unit: Count, Timestamp:
 2016-10-31T12:30:00Z, Value: 105
Dimensions: Server=Beta, Domain=Frankfurt, Unit: Count, Timestamp:
 2016-10-31T12:31:00Z, Value: 115
Dimensions: Server=Prod, Domain=Rio,       Unit: Count, Timestamp:
 2016-10-31T12:32:00Z, Value: 95
```

```
Dimensions: Server=Beta, Domain=Rio,       Unit: Count, Timestamp:
 2016-10-31T12:33:00Z, Value: 97
```

如果您僅發佈這四個指標，您可以為這些維度組合擷取統計資料：

- `Server=Prod,Domain=Frankfurt`
- `Server=Prod,Domain=Rio`
- `Server=Beta,Domain=Frankfurt`
- `Server=Beta,Domain=Rio`

您無法擷取下列維度的統計資料，或是若未指定維度，也無法擷取統計資料。(例外狀況是使用指標數學 SEARCH 函數，這可擷取多個指標的統計資料。如需詳細資訊，請參閱「在圖形中使用搜尋運算式」。)

- `Server=Prod`
- `Server=Beta`
- `Domain=Frankfurt`
- `Domain=Rio`

解析度

每個指標皆為下列其中一種：

- 標準解析度，包含 1 分鐘精細度的資料
- 高解析度，包含 1 秒鐘精細度的資料

依預設， AWS 服務產生的度量為標準解析度。當您發佈自訂指標時，可將它定義為標準解析度或高解析度。當您發佈高解析度量時，請以 1 秒的解析度 CloudWatch 儲存該指標，而且您可以讀取和擷取它，時間為 1 秒、5 秒、10 秒、30 秒或 60 秒的任意倍數。

高解析度的指標可讓您以高於分鐘層級的精細度，更即時地深入了解您應用程式的活動。請注意，對自訂指標進行的每個 `PutMetricData` 呼叫皆需付費，因此經常對高解析度指標呼叫 `PutMetricData` 將導致更高的費用。如需有關 CloudWatch 定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

如果您在高解析度指標設定警示，您可以指定期間為 10 秒或 30 秒的高解析度警示，或者設定期間為 60 秒的任何倍數的定期警示。期間為 10 或 30 秒的高解析度警示將產生更高的費用。

統計資料

統計資料是指定期間內的測量結果資料彙總。 CloudWatch 根據您的自訂資料提供或其他 AWS 服務提供給的指標資料點來提供統計資料 CloudWatch。彙總會使用在您指定期間的命名空間、公制名稱、維度和量測資料點單位來進行。

如需支援之統計資料的詳細定義 CloudWatch，請參閱CloudWatch 統計定義。

單位

每個統計單位有量測單位。範例單位包括 `Bytes`、`Seconds`、`Count` 和 `Percent`。如需 CloudWatch 支援單位的完整清單，請參閱 Amazon CloudWatch API 參考中的MetricDatum資料類型。

您可以在建立自訂指標時指定單位。如果您未指定單位，則 CloudWatch 使用None作為單位。單位可對您的資料提供概念性意義。雖然在內部對單元沒有 CloudWatch 重要性，但其他應用程序可以基於該單元導出語義信息。

指定量測單位的指標資料點會單獨彙總。當您取得統計資料而未指定單位時，會將同一單位的所有資料點 CloudWatch 彙總在一起。如果您有兩個含不同單位的指標，將會傳回兩個不同的資料串流，每個串流適用於每個單位。

期間

週期是與特定 Amazon CloudWatch 統計數據相關聯的時間長度。每個統計資料代表在特定時間段內收集的指標資料彙集。期間會以秒數定義，有效的期間值是 1、5、10、30 或任何多個 60。例如，若要指定六分鐘的期間，請使用 360 做為期間值。您可以透過變化不同的期間長度來調整資料彙總的方式。週期的預設值為 60 秒。週期可短至一秒鐘，如果週期大於預設值 60 秒，則必須為 60 的倍數。

只有您使用 1 秒儲存解析度定義的自訂指標支援子分鐘的期間。即使設定為低於 60 期間的選擇在主控台中一律可供使用，您應選擇一個期間以符合指標存放的方式。如需有關支援子分鐘期間指標的更多資訊，請參閱 高解析度指標。

當您擷取統計資料時，您可以指定期間、開始時間和結束時間。這些參數會判斷與統計資料相關的整體時間長度。開始時間和結束時間的預設值會取得最後一個小時的統計資料。您為開始時間和結束時間指定的值會決定 CloudWatch 傳回多少個週期。例如，使用期間、開始時間和結束時間的預設值擷取統計資料，會傳回前一小時每分鐘的一組彙總統計資料。如果您偏好在十分鐘區塊內彙總的統計資料，指定期間為 600。對於在整個小時彙總的統計資料、指定期間為 3600。

當統計資料彙總一段時間後，它們都會標記對應於該期間之起始時間的時間。例如，從下午 7:00 至下午 8:00 的資料彙總標記為下午 7:00。此外，晚上 7:00 到晚上 8:00 之間彙總的資料會在晚上 7:00 開始顯示，然後該彙總資料的值可能會隨著在此期間 CloudWatch 收集更多樣本而改變。

週期對於 CloudWatch 警報也很重要。當您建立警示來監視特定測量結果時，您會 CloudWatch 要求比較該測量結果與您指定的臨界值。您可以廣泛控制如何 CloudWatch 進行比較。您不但可以指定比較進行的期間，但您也可以指定在結尾時使用的評估期間數量。例如，如果您指定三個評估期間，則 CloudWatch 會比較由三個資料點組成的視窗。 CloudWatch 僅在最舊的資料點違規且其他資料點違規或遺失時才會通知您。

聚合

Amazon 會根據您在擷取統計資料時指定的期間長度 CloudWatch 彙總統計資料。您可以使用相同或類似的時間戳記發佈任意數量的資料點。 CloudWatch 根據指定的週期長度彙總它們。CloudWatch 不會跨區域自動彙總資料，但您可以使用量度數學彙總來自不同區域的量度。

您可以發佈量度的資料點，這些量度不僅共用相同的時間戳記，還可以共用相同的命名空間和維度。CloudWatch 傳回這些資料點的彙總統計資料。您也可以發佈適用於相同或不同指標的多個資料點，內含任何時間戳記。

對於大型資料集，您可以插入稱為數據集的預先彙總資料集。使用統計集，您可以提供 CloudWatch 最小值、最大值、總和SampleCount ，以及數個資料點。當您需要在一分鐘多次收集資料就會常常使用此數據集。例如，假設您有適用於網頁的請求延遲指標。它無法發佈含每個網頁命中的資料。我們建議您收集該網頁所有點擊的延遲，每分鐘彙總一次，然後將該統計資料集傳送至CloudWatch。

Amazon CloudWatch 不區分指標的來源。如果您發佈的指標具有相同命名空間和來自不同來源的維度，則 CloudWatch 會將其視為單一量度。這對於分散式、擴展系統中的服務指標可能很有用。例如，Web 伺服器應用程式中的所有主機都可以發佈相同的指標，代表它們正在處理的要求延遲。CloudWatch將這些值視為單一量度，可讓您取得應用程式中所有要求之最小值、最大值、平均值和總和的統計資料。

百分位數

百分位數會指出資料集中相關準備好的值。例如，第 95 個百分位數表示 95% 的資料是低於此值且 5% 資料高於這個值。百分位數協助您更加了解您指標資料的分佈。

百分位數通常用於隔離異常。在標準分佈中，95% 的資料會在離平均值的兩個標準偏差，而 99.7% 的資料是離平均值的三個標準偏差。超出三個標準偏差的任何資料，通常視為異常，因為它與平均值的差異非常大。例如，假設您監控的是 EC2 執行個體的 CPU 使用率，以確保您的客戶獲得良好的體驗。如果您監控的是平均值，這可以隱藏異常。如果您監控最大值，單一異常可以影響結果。您可以使用百分位數，監控 CPU 使用率的第 95 個百分位數，以查看具異常負載的執行個體。

某些 CloudWatch 量度支援百分位數作為統計資料。對於這些指標，您可以使用百分位數來監視系統和應用程式，就像使用其他 CloudWatch 統計資料 (平均值、最小值、最大值和總和) 一樣。例如，

當您建立警示時，您可以使用百分位數做為統計函數。您可以指定百分位數，最多使用十位小數 (例如，p95.0123456789)。

只要您發佈自訂指標的原始、未彙整的資料點，百分比統計資料便可用於自訂指標。當任何指標值為負數時，百分位數統計資料不適用於指標。

CloudWatch 需要原始資料點來計算百分位數。如果您使用統計資料集來發佈資料，只有在以下條件之一為 true 時，您才能擷取此資料的百分位數統計資料：

- 統計資料集的 SampleCount 值為 1，而最小值、最大值和總和都是相等的。
- 最小值和最大值相等，並且總和等於最小乘以 SampleCount。

下列 AWS 服務包含支援百分位數統計資料的度量。

- API Gateway
- Application Load Balancer
- Amazon EC2
- Elastic Load Balancing
- Kinesis
- Amazon RDS

CloudWatch 還支持修剪均值和其他性能統計信息，這些統計信息可以與百分位數類似。如需詳細資訊，請參閱 CloudWatch 統計定義。

警示

您可以使用警示，以代表您自動啟動動作。警示會監看指定時段內的單一指標，並根據隨著時間與閾值相對的指標值來執行一或多個指定動作。此動作是傳送到 Amazon SNS 主題或 Auto Scaling 政策的通知。您也可以將警示新增至儀表板。

警示只會呼叫持續狀態變更的動作。 CloudWatch 警報不會僅僅因為它們處於特定狀態而叫用動作。狀態必須已變更，且在指定的期間數內維持此狀態。

在建立警示時，選取大於或等於指標的解析度的警示監控期間。例如，Amazon EC2 的基本監控會每 5 分鐘為您的執行個體提供指標。當對基本監控指標設定警示，選取至少 300 秒 (5 分鐘) 的期間。Amazon EC2 的詳細監控會以 1 分鐘的解析度為您的執行個體提供指標。當對詳細監控指標設定警示，選取至少 60 秒 (1 分鐘) 的期間。

如果您在高解析度指標設定警示，您可以指定期間為 10 秒或 30 秒的高解析度警示，或者設定期間為 60 秒的任何倍數的定期警示。高解析度警示費用更高。如需高解析度指標的詳細資訊，請參閱 發佈 自訂指標。

如需更多詳細資訊，請參閱「 使用 Amazon CloudWatch 警報」和「從圖形的指標建立警示」。

帳單與成本

如需有關 CloudWatch 定價的完整資訊，請參閱 Amazon CloudWatch 定價。

如需可協助您分析帳單及可能最佳化和降低成本的資訊，請參閱 CloudWatch 帳單和成本。

Amazon CloudWatch 資源

以下相關資源可協助您使用此服務。

資源	描述
Amazon CloudWatch 問題	此「常見問答集」涵蓋開發人員針對此產品最常詢問的問題。
AWS 開發者中心	尋找文件、程式碼範例、版本說明及其他資訊的中心起點，協助您使用 AWS.
AWS Management Console	該控制台允許您執行 Amazon 的大多數功能 CloudWatch 和各種其他 AWS 產品，而無需進行編程。
Amazon CloudWatch 論壇	以社群為基礎的論壇，讓開發人員討論與 Amazon CloudWatch 相關的技術問題。
AWS Support	建立和管理 AWS Support 案例的中心。同時也包含其他實用資源的連結，例如論壇、技術常見問答集、服務健康狀態和 AWS Trusted Advisor。
Amazon CloudWatch 產品信息	有關 Amazon 信息的主要網頁 CloudWatch。
聯絡我們	有關帳 AWS 單，帳戶，事件，濫用等查詢的中心聯繫窗口。

開始設定

要使用 Amazon， CloudWatch 您需要一個 AWS 帳戶。您的 AWS 帳戶可讓您使用服務 (例如 Amazon EC2) 產生可在 CloudWatch 主控台 (point-and-clickWeb 型介面) 中檢視的指標。此外，您還可以安裝和設定 AWS 命令列介面 (CLI)。

註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊一個時 AWS 帳戶，將創建AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

AWS 註冊過程完成後，會向您發送確認電子郵件。您可以隨時登錄 https://aws.amazon.com/ 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立具有管理權限的使用者

註冊後，請保護您的 AWS 帳戶 AWS 帳戶根使用者 AWS IAM Identity Center、啟用和建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

保護您的 AWS 帳戶根使用者

1. 選擇 Root 使用者並輸入您的 AWS 帳戶 電子郵件地址，以帳戶擁有者身分登入。AWS Management Console在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入 使用者指南中的以根使用者身分登入。
2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶 根使用者啟用虛擬 MFA 裝置 (主控台)。

建立具有管理權限的使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。

2. 在 IAM 身分中心中，將管理存取權授予使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。

以具有管理權限的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者簽署，請使用建立 IAM Identity Center 使用者時傳送至您電子郵件地址的簽署 URL。

  如需使用 IAM 身分中心使用者登入的說明，請參閱使用AWS 登入 者指南中的登入 AWS 存取入口網站。

指派存取權給其他使用者

1. 在 IAM 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「建立權限集」。

2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「新增群組」。

登錄到 Amazon 控 CloudWatch 制台

登錄到 Amazon 控 CloudWatch 制台

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 如有必要，請使用導覽列將「地區」變更為擁有 AWS 資源的「地區」。
3. 即使這是您第一次使用 CloudWatch 控制台，您的指標也可能已經報告指標，因為您使用的 AWS 產品會自動將指標免 CloudWatch 費推送到 Amazon。其他服務則會要求您啟用指標。

   如果您沒有任何警示，Your Alarms (您的警示) 區段將會有 Create Alarm (建立警示) 按鈕。

設定 AWS CLI

您可以使用 AWS CLI 或 Amazon CloudWatch CLI 來執行 CloudWatch 命令。請注意，會 AWS CLI 取代 CloudWatch CLI；我們只在 AWS CLI. CloudWatch

若要取得有關如何安裝和規劃的資訊 AWS CLI，請參閱《使用指南》中的〈AWS Command Line Interface 使用指 AWS 令行介面進行設置〉。

如需如何安裝和設定 Amazon CloudWatch CLI 的詳細資訊，請參閱 Amazon CloudWatch CLI 參考中的設定命令列界面。

開始使用 Amazon CloudWatch

[請在以下位置開啟 CloudWatch 主控台。](https://console.aws.amazon.com/cloudwatch/) https://console.aws.amazon.com/cloudwatch/

這時系統顯示 CloudWatch 概述首頁。

概觀會顯示下列項目，並自動重新整理。

- 依 AWS 服務的警示會顯示您在帳戶中使用的 AWS 服務清單，以及這些服務中的警示狀態。在此旁邊，顯示您帳戶中的兩個或四個警報。數量取決於您使用的 AWS 服務數量。顯示的警示是處於 ALARM 狀態或最近變更狀態的警示。

  這些上方區域可協助您快速評估 AWS 服務的健康狀態，方法是查看每項服務中的警示，以及最近變更狀態的警示。這可協助您監控和快速診斷問題。
- 這些區域的下方是「預設儀表板」(若存在的話)。預設儀表板是您已建立並命名為 CloudWatch-Default 的自訂儀表板窗格。這是一種方便的方式，可讓您將自己自訂服務或應用程式的指標新增至概觀頁面，或從您最想監視的 AWS 服務提供其他關鍵指標。

> **Note**
>
> CloudWatch 首頁上的自動儀表板僅顯示來自當前帳戶的信息，即使該帳戶是針對 CloudWatch 跨帳戶觀察設置的監視帳戶也是如此。如需有關建立跨帳戶儀表板的資訊，請參閱 CloudWatch 跨帳戶可觀察儀表板。

從此概觀中，您可以查看來自多個服務指標的跨 AWS 服務儀表板，或將檢視集中在特定資源群組或特定 AWS 服務上。這可讓您將檢視限縮於您感興趣的資源子集。如需詳細資訊，請參閱下列區段。

查看單一服務的自動預建儀表板

若要查看單一服務的自動預建儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/

   首頁出現。
2. 在左側導覽窗格中，選擇 [儀表板]。
3. 選擇 [自動儀表板] 索引標籤，然後選擇您要查看的服務。
4. 若要切換至檢視此服務的警示，請選取位於目前顯示服務名稱之螢幕頂端附近的 [進入警示]、[資料不足] 或 [確定] 核取方塊。
5. 檢視指標時，您有多種方式可以專注於特定指標：

   a. 若要查看任何圖形中的指標的詳細資訊，請將滑鼠游標移至圖形上，然後選擇動作圖示：View in metrics (在指標中檢視)。

      圖形會出現在新的標籤中，相關的指標會列在圖形下方。您可以自訂此圖形的檢視，包括變更所顯示的指標和資源、統計、期間和其他因素，以更加了解目前的情況。

   b. 您可以從圖形所顯示的時間範圍檢視日誌事件。這可協助您找出基礎設施中造成指標意外變更的事件。

      若要查看日誌事件，請將滑鼠游標移至圖形上，然後選擇動作圖示：View in logs (在日誌中檢視)。

      [記 CloudWatch 錄] 檢視會出現在新索引標籤中，並顯示記錄群組的清單。若要查看原始圖形所示時間範圍內在這些日誌群組中發生的日誌事件，請選擇該日誌群組。
6. 檢視警示時，您有多種方式可以專注於特定警示：

   - 若要查看警示的詳細資訊，請將滑鼠游標移至警示上，然後選擇動作圖示：View in alarms (在警示中檢視)。

     警示檢視會出現在新的標籤中，其中顯示警示的清單，以及所選擇警示的詳細資訊。若要查看此警示的歷史記錄，請選擇 History (歷史記錄) 標籤。

7. 警示一律每分鐘重新整理一次。若要重新整理檢視，請選擇螢幕右上角的重新整理圖示 (兩個彎曲箭頭)。若要變更螢幕上警示以外項目的自動重新整理率，請選擇重新整理圖示旁的向下箭頭，然後選擇重新整理速率。您也可以選擇關閉自動重新整理。
8. 若要變更目前顯示的所有圖形和警示中出現的時間範圍，請在螢幕上方的 Time range (時間範圍) 旁選擇範圍。若要從比預設顯示的還要更多的時間範圍選項中選擇，請選擇 custom (自訂)。
9. 若要返回跨服務儀表板，請在螢幕上方的清單 (目前顯示您專注的服務) 中選擇 Overview (概觀)。

   或者，您也可以從任何檢視畫面CloudWatch頂端選擇清除所有篩選條件，然後返回概覽頁面。

查看預先建置的跨服務儀表板

您可以切換至跨服務儀表板畫面，並與您正在使用的所有 AWS 服務的儀表板互動。主 CloudWatch 控台會依字母順序顯示儀表板，並在每個儀表板上顯示一或兩個關鍵指標。

> **Note**
>
> 如果您使用五個以上的 AWS 服務， CloudWatch 主控台不會在 [概觀] 畫面上顯示跨服務儀表板。

開啟跨服務儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/

   您將被導向到 Overview (概觀) 畫面。
2. 從 Overview (概觀) 畫面中，選取顯示 Overview (概觀) 的下拉式清單，然後選擇 Cross service dashboard (跨服務儀表板)。

   您將被導向到跨服務儀表板畫面。
3. (選用) 如果您使用的是原始介面，請滾動至區段 Cross-service dashboard (跨服務儀表板)，然後選擇 View Cross-service dashboard (檢視跨服務儀表板)。

   您將被導向到跨服務儀表板畫面。
4. 您有兩種方式可專注於特定服務：

   a. 若要查看服務的更多關鍵指標，請從螢幕上方的清單 (目前顯示 Cross service dashboard (跨服務儀表板)) 中選擇其名稱。或者，您可以選擇服務名稱旁的 View Service dashboard (檢視服務儀表板)。

出現該服務的自動儀表板，其中顯示該服務的更多指標。另外，對於某些服務，服務儀表板底部會顯示該服務的相關資源。您可以選擇其中一個資源給該服務主控台，並進一步專注於該資源。

b. 若要查看服務相關的所有警示，請在螢幕右邊選擇該服務名稱旁的按鈕。這個按鈕的文字指出您在這項服務中已建立多少警示，以及是否有任何警示處於 ALARM 狀態。

出現多個警示時，具有類似設定 (例如，維度、閾值或期間) 的多個警示可能顯示在單一圖形中。

然後，您可以檢視警示的詳細資訊，並查看警示歷史記錄。若要這樣做，請將滑鼠游標移至警示圖形上，然後選擇動作圖示：View in alarms (在警示中檢視)。

警示檢視會出現在新的瀏覽器標籤中，其中顯示警示的清單，以及所選擇警示的詳細資訊。若要查看此警示的歷史記錄，請選擇 History (歷史記錄) 標籤。

5. 您可以專注於特定資源群組中的資源。若要這樣做，請從頁面頂端顯示 All resources (所有資源) 的清單中選擇資源群組。

如需詳細資訊，請參閱 查看資源群組的預先建置儀表板。

6. 若要變更目前顯示的所有圖形和警示中出現的時間範圍，請在螢幕上方的 Time range (時間範圍) 旁選取您要的範圍。選擇 custom (自訂)，從比預設顯示的還要更多的時間範圍選項中選擇。
7. 警示一律每分鐘重新整理一次。若要重新整理檢視，請選擇螢幕右上角的重新整理圖示 (兩個彎曲箭頭)。若要變更螢幕上警示以外項目的自動重新整理率，請選擇重新整理圖示旁的向下箭頭，然後選擇您要的重新整理速率。您也可以選擇關閉自動重新整理。

從跨服務儀表板移除服務

您可以防止服務的指標出現在跨服務儀表板中。這可協助您在跨服務儀表板上專注於您最想監控的服務。

如果您從跨服務儀表板中移除服務，該服務的警示仍然會顯示在警示的檢視中。

若要從跨服務儀表板中移除服務的指標

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/

首頁出現。

2. 在頁面頂端，在 Overview (概觀) 下選擇您想要移除的服務。

檢視會變更成只顯示該服務中的指標。

3. 選擇 Actions (動作)，然後清除 Show on cross service dashboard (顯示在跨服務儀表板) 旁的核取方塊。

查看資源群組的預先建置儀表板

您可以將檢視專注於顯示單一資源群組中的指標和警示。使用資源群組可讓您使用標籤來組織專案、專注於架構的子集，或區別生產和開發環境。它們還可讓您將焦點放在 CloudWatch 概觀上的每個資源群組。如需詳細資訊，請參閱什麼是 AWS Resource Groups？。

當您專注於資源群組時，畫面會變成只顯示您已將資源標記為此資源群組一部分的服務。最近警示區域只會顯示與屬於資源群組的資源相關聯的警示。此外，如果您已建立名稱為-D CloudWatchefault- 的儀表板 ResourceGroupName，則它會顯示在「預設」管控面板區域中。

您可以同時著重於單一 AWS 服務與資源群組，進一步向下鑽研。下列程序僅說明如何將焦點放在資源群組上。

專注於單一資源群組

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在頁面頂端 (顯示 All resources (所有資源))，選擇資源群組。
3. 若要查看此資源群組的更多相關指標，請在螢幕底部附近選擇 View cross service dashboard (檢視跨服務儀表板)。

   出現跨服務儀表板，其中只顯示此資源群組相關的服務。每個服務會顯示一兩個關鍵指標。
4. 若要變更目前顯示的所有圖形和警示中出現的時間範圍，請在螢幕上方的 Time range (時間範圍) 中選取範圍。若要從比預設顯示的還要更多的時間範圍選項中選擇，請選擇 custom (自訂)。
5. 警示一律每分鐘重新整理一次。若要重新整理檢視，請選擇螢幕右上角的重新整理圖示 (兩個彎曲箭頭)。若要變更螢幕上警示以外項目的自動重新整理率，請選擇重新整理圖示旁的向下箭頭，然後選擇重新整理速率。您也可以選擇關閉自動重新整理。
6. 若要恢復顯示您的帳戶中所有資源的相關資訊，請在目前顯示資源群組名稱的螢幕上方附近，選擇 All resources (所有資源)。

查看預先建置的跨服務儀表板

您可以切換至跨服務儀表板畫面，並與您正在使用的所有 AWS 服務的儀表板互動。主 CloudWatch 控台會依字母順序顯示儀表板，並顯示每個服務的一或兩個關鍵指標。

> **Note**
>
> 如果您使用五個以上的 AWS 服務， CloudWatch 主控台不會在 [概觀] 畫面上顯示跨服務儀表板。

開啟跨服務儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/

   您將被導向到 Overview (概觀) 畫面。

2. 從 Overview (概觀) 畫面中，選取顯示 Overview (概觀) 的下拉式清單，然後選擇 Cross service dashboard (跨服務儀表板)。

   您將被導向到跨服務儀表板畫面。

3. (選用) 如果您使用的是原始介面，請滾動至區段 Cross-service dashboard (跨服務儀表板)，然後選擇 View Cross-service dashboard (檢視跨服務儀表板)。

   您將被導向到跨服務儀表板畫面。

4. 您有兩種方式可專注於特定服務：

   a. 若要查看服務的更多關鍵指標，請從螢幕上方的清單 (目前顯示 Cross service dashboard (跨服務儀表板)) 中選擇其名稱。或者，您可以選擇服務名稱旁的 View Service dashboard (檢視服務儀表板)。

      出現該服務的自動儀表板，其中顯示該服務的更多指標。另外，對於某些服務，服務儀表板底部會顯示該服務的相關資源。您可以選擇其中一個資源給該服務主控台，並進一步專注於該資源。

   b. 若要查看服務相關的所有警示，請在螢幕右邊選擇該服務名稱旁的按鈕。這個按鈕的文字指出您在這項服務中已建立多少警示，以及是否有任何警示處於 ALARM 狀態。

      出現多個警示時，具有類似設定 (例如，維度、閾值或期間) 的多個警示可能顯示在單一圖形中。

然後，您可以檢視警示的詳細資訊，並查看警示歷史記錄。若要這樣做，請將滑鼠游標移至警示圖形上，然後選擇動作圖示：View in alarms (在警示中檢視)。

警示檢視會出現在新的瀏覽器標籤中，其中顯示警示的清單，以及所選擇警示的詳細資訊。若要查看此警示的歷史記錄，請選擇 History (歷史記錄) 標籤。

5. 您可以專注於特定資源群組中的資源。若要這樣做，請從頁面頂端顯示 All resources (所有資源) 的清單中選擇資源群組。

   如需詳細資訊，請參閱 查看資源群組的預先建置儀表板。

6. 若要變更目前顯示的所有圖形和警示中出現的時間範圍，請在螢幕上方的 Time range (時間範圍) 旁選取您要的範圍。選擇 custom (自訂)，從比預設顯示的還要更多的時間範圍選項中選擇。
7. 警示一律每分鐘重新整理一次。若要重新整理檢視，請選擇螢幕右上角的重新整理圖示 (兩個彎曲箭頭)。若要變更螢幕上警示以外項目的自動重新整理率，請選擇重新整理圖示旁的向下箭頭，然後選擇您要的重新整理速率。您也可以選擇關閉自動重新整理。

移除服務以避免出現在跨服務儀表板中

您可以防止服務的指標出現在跨服務儀表板中。這可協助您在跨服務儀表板上專注於您最想監控的服務。

如果您從跨服務儀表板中移除服務，該服務的警示仍然會顯示在警示的檢視中。

若要從跨服務儀表板中移除服務的指標

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/

   首頁出現。
2. 在頁面頂端，在 Overview (概觀) 下選擇您想要移除的服務。

   檢視會變更成只顯示該服務中的指標。
3. 選擇 Actions (動作)，然後清除 Show on cross service dashboard (顯示在跨服務儀表板) 旁的核取方塊。

查看單 AWS 一服務的預先建置儀表板

在 CloudWatch 首頁上，您可以將檢視集中在單一 AWS 服務上。您可以同時著重於單一 AWS 服務與資源群組，進一步向下鑽研。下列程序僅顯示如何將焦點放在 AWS 服務上。

專注於單一服務

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/

   首頁出現。
2. 對於 [概觀]，其中 [概觀] 目前顯示在下拉式功能表中，選擇 [服務儀表板]。
3. 選擇您要關注的服務。

   檢視會變更成顯示所選服務的關鍵指標圖形。
4. 若要切換至檢視此服務的警示，請選取位於目前顯示服務名稱之螢幕頂端附近的 [進入警示]、[資料不足] 或 [確定] 核取方塊。
5. 檢視指標時，您有多種方式可以專注於特定指標：

   a. 若要查看任何圖形中的指標的詳細資訊，請將滑鼠游標移至圖形上，然後選擇動作圖示：View in metrics (在指標中檢視)。

      圖形會出現在新的標籤中，相關的指標會列在圖形下方。您可以自訂此圖形的檢視，包括變更所顯示的指標和資源、統計、期間和其他因素，以更加了解目前的情況。

   b. 您可以從圖形所顯示的時間範圍檢視日誌事件。這可協助您找出基礎設施中造成指標意外變更的事件。

      若要查看日誌事件，請將滑鼠游標移至圖形上，然後選擇動作圖示：View in logs (在日誌中檢視)。

      [記 CloudWatch 錄] 檢視會出現在新索引標籤中，並顯示記錄群組的清單。若要查看原始圖形所示時間範圍內在這些日誌群組中發生的日誌事件，請選擇該日誌群組。
6. 檢視警示時，您有多種方式可以專注於特定警示：

   - 若要查看警示的詳細資訊，請將滑鼠游標移至警示上，然後選擇動作圖示：View in alarms (在警示中檢視)。

     警示檢視會出現在新的標籤中，其中顯示警示的清單，以及所選擇警示的詳細資訊。若要查看此警示的歷史記錄，請選擇 History (歷史記錄) 標籤。
7. 警示一律每分鐘重新整理一次。若要重新整理檢視，請選擇螢幕右上角的重新整理圖示 (兩個彎曲箭頭)。若要變更螢幕上警示以外項目的自動重新整理率，請選擇重新整理圖示旁的向下箭頭，然後選擇重新整理速率。您也可以選擇關閉自動重新整理。
8. 若要變更目前顯示的所有圖形和警示中出現的時間範圍，請在螢幕上方的 Time range (時間範圍) 旁選擇範圍。若要從比預設顯示的還要更多的時間範圍選項中選擇，請選擇 custom (自訂)。

9. 若要返回跨服務儀表板，請在螢幕上方的清單 (目前顯示您專注的服務) 中選擇 Overview (概觀)。

   或者，您也可以從任何檢視畫面CloudWatch頂端選擇清除所有篩選條件，然後返回概覽頁面。

查看資源群組的預先建置儀表板

您可以將檢視專注於顯示單一資源群組中的指標和警示。使用資源群組可讓您使用標籤來組織專案、專注於架構的子集，或區別生產和開發環境。它們還可讓您將焦點放在 CloudWatch 概觀上的每個資源群組。如需詳細資訊，請參閱什麼是 AWS Resource Groups？。

當您專注於資源群組時，畫面會變成只顯示您已將資源標記為此資源群組一部分的服務。最近警示區域只會顯示與屬於資源群組的資源相關聯的警示。此外，如果您已建立名稱為-D CloudWatchefault- 的儀表板 ResourceGroupName，則它會顯示在「預設」管控面板區域中。

您可以同時著重於單一 AWS 服務與資源群組，進一步向下鑽研。下列程序只顯示如何專注於資源群組。

專注於單一資源群組

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在頁面頂端 (顯示 All resources (所有資源))，選擇資源群組。
3. 若要查看此資源群組的更多相關指標，請在螢幕底部附近選擇 View cross service dashboard (檢視跨服務儀表板)。

   出現跨服務儀表板，其中只顯示此資源群組相關的服務。每個服務會顯示一兩個關鍵指標。
4. 若要變更目前顯示的所有圖形和警示中出現的時間範圍，請在螢幕上方的 Time range (時間範圍) 中選取範圍。若要從比預設顯示的還要更多的時間範圍選項中選擇，請選擇 custom (自訂)。
5. 警示一律每分鐘重新整理一次。若要重新整理檢視，請選擇螢幕右上角的重新整理圖示 (兩個彎曲箭頭)。若要變更螢幕上警示以外項目的自動重新整理率，請選擇重新整理圖示旁的向下箭頭，然後選擇重新整理速率。您也可以選擇關閉自動重新整理。
6. 若要恢復顯示您的帳戶中所有資源的相關資訊，請在目前顯示資源群組名稱的螢幕上方附近，選擇 All resources (所有資源)。

CloudWatch 帳單和成本

本節說明 Amazon CloudWatch 功能如何產生成本。它還提供了可以幫助您分析，優化和降低 CloudWatch 成本的方法。在本節中，我們有時會在描述 CloudWatch 功能時參考定價。如需有關定價的資訊，請參閱 Amazon CloudWatch 定價。

主題

- 使用 CloudWatch Cost Explorer 分析成本和用量資料
- 使用 s 和 Athena 分析 CloudWatch 成本和 AWS Cost and Usage Report用量資料
- 最佳化和減少成本的最佳實務

使用 CloudWatch Cost Explorer 分析成本和用量資料

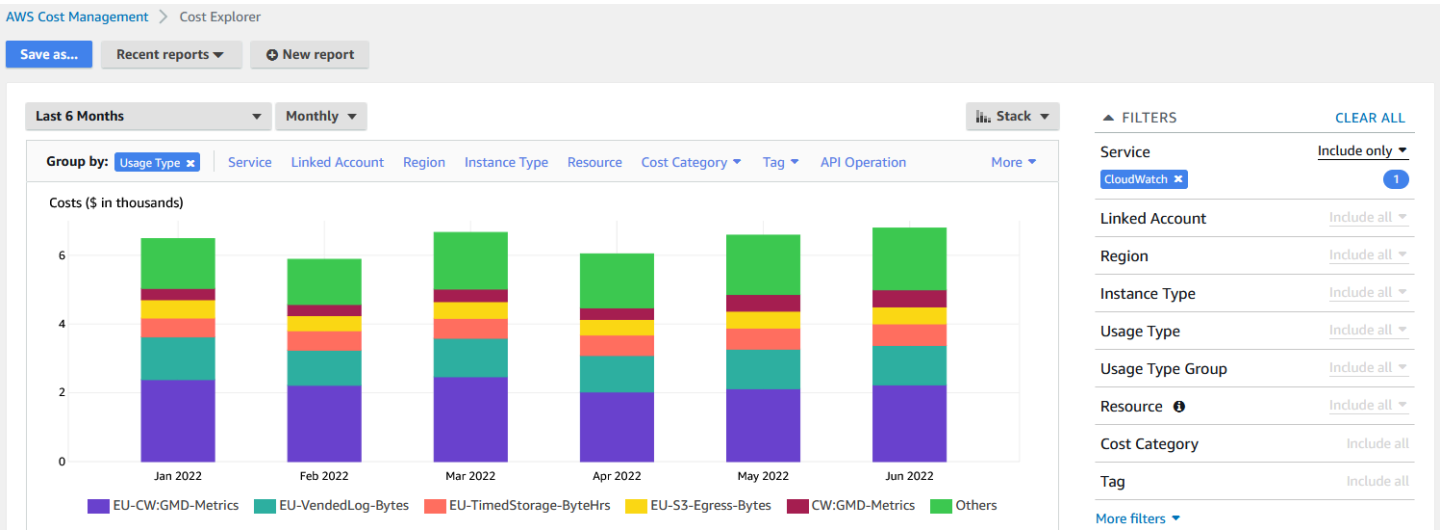
使用 AWS Cost Explorer，您可以視覺化並分析一段時間內的成本和使用情況資料，包括 CloudWatch. AWS 服務 如需詳細資訊，請參閱開始使用 AWS Cost Explorer。

下列程序說明如何使用 Cost Explorer 來視覺化及分析 CloudWatch 成本與使用量資料。

視覺化和分析 CloudWatch 成本和使用情況資料

1. 登入 Cost Explorer 主控台，網址為：https://console.aws.amazon.com/cost-management/home#/custom。
2. 在「篩選器」下，針對「服務」選取CloudWatch。
3. 針對 Group by (分組依據)，選擇 Usage Type (用量類型)。您也可以依其他類別來分組結果，例如：
   - API Operation (API 操作)：查看哪些 API 操作產生的成本最多。
   - Region (區域)：查看哪些區域產生的成本最多。

下圖顯示了超過六個月的 CloudWatch 特徵產生的成本範例。



若要查看哪些 CloudWatch 圖徵產生的成本最高，請查看的值`UsageType`。例如，`EU-CW:GMD-Metrics`代表 CloudWatch 大量 API 請求產生的成本。

> **Note**
>
> `UsageType` 的字串符合特定功能和區域。例如，`EU-CW:GMD-Metrics`(EU) 的第一部分符合歐洲 (愛爾蘭) 區域，而 `EU-CW:GMD-Metrics` (GMD-Metrics) 的第二部分則符合 CloudWatch 大量 API 要求。
>
> `UsageType` 的整個字串可格式化如下所示：`<Region>-CW:<Feature>` 或 `<Region>-<Feature>`。
>
> 為了增強可讀性，整個文件的表格中 `UsageType` 的字串已縮短為其字串尾碼。例如：`EU-CW:GMD-Metrics` 縮短為 `GMD-Metrics`。

下表包括每個 CloudWatch 特徵的名稱、列出每個子特徵的名稱，以及列出的字串。`UsageType`

CloudWatch 特徵	CloudWatch 子特徵	**UsageType**
CloudWatch 度量	自訂指標	`MetricMonitorUsage`
	詳細監控	`MetricMonitorUsage`
	內嵌指標	`MetricMonitorUsage`

CloudWatch 特徵	CloudWatch 子特徵	**UsageType**
CloudWatch API 要求	API 請求	Requests
	大量 (取得)	GMD-Metrics
	Contributor Insights	GIRR-Metrics
	點陣圖片快照	GMWI-Metrics
CloudWatch 測量結果流	指標串流	MetricStreamUsage
CloudWatch 儀表板	含 50 個或更少指標的儀表板	DashboardsUsageHour-Basic
	含超過 50 個指標的儀表板	DashboardsUsageHour
CloudWatch 警報	標準 (指標警示)	AlarmMonitorUsage
	高解析度 (指標警示)	HighResAlarmMonitorUsage
	Metrics Insights 查詢警示	MetricInsightAlarmUsage
	複合 (彙總警示)	CompositeAlarmMonitorUsage
CloudWatch 應用信號	應用信號	Application-Signals

CloudWatch 特徵	CloudWatch 子特徵	**UsageType**
CloudWatch 自訂記錄	收集 (擷取)	DataProcessing-Bytes
	儲存 (封存)	TimedStorage-ByteHrs
	分析 (查詢)	DataScanned-Bytes
CloudWatch 不常存取記錄	收集 (擷取)	DataProcessingIA-Bytes
CloudWatch 付費記錄	交付 (Amazon CloudWatch 日誌)	VendedLog-Bytes
	傳遞 (CloudWatch 記錄不常存取記錄)	VendedLogIA-Bytes
	傳送 (Amazon Simple Storage Service)	S3-Egress-ComprBytes S3-Egress-Bytes
	交付 (Amazon 數據 Firehose)	FH-Egress-Bytes
Contributor Insights	CloudWatch 記錄檔 (規則)	ContributorInsightRules
	CloudWatch 記錄檔 (事件)	ContributorInsightEvents
	Amazon DynamoDB (規則)	ContributorRulesManaged

CloudWatch 特徵	CloudWatch 子特徵	**UsageType**
	DynamoDB (事件)	`ContributorEventsManaged`
金絲雀 (Synthetics)	Run (執行)	`Canary-runs`
Evidently	事件	`Evidently-event`
	分析單位	`Evidently-eau`
RUM	事件	`RUM-event`

使用 s 和 Athena 分析 CloudWatch 成本和 AWS Cost and Usage Report用量資料

另一種分析成 CloudWatch 本和用量資料的方法是搭配 Amazon Athena 使用 AWS Cost and Usage Report s。 AWS Cost and Usage Report s 包含一組完整的成本和使用情況資料。您可以建立可追蹤成本和用量的報告，並可將這些報告發佈至您所選的 S3 儲存貯體。您也可以從您的 S3 儲存貯體下載並刪除您的報告。如需詳細資訊，請參閱什麼是 AWS Cost and Usage Report s？ 在「使用AWS Cost and Usage Report者指南」中。

> **Note**
>
> 使用 s 不收取任何費 AWS Cost and Usage Report用。您只需在將報告發佈至 Amazon Simple Storage Service (Amazon S3) 時支付儲存費用。如需詳細資訊，請參閱《AWS Cost and Usage Report使用者指南》中的配額和限制。

Athena 是一項查詢服務，您可以使用它來 AWS Cost and Usage Report分析成本和使用情況資料。您可以在 S3 儲存貯體中查詢報告，而無需先進行下載。如需詳細資訊，請參閱《Amazon Athena 使用者指南》中的什麼是 Amazon Athena？。如需詳細資訊，請參閱《Amazon Athena 使用者指南》中的什麼是 Amazon Athena？。如需有關定價的資訊，請參閱 Amazon Athena 定價。

下列程序說明啟 AWS Cost and Usage Report用服務及與 Athena 整合服務的程序。此程序包含兩個範例查詢，可用來分析 CloudWatch 成本與使用情況資料。

> **Note**
>
> 您可以使用本文件中的任何範例查詢。本文件中的所有範例查詢均對應於名為 costandusagereport 的資料庫，並顯示 2022 年 4 月的結果。您可以變更此資訊。不過，在執行查詢之前，請確定您的資料庫名稱與查詢中的資料庫名稱相符。

與 AWS Cost and Usage Report s 和 Athena 分析成本和使用情況資料

1. 啟用 AWS Cost and Usage Report如需詳細資訊，請參閱《AWS Cost and Usage Report使用者指南》中的建立成本和用量報告。

   > **Tip**
   >
   > 建立報告時，請確認選取 Include resource IDs (包含資源 ID)。否則，您的報告將不會包含 `line_item_resource_id` 資料欄。此行可協助您在分析成本和用量資料時進一步識別成本。

2. 與 AWS Cost and Usage Report Athena 集成。如需詳細資訊，請參閱《使用手冊》中的「AWS Cost and Usage Report使用 AWS CloudFormation 範本設定 Athena」。
3. 查詢您的成本和用量報告。

範例：Athena 查詢

您可以使用下列查詢來顯示在 CloudWatch 特定月份產生最多成本的功能。

```
SELECT
CASE
-- Metrics
WHEN line_item_usage_type LIKE '%%MetricMonitorUsage%%' THEN 'Metrics (Custom, Detailed
 monitoring management portal EMF)'
WHEN line_item_usage_type LIKE '%%Requests%%' THEN 'Metrics (API Requests)'
WHEN line_item_usage_type LIKE '%%GMD-Metrics%%' THEN 'Metrics (Bulk API Requests)'
WHEN line_item_usage_type LIKE '%%MetricStreamUsage%%' THEN 'Metric Streams'
-- Dashboard
WHEN line_item_usage_type LIKE '%%DashboardsUsageHour%%' THEN 'Dashboards'
```

```
-- Alarms
WHEN line_item_usage_type LIKE '%%AlarmMonitorUsage%%' THEN 'Alarms (Standard)'
WHEN line_item_usage_type LIKE '%%HighResAlarmMonitorUsage%%' THEN 'Alarms (High
 Resolution)'
WHEN line_item_usage_type LIKE '%%MetricInsightAlarmUsage%%' THEN 'Alarms (Metrics
 Insights)'
WHEN line_item_usage_type LIKE '%%CompositeAlarmMonitorUsage%%' THEN 'Alarms
 (Composite)'
-- Logs
WHEN line_item_usage_type LIKE '%%DataProcessing-Bytes%%' THEN 'Logs (Collect - Data
 Ingestion)'
-- Logs
WHEN line_item_usage_type LIKE '%%DataProcessingIA-Bytes%%' THEN 'Infrequent Access
 Logs (Collect - Data Ingestion)'
WHEN line_item_usage_type LIKE '%%TimedStorage-ByteHrs%%' THEN 'Logs (Storage -
 Archival)'
WHEN line_item_usage_type LIKE '%%DataScanned-Bytes%%' THEN 'Logs (Analyze - Logs
 Insights queries)'
-- Vended Logs
WHEN line_item_usage_type LIKE '%%VendedLog-Bytes%%' THEN 'Vended Logs (Delivered to
 CW)'
WHEN line_item_usage_type LIKE '%%VendedLogIA-Bytes%%' THEN 'Vended Infrequent Access
 Logs (Delivered to CW)'
WHEN line_item_usage_type LIKE '%%FH-Egress-Bytes%%' THEN 'Vended Logs (Delivered to
 Kinesis FH)'
WHEN (line_item_usage_type LIKE '%%S3-Egress-Bytes%%') OR (line_item_usage_type LIKE '%
%S3-Egress-
ComprBytes%%') THEN 'Vended Logs (Delivered to S3)'
-- Other
WHEN line_item_usage_type LIKE '%%Application-Signals%%' THEN 'Application Signals'
WHEN line_item_usage_type LIKE '%%Canary-runs%%' THEN 'Synthetics'
WHEN line_item_usage_type LIKE '%%Evidently%%' THEN 'Evidently'
WHEN line_item_usage_type LIKE '%%RUM-event%%' THEN 'RUM'
ELSE 'Others'
END AS UsageType,
-- REGEXP_EXTRACT(line_item_resource_id,'^(?:.+?:){5}(.+)$',1) as ResourceID,
-- SUM(CAST(line_item_usage_amount AS double)) AS UsageQuantity,
SUM(CAST(line_item_unblended_cost AS decimal(16,8))) AS TotalSpend
FROM
costandusagereport
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
```

```
AND line_item_line_item_type NOT IN
 ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')
-- AND line_item_usage_account_id = '123456789012' - If you want to filter on a
 specific account, you can
remove this comment at the beginning of the line and specify an AWS account.
GROUP BY
1
ORDER BY
TotalSpend DESC,
UsageType;
```

範例：Athena 查詢

您可以使用以下查詢來顯示 UsageType 和 Operation 的結果。這會顯示 CloudWatch 功能如何產生成本。結果也會顯示 UsageQuantity 和 TotalSpend 的值，以便您可以查看總用量成本。

> **Tip**
>
> 如需有關 UsageType 的詳細資訊，請將下列一行新增至此查詢：
> `line_item_line_item_description`
> 此行會建立一個名為 Description (描述) 的資料欄。

```
SELECT
bill_payer_account_id as Payer,
line_item_usage_account_id as LinkedAccount,
line_item_usage_type AS UsageType,
line_item_operation AS Operation,
line_item_resource_id AS ResourceID,
SUM(CAST(line_item_usage_amount AS double)) AS UsageQuantity,
SUM(CAST(line_item_unblended_cost AS decimal(16,8))) AS TotalSpend
FROM
costandusagereport
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
AND line_item_line_item_type NOT IN
 ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')
GROUP BY
bill_payer_account_id,
line_item_usage_account_id,
```

```
line_item_usage_type,
line_item_resource_id,
line_item_operation
```

最佳化和減少成本的最佳實務

CloudWatch 度量

許多人 AWS 服務，如 Amazon Elastic Compute Cloud (Amazon EC2)，Amazon S3 和 Amazon 數據 Firehose，自動將指標發送到 CloudWatch，不收費。不過，歸入以下類別的指標可能會產生額外的成本：

- Custom metrics, detailed monitoring, and embedded metrics (自訂指標、詳細監控和內嵌指標)
- API requests (API 請求)
- Metric streams (指標串流)

如需詳細資訊，請參閱使用 Amazon CloudWatch 指標。

Custom metrics, detailed monitoring, and embedded metrics (自訂指標、詳細監控和內嵌指標)

自訂指標

您可以建立自訂指標，以任何順序和任何速率來組織資料點。

所有自訂指標均依小時按比例分配。只有在傳送到 CloudWatch時才會計量它們。有關指標定價方式的詳細資訊，請參閱 Amazon CloudWatch 定價。

下表列出 CloudWatch 量度的相關子功能名稱。資料表包含 `UsageType` 和 `Operation` 的字串，可協助您分析和識別指標相關的成本。

> **Note**
>
> 若要在使用 Athena 查詢成本和用量資料時，取得有關下表所列指標的詳細資訊，請使 `Operation` 的字串與 `line_item_operation` 的顯示結果相符。

CloudWatch子特徵	**`UsageType`**	**`Operation`**	用途

CloudWatch子特徵	**UsageType**	**Operation**	用途
自訂指標	MetricMonitorUsage	MetricStorage	自訂指標
詳細監控	MetricMonitorUsage	MetricStorage:AWS/*{Service}*	詳細監控
內嵌指標	MetricMonitorUsage	MetricStorage:AWS/Logs-EMF	日誌內嵌指標
日誌篩選條件	MetricMonitorUsage	MetricStorage:AWS/CloudWatchLogs	日誌群組指標篩選條件

詳細監控

CloudWatch 有兩種類型的監控：

- 基本監控

  基本監控免費，並為所有支援此功能的 AWS 服務 自動啟用。
- 詳細監控

  詳細監控會產生成本，並根據. AWS 服務針對每個支援詳細監控的 AWS 服務 ，您可以選擇是否為該服務將其啟用。如需詳細資訊，請參閱基本和詳細監控。

> **Note**
>
> 其他 AWS 服務 支持詳細監控，並可能使用不同的名稱引用此功能。例如，Amazon S3 的詳細監控稱為 request metrics (請求指標)。

與自訂指標類似，詳細監控會按小時按比例分配，並且僅在資料傳送至時計量。 CloudWatch詳細監控會根據傳送至的指標數量產生成本 CloudWatch。為了降低成本，請僅在必要時啟用詳細監控。如需詳細監控定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

範例：Athena 查詢

您可以使用以下查詢來顯示哪些 EC2 執行個體已啟用詳細監控。

```
SELECT
bill_payer_account_id as Payer,
line_item_usage_account_id as LinkedAccount,
line_item_usage_type AS UsageType,
line_item_operation AS Operation,
line_item_resource_id AS ResourceID,
SUM(CAST(line_item_usage_amount AS double)) AS UsageQuantity,
SUM(CAST(line_item_unblended_cost AS decimal(16,8))) AS TotalSpend
FROM
costandusagereport
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
AND line_item_operation='MetricStorage:AWS/EC2'
AND line_item_line_item_type NOT IN
 ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')
GROUP BY
bill_payer_account_id,
line_item_usage_account_id,
line_item_usage_type,
line_item_resource_id,
line_item_operation,
line_item_line_item_description
ORDER BY line_item_operation
```

內嵌指標

透過內 CloudWatch 嵌指標格式，您可以將應用程式資料擷取為記錄資料，以便產生可採取動作的指標。如需詳細資訊，請參閱擷取高基數記錄和使用內嵌指標格式產生指標 CloudWatch 。

內嵌指標會依擷取的日誌數量、封存的日誌數量以及產生的自訂指標數量來產生成本。

下表列出內 CloudWatch 嵌量度格式的相關子功能名稱。資料表包含 `UsageType` 和 `Operation` 的字串，可協助您分析和識別成本。

CloudWatch 子特徵	**`UsageType`**	**`Operation`**	用途
			日誌內嵌指標

CloudWatch 子特徵	**UsageType**	**Operation**	用途
自訂指標	MetricMonitorUsage	MetricStorage:AWS/Logs-EMF	
日誌擷取	DataProcessing-Bytes	PutLogEvents	將批次日誌事件上傳至指定的日誌群組或日誌串流
日誌封存	TimedStorage-ByteHrs	HourlyStorageMetering	在記錄檔中儲存每小時記錄和每位元組的 CloudWatch 記錄

若要分析成本，請搭配 Athena 使用 AWS Cost and Usage Report s，以便識別哪些指標會產生成本，並決定如何產生成本。

為了充分利用 CloudWatch 內嵌指標格式產生的成本，請避免根據高基數維度建立指標。如此一來，就 CloudWatch 不會為每個唯一維度組合建立自訂量度。如需詳細資訊，請參閱維度。

如果您使用 CloudWatch 容器洞察來利用嵌入式指標格式，則可以使用開放遙測發行 AWS 版作為充分利用指標相關成本的替代方法。使用 Container Insights，您可收集、彙整和摘要您容器化的應用程式及微服務中的指標及日誌。當您啟用 Container Insights 時， CloudWatch 代理程式會將您的記錄檔傳送至 CloudWatch，以便使用記錄檔產生內嵌指標。不過， CloudWatch 代理程式只會傳送固定數量的指標給您 CloudWatch，而且您需支付所有可用指標 (包括未使用的任何指標) 的費用。使用開放遙測發行 AWS 版，您可以配置和自定義發送到 CloudWatch哪些指標和維度。這可協助您減少 Container Insights 所產生的資料量和成本。如需詳細資訊，請參閱下列資源：

- 使用 Container Insights
- AWS 開放式遙測發行版

API 請求

CloudWatch 具有以下類型的 API 請求：

- API requests (API 請求)
- Bulk (Get) (大量 (取得))
- Contributor Insights

- Bitmap image snapshot (點陣圖影像快照)

API 請求會根據請求類型和請求的指標數量產生成本。

下列資料表列出了 API 請求的類型，並包含 `UsageType` 和 `Operation` 的字串，可協助您分析和識別 API 相關的成本。

API 請求類型	**`UsageType`**	**`Operation`**	用途
API 請求	`Requests`	`GetMetricStatistics`	擷取指定指標的統計資料
	`Requests`	`ListMetrics`	列出指定的指標
	`Requests`	`PutMetricData`	將測量結果資料點發佈至 CloudWatch
	`Requests`	`GetDashboard`	顯示指定儀表板的詳細資訊
	`Requests`	`ListDashboards`	列出帳戶中的儀表板
	`Requests`	`PutDashboard`	建立或更新儀表板
	`Requests`	`DeleteDashboards`	刪除所有指定的儀表板
大量 (取得)	`GMD-Metrics`	`GetMetricData`	擷取 CloudWatch 測量結果值
Contributor Insights	`GIRR-Metrics`	`GetInsightRuleReport`	傳回 Contributor Insights 規則所收集的時間序列資料
點陣圖片快照	`GMWI-Metrics`	`GetMetricWidgetImage`	檢索一個或多個 CloudWatch 度量的快照作為點陣圖圖像

若要分析成本，請使用 Cost Explorer，並將結果依據 API Operation (API 操作) 分組。

API 請求的費用會有所不同，而且當您超過 AWS 免費方案限制下提供給您的 API 呼叫數量時會產生費用。

> **Note**
>
> GetMetricData且GetMetricWidgetImage不包含在 AWS 免費方案限制之內。如需詳細資訊，請參閱使用指南中的使AWS Billing 用 AWS 免費方案。

通常會增加成本的 API 請求是 Put 和 Get 請求。

PutMetricData

PutMetricData 每次呼叫時都會產生成本，並且根據使用案例可能會產生可觀的成本。如需詳細資訊，請PutMetricData參閱 Amazon CloudWatch API 參考中的。

為了充分利用 PutMetricData 所產生的成本，請在您的 API 呼叫中批次處理更多資料。根據您的使用案例，請考慮使用 CloudWatch 記錄檔或內 CloudWatch 嵌指標格式來插入指標資料。如需詳細資訊，請參閱下列資源：

- 什麼是 Amazon CloudWatch 日誌？ 在 Amazon CloudWatch 日誌用戶指南
- 擷取高基數日誌並使用內嵌指標格式產生指標 CloudWatch
- 利用 Amazon CloudWatch 嵌入式自訂指標降低成本並專注於客戶

GetMetricData

GetMetricData 也會產生可觀的成本。增加成本的常見使用案例涉及第三方監控工具，這些工具會提取資料以產生深入解析。如需詳細資訊，請GetMetricData參閱 Amazon CloudWatch API 參考中的。

為了降低 GetMetricData 產生的成本，請僅考慮提取受監控和使用的資料，或考慮減少提取資料的頻率。視您的使用案例而定，您可能會考慮使用指標串流 (而不是 GetMetricData)，以便您以較低的成本，將資料以近乎即時的方式推播給第三方。如需詳細資訊，請參閱下列資源：

- 使用指標串流
- CloudWatch 指標串流-即時傳送 AWS 指標給合作夥伴和您的應用程式

GetMetricStatistics

視您的使用案例而定，您可能會考慮使用 GetMetricStatistics 而不是 GetMetricData。透過 GetMetricData，您可以快速且大規模地擷取資料。但GetMetricStatistics是，最多可針對一百萬個 API 請求包含在 AWS 免費方案限制內，如果您不需要每次呼叫擷取盡可能多的指標和資料點，可協助您降低成本。如需詳細資訊，請參閱下列資源：

- GetMetricStatistics在 Amazon CloudWatch API 參考
- 我應該使用 GetMetricData 還是 GetMetricStatistics？

> **Note**
>
> 外部呼叫者進行 API 呼叫。目前，識別這些呼叫者的唯一方法是向 CloudWatch 團隊提出技術支持請求並詢問有關他們的信息。如需建立技術支援要求的相關資訊，請參閱如何取得技術支援 AWS？ 。

CloudWatch 測量結果流

透過指 CloudWatch 標串流，您可以將指標持續傳送至 AWS 目的地和第三方服務提供者目的地。

指標串流會根據指標更新的數量產生成本。指標更新一律包含以下統計資料的值：

- Minimum
- Maximum
- Sample Count
- Sum

如需詳細資訊，請參閱可供串流的統計資料。

若要分析 CloudWatch 指標串流產生的成本，請搭配 Athena 使用 AWS Cost and Usage Report s。如此一來，您就可以識別哪些指標串流會產生成本，並決定成本的產生方式。

範例：Athena 查詢

您可以使用以下查詢來依 Amazon Resource Name (ARN) 追蹤哪些指標串流會產生成本。

```
SELECT
SPLIT_PART(line_item_resource_id,'/',2) AS "Stream Name",
line_item_resource_id as ARN,
SUM(CAST(line_item_unblended_cost AS decimal(16,2))) AS TotalSpend
```

```
FROM
costandusagereport
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
AND line_item_line_item_type NOT IN
 ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')
-- AND line_item_usage_account_id = '123456789012' - If you want to filter on a
 specific account, you can
remove this comment at the beginning of the line and specify an AWS account.
AND line_item_usage_type LIKE '%%MetricStreamUsage%%'
GROUP BY line_item_resource_id
ORDER BY TotalSpend DESC
```

若要降低指 CloudWatch 標串流產生的成本，請僅串流帶來商業價值的指標。您也可以停止或暫停未使用的任何指標串流。

CloudWatch 警報

透過 CloudWatch警示，您可以根據單一量度建立警示、根據 Metric Insights 查詢建立警示，以及監視其他警示的複合警示。

> **Note**
>
> 指標和複合警示的費用按小時按比例分配。只有當警示存在時，您才會產生警示費用。為了優化成本，請確保不要留下配置錯誤或低價值的警報。若要協助解決此問題，您可以自動清除不再需要的 CloudWatch 警示。如需詳細資訊，請參閱大規模自動化 Amazon CloudWatch 警示清理

指標警示

指標警示具有下列解析度設定：

- Standard (標準) (每 60 秒評估一次)
- High resolution (高解析度) (每 10 秒評估一次)

如果建立指標警示，費用會根據警示的解析設定和警示參考的指標數量而定。例如，指標警示每參考一個指標，每小時就會產生一個警示指標成本。如需詳細資訊，請參閱使用 Amazon CloudWatch 警示。

如果建立的指標警示包含參考多個指標的指標數學表達式，則指標數學表達式中參考的每個警示指標都會產生費用。如需如何建立包含度量數學運算式的度量警示的詳細資訊，請參閱根據度量數學運算式建立 CloudWatch 警示。

如果建立的是異常偵測警示 (警示會分析過去的指標資料來建立預期值模型)，則警示中參考的每個警示指標 (以及兩個額外指標，其中一個用於異常偵測模型建立的上下限範圍指標) 都會產生費用。如需如何建立異常偵測警示的相關資訊，請參閱根據異常偵測建立 CloudWatch 警示。

Metrics Insights 查詢警示

Metric Insights 查詢警示是一種特定類型的指標警示，僅提供標準解析度 (每 60 秒評估一次)。

建立 Metric Insights 查詢警示時，費用會根據警示參考的查詢所分析的指標數量而定。例如，某個 Metric Insights 查詢警示所參考之查詢的篩選條件符合十個指標，那麼此查詢警示每小時會產生分析十個指標的成本。如需詳細資訊，請參閱 Amazon 定價上的 CloudWatch 定價範例。

如果您建立的警示同時包含 Metrics Insights 查詢和指標數學運算式，則會將其回報為 Metrics Insights 查詢警示。如果您的警示包含一個指標數學運算式，它不僅參考 Metrics Insights 查詢分析的指標，還參考其他指標，那麼指標數學運算式中參考的每個警報指標都會產生額外費用。如需如何建立包含度量數學運算式的度量警示的詳細資訊，請參閱根據度量數學運算式建立 CloudWatch 警示。

複合警示

複合警示包含指定如何評估其他警示狀態以判斷其自身狀態的規則表達式。複合警示會每小時產生標準費用，無論它們評估了多少個其他警示。複合警示的規則表達式參考的每個警示都會產生費用。如需詳細資訊，請參閱建立複合警示。

警示使用類型

下表列出 CloudWatch 警示的相關子功能名稱。資料表包含 UsageType 的字串，可協助您分析和識別與警示相關的成本。

CloudWatch子特徵	**UsageType**
標準指標警示	`AlarmMonitorUsage`
高解析度指標警示	`HighResAlarmMonitorUsage`
Metrics Insights 查詢警示	`MetricInsightAlarmUsage`
複合警示	`CompositeAlarmMonitorUsage`

降低警示成本

若要將彙總四個或更多量度的指標數學警示所產生的成本最佳化，您可以在資料傳送至之前彙總資料 CloudWatch。如此一來，您就可以為單一指標建立警示，而不是為多個指標彙總資料的警示。如需詳細資訊，請參閱發佈自訂指標。

若要最佳化 Metric Insights 查詢警示產生的成本，您可以確保用於查詢的篩選條件僅符合您要監控的指標。

降低成本的最佳方法是移除所有不必要或未使用的警示。例如，您可以刪除警示，以評估不再存在的 AWS 資源發出的指標。

範例：使用 ***DescribeAlarms*** 檢查狀態為 ***INSUFFICIENT_DATA*** 的警示

如果您刪除資源，但不刪除資源發出的指標警示，則警示會仍然存在，通常會進入 INSUFFICIENT_DATA 狀態。若要檢查處於此INSUFFICIENT_DATA狀態的警示，請使用下列 AWS Command Line Interface (AWS CLI) 指令。

```
$ aws cloudwatch describe-alarms –state-value INSUFFICIENT_DATA
```

其他降低成本的方法包括：

- 請確認為正確的指標建立警示。
- 請確認您未在您沒有運作的地區啟用任何警示。
- 請記住，儘管複合警示可以減少雜訊，但它們也會產生額外的費用。
- 在決定要建立標準警示或高解析度警示時，請考慮您的使用案例以及每種警示類型帶來的價值。

CloudWatch 日誌

Amazon CloudWatch 日誌具有以下日誌類型：

- 自訂日誌 (您為應用程式建立的日誌)
- 付費日誌（其他日誌 AWS 服務，例如 Amazon Virtual Private Cloud（Amazon VPC）和 Amazon 路線 53，代表您創建的日誌）

如需有關付費記錄的詳細資訊，請參閱 Amazon Logs 使用者指南中的啟用特定 AWS 服務的記 CloudWatch 錄功能。

自訂和付費日誌會根據收集、儲存及分析的日誌數量產生成本。另外，付費日誌會產生交付給 Amazon S3 和 Firehose 的成本。

下表列出 CloudWatch 記錄檔功能的名稱以及相關子功能的名稱。資料表包含 `UsageType` 和 `Operation` 的字串，可協助您分析和識別日誌相關的成本。

CloudWatch 日誌功能	CloudWatch 記錄子功能	**UsageType**	**Operation**	用途
自訂日誌	收集 (擷取)	`DataProcessing-Bytes`	`PutLogEvents`	將一批日誌上傳至特定的日誌串流
	儲存 (封存)	`TimedStorage-ByteHrs`	`HourlyStorageMetering`	在記錄檔中儲存每小時記錄和每位元組的 CloudWatch 記錄
	分析 (Logs Insights 查詢)	`DataScanned-Bytes`	`StartQuery`	日誌見解查詢掃描的 CloudWatch 日誌資料
付費日誌	傳送 (CloudWatch 記錄檔)	`VendedLog-Bytes`	`PutLogEvents`	將一批日誌上傳至特定的日誌串流
	傳送 (Amazon S3)	`S3-Egress-ComprBytes` `S3-Egress-Bytes`	`LogDelivery`	傳送付費日誌 (CloudWatch、Amazon S3 或 Firehose)
	送貨服務 (Firehose)	`FH-Egress-Bytes`	`LogDelivery`	傳送付費日誌 (CloudWatch、Amazon S3 或 Firehose)

若要分析成本，請搭配 Athena 使用 AWS Cost and Usage Report s，以便識別哪些記錄會產生成本，並判斷成本的產生方式。

範例：Athena 查詢

您可以使用以下查詢來依資源 ID 追蹤哪些日誌會產生成本。

```
SELECT
bill_payer_account_id as Payer,
line_item_usage_account_id as LinkedAccount,
line_item_resource_id AS ResourceID,
line_item_usage_type AS UsageType,
SUM(CAST(line_item_unblended_cost AS decimal(16,8))) AS TotalSpend,
SUM(CAST(line_item_usage_amount AS double)) AS UsageQuantity
FROM
costandusagereport
WHERE
product_product_name = 'AmazonCloudWatch'
AND year='2022'
AND month='4'
AND line_item_operation IN
 ('PutLogEvents','HourlyStorageMetering','StartQuery','LogDelivery')
AND line_item_line_item_type NOT IN
 ('Tax','Credit','Refund','EdpDiscount','Fee','RIFee')
GROUP BY
bill_payer_account_id,
line_item_usage_account_id,
line_item_usage_type,
line_item_resource_id,
line_item_operation
ORDER BY
TotalSpend DESC
```

若要充分利用 CloudWatch 記錄所產生的成本，請考慮下列事項：

- 僅記錄帶來商業價值的事件。這可協助您減少擷取的成本。
- 變更您的日誌保留設定，以降低儲存成本。如需詳細資訊，請參閱 Amazon CloudWatch 日誌使用者指南中的變更 CloudWatch 日誌資料保留。
- 執行 CloudWatch 日誌深入解析會自動儲存在記錄中的查詢。如此一來，您產生的分析成本就會減少。如需詳細資訊，請參閱 Amazon CloudWatch 日誌使用者指南中的檢視執行中的查詢或查詢歷史記錄。

- 使用 CloudWatch 代理程式收集系統和應用程式記錄檔，並將其傳送至 CloudWatch。如此一來，您就可以只收集符合條件的日誌事件。如需詳細資訊，請參閱 Amazon CloudWatch 代理程式新增日誌篩選器運算式的 Support。

若要降低付費日誌的成本，請考慮您的使用案例，然後判斷是否應將日誌傳送到 CloudWatch Amazon S3。如需詳細資訊，請參閱 Amazon 日誌使用者指南中的傳送至 Amazon S3 的 CloudWatch 日誌。

> **Tip**
>
> 如果您想要使用指標篩選器、訂閱篩選器、 CloudWatch 日誌深入解析和參與者見解，請將付費記錄傳送至。 CloudWatch
> 或者，如果您正在使用 VPC 流量日誌並將其用於稽核和合規目的，請將付費日誌傳送至 Amazon S3。
> 如需如何追蹤透過將 VPC 流程日誌發佈到 S3 儲存貯體所產生的費用的詳細資訊，請參閱使用 AWS Cost and Usage Report s 和成本分配標籤以了解 Amazon S3 中的 VPC 流程日誌資料擷取。

如需有關如何充分利用 Logs 產生的成本的其他資訊，請參閱哪個 CloudWatch 記錄群組導致我的 CloudWatch 記錄帳單突然增加？ 。

使用 Amazon CloudWatch 儀表

Amazon CloudWatch 儀表板是 CloudWatch 主控台中可自訂的首頁，您可以使用它們在單一檢視中監控資源，甚至是分散在不同區域的資源。您可以使用 CloudWatch 儀表板為 AWS 資源建立指標和警示的自訂檢視。

您可以利用儀表板來建立以下項目：

- 選取指標和警示的單一檢視，可協助您評估在一或多個區域之資源和應用程式的運作狀態。您可以選擇在每個圖形上用於每個指標的顏色，如此您就可以輕鬆地追蹤跨多個圖形的相同指標。
- 運作手冊，其會提供運作事件期間適用於團隊成員與如何回應特定事件的指導。
- 關鍵資源和應用程式測量的常用檢視，可以由團隊成員共用以在運作事件期間形成較快的通訊流程。

如果您有多個 AWS 帳戶，則可以設定CloudWatch 跨帳戶可觀察性，然後在監控帳戶中建立豐富的跨帳戶儀表板。這些儀表板可以包含來源帳戶和 CloudWatch 日誌見解 Widget 的指標圖形，以及來源帳戶的記錄群組查詢。此外，您在監控帳戶中建立的警示可監看來源帳戶中的指標。如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。

您可以從主控台或使用 AWS CLI 或 `PutDashboard` API 作業建立儀表板。您可以將儀表板新增至我的最愛清單中，從而可以在其中存取最愛的儀表板以及最近造訪過的儀表板。如需詳細資訊，請參閱將儀表板新增至我的最愛清單。

若要存取 CloudWatch 儀表板，您需要下列其中一項：

- `AdministratorAccess` 政策
- `CloudWatchFullAccess` 政策
- 自訂政策，其中包含一或多個特定許可：
  - `cloudwatch:GetDashboard` 和 `cloudwatch:ListDashboards` 能夠檢視儀表板
  - `cloudwatch:PutDashboard` 能夠建立或修改儀表板
  - `cloudwatch:DeleteDashboards` 能夠刪除儀表板

目錄

- 建立 CloudWatch 儀表板
- CloudWatch 跨帳戶可觀察儀表板
- 跨帳戶跨區域儀表板

- 使用儀表板變數建立彈性儀表板
- 在 CloudWatch 儀表板上建立和使用小器具
- 共用 CloudWatch 儀表板
- 使用即時資料
- 檢視動畫儀表板
- 將 CloudWatch 儀表板新增至我的最愛清單
- 變更 CloudWatch 儀表板的期間覆寫設定或重新整理間隔
- 變更 CloudWatch儀表板的時間範圍或時區格式

建立 CloudWatch 儀表板

若要開始使用，請建立 CloudWatch 儀表板。您可以建立多個儀表板，也可以將儀表板新增至我的最愛清單。您在 AWS 帳戶中可以擁有的儀表板數量不受限制。所有儀表板都是全域的，並不只屬於特定區域。

下列程序說明如何從 CloudWatch 主控台建立儀表板。您可以使用 `PutDashboard` API 操作從命令列介面建立儀表板。API 操作包含定義儀表板內容的 JSON 字串。如需使用 `PutDashboard` API 作業建立儀表板的詳細資訊，請參閱 Amazon CloudWatch API 參考PutDashboard中的。

> **Tip**
>
> 如果您使用 `PutDashboard` API 操作建立新的儀表板，您可以使用已存在的儀表板中的 JSON 字串。

從主控台建立儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇 Create dashboard (建立儀表板)。
3. 在 Create new dashboard (建立新的儀表板) 對話方塊中，輸入儀表板的名稱，然後選擇 Create dashboard (建立儀表板)。

   如果您使用名稱 CloudWatch-Default 或 CloudWatch-Default-***ResourceGroupName***，則儀表板會顯示在「預設儀表板」下 CloudWatch 首頁的概觀中。如需詳細資訊，請參閱 開始使用 Amazon CloudWatch。

4. 在 Add to this dashboard (新增至此儀表板) 對話方塊中，執行下列其中一項操作：
   - 若要將圖表新增至儀表板，選擇 Line (線條) 或 Stacked area (堆疊區域)，然後選擇 Configure (設定)。在 Add metric graph (新增指標圖表) 對話方塊中，選取要繪製圖表的指標，然後選擇 Create widget (建立小工具)。如果某個指標因為已超過 14 天沒有發佈資料，而未出現在對話方塊中，您可以手動新增該指標。如需詳細資訊，請參閱 在 CloudWatch 儀表板上手動繪製指標。
   - 若要將顯示指標的數字新增至儀表板，請選擇 Number (數字)，然後選擇 Configure (設定)。在 Add metric graph (新增指標圖表) 對話方塊中，選取要繪製圖表的指標，然後選擇 Create widget (建立小工具)。
   - 若要將文字區塊新增至儀表板，請選擇 Text (文字)，然後選擇 Configure (設定)。在 New text widget (新的文字小工具) 對話方塊中，對於 Markdown，使用 Markdown 格式化文字，然後選擇 Create widget (建立小工具)。
5. (選用) 選擇 Add widget (新增小工具)，然後重複步驟 4 以將另一個小工具新增至儀表板。您可以多次重複此步驟。

   在儀表板上的每個圖形，右上角都會有一個資訊圖示。選擇此圖示，可在圖表中查看指標的描述。
6. 選擇 Save dashboard (儲存儀表板)。

CloudWatch 跨帳戶可觀察儀表板

如果您有多個 AWS 帳戶，則可以設定CloudWatch 跨帳戶可觀察性，然後在監控帳戶中建立豐富的跨帳戶儀表板。您可以無縫地搜尋、視覺化和分析指標、日誌和追蹤，不受帳戶限制。

如需設定 CloudWatch 跨帳戶觀察性的詳細資訊，請參閱〈〉。CloudWatch 跨帳戶可觀察性

透過 CloudWatch 跨帳戶可觀察性，您可以在監控帳戶的儀表板中執行以下操作：

- 搜尋、檢視和建立來源帳戶中指標的圖表。單一圖表可包含來自多個帳戶的指標。
- 在監控帳戶中建立警示，監看來源帳戶中的指標。
- 從位於來源帳戶中的記錄群組檢視記錄事件，並在來源帳戶中執行 CloudWatch 記錄群組的 Logs Insights 查詢。監控帳戶中的單一 CloudWatch Logs Insights 查詢可以一次查詢多個來源帳戶中的多個記錄群組。
- 在 X-Ray 的追蹤地圖中檢視來源帳戶的節點。然後，您可以對地圖進行篩選，找出特定來源帳戶。

當您登錄到監控帳戶時，支持 CloudWatch 跨帳戶觀察功能的每個頁面的右上角都會顯示一個藍色的監控帳戶徽章。

跨帳戶跨區域儀表板

您可以建立跨帳戶跨區域儀表板，將來自多個 AWS 帳戶和多個區域的 CloudWatch 資料彙總到一個儀表板中。在此高階儀表板中，您可以取得整個應用程式的檢視，也可以深入了解更特定的儀表板，而不必登入和登出帳戶或切換區域。

您可以在 AWS Management Console 和程式設計方式中建立跨帳戶跨區域儀表板。

必要條件

在建立跨帳戶跨區域儀表板之前，您必須至少啟用一個共用帳戶和至少一個監控帳戶。此外，若要使用 CloudWatch 主控台建立跨帳戶儀表板，您必須啟用跨帳戶功能的主控台。如需詳細資訊，請參閱 跨帳戶跨 CloudWatch 區域主控台。

透過 AWS Management Console建立和使用的跨帳戶跨區域儀表板

您可以使用建 AWS Management Console 立跨帳戶跨區域儀表板。

若要建立跨帳戶跨區域儀表板

1. 登入至監控帳戶。
2. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
3. 在導覽窗格中，選擇 Dashboards (儀表板)。
4. 選擇儀表板，或建立新儀表板。
5. 在螢幕上方，您可以在帳戶和區域之間切換。當您建立儀表板時，您可以包含來自多個帳戶和區域的小工具。Widget 包括圖表、警示和 CloudWatch 記錄見解 Widget。

使用來自不同帳戶和區域的指標建立圖形

1. 登入至監控帳戶。
2. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
3. 在導覽窗格中，選擇 Metrics (指標)，然後選擇 All metrics (所有指標)。
4. 選擇您要新增指標的帳戶和區域。您可以從畫面右上方附近的帳戶和區域下拉式選單中選擇您的帳戶和區域。
5. 將您想要的指標新增至圖形。如需詳細資訊，請參閱 建立指標圖形。
6. 重複步驟 4-5，新增來自其他帳戶和區域的指標。

7. (選擇性) 選擇 Graphed metrics (圖形指標) 標籤，並新增使用您所選指標的指標數學函數。如需詳細資訊，請參閱 使用指標數學。

   您也可以設定單一圖形以包含多個 SEARCH 函數。每個搜尋都可以參照不同的帳戶或區域。
8. 當您完成圖形時，請選擇 Actions (動作)、Add to dashboard (新增至儀表板)。

   選取您的跨帳戶儀表板，然後選擇 Add to dashboard (新增至儀表板)。

將不同帳戶的警示新增至跨帳戶儀表板

1. 登入至監控帳戶。
2. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
3. 在頁面頂端，選擇警示所在的帳戶。
4. 在導覽窗格中，選擇警示。
5. 選取您要新增之警示旁的核取方塊，然後選擇 Add to dashboard (新增至儀表板)。
6. 選取您要新增至的跨帳戶儀表板，然後選擇 Add to dashboard (新增至儀表板)。

以程式設計方式建立跨帳戶跨區域儀表板

您可以使用 AWS API 和 SDK 以程式設計方式建立儀表板。如需詳細資訊，請參閱 PutDashboard。

為啟用跨帳戶跨區域儀表板，我們已將新參數新增至儀表板主體結構，如下表和範例所示。如需有關整體儀表板主體結構的詳細資訊，請參閱儀表板主體結構和語法。

參數	使用	範圍	預設
`accountId`	指定小工具或指標所在的帳戶 ID。	小工具或指標	目前已登入的帳戶
`region`	指定指標的區域。	小工具或指標	目前在主控台中選取的區域

下列範例說明跨帳戶跨區域儀表板中小工具的 JSON 來源。

此範例會在小工具層級將 `accountId` 欄位設為共用帳戶的 ID。這會指定此小工具中的所有指標將來自該共用帳戶和區域。

```
{
    "widgets": [
        {
            ...
            "properties": {
              "metrics": [
                     …..
                     ],
                    "accountId": "111122223333",
                    "region": "us-east-1"
           }
       }
  ]
}
```

此範例會在每個指標層級以不同方式設定 accountId 欄位。在此範例中，此指標數學運算式中的不同指標來自不同的共用帳戶和不同的區域。

```
{
"widgets": [
    {
        ...
        "properties": {
            "metrics": [
                [ { "expression": "SUM(METRICS())", "label": "[avg: ${AVG}]
 Expression1", "id": "e1", "stat": "Sum" } ],
                [ "AWS/EC2", "CPUUtilization", { "id": "m2", "accountId":
 "5555666677778888", "region": "us-east-1", "label": "[avg: ${AVG}] ApplicationALabel
 " } ],
                [ ".", ".", { "id": "m1", "accountId": "9999000011112222", "region":
 "eu-west-1", "label": "[avg: ${AVG}] ApplicationBLabel" } ]
            ],
            "view": "timeSeries",
            "region":"us-east-1", ---> home region of the metric. Not present in above
 example
            "stacked": false,
            "stat": "Sum",
            "period": 300,
            "title": "Cross account example"
        }
    }
]
```

```
}
```

此範例顯示了一個警示小工具。

```
{
    "type": "metric",
    "x": 6,
    "y": 0,
    "width": 6,
    "height": 6,
    "properties": {
        "accountID": "111122223333",
        "title": "over50",
        "annotations": {
            "alarms": [
                "arn:aws:cloudwatch:us-east-1:379642911888:alarm:over50"
            ]
        },
        "view": "timeSeries",
        "stacked": false
    }
}
```

此範例適用於 CloudWatch 日誌見解小器具。

```
{
    "type": "log",
    "x": 0,
    "y": 6,
    "width": 24,
    "height": 6,
    "properties": {
        "query": "SOURCE 'route53test' | fields @timestamp, @message\n| sort @timestamp
 desc\n| limit 20",
        "accountId": "111122223333",
        "region": "us-east-1",
        "stacked": false,
        "view": "table"
    }
}
```

另一種以程式設計方式建立儀表板的方法是先在中建立儀表板 AWS Management Console，然後複製此儀表板的 JSON 來源。若要執行此操作，請載入儀表板並選擇 Actions (動作)、View/edit source (檢視/編輯來源)。然後，您可以複製此儀表板 JSON 作為範本，以建立類似的儀表板。

使用儀表板變數建立彈性儀表板

使用儀表板變數建立彈性儀表板，可根據儀表板中輸入欄位的值，在多個小工具中快速顯示不同的內容。例如，您可以建立一個儀表板，以便在不同的 Lambda 函數或 Amazon EC2 執行個體 ID 之間快速切換，或切換至不同 AWS 區域的執行個體 ID。

建立使用變數的儀表板後，您可以將相同的變數模式複製至其他現有儀表板。

使用儀表板變數可為使用您儀表板的人員，改善操作工作流程。這也可以降低成本，因為您在單一儀表板中使用儀表板變數，而不是建立多個類似的儀表板。

> **Note**
>
> 如果您共用包含儀表板變數的儀表板，您與之共用的人員會無法在變數值之間變更。

儀表板變數的類型

儀表板變數可為屬性變數或模式變數。

- 屬性變數會變更儀表板內所有小工具中屬性的所有執行個體。此屬性可以是儀表板 JSON 來源中的任何 JSON 屬性，例如 `region`。或者，可以是指標的維度名稱，例如 `InstanceID` 或 `FunctionName`。

  如需使用屬性變數的教學課程，請參閱教學課程：建立以函數名稱為變數的 Lambda 儀表板。

  如需儀表板 JSON 來源的詳細資訊，請參閱儀表板內文結構及語法。在 CloudWatch 控制台中，您可以通過選擇操作，查看/編輯源來查看任何自定義儀表板的 JSON 源。
- 模式變數使用規則運算式模式，變更 JSON 屬性的所有部分或僅變更其特定部分。

  如需使用模式變數的教學課程，請參閱教學課程：建立使用規則運算式模式在區域之間切換的儀表板。

屬性變數適用於大多數使用案例，且設定較不複雜。

主題

- 教學課程：建立以函數名稱為變數的 Lambda 儀表板
- 教學課程：建立使用規則運算式模式在區域之間切換的儀表板
- 將變數複製至另一個儀表板

教學課程：建立以函數名稱為變數的 Lambda 儀表板

此程序中的步驟說明如何使用屬性變數，建立顯示各種指標圖形的彈性儀表板。這包括儀表板上的下拉式選取方塊，可用來在不同 Lambda 函數之間切換所有圖形中的指標。

此類型儀表板的其他使用案例範例包括將 `InstanceId` 當作變數，建立具有執行個體 ID 下拉式清單的指標儀表板。或者，您也可以建立將 `region` 當作變數的儀表板，顯示來自不同區域的相同指標集。

使用儀表板屬性變數建立彈性 Lambda 儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)、Create dashboard (建立儀表板)。
3. 輸入儀表板的名稱，然後選擇建立儀表板。
4. 將小工具新增至顯示 Lambda 函數指標的儀表板。建立這些小工具時，請為小工具指標指定 Lambda、依照函數名稱。針對該函數，請指定您要包含在此儀表板中的任一 Lambda 函數。

   如需將小工具新增至儀表板的詳細資訊，請參閱在 CloudWatch 儀表板上建立和使用小器具。
5. 新增小工具之後，檢視儀表板時，請依序選擇動作、變數、建立變數。
6. 選擇屬性變數。
7. 對於變數變更的屬性，請選擇FunctionName。
8. 針對輸入類型，以此使用案例而言，我們建議選擇選擇功能表 (下拉式清單)。這會在儀表板中建立下拉式功能表，您可以在其中選取要顯示指標的 Lambda 函數名稱。

   如果這是針對只在兩個或三個不同值之間切換之變數的儀表板，選項按鈕會是好選擇。

   如果您想要為變數輸入或貼上值，請選擇文字輸入。此選項不包含下拉式清單或選項按鈕。
9. 選擇選擇功能表 (下拉式清單) 時，您必須選擇輸入值或使用指標搜尋，填入功能表。針對此使用案例，假設您有大量 Lambda 函數，且不想手動輸入所有函數。選擇使用指標搜尋的結果，然後執行下列動作：

   a. 依序選擇預先建立的查詢、Lambda、錯誤。

(選擇「錯誤」並不會將「錯誤」量度新增至儀表板。 但是，它會快速填入FunctionName變數選取方塊。)

b. 選擇依照函數名稱，然後選擇搜尋。

在「搜索」按鈕下，您將看到「FunctionName已選」。您也會看到有關找到多少FunctionName標註值來填入輸入方塊的訊息。

10. (選用) 如需更多設定，請選擇次要設定，然後執行下列一或多個動作：

- 若要自訂變數的名稱，請在自訂變數名稱中輸入名稱。
- 若要自訂變數輸入欄位的標籤，請在輸入標籤中輸入標籤。
- 若要在第一次開啟儀表板時設定此變數的預設值，請在預設值中輸入預設值。

11. 選擇新增變數。

儀表板頂部附近會出現一個FunctionName下拉式選取方塊。您可以在此方塊中選取 Lambda 函數，使用該變數的所有小工具都會顯示所選函數的相關資訊。

稍後，如果您將更多小器具新增至儀表板以監視 Lambda 指標及FunctionName維度，它們將自動使用該變數。

教學課程：建立使用規則運算式模式在區域之間切換的儀表板

此程序中的步驟說明如何建立可在區域之間切換的彈性儀表板。本教學課程使用規則運算式模式變數，而不是屬性變數。如需使用屬性變數的教學課程，請參閱教學課程：建立以函數名稱為變數的 Lambda 儀表板。

針對許多使用案例，您可以使用屬性變數建立儀表板，在區域之間切換。但如果小工具依賴包含區域名稱的 Amazon Resource Names (ARN)，您必須使用模式變數，變更 ARN 中的區域名稱。

使用儀表板模式變數建立彈性多區域儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)、Create dashboard (建立儀表板)。
3. 輸入儀表板的名稱，然後選擇建立儀表板。
4. 將小工具新增至儀表板。新增想要顯示區域專屬資料的小工具時，請避免使用只顯示在一個區域中的值來指定任何維度。例如，對於 Amazon EC2 指標，請指定彙總的指標，而不是將 InstanceID 當作維度的指標。

如需將小工具新增至儀表板的詳細資訊，請參閱在 CloudWatch 儀表板上建立和使用小器具。

5. 新增小工具之後，檢視儀表板時，請依序選擇動作、變數、建立變數。
6. 選擇模式變數。
7. 針對變數變更的屬性，輸入目前儀表板區域的名稱，例如 **us-east-2**。

   如果該方塊下方的標籤顯示會受該變數影響的小工具，則您輸入了正確的區域。
8. 針對輸入類型，以此使用案例而言，請選取選項按鈕。
9. 針對定義輸入的填入方式，請選擇建立自訂值清單。
10. 針對建立您的自訂值，輸入您要切換的區域，每行一個區域。在每個區域之後，輸入英文逗號，然後輸入要針對該選項按鈕顯示的標籤。例如：

    ```
    us-east-1, N. Virginia

    us-east-2, Ohio

    eu-west-3, Paris
    ```

    填入自訂值時，預覽窗格會更新以顯示選項按鈕的外觀。
11. (選用) 如需更多設定，請選擇次要設定，然後執行下列一或多個動作：

    - 若要自訂變數的名稱，請在自訂變數名稱中輸入名稱。
    - 若要自訂變數輸入欄位的標籤，請在輸入標籤中輸入標籤。針對本教學，輸入 **Region:**。

      如果您在此輸入值，預覽窗格會更新以顯示選項按鈕的外觀。
    - 若要在第一次開啟儀表板時設定此變數的預設值，請在預設值中輸入預設值。
12. 選擇新增變數。

    儀表板隨即顯示，並在靠近頂端的區域選項按鈕旁顯示區域：標籤。在區域之間切換時，使用該變數的所有小工具都會顯示所選區域的相關資訊。

將變數複製至另一個儀表板

使用實用變數建立儀表板後，您可以將這些變數複製至其他現有的儀表板。如需儀表板變數的詳細資訊，請參閱使用儀表板變數建立彈性儀表板。

將儀表板變數複製至其他儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇儀表板，然後選擇儀表板中包含您要複製之變數的儀表板名稱。視需要輸入字串以尋找具有相符名稱的儀表板。
3. 依序選擇動作、變數、管理變數。
4. 選擇您要複製之變數旁的選項按鈕，然後選擇複製至另一個儀表板。
5. 選擇選取方塊，然後開始輸入要將變數複製至其中的儀表板名稱。
6. 選取儀表板名稱，然後選擇複製變數。

在 CloudWatch 儀表板上建立和使用小器具

使用本節中的主題來建立並處理儀表板中的圖形、警示和文字小工具。

目錄

- 從 CloudWatch 儀表板新增或移除圖形
- 在 CloudWatch 儀表板上手動繪製指標
- 使用現有圖形
- 將指標總管小器具新增至 CloudWatch 儀表板
- 在 CloudWatch 儀表板上新增或移除線條小器具
- 從 CloudWatch 儀表板中新增或移除數字 Widget
- 從 CloudWatch 儀表板新增或移除量測計小器具
- 將自訂小器具新增至 CloudWatch 儀表板
- 從 CloudWatch 儀表板中新增或移除文字 Widget
- 從 CloudWatch 儀表板新增或移除警報小工具
- 從 CloudWatch 儀表板新增或移除資料表小器具
- 在儀表板上連結和取消連結圖形 CloudWatch

從 CloudWatch 儀表板新增或移除圖形

您可以將包含一或多個量度的圖形新增至 CloudWatch 儀表板。您可以新增至儀表板的圖表類型包括 Line (線條)、Stacked area (堆疊區域)、Number (數字)、Gauge (量測計)、Bar (長條)，以及 Pie (圓

形)。當您不再需要圖表時，您可以將其從儀表板中移除。本節中的程序介紹如何在儀表板中新增和移除圖表。有關如何在儀表板上編輯圖形的資訊，請參閱編輯 CloudWatch 儀表板上的圖形。

若要將圖形新增到儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 選擇 + 符號，然後選擇您要新增至儀表板的圖形類型，然後選擇下一步。
   - 如果選取 Line (線條)、Stacked area (堆疊區域)、Bar (長條) 或 Pie (圓形)，請選擇 Metrics (指標)。
4. 在瀏覽索引標籤中，搜尋或瀏覽要以圖形表示的指標，然後選取您需要的指標。
5. (選用) 若要變更圖表的時間範圍，請選取畫面上方預先定義的時間範圍之一。時間範圍為 1 小時到 1 週 (1h (1 小時)、3h (3 小時)、12h (12 小時)、1d (1 天)、3d (3 天) 或 1w (1 週))。

   若要設定自己的時間範圍，請選擇 Custom (自訂)。
   - (選用) 若要在稍後變更儀表板其餘部分的時間範圍，讓此小工具仍可繼續使用您選取的此時間範圍，請選擇持續時間範圍。
6. (選用) 若要變更圖表的小工具類型，請使用預先定義時間範圍旁的下拉式選單。
7. (選用) 在 Graphed metrics (圖表化指標) 中，您可以將動態標籤新增至指標，並變更指標的標籤、標籤顏色、統計資料和週期。您還可以決定在 Y 軸上從左到右標籤的位置。
   a. 若要新增動態標籤，請選擇 Graphed metrics (圖表化指標)，然後選擇 Add dynamic labels (新增動態標籤)。動態標籤會在圖表圖例中顯示有關指標的統計資料。當儀表板或圖表重新整理時，動態標籤會自動更新。根據預設，您新增至標籤的動態值會出現在標籤的開頭。如需詳細資訊，請參閱 使用動態標籤。
   b. 若要變更指標的顏色，請選擇指標旁邊的顏色方塊。
   c. 若要變更統計資料，請選取 Statistic (統計資料) 下的下拉式選單，然後選擇新值。如需詳細資訊，請參閱統計資料。
   d. 若要變更週期，請選取 Period (週期) 資料欄下的下拉式選單，然後選擇新值。
8. 如果您要建立量測計小工具，則必須選擇選項索引標籤，並指定要用於量測計兩端的最小值和最大值。
9. (選用) 若要自訂 Y 軸，請選擇 Options (選項)。您可以在標籤欄位中的 Left Y Axis (左 Y 軸) 下新增自訂標籤。若圖表顯示 Y 軸右側的值，您也可以自訂該標籤。您也可以設定 Y 軸值的最小值和最大值，如此圖表只會顯示您指定的值範圍。

10. (選用) 若要新增或編輯線條或堆疊區域圖的水平註釋，或將閾值新增到量測計小工具，請選擇選項：

    a. 若要新增水平註釋或閾值，請選擇新增水平註釋或新增閾值。
    b. 在標籤中，輸入註釋的標籤，然後選擇核取記號圖示。
    c. 針對 Value (值)，請選擇目前值旁邊的筆和紙圖示，然後輸入新的值。輸入值後，請選擇核取標記圖示。
    d. 針對 Fill (填充)，請選取下拉式選單，然後指定註釋如何使用著色。您可以選擇 None (無)、Above (上方)、Between (之間) 或 Below (下方)。若要變更填充顏色，請選擇註釋旁的顏色方塊。
    e. 針對 Axis (軸)，請指定註釋是否顯示在 Y 軸的左側或右側。
    f. 若要隱藏註釋，請清除在您要隱藏的註釋旁邊的核取方塊。
    g. 若要刪除註釋，請選擇 Actions (動作) 下的 X。

    > **Note**
    >
    > 您可以重複這些步驟，將多個水平註釋或閾值新增到同一個圖形或量測計。

11. (選用) 若要新增或編輯垂直註釋，請選擇 Options (選項)：

    a. 若要新增垂直標註，請選擇 Add vertical annotation (新增垂直標註)。
    b. 針對 Label (標籤)，請選擇目前註釋旁邊的筆和紙圖示，然後輸入新的註釋。若要僅顯示日期與時間，請將 Label (標籤) 欄位留白。
    c. 針對 Date (日期)，選取目前的日期和時間，然後輸入新的日期和時間。
    d. 針對 Fill (填充)，選取下拉式選單，然後指定註釋如何使用著色。您可以選擇 None (無)、Above (上方)、Between (之間) 或 Below (下方)。若要變更填充顏色，請選取註釋旁的顏色方塊。
    e. 若要隱藏註釋，請清除您要隱藏的註釋旁邊的核取方塊。
    f. 若要刪除註釋，請選擇 Actions (動作) 下的 X。

    > **Note**
    >
    > 您可以重複這些步驟，將多個垂直註釋新增到同一個圖形。

12. 選擇 Create widget (建立 widget)。

13. 選擇 Save dashboard (儲存儀表板)。

從儀表板移除圖形

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 在您要移除的圖表的右上角，選擇 Widget actions (小工具動作)，然後選擇 Delete (刪除)。
4. 選擇 Save dashboard (儲存儀表板)。

在 CloudWatch 儀表板上手動繪製指標

如果指標過去 14 天未發佈資料，則在搜尋要新增至 CloudWatch 儀表板圖形的量度時，您將找不到該資料。使用下列步驟，將任何指標手動新增至現有圖形。

將您無法在搜尋中找到的指標新增到圖形

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 儀表板必須已包含新增指標的圖形。如果沒有，請建立圖形並將任何指標新增到它。如需詳細資訊，請參閱 從 CloudWatch 儀表板新增或移除圖形 。
4. 選擇 Actions (動作)、View/edit sources (檢視/編輯來源)。

   出現 JSON 區塊。此區塊指定儀表板上的 Widget 及其內容。以下範例是此區塊的一部分，其中定義一個圖形。

```
{
            "type": "metric",
            "x": 0,
            "y": 0,
            "width": 6,
            "height": 3,
            "properties": {
                "view": "singleValue",
                "metrics": [
                    [ "AWS/EBS", "VolumeReadOps", "VolumeId",
 "vol-1234567890abcdef0" ]
                ],
                "region": "us-west-1"
```

```
            }
        },
```

在此範例中，下一節會定義此圖形上顯示的指標。

```
[ "AWS/EBS", "VolumeReadOps", "VolumeId", "vol-1234567890abcdef0" ]
```

5. 如果右括號後面沒有逗號，請加上逗號，然後在逗號之後新增類似的方括號區段。在這個新區段中，針對您要新增至圖形的指標，指定命名空間、指標名稱及任何需要的維度。以下是範例。

```
[ "AWS/EBS", "VolumeReadOps", "VolumeId", "vol-1234567890abcdef0" ],
[ "MyNamespace", "MyMetricName", "DimensionName", "DimensionValue" ]
```

如需以 JSON 將指標格式化的詳細資訊，請參閱指標 Widget 物件的屬性。

6. 選擇更新。

使用現有圖形

請遵循這些章節中的程序來編輯和修改現有的儀表板圖形小工具。

主題

- 在 CloudWatch 儀表板上編輯圖形
- 在 CloudWatch 儀表板上移動圖形或調整圖形大小
- 重新命名 CloudWatch 儀表板上的圖形

在 CloudWatch 儀表板上編輯圖形

您可以編輯新增至 CloudWatch 控制面板的圖形。您可以變更圖形的標題、統計數字或週期。您還可以在圖表中新增、更新和移除指標。如果圖表包含多個指標，您可以隱藏未使用的指標，以減少混亂。本節中的程序介紹如何在儀表板上編輯圖表。如需有關建立圖形的資訊，請參閱從 CloudWatch儀表板新增或移除圖形。

New interface

若要在儀表板上編輯圖形

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/

2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 在您要編輯的圖表的右上角，選擇 Widget actions (小工具動作)，然後選擇 Edit (編輯)。
4. 若要變更圖表的標題，請選擇目前標題旁的筆和紙圖示。輸入新標題，然後選擇 Apply (套用)。
5. (選用) 若要變更圖表的時間範圍，請選取圖表上方區域中預先定義的時間範圍之一。時間範圍為 1 小時到 1 週 (1h (1 小時)、3h (3 小時)、12h (12 小時)、1d (1 天)、3d (3 天) 或 1w (1 週))。

   若要設定自己的時間範圍，請選擇 Custom (自訂)。

   - (選用) 若要在稍後變更儀表板其餘部分的時間範圍，讓此小工具仍可繼續使用您選取的此時間範圍，請選擇持續時間範圍。
6. (選用) 若要變更圖形的小工具類型，請使用預先定義時間範圍旁的下拉式選單。
7. 在 Graphed metrics (圖表化指標) 中，您可以將動態標籤新增至指標，並變更指標的標籤、標籤顏色、統計資料和週期。您還可以決定在 Y 軸上從左到右標籤的位置。

   a. 若要為指標新增動態標籤，請選擇 Dynamic labels (動態標籤)。動態標籤會在圖表圖例中顯示有關指標的統計資料。當儀表板或圖表重新整理時，動態標籤會自動更新。根據預設，您新增至標籤的動態值會出現在標籤的開頭。如需詳細資訊，請參閱 使用動態標籤。

   b. 若要變更指標的顏色，請選擇指標旁邊的顏色方塊。

   c. 若要變更統計資料，請選擇 Statistic (統計資料) 資料欄下的統計資料值，然後選擇新值。如需詳細資訊，請參閱 統計資料。

   d. 若要變更週期，請選擇 Period (週期) 資料欄下的週期值，然後選擇新值。
8. 若要新增或編輯水平註釋，請選擇 Options (選項)：

   a. 若要新增水平標註，請選擇 Add horizontal annotation (新增水平標註)。

   b. 針對 Label (標籤)，請選擇目前批註旁邊的筆和紙圖示。然後輸入新註釋。輸入註釋後，請選擇核取標記圖示。

   c. 針對 Value (值)，請選擇目前指標值旁邊的筆和紙圖示。然後輸入新指標值。輸入值後，請選擇核取標記圖示。

   d. 針對 Fill (填寫)，請選擇欄下的下拉式選單，然後指定註釋如何使用著色。您可以選擇 None (無)、Above (上方)、Between (之間) 或 Below (下方)。如果選擇 Between (之間)，則會出現另一個新的標籤和值欄位。

> **Tip**
>
> 您可以選擇註釋旁的彩色方塊，以變更填充顏色。

e. 針對 Axis (軸)，請指定註釋是否顯示在 Y 軸的左側或右側。

f. 若要隱藏註釋，請取消選中要在圖形上隱藏的註釋旁邊的核取方塊。

g. 若要刪除註釋，請選擇 Actions (動作) 欄下的 X 。

> **Note**
>
> 您可以重複這些步驟，將多個水平註釋新增到同一個圖形。

9. 若要新增或編輯垂直註釋，請選擇 Options (選項)：

a. 若要新增垂直標註，請選擇 Add vertical annotation (新增垂直標註)。

b. 針對 Label (標籤)，請選擇目前批註旁邊的筆和紙圖示。然後輸入新註釋。輸入註釋後，請選擇核取標記圖示。

> **Tip**
>
> 若要僅顯示日期與時間，請將 Label (標籤) 欄位留白。

c. 針對 Date (日期)，請選擇目前的日期與時間。然後輸入新的日期與時間。

d. 針對 Fill (填寫)，請選擇欄下的下拉式選單，然後指定註釋如何使用著色。您可以選擇 None (無)、Above (上方)、Between (之間) 或 Below (下方)。如果選擇 Between (之間)，則會出現新的標籤和值欄位。

> **Tip**
>
> 您可以選擇註釋旁的彩色方塊，以變更填充顏色。

> **Note**
>
> 您可以重複這些步驟，將多個垂直註釋新增到同一個圖形。

e. 若要隱藏註釋，請取消選中要在圖形上隱藏的註釋旁邊的核取方塊。

f. 若要刪除註釋，請選擇 Actions (動作) 欄下的 X 。

10. 若要自訂 Y 軸，請選擇 Options (選項)。您可以在 Left Y Axis (左 Y 軸) 下的 Label (標籤) 中輸入自訂標籤。若圖表顯示右側 Y 軸的值，您也可以自訂該標籤。您也可以設定 Y 軸值的最小值和最大值，如此圖表只會顯示您指定的值範圍。
11. 完成變更之後，請選擇 Update widget (更新小工具)。

隱藏或變更圖形圖例的位置

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 在您要編輯的圖表的右上角，選擇 Widget actions (小工具動作)。選擇 Legend (圖例)，然後選取 Hidden (隱藏)、Bottom (底部) 或 Right (右側)。

若要暫時隱藏儀表板上圖形的指標

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 針對您要在圖表頁尾隱藏的指標選擇顏色方塊。將滑鼠停留在顏色方塊上時，X 即會出現，當您選中時方塊會變為灰色。
4. 若要還原隱藏的指標，請清除灰色方形中的 X。

Original interface

若要在儀表板上編輯圖形

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 將滑鼠停留在您要編輯的圖表的右上角。選擇 Widget actions (小工具動作)，然後選擇 Edit (編輯)。
4. 若要變更圖表的標題，請選擇目前標題旁的筆和紙圖示，然後輸入新的標題。
5. 若要變更圖表的時間範圍，請選擇圖表上方區域中預先定義的時間範圍之一。範圍為 1 小時到 1 週 (1 小時、3 小時、12 小時、1 天、3 天或 1 週。

   - 若要設定自己的時間範圍，請選擇 Custom (自訂)。

6. 若要變更圖形的小工具類型，請選擇 Graph options (圖形選項) 索引標籤。您可以選擇 Line (線條)、Stacked area (堆疊區域)、Number (數字)、Bar (長條) 或 Pie (圓形)。

   > **Tip**
   >
   > 您還可以選擇預先定義時間範圍旁邊的下拉式選單，來變更圖表的小工具類型。

7. 在 Graphed metrics (圖表化指標) 中，您可以將動態標籤新增至指標，並變更指標的標籤、標籤顏色、統計資料和週期。您還可以決定在 Y 軸上從左到右標籤的位置。

   a. 若要為指標新增動態標籤，請選擇 Dynamic labels (動態標籤)。動態標籤會在圖表圖例中顯示有關指標的統計資料。當儀表板或圖表重新整理時，動態標籤會自動更新。根據預設，您新增至標籤的動態值會出現在標籤的開頭。如需詳細資訊，請參閱 使用動態標籤。

   b. 若要變更指標的顏色，請選擇指標旁邊的顏色方塊。

   c. 若要變更統計資料，請選擇 Statistic (統計資料) 資料欄下的統計資料值，然後選擇新值。如需詳細資訊，請參閱 統計資料。

   d. 若要變更週期，請選擇 Period (週期) 資料欄下的週期值，然後選擇新值。

8. 若要新增或編輯水平註釋，選擇 Graph options (圖形選項)：

   a. 若要新增水平標註，請選擇 Add horizontal annotation (新增水平標註)。

   b. 針對 Label (標籤)，請選擇目前批註旁邊的鉛筆圖示。然後輸入新註釋。輸入註釋後，請選擇核取標記圖示。

   c. 針對 Value (值)，請選擇目前指標值旁邊的鉛筆圖示。然後輸入新指標值。輸入值後，請選擇核取標記圖示。

   d. 針對 Fill (填寫)，請選擇欄下的下拉式選單，然後指定註釋如何使用著色。您可以選擇 None (無)、Above (上方)、Between (之間) 或 Below (下方)。如果選擇 Between (之間)，則會出現新的標籤和值欄位。

      > **Tip**
      >
      > 您可以選擇註釋旁的彩色方塊，以變更填充顏色。

   e. 針對 Axis (軸)，請指定註釋是否顯示在 Y 軸的左側或右側。

   f. 若要隱藏註釋，請取消選中要在圖形上隱藏的註釋旁邊的核取方塊。

   g. 若要刪除註釋，請選擇 Actions (動作) 欄下的 X 。

> **Note**
>
> 您可以重複這些步驟，將多個水平註釋新增到同一個圖形。

9. 若要新增或編輯垂直註釋，請選擇 Graph options (圖形選項)：

   a. 若要新增垂直標註，請選擇 Add vertical annotation (新增垂直標註)。

   b. 針對 Label (標籤)，請選擇目前批註旁邊的鉛筆圖示。然後輸入新註釋。輸入註釋後，請選擇核取標記圖示。

   > **Tip**
   >
   > 若要僅顯示日期與時間，請將 Label (標籤) 欄位留白。

   c. 針對 Date (日期)，請選擇目前日期與時間旁的鉛筆圖示。然後輸入新的日期與時間。

   d. 針對 Fill (填寫)，請選擇欄下的下拉式選單，然後指定註釋如何使用著色。您可以選擇 None (無)、Above (上方)、Between (之間) 或 Below (下方)。如果選擇 Between (之間)，則會出現新的標籤和值欄位。

   > **Tip**
   >
   > 您可以選擇註釋旁的彩色方塊，以變更填充顏色。

   > **Note**
   >
   > 您可以重複這些步驟，將多個垂直註釋新增到同一個圖形。

   e. 若要隱藏註釋，請取消選中要在圖形上隱藏的註釋旁邊的核取方塊。

   f. 若要刪除註釋，請選擇 Actions (動作) 欄下的 X 。

10. 若要自訂 Y 軸，選擇 Graph options (圖形選項)。您可以在 Left Y Axis (左 Y 軸) 下的 Label (標籤) 中輸入自訂標籤。若圖表顯示右側 Y 軸的值，您也可以自訂該標籤。您也可以設定 Y 軸值的最小值和最大值，如此圖表只會顯示您指定的值範圍。

11. 完成變更之後，請選擇 Update widget (更新小工具)。

隱藏或變更圖形圖例的位置

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 將滑鼠停留在您要編輯的圖表的右上角，然後選擇 Widget actions (小工具動作)。選擇 Legend (圖例)，然後選取 Hidden (隱藏)、Bottom (底部) 或 Right (右側)。

若要暫時隱藏儀表板上圖形的指標

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 針對您要在圖表頁尾隱藏的指標選擇顏色方塊。將滑鼠停留在顏色方塊上時，X 即會出現，當您選中時方塊會變為灰色。
4. 若要還原隱藏的指標，請清除灰色方形中的 X。

在 CloudWatch 儀表板上移動圖形或調整圖形大小

您可以在 CloudWatch 儀表板上排列和調整圖形大小。

若要在儀表板上移動圖形

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 執行以下任意一項：
   - 請將滑鼠游標移到圖形的標題，等待選擇圖示出現。選取圖形並將其拖曳到儀表板上的新位置。
   - 若要將小工具移至儀表板的左上方或左下方，請選擇小工具右上角的垂直省略符號以開啟小工具動作選單。接著選擇移動，並選擇要將 tbe 小工具移動到的位置。
4. 選擇 Save dashboard (儲存儀表板)。

若要調整圖形

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 為了增加或減少大小，請將滑鼠游標移到圖形上方，並拖曳圖形的右下角。當您獲得所需的大小時，請停止拖曳角落。

4. 選擇 Save dashboard (儲存儀表板)。

若要暫時放大圖形

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 選取圖形。或者，請將滑鼠游標移到圖形標題上方，然後選擇 Widget actions (Widget 動作)，Enlarge (放大)。

重新命名 CloudWatch 儀表板上的圖形

您可以變更指 CloudWatch 派給儀表板上圖形的預設名稱。

若要在儀表板上重新命名圖形

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 請將滑鼠游標移到圖形的標題，然後選擇 Widget actions (Widget 動作) 以及 Edit (編輯)。
4. 在 Edit graph (編輯圖形) 畫面上 (接近頂部)，選擇圖形標題。
5. 針對 Title (標題)，輸入新的名稱，然後選擇 Ok (確定) (核取記號)。在 Edit graph (編輯圖形) 畫面的右下角，請選擇 Update widget (更新 Widget)。

將指標總管小器具新增至 CloudWatch 儀表板

指標瀏覽器小工具包含具有相同標籤或共用相同資源屬性 (例如執行個體類型) 的多個資源圖形。這些小工具會保持最新狀態，因為會建立或刪除相符的資源。將指標瀏覽器小工具新增至儀表板可協助您更有效率地對環境進行故障排除。

例如，您可以透過指派表示其環境的標籤 (例如生產或測試) 來監控 EC2 執行個體機群。然後，您可以使用這些標籤來篩選和彙總操作指標，例如 `CPUUtilization`，以了解與每個標籤相關聯之 EC2 執行個體的運作狀態和效能。

下列步驟說明如何使用主控台將指標瀏覽器小工具新增至儀表板。您也可以通過編程或使用 AWS CloudFormation. 如需詳細資訊，請參閱度量總管 Widget 物件定義和 AWS::CloudWatch::Dashboard。

若要將指標瀏覽器小工具新增至儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)。
3. 選擇您要新增指標瀏覽器小工具的儀表板名稱。
4. 選擇 + 符號。
5. 選擇 Explorer，然後選擇 Next (下一步)。

> **Note**
>
> 您必須選擇加入新的儀表板檢視，才能新增指標瀏覽器小工具。若要選擇加入，請選擇導覽窗格中的 Dashboards (儀表板)，然後選擇頁面頂端的橫幅中的 try out the new interface (試用新的介面)。

6. 執行以下任意一項：
   - 若要使用範本，請選擇 Pre-filled Explorer widget (預先填寫的瀏覽器小工具)，然後選取要使用的範本。
   - 若要建立自訂視覺化，請選擇 Empty Explorer widget (空的瀏覽器小工具)。
7. 選擇建立。

   如果您使用範本，小工具會顯示在儀表板上，其中包含選取的指標。如果您對瀏覽器小工具和儀表板感到滿意，請選擇 Save dashboard (儲存儀表板)。

   如您未使用範本，請繼續以下步驟。
8. Explorer (瀏覽器) 下的新的小工具中，在 Metrics (指標) 方塊中，選擇單一指標或服務中的所有可用指標。

   選擇指標後，您可以選擇性地重複此步驟，以新增更多指標。
9. 針對每個選取的測量結果，會在測量結果名稱後立即 CloudWatch 顯示要使用的統計值。若要對其進行變更，請選擇統計資料名稱，然後選擇您要的統計資料。
10. 在 From (來源) 下，選擇標籤或資源屬性以篩選結果。

    執行這項操作之後，您可以選擇性地重複此步驟，以選擇更多標籤或資源特性。

    如果您選擇同一屬性的多個值 (例如兩個 EC2 執行個體類型)，則瀏覽器會顯示符合任一所選屬性的所有資源。它會被視為 OR 運算。

如果您選擇不同的屬性或標籤，例如 **Production** 標籤和 M5 執行個體類型時，只會顯示符合所有這些選項的資源。這會被視為 AND 運算。

11. (選用) 對於 Aggregate by (彙總依據)，選擇用來彙總指標的統計資料。然後，在 for (方式) 旁，從清單中選擇彙總指標的方式。您可以將目前顯示的所有資源彙總在一起，或依單一標籤或資源屬性彙總。

    根據您選擇聚總的方式，結果可能是單一時間序列或多個時間序列。

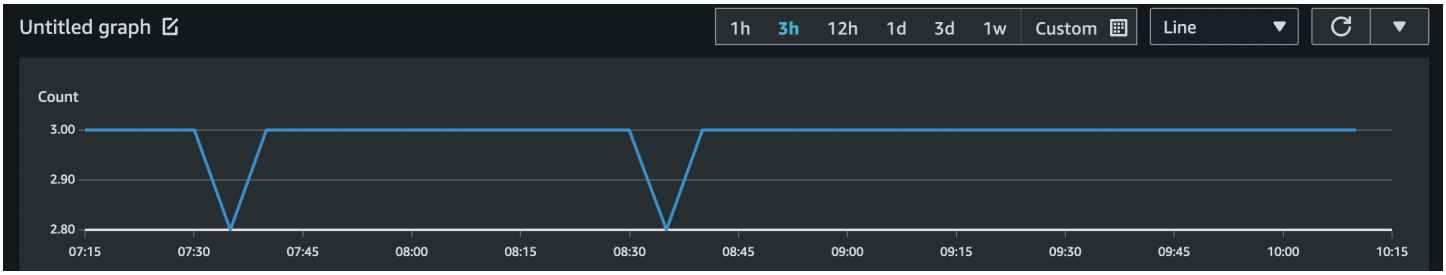
12. 在 Split by (分割依據) 下，您可以選擇將具有多個時間序列的單一圖形分割為多個圖形。分割可以透過各種標準進行，您可以在 Split by (分割依據) 下進行選擇。

13. 在 Graph options (圖形選項) 下，您可以透過變更週期、圖形類型、圖例位置和版面配置來精簡圖形。

14. 如果您對瀏覽器小工具和儀表板感到滿意，請選擇 Save dashboard (儲存儀表板)。

在 CloudWatch 儀表板上新增或移除線條小器具

您可以使用線條小工具，比較一段時間段內的指標。您還可以使用小工具的迷你地圖縮放功能檢查折線圖的部分，而無需在放大和縮小檢視之間進行變更。本節中的程序說明如何在儀表板上新增和移除線條 CloudWatch Widget。有關將小工具的迷你地圖縮放功能用於折線圖的資訊，請參閱放大折線圖或堆疊區域圖。



將線條小工具新增至儀表板

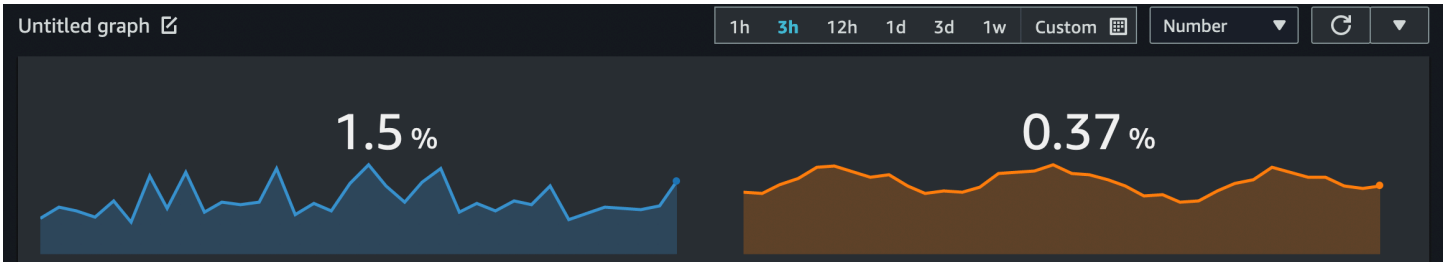
1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 選擇 + 符號，然後選擇 Line (行)。
4. 選擇 Metrics (指標)。
5. 選擇 Browse (瀏覽)，然後選取您要繪製圖表的指標。
6. 選擇 Create widget (建立小工具)，然後選擇 Save dashboard (儲存儀表板)。

從儀表板移除線條小工具

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 在您要移除的線條小工具的右上角，選擇 Widget actions (小工具動作)，然後選擇 Delete (刪除)。
4. 選擇 Save dashboard (儲存儀表板)。

從 CloudWatch 儀表板中新增或移除數字 Widget

您可以使用數字小工具，在最新的指標值和趨勢出現後立即進行查看。數字小工具包括走勢圖功能，因此您可以在單一圖表中視覺化指標趨勢的上半部分和下半部分。本節中的程序說明如何從儀表板新增和移除數字 CloudWatch Widget。



> **Note**
>
> 只有新介面支援走勢圖功能。建立數字小工具時，走勢圖功能將自動包括在內。

如要將數字小工具新增至儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 選擇 + 符號，然後選擇 Number (編號)。
4. 在瀏覽索引標籤中，搜尋或瀏覽您要顯示的指標。
5. (選用) 若要變更走勢圖功能的顏色，請選擇 Graphed metrics (圖表化指標)，然後選取指標標籤旁的顏色方塊。此時會顯示選單，您可以選擇其他顏色或輸入六位數的十六進位顏色代碼來指定顏色。
6. (選用) 若要關閉走勢圖功能，請選擇 Options (選項)。在 Sparkline (走勢圖) 下，對相應核取方塊執行相關動作。

7. (選用) 若要變更數字小工具的時間範圍，請選取小工具上方區域中預先定義的時間範圍。時間範圍為 1 小時到 1 週 (1h (1 小時)、3h (3 小時)、12h (12 小時)、1d (1 天)、3d (3 天) 或 1w (1 週))。

   若要設定自己的時間範圍，請選擇 Custom (自訂)。

   - (選用) 若要在稍後變更儀表板其餘部分的時間範圍，讓此小工具仍可繼續使用您選取的此時間範圍，請選擇持續時間範圍。

8. （可選）若要讓數字小器具顯示彙總 (1 小時、3 小時、12 小時、1d、3D 或 1 w)。

   若要設定自己的時間範圍，請選擇 Custom (自訂)。

   - (選擇性) 若要讓此 Widget 顯示整個時間範圍內的測量結果值平均值，而非最新值，請選擇選項，「時間範圍」值會顯示整個時間範圍的值。

9. 依序選擇 Create widget (建立小工具)、Save dashboard (儲存儀表板)。

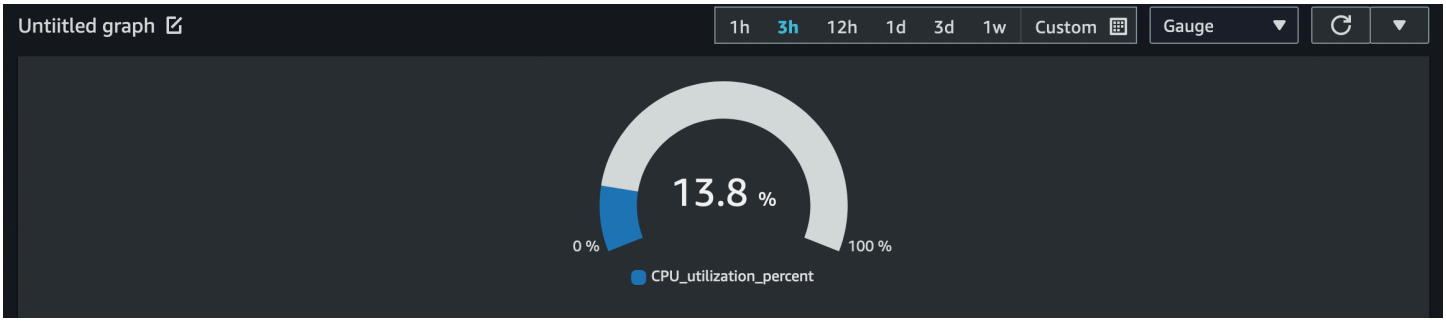
> **Tip**
>
> 您可以從儀表板畫面上的數字小工具關閉走勢圖功能。在您要修改的數字小工具的右上角，選擇 Widget actions (小工具動作)。選取 Sparkline (走勢圖)，然後選擇 Hide sparkline (隱藏走勢圖)。

如要從儀表板移除數字小工具

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇其中包含您要刪除的數字小工具的儀表板。
3. 在您要移除的數字小工具的右上角，選擇 Widget actions (小工具動作)，然後選擇 Delete (刪除)。
4. 選擇 Save dashboard (儲存儀表板)。

從 CloudWatch 儀表板新增或移除量測計小器具

您可以使用量測計小工具，視覺化範圍之間的指標值。例如，您可以使用量測計小工具繪製百分比和 CPU 使用率的圖表，以便觀察和診斷出現的任何效能問題。本節中的程序說明如何從儀表板新增和移除量測計 CloudWatch Widget。



> **Note**
>
> 只有主 CloudWatch 控台中的新介面支援建立量測計小器具。建立此小工具時，必須設定量測計範圍。

將量測計小工具新增至儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 在儀表板畫面中，選擇 + 符號，然後選取 Gauge (量測計)。
4. 選擇 Browse (瀏覽)，然後選取您要繪製圖表的指標。
5. 選擇 Options (選項)。在 Gauge range (量測計範圍) 下，設定 Min (最小值) 和 Max (最大值) 的值。針對百分比 (例如 CPU 使用率)，建議您將 `Min` 的值設為 `0`，將 `Max` 的值設為 `100`。
6. (選用) 若要變更量測計小工具的顏色，請選擇 Graphed metrics (圖表化指標)，然後選取指標標籤旁的顏色方塊。此時會顯示選單，您可以選擇其他顏色或輸入六位數的十六進位顏色代碼來指定顏色。
7. 依序選擇 Create widget (建立小工具)、Save dashboard (儲存儀表板)。

從儀表板移除量測計小工具

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇其中包含您要刪除的量測計小工具的儀表板。
3. 在您要刪除的量測計小工具的右上角，選擇 Widget actions (小工具動作)，然後選擇 Delete (刪除)。
4. 選擇 Save dashboard (儲存儀表板)。

將自訂小器具新增至 CloudWatch 儀表板

自定義小部件是一個 CloudWatch 儀表板小部件，可以調用具有自定義參數的任何 AWS Lambda 函數。然後顯示傳回的 HTML 或 JSON。自訂小工具是在儀表板上建置自訂資料檢視的簡單方法。如果您可以編寫 Lambda 程式碼並建立 HTML，則可以建立一個有用的自訂小工具。此外，Amazon 還提供了幾個預先建置的自訂小工具，您無需任何程式碼即可建立。

當您建立 Lambda 函數作為自訂小工具時，強烈建議您在函數名稱中納入字首 customWidget。這可協助您了解當您將自訂小工具新增至儀表板時，可安全使用哪些 Lambda 函數。

自訂小工具的行為就像儀表板上的其他小工具一樣。它們可以重新整理和自動重新整理、調整大小以及四處移動。他們會對儀表板的時間範圍做出反應。

如果您已設定 CloudWatch 主控台跨帳戶功能，則可以將在某個帳戶中建立的自訂 Widget 新增至其他帳戶的儀表板。如需詳細資訊，請參閱 跨帳戶跨 CloudWatch 區域主控台。

您也可以使用 CloudWatch 儀表板共用功能，在自己的網站上使用自訂 Widget。如需詳細資訊，請參閱 共用 CloudWatch 儀表板。

主題

- 自訂小工具的詳細資訊
- 安全性和 JavaScript
- 自訂小工具中的互動
- 建立自訂小工具
- 範例自訂小工具

自訂小工具的詳細資訊

自訂小工運作方式如下：

1. CloudWatch 儀表板會呼叫包含小器具程式碼的 Lambda 函數。它會傳遞在小工具中定義的任何自訂參數。
2. Lambda 函數會傳回 HTML、JSON 或 Markdown 的字串。Markdown 會以下列各式傳回為 JSON：

   ```
   {"markdown":"markdown content"}
   ```

3. 儀表板會顯示傳回的 HTML 或 JSON。

如果函數傳回 HTML，則會支援大多數 HTML 標籤。您可以使用 Cascading 樣式表 (CSS) 樣式和可擴展向量圖形 (SVG) 來建置複雜的檢視。

HTML 元素 (例如連結和表格) 的預設樣式遵循 CloudWatch 儀表板的樣式。您可以透過使用內嵌樣式和 <style> 標籤來自訂此樣式。您也可以透過包括具有 cwdb-no-default-styles 類別的單一 HTML 元素來停用預設樣式。下列範例會停用預設樣式： 。

每次透過自訂小工具對 Lambda 的呼叫都包含 widgetContext 元素，可提供 Lambda 函數開發人員有用的內容資訊。

```
{
    "widgetContext": {
        "dashboardName": "Name-of-current-dashboard",
        "widgetId": "widget-16",
        "accountId": "012345678901",
        "locale": "en",
        "timezone": {
            "label": "UTC",
            "offsetISO": "+00:00",
            "offsetInMinutes": 0
        },
        "period": 300,
        "isAutoPeriod": true,
        "timeRange": {
            "mode": "relative",
            "start": 1627236199729,
            "end": 1627322599729,
            "relativeStart": 86400012,
            "zoom": {
                "start": 1627276030434,
                "end": 1627282956521
             }
        },
        "theme": "light",
        "linkCharts": true,
        "title": "Tweets for Amazon website problem",
        "forms": {
            "all": {}
        },
        "params": {
            "original": "param-to-widget"
        },
```

```
        "width": 588,
        "height": 369
    }
}
```

預設 CSS 樣式

自訂小工具提供以下預設 CSS 樣式元素：

- 您可以使用 CSS 類別 btn 以新增按鈕。它將錨點 (<a>) 轉變為按鈕，如下列範例所示：

  ```
  <a class="btn" href=https://amazon.com”>Open Amazon</a>
  ```

- 您可以使用 CSS 類別 btn btn-primary 以新增主要按鈕。
- 依預設，下列為樣式元素：table、select、headers (h1, h2, and h3)、preformatted text (pre)、input 和 text area。

使用描述參數

強烈建議您支援 describe (描述) 參數，即使它只會傳回一個空字串。如果您不支援它，並且它在自訂小工具中被呼叫，它會像顯示文件那樣顯示小工具內容。

如果您包含 describe (描述) 參數時，Lambda 函數會傳回 Markdown 格式的文件，而且不會執行任何其他動作。

當您在主控台中建立自訂小工具時，在您選取 Lambda 函數之後，隨即便會出現 Get documentation (取得文件) 按鈕。如果您選擇此按鈕，則會使用 describe (描述) 參數，並傳回函數的文件。如果文檔在 markdown 中格式良好，則 CloudWatch 解析文檔中由三個單個反引號字符 (```) 包圍的 YAML 中的第一個條目。然後，它會自動填充參數中的文件。以下是格式良好的文件的範例。

```
``` yaml
echo: <h1>Hello world</h1>
```
```

安全性和 JavaScript

出於安全原因 JavaScript ，返回的 HTML 中不允許使用。移除可 JavaScript 防止權限提升問題，Lambda 函數的撰寫器會輸入程式碼，而這些程式碼可以比在儀表板上檢視 Widget 的使用者更高的權限執行。

如果返回的 HTML 包含任何 JavaScript 代碼或其他已知的安全漏洞，則在將其呈現在儀表板上之前從 HTML 中清除。例如，不允許 <iframe> 和 <use> 標籤，並且會將其移除。

自訂小工具預設不會在儀表板中執行。相反地，如果您信任其叫用的 Lambda 函數，則必須明確允許自訂小工具。您可以針對個別小工具和整個儀表板選擇允許一次或一律允許。您也可以拒絕個別小工具和整個儀表板的許可。

自訂小工具中的互動

即使 JavaScript 不允許，還有其他方法可以允許與返回的 HTML 進行交互。

- 在傳回的 HTML 中的任何元素都可以在 <cwdb-action> 標籤中使用特殊組態加上標籤，它可以在快顯中顯示資訊，要求點按確認，並在選擇該元素時呼叫任何 Lambda 函數。例如，您可以定義使用 Lambda 函數呼叫任何 AWS API 的按鈕。傳回的 HTML 可以設定為取代現有的 Lambda 小工具的內容，或在模態內顯示。
- 傳回的 HTML 可以包含開啟新主控台、開啟其他客戶頁面或載入其他儀表板的連結。
- HTML 可以包含元素的 `title` 屬性，如果使用者將滑鼠游標暫留在該元素上，將可獲得額外的資訊。
- 元素可以包含 CSS 選擇器，如 `:hover`，而它可以叫用動畫或其他 CSS 效果。您也可以顯示或隱藏頁面中的元素。

<cwdb-action> 定義和用法

<cwdb-action> 元素定義了緊接前一個元素的行為。<cwdb-action> 的內容是要顯示的 HTML 還是要傳遞給 Lambda 函數的參數 JSON 區塊。

以下 <cwdb-action> 元素的範例。

```
<cwdb-action
     action="call|html"
     confirmation="message"
     display="popup|widget"
     endpoint="<lambda ARN>"
     event="click|dblclick|mouseenter">

     html | params in JSON
</cwdb-action>
```

- action— 有效值為 `call` 和 `html`，其中前者會叫用一個 Lambda 函數，後者則會顯示包含在 `<cwdb-action>` 中的任何 HTML。預設為 `html`。
- 確認— 顯示必須在採取動作之前確認的確認訊息，允許客戶取消。
- 顯示— 有效值為 `popup` 和 `widget`，可取代小工具本身的內容。預設值為 `widget`。
- 端點— 要呼叫之 Lambda 函數的 Amazon Resource Name (ARN)。如果 `action` 為 `call`，則必須如此。
- 事件— 定義上一個叫用動作的元素上的事件。有效值為 `click`、`dblclick` 和 `mouseenter`。`mouseenter` 事件只能與 `html` 動作合併使用。預設值為 `click`。

範例

以下範例說明如何使用 `<cwdb-action>` 標籤建立按鈕，以使用 Lambda 函數呼叫來重新啟動 Amazon EC2 執行個體。它會在快顯中顯示呼叫的成功或失敗。

```
<a class="btn">Reboot Instance</a>
<cwdb-action action="call" endpoint="arn:aws:lambda:us-
east-1:123456:function:rebootInstance" display="popup">
       { "instanceId": "i-342389adbfef" }
</cwdb-action>
```

下一個範例會在快顯中顯示更多資訊。

```
<a>Click me for more info in popup</a>
<cwdb-action display="popup">
   <h1>Big title</h1>
   More info about <b>something important</b>.
</cwdb-action>
```

此範例為 Next (下一步) 按鈕，可用 Lambda 函數的呼叫取代小工具的內容。

```
<a class="btn btn-primary">Next</a>
<cwdb-action action="call" endpoint="arn:aws:lambda:us-
east-1:123456:function:nextPage">
   { "pageNum": 2 }
</cwdb-action>
```

建立自訂小工具

要創建自定義窗口小部件，您可以使用提供的樣本之一 AWS，也可以創建自己的樣本。這些示 AWS 例包括 Python JavaScript 和 Python 中的樣本，並且由 AWS CloudFormation 堆棧創建。如需範例清單，請參閱 範例自訂小工具。

若要在 CloudWatch 儀表板窗格中建立自訂小器具

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 選擇 + 符號。
4. 選擇 Custom widget (自訂小工具)。
5. 使用下列其中一種方法：
   - 若要使用提供的範例自訂 Widget AWS，請執行下列動作：
     a. 在下拉式方塊中選取範例。

        主 AWS CloudFormation 控台會在新的瀏覽器中啟動。在 AWS CloudFormation 主控台中，執行下列動作：
     b. (選擇性) 自訂 AWS CloudFormation 堆疊名稱。
     c. 選取範例所使用的任何參數。
     d. 選取 [我確認 AWS CloudFormation 可能會建立 IAM 資源]，然後選擇 [建立堆疊]。
   - 要創建自己的自定義小部件提供 AWS，請執行以下操作：
     a. 選擇下一步。
     b. 選擇是從清單中選取 Lambda 函數，還是輸入其 Amazon Resource Name (ARN)。如果您從清單中選取它，也請指定函數所在的區域以及要使用的版本。
     c. 對於 Parameters (參數)，針對函數使用的任何參數進行選取。
     d. 輸入小工具的標題。
     e. 對於 Update on (更新時間)，設定應該更新小工具的時間 (當應該再次呼叫 Lambda 函數時)。此選項可以為下列其中一或多個：Refresh (重新整理) 以在儀表板自動重新整理時進行更新，Resize (調整大小) 以在重新調整小工具時進行更新，或者 Time Range (時間範圍) 以在調整儀表板的時間範圍時 (包括放大圖形時) 進行更新。
     f. 如果您對預覽感到滿意，選擇 Create widget (建立小工具)。

範例自訂小工具

AWS 在兩個 JavaScript 和 Python 中提供了示例自定義部件。您可以使用此清單中每個小工具的連結來建立這些範例小工具。或者，您也可以使用 CloudWatch 主控台建立和自訂 Widget。此清單中的連結會開啟 AWS CloudFormation 主控台，並使用 AWS CloudFormation 快速建立連結來建立自訂 Widget。

您也可以在上存取自訂小器具範例GitHub。

在此清單之後，將顯示每種語言的 Echo 小工具的完整範例。

JavaScript

在示例自定義部件 JavaScript

- Echo – 基本迴聲器，您可以用來測試 HTML 在自訂小工具中的顯示方式，而不必撰寫新的小工具。
- Hello World – 一個非常基本的入門小工具。
- 自訂小工具除錯器 – 偵錯器小工具，可顯示 Lambda 執行階段環境的實用資訊。
- 查詢 CloudWatch 日誌洞察 — 運行和編輯 CloudWatch 日誌見解查詢。
- 執行 Amazon Athena 查詢 — 執行和編輯 Athena 查詢。
- 呼叫 AWS API — 呼叫任何唯讀 AWS API，並以 JSON 格式顯示結果。
- 快速 CloudWatch 點陣 CloudWatch 圖 — 使用伺服器端呈現圖形，以便快速顯示。
- 從 CloudWatch 儀表板文本控件-顯示從指定 CloudWatch 儀表板的第一個文本控件。
- CloudWatch 以表格形式顯示測量結果資料 — 在表格中顯示原始 CloudWatch 測量結果資料。
- Amazon EC2 表 — 依 CPU 用量顯示前幾個 EC2 執行個體。此小工具也包含重新開機按鈕，預設為停用。
- AWS CodeDeploy 部署 — 顯示 CodeDeploy 部署。
- AWS Cost Explorer 報告 — 顯示所選時間範圍內每項 AWS 服務成本的報告。
- 顯示外部 URL 的內容 — 顯示可外部存取之 URL 的內容。
- 顯示 Simple Storage Service (Amazon S3) 物件 – 在您帳戶中顯示 Simple Storage Service (Amazon S3) 儲存貯體中的物件。
- 簡單的 SVG 圓餅圖 — 以 SVG 為基礎的圖形小工具範例。

Python

Python 中的範例自訂小工具

- Echo – 基本迴聲器，您可以用來測試 HTML 在自訂小工具中的顯示方式，而不必撰寫新的小工具。
- Hello World – 一個非常基本的入門小工具。
- 自訂小工具除錯器 – 偵錯器小工具，可顯示 Lambda 執行階段環境的實用資訊。
- 呼叫 AWS API — 呼叫任何唯讀 AWS API，並以 JSON 格式顯示結果。
- 快速 CloudWatch 點陣 CloudWatch 圖 — 使用伺服器端呈現圖形，以便快速顯示。
- 以電子郵件傳送儀表板快照 — 擷取目前儀表板的快照，然後將其傳送給電子郵件收件人。
- 將儀表板快照傳送至 Simple Storage Service (Amazon S3) — 擷取目前儀表板的快照，並將其存放在 Simple Storage Service (Amazon S3) 中。
- 從 CloudWatch 儀表板文本控件-顯示從指定 CloudWatch 儀表板的第一個文本控件。
- 顯示外部 URL 的內容 — 顯示可外部存取之 URL 的內容。
- RSS 讀取器 — 顯示 RSS 摘要。
- 顯示 Simple Storage Service (Amazon S3) 物件 – 在您帳戶中顯示 Simple Storage Service (Amazon S3) 儲存貯體中的物件。
- 簡單的 SVG 圓餅圖 — 以 SVG 為基礎的圖形小工具範例。

迴聲小部件 JavaScript

以下是中的 Echo 範例小器具 JavaScript。

```
const DOCS = `
## Echo
A basic echo script. Anything passed in the \`\`\`echo\`\`\` parameter is returned as
 the content of the custom widget.
### Widget parameters
Param | Description
---|---
**echo** | The content to echo back

### Example parameters
\`\`\` yaml
echo: <h1>Hello world</h1>
```

```
\`\`\`
`;

exports.handler = async (event) => {
    if (event.describe) {
        return DOCS;
    }

    let widgetContext = JSON.stringify(event.widgetContext, null, 4);
    widgetContext = widgetContext.replace(/</g, '&lt;');
    widgetContext = widgetContext.replace(/>/g, '&gt;');

    return `${event.echo || ''}<pre>${widgetContext}</pre>`;
};
```

Python 中的 Echo 小工具

以下是在 Python 中的 Echo 範例小工具。

```
import json

DOCS = """
## Echo
A basic echo script. Anything passed in the ```echo``` parameter is returned as the
 content of the custom widget.
### Widget parameters
Param | Description
---|---
**echo** | The content to echo back

### Example parameters
``` yaml
echo: <h1>Hello world</h1>
```"""

def lambda_handler(event, context):
    if 'describe' in event:
        return DOCS

    echo = event.get('echo', '')
    widgetContext = event.get('widgetContext')
    widgetContext = json.dumps(widgetContext, indent=4)
    widgetContext = widgetContext.replace('<', '<')
    widgetContext = widgetContext.replace('>', '>')
```

```
    return f'{echo}<pre>{widgetContext}</pre>'
```

Java 中的 Echo 小工具

以下是在 Java 中的 Echo 範例小工具。

```
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class Handler implements RequestHandler<Event, String>{

  static String DOCS = ""
    + "## Echo\n"
    + "A basic echo script. Anything passed in the ```echo``` parameter is returned as
 the content of the custom widget.\n"
    + "### Widget parameters\n"
    + "Param | Description\n"
    + "---|---\n"
    + "**echo** | The content to echo back\n\n"
    + "### Example parameters\n"
    + "```yaml\n"
    + "echo: <h1>Hello world</h1>\n"
    + "```\n";

  Gson gson = new GsonBuilder().setPrettyPrinting().create();

  @Override
  public String handleRequest(Event event, Context context) {

    if (event.describe) {
      return DOCS;
    }

    return (event.echo != null ? event.echo : "") + "<pre>" +
 gson.toJson(event.widgetContext) + "</pre>";
  }
}
```

```
class Event {

    public boolean describe;
    public String echo;
    public Object widgetContext;

    public Event() {}

    public Event(String echo, boolean describe, Object widgetContext) {
        this.describe = describe;
        this.echo = echo;
        this.widgetContext = widgetContext;
    }
}
```

從 CloudWatch 儀表板中新增或移除文字 Widget

文字 widget 包含格式為 Markdown 的文字區塊。您可以從儀表板新增、編輯或移除文字 CloudWatch Widget。

若要將文字 widget 新增到儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 選擇 + 符號。
4. 選擇文字。
5. 針對 Markdown，請使用 Markdown 並選擇 Create widget (建立 widget) 來新增和格式化您的文字。
6. 若要讓文字小工具變透明，請選擇透明背景。
7. 選擇 Save dashboard (儲存儀表板)。

若要在儀表板上編輯文字 widget

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 將滑鼠游標停留在文字區塊的右上角，然後選擇 Widget actions (小工具動作)。接著選擇 Edit (編輯)。

4. 視需要更新文字，然後選擇 Update widget (更新 widget)。
5. 選擇 Save dashboard (儲存儀表板)。

從儀表板移除文字 widget

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 將滑鼠游標停留在文字區塊的右上角，然後選擇 Widget actions (小工具動作)。再選擇 Delete (刪除)。
4. 選擇 Save dashboard (儲存儀表板)。

從 CloudWatch 儀表板新增或移除警報小工具

若要將警示小工具新增至儀表板，請選擇下列其中一種選項：

- 在小工具中新增單一警示，該小工具會同時顯示警示指標圖表和警示狀態。
- 新增警示狀態小工具，這會顯示網格中多個警示的狀態。只會顯示警示名稱和目前狀態，不會顯示圖表。一個警示狀態小工具最多可包含 100 個警示。

若要將單一警示 (包括其圖形) 新增至儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中選擇新增 Alarms (警示)，選取要新增的警示，然後選擇 Add to Dashboard (新增至儀表板)。
3. 選取儀表板，選擇 widget 類型 (Line (列)、Stacked area (堆疊區域) 或 Number (數字))，然後選擇 Add to dashboard (新增至儀表板)。
4. 若要在儀表板上查看警示，請選擇導覽窗格中的 Dashboards (儀表板) 以選取儀表板。
5. (選用) 如果要讓警示圖形暫時變大，請選取圖形。
6. (選用) 如果要變更小工具類型，請將滑鼠游標暫留在圖形的標題上，選擇小工具動作，然後選擇小工具類型。

若要將警示狀態小工具新增至儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/

2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 選擇 + 符號。
4. 選擇 Alarm status (警示狀態)。
5. 選取您要新增至小工具之警示旁的核取方塊，然後選擇 Create widget (建立小工具)。
6. 選擇 Add to dashboard (新增至儀表板)。

若要從儀表板移除警示小工具

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 將滑鼠游標暫留在小工具上，選擇小工具動作，然後選擇刪除。
4. 選擇 Save dashboard (儲存儀表板)。如果您嘗試在儲存變更前離開儀表板，您會收到提示，告知您儲存或捨棄變更。

從 CloudWatch 儀表板新增或移除資料表小器具

使用資料表小工具，可以查看指標的原始資料點以及原始資料的快速摘要。由於資料表小工具不是將實際資料抽象出來的圖表，因此更容易理解所呈現的資料點。本節中的程序說明如何從儀表板新增和移除資料表 CloudWatch Widget。

資料表屬性

資料表具有一組預設屬性，不需要對選項或來源進行任何變更。這些屬性包括一個黏性標籤欄、所有已啟用的摘要欄、四捨五入的資料點以及它們轉換的單位。

每個資料表小工具擁有下列屬性。每個屬性的相關資訊包括如何在儀表板的 JSON 來源中進行設定。如需儀表板 JSON 的詳細資訊，請參閱儀表板內文結構及語法。

摘要

摘要欄是隨資料表小工具引入的新屬性。這些資料欄是目前表格摘要的特定子集。例如，總和摘要是其列中所有已顯示資料點的總和。摘要資料行與 CloudWatch 統計資料不一樣。在源程式碼中表示為：

```
 "table": {
        "summaryColumns": [
            "MIN",
            "MAX",
            "SUM",
            "AVG"
        ]
    },
```

閾值

使用此選項可將閾值套用至您的表格。當資料點在閾值內時，其儲存格會以閾值顏色反白顯示。在源程式碼中表示為：

```
"annotations": {
    "horizontal": [
        {
            "label": string,
            "value": int,
            "fill": "above" | "below"
        }
    ]
}
```

標籤欄中的單位

若要顯示與指標相關聯的單位，可以啟用此選項，在標籤旁邊的標籤欄中顯示單位。在源程式碼中表示為：

```
"yAxis": {
    "left": {
        "showUnits": true | false
    }
```

```
}
```

反轉列和欄

這會轉換資料表，以便資料點從欄轉換為列，而指標變為欄。在源程式碼中表示為：

```
 "table": {
    "layout": "vertical" | "horizontal"
}
```

黏性摘要欄

這可讓摘要欄具有黏性，以便在捲動時保留在檢視中。標籤已具有黏性。在源程式碼中表示為：

```
"table": {
    "stickySummary": true | false
}
```

僅顯示摘要欄

這樣可以防止顯示資料點欄，因此只會顯示標籤欄和摘要欄。在源程式碼中表示為：

```
 "table": {
    "showTimeSeriesData": false | true
}
```

即時資料

顯示最新的資料點，即使資料點尚未完全彙總。在源程式碼中表示為：

```
"liveData": true | false
```

數字小工具格式

在四捨五入和轉換之前，顯示儲存格中可以容納的位數。在源程式碼中表示為：

```
"singleValueFullPrecision": true | false
```

將資料表小工具新增至儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/

2. 在導覽窗格中，選擇儀表板，然後選擇一個儀表板。
3. 選擇 + 按鈕，選取資料表，然後選擇下一步。
4. 在瀏覽索引標籤中，搜尋或瀏覽要顯示在資料表小工具中的指標。然後選取指標。
5. (選用) 若要變更表格的版面配置，請選擇選項索引標籤，然後選取反轉列和欄。

   也可以使用選項索引標籤來變更表格中顯示的欄，並顯示標籤欄中使用的單位。

   > **Tip**
   >
   > 若要顯示更精確的閾值，請選擇在四捨五入之前顯示盡可能多的數字。

6. (選用) 若要變更資料表小工具的時間範圍，請選取小工具上方區域中預先定義的時間範圍。時間範圍從 1 小時到 1 週。若要設定自己的時間範圍，請選擇 Custom (自訂)。
7. (選用) 若要變更資料表小工具的時間範圍，請選取小工具上方區域中預先定義的時間範圍。時間範圍從 1 小時到 1 週。若要設定自己的時間範圍，請選擇 Custom (自訂)。
8. (選用) 若要讓此小工具繼續使用您選取的時間範圍，即時儀表板其餘部分的時間範圍稍後會變更，請選擇持續時間範圍。
9. 選擇建立小工具，然後選擇儲存儀表板。

從儀表板中刪除資料表小工具

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 在您要移除的小工具的右上角，選擇小工具動作，然後選擇刪除。
4. 選擇 Save dashboard (儲存儀表板)。

在儀表板上連結和取消連結圖形 CloudWatch

您可以在儀表板中同時連結圖形，如此一來，當您放大或縮小一個圖形時，其他圖形也會跟著放大或縮小。您可以取消連結圖形，以限制對一個圖形進行縮放。

若要在儀表板上連結圖形

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。

3. 選擇 Actions (動作)，然後選擇 Link graph (連結圖形)。

在儀表板上取消連結圖形

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 清除 Actions (動作)，然後清除 Link graphs (連結圖形)。

共用 CloudWatch 儀表板

您可以與無法直接存取您 AWS 帳戶的使用者共用 CloudWatch 儀表板。這樣一來，您即可跨小組、與利害關係人以及組織外部人員共享儀表板。您甚至可以在小組區域的大螢幕上顯示儀表板，或將儀表板嵌入 Wiki 和其他網頁中。

> **Warning**
>
> 與您共享儀表板的所有人員都會授予帳戶 授予與您共享儀表板之人員的許可 中所列的許可。如果您公開共享許可儀表板，則擁有儀表板連結的所有人都會擁有這些許可。`cloudwatch:GetMetricData`和`ec2:DescribeTags`許可不限於特定指標或 EC2 執行個體的範圍，因此可存取儀表板的人員可以查詢帳戶中所有 CloudWatch EC2 執行個體的所有指標以及名稱和標籤。

共享儀表板時，您可以透過三種方式指定可以檢視儀表板的人員：

- 共用單一儀表板，並指定最多五個可檢視儀表板之人員的電子郵件地址。這些使用者都會建立自己的密碼，並且他們必須輸入密碼才能檢視儀表板。
- 公開共享單一儀表板，讓擁有該連結的任何人員都可以檢視儀表板。
- 共用您帳戶中的所有 CloudWatch 儀表板，並指定第三方單一登入 (SSO) 提供者來存取儀表板。屬於此 SSO 供應商名單的成員的所有使用者都可以存取帳戶中的所有儀表板。若要啟用此功能，請將 SSO 供應商與 Amazon Cognito 整合。SSO 供應商必須支援安全性聲明標記語言 (SAML)。如需 Amazon Cognito 定價的相關資訊，請參閱 什麼是 Amazon Cognito？

共用儀表板不會產生費用，但共用儀表板內的 Widget 會以標準費 CloudWatch 率收費。如需有關 CloudWatch 定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

共用儀表板時，Amazon Cognito 資源會在美國東部 (維吉尼亞北部) 區域建立。

> **Important**
>
> 請勿修改透過儀表板共用程序建立的資源名稱和識別符。這包括 Amazon Cognito 和 IAM 資源。修改這些資源可能會導致共用儀表板出現非預期和不正確的功能。

> **Note**
>
> 如果您共用的儀表板具有帶警示註釋的指標小工具，則您與之共用儀表板的人員將不會看到這些小工具。他們將看到一個空白小工具，且文字寫著小工具不可用。自行檢視儀表板時，您仍會看到帶有警示註釋的指標小工具。

共享儀表板所需的許可

若要能夠使用下列任何方法共用儀表板並查看已共用哪些儀表板，您必須以使用者身分或具有特定許可的 IAM 角色登入。

若要能夠共用儀表板，您的使用者或 IAM 角色必須包含下列政策陳述式中包含的許可：

```
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateRole",
        "iam:CreatePolicy",
        "iam:AttachRolePolicy",
        "iam:PassRole"
    ],
    "Resource": [
        "arn:aws:iam::*:role/service-role/CWDBSharing*",
        "arn:aws:iam::*:policy/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "cognito-idp:*",
        "cognito-identity:*",
    ],
```

```
    "Resource": [
        "*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "cloudwatch:GetDashboard",
    ],
    "Resource": [
        "*"
        // or the ARNs of dashboards that you want to share
    ]
}
```

若要能夠查看已共用哪些儀表板，但無法共用儀表板，使用者或 IAM 角色可以包含類似下列內容的政策陳述式：

```
{
    "Effect": "Allow",
    "Action": [
        "cognito-idp:*",
        "cognito-identity:*"
    ],
    "Resource": [
        "*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "cloudwatch:ListDashboards",
    ],
    "Resource": [
        "*"
    ]
}
```

授予與您共享儀表板之人員的許可

共用儀表板時，請在帳戶中 CloudWatch 建立 IAM 角色，將下列權限提供給與您共用儀表板的人員：

- `cloudwatch:GetInsightRuleReport`

- `cloudwatch:GetMetricData`
- `cloudwatch:DescribeAlarms`
- `ec2:DescribeTags`

> **Warning**
>
> 與您共享儀表板的所有人員都會授予帳戶的這些許可。如果您公開共享許可儀表板，則擁有儀表板連結的所有人都會擁有這些許可。
> `cloudwatch:GetMetricData`和`ec2:DescribeTags`許可不限於特定指標或 EC2 執行個體的範圍，因此可存取儀表板的人員可以查詢帳戶中所有 CloudWatch EC2 執行個體的所有指標以及名稱和標籤。

當您共用儀表板時，依預設會 CloudWatch 建立僅限共用儀表板上警示和 Contributor Insights 規則的存取權限。如果您將新的警示或 Contributor Insights 規則新增至儀表板，並希望與您共享儀表板的人員也能看到這些規則，則必須更新政策以允許這些資源。

與特定使用者共享單一儀表板

使用本節中的步驟，與最多五個您選擇的電子郵件地址共用儀表板。

> **Note**
>
> 根據預設，與您共用儀表板的人員看不到儀表板上的任何 CloudWatch Logs Widget。如需詳細資訊，請參閱 允許您與其共享的對象查看日誌表小工具。
> 根據預設，與您共享儀表板的人員無法看見儀表板上的任何複合警示小工具。如需詳細資訊，請參閱 允許您共享的對象查看複合警示。

若要與特定使用者共享儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)。
3. 選擇儀表板的名稱。
4. 選擇 Actions (動作)，Share dashboard (共享儀表板)。
5. 在 Share your dashboard and require a username and password (共享儀表板並要求使用者名稱和密碼) 旁，選擇 Start sharing (開始共享)。

6. 在 Add email addresses (新增電子郵件地址) 下，輸入您要與之共享儀表板的電子郵件地址。您最多可以包含五個電子郵件地址。
7. 輸入所有電子郵件地址後，請閱讀合約並選取確認方塊。接著選擇 Preview policy (預覽政策)。
8. 確認要共享的資源是您想要的，然後選擇 Confirm and generate shareable link (確認並產生可共享的連結)。
9. 在下一頁中，選擇 Copy link to clipboard (複製連結到剪貼簿)。然後，您可以將此連結貼到電子郵件中，並將其傳送給受邀的使用者。他們會自動接收到一封電子郵件，其中包含其使用者名稱和用來連線至儀表板的臨時密碼。

公開共享單一儀表板

請遵循本節中的步驟，以公開共享儀表板。這對於在團隊會議室的大螢幕上顯示儀表板，或將其內嵌於 Wiki 頁面中都非常有用。

> **Important**
>
> 公開共享儀表板可讓擁有該連結的任何人員存取儀表板，而不需要身分驗證。僅對不包含敏感資訊的儀表板執行此動作。

> **Note**
>
> 根據預設，與您共用儀表板的人員看不到儀表板上的任何 CloudWatch Logs Widget。如需詳細資訊，請參閱 允許您與其共享的對象查看日誌表小工具。
> 根據預設，與您共享儀表板的人員無法看見儀表板上的任何複合警示小工具。如需詳細資訊，請參閱 允許您共享的對象查看複合警示。

若要公開共享儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)。
3. 選擇儀表板的名稱。
4. 選擇 Actions (動作)，Share dashboard (共享儀表板)。
5. 在 Share your dashboard publicly (公開共享儀表板) 旁，選擇 Start sharing (開始共享)。
6. 在文字方塊中輸入 **`Confirm`**。

7. 閱讀合約並選取確認方塊。接著選擇 Preview policy (預覽政策)。
8. 確認要共享的資源是您想要的，然後選擇 Confirm and generate shareable link (確認並產生可共享的連結)。
9. 在下一頁中，選擇 Copy link to clipboard (複製連結到剪貼簿)。然後您可以共享此連結。您與之共享連結的任何人員都可以存取儀表板，而不需提供憑證。

使用 SSO 共用帳戶中的所有 CloudWatch 儀表板

使用本節中的步驟，以使用單一登入 (SSO) 與使用者共享帳戶中的所有儀表板。

> **Note**
>
> 根據預設，與您共用儀表板的人員看不到儀表板上的任何 CloudWatch Logs Widget。如需詳細資訊，請參閱 允許您與其共享的對象查看日誌表小工具。
> 根據預設，與您共享儀表板的人員無法看見儀表板上的任何複合警示小工具。如需詳細資訊，請參閱 允許您共享的對象查看複合警示。

與 SSO 提供者清單中的使用者共用 CloudWatch 儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)。
3. 選擇儀表板的名稱。
4. 選擇 Actions (動作)，Share dashboard (共享儀表板)。
5. 選擇「前往 CloudWatch 設定」。
6. 如果您想要的 SSO 供應商未列在可用的 SSO 供應商中，選擇 Manage SSO providers (管理 SSO 供應商)，並遵循 設定 CloudWatch 儀表板共用的 SSO 中的指示。

   然後返回 CloudWatch 主控台並重新整理瀏覽器。您啟用的 SSO 供應商現在應該會顯示在清單中。
7. 在可用的 SSO 供應商清單中選擇您想要的 SSO 供應商。
8. 選擇儲存變更。

設定 CloudWatch 儀表板共用的 SSO

若要透過支援 SAML 的第三方單一登入供應商來設定儀表板共享，請遵循這些步驟。

> **Important**
>
> 強烈建議您不要使用非 SAML SSO 供應商共享儀表板。這樣做會導致無意中允許第三方存取您帳戶的儀表板的風險。

若要設定 SSO 供應商以啟用儀表板共享

1. 將 SSO 供應商與 Amazon Cognito 整合。如需詳細資訊，請參閱將第三方 SAML 身分提供者與 Amazon Cognito 使用者集區整合。
2. 從 SSO 供應商下載中繼資料 XML 檔案。
3. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
4. 在導覽窗格中，選擇設定。
5. 在 Dashboard sharing (共享儀表板) 區段中，選擇 Configure (設定)。
6. 選擇 Manage SSO providers (管理 SSO 供應商)。

   這會在美國東部 (維吉尼亞北部) 區域 (us-east-1) 中開啟 Amazon Cognito 主控台。如果您沒有看到使用者集區，則 Amazon Cognito 主控台可能已在不同的區域中開啟。如果是這樣，請將區域變更為美國東部 (維吉尼亞北部) us-east-1，然後繼續進行後續步驟。
7. 選擇CloudWatchDashboardSharing池。
8. 在導覽窗格中，請選擇 Identity providers (身分提供者)。
9. 選擇 SAML。
10. 在 Provider name (供應商名稱) 中，輸入您的 SSO 供應商的名稱。
11. 選擇 Select file (選取檔案)，然後選取您在步驟 1 中下載的中繼資料 XML 檔案。
12. 選擇 Create provider (建立供應商)。

查看共享的儀表板數目

您可以使用主 CloudWatch 控台來查看目前有多少 CloudWatch 儀表板正在與其他人共用。

若要查看共享的儀表板數目

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇設定。
3. 儀表板共享區段會顯示共享的儀表板數目。

4. 若要查看共享的儀表板，請選擇 Username and password (使用者名稱和密碼) 下和 Public dashboards (公開儀表板) 下的 ***number*** dashboards shared (共享的儀表板數目)。

查看要共享哪些儀表板

您可以使用主 CloudWatch 控台來查看目前正在與其他人共用的儀表板。

若要查看要共享哪些儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)。
3. 在儀表板清單中，請參閱 Share (共享) 欄。已填入此欄中的圖示的儀表板目前正在共享中。
4. 若要查看與哪些使用者共享儀表板，請選擇儀表板名稱，然後選擇 Actions (動作)、Share dashboard (共享儀表板)。

   Share dashboard ***dashboard name*** (共享儀表板 (儀表板名稱)) 頁面會顯示如何共享儀表板。如果需要，您可以選擇 Stop sharing (停止共享)，進而停止共享儀表板。

停止共享一個或多個儀表板

您可以停止共享單一共享儀表板，或一次停止共享所有共享儀表板。

若要停止共享單一儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)。
3. 選擇共享儀表板的名稱。
4. 選擇 Actions (動作)，Share dashboard (共享儀表板)。
5. 選擇 Stop sharing (停止共享)。
6. 在確認對話方塊中，選擇 Stop sharing (停止共享)。

若要停止共享所有共享儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Settings (設定)。

3. 在 Dashboard sharing (儀表板共享) 區段中，選擇 Stop sharing all dashboards (停止共享所有儀表板)。
4. 在確認對話方塊中，選擇 Stop sharing all dashboards (停止共享所有儀表板)。

檢閱共享儀表板許可並變更許可範圍

如果您想要檢閱共享儀表板的使用者許可，或變更共享儀表板許可的範圍，請使用本節中的步驟。

若要檢閱共享儀表板許可

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)。
3. 選擇共享儀表板的名稱。
4. 選擇 Actions (動作)，Share dashboard (共享儀表板)。
5. 在 Resources (資源) 下，選擇 IAM Role (IAM 角色)。
6. 在 IAM 主控台，選擇顯示的政策。
7. (選用) 若要限制共享儀表板使用者可以看到的警示，請選擇 Edit policy (編輯政策)，然後將 `cloudwatch:DescribeAlarms` 許可從目前位置移到新的 `Allow` 陳述式，其中只列出您想要讓共享儀表板使用者看到的警示 ARN。請參閱以下範例。

```
{
   "Effect": "Allow",
    "Action": "cloudwatch:DescribeAlarms",
    "Resource": [
        "AlarmARN1",
        "AlarmARN2"
    ]
}
```

如果您執行此動作，請務必從目前政策的區段中移除 `cloudwatch:DescribeAlarms` 許可，例如：

```
{
   "Effect": "Allow",
    "Action": [
        "cloudwatch:GetInsightRuleReport",
        "cloudwatch:GetMetricData",
        "cloudwatch:DescribeAlarms",
```

```
        "ec2:DescribeTags"
    ],
    "Resource": "*"
}
```

8. (選用) 若要限制共享儀表板使用者可以看到的 Contributor Insights 規則的適用範圍，請選擇 Edit policy (編輯政策)，然後將 `cloudwatch:GetInsightRuleReport` 從目前位置移到新的 `Allow` 陳述式，其中只列出您想要讓共享儀表板使用者看到的 Contributor Insights 規則 ARN。請參閱以下範例。

```
{
   "Effect": "Allow",
    "Action": "cloudwatch:GetInsightRuleReport",
    "Resource": [
        "PublicContributorInsightsRuleARN1",
        "PublicContributorInsightsRuleARN2"
    ]
}
```

如果您執行此動作，請務必從目前政策的區段中移除 `cloudwatch:GetInsightRuleReport`，例如：

```
{
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetInsightRuleReport",
                "cloudwatch:GetMetricData",
                "cloudwatch:DescribeAlarms",
                "ec2:DescribeTags"
            ],
            "Resource": "*"
        }
```

允許您共享的對象查看複合警示

當您共享儀表板時，根據預設，與您共享儀表板的人員無法看見儀表板上的任何複合警示小工具。為了使複合警示小工具可見，您需要將 `DescribeAlarms: *` 許可新增至儀表板共享政策。該許可如下所示：

```
{
```

```
    "Effect": "Allow",
    "Action": "cloudwatch:DescribeAlarms",
    "Resource": "*"
}
```

Warning

上述政策陳述式可讓您存取帳戶中所有警示。若要減少 `cloudwatch:DescribeAlarms` 的範圍，您必須使用 `Deny` 陳述式。您可以將 `Deny` 陳述式新增至政策，並指定您要鎖定之警示的 ARN。拒絕陳述式應類似以下內容：

```
{
    "Effect": "Allow",
    "Action": "cloudwatch:DescribeAlarms",
    "Resource": "*"
},
{
    "Effect": "Deny",
    "Action": "cloudwatch:DescribeAlarms",
    "Resource": [
        "SensitiveAlarm1ARN",
        "SensitiveAlarm1ARN"
    ]
}
```

允許您與其共享的對象查看日誌表小工具

當您共用儀表板時，預設情況下，您共用儀表板的人員看不到儀表板上的「CloudWatch 記錄見解」Widget。這會影響現在存在的「CloudWatch 記錄見解」Widget，以及在您共用之後新增至儀表板的任何小器具。

如果您希望這些人員能夠看到 CloudWatch 日誌小部件，則必須向 IAM 角色添加許可以進行儀表板共享。

允許與您共用控制面板的人員查看記 CloudWatch 錄 Widget

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)。
3. 選擇共享儀表板的名稱。

4. 選擇 Actions (動作)，Share dashboard (共享儀表板)。
5. 在 Resources (資源) 下，選擇 IAM Role (IAM 角色)。
6. 在 IAM 主控台，選擇顯示的政策。
7. 選擇 Edit policy (編輯政策)，然後新增下列陳述式。在新的陳述式中，我們建議您指定只有您想要共享的日誌群組的 ARN。請參閱以下範例。

```
{
            "Effect": "Allow",
            "Action": [
                "logs:FilterLogEvents",
                "logs:StartQuery",
                "logs:StopQuery",
                "logs:GetLogRecord",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "SharedLogGroup1ARN",
                "SharedLogGroup2ARN"
           ]
        },
```

8. 選擇 Save Changes (儲存變更)。

如果您的儀表板共享的 IAM 政策已包含以 * 作為資源的這五個許可，強烈建議您變更政策，並僅指定您想要共享之日誌群組的 ARN。例如，如果這些許可的 Resource 區段如下所示：

```
"Resource": "*"
```

變更政策以僅指定您要共享之日誌群組的 ARN，如下列範例所示：

```
"Resource": [
  "SharedLogGroup1ARN",
  "SharedLogGroup2ARN"
]
```

允許您與其共享的對象查看自訂小工具

當您共享儀表板時，根據預設，與您共享儀表板的人員無法看見儀表板上的自訂小工具。這會影響現在存在的自訂小工具，以及您共享後新增至儀表板的任何小工具。

如果您希望這些人員能夠看到自訂小工具，則必須為 IAM 角色新增許可以進行儀表板共享。

若要允許您共享儀表板的對象查看自訂小工具

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)。
3. 選擇共享儀表板的名稱。
4. 選擇 Actions (動作)，Share dashboard (共享儀表板)。
5. 在 Resources (資源) 下，選擇 IAM Role (IAM 角色)。
6. 在 IAM 主控台，選擇顯示的政策。
7. 選擇 Edit policy (編輯政策)，然後新增下列陳述式。在新的陳述式中，我們建議您指定只有您想要共享的 Lambda 函數的 ARN。請參閱以下範例。

```
{
  "Sid": "Invoke",
  "Effect": "Allow",
  "Action": [
      "lambda:InvokeFunction"
  ],
  "Resource": [
    "LambdaFunction1ARN",
    "LambdaFunction2ARN"
  ]
 }
```

8. 選擇 Save Changes (儲存變更)。

如果您的儀表板共享的 IAM 政策已包含以 * 作為資源的許可，我們強烈建議您變更政策，並僅指定您想要共享之 Lambda 函數的 ARN。例如，如果這些許可的 `Resource` 區段如下所示：

```
"Resource": "*"
```

變更政策以僅指定您要共享之自訂小工具的 ARN，如下列範例所示：

```
"Resource": [
  "LambdaFunction1ARN",
  "LambdaFunction2ARN"
]
```

使用即時資料

您可以選擇是否要讓指標 Widget 顯示即時資料。即時資料是在尚未完全彙總的最後一分鐘內所發佈的資料。

- 如果即時資料已關閉，則只會顯示至少過去一分鐘彙總期間內的資料點。例如，若使用 5 分鐘的期間，則 12:35 的資料點會從 12:35 彙總到 12:40，並在 12:41 顯示。
- 如果即時資料已開啟，則會在對應彙總時間間隔內發佈任何資料時，立即顯示最新資料點。每次重新整理顯示畫面時，最新資料點可能會在該彙總期間內發佈新資料的同時隨之變更。如果您使用累積統計資料 (例如 Sum (總和) 或 Sample Count (範例計數))，使用即時資料可能會導致圖形結尾探底。

您可以選擇為整個儀表板或儀表板上的個別 Widget 啟用即時資料。

選擇是否在整個儀表板上使用即時資料

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 若要永久開啟或關閉儀表板上所有小工具的即時資料，請執行下列操作：
   a. 依序選擇 Actions (動作)、Settings (設定)、Bulk update live data (大量更新即時資料)。
   b. 選擇 Live Data on (即時資料開啟) 或 Live Data off (即時資料關閉)，然後選擇 Set (設定)。
4. 若要暫時覆寫每個小工具的即時資料設定，請選擇 Actions (動作)。然後，在 Overrides (覆寫) 下的 Live data (即時資料) 旁，執行以下操作：
   - 選擇 On (開啟) 以暫時開啟所有 Widget 的即時資料。
   - 選擇 Off (關閉) 以暫時關閉所有 Widget 的即時資料。
   - 選擇 Do not override (請勿覆寫) 以保留每個 Widget 的即時資料設定。

選擇是否在單一 Widget 上使用即時資料

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 選取 Widget，然後依序選擇 Actions (動作)、Edit (編輯)。
4. 選擇 Graph options (圖形選項) 標籤。
5. 選取或清除 Live Data (即時資料) 下方的核取方塊。

檢視動畫儀表板

您可以檢視動畫儀表板，該儀表板會重播隨時間擷取的 CloudWatch 量度資料。這可協助您查看趨勢、進行簡報，或在問題發生後分析問題。

儀表板中的動畫小工具包括折線圖小工具、堆疊區域圖小工具、數字小工具和指標瀏覽器小工具。圓餅圖、長條圖、文字小工具和日誌小工具會顯示在儀表板中，但不會以動畫方式顯示。

若要檢視動畫儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)。
3. 選擇儀表板的名稱。
4. 選擇 Actions (動作)，Replay dashboard (重播儀表板)。
5. (選用) 根據預設，當您啟動動畫時，其會顯示為滑動視窗。如果您希望動畫顯示為 point-by-point 動畫，請在動畫暫停時選擇放大鏡圖示，然後重設縮放。
6. 若要開始動畫，請選擇 Play (播放) 按鈕。您也可以選擇 back (返回) 和 forward (前進) 按鈕，以便移動到其他時間點。
7. (選用) 若要變更動畫的時間視窗，請選擇行事曆並選取時間期間。
8. 若要變更動畫的速度，請選擇 Auto speed (自動速度)，然後選取新的速度。
9. 完成後，選擇 Exit animate (結束動畫)。

將 CloudWatch 儀表板新增至我的最愛清單

在 CloudWatch 主控台中，您可以將儀表板、警示和記錄群組新增至我的最愛清單。您可以從導覽窗格存取我的最愛清單。下列程序介紹如何將儀表板新增至我的最愛清單。

將儀表板新增至我的最愛清單

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)。
3. 從儀表板清單中，選取要加入我的最愛的儀表板名稱旁邊的星號。
   - (選用) 您也可以從清單中選取儀表板，然後選擇儀表板名稱旁邊的星號，將儀表板加入我的最愛。

4. 若要存取我的最愛清單，請選擇導覽窗格中的 Favorites and recents (我的最愛和最近的項目)。選單包含兩個資料欄。一個資料欄包含您最愛的儀表板、警示和日誌群組，另一個資料欄包含您最近造訪的儀表板、警示和日誌群組。

> **Tip**
>
> 您可以從 CloudWatch 主控台導覽窗格中的 [我的最愛] 和 [最近項目] 功能表，將儀表板以及警示和記錄群組加入我的最愛。在 Recently visited (最近造訪) 資料欄下，將滑鼠游標停留在要加入我的最愛的儀表板上，然後選擇其旁邊的星號。

變更 CloudWatch 儀表板的期間覆寫設定或重新整理間隔

您可以指定如何保留或修改新增到此儀表板的圖形期間設定。

將自動期間或持續時間範圍套用至 Widget 時，圖形的整體時間範圍可能會影響您已設定的期間。

- 如果時間範圍為一天或更短，不會變更期間設定。
- 如果時間範圍介於一天和三天之間，則設定為五分鐘以下的期間會變更為 5 分鐘。
- 如果時間範圍超過三天，則設定為一小時以下的期間會變更為一小時。

下列步驟說明如何使用主控台來變更週期覆寫選項。也可以使用儀表板 JSON 結構中的 `period0verride` 欄位來變更它們。如需詳細資訊，請參閱儀表板內文整體結構。

變更期間覆寫選項

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 選擇動作。
3. 在 Period override (期間覆寫) 下，選擇以下其中一項：

   - 選擇 Auto (自動) 可使每個圖形上的指標期間自動調整為儀表板的時間範圍。
   - 選擇 Do not override (不要覆寫)，以確保一律遵守每個圖形的期間設定。
   - 選擇其他選項之一，以使新增到儀表板的圖形一律將所選時間作為期間設定以進行調整。

   在儀表板關閉或瀏覽器重新整理時，Period override (期間覆寫) 將一律還原為 Auto (自動)。Period override (期間覆寫) 的不同設定無法儲存。

您可以變更 CloudWatch 儀表板上資料重新整理的頻率。

若要變更儀表板重新整理間隔

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 在 Refresh options (重新整理選項) 功能表 (右上角) 上，選擇 10 Seconds (10 秒)、1 Minute (1 分鐘)、2 Minutes (2 分鐘)、5 Minutes (5 分鐘) 或 15 Minutes (15 分鐘)。

變更 CloudWatch儀表板的時間範圍或時區格式

您可以變更時間範圍以顯示幾分鐘、幾小時、幾天或幾週內的儀表板資料。您也可以變更時區格式，以 UTC 或本機時間顯示儀表板資料。當地時間是在電腦作業系統中指定的時區。

> **Note**
>
> 如果您建立含圖形的儀表板，其中包含 100 個或超過 100 個高解析度指標，我們建議您設定的時間範圍不要超過 1 小時。如需詳細資訊，請參閱 高解析度指標。

> **Note**
>
> 如果儀表板的時間範圍短於儀表板 Widget 使用的期間，則會發生下列情況：
>
> - 即使這比儀表板時間範圍更長，Widget 也會修改為顯示與該 Widget 相對應的一個完整期間的資料量。這可確保圖形上至少有一個資料點。
> - 此資料點期間的開始時間會向後調整，以確保至少可顯示一個資料點。

New console

若要變更儀表板時間範圍

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 在儀表板畫面執行下列操作：

- 在儀表板的上方區域中，選取其中一個預先定義的時間範圍。範圍為 1 小時到 1 週 (1 小時、3 小時、12 小時、1 天或 1 週)。
- 或者，您也可以選擇以下其中一個自訂時間範圍選項：
  - 選擇 Custom (自訂)，然後選擇 Relative (相對) 索引標籤。選擇從 1 分鐘到 15 個月的時間範圍。
  - 選擇 custom (自訂)，然後選擇 Absolute (絕對) 索引標籤。使用行事曆或文字欄位指定時間範圍。

> **Tip**
>
> 如果在變更圖表的時間範圍時，彙總週期設定為「自動」，CloudWatch 可能會變更期間。若要手動設定期間，請選擇 Actions (動作) 下拉式選單，然後選擇 Period override (期間覆寫)。

若要變更儀表板時區格式

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 在儀表板的上方區域中，選擇自訂。

4. 在出現的方塊右上角，選取下拉式選單中的 UTC 或 Local time (當地時間)。
5. 選擇套用。

Old console

若要變更儀表板時間範圍

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 在儀表板畫面執行下列操作：
   - 在儀表板的上方區域中，選取其中一個預先定義的時間範圍。範圍為 1 小時到 1 週 (1 小時、3 小時、12 小時、1 天、3 天或 1 週。

- 或者，您也可以選擇以下其中一個自訂時間範圍選項：
  - 選擇 Custom (自訂) 下拉式選單，然後選擇 Relative (相對) 索引標籤。選取其中一個預先定義的範圍，範圍從 1 分鐘到 15 個月。
  - 選擇 Custom (自訂) 下拉式選單，然後選擇 Absolute (絕對) 索引標籤。使用行事曆或文字欄位指定時間範圍。

> **Tip**
>
> 如果在變更圖表的時間範圍時，彙總週期設定為「自動」，CloudWatch 可能會變更期間。若要手動設定期間，請選擇 Actions (動作) 下拉式選單，然後選擇 Period override (期間覆寫)。

若要變更儀表板時區格式

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選擇一個儀表板。
3. 在儀表板畫面右上角，選擇 Custom (自訂) 下拉式選單。
4. 在所出現方塊的右上角，選取下拉式選單中的 UTC 或 Local timezone (當地時區)。

使用 Amazon CloudWatch 指標

指標是有關您系統效能的資料。根據預設，有許多服務提供免費的資源指標 (例如，Amazon EC2 執行個體、Amazon EBS 磁碟區，以及 Amazon RDS 資料庫執行個體)。您也可以啟用詳細監控一些資源，例如您的 Amazon EC2 執行個體，或發佈自己的應用程式指標。Amazon CloudWatch 可以載入您帳戶中的所有指標 (包括您提供的 AWS 資源指標和應用程式指標) 以進行搜尋、繪圖和警示。

指標資料會保留 15 個月，可讓您同時檢視 up-to-the-minute資料和歷史資料。

若要在主控台中繪製指標圖表，您可以使用 CloudWatch Metric Insights，這是一種高效能 SQL 查詢引擎，可用來即時識別所有指標中的趨勢和模式。

目錄

- 基本監控和詳細監控
- 使用指標洞察查詢您的 CloudWatch 指標
- 使用指標瀏覽器依其標籤和屬性監控資源
- 使用指標串流
- 檢視可用的指標
- 建立指標圖形
- 使用 CloudWatch 異常偵測
- 使用指標數學
- 在圖形中使用搜尋運算式
- 取得指標的統計資訊
- 發佈 自訂指標

基本監控和詳細監控

CloudWatch 提供兩種監控類別：基本監控和詳細監控。

許多 AWS 服務透過將一組預設的指標發佈到，提供基本監控，而不會向客戶收取任何費 CloudWatch 用。根據預設，當您開始使用其中一種時 AWS 服務，會自動啟用基本監視。如需提供基本監控的服務清單，請參閱AWS 發佈指 CloudWatch標的服務。

僅部分服務提供詳細監控。詳細監控會產生費用。要將其用於 AWS 服務，您必須選擇激活它。如需有關定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

詳細監控選項會因提供此監控的服務而異。例如，Amazon EC2 詳細監控提供時間間隔更小的指標，發佈間隔為一分鐘，而 Amazon EC2 基本監控的發佈間隔為五分鐘。Amazon S3 和適用於 Apache Kafka 的 Amazon Managed Streaming 的詳細監控則提供更為精細的指標。

在不同的 AWS 服務中，詳細監控也有不同的名稱。例如，在 Amazon EC2 中，它被稱為詳細監控，在 AWS Elastic Beanstalk 其中稱為增強型監控，在 Amazon S3 中，它被稱為請求指標。

使用 Amazon EC2 的詳細監控，可幫助您更好地管理 Amazon EC2 資源，以便您可以更快地尋找趨勢並採取行動。若是使用 Amazon S3，則可以使用一分鐘間隔的請求指標，幫助您快速找出操作問題並採取行動。在 Amazon MSK 上，如果啟用 `PER_BROKER`、`PER_TOPIC_PER_BROKER` 或 `PER_TOPIC_PER_PARTITION` 層級監控，可以使用更多指標，獲得更詳細的資訊。

下表列出提供詳細監控的服務。表中還包括這些服務的文件連結，這些文件會詳細介紹詳細監控，並提供使用說明。

服務	文件
Amazon API Gateway	API Gateway 指標的維度
Amazon CloudFront	檢視其他 CloudFront 分佈量度
Amazon EC2	啟用或關閉執行個體的詳細監控
Elastic Beanstalk	增強型運作狀態報告與監控
Amazon Kinesis Data Streams	增強型分區層級指標
Amazon MSK	用於監控的 Amazon MSK 指標 CloudWatch

服務	文件
Amazon S3	Amazon S3 請求指標 CloudWatch
Amazon SES	使用 Amazon SES 事件發佈收集 CloudWatch 詳細的監控指標。

此外，還 CloudWatch 提供 out-of-the-box 監控解決方案，其中包含更詳細的指標和某些 AWS 服務的預先建立儀表板，如下表所示。

服務	功能文件
Lambda	Lambda 洞察
Amazon ECS	Amazon ECS 的容器洞察
Amazon EKS	適用於 Amazon EKS 和庫伯尼特的容器洞察

使用指標洞察查詢您的 CloudWatch 指標

CloudWatch 指標見解是功能強大的高效能 SQL 查詢引擎，您可以使用它來大規模查詢指標。您可以即時識別所有 CloudWatch 指標中的趨勢和模式。

您也可以在傳回單一時間序列的任何 Metrics Insights 查詢上設定警示。這對於建立警示來監看基礎結構或應用程式機群的彙總指標尤其有用。建立警示後，在資源新增至機群或從機群中移除時，它會動態進行調整。

您可以使用「 CloudWatch 指標見解」查詢編輯器，在主控台中執行 CloudWatch 指標見解查詢。您也可以透過執行 AWS CLI 或或 AWS SDK 執`GetMetricData`行 CloudWatch 指標見解查

詢PutDashboard。使用「 CloudWatch 指標見解」查詢編輯器執行的查詢不會收取任何費用。如需有關 CloudWatch 定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

使用 Metric In CloudWatch sights 查詢編輯器，您可以從各種預先建立的範例查詢中進行選擇，也可以建立自己的查詢。建立查詢時，可以使用建置器檢視來瀏覽現有的指標和維度。或者，使用編輯器檢視手動編寫查詢。

您也可以使用自然語言來建立 CloudWatch 指標見解查詢。因此，請提出問題或描述您正在尋找的資料。此 AI 輔助功能會根據您的提示產生查詢，並提供查詢運作方式的 line-by-line 說明。如需詳細資訊，請參閱使用自然語言產生和更新 CloudWatch 指標見解查詢。

透過 Metrics Insights，可以大規模執行查詢。使用 GROUP BY 子句，可以按特定維度值將指標即時分組為單獨的時間序列。由於 Metrics Insights 查詢包含 ORDER BY 功能，因此您可以使用 Metrics Insights 進行 "Top N" 類型查詢。例如，"Top N" 類型查詢可以掃描您帳戶中的數百萬個指標，並傳回耗用最多 CPU 的 10 個執行個體。這可協助您精確找出應用程式中的延遲問題並加以修正。

主題

- 建置查詢
- Metrics Insights 查詢元件和語法
- 在 Metrics Insights 查詢上建立警示
- 使用 Metrics Insights 查詢搭配指標數學
- 使用自然語言產生和更新 CloudWatch 指標見解查詢
- SQL 推斷
- Metrics Insights 範例查詢
- Metrics Insights 限制
- Metrics Insights 字彙
- 對 Metrics Insights 進行疑難排解

建置查詢

您可以使用 CloudWatch 主控台、或 AWS SDK 執行 CloudWatch 指標見解查詢。 AWS CLI在主控台中執行的查詢無需支付任何費用。如需有關 CloudWatch 定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

如需有關使用 AWS SDK 執行指標見解查詢的詳細資訊，請參閱GetMetricData。

若要使用 CloudWatch 主控台執行查詢，請依照下列步驟執行：

使用 Metrics Insights 查詢指標

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Metrics (指標)、All metrics (所有指標)。
3. 選擇 Queries (查詢) 索引標籤。
4. (選用) 若要執行預先建置的範例查詢，請選擇 Add query (新增查詢)，然後選取要執行的查詢。如果您對此查詢感到滿意，則可以略過此處理程序的剩餘部分。或者，您也可以選擇 Editor (編輯器) 以編輯範例查詢，然後選擇 Run (執行) 來執行修改的查詢。
5. 若要建立自己的查詢，您可以使用 Builder (建置器) 檢視、Editor (編輯器) 檢視，也可以同時使用兩者的組合。您可以隨時在兩個檢視之間切換，並在兩個檢視中查看工作進度。

   在 Builder (建置器) 檢視中，您可以瀏覽並選取指標命名空間、指標名稱、篩選條件、群組和排序選項。對於這些選項中的每一個，查詢建置器都會為您提供環境中可能的選項清單，以供您選擇。

   在 Editor (編輯器) 檢視中，您可以開始編寫查詢。輸入時，編輯器會根據您目前輸入的字元提供建議。
6. 當您對您的查詢感到滿意時，請選擇 Run (執行)。
7. (選用) 另一種編輯已繪製圖形的查詢方式是選擇 Graphed metrics (繪製指標) 索引標籤，然後選擇 Details (詳細資訊) 資料欄中查詢公式旁的編輯圖示。
8. (選用) 若要從繪製中移除查詢，請選擇 Graphed metrics (繪製指標)，然後選擇顯示查詢之列右側的 X 圖示。

Metrics Insights 查詢元件和語法

CloudWatch 度量見解語法如下。

```
SELECT FUNCTION(metricName)
FROM namespace | SCHEMA(...)
[ WHERE labelKey OPERATOR labelValue [AND ... ] ]
[ GROUP BY labelKey [ , ... ] ]
[ ORDER BY FUNCTION() [ DESC | ASC ] ]
[ LIMIT number ]
```

Metrics Insights 查詢中可能的子句如下。所有關鍵字皆不區分大小寫，但識別符 (例如指標名稱、命名空間和維度) 會區分大小寫。

SELECT

必要。指定用於彙總每個時段 (由提供的時段決定) 中之觀察值的函數。並指定要查詢的指標名稱。

FUNCTION (函數) 的有效值是 AVG、COUNT、MAX、MIN 和 SUM。

- AVG 會計算查詢相符之觀察值的平均值。
- COUNT 會傳回查詢相符之觀察值的計數。
- MAX 會傳回查詢相符之觀察值的最大值。
- MIN 會傳回查詢相符之觀察值的最小值。
- SUM 會計算查詢相符之觀察值的加總。

FROM

必要。指定指標的來源。您可以指定包含要查詢之指標的指標命名空間，或 SCHEMA (結構描述) 資料表函數。指標命名空間的範例包括 "AWS/EC2"、"AWS/Lambda"，以及您為自訂指標建立的指標命名空間。

其中包含 / 或任何其他非字母、數字或底線字元的指標命名空間，必須以雙引號括住。如需詳細資訊，請參閱 什麼需要引號或跳脫字元？。

結構描述

可選資料表函數，可以在 FROM (從) 子句中使用。使用 SCHEMA (結構描述)，將查詢結果範圍縮小為僅完全符合維度清單的指標，或是沒有維度的指標。

如果您使用 SCHEMA (結構描述) 子句，其中必須至少包含一個引數，而且該第一個引數必須是要查詢的指標命名空間。如果您指定僅具有此命名空間引數的 SCHEMA (結構描述)，則結果的範圍縮小為僅限於沒有任何維度的指標。

如果您指定具有其他引數的 SCHEMA (結構描述)，則命名空間引數之後的其他引數必須是標籤索引鍵。標籤索引鍵必須是維度名稱。如果您指定一或多個這些標籤索引鍵，則結果的範圍僅限於具有完全相同維度集的指標。這些標籤索引鍵的排序並不重要。

例如：

- SELECT AVG(CPUUtilization) FROM "AWS/EC2" 符合 AWS/EC2 命名空間中的全部 CPUUtilization 指標，而無論其維度，並傳回單一彙總時間序列。
- SELECT AVG(CPUUtilization) FROM SCHEMA("AWS/EC2") 僅符合沒有任何已定義維度之 AWS/EC2 命名空間中的 CPUUtilization 指標。

- 從綱要選取 AVG (CPU 使用率) (「AWS/EC2", InstanceId) 只符合只 CloudWatch 有一個維度報告的CPUUtilization測量結果。InstanceId
- 從綱要中選取 SUM (RequestCount) (「AWS/ ApplicationELB」 LoadBalancer, AvailabilityZone) 僅符合剛好AWS/ApplicationELB有兩個維RequestCount度和的 CloudWatch 來源報告的測量結果。LoadBalancer AvailabilityZone

WHERE

選用。使用一或多個標籤索引鍵的特定標籤值，將結果篩選為僅符合指定表達式的指標。例如，WHERE InstanceType = 'c3.4xlarge' 會將結果篩選為僅**c3.4xlarge**執行個體類型，以及 WHERE! InstanceType = 'c3.4xlarge' 會將結果篩選為除外的所有執行個體類型。c3.4xlarge

當您在監控帳戶中執行查詢時，可以使用 WHERE AWS.AccountId 將結果僅限制給您指定的帳戶。例如，WHERE AWS.AccountId=444455556666 僅查詢來自帳戶 444455556666 的指標。若要將查詢僅限制給監控帳戶本身中的指標，請使用 WHERE AWS.AccountId=CURRENT_ACCOUNT_ID()。

標籤值一律必須以單引號括住。

支援的運算子

WHERE (哪裡) 子句支援下列運算子：

- = 標籤值必須與指定字串相符。
- != 標籤值必須與指定字串不相符。
- AND 指定的兩個條件都必須為 true 才能相符。您可以使用多個 AND 關鍵字來指定兩個或多個條件。

GROUP BY

選用。將查詢結果分組為多個時間序列，每個時間序列對應於指定的標籤索引鍵或索引鍵的不同值。例如，使用 GROUP BY InstanceId 為每個 InstanceId 值傳回不同的時間序列。使用 GROUP BY ServiceName, Operation 為每個可能的 ServiceName 和 Operation 值組合建立不同的時間序列。

使用 GROUP BY (分組依據) 子句，預設情況下，結果會根據 GROUP BY (分組依據) 子句中指定的標籤序列按字母升冪排序。若要變更結果排序，請新增 ORDER BY (排序依據) 子句至您的查詢。

當您在監控帳戶中執行查詢時，可以使用 GROUP BY AWS.AccountId 根據結果來自的帳戶對結果進行分組。

> **Note**
>
> 如果某些相符指標並未包含 GROUP BY (分組依據) 子句中指定的特定標籤索引鍵，則會傳回名為 `Other` 的空值群組。例如，如果您指定 `GROUP BY ServiceName, Operation`，且某些傳回的指標不包含 `ServiceName` 作為維度，則這些指標會顯示為像 `ServiceName` 值一樣擁有 `Other`。

ORDER BY

選用。如果查詢傳回一個以上的時間序列，則請指定要用於傳回之時間序列的排序。順序是基於您在 ORDER BY (排序依據) 子句中指定的 FUNCTION (函數) 所找到的值而定。FUNCTION (函數) 用於從每個傳回的時間序列計算單個純量值，並且該值用於確定排序。

您也可以指定是否使用升冪 ASC 或降冪 DESC 排序。如果您省略此參數，則預設為升冪 ASC。

例如，新增 `ORDER BY MAX() DESC` 子句會依照時間範圍內觀察到的最大資料點來降冪排序結果：表示會先傳回具有最高最大資料點的時間序列。

在 ORDER BY (排序依據) 子句中使用的有效函數是 `AVG()`、`COUNT()`、`MAX()`、`MIN()` 和 `SUM()`。

如果您將 ORDER BY (排序依據) 子句與 LIMIT (限制) 子句搭配使用，則產生的查詢是「前 N」查詢。ORDER BY (排序依據) 對可能會傳回大量指標的查詢也很有用，因為每個查詢可傳回不超過 500 個時間序列。如果查詢符合 500 個以上的時間序列，並且您使用 ORDER BY (排序依據) 子句，則會排序時間序列，然後排序順序中前 500 個時間序列就是傳回的時間序列。

LIMIT

選用。將查詢傳回的時間序列數量限制為您指定的值。您可以指定的最大值是 500，未指定 LIMIT (限制) 的查詢也可以傳回不超過 500 個時間序列。

將 LIMIT (限制) 子句與 ORDER BY (排序依據) 子句搭配使用會為您提供「前 N」查詢。

什麼需要引號或跳脫字元？

在查詢中，標籤值一律必須以單引號括住。例如，從「AWS/EC2」中選取最大值 (CPU 使用率)，其中 = "。 AutoScalingGroupName my-production-fleet

包含字母、數字和底線 (_) 以外字元的指標命名空間、指標名稱和標籤索引鍵必須以雙引號括住。例如，SELECT MAX("My.Metric")。

如果其中一個本身即包含雙引號或單引號 (例如 `Bytes"Input"`)，則必須使用反斜線跳脫每個引號，如 SELECT AVG("Bytes\"Input\"")。

如果指標命名空間、指標名稱或標籤索引鍵包含在 Metrics Insights 中保留關鍵字的文字，則這些文字也必須以雙引號括住。例如，如果您的指標名為 `LIMIT`，則您可以使用 `SELECT AVG("LIMIT")`。也可以用雙引號括住任何命名空間、指標名稱或標籤，即使其不包含保留的關鍵字也一樣。

如需保留關鍵字的完整清單，請參閱 保留的關鍵字。

逐步建置豐富的查詢

本節將逐步說明建置使用所有可能子句的完整範例。

我們從以下查詢開始，其彙總了以 `LoadBalancer` 和 `AvailabilityZone` 維度收集的所有 Application Load Balancer `RequestCount` 指標。

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
```

現在，如果我們只想查看來自特定負載平衡器的指標，可以新增 WHERE (哪裡) 子句，將傳回的指標限制為僅 `LoadBalancer` 維度值為 `app/load-balancer-1` 的指標。

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
WHERE LoadBalancer = 'app/load-balancer-1'
```

之前的查詢會將來自此負載平衡器的所有可用區域的 `RequestCount` 指標彙總至一個時間序列。如果想查看每個可用區域的不同時間序列，我們可以新增 GROUP BY (分組依據) 子句。

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
WHERE LoadBalancer = 'app/load-balancer-1'
GROUP BY AvailabilityZone
```

接下來，我們可能想要排序這些結果，以便先看到最高值。以下 ORDER BY (排序依據) 子句按查詢時間範圍內每個時間序列回報的最大值，以降冪排序時間序列：

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
WHERE LoadBalancer = 'app/load-balancer-1'
```

```
GROUP BY AvailabilityZone
ORDER BY MAX() DESC
```

最後，如果主要對「前 N」類型的查詢感興趣，我們可以使用 LIMIT (限制) 子句。這最後一個範例將結果限制為僅具有五個最高 MAX 值的時間序列。

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer, AvailabilityZone)
WHERE LoadBalancer = 'app/load-balancer-1'
GROUP BY AvailabilityZone
ORDER BY MAX() DESC
LIMIT 5
```

跨帳戶查詢範例

這些範例在設定為 CloudWatch 跨帳戶觀察性監視帳戶的帳戶中執行時是有效的。

下列範例會搜尋來源帳戶 123456789012 中的所有 Amazon EC2 執行個體，並傳回平均值。

```
SELECT AVG(CpuUtilization)
FROM "AWS/EC2"
WHERE AWS.AccountId ='123456789012'
```

下列範例會在所有連結的來源帳戶的 AWS/EC2 中查詢 CPUUtilization 指標，並依帳戶 ID 和執行個體類型將結果分組。

```
SELECT AVG(CpuUtilization)
FROM "AWS/EC2"
GROUP BY AWS.AccountId, InstanceType
```

下列範例會在監控帳戶本身中查詢 CPUUtilization。

```
SELECT AVG(CpuUtilization)
FROM "AWS/EC2"
WHERE AWS.AccountId = CURRENT_ACCOUNT_ID()
```

保留的關鍵字

以下是 CloudWatch 指標深入解析中的保留關鍵字。如果查詢中的命名空間、指標名稱或標籤索引鍵中包含任何這些關鍵字，則您必須以雙引號將其括住。保留的關鍵字不區分大小寫。

```
"ABORT" "ABORTSESSION" "ABS" "ABSOLUTE" "ACCESS" "ACCESSIBLE" "ACCESS_LOCK" "ACCOUNT"
 "ACOS" "ACOSH" "ACTION" "ADD" "ADD_MONTHS"
"ADMIN" "AFTER" "AGGREGATE" "ALIAS" "ALL" "ALLOCATE" "ALLOW" "ALTER" "ALTERAND" "AMP"
 "ANALYSE" "ANALYZE" "AND" "ANSIDATE" "ANY" "ARE" "ARRAY",
"ARRAY_AGG" "ARRAY_EXISTS" "ARRAY_MAX_CARDINALITY" "AS" "ASC" "ASENSITIVE" "ASIN"
 "ASINH" "ASSERTION" "ASSOCIATE" "ASUTIME" "ASYMMETRIC" "AT",
"ATAN" "ATAN2" "ATANH" "ATOMIC" "AUDIT" "AUTHORIZATION" "AUX" "AUXILIARY" "AVE"
 "AVERAGE" "AVG" "BACKUP" "BEFORE" "BEGIN" "BEGIN_FRAME" "BEGIN_PARTITION",
"BETWEEN" "BIGINT" "BINARY" "BIT" "BLOB" "BOOLEAN" "BOTH" "BREADTH" "BREAK" "BROWSE"
 "BT" "BUFFERPOOL" "BULK" "BUT" "BY" "BYTE" "BYTEINT" "BYTES" "CALL",
"CALLED" "CAPTURE" "CARDINALITY" "CASCADE" "CASCADED" "CASE" "CASESPECIFIC" "CASE_N"
 "CAST" "CATALOG" "CCSID" "CD" "CEIL" "CEILING" "CHANGE" "CHAR",
"CHAR2HEXINT" "CHARACTER" "CHARACTERS" "CHARACTER_LENGTH" "CHARS" "CHAR_LENGTH" "CHECK"
 "CHECKPOINT" "CLASS" "CLASSIFIER" "CLOB" "CLONE" "CLOSE" "CLUSTER",
"CLUSTERED" "CM" "COALESCE" "COLLATE" "COLLATION" "COLLECT" "COLLECTION" "COLLID"
 "COLUMN" "COLUMN_VALUE" "COMMENT" "COMMIT" "COMPLETION" "COMPRESS" "COMPUTE",
"CONCAT" "CONCURRENTLY" "CONDITION" "CONNECT" "CONNECTION" "CONSTRAINT" "CONSTRAINTS"
 "CONSTRUCTOR" "CONTAINS" "CONTAINSTABLE" "CONTENT" "CONTINUE" "CONVERT",
"CONVERT_TABLE_HEADER" "COPY" "CORR" "CORRESPONDING" "COS" "COSH" "COUNT" "COVAR_POP"
 "COVAR_SAMP" "CREATE" "CROSS" "CS" "CSUM" "CT" "CUBE" "CUME_DIST",
"CURRENT" "CURRENT_CATALOG" "CURRENT_DATE" "CURRENT_DEFAULT_TRANSFORM_GROUP"
 "CURRENT_LC_CTYPE" "CURRENT_PATH" "CURRENT_ROLE" "CURRENT_ROW" "CURRENT_SCHEMA",
"CURRENT_SERVER" "CURRENT_TIME" "CURRENT_TIMESTAMP" "CURRENT_TIMEZONE"
 "CURRENT_TRANSFORM_GROUP_FOR_TYPE" "CURRENT_USER" "CURRVAL" "CURSOR" "CV" "CYCLE"
 "DATA",
"DATABASE" "DATABASES" "DATABLOCKSIZE" "DATE" "DATEFORM" "DAY" "DAYS" "DAY_HOUR"
 "DAY_MICROSECOND" "DAY_MINUTE" "DAY_SECOND" "DBCC" "DBINFO" "DEALLOCATE" "DEC",
"DECFLOAT" "DECIMAL" "DECLARE" "DEFAULT" "DEFERRABLE" "DEFERRED" "DEFINE" "DEGREES"
 "DEL" "DELAYED" "DELETE" "DENSE_RANK" "DENY" "DEPTH" "DEREF" "DESC" "DESCRIBE",
"DESCRIPTOR" "DESTROY" "DESTRUCTOR" "DETERMINISTIC" "DIAGNOSTIC" "DIAGNOSTICS"
 "DICTIONARY" "DISABLE" "DISABLED" "DISALLOW" "DISCONNECT" "DISK" "DISTINCT",
"DISTINCTROW" "DISTRIBUTED" "DIV" "DO" "DOCUMENT" "DOMAIN" "DOUBLE" "DROP" "DSSIZE"
 "DUAL" "DUMP" "DYNAMIC" "EACH" "ECHO" "EDITPROC" "ELEMENT" "ELSE" "ELSEIF",
"EMPTY" "ENABLED" "ENCLOSED" "ENCODING" "ENCRYPTION" "END" "END-EXEC" "ENDING"
 "END_FRAME" "END_PARTITION" "EQ" "EQUALS" "ERASE" "ERRLVL" "ERROR" "ERRORFILES",
"ERRORTABLES" "ESCAPE" "ESCAPED" "ET" "EVERY" "EXCEPT" "EXCEPTION" "EXCLUSIVE" "EXEC"
 "EXECUTE" "EXISTS" "EXIT" "EXP" "EXPLAIN" "EXTERNAL" "EXTRACT" "FALLBACK
"FALSE" "FASTEXPORT" "FENCED" "FETCH" "FIELDPROC" "FILE" "FILLFACTOR" "FILTER" "FINAL"
 "FIRST" "FIRST_VALUE" "FLOAT" "FLOAT4" "FLOAT8" "FLOOR"
"FOR" "FORCE" "FOREIGN" "FORMAT" "FOUND" "FRAME_ROW" "FREE" "FREESPACE" "FREETEXT"
 "FREETEXTTABLE" "FREEZE" "FROM" "FULL" "FULLTEXT" "FUNCTION"
"FUSION" "GE" "GENERAL" "GENERATED" "GET" "GIVE" "GLOBAL" "GO" "GOTO" "GRANT" "GRAPHIC"
 "GROUP" "GROUPING" "GROUPS" "GT" "HANDLER" "HASH"
```

```
"HASHAMP" "HASHBAKAMP" "HASHBUCKET" "HASHROW" "HAVING" "HELP" "HIGH_PRIORITY" "HOLD"
 "HOLDLOCK" "HOUR" "HOURS" "HOUR_MICROSECOND" "HOUR_MINUTE"
"HOUR_SECOND" "IDENTIFIED" "IDENTITY" "IDENTITYCOL" "IDENTITY_INSERT" "IF" "IGNORE"
 "ILIKE" "IMMEDIATE" "IN" "INCLUSIVE" "INCONSISTENT" "INCREMENT"
"INDEX" "INDICATOR" "INFILE" "INHERIT" "INITIAL" "INITIALIZE" "INITIALLY" "INITIATE"
 "INNER" "INOUT" "INPUT" "INS" "INSENSITIVE" "INSERT" "INSTEAD"
"INT" "INT1" "INT2" "INT3" "INT4" "INT8" "INTEGER" "INTEGERDATE" "INTERSECT"
 "INTERSECTION" "INTERVAL" "INTO" "IO_AFTER_GTIDS" "IO_BEFORE_GTIDS"
"IS" "ISNULL" "ISOBID" "ISOLATION" "ITERATE" "JAR" "JOIN" "JOURNAL" "JSON_ARRAY"
 "JSON_ARRAYAGG" "JSON_EXISTS" "JSON_OBJECT" "JSON_OBJECTAGG"
"JSON_QUERY" "JSON_TABLE" "JSON_TABLE_PRIMITIVE" "JSON_VALUE" "KEEP" "KEY" "KEYS"
 "KILL" "KURTOSIS" "LABEL" "LAG" "LANGUAGE" "LARGE" "LAST"
"LAST_VALUE" "LATERAL" "LC_CTYPE" "LE" "LEAD" "LEADING" "LEAVE" "LEFT" "LESS" "LEVEL"
 "LIKE" "LIKE_REGEX" "LIMIT" "LINEAR" "LINENO" "LINES"
"LISTAGG" "LN" "LOAD" "LOADING" "LOCAL" "LOCALE" "LOCALTIME" "LOCALTIMESTAMP" "LOCATOR"
 "LOCATORS" "LOCK" "LOCKING" "LOCKMAX" "LOCKSIZE" "LOG"
"LOG10" "LOGGING" "LOGON" "LONG" "LONGBLOB" "LONGTEXT" "LOOP" "LOWER" "LOW_PRIORITY"
 "LT" "MACRO" "MAINTAINED" "MAP" "MASTER_BIND"
"MASTER_SSL_VERIFY_SERVER_CERT" "MATCH" "MATCHES" "MATCH_NUMBER" "MATCH_RECOGNIZE"
 "MATERIALIZED" "MAVG" "MAX" "MAXEXTENTS" "MAXIMUM" "MAXVALUE"
"MCHARACTERS" "MDIFF" "MEDIUMBLOB" "MEDIUMINT" "MEDIUMTEXT" "MEMBER" "MERGE" "METHOD"
 "MICROSECOND" "MICROSECONDS" "MIDDLEINT" "MIN" "MINDEX"
"MINIMUM" "MINUS" "MINUTE" "MINUTES" "MINUTE_MICROSECOND" "MINUTE_SECOND" "MLINREG"
 "MLOAD" "MLSLABEL" "MOD" "MODE" "MODIFIES" "MODIFY"
"MODULE" "MONITOR" "MONRESOURCE" "MONSESSION" "MONTH" "MONTHS" "MSUBSTR" "MSUM"
 "MULTISET" "NAMED" "NAMES" "NATIONAL" "NATURAL" "NCHAR" "NCLOB"
"NE" "NESTED_TABLE_ID" "NEW" "NEW_TABLE" "NEXT" "NEXTVAL" "NO" "NOAUDIT" "NOCHECK"
 "NOCOMPRESS" "NONCLUSTERED" "NONE" "NORMALIZE" "NOT" "NOTNULL"
"NOWAIT" "NO_WRITE_TO_BINLOG" "NTH_VALUE" "NTILE" "NULL" "NULLIF" "NULLIFZERO" "NULLS"
 "NUMBER" "NUMERIC" "NUMPARTS" "OBID" "OBJECT" "OBJECTS"
"OCCURRENCES_REGEX" "OCTET_LENGTH" "OF" "OFF" "OFFLINE" "OFFSET" "OFFSETS" "OLD"
 "OLD_TABLE" "OMIT" "ON" "ONE" "ONLINE" "ONLY" "OPEN" "OPENDATASOURCE"
"OPENQUERY" "OPENROWSET" "OPENXML" "OPERATION" "OPTIMIZATION" "OPTIMIZE"
 "OPTIMIZER_COSTS" "OPTION" "OPTIONALLY" "OR" "ORDER" "ORDINALITY" "ORGANIZATION"
"OUT" "OUTER" "OUTFILE" "OUTPUT" "OVER" "OVERLAPS" "OVERLAY" "OVERRIDE" "PACKAGE" "PAD"
 "PADDED" "PARAMETER" "PARAMETERS" "PART" "PARTIAL" "PARTITION"
"PARTITIONED" "PARTITIONING" "PASSWORD" "PATH" "PATTERN" "PCTFREE" "PER" "PERCENT"
 "PERCENTILE" "PERCENTILE_CONT" "PERCENTILE_DISC" "PERCENT_RANK" "PERIOD" "PERM"
"PERMANENT" "PIECESIZE" "PIVOT" "PLACING" "PLAN" "PORTION" "POSITION" "POSITION_REGEX"
 "POSTFIX" "POWER" "PRECEDES" "PRECISION" "PREFIX" "PREORDER"
"PREPARE" "PRESERVE" "PREVVAL" "PRIMARY" "PRINT" "PRIOR" "PRIQTY" "PRIVATE"
 "PRIVILEGES" "PROC" "PROCEDURE" "PROFILE" "PROGRAM" "PROPORTIONAL"
"PROTECTION" "PSID" "PTF" "PUBLIC" "PURGE" "QUALIFIED" "QUALIFY" "QUANTILE" "QUERY"
 "QUERYNO" "RADIANS" "RAISERROR" "RANDOM" "RANGE" "RANGE_N" "RANK"
```

```
"RAW" "READ" "READS" "READTEXT" "READ_WRITE" "REAL" "RECONFIGURE" "RECURSIVE" "REF"
 "REFERENCES" "REFERENCING" "REFRESH" "REGEXP" "REGR_AVGX" "REGR_AVGY"
"REGR_COUNT" "REGR_INTERCEPT" "REGR_R2" "REGR_SLOPE" "REGR_SXX" "REGR_SXY" "REGR_SYY"
 "RELATIVE" "RELEASE" "RENAME" "REPEAT" "REPLACE" "REPLICATION"
"REPOVERRIDE" "REQUEST" "REQUIRE" "RESIGNAL" "RESOURCE" "RESTART" "RESTORE" "RESTRICT"
 "RESULT" "RESULT_SET_LOCATOR" "RESUME" "RET" "RETRIEVE" "RETURN"
"RETURNING" "RETURNS" "REVALIDATE" "REVERT" "REVOKE" "RIGHT" "RIGHTS" "RLIKE" "ROLE"
 "ROLLBACK" "ROLLFORWARD" "ROLLUP" "ROUND_CEILING" "ROUND_DOWN"
"ROUND_FLOOR" "ROUND_HALF_DOWN" "ROUND_HALF_EVEN" "ROUND_HALF_UP" "ROUND_UP" "ROUTINE"
 "ROW" "ROWCOUNT" "ROWGUIDCOL" "ROWID" "ROWNUM" "ROWS" "ROWSET"
"ROW_NUMBER" "RULE" "RUN" "RUNNING" "SAMPLE" "SAMPLEID" "SAVE" "SAVEPOINT" "SCHEMA"
 "SCHEMAS" "SCOPE" "SCRATCHPAD" "SCROLL" "SEARCH" "SECOND" "SECONDS"
"SECOND_MICROSECOND" "SECQTY" "SECTION" "SECURITY" "SECURITYAUDIT" "SEEK" "SEL"
 "SELECT" "SEMANTICKEYPHRASETABLE" "SEMANTICSIMILARITYDETAILSTABLE"
"SEMANTICSIMILARITYTABLE" "SENSITIVE" "SEPARATOR" "SEQUENCE" "SESSION" "SESSION_USER"
 "SET" "SETRESRATE" "SETS" "SETSESSRATE" "SETUSER" "SHARE" "SHOW"
"SHUTDOWN" "SIGNAL" "SIMILAR" "SIMPLE" "SIN" "SINH" "SIZE" "SKEW" "SKIP" "SMALLINT"
 "SOME" "SOUNDEX" "SOURCE" "SPACE" "SPATIAL" "SPECIFIC" "SPECIFICTYPE"
"SPOOL" "SQL" "SQLEXCEPTION" "SQLSTATE" "SQLTEXT" "SQLWARNING" "SQL_BIG_RESULT"
 "SQL_CALC_FOUND_ROWS" "SQL_SMALL_RESULT" "SQRT" "SS" "SSL" "STANDARD"
"START" "STARTING" "STARTUP" "STAT" "STATE" "STATEMENT" "STATIC" "STATISTICS" "STAY"
 "STDDEV_POP" "STDDEV_SAMP" "STEPINFO" "STOGROUP" "STORED" "STORES"
"STRAIGHT_JOIN" "STRING_CS" "STRUCTURE" "STYLE" "SUBMULTISET" "SUBSCRIBER" "SUBSET"
 "SUBSTR" "SUBSTRING" "SUBSTRING_REGEX" "SUCCEEDS" "SUCCESSFUL"
"SUM" "SUMMARY" "SUSPEND" "SYMMETRIC" "SYNONYM" "SYSDATE" "SYSTEM" "SYSTEM_TIME"
 "SYSTEM_USER" "SYSTIMESTAMP" "TABLE" "TABLESAMPLE" "TABLESPACE" "TAN"
"TANH" "TBL_CS" "TEMPORARY" "TERMINATE" "TERMINATED" "TEXTSIZE" "THAN" "THEN"
 "THRESHOLD" "TIME" "TIMESTAMP" "TIMEZONE_HOUR" "TIMEZONE_MINUTE" "TINYBLOB"
"TINYINT" "TINYTEXT" "TITLE" "TO" "TOP" "TRACE" "TRAILING" "TRAN" "TRANSACTION"
 "TRANSLATE" "TRANSLATE_CHK" "TRANSLATE_REGEX" "TRANSLATION" "TREAT"
"TRIGGER" "TRIM" "TRIM_ARRAY" "TRUE" "TRUNCATE" "TRY_CONVERT" "TSEQUAL" "TYPE" "UC"
 "UESCAPE" "UID" "UNDEFINED" "UNDER" "UNDO" "UNION" "UNIQUE"
"UNKNOWN" "UNLOCK" "UNNEST" "UNPIVOT" "UNSIGNED" "UNTIL" "UPD" "UPDATE" "UPDATETEXT"
 "UPPER" "UPPERCASE" "USAGE" "USE" "USER" "USING" "UTC_DATE"
"UTC_TIME" "UTC_TIMESTAMP" "VALIDATE" "VALIDPROC" "VALUE" "VALUES" "VALUE_OF"
 "VARBINARY" "VARBYTE" "VARCHAR" "VARCHAR2" "VARCHARACTER" "VARGRAPHIC"
"VARIABLE" "VARIADIC" "VARIANT" "VARYING" "VAR_POP" "VAR_SAMP" "VCAT" "VERBOSE"
 "VERSIONING" "VIEW" "VIRTUAL" "VOLATILE" "VOLUMES" "WAIT" "WAITFOR"
"WHEN" "WHENEVER" "WHERE" "WHILE" "WIDTH_BUCKET" "WINDOW" "WITH" "WITHIN"
 "WITHIN_GROUP" "WITHOUT" "WLM" "WORK" "WRITE" "WRITETEXT" "XMLCAST" "XMLEXISTS"
"XMLNAMESPACES" "XOR" "YEAR" "YEARS" "YEAR_MONTH" "ZEROFILL" "ZEROIFNULL" "ZONE"
```

在 Metrics Insights 查詢上建立警示

您可在 Metrics Insights 查詢上建立警示。這有助您讓警示追蹤多個資源，且之後亦無需更新。查詢會擷取新資源和變更的資源。例如，您可以建立警示，以監看機群的 CPU 使用率，且該警示會自動評估您在警示建立後啟動的新執行個體。

在設定為 CloudWatch 跨帳戶觀察性的監控帳戶中，您的指標見解警示可以監視來源帳戶和監視帳戶本身中的資源。如需有關如何將警示查詢限制給特定帳戶或按帳戶 ID 對結果分組的詳細資訊，請參閱 Metrics Insights 查詢元件和語法 中的 `WHERE` 和 `GROUP BY` 一節。

內容

- 建立 Metrics Insights 警示
- 部分資料案例

建立 Metrics Insights 警示

使用主控台在 Metrics Insights 查詢上建立警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Metrics (指標)、All metrics (所有指標)。
3. 選擇 Queries (查詢) 索引標籤。
4. (選用) 若要執行預先建置的範例查詢，請選擇 Add query (新增查詢)，然後選取要執行的查詢。或者，您也可以選擇 Editor (編輯器) 以編輯範例查詢，然後選擇 Run (執行) 來執行修改的查詢。
5. 若要建立自己的查詢，您可以使用 Builder (建置器) 檢視、Editor (編輯器) 檢視，或同時使用兩者。您可以隨時在兩個檢視之間切換，並在兩個檢視中查看工作進度。

   在 Builder (建置器) 檢視中，您可以瀏覽並選取指標命名空間、指標名稱、篩選條件、群組和排序選項。對於這些選項中的每一個，查詢建置器都會為您提供環境中可能的選項清單，以供您選擇。

   在 Editor (編輯器) 檢視中，您可以開始編寫查詢。輸入時，編輯器會根據您目前輸入的字元提供建議。

   > **Important**
   >
   > 若要在 Metrics Insights 查詢上設定警示，則該查詢必須傳回單一時間序列。若其包含 GROUP BY 陳述式，則 GROUP BY 陳述式必須包裝在指標數學表達式中，且此表達式僅會傳回一個時間序列作為表達式的最終結果。

6. 當您對您的查詢感到滿意時，請選擇 Run (執行)。
7. 選擇 Create alarm (建立警示)。
8. 在 Conditions (條件) 下，指定以下內容：

   a. 針對 Whenever ***metric*** is (指標為...時)，請指定指標是否必須大於、小於，或等於閾值。在 than... (於...) 下，指定閾值。

   b. 選擇 Additional configuration (其他組態)。針對 Datapoints to alarm (要警示的資料點)，請指定 (資料點) 必須處於 ALARM 狀態多少評估期間，才會觸發警示。如果此處的兩個值相符，您便可以建立警示，在許多連續期間違規時移至 ALARM 狀態。

      若要建立 N 個中有 M 個警示，請針對第一個值，指定低於您為第二個值所指定值的值。如需詳細資訊，請參閱 評估警示。

   c. 針對 Missing data treatment (遺失資料處理)，選擇警示在遺失某些資料點時的行為。如需詳細資訊，請參閱 設定 CloudWatch 警示如何處理遺失的資料。
9. 選擇下一步。
10. 在 Notification (通知) 下，選取 SNS 主題來在警示處於 ALARM 狀態、OK 狀態或 INSUFFICIENT_DATA 狀態時進行通知。

    若要讓警示針對相同的警示狀態或不同警示狀態傳送多個通知，請選擇 Add notification (新增通知)。

    若要讓警示不傳送通知，請選擇 Remove (移動)。
11. 若要讓警示執行 Auto Scaling、 EC2 或 Systems Manager 動作，請選擇適當的按鈕，然後選擇警示狀態及要執行的動作。警示只能在進入 ALARM 狀態時執行 Systems Manager 動作。如需有關「Systems Manager」動作的詳細資訊，請參 CloudWatch 閱設定為 OpsItems 從警示建立和事件建立。

    > **Note**
    >
    > 若要建立執行 SSM Incident Manager 動作的警示，您必須具備特定許可。如需詳細資訊，請參閱AWS 系統管理員事件管理員的身分識別原則範例。
12. 完成時，請選擇下一步。
13. 輸入警示的名稱與說明。名稱只能包含 ASCII 字元。然後選擇下一步。
14. 在 Preview and create (預覽及建立) 下，請確認資訊和條件都是您希望的內容，然後選擇 Create alarm (建立警示)。

若要在指標見解查詢上建立警示，請使用 AWS CLI

- 使用 `put-metric-alarm` 命令，在 `metrics` 參數中指定 Metrics Insights 查詢。例如，若任何執行個體的 CPU 使用率超過 50%，則下列命令會讓警示進入 ALARM 狀態。

```
aws cloudwatch put-metric-alarm --alarm-name Metrics-Insights-alarm --
evaluation-periods 1 --comparison-operator GreaterThanThreshold --metrics
 '[{"Id":"m1","Expression":"SELECT MAX(CPUUtilization) FROM SCHEMA(\"AWS/EC2\",
 InstanceId)", "Period":60}]' --threshold 50
```

部分資料案例

如果用於警示的 Metrics Insights 查詢與 10,000 個以上指標相符，則會根據查詢找到的前 10,000 個指標來評估警示。這表示系統正在依據部分資料評估警示。

您可以使用下列方法了解 Metrics Insights 警示目前是否正在根據部分資料評估其警示狀態：

- 在主控台中，若您選擇某個警示以查看 Details (詳細資料) 頁面，則該頁面上會顯示 Evaluation warning: Not evaluating all data (評估警告：未評估所有資料) 訊息。
- 當您使用描述警示 AWS CLI 命令或 API 時，您會`PARTIAL_DATA`在`EvaluationState`欄位中看到該值。 DescribeAlarms

警示也會在 Amazon 進入部分資料狀態 EventBridge 時向 Amazon 發佈事件，因此您可以建立 EventBridge 規則來監視這些事件。在這些事件中，`evaluationState` 欄位的值為 `PARTIAL_DATA`。以下是範例。

```
{
    "version": "0",
    "id": "12345678-3bf9-6a09-dc46-12345EXAMPLE",
    "detail-type": "CloudWatch Alarm State Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-11-08T11:26:05Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:my-alarm-name"
    ],
    "detail": {
        "alarmName": "my-alarm-name",
        "state": {
```

```
            "value": "ALARM",
            "reason": "Threshold Crossed: 3 out of the last 3 datapoints [20000.0
 (08/11/22 11:25:00), 20000.0 (08/11/22 11:24:00), 20000.0 (08/11/22 11:23:00)] were
 greater than the threshold (0.0) (minimum 1 datapoint for OK -> ALARM transition).",
            "reasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2022-11-08T11:26:05.399+0000\",\"startDate\":\"2022-11-08T11:23:00.000+0000\",
\"period\":60,\"recentDatapoints\":[20000.0,20000.0,20000.0],\"threshold\":0.0,
\"evaluatedDatapoints\":[{\"timestamp\":\"2022-11-08T11:25:00.000+0000\",\"value
\":20000.0}]}",
            "timestamp": "2022-11-08T11:26:05.401+0000",
            "evaluationState": "PARTIAL_DATA"
        },
        "previousState": {
            "value": "INSUFFICIENT_DATA",
            "reason": "Unchecked: Initial alarm creation",
            "timestamp": "2022-11-08T11:25:51.227+0000"
        },
        "configuration": {
            "metrics": [
                {
                    "id": "m2",
                    "expression": "SELECT SUM(PartialDataTestMetric) FROM
 partial_data_test",
                    "returnData": true,
                    "period": 60
                }
            ]
        }
    }
}
```

若警示的查詢包含 GROUP BY 陳述式，且該陳述式初始傳回超過 500 個時間序列，則系統會根據查詢找到的前 500 個時間序列來評估警示。不過，若您使用 ORDER BY 子句，則系統會排序查詢找到的所有時間序列，且會根據您的 ORDER BY 子句，使用有最高或最低值的 500 個時間序列來評估警示。

使用 Metrics Insights 查詢搭配指標數學

您可以使用 Metrics Insights 查詢作為指標數學函數的輸入。如需指標數學的詳細資訊，請參閱 使用指標數學。

不包含 GROUP BY (分組依據) 子句的 Metrics Insights 查詢會傳回單一時間序列。因此，其傳回的結果可以搭配使用單一時間序列作為輸入的任何指標數學函數使用。

包含 GROUP BY (分組依據) 子句的 Metrics Insights 查詢會傳回多個時間序列。因此，其傳回的結果可以搭配使用時間序列陣列作為輸入的任何指標數學函數使用。

例如，下列查詢會以時間序列陣列傳回區域中每個儲存貯體下載的位元組總數：

```
SELECT SUM(BytesDownloaded)
FROM SCHEMA("AWS/S3", BucketName, FilterId)
WHERE FilterId = 'EntireBucket'
GROUP BY BucketName
```

在控制台或GetMetricData操作中的圖形上，此查詢的結果是q1。此查詢會傳回以位元組為單位的結果，所以如果想看到改以 MB 為單位的結果，您可以使用以下數學函數：

```
q1/1024/1024
```

使用自然語言產生和更新 CloudWatch 指標見解查詢

此功能在美國東部 (維吉尼亞北部)、美國西部 (奧勒岡) 和亞太區域 (東京) 推出預覽版， CloudWatch 且可能會變更。

CloudWatch 支援自然語言查詢功能，以協助您產生和更新CloudWatch 指標見解和CloudWatch 日誌深入解析的查詢。

有了這項功能，您可以用簡單的英文詢問或描述您要尋找的 CloudWatch 資料。自然語言功能會根據您輸入的提示產生查詢，並提供查詢運作方式的 line-by-line 說明。也可以更新查詢以進一步調查您的資料。

根據您的環境，可以輸入「哪個 Amazon Elastic Compute Cloud 執行個體的網路輸出最高？」 和「根據取用的讀取顯示排名前 10 位的 Amazon DynamoDB 資料表」等提示。

若要使用此功能產生 CloudWatch 指標見解查詢，請在產生器或編輯器檢視中開啟「 CloudWatch 指標見解」查詢編輯器，然後選擇「產生查詢」。

> **Important**
>
> 若要使用自然語言查詢功能，您必須使用CloudWatchFullAccessCloudWatchReadOnlyAccess、AdministratorAccess、CloudWatchFullAccess或ReadOnlyAccess原則。

也可以在新的或現有的客戶管理政策或內嵌政策中包含 `cloudwatch:GenerateQuery` 動作。

查詢範例

本節中的範例說明如何使用自然語言功能產生及更新查詢。

> **Note**
>
> 如需有關 CloudWatch 指標見解查詢編輯器和語法的詳細資訊，請參閱CloudWatch 指標見解查詢元件和語法。

範例：產生自然語言查詢

若要使用自然語言產生查詢，請輸入提示並選擇產生新查詢。此範例顯示執行基本搜尋的查詢。

提示

以下是一個提示範例，它指示功能去搜尋耗用最多讀取容量的前 10 個 DynamoDB 表格。

```
Show top 10 DynamoDB Tables by consumed reads
```

Query

以下是自然語言功能根據提示產生的查詢範例。請注意提示在查詢前出現在註解中的方式。查詢之後，您可以閱讀描述查詢運作方式的說明。

```
# Show top 10 DynamoDB Tables by consumed reads
SELECT SUM("ConsumedReadCapacityUnits")
FROM "AWS/DynamoDB"
GROUP BY TableName
ORDER BY SUM() DESC
LIMIT 10
# This query selects the sum of consumed read capacity units for each DynamoDB table,
 groups the results by table name, orders the results from highest to lowest read
 capacity consumption, and limits the results to the top 10 tables.
```

> **Note**
>
> 若要關閉提示的外觀以及查詢運作方式的說明，請使用編輯器中的齒輪圖示。

範例：更新自然語言查詢

可以透過編輯初始提示，然後選擇更新查詢來更新查詢。

更新提示

下列範例顯示先前提示的更新版本。現在，此提示會指示功能按照傳回的位元組數排序結果，而不是搜尋消耗最多讀取容量的前 10 個 DynamoDB 表格。

```
Sort by bytes returned instead
```

更新查詢

以下是更新查詢的範例。請注意更新後的提示在更新後的查詢前出現在註解中的方式。查詢之後，您可以閱讀描述原始查詢更新方式的說明。

```
# Sort by bytes returned instead
SELECT SUM("ReturnedBytes")
FROM "AWS/DynamoDB"
GROUP BY TableName
ORDER BY SUM() DESC
LIMIT 10
# This query modifies the original query to select the sum of returned bytes instead
 of consumed read capacity units, and orders the results from highest to lowest sum of
 returned bytes, limiting the results to the top 10 tables.
```

選擇不使用您的資料以改善服務

您提供用於訓練 AI 模型並產生相關查詢的自然語言提示資料僅用於提供和維護您的服務。此資料可能會用來改善 CloudWatch 指標深入解析的品質。我們將您的信任和隱私以及內容安全性放在首位。如需詳細資訊，請參閱 AWS 服務條款和 AWS 負責任的 AI 政策。

透過建立 AI 服務退出政策，可選擇不將您的內容用於開發或改進自然語言查詢的品質。若要選擇退出所有 CloudWatch AI 功能的資料收集 (包括查詢產生功能)，您必須為 CloudWatch. 如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的 AI 服務退出政策。

SQL 推斷

CloudWatch 指標見解使用多種機制來推斷給定 SQL 查詢的意圖。

主題

- 時段
- 欄位投影
- ORDER BY 全域彙總

時段

查詢產生的時間序列資料點會根據請求的期間以時段彙整。若要彙總標準 SQL 中的值，必須定義明確的 GROUP BY 子句，以便一起收集指定期間的所有觀察值。由於這是查詢時間序列資料的標準方式，因此 CloudWatch 指標深入解析會推斷時段，而不需要表示明確的 GROUP BY 子句。

例如，當以一分鐘的期間執行查詢時，屬於該分鐘直到下一分鐘 (不包含) 的所有觀察值都會彙整到該時段的開始時間。這讓 Metrics Insights SQL 陳述式更簡潔且更簡短。

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
```

上一個查詢會傳回單一時間序列 (時間戳記-值對)，代表所有 Amazon EC2 執行個體的平均 CPU 使用率。假設請求的期間為一分鐘，傳回的每個資料點代表在特定一分鐘間隔內測量之所有觀測值的平均值 (包含開始時間、不包含結束時間)。與特定資料點相關的時間戳記是儲存貯體的開始時間

欄位投影

Metrics Insights 查詢一律會傳回時間戳記投影。您不需要在 SELECT (選取) 子句中指定時間戳記欄來取得每個相應資料點值的時間戳記。如需如何計算時間戳記的詳細資訊，請參閱 時段。

使用 GROUP BY (分組依據) 時，每個群組名稱也會在結果中推斷和投影，以便您可以將傳回的時間序列分組。

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
GROUP BY InstanceId
```

上一個查詢會傳回每個 Amazon EC2 執行個體的時間序列。每個時間序列都會在執行個體 ID 的值之後進行標記。

ORDER BY 全域彙總

使用 ORDER BY (排序依據) 時，FUNCTION() (函數 ()) 推斷您要排序的彙總函數 (查詢指標的資料點值)。彙總操作會在所有查詢的時段中，跨每個個別時間序列的所有相符資料點執行。

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
GROUP BY InstanceId
ORDER BY MAX()
LIMIT 10
```

上一個查詢會傳回每個 Amazon EC2 執行個體的 CPU 使用率，將結果集限制為 10 個項目。結果是根據請求的時段內個別時間序列的最大值來排序。ORDER BY (排序依據) 子句會在 LIMIT (限制) 之前套用，以便在超過 10 個時間序列時計算排序。

Metrics Insights 範例查詢

本節包含有用的 CloudWatch 指標見解查詢範例，您可以直接複製和使用，或在查詢編輯器中複製和修改這些查詢。部分範例已在主控台中提供，您可以藉由在 Metrics (指標) 檢視中選擇 Add query (新增查詢)，來存取這些範例。

Application Load Balancer 範例

所有負載平衡器的請求總計

```
SELECT SUM(RequestCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer)
```

前 10 名最活躍的負載平衡器

```
SELECT MAX(ActiveConnectionCount)
FROM SCHEMA("AWS/ApplicationELB", LoadBalancer)
GROUP BY LoadBalancer
ORDER BY SUM() DESC
LIMIT 10
```

AWS API 使用範例

按您帳戶中呼叫次數排名的前 20 個 AWS API

```
SELECT COUNT(CallCount)
```

```
FROM SCHEMA("AWS/Usage", Class, Resource, Service, Type)
WHERE Type = 'API'
GROUP BY Service, Resource
ORDER BY COUNT() DESC
LIMIT 20
```

CloudWatch 依呼叫排序的 API

```
SELECT COUNT(CallCount)
FROM SCHEMA("AWS/Usage", Class, Resource, Service, Type)
WHERE Type = 'API' AND Service = 'CloudWatch'
GROUP BY Resource
ORDER BY COUNT() DESC
```

DynamoDB 範例

取用讀取的前 10 個資料表

```
SELECT SUM(ProvisionedWriteCapacityUnits)
FROM SCHEMA("AWS/DynamoDB", TableName)
GROUP BY TableName
ORDER BY MAX() DESC LIMIT 10
```

依傳回位元組排序的前 10 個資料表

```
SELECT SUM(ReturnedBytes)
FROM SCHEMA("AWS/DynamoDB", TableName)
GROUP BY TableName
ORDER BY MAX() DESC LIMIT 10
```

依使用者錯誤排序的前 10 個資料表

```
SELECT SUM(UserErrors)
FROM SCHEMA("AWS/DynamoDB", TableName)
GROUP BY TableName
ORDER BY MAX() DESC LIMIT 10
```

Amazon Elastic Block Store 範例

依寫入位元組排序的前 10 個 Amazon EBS 磁碟區

```
SELECT SUM(VolumeWriteBytes)
FROM SCHEMA("AWS/EBS", VolumeId)
GROUP BY VolumeId
ORDER BY SUM() DESC
LIMIT 10
```

Amazon EBS 磁碟區的平均寫入時間

```
SELECT AVG(VolumeTotalWriteTime)
FROM SCHEMA("AWS/EBS", VolumeId)
```

Amazon EC2 範例

依最高排序的 EC2 執行個體 CPU 使用率

```
SELECT AVG(CPUUtilization)
  FROM SCHEMA("AWS/EC2", InstanceId)
  GROUP BY InstanceId
  ORDER BY AVG() DESC
```

整個機群的平均 CPU 使用率

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
```

依最高 CPU 使用率排序的前 10 個執行個體

```
SELECT MAX(CPUUtilization)
FROM SCHEMA("AWS/EC2", InstanceId)
GROUP BY InstanceId
ORDER BY MAX() DESC
LIMIT 10
```

在此情況下，CloudWatch 代理程式會收集每個應用程式的**CPUUtilization**測量結果。此查詢會針對特定應用程式名稱篩選此指標的平均值。

```
SELECT AVG(CPUUtilization)
FROM "AWS/CWAgent"
WHERE ApplicationName = 'eCommerce'
```

Amazon Elastic Container Service 範例

所有 ECS 叢集的平均 CPU 使用率

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/ECS", ClusterName)
```

依記憶體使用率排序的前 10 個叢集

```
SELECT AVG(MemoryUtilization)
FROM SCHEMA("AWS/ECS", ClusterName)
GROUP BY ClusterName
ORDER BY AVG() DESC
LIMIT 10
```

依 CPU 使用率排序的前 10 項服務

```
SELECT AVG(CPUUtilization)
FROM SCHEMA("AWS/ECS", ClusterName, ServiceName)
GROUP BY ClusterName, ServiceName
ORDER BY AVG() DESC
LIMIT 10
```

依執行任務排序的前 10 項服務 (Container Insights)

```
SELECT AVG(RunningTaskCount)
FROM SCHEMA("ECS/ContainerInsights", ClusterName, ServiceName)
GROUP BY ClusterName, ServiceName
ORDER BY AVG() DESC
LIMIT 10
```

Amazon Elastic Kubernetes Service Container Insights 範例

所有 EKS 叢集的平均 CPU 使用率

```
SELECT AVG(pod_cpu_utilization)
FROM SCHEMA("ContainerInsights", ClusterName)
```

依節點 CPU 使用率排序的前 10 個叢集

```
SELECT AVG(node_cpu_utilization)
FROM SCHEMA("ContainerInsights", ClusterName)
GROUP BY ClusterName
ORDER BY AVG() DESC LIMIT 10
```

依 Pod 記憶體使用率排序的前 10 個叢集

```
SELECT AVG(pop_memory_utilization)
FROM SCHEMA("ContainerInsights", ClusterName)
GROUP BY ClusterName
ORDER BY AVG() DESC LIMIT 10
```

依 CPU 使用率排序的前 10 個節點

```
SELECT AVG(node_cpu_utilization)
FROM SCHEMA("ContainerInsights", ClusterName, NodeName)
GROUP BY ClusterName, NodeName
ORDER BY AVG() DESC LIMIT 10
```

依記憶體使用率排序的前 10 個 Pod

```
SELECT AVG(pod_memory_utilization)
FROM SCHEMA("ContainerInsights", ClusterName, PodName)
GROUP BY ClusterName, PodName
ORDER BY AVG() DESC LIMIT 10
```

EventBridge 例子

依調用排序的前 10 個規則

```
SELECT SUM(Invocations)
FROM SCHEMA("AWS/Events", RuleName)
GROUP BY RuleName
ORDER BY MAX() DESC LIMIT 10
```

依失敗調用排序的前 10 個規則

```
SELECT SUM(FailedInvocations)
FROM SCHEMA("AWS/Events", RuleName)
```

```
GROUP BY RuleName
ORDER BY MAX() DESC LIMIT 10
```

依相符規則排序的前 10 個規則

```
SELECT SUM(MatchedEvents)
FROM SCHEMA("AWS/Events", RuleName)
GROUP BY RuleName
ORDER BY MAX() DESC LIMIT 10
```

Kinesis 範例

依寫入位元組排序的前 10 個串流

```
SELECT SUM("PutRecords.Bytes")
FROM SCHEMA("AWS/Kinesis", StreamName)
GROUP BY StreamName
ORDER BY SUM() DESC LIMIT 10
```

依串流中最早項目排序的前 10 個串流

```
SELECT MAX("GetRecords.IteratorAgeMilliseconds")
FROM SCHEMA("AWS/Kinesis", StreamName)
GROUP BY StreamName
ORDER BY MAX() DESC LIMIT 10
```

Lambda 範例

依調用次數排序的 Lambda 函數

```
SELECT SUM(Invocations)
FROM SCHEMA("AWS/Lambda", FunctionName)
GROUP BY FunctionName
ORDER BY SUM() DESC
```

依最長執行時間排序的前 10 個 Lambda 函數

```
SELECT AVG(Duration)
FROM SCHEMA("AWS/Lambda", FunctionName)
```

```
GROUP BY FunctionName
ORDER BY MAX() DESC
LIMIT 10
```

依錯誤計數排序的前 10 個 Lambda 函數

```
SELECT SUM(Errors)
FROM SCHEMA("AWS/Lambda", FunctionName)
GROUP BY FunctionName
ORDER BY SUM() DESC
LIMIT 10
```

CloudWatch 記錄檔範例

依傳入事件排序的前 10 個日誌群組

```
SELECT SUM(IncomingLogEvents)
FROM SCHEMA("AWS/Logs", LogGroupName)
GROUP BY LogGroupName
ORDER BY SUM() DESC LIMIT 10
```

依寫入位元組排序的前 10 個日誌群組

```
SELECT SUM(IncomingBytes)
FROM SCHEMA("AWS/Logs", LogGroupName)
GROUP BY LogGroupName
ORDER BY SUM() DESC LIMIT 10
```

Amazon RDS 範例

依最高 CPU 使用率排序的前 10 個 Amazon RDS 執行個體

```
SELECT MAX(CPUUtilization)
FROM SCHEMA("AWS/RDS", DBInstanceIdentifier)
GROUP BY DBInstanceIdentifier
ORDER BY MAX() DESC
LIMIT 10
```

按寫入排序的前 10 個 Amazon RDS 叢集

```
SELECT SUM(WriteIOPS)
FROM SCHEMA("AWS/RDS", DBClusterIdentifier)
GROUP BY DBClusterIdentifier
ORDER BY MAX() DESC
LIMIT 10
```

Amazon Simple Storage Service 範例

儲存貯體的平均延遲

```
SELECT AVG(TotalRequestLatency)
FROM SCHEMA("AWS/S3", BucketName, FilterId)
WHERE FilterId = 'EntireBucket'
GROUP BY BucketName
ORDER BY AVG() DESC
```

依位元組下載的前 10 個儲存貯體

```
SELECT SUM(BytesDownloaded)
FROM SCHEMA("AWS/S3", BucketName, FilterId)
WHERE FilterId = 'EntireBucket'
GROUP BY BucketName
ORDER BY SUM() DESC
LIMIT 10
```

Amazon Simple Notification Service 範例

SNS 主題發佈的訊息總數

```
SELECT SUM(NumberOfMessagesPublished)
FROM SCHEMA("AWS/SNS", TopicName)
```

依已發佈訊息排序的前 10 大主題

```
SELECT SUM(NumberOfMessagesPublished)
FROM SCHEMA("AWS/SNS", TopicName)
GROUP BY TopicName
ORDER BY SUM() DESC
LIMIT 10
```

依訊息傳遞失敗排序的前 10 大主題

```
SELECT SUM(NumberOfNotificationsFailed)
FROM SCHEMA("AWS/SNS", TopicName)
GROUP BY TopicName
ORDER BY SUM() DESC
LIMIT 10
```

Amazon SQS 範例

依可見訊息數目排列的前 10 個佇列

```
SELECT AVG(ApproximateNumberOfMessagesVisible)
FROM SCHEMA("AWS/SQS", QueueName)
GROUP BY QueueName
ORDER BY AVG() DESC
LIMIT 10
```

前 10 個最活躍的佇列

```
SELECT SUM(NumberOfMessagesSent)
FROM SCHEMA("AWS/SQS", QueueName)
GROUP BY QueueName
ORDER BY SUM() DESC
LIMIT 10
```

依最早訊息存留天數排序的前 10 個佇列

```
SELECT AVG(ApproximateAgeOfOldestMessage)
FROM SCHEMA("AWS/SQS", QueueName)
GROUP BY QueueName
ORDER BY AVG() DESC
LIMIT 10
```

Metrics Insights 限制

CloudWatch 量度洞察目前有下列限制：

- 目前，您只能查詢最近三個小時的資料。

- 單一查詢可處理不超過 10,000 個指標。這表示如果 SELECT (選取)、FROM (從) 和 WHERE (哪裡) 子句符合 10,000 個以上的指標時，查詢只會處理所找到這些指標中的前 10,000 個。
- 單個查詢可以傳回不超過 500 個時間序列。這表示如果查詢傳回超過 500 個指標，則並非所有指標都會在查詢結果中傳回。如果您使用 ORDER BY (排序依據) 子句，則會排序所有正在處理的指標，並根據您的 ORDER BY (排序依據) 子句會傳回擁有最高或最低值的 500 個指標。

  如果沒有包含 ORDER BY (排序依據) 子句，則無法控制傳回哪 500 個相符指標。
- 每個區域最多可以有 200 個指標見解警示。
- Metrics Insights 不支援高解析度資料，因為此類資料是回報的精細度低於一分鐘的指標資料。若您請求高解析度資料，請求雖不會失敗，但輸出資料會以一分鐘的精細度彙總。
- 每個GetMetricData作業只能有一個查詢，但儀表板中可以有多個小器具，每個小器具都包含一個查詢。

Metrics Insights 字彙

label

在 Metrics Insights 中，標籤是索引鍵值組，用來定義查詢範圍以傳回特定資料集，或定義將查詢結果分隔成不同時間序列的準則。標籤索引鍵類似於 SQL 中的資料欄名稱。目前，標籤必須是度 CloudWatch 量維度。

觀察

觀察值是針對指定時間之指定指標記錄的值。

對 Metrics Insights 進行疑難排解

結果包括「Other」，但我沒有以此作為維度

這表示查詢包含 GROUP BY (分組依據) 子句，該子句會指定查詢所傳回之某些指標中未使用的標籤索引鍵。在此情況下，會傳回名為 `Other` 的空值群組。不包含該標籤索引鍵的指標可能是彙總的指標，該指標會傳回跨該標籤索引鍵所有值彙總的值。

例如，假設我們有下列查詢：

```
SELECT AVG(Faults)
FROM MyCustomNamespace
GROUP BY Operation, ServiceName
```

如果某些傳回的指標不包含 ServiceName 作為維度，則這些指標會顯示為像 ServiceName 值一樣擁有 Other。

為了防止在結果中看到「Other」，請在您的 FROM (從) 子句中使用 SCHEMA (結構描述)，如下列範例所示：

```
SELECT AVG(Faults)
FROM SCHEMA(MyCustomNamespace, Operation)
GROUP BY Operation, ServiceName
```

這會將傳回的結果限制為僅同時具有 Operation 和 ServiceName 維度的指標。

我的圖形中最舊的時間戳記比其他時間戳記的指標值低

CloudWatch 指標見解目前僅支援最近三小時的資料。當您以大於一分鐘的時段繪製時，可能會出現最舊的資料點與預期值不同的情況。這是因為「指標見解」查詢只會傳回最近三小時的資料。在這種情況下，查詢中最舊的資料點只會傳回過去三小時邊界內測量的觀測值，而不是傳回該資料點時段內的所有觀測值。

使用指標瀏覽器依其標籤和屬性監控資源

指標瀏覽器是標籤型工具，可讓您依照標籤和資源屬性篩選、彙總和視覺化您的指標，進而增強服務的可觀察性。這樣一來，可為您提供靈活且動態的疑難排解體驗，進而讓您一次建立多個圖形，並使用這些圖形來建置您的應用程式運作狀態儀表板。

量度總管視覺效果是動態的，因此，如果在您建立量度總管 Widget 並將其新增至 CloudWatch 儀表板之後建立相符的資源，則新資源會自動顯示在檔案總管 Widget 中。

例如，如果您的所有 EC2 生產執行個體都具有 **production** 標籤，您可以使用指標瀏覽器來篩選和彙總所有這些執行個體的指標，以了解其運作狀態和效能。如果稍後建立具有相符標籤的新執行個體，它會自動新增至指標瀏覽器小工具。

> **Note**
>
> 度量資源管理器提供 point-in-time 體驗。視覺效果不會顯示已終止或不再有您所指定屬性或標籤的資源。不過，您仍然可以在量度檢視中找到這些資源的 CloudWatch 指標。

使用指標瀏覽器，您可以選擇如何彙總符合準則的資源中的指標，以及是否要將其全部顯示在單一圖形或單一指標瀏覽器小工具中的不同圖形上。

指標瀏覽器包含範本，您只要按一下即可查看實用的視覺化圖形，您也可以擴展這些範本，以建立完全自訂的指標瀏覽器小工具。

指標資源管理器支援由 CloudWatch 代理程式發佈的 EC2 指標 AWS 和 EC2 指標，包括記憶體、磁碟和 CPU 指標。若要使用指標總管來查看 CloudWatch 代理程式所發佈的測量結果，您可能必須更新 CloudWatch 代理程式組態檔。如需詳細資訊，請參閱 CloudWatch 指標總管的代理程式組態

若要使用指標瀏覽器建立視覺化，並選擇性地將其新增至儀表板，請遵循下列步驟。

若要使用指標瀏覽器建立視覺化

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Explorer (瀏覽器)。
3. 執行以下任意一項：
   - 若要使用範本，請選取目前顯示 Empty Explorer (空的瀏覽器) 的方塊。

     視範本而定，瀏覽器可能會立即顯示指標圖形。如果沒有，請在 From (來源) 方塊中，選擇一個或兩個標籤或屬性，然後應該會出現資料。如果沒有，請使用頁面頂端的選項，在圖形中顯示較長的時間範圍。
   - 若要建立自訂視覺化，請在 Metrics (指標)中，從服務選擇單一指標或所有可用指標。

     選擇指標後，您可以選擇性地重複此步驟，以新增更多指標。
4. 針對每個選取的測量結果，會在測量結果名稱後立即 CloudWatch 顯示要使用的統計值。若要對其進行變更，請選擇統計資料名稱，然後選擇您要的統計資料。
5. 在 From (來源) 下，選擇標籤或資源屬性以篩選結果。

   執行這項操作之後，您可以選擇性地重複此步驟，以選擇更多標籤或資源特性。

   如果您選擇同一屬性的多個值 (例如兩個 EC2 執行個體類型)，則瀏覽器會顯示符合任一所選屬性的所有資源。它會被視為 OR 運算。

   如果您選擇不同的屬性或標籤，例如 **`Production`** 標籤和 M5 執行個體類型時，只會顯示符合所有這些選項的資源。它會被視為 AND 運算。
6. (選用) 對於 Aggregate by (彙總依據)，選擇用來彙總指標的統計資料。然後，在 for (方式) 旁，從清單中選擇彙總指標的方式。您可以將目前顯示的所有資源彙總在一起，或依單一標籤或資源屬性彙總。

   根據您選擇聚總的方式，結果可能是單一時間序列或多個時間序列。

7. 在 Split by (分割依據) 下，您可以選擇將具有多個時間序列的單一圖形分割為多個圖形。分割可以透過各種標準進行，您可以在 Split by (分割依據) 下進行選擇。
8. 在 Graph options (圖形選項) 下，您可以透過變更週期、圖形類型、圖例位置和版面配置來精簡圖形。
9. 若要將此視覺效果新增為 CloudWatch 控制面板的 Widget，請選擇 [新增至儀表板]。

CloudWatch 指標總管的代理程式組態

若要啟用指標總管以探索 CloudWatch 代理程式發佈的 EC2 指標，請確定 CloudWatch 代理程式組態檔包含下列值：

- 在 `metrics` 區段中，確定 `aggregation_dimensions` 參數包括 `["InstanceId"]`。也可以包含其他維度。
- 在 `metrics` 區段中，確定 `append_dimensions` 參數包括 `{"InstanceId":"${aws:InstanceId}"}` 行。也可以包含其他行。
- 於 `metrics` 區段中的 `metrics_collected` 區段內，檢查您希望指標瀏覽器探索的每個資源類型區段，例如 `cpu`、`disk` 及 `memory` 區段。請確定這些區段中的每個區段都有 `"resources": [ "*"] line.`。
- 在 `metrics_collected` 區段的 `cpu` 區段中，請確定包含 `"totalcpu": true` 行。
- 您必須針對 CloudWatch 代理程式收集的測量結果使用預設`CWAgent`命名空間，而非自訂命名空間。

上一個清單中的設定會導致 CloudWatch 代理程式發佈磁碟、CPU 和其他資源的彙總指標，這些指標可以在量度總管中繪製，以供使用它的所有執行個體使用該指標的執行個體。

這些設定會重新發佈您先前設定為發佈多個維度的指標，並增加您的指標成本。

如需編輯 CloudWatch 代理程式組態檔的詳細資訊，請參閱 手動建立或編輯 CloudWatch 代理程式組態檔。

使用指標串流

您可以使用指標串流，持續將 CloudWatch 指標串流至您選擇的目的地，並具有 near-real-time 傳遞和低延遲。支援的目的地包括 Amazon 簡易儲存服務等 AWS 目的地，以及數個第三方服務供應商目的地

CloudWatch 測量結果串流有三種主要使用情境：

- 使用 Firehose 自訂設定 — 建立指標串流並將其導向至 Amazon Data Firehose 交付串流，以便將您的 CloudWatch 指標傳送到您想要的位置。您可以將它們串流到 Amazon S3 等資料湖，或是任何 Firehose 支援的目的地或端點，包括第三方供應商。本機支援 JSON、 OpenTelemetry 1.0.0 和 OpenTelemetry 0.7.0 格式，或者您也可以在 Firehose 交付串流中設定轉換，將資料轉換為不同的格式 (例如實木地板)。透過指標串流，您可以持續更新監控資料，或將此 CloudWatch 指標資料與計費和效能資料結合，以建立豐富的資料集。然後，您可以使用 Amazon Athena 之類的工具，深入了解成本最佳化、資源效能和資源使用率。
- Quick S3 Setup – 透過快速設定程序串流至 Amazon Simple Storage 服務。依預設， CloudWatch 會建立串流所需的資源。支援 JSON 格式、 OpenTelemetry 1.0.0 和 OpenTelemetry 0.7.0 格式。
- 快速 AWS 合作夥伴設定 — CloudWatch 為某些第三方合作夥伴提供快速的設定體驗。您可以使用第三方服務供應商，使用串流 CloudWatch 資料來監控、疑難排解和分析應用程式。使用快速合作夥伴設定工作流程時，您只需要為目的地提供目的地 URL 和 API 金鑰，並 CloudWatch 處理其餘的設定。下列第三方供應商可使用快速合作夥伴設定：

  - Datadog
  - Dynatrace
  - New Relic
  - Splunk Observability Cloud
  - SumoLogic

您可以串流所有指 CloudWatch 標，或使用篩選器僅串流指定的指標。每個指標串流最多可包含 1000 個篩選條件，其中包含或排除指標命名空間或特定指標。單一指標串流只能包含或排除篩選條件，但不能同時包含。

建立指標串流後，如果建立符合篩選條件的新指標，則新指標會自動包含在串流中。

每個帳戶或每個區域的指標串流數目沒有限制，要串流的指標更新數目也沒有限制。

每個串流都可以使用 JSON 格式、 OpenTelemetry 1.0.0 或 OpenTelemetry 0.7.0 格式。您可以隨時編輯度量資料流的輸出格式，例如從 OpenTelemetry 0.7.0 升級到 OpenTelemetry 1.0.0。如需輸出格式的詳細資訊，請參閱 指標串流輸出格式。

對於監控帳戶中的指標串流，您可以選擇是否包含來自連結至該監控帳戶之來源帳戶的指標。如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。

指標串流始終包含 `Minimum`、`Maximum`、`SampleCount` 以及 `Sum` 統計數字。您也可以選擇包含額外統計數字，但需另行付費。如需詳細資訊，請參閱 可供串流的統計數字。

指標串流定價是根據指標更新的數目而定的。您也會對用於量度串流的傳送串流產生 Firehose 的費用。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

主題

- 設定指標串流
- 可供串流的統計數字
- 指標串流操作與維護
- 使用指標監控您的 CloudWatch 指標串流
- CloudWatch 與 Firehose 之間的信任
- 指標串流輸出格式
- 故障診斷

設定指標串流

請使用以下各節中的步驟來設定 CloudWatch 量度串流。

建立量度串流之後，指標資料在目的地顯示所需的時間取決於 Firehose 傳送串流上設定的緩衝設定。緩衝以最大酬載大小或最大等待時間表示，以先到達者為準。如果將這些值設定為最小值 (60 秒、1MB)，如果選取的 CloudWatch 命名空間具有作用中的量度更新，則預期的延遲會在 3 分鐘內。

在 CloudWatch 指標串流中，資料會每分鐘傳送一次。資料可能不會按順序到達最終目的地。指定命名空間中的所有指定測量結果都會在指標串流中傳送，但時間戳記超過兩天的測量結果除外。

對於您串流的每個指標名稱與命名空間組合，系統都會對該指標名稱與命名空間的所有維度組合進行串流處理。

對於監控帳戶中的指標串流，您可以選擇是否包含來自連結至該監控帳戶之來源帳戶的指標。如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。

若要建立和管理測量結果串流，您必須登入具有CloudWatchFullAccess原則和`iam:PassRole`權限的帳戶，或是具有下列權限清單的帳戶：

- `iam:PassRole`
- `cloudwatch:PutMetricStream`
- `cloudwatch:DeleteMetricStream`
- `cloudwatch:GetMetricStream`
- `cloudwatch:ListMetricStreams`

- cloudwatch:StartMetricStreams
- cloudwatch:StopMetricStreams

如果您要 CloudWatch 設定指標串流所需的 IAM 角色，則還必須擁有iam:CreateRole和iam:PutRolePolicy許可。

> **Important**
>
> cloudwatch:PutMetricStream具有的使用者可以存取正在串流的 CloudWatch 指標資料，即使他們沒有cloudwatch:GetMetricData權限也是如此。

主題

- 使用 Firehose 進行自訂設定
- 使用 Quick Amazon S3 設定
- 快速合作夥伴設定

使用 Firehose 進行自訂設定

使用此方法建立指標串流，並將其導向 Amazon Data Firehose 交付串流，以便將 CloudWatch 指標傳送到您想要的位置。您可以將它們串流到 Amazon S3 等資料湖，或是任何 Firehose 支援的目的地或端點，包括第三方供應商。

本機支援 JSON、 OpenTelemetry 1.0.0 和 OpenTelemetry 0.7.0 格式，或者您也可以在 Firehose 交付串流中設定轉換，將資料轉換為不同的格式 (例如實木地板)。透過指標串流，您可以持續更新監控資料，或將此 CloudWatch 指標資料與計費和效能資料結合，以建立豐富的資料集。然後，您可以使用 Amazon Athena 之類的工具，深入了解成本最佳化、資源效能和資源使用率。

您可以使用 CloudWatch 控制台 AWS CLI、 AWS CloudFormation、或 AWS Cloud Development Kit (AWS CDK) 來設定量度串流。

您用於指標串流的 Firehose 交付串流必須位於設定量度串流的相同帳戶和相同區域。若要實現跨區域功能，您可以將 Firehose 交付串流設定為串流至不同帳戶或不同區域的最終目的地。

CloudWatch 控制台

本節說明如何使用主 CloudWatch 控台使用 Firehose 設定量度串流。

若要使用 Firehose 設定自訂量度串流

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇 Metrics (指標)、Streams (串流)。然後選擇 Create metric stream (建立指標串流)。
3. (選擇性) 如果您登入的帳戶已設定為 CloudWatch 跨帳戶觀察性的監視帳戶，您可以選擇是否要在此指標串流中包含連結來源帳戶的量度。若要包含來自來源帳戶的指標，請選擇 Include source account metrics (包含來源帳戶指標)。
4. 選擇使用 Firehose 的自定義設置。
5. 對於 「選取您的 Kinesis Data Firehose 串流」，請選取要使用的 Firehose 傳送串流。它必須在同一個帳戶中。此選項的預設格式為 OpenTelemetry 0.7.0，但您可以稍後在此程序中變更格式。

   然後在「選取您的 Firehose 傳送串流」下選取要使用的 Firehose 交付串流。
6. (選擇性) 您可以選擇選取現有的服務角色來使用現有的 IAM 角色，而不必為您 CloudWatch建立新的 IAM 角色。
7. (選用) 若要變更案例的預設格式的輸出格式，請選擇 Change output format (變更輸出格式)。支援的格式為 JSON、 OpenTelemetry 1.0.0 和 OpenTelemetry 0.7.0。
8. 對於要串流的量度，請選擇「所有量度」或「選取量度」。

   如果您選擇 [所有量度]，則此帳戶中的所有指標都會包含在串流中。

   請仔細考慮是否要串流所有指標，因為串流的指標數量越多，指標串流的費用就越高。

   如果您選擇 「選取量度」，請執行下列其中一個動作：

   - 若要串流大部分的測量結果命名空間，請選擇排除並選取要排除的命名空間或測量結果。當您在 Exclude 中指定命名空間時，您可以選擇性地從該命名空間中選取一些要排除的特定度量。如果您選擇排除命名空間，但未選取該命名空間中的量度，則會排除該命名空間中的所有量度。
   - 若只要在測量結果串流中包含少數測量結果命名空間或測量結果，請選擇「包含」，然後選取要包含的命名空間或測量結果。如果您選擇包含命名空間，但未在該命名空間中選取量度，則會包含該命名空間中的所有量度。
9. (選擇性) 若要為「最小值」、「上限」和「總和」以外的某些測量結果串流其他統計資料，請選擇「新增其他統計值」 SampleCount 您可以選擇 Add recommended metrics (新增建議的指標) 以新增常用統計數字，或手動選取要為其串流額外統計數字的命名空間和指標名稱。然後選取要串流的額外統計數字。

選擇要為其串流不同額外統計數字集的另一組指標，然後選擇 Add additional statistics (新增額外統計數字)。每個指標可以包含多達 20 個額外統計數字，而一個指標串流中具有多達 100 個能夠包含額外統計數字的指標。

串流額外統計數字會產生更多費用。如需詳細資訊，請參閱 可供串流的統計數字。

如需有關額外統計數字的定義，請參閱 CloudWatch 統計定義。

10. (選用) 在 Metric stream name (指標串流名稱) 下，自訂新指標串流的名稱。
11. 選擇 Create metric stream (建立指標串流)。

AWS CLI 或 AWS API

請使用下列步驟建立 CloudWatch 度量資料流。

使用 AWS CLI 或 AWS API 建立量度串流

1. 如果您要串流到 Simple Storage Service (Amazon S3)，請先建立儲存貯體。如需詳細資訊，請參閱建立儲存貯體。
2. 建立 Firehose 傳送串流。如需詳細資訊，請參閱建立 Firehose 串流。
3. 建立可 CloudWatch 寫入 Firehose 交付串流的 IAM 角色。如需此角色之內容的詳細資訊，請參閱 CloudWatch 與 Firehose 之間的信任。
4. 使用 `aws cloudwatch put-metric-stream` CLI 命令或 `PutMetricStream` API 建立指 CloudWatch 標串流。

AWS CloudFormation

您可以使用 AWS CloudFormation 來設定指標串流。如需詳細資訊，請參閱 AWS::CloudWatch::MetricStream。

用於 AWS CloudFormation 建立度量串流

1. 如果您要串流到 Simple Storage Service (Amazon S3)，請先建立儲存貯體。如需詳細資訊，請參閱建立儲存貯體。
2. 建立 Firehose 傳送串流。如需詳細資訊，請參閱建立 Firehose 串流。
3. 建立可 CloudWatch 寫入 Firehose 交付串流的 IAM 角色。如需此角色之內容的詳細資訊，請參閱 CloudWatch 與 Firehose 之間的信任。

4. 在中建立串流 AWS CloudFormation。如需詳細資訊，請參閱 AWS::CloudWatch::MetricStream。

AWS Cloud Development Kit (AWS CDK)

您可以使用 AWS Cloud Development Kit (AWS CDK) 來設定指標串流。

使用建 AWS CDK 立度量資料流

1. 如果您要串流到 Simple Storage Service (Amazon S3)，請先建立儲存貯體。如需詳細資訊，請參閱建立儲存貯體。
2. 建立 Firehose 傳送串流。如需詳細資訊，請參閱建立 Amazon 資料 Firehose 交付串流。
3. 建立可 CloudWatch 寫入 Firehose 交付串流的 IAM 角色。如需此角色之內容的詳細資訊，請參閱 CloudWatch 與 Firehose 之間的信任。
4. 建立指標串流。度量串流資源在中提供名 AWS CDK `CfnMetricStream`為的層級 1 (L1) 建構。如需詳細資訊，請參閱使用 L1 建構。

使用 Quick Amazon S3 設定

如果您想要快速設定 Amazon S3 的串流，而且不需要超出支援 JSON、 OpenTelemetry 1.0.0 和 OpenTelemetry 0.7.0 格式的任何格式轉換，快速 S3 安裝方法可以很好地運作。 CloudWatch 將建立所有必要的資源，包括 Firehose 交付串流和必要的 IAM 角色。此選項的預設格式為 JSON，但您可在設定串流時變更格式。

或者，如果您希望最終格式為 Parquet 格式或 Optimized Row Columnar (ORC)，您應改為遵循 使用 Firehose 進行自訂設定 中的步驟。

CloudWatch 控制台

本節說明如何使用主 CloudWatch 控台使用快速 S3 設定來設定指標串流 Amazon S3。

使用 Quick S3 Setup 設定指標串流

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇 Metrics (指標)、Streams (串流)。然後選擇 Create metric stream (建立指標串流)。
3. (選擇性) 如果您登入的帳戶已設定為 CloudWatch 跨帳戶觀察性的監視帳戶，您可以選擇是否要在此指標串流中包含連結來源帳戶的量度。若要包含來自來源帳戶的指標，請選擇 Include source account metrics (包含來源帳戶指標)。

4. 選擇 Quick S3 Setup。 CloudWatch 將建立所有必要的資源，包括 Firehose 交付串流和必要的 IAM 角色。此選項的預設格式為 JSON，但您可以稍後在此程序中變更格式。
5. (選擇性) 選擇選取現有資源以使用現有的 S3 儲存貯體或現有 IAM 角色，而不必為您 CloudWatch 建立新角色。
6. (選用) 若要變更案例的預設格式的輸出格式，請選擇 Change output format (變更輸出格式)。支援的格式為 JSON、 OpenTelemetry 1.0.0 和 OpenTelemetry 0.7.0。
7. 對於要串流的量度，請選擇「所有量度」或「選取量度」。

   如果您選擇 [所有量度]，則此帳戶中的所有指標都會包含在串流中。

   請仔細考慮是否要串流所有指標，因為串流的指標數量越多，指標串流的費用就越高。

   如果您選擇 「選取量度」，請執行下列其中一個動作：

   - 若要串流大部分的測量結果命名空間，請選擇排除並選取要排除的命名空間或測量結果。當您在 Exclude 中指定命名空間時，您可以選擇性地從該命名空間中選取一些要排除的特定度量。如果您選擇排除命名空間，但未選取該命名空間中的量度，則會排除該命名空間中的所有量度。
   - 若只要在測量結果串流中包含少數測量結果命名空間或測量結果，請選擇「包含」，然後選取要包含的命名空間或測量結果。如果您選擇包含命名空間，但未在該命名空間中選取量度，則會包含該命名空間中的所有量度。
8. (選擇性) 若要為「最小值」、「上限」和「總和」以外的某些測量結果串流其他統計資料，請選擇「新增其他統計值」 SampleCount 您可以選擇 Add recommended metrics (新增建議的指標) 以新增常用統計數字，或手動選取要為其串流額外統計數字的命名空間和指標名稱。然後選取要串流的額外統計數字。

   選擇要為其串流不同額外統計數字集的另一組指標，然後選擇 Add additional statistics (新增額外統計數字)。每個指標可以包含多達 20 個額外統計數字，而一個指標串流中具有多達 100 個能夠包含額外統計數字的指標。

   串流額外統計數字會產生更多費用。如需詳細資訊，請參閱 可供串流的統計數字。

   如需有關額外統計數字的定義，請參閱 CloudWatch 統計定義。
9. (選用) 在 Metric stream name (指標串流名稱) 下，自訂新指標串流的名稱。
10. 選擇 Create metric stream (建立指標串流)。

快速合作夥伴設定

CloudWatch 為下列第三方合作夥伴提供快速的設定體驗。若要使用此工作流程，您只需要為目的地提供目的地 URL 和 API 金鑰。 CloudWatch 處理其餘的設定，包括建立 Firehose 交付串流和必要的 IAM 角色。

> **Important**
>
> 在您使用快速合作夥伴設定來建立指標串流之前，強烈建議您先閱讀下列清單中連結的合作夥伴文件。

- Datadog
- Dynatrace
- New Relic
- Splunk Observability Cloud
- SumoLogic

當您將指標串流設定到這些合作夥伴之一時，系統會使用某些預設設定來建立串流，如以下章節所列示。

主題

- 使用快速合作夥伴設定來設定指標串流
- Datadog 串流預設值
- Dynatrace 串流預設值
- New Relic 串流預設值
- Splunk 可觀測性雲端串流預設值
- Sumo Logic 串流預設值

使用快速合作夥伴設定來設定指標串流

CloudWatch 為某些第三方合作夥伴提供快速設定選項。在開始本節中的步驟之前，您必須先了解合作夥伴的特定資訊。此資訊可能包括您的合作夥伴目的地的目的地 URL 及/或 API 金鑰。您還應閱讀上一節中連結的合作夥伴網站上的文件，以及下列章節中列出的該合作夥伴的預設值。

若要串流至快速設定不支援的第三方目的地，您可以依照中的指示使用 Firehose 設定串流中的指示，然後使用 Firehose 進行自訂設定將這些量度從 Firehose 傳送到最終目的地。

使用快速合作夥伴設定來建立到第三方供應商的指標串流

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇 Metrics (指標)、Streams (串流)。然後選擇 Create metric stream (建立指標串流)。
3. (選擇性) 如果您登入的帳戶已設定為 CloudWatch 跨帳戶觀察性的監視帳戶，您可以選擇是否要在此指標串流中包含連結來源帳戶的量度。若要包含來自來源帳戶的指標，請選擇 Include source account metrics (包含來源帳戶指標)。
4. 選擇快速 Amazon Web Services 合作夥伴設定
5. 選取您想要串流指標的目標合作夥伴。
6. 針對端點 URL，輸入目的地 URL。
7. 針對存取金鑰或 API 金鑰，輸入合作夥伴的存取金鑰。並非所有合作夥伴都需要存取金鑰。
8. 對於要串流的量度，請選擇「所有量度」或「選取量度」。

   如果您選擇 [所有量度]，則此帳戶中的所有指標都會包含在串流中。

   請仔細考慮是否要串流所有指標，因為串流的指標數量越多，指標串流的費用就越高。

   如果您選擇 「選取量度」，請執行下列其中一個動作：

   - 若要串流大部分的測量結果命名空間，請選擇排除並選取要排除的命名空間或測量結果。當您在 Exclude 中指定命名空間時，您可以選擇性地從該命名空間中選取一些要排除的特定度量。如果您選擇排除命名空間，但未選取該命名空間中的量度，則會排除該命名空間中的所有量度。
   - 若只要在測量結果串流中包含少數測量結果命名空間或測量結果，請選擇「包含」，然後選取要包含的命名空間或測量結果。如果您選擇包含命名空間，但未在該命名空間中選取量度，則會包含該命名空間中的所有量度。
9. (選擇性) 若要為「最小值」、「上限」和「總和」以外的某些測量結果串流其他統計資料，請選擇「新增其他統計值」 SampleCount 您可以選擇 Add recommended metrics (新增建議的指標) 以新增常用統計數字，或手動選取要為其串流額外統計數字的命名空間和指標名稱。然後選取要串流的額外統計數字。

選擇要為其串流不同額外統計數字集的另一組指標，然後選擇 Add additional statistics (新增額外統計數字)。每個指標可以包含多達 20 個額外統計數字，而一個指標串流中具有多達 100 個能夠包含額外統計數字的指標。

串流額外統計數字會產生更多費用。如需詳細資訊，請參閱 可供串流的統計數字。

如需有關額外統計數字的定義，請參閱 CloudWatch 統計定義。

10. (選用) 在 Metric stream name (指標串流名稱) 下，自訂新指標串流的名稱。
11. 選擇 Create metric stream (建立指標串流)。

Datadog 串流預設值

到 Datadog 的快速合作夥伴設定串流會使用下列預設值：

- 輸出格式: OpenTelemetry 0.7.0
- Firehose 串流內容編碼 GZIP
- Firehose 串流緩衝選項間隔為 60 秒，大小為 4 MB
- Firehose 串流重試選項持續時間 60 秒

若您使用快速合作夥伴設定來建立 Datadog 的指標串流，並串流某些指標，這些指標預設會包括一些額外的統計資料。串流額外統計資料會產生額外費用。如需有關統計資料的詳細資訊，請參閱 可供串流的統計數字。

如果您選擇串流這些指標，下列清單將顯示預設會串流其他統計資料的指標。您可以選擇在開始串流之前，取消選取這些額外的統計資料。

- **AWS/Lambda** 中的 **Duration**：p50、p80、p95、p99、p99.9
- **AWS/Lambda** 中的 **PostRuntimeExtensionDuration**：p50、p99
- **AWS/S3** 中的 **FirstByteLatency** 和 **TotalRequestLatency**：p50、p90、p95、p99、p99.9
- **AWS/Polly** 中的 **ResponseLatency** 和 AWS/ApplicationELB 中的 **TargetResponseTime**：p50、p90、p95、p99
- **AWS/ApiGateway** 中的 **Latency** 和 **IntegrationLatency**：p90、p95、p99
- **AWS/ELB** 中的 **Latency** 和 **TargetResponseTime**：p95、p99
- **AWS/AppRunner** 中的 **RequestLatency**：p50、p95、p99

- **AWS/States** 中的 **ActivityTime**、**ExecutionTime**、**LambdaFunctionRunTime**、**LambdaFunctionScheduleTime**、和 **ActivityScheduleTime**：p95、p99
- **AWS/MediaLive** 中的 **EncoderBitRate**、**ConfiguredBitRate** 和 **ConfiguredBitRateAvailable**：p90
- **AWS/AppSync** 中的 **Latency**：p90

Dynatrace 串流預設值

到 Dynatrace 的快速合作夥伴設定串流會使用下列預設值：

- 輸出格式: OpenTelemetry 0.7.0
- Firehose 串流內容編碼 GZIP
- Firehose 串流緩衝選項間隔為 60 秒，大小為 5 MB
- Firehose 串流重試選項持續時間 600 秒

New Relic 串流預設值

到 New Relic 的快速合作夥伴設定串流會使用下列預設值：

- 輸出格式: OpenTelemetry 0.7.0
- Firehose 串流內容編碼 GZIP
- Firehose 串流緩衝選項間隔 60 秒，大小為 1 MB
- Firehose 串流重試選項持續時間 60 秒

Splunk 可觀測性雲端串流預設值

到 Splunk 可觀測性雲端的快速合作夥伴設定串流會使用下列預設值：

- 輸出格式: OpenTelemetry 0.7.0
- Firehose 串流內容編碼 GZIP
- Firehose 串流緩衝選項間隔 60 秒，大小為 1 MB
- Firehose 串流重試選項持續時間為 300 秒

Sumo Logic 串流預設值

到 Sumo Logic 的快速合作夥伴設定串流會使用下列預設值：

- 輸出格式: OpenTelemetry 0.7.0
- Firehose 串流內容編碼 GZIP
- Firehose 串流緩衝選項間隔 60 秒，大小為 1 MB
- Firehose 串流重試選項持續時間 60 秒

可供串流的統計數字

指標串流始終包含下列統計數字：`Minimum`、`Maximum`、`SampleCount` 以及 `Sum`。您還可以選擇在指標串流中包含下列額外統計數字。此選項是以每個指標為基礎。如需這些統計數字的詳細資訊，請參閱 CloudWatch 統計定義。

- 百分位數值，例如 p95 或 p99 (適用於具有 JSON 或格式的串流) OpenTelemetry
- 裁剪平均值 (僅適用於 JSON 格式的串流)
- 縮尾平均值 (僅適用於 JSON 格式的串流)
- 裁剪計數 (僅適用於 JSON 格式的串流)
- 裁剪總和 (僅適用於 JSON 格式的串流)
- 百分位數排名 (僅適用於 JSON 格式的串流)
- 四分位平均值 (僅適用於 JSON 格式的串流)

串流額外統計數字會產生額外費用。每為特定指標串流 1-5 個額外統計數字，就會算為一次指標更新，並加以計費。此後，每增加一組統計數字 (一組最多包含 5 個統計數字)，都將算為一次指標更新，並加以計費。

例如，假設您為一個指標串流下列 6 個額外統計數字：p95、p99、p99.9、裁剪平均值、縮尾平均值和裁剪總和。則此指標的每次更新都會加以計費，而且計算為 3 次更新：一次是包含預設統計數字的指標更新，一次是包含前 5 個額外統計數字的指標更新，最後一次是包含第 6 個額外統計數字的指標更新。此時如果再加上 4 個額外統計數字 (此時額外統計數字總共為 10 個)，並不會增加計費，但加上第 11 個額外統計數字時便會導致計費增加。

當您指定某指標名稱與命名空間組合來串流額外統計數字時，系統會對該指標名稱與命名空間的所有維度組合進行串流，包括額外統計數字。

CloudWatch 測量結果串流會發佈新的測量結果`TotalMetricUpdate`，反映測量結果更新的基礎數目，以及串流其他統計資料所產生的額外測量結果更新。如需詳細資訊，請參閱 使用指標監控您的 CloudWatch 指標串流。

如需詳細資訊，請參閱 Amazon CloudWatch 定價。

> **Note**
>
> 某些指標不支援百分位數。這些指標的百分位數統計數字會排除在串流外，並且不會產生指標串流費用。您可以在 AWS/ECS 命名空間中的某些指標中發現這些不支援百分位數的統計數字範例。

只有當您設定的額外統計數字與串流篩選條件相符時，該統計數字才會進行串流。例如，如果您建立一個在所包含的篩選條件中只有 EC2 和 RDS 的串流，然後您的統計數字組態列出 EC2 和 Lambda，那麼該串流將包含具有額外統計數字的 EC2 指標和僅具有預設統計數字的 RDS 指標，但完全不包含 Lambda 統計數字。

指標串流操作與維護

指標串流始終處於兩種狀態之一，執行中或已停止。

- 執行中— 指標串流正常運作。由於串流上的篩選條件，可能沒有任何指標資料串流至目的地。
- 已停止— 指標串流已被某人明確停止，但卻不是因為錯誤。停止串流以暫時暫停資料串流而不刪除串流，可能會很有用。

如果您停止並重新啟動測量結果串流，則在測量結果串流停止 CloudWatch時發佈的測量結果資料不會回填至測量結果串流。

如果您變更指標串流的輸出格式，在某些情況下，您可能會看到少了指標資料會以舊格式和新格式寫入目的地。為了避免這種情況，您可以使用與當前設置相同的配置來創建新的 Firehose 交付流，然後更改為新的 Firehose 交付流並同時更改輸出格式。如此一來，具有不同輸出格式的 Kinesis 記錄會以單獨的物件存放在 Simple Storage Service (Amazon S3) 上。稍後，您可以將流量導向回原始 Firehose 交付串流，並刪除第二個傳遞串流。

若要檢視、編輯、停止及啟動指標串流

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/

2. 在導覽窗格中，選擇 Metrics (指標)、Streams (串流)。

   隨即顯示串流清單，並且 Status (狀態) 欄會顯示每個串流是在執行中還是已停止。
3. 若要停止或啟動指標串流，請選取串流並選擇 Stop (停止) 或 Start (啟動)。
4. 若要查看指標串流的詳細資訊，請選取串流並選擇 View details (檢視詳細資訊)。
5. 若要變更串流的輸出格式、篩選器、目標 Firehose 串流或角色，請選擇 [編輯]，然後進行您想要的變更。

   如果您變更篩選條件，轉換期間指標資料可能會存在一些間隙。

使用指標監控您的 CloudWatch 指標串流

測 CloudWatch 量結果串流會發出其狀況和AWS/CloudWatch/MetricStreams命名空間中作業的測量結果。會發出下列指標。會發出具有 MetricStreamName 維度和沒有維度的指標。您可以使用沒有維度的指標，以查看所有指標串流的彙總指標。您可以將指標搭配使用 MetricStreamName 維度，以僅查看僅有關該指標串流的指標。

對於所有這種指標，只會針對處於執行中狀態的指標串流發出值。

指標	描述
MetricUpdate	傳送至指標串流的指標更新數量。如果某段時間內沒有串流任何指標更新，則在該時間段內不會發出此指標。 如果停止指標串流，此指標會停止發出，直到再次啟動指標串流為止。 有效統計數字：Sum 單位：無
TotalMetricUpdate	這是根據正在流式傳輸的其他統計數據計算為 MetricUpdate + 一個數字。 對於每個唯一命名空間與指標名稱組合而言，串流 1-5 個額外統計數字，TotalMetricUpdate 增加 1，串流 6-10 個額外統計數字，TotalMetricUpdate 增加 2，以此類推。 有效統計數字：Sum 單位：無

指標	描述
PublishErrorRate	將資料放入 Firehose 傳送串流時，發生的無法復原錯誤數目。如果某段時間內沒有發生錯誤，則在該時間段內不會發出此指標。 如果停止指標串流，此指標會停止發出，直到再次啟動指標串流為止。 有效統計數字：Average，查看無法寫入的指標更新速率。此值將介於 0.0 和 1.0 之間。 單位：無

CloudWatch 與 Firehose 之間的信任

Firehose 交付串流必須 CloudWatch 透過具有 Firehose 寫入權限的 IAM 角色進行信任。這些權限可以限 CloudWatch 制在指標串流使用的單一 Firehose 傳送串流。IAM 角色必須信任 streams.metrics.cloudwatch.amazonaws.com 服務委託人。

如果您使用 CloudWatch 主控台建立指標串流，您可以 CloudWatch 建立具有正確權限的角色。如果您使用其他方法建立指標串流，或想要建立 IAM 角色本身，則其必須包含下列許可政策和信任政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:firehose:region:account-id:deliverystream/*"
        }
    ]
}
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Principal": {
        "Service": "streams.metrics.cloudwatch.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

指標資料會代表擁有量度串流資源的來源， CloudWatch 將指標資料串流至目的地 Firehose 傳送串流。

指標串流輸出格式

CloudWatch 指標串流中的資料可以是 JSON 格式或 OpenTelemetry 格式。目前，同時支援 OpenTelemetry 1.0.0 和 0.7.0 格式。

內容

- JSON format (JSON 格式)
  - 我應該為 JSON 輸出格式使用哪種 AWS Glue 結構描述？
- OpenTelemetry 格式
  - 使用 OpenTelemetry 1.0.0 格式的翻譯
  - 如何解析 OpenTelemetry 1.0.0 消息
- OpenTelemetry 0.7.0 格式化
  - 使用 OpenTelemetry 0.7.0 格式的翻譯
  - 如何解析 OpenTelemetry 0.7.0 的消息

JSON format (JSON 格式)

在使用 JSON 格式的 CloudWatch 度量串流中，每個 Firehose 記錄都包含多個以換行字元 (\ n) 分隔的 JSON 物件。每個物件都包含單一指標的單一資料點。

使用的 JSON 格式與 AWS Glue Amazon Athena 完全兼容。如果您的 Firehose 交付串流和 AWS Glue 表格格式正確，則在存放到 S3 之前，可以將格式自動轉換為實木複合格式或最佳化資料列單欄 (ORC) 格式。如需有關轉換格式的詳細資訊，請參閱在 Firehose 中轉換輸入記錄格式。若要取得有關的正確格式的更多資訊 AWS Glue，請參閱我應該為 JSON 輸出格式使用哪種 AWS Glue 結構描述？。

在 JSON 格式中，unit 的有效值與 MetricDatum API 結構中的 unit 的值相同。如需詳細資訊，請參閱 MetricDatum。timestamp 欄位的值以 epoch 毫秒為單位，例如 1616004674229。

以下是格式的範例。在此範例中，JSON 已經過格式化以利閱讀；但實際上，整個格式位於單一行上。

```
{
    "metric_stream_name": "MyMetricStream",
    "account_id": "1234567890",
    "region": "us-east-1",
    "namespace": "AWS/EC2",
    "metric_name": "DiskWriteOps",
    "dimensions": {
        "InstanceId": "i-123456789012"
    },
    "timestamp": 1611929698000,
    "value": {
        "count": 3.0,
        "sum": 20.0,
        "max": 18.0,
        "min": 0.0,
        "p99": 17.56,
        "p99.9": 17.8764,
        "TM(25%:75%)": 16.43
    },
    "unit": "Seconds"
}
```

我應該為 JSON 輸出格式使用哪種 AWS Glue 結構描述？

以下是資料表的 JSON AWS Glue 表示法範例，然後 Firehose 會使用該資料表。StorageDescriptor如需有關的更多資訊StorageDescriptor，請參閱 StorageDescriptor。

```
{
  "Columns": [
    {
      "Name": "metric_stream_name",
      "Type": "string"
    },
    {
      "Name": "account_id",
      "Type": "string"
    },
    {
```

```
      "Name": "region",
      "Type": "string"
    },
    {
      "Name": "namespace",
      "Type": "string"
    },
    {
      "Name": "metric_name",
      "Type": "string"
    },
    {
      "Name": "timestamp",
      "Type": "timestamp"
    },
    {
      "Name": "dimensions",
      "Type": "map<string,string>"
    },
    {
      "Name": "value",
      "Type":
 "struct<min:double,max:double,count:double,sum:double,p99:double,p99.9:double>"
    },
    {
      "Name": "unit",
      "Type": "string"
    }
  ],
  "Location": "s3://my-s3-bucket/",
  "InputFormat": "org.apache.hadoop.mapred.TextInputFormat",
  "OutputFormat": "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat",
  "SerdeInfo": {
    "SerializationLibrary": "org.apache.hive.hcatalog.data.JsonSerDe"
  },
  "Parameters": {
    "classification": "json"
  }
}
```

上述範例適用於以 JSON 格式撰寫在 Simple Storage Service (Amazon S3) 上的資料。將下列欄位中的值取代為指出的值，以 Parquet 格式或最佳化列單欄式 (ORC) 格式存放資料。

- Parquet：
  - 輸入格式：或 .apache.hadoop.hive.ql.io. MapredParquetInputFormat
  - 輸出格式:或. 阿帕奇 .hadoop.hive.ql.io. MapredParquetOutputFormat
  - SerDeInfo序列化庫：組織 .apache.hadoop.hive.ql.io.parquet.serde。 ParquetHiveSerDe
  - parameters.classification: parquet
- ORC：
  - 輸入格式：組織 .apache.hadoop.hive.ql.io.orc。 OrcInputFormat
  - 輸出格式：組織 .hadoop.hive.ql.io.orc。 OrcOutputFormat
  - SerDeInfo序列化程式庫：組織 .hadoop.hive.ql.io.orc。 OrcSerde
  - parameters.classification: orc

OpenTelemetry 格式

> **Note**
>
> 使用 OpenTelemetry 1.0.0 格式時，度量屬性會編碼為`KeyValue`物件清單，而不是 0.7.0 格式中使用的`StringKeyValue`類型。作為消費者，這是 0.7.0 和 1.0.0 格式之間唯一的重大變化。從 0.7.0 proto 檔案產生的剖析器不會剖析以 1.0.0 格式編碼的指標屬性。反之亦然，從 1.0.0 proto 檔案產生的剖析器不會剖析以 0.7.0 格式編碼的指標屬性。

OpenTelemetry 是工具、API 和 SDK 的集合。您可以使用它來檢測、產生、收集和匯出遙測資料 (度量、記錄和追蹤) 以進行分析。 OpenTelemetry 是雲端原生運算基金會的一部分。如需詳細資訊，請參閱OpenTelemetry。

如需完整 OpenTelemetry 1.0.0 規格的相關資訊，請參閱版本 1. 0.0。

Kinesis 記錄可以包含一或多個`ExportMetricsServiceRequest` OpenTelemetry 資料結構。每個資料結構都以具有 `UnsignedVarInt32` 的標題為開頭，該標題可指示記錄長度 (以位元組為單位)。每個 `ExportMetricsServiceRequest` 可能一次包含來自多個指標的資料。

以下是`ExportMetricsServiceRequest` OpenTelemetry 數據結構的消息的字符串表示形式。OpenTelemetry 序列化 Google 協議緩衝區二進制協議，這不是人類可讀的。

```
resource_metrics {
  resource {
    attributes {
```

```
      key: "cloud.provider"
      value {
        string_value: "aws"
      }
    }
    attributes {
      key: "cloud.account.id"
      value {
        string_value: "123456789012"
      }
    }
    attributes {
      key: "cloud.region"
      value {
        string_value: "us-east-1"
      }
    }
    attributes {
      key: "aws.exporter.arn"
      value {
        string_value: "arn:aws:cloudwatch:us-east-1:123456789012:metric-stream/
MyMetricStream"
      }
    }
  }
  scope_metrics {
    metrics {
      name: "amazonaws.com/AWS/DynamoDB/ConsumedReadCapacityUnits"
      unit: "NoneTranslated"
      summary {
        data_points {
          start_time_unix_nano: 60000000000
          time_unix_nano: 120000000000
          count: 1
          sum: 1.0
          quantile_values {
            value: 1.0
          }
          quantile_values {
            quantile: 0.95
            value: 1.0
          }
          quantile_values {
            quantile: 0.99
```

```
            value: 1.0
          }
          quantile_values {
            quantile: 1.0
            value: 1.0
          }
          attributes {
            key: "Namespace"
            value {
              string_value: "AWS/DynamoDB"
            }
          }
          attributes {
            key: "MetricName"
            value {
              string_value: "ConsumedReadCapacityUnits"
            }
          }
          attributes {
            key: "Dimensions"
            value {
              kvlist_value {
                values {
                  key: "TableName"
                  value {
                    string_value: "MyTable"
                  }
                }
              }
            }
          }
        }
        data_points {
          start_time_unix_nano: 70000000000
          time_unix_nano: 130000000000
          count: 2
          sum: 5.0
          quantile_values {
            value: 2.0
          }
          quantile_values {
            quantile: 1.0
            value: 3.0
          }
```

```
          attributes {
            key: "Namespace"
            value {
              string_value: "AWS/DynamoDB"
            }
          }
          attributes {
            key: "MetricName"
            value {
              string_value: "ConsumedReadCapacityUnits"
            }
          }
          attributes {
            key: "Dimensions"
            value {
              kvlist_value {
                values {
                  key: "TableName"
                  value {
                    string_value: "MyTable"
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```

序列化 OpenTelemetry 指標資料的頂層物件

ExportMetricsServiceRequest是序列化 OpenTelemetry 匯出器有效載荷的頂層包裝器。它包含一個或多個 ResourceMetrics。

```
message ExportMetricsServiceRequest {
  // An array of ResourceMetrics.
  // For data coming from a single resource this array will typically contain one
  // element. Intermediary nodes (such as OpenTelemetry Collector) that receive
  // data from multiple origins typically batch the data before forwarding further and
  // in that case this array will contain multiple elements.
  repeated opentelemetry.proto.metrics.v1.ResourceMetrics resource_metrics = 1;
```

```
}
```

ResourceMetrics是表示 MetricData 物件的頂層物件。

```
// A collection of ScopeMetrics from a Resource.
message ResourceMetrics {
  reserved 1000;

  // The resource for the metrics in this message.
  // If this field is not set then no resource info is known.
  opentelemetry.proto.resource.v1.Resource resource = 1;

  // A list of metrics that originate from a resource.
  repeated ScopeMetrics scope_metrics = 2;

  // This schema_url applies to the data in the "resource" field. It does not apply
  // to the data in the "scope_metrics" field which have their own schema_url field.
  string schema_url = 3;
}
```

資源物件

Resource 物件是值對物件，其中包含有關產生指標之資源的一些資訊。對於由 AWS建立的指標，資料結構包含與指標相關的資源的 Amazon Resource Name (ARN)，例如 EC2 執行個體或 S3 儲存貯體。

Resource 物件包含稱為 attributes 的屬性，其中會存放鍵/值對的清單。

- cloud.account.id 包含帳戶 ID
- cloud.region 包含區域
- aws.exporter.arn 包含指標串流 ARN
- cloud.provider 始終是 aws。

```
// Resource information.
message Resource {
  // Set of attributes that describe the resource.
  // Attribute keys MUST be unique (it is not allowed to have more than one
  // attribute with the same key).
  repeated opentelemetry.proto.common.v1.KeyValue attributes = 1;
```

```
  // dropped_attributes_count is the number of dropped attributes. If the value is 0,
 then
  // no attributes were dropped.
  uint32 dropped_attributes_count = 2;
}
```

該 ScopeMetrics 對象

將不會填充 scope 欄位。我們只填寫我們正在匯出的指標欄位。

```
// A collection of Metrics produced by an Scope.
message ScopeMetrics {
  // The instrumentation scope information for the metrics in this message.
  // Semantically when InstrumentationScope isn't set, it is equivalent with
  // an empty instrumentation scope name (unknown).
  opentelemetry.proto.common.v1.InstrumentationScope scope = 1;

  // A list of metrics that originate from an instrumentation library.
  repeated Metric metrics = 2;

  // This schema_url applies to all metrics in the "metrics" field.
  string schema_url = 3;
}
```

指標物件

指標物件包含一些中繼資料，以及包含 SummaryDataPoint 清單的 Summary 資料欄位。

對於指標串流，中繼資料如下：

- name 將為 amazonaws.com/*metric_namespace*/*metric_name*
- description 將為空白
- unit 可藉由將指標基準的單位映射至統一計量單位代碼的變體 (區分大小寫) 來進行填充。如需詳細資訊，請參閱 使用 OpenTelemetry 1.0.0 格式的翻譯 和統一計量單位代碼。
- type 將為 SUMMARY

```
message Metric {
  reserved 4, 6, 8;

  // name of the metric, including its DNS name prefix. It must be unique.
  string name = 1;
```

```
  // description of the metric, which can be used in documentation.
  string description = 2;

  // unit in which the metric value is reported. Follows the format
  // described by http://unitsofmeasure.org/ucum.html.
  string unit = 3;

  // Data determines the aggregation type (if any) of the metric, what is the
  // reported value type for the data points, as well as the relatationship to
  // the time interval over which they are reported.
  oneof data {
    Gauge gauge = 5;
    Sum sum = 7;
    Histogram histogram = 9;
    ExponentialHistogram exponential_histogram = 10;
    Summary summary = 11;
  }
}

message Summary {
  repeated SummaryDataPoint data_points = 1;
}
```

該 SummaryDataPoint 對象

SummaryDataPoint 物件包含量度中時間序列中單一資料點的 DoubleSummary 值。

```
// SummaryDataPoint is a single data point in a timeseries that describes the
// time-varying values of a Summary metric.
message SummaryDataPoint {
  reserved 1;

  // The set of key/value pairs that uniquely identify the timeseries from
  // where this point belongs. The list may be empty (may contain 0 elements).
  // Attribute keys MUST be unique (it is not allowed to have more than one
  // attribute with the same key).
  repeated opentelemetry.proto.common.v1.KeyValue attributes = 7;

  // StartTimeUnixNano is optional but strongly encouraged, see the
  // the detailed comments above Metric.
  //
  // Value is UNIX Epoch time in nanoseconds since 00:00:00 UTC on 1 January
  // 1970.
```

```
  fixed64 start_time_unix_nano = 2;

  // TimeUnixNano is required, see the detailed comments above Metric.
  //
  // Value is UNIX Epoch time in nanoseconds since 00:00:00 UTC on 1 January
  // 1970.
  fixed64 time_unix_nano = 3;

  // count is the number of values in the population. Must be non-negative.
  fixed64 count = 4;

  // sum of the values in the population. If count is zero then this field
  // must be zero.
  //
  // Note: Sum should only be filled out when measuring non-negative discrete
  // events, and is assumed to be monotonic over the values of these events.
  // Negative events *can* be recorded, but sum should not be filled out when
  // doing so.  This is specifically to enforce compatibility w/ OpenMetrics,
  // see: https://github.com/OpenObservability/OpenMetrics/blob/main/specification/
OpenMetrics.md#summary
  double sum = 5;

  // Represents the value at a given quantile of a distribution.
  //
  // To record Min and Max values following conventions are used:
  // - The 1.0 quantile is equivalent to the maximum value observed.
  // - The 0.0 quantile is equivalent to the minimum value observed.
  //
  // See the following issue for more context:
  // https://github.com/open-telemetry/opentelemetry-proto/issues/125
  message ValueAtQuantile {
    // The quantile of a distribution. Must be in the interval
    // [0.0, 1.0].
    double quantile = 1;

    // The value at the given quantile of a distribution.
    //
    // Quantile values must NOT be negative.
    double value = 2;
  }

  // (Optional) list of values at different quantiles of the distribution calculated
  // from the current snapshot. The quantiles must be strictly increasing.
  repeated ValueAtQuantile quantile_values = 6;
```

```
  // Flags that apply to this specific data point.  See DataPointFlags
  // for the available flags and their meaning.
  uint32 flags = 8;
}
```

如需詳細資訊，請參閱 使用 OpenTelemetry 1.0.0 格式的翻譯。

使用 OpenTelemetry 1.0.0 格式的翻譯

CloudWatch 執行一些轉換將 CloudWatch 數據轉換為 OpenTelemetry 格式。

轉換命名空間、指標名稱和維度

這些屬性是在映射中編碼的鍵值對。

- 一個屬性，它具有索引鍵 `Namespace`，並且其值是指標的命名空間
- 一個屬性，它具有索引鍵 `MetricName`，並且其值是指標的名稱
- 一對值，它具有索引鍵 `Dimensions`，並且其值是鍵值對的巢狀清單。此清單中的每個配對都會對應至 CloudWatch 量度維度，其中配對的索引鍵是維度的名稱，其值即為維度的值。

轉換平均值、總和 SampleCount、最小值和最大值

摘要資料點可讓您使用一個 CloudWatch 資料點匯出所有這些統計資料。

- `startTimeUnixNano`包含 CloudWatch `startTime`
- `timeUnixNano`包含 CloudWatch `endTime`
- `sum` 包含總和統計資料。
- `count`包含統 SampleCount 計資料。
- `quantile_values` 包含兩個 `valueAtQuantile.value` 物件：
  - `valueAtQuantile.quantile = 0.0` 取代為 `valueAtQuantile.value = Min value`
  - `valueAtQuantile.quantile = 0.99` 取代為 `valueAtQuantile.value = p99 value`
  - `valueAtQuantile.quantile = 0.999` 取代為 `valueAtQuantile.value = p99.9 value`
  - `valueAtQuantile.quantile = 1.0` 取代為 `valueAtQuantile.value = Max value`

使用量度資料流的資源可以將「平均」統計值計算為「總和 SampleCount/」。

轉換單位

CloudWatch 單位會對應至「測量單位」統一程式碼的區分大小寫變體，如下表所示。如需詳細資訊，請參閱統一計量單位代碼。

CloudWatch	OpenTelemetry
秒	s
秒或秒	s
微秒	us
毫秒	ms
位元組	By
KB	kBy
MB	MBy
GB	GBy
TB	TBy
位元	bit
千位元數	kbit
百萬位元數	MBit
十億位元數	GBit
兆位元數	Tbit
百分比	%
計數	{Count}
無	1

透過套用兩個單位的 OpenTelemetry 轉換來對應與斜線組合的單位。例如，位元組/秒映射為 By/s。

如何解析 OpenTelemetry 1.0.0 消息

本節提供的資訊可協助您開始剖析 OpenTelemetry 1.0.0。

首先，您應該獲得特定於語言的綁定，這使您能夠以首選語言解析 OpenTelemetry 1.0.0 消息。

若要取得特定語言的連結

- 步驟取決於您偏好的語言。
  - 要使用 Java，下面的 Maven 依賴項添加到您的 OpenTelemetry Java 項目：
  - 若要使用任何其他語言，請依照下列步驟執行：

    a. 確定已支援您的語言，方法是檢查產生您的類別的清單。

    b. 依照下載協定緩衝區上的步驟安裝 Protobuf 編譯器。

    c. 請在發行版本 1. 0.0 下載 OpenTelemetry 0.7.0 ProtoBuf 定義檔。

    d. 確認您位於已下載 OpenTelemetry 0.7.0 ProtoBuf 定義檔的根資料夾中。隨後依序建立 `src` 資料夾，並執行命令以產生特定語言的綁定。如需詳細資訊，請參閱產生您的類別。

    下列是如何產生 Javascript 連結的範例。

```
protoc --proto_path=./ --js_out=import_style=commonjs,binary:src \
opentelemetry/proto/common/v1/common.proto \
opentelemetry/proto/resource/v1/resource.proto \
opentelemetry/proto/metrics/v1/metrics.proto \
opentelemetry/proto/collector/metrics/v1/metrics_service.proto
```

下列章節包含使用特定語言的連結的範例，您可以依照先前指示建置相關連結。

Java

```
package com.example;

import io.opentelemetry.proto.collector.metrics.v1.ExportMetricsServiceRequest;

import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
```

```
import java.util.List;

public class MyOpenTelemetryParser {

    public List<ExportMetricsServiceRequest> parse(InputStream inputStream) throws
 IOException {
        List<ExportMetricsServiceRequest> result = new ArrayList<>();

        ExportMetricsServiceRequest request;
        /* A Kinesis record can contain multiple `ExportMetricsServiceRequest`
           records, each of them starting with a header with an
           UnsignedVarInt32 indicating the record length in bytes:
            ------ -------------------------- ------ ----------------------
           |UINT32|ExportMetricsServiceRequest|UINT32|ExportMetricsService...
            ------ -------------------------- ------ ----------------------
         */
        while ((request =
 ExportMetricsServiceRequest.parseDelimitedFrom(inputStream)) != null) {
            // Do whatever we want with the parsed message
            result.add(request);
        }

        return result;
    }
}
```

Javascript

此範例假設已產生連結的根資料夾為 `./`

函數的資料引數 `parseRecord` 可為下列其中一種類型：

- `Uint8Array` 這是最佳的
- `Buffer` 在節點下最佳
- `Array.number` 8 位元整數

```
const pb = require('google-protobuf')
const pbMetrics =
    require('./opentelemetry/proto/collector/metrics/v1/metrics_service_pb')

function parseRecord(data) {
    const result = []
```

```
    // Loop until we've read all the data from the buffer
    while (data.length) {
        /* A Kinesis record can contain multiple `ExportMetricsServiceRequest`
           records, each of them starting with a header with an
           UnsignedVarInt32 indicating the record length in bytes:
            ------ -------------------------- ------ ----------------------
           |UINT32|ExportMetricsServiceRequest|UINT32|ExportMetricsService...
            ------ -------------------------- ------ ----------------------
         */
        const reader = new pb.BinaryReader(data)
        const messageLength = reader.decoder_.readUnsignedVarint32()
        const messageFrom = reader.decoder_.cursor_
        const messageTo = messageFrom + messageLength

        // Extract the current `ExportMetricsServiceRequest` message to parse
        const message = data.subarray(messageFrom, messageTo)

        // Parse the current message using the ProtoBuf library
        const parsed =
            pbMetrics.ExportMetricsServiceRequest.deserializeBinary(message)

        // Do whatever we want with the parsed message
        result.push(parsed.toObject())

        // Shrink the remaining buffer, removing the already parsed data
        data = data.subarray(messageTo)
    }

    return result
}
```

Python

您必須自行閱讀 var-int 分隔符或使用內部方法 _VarintBytes(size) 和 _DecodeVarint32(buffer, position)。這些傳回緩衝區中的位置恰好在大小位元組之後。讀取端可構造一個新的緩衝區，且僅限於讀取訊息的位元組。

```
size = my_metric.ByteSize()
f.write(_VarintBytes(size))
f.write(my_metric.SerializeToString())
msg_len, new_pos = _DecodeVarint32(buf, 0)
msg_buf = buf[new_pos:new_pos+msg_len]
```

```
request = metrics_service_pb.ExportMetricsServiceRequest()
request.ParseFromString(msg_buf)
```

Go

請使用 `Buffer.DecodeMessage()`。

C#

請使用 `CodedInputStream`。該列表可以讀取以大小分隔的訊息。

C++

`google/protobuf/util/delimited_message_util.h` 中描述的函數可以讀取以大小分隔的訊息。

其他語言

如需其他語言，請參閱下載協定緩衝區。

實作剖析器時，請考慮 Kinesis 記錄可以包含多個 `ExportMetricsServiceRequest` 協定緩衝區訊息，每個訊息都以具有 `UnsignedVarInt32` 的標題為開頭，而該標題可指示記錄長度 (以位元組為單位)。

OpenTelemetry 0.7.0 格式化

OpenTelemetry 是工具、API 和 SDK 的集合。您可以使用它來檢測、產生、收集和匯出遙測資料 (度量、記錄和追蹤) 以進行分析。 OpenTelemetry 是雲端原生運算基金會的一部分。如需詳細資訊，請參閱OpenTelemetry。

如需完整 OpenTelemetry 0.7.0 規格的相關資訊，請參閱 v0. 7.0 版本。

Kinesis 記錄可以包含一或多個`ExportMetricsServiceRequest` OpenTelemetry 資料結構。每個資料結構都以具有 `UnsignedVarInt32` 的標題為開頭，該標題可指示記錄長度 (以位元組為單位)。每個 `ExportMetricsServiceRequest` 可能一次包含來自多個指標的資料。

以下是`ExportMetricsServiceRequest` OpenTelemetry 數據結構的消息的字符串表示形式。OpenTelemetry 序列化 Google 協議緩衝區二進制協議，這不是人類可讀的。

```
resource_metrics {
  resource {
    attributes {
      key: "cloud.provider"
      value {
```

```
        string_value: "aws"
      }
    }
    attributes {
      key: "cloud.account.id"
      value {
        string_value: "2345678901"
      }
    }
    attributes {
      key: "cloud.region"
      value {
        string_value: "us-east-1"
      }
    }
    attributes {
      key: "aws.exporter.arn"
      value {
        string_value: "arn:aws:cloudwatch:us-east-1:123456789012:metric-stream/
MyMetricStream"
      }
    }
  }
  instrumentation_library_metrics {
    metrics {
      name: "amazonaws.com/AWS/DynamoDB/ConsumedReadCapacityUnits"
      unit: "1"
      double_summary {
        data_points {
          labels {
            key: "Namespace"
            value: "AWS/DynamoDB"
          }
          labels {
            key: "MetricName"
            value: "ConsumedReadCapacityUnits"
          }
          labels {
            key: "TableName"
            value: "MyTable"
          }
          start_time_unix_nano: 1604948400000000000
          time_unix_nano: 1604948460000000000
          count: 1
```

```
          sum: 1.0
          quantile_values {
            quantile: 0.0
            value: 1.0
          }
          quantile_values {
            quantile: 0.95
            value: 1.0
          }
          quantile_values {
            quantile: 0.99
            value: 1.0
          }
          quantile_values {
            quantile: 1.0
            value: 1.0
          }
        }
        data_points {
          labels {
            key: "Namespace"
            value: "AWS/DynamoDB"
          }
          labels {
            key: "MetricName"
            value: "ConsumedReadCapacityUnits"
          }
          labels {
            key: "TableName"
            value: "MyTable"
          }
          start_time_unix_nano: 1604948460000000000
          time_unix_nano: 1604948520000000000
          count: 2
          sum: 5.0
          quantile_values {
            quantile: 0.0
            value: 2.0
          }
          quantile_values {
            quantile: 1.0
            value: 3.0
          }
        }
```

```
      }
    }
  }
}
```

序列化 OpenTelemetry 指標資料的頂層物件

`ExportMetricsServiceRequest`是序列化 OpenTelemetry 匯出器有效載荷的頂層包裝器。它包含一個或多個 `ResourceMetrics`。

```
message ExportMetricsServiceRequest {
  // An array of ResourceMetrics.
  // For data coming from a single resource this array will typically contain one
  // element. Intermediary nodes (such as OpenTelemetry Collector) that receive
  // data from multiple origins typically batch the data before forwarding further and
  // in that case this array will contain multiple elements.
  repeated opentelemetry.proto.metrics.v1.ResourceMetrics resource_metrics = 1;
}
```

`ResourceMetrics`是表示 MetricData 物件的頂層物件。

```
// A collection of InstrumentationLibraryMetrics from a Resource.
message ResourceMetrics {
  // The resource for the metrics in this message.
  // If this field is not set then no resource info is known.
  opentelemetry.proto.resource.v1.Resource resource = 1;

  // A list of metrics that originate from a resource.
  repeated InstrumentationLibraryMetrics instrumentation_library_metrics = 2;
}
```

資源物件

`Resource` 物件是值對物件，其中包含有關產生指標之資源的一些資訊。對於由 AWS建立的指標，資料結構包含與指標相關的資源的 Amazon Resource Name (ARN)，例如 EC2 執行個體或 S3 儲存貯體。

`Resource` 物件包含稱為 `attributes` 的屬性，其中會存放鍵/值對的清單。

- `cloud.account.id` 包含帳戶 ID
- `cloud.region` 包含區域

- aws.exporter.arn 包含指標串流 ARN
- cloud.provider 始終是 aws。

```
// Resource information.
message Resource {
  // Set of labels that describe the resource.
  repeated opentelemetry.proto.common.v1.KeyValue attributes = 1;

  // dropped_attributes_count is the number of dropped attributes. If the value is 0,
  // no attributes were dropped.
  uint32 dropped_attributes_count = 2;
}
```

該 InstrumentationLibraryMetrics 對象

instrumentation_library 欄位將不會被填充。我們將只填寫我們正在匯出的指標欄位。

```
// A collection of Metrics produced by an InstrumentationLibrary.
message InstrumentationLibraryMetrics {
  // The instrumentation library information for the metrics in this message.
  // If this field is not set then no library info is known.
  opentelemetry.proto.common.v1.InstrumentationLibrary instrumentation_library = 1;
  // A list of metrics that originate from an instrumentation library.
  repeated Metric metrics = 2;
}
```

指標物件

指標物件包含 DoubleSummary 資料欄位，其中包含 DoubleSummaryDataPoint 的清單。

```
message Metric {
  // name of the metric, including its DNS name prefix. It must be unique.
  string name = 1;

  // description of the metric, which can be used in documentation.
  string description = 2;

  // unit in which the metric value is reported. Follows the format
  // described by http://unitsofmeasure.org/ucum.html.
  string unit = 3;
```

```
  oneof data {
    IntGauge int_gauge = 4;
    DoubleGauge double_gauge = 5;
    IntSum int_sum = 6;
    DoubleSum double_sum = 7;
    IntHistogram int_histogram = 8;
    DoubleHistogram double_histogram = 9;
    DoubleSummary double_summary = 11;
  }
}

message DoubleSummary {
  repeated DoubleSummaryDataPoint data_points = 1;
}
```

該 MetricDescriptor 對象

MetricDescriptor 物件包含中繼資料。如需詳細資訊，請參閱上的量度。 GitHub

對於測量結果串流， MetricDescriptor 具有下列內容：

- `name` 將為 `amazonaws.com/`*`metric_namespace`*`/`*`metric_name`*
- `description` 將為空白。
- `unit` 可藉由將指標基準的單位映射至統一計量單位代碼的變體 (區分大小寫) 來進行填充。如需詳細資訊，請參閱 使用 OpenTelemetry 0.7.0 格式的翻譯 和統一計量單位代碼。
- `type` 將為 `SUMMARY`。

該 DoubleSummaryDataPoint 對象

DoubleSummaryDataPoint 物件包含量度中時間序列中單一資料點的 DoubleSummary 值。

```
// DoubleSummaryDataPoint is a single data point in a timeseries that describes the
// time-varying values of a Summary metric.
message DoubleSummaryDataPoint {
  // The set of labels that uniquely identify this timeseries.
  repeated opentelemetry.proto.common.v1.StringKeyValue labels = 1;

  // start_time_unix_nano is the last time when the aggregation value was reset
  // to "zero". For some metric types this is ignored, see data types for more
  // details.
  //
```

```
  // The aggregation value is over the time interval (start_time_unix_nano,
  // time_unix_nano].
  //
  // Value is UNIX Epoch time in nanoseconds since 00:00:00 UTC on 1 January
  // 1970.
  //
  // Value of 0 indicates that the timestamp is unspecified. In that case the
  // timestamp may be decided by the backend.
  fixed64 start_time_unix_nano = 2;

  // time_unix_nano is the moment when this aggregation value was reported.
  //
  // Value is UNIX Epoch time in nanoseconds since 00:00:00 UTC on 1 January
  // 1970.
  fixed64 time_unix_nano = 3;

  // count is the number of values in the population. Must be non-negative.
  fixed64 count = 4;

  // sum of the values in the population. If count is zero then this field
  // must be zero.
  double sum = 5;

  // Represents the value at a given quantile of a distribution.
  //
  // To record Min and Max values following conventions are used:
  // - The 1.0 quantile is equivalent to the maximum value observed.
  // - The 0.0 quantile is equivalent to the minimum value observed.
  message ValueAtQuantile {
    // The quantile of a distribution. Must be in the interval
    // [0.0, 1.0].
    double quantile = 1;

    // The value at the given quantile of a distribution.
    double value = 2;
  }

  // (Optional) list of values at different quantiles of the distribution calculated
  // from the current snapshot. The quantiles must be strictly increasing.
  repeated ValueAtQuantile quantile_values = 6;
}
```

如需詳細資訊，請參閱 使用 OpenTelemetry 0.7.0 格式的翻譯。

使用 OpenTelemetry 0.7.0 格式的翻譯

CloudWatch 執行一些轉換將 CloudWatch 數據轉換為 OpenTelemetry 格式。

轉換命名空間、指標名稱和維度

這些屬性是在映射中編碼的鍵值對。

- 一對包含指標的命名空間
- 一對包含指標的名稱
- 針對每個維度，CloudWatch 儲存下列配對：`metricDatum.Dimensions[i].`*`Name`*`, metricDatum.Dimensions[i].`*`Value`*

轉換平均值、總和 SampleCount、最小值和最大值

摘要資料點可讓您使用一個 CloudWatch 資料點匯出所有這些統計資料。

- `startTimeUnixNano`包含 CloudWatch `startTime`
- `timeUnixNano`包含 CloudWatch `endTime`
- `sum` 包含總和統計資料。
- `count`包含統 SampleCount 計資料。
- `quantile_values` 包含兩個 `valueAtQuantile.value` 物件：
  - `valueAtQuantile.quantile = 0.0` 取代為 `valueAtQuantile.value =` *`Min value`*
  - `valueAtQuantile.quantile = 0.99` 取代為 `valueAtQuantile.value =` *`p99 value`*
  - `valueAtQuantile.quantile = 0.999` 取代為 `valueAtQuantile.value =` *`p99.9 value`*
  - `valueAtQuantile.quantile = 1.0` 取代為 `valueAtQuantile.value =` *`Max value`*

使用量度資料流的資源可以將「平均」統計值計算為「總和 SampleCount/」。

轉換單位

CloudWatch 單位會對應至「測量單位」統一程式碼的區分大小寫變體，如下表所示。如需詳細資訊，請參閱統一計量單位代碼。

CloudWatch	OpenTelemetry
秒	s
秒或秒	s
微秒	us
毫秒	ms
位元組	By
KB	kBy
MB	MBy
GB	GBy
TB	TBy
位元	bit
千位元數	kbit
百萬位元數	MBit
十億位元數	GBit
兆位元數	Tbit
百分比	%
計數	{Count}
無	1

透過套用兩個單位的 OpenTelemetry 轉換來對應與斜線組合的單位。例如，位元組/秒映射為 By/s。

如何解析 OpenTelemetry 0.7.0 的消息

本節提供的資訊可協助您開始剖析 OpenTelemetry 0.7.0。

首先，您應該獲得特定於語言的綁定，這使您能夠以首選語言解析 OpenTelemetry 0.7.0 消息。

若要取得特定語言的連結

- 步驟取決於您偏好的語言。
  - 要使用 Java，下面的 Maven 依賴項添加到您的 OpenTelemetry Java 項目：
  - 若要使用任何其他語言，請依照下列步驟執行：

    a. 確定已支援您的語言，方法是檢查產生您的類別的清單。

    b. 依照下載協定緩衝區上的步驟安裝 Protobuf 編譯器。

    c. 請在 v OpenTelemetry 0.7.0 版本中下載 0.7.0 ProtoBuf 定義檔。

    d. 確認您位於已下載 OpenTelemetry 0.7.0 ProtoBuf 定義檔的根資料夾中。隨後依序建立 `src` 資料夾，並執行命令以產生特定語言的綁定。如需詳細資訊，請參閱產生您的類別。

    下列是如何產生 Javascript 連結的範例。

```
protoc --proto_path=./ --js_out=import_style=commonjs,binary:src \
opentelemetry/proto/common/v1/common.proto \
opentelemetry/proto/resource/v1/resource.proto \
opentelemetry/proto/metrics/v1/metrics.proto \
opentelemetry/proto/collector/metrics/v1/metrics_service.proto
```

下列章節包含使用特定語言的連結的範例，您可以依照先前指示建置相關連結。

Java

```
package com.example;

import io.opentelemetry.proto.collector.metrics.v1.ExportMetricsServiceRequest;

import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

public class MyOpenTelemetryParser {

    public List<ExportMetricsServiceRequest> parse(InputStream inputStream) throws
 IOException {
```

```
        List<ExportMetricsServiceRequest> result = new ArrayList<>();

        ExportMetricsServiceRequest request;
        /* A Kinesis record can contain multiple `ExportMetricsServiceRequest`
           records, each of them starting with a header with an
           UnsignedVarInt32 indicating the record length in bytes:
            ------ -------------------------- ------ ----------------------
           |UINT32|ExportMetricsServiceRequest|UINT32|ExportMetricsService...
            ------ -------------------------- ------ ----------------------
         */
        while ((request =
 ExportMetricsServiceRequest.parseDelimitedFrom(inputStream)) != null) {
            // Do whatever we want with the parsed message
            result.add(request);
        }

        return result;
    }
}
```

Javascript

此範例假設已產生連結的根資料夾為 `./`

函數的資料引數 `parseRecord` 可為下列其中一種類型：

- `Uint8Array` 這是最佳的
- `Buffer` 在節點下最佳
- `Array.number` 8 位元整數

```
const pb = require('google-protobuf')
const pbMetrics =
    require('./opentelemetry/proto/collector/metrics/v1/metrics_service_pb')

function parseRecord(data) {
    const result = []

    // Loop until we've read all the data from the buffer
    while (data.length) {
        /* A Kinesis record can contain multiple `ExportMetricsServiceRequest`
           records, each of them starting with a header with an
           UnsignedVarInt32 indicating the record length in bytes:
```

```
            ------ -------------------------- ------ ----------------------
           |UINT32|ExportMetricsServiceRequest|UINT32|ExportMetricsService...
            ------ -------------------------- ------ ----------------------
         */
        const reader = new pb.BinaryReader(data)
        const messageLength = reader.decoder_.readUnsignedVarint32()
        const messageFrom = reader.decoder_.cursor_
        const messageTo = messageFrom + messageLength

        // Extract the current `ExportMetricsServiceRequest` message to parse
        const message = data.subarray(messageFrom, messageTo)

        // Parse the current message using the ProtoBuf library
        const parsed =
            pbMetrics.ExportMetricsServiceRequest.deserializeBinary(message)

        // Do whatever we want with the parsed message
        result.push(parsed.toObject())

        // Shrink the remaining buffer, removing the already parsed data
        data = data.subarray(messageTo)
    }

    return result
}
```

Python

您必須自行閱讀 var-int 分隔符或使用內部方法 _VarintBytes(size) 和 _DecodeVarint32(buffer, position)。這些傳回緩衝區中的位置恰好在大小位元組之後。讀取端可構造一個新的緩衝區，且僅限於讀取訊息的位元組。

```
size = my_metric.ByteSize()
f.write(_VarintBytes(size))
f.write(my_metric.SerializeToString())
msg_len, new_pos = _DecodeVarint32(buf, 0)
msg_buf = buf[new_pos:new_pos+msg_len]
request = metrics_service_pb.ExportMetricsServiceRequest()
request.ParseFromString(msg_buf)
```

Go

請使用 Buffer.DecodeMessage()。

C#

請使用 `CodedInputStream`。該列表可以讀取以大小分隔的訊息。

C++

`google/protobuf/util/delimited_message_util.h` 中描述的函數可以讀取以大小分隔的訊息。

其他語言

如需其他語言，請參閱下載協定緩衝區。

實作剖析器時，請考慮 Kinesis 記錄可以包含多個 `ExportMetricsServiceRequest` 協定緩衝區訊息，每個訊息都以具有 `UnsignedVarInt32` 的標題為開頭，而該標題可指示記錄長度 (以位元組為單位)。

故障診斷

如果您在最終目的地沒有看到指標資料，請檢查下列項目：

- 檢查指標串流處於執行中狀態。有關如何使用 CloudWatch 控制台執行此操作的步驟，請參閱指標串流操作與維護。
- 過去兩天以上發佈的量度不會串流。若要判斷是否要串流特定測量結果，請在 CloudWatch 主控台中繪製度量圖形，並檢查最後一個可見資料點的年齡。如果過去超過兩天，則指標流將不會拾取它。
- 檢查指標串流發出的指標。在 CloudWatch 主控台的「度量」下，查看、和度PublishErrorRate量的AWSCloudWatch//MetricStreams命名空間。MetricUpdateTotalMetricUpdate
- 如果指PublishErrorRate標很高，請確認 Firehose 交付串流所使用的目的地存在，且指標串流組態中指定的 IAM 角色授予寫入`CloudWatch`服務主體權限。如需詳細資訊，請參閱 CloudWatch 與 Firehose 之間的信任。
- 檢查 Firehose 交付串流是否有寫入最終目的地的權限。
- 在 Firehose 主控台中，檢視用於指標串流的 Firehose 傳送串流，並查看「監視」索引標籤，查看 Firehose 傳送串流是否正在接收資料。
- 確認您已使用正確的詳細資料設定 Firehose 傳送串流。
- 檢查 Firehose 傳送串流寫入的最終目的地的任何可用記錄或指標。
- 若要取得更多詳細資訊，請在 Firehose 傳送串流上啟用記 CloudWatch 錄錯誤記錄。如需詳細資訊，請參閱使用 CloudWatch 記錄監控 Amazon 資料 Firehose。

檢視可用的指標

指標會先依據命名空間進行分組，再依據各命名空間內不同的維度組合進行分組。例如，您可以檢視所有 EC2 指標，以執行個體分組的 EC2 指標，或以 Auto Scaling 群組分組的 EC2 指標。

只有您正在使用的 AWS 服務將指標發送到 Amazon CloudWatch。

如需將量度傳送至的 AWS 服務清單 CloudWatch，請參閱AWS 發佈指 CloudWatch標的服務。您也可以在此頁面查看每個服務所發布的指標和維度。

> **Note**
>
> 過去兩週內沒有任何新資料點的指標不會顯示在主控台中。當您在主控台的 All metrics (所有指標) 索引標籤的搜尋方塊中輸入公制名稱或維度名稱時，它們也不會顯示，而且在 list-metrics 命令的結果中不會傳回這些名稱。擷取這些度量的最佳方式是使用中的get-metric-data或get-metric-statistics命令 AWS CLI。
>
> 如果您要檢視的舊指標具有類似維度的目前指標，您可以檢視那個類似的目前指標，然後選擇 Source (來源) 索引標籤，並將公制名稱和維度欄位變更為您想要的，也可以將時間範圍變更為在報告指標時的時間。

下列步驟可協助您瀏覽指標命名空間，以尋找和檢視指標。您也可以使用目標搜尋詞彙來搜尋指標。如需詳細資訊，請參閱 搜尋可用的指標。

如果您在設定為 CloudWatch 跨帳戶觀察性監控帳戶的帳戶中瀏覽，則可以從連結至此監控帳戶的來源帳戶中檢視指標。在顯示來源帳戶的指標時，也會顯示來源帳戶的 ID 或標籤。如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。

使用主控台依據命名空間和維度查看可用的指標

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Metrics (指標)、All metrics (所有指標)。
3. 選取指標命名空間 (例如 EC2 或 Lambda)。
4. 選取指標維度 (例如 Per-Instance Metrics (每個執行個體指標) 或 By Function Name (依函數名稱))。
5. Browse (瀏覽) 索引標籤會顯示命名空間中該維度的所有指標。每個指標名稱都是一個資訊按鈕，您可選擇查看指標定義的快顯視窗。

如果這是 CloudWatch 跨帳戶可觀察性的監控帳戶，您還可以看到鏈接到此監視帳戶的來源帳戶中的指標。表格中的 Account label (帳戶標籤) 和 Account id (帳戶 ID) 資料欄會顯示每個指標來自哪個帳戶。

您可以執行下列作業：

a. 若要將資料表排序，請使用直欄標題。

b. 若要將指標圖形化，請勾選指標旁的核取方塊。若要選擇所有指標，請勾選表格標題列中的核取方塊。

c. 若要依帳戶篩選，請選擇帳戶標籤或帳戶 ID，然後選擇 Add to search (新增至搜尋)。

d. 若要依資源篩選，請選擇資源 ID，然後選擇 Add to search (新增至搜尋)。

e. 若要依指標篩選，請選擇指標名稱，然後選擇 Add to search (新增至搜尋)。

6. (選擇性) 若要將此圖形新增至 CloudWatch 儀表板，請選擇動作 > 新增至儀表板。

若要使用帳戶命名空間、維度或量度檢視可用量度 AWS CLI

使用列表公制指令列出 CloudWatch 量度。如需發佈指標之所有服務的命名空間、指標和維度清單，請參閱 AWS 發佈指 CloudWatch標的服務

下列範例命令會列出 Amazon EC2 的所有指標。

```
aws cloudwatch list-metrics --namespace AWS/EC2
```

下列為範例輸出。

```
{
  "Metrics" : [
    ...
    {
        "Namespace": "AWS/EC2",
        "Dimensions": [
            {
                "Name": "InstanceId",
                "Value": "i-1234567890abcdef0"
            }
        ],
        "MetricName": "NetworkOut"
    },
```

```
    {
        "Namespace": "AWS/EC2",
        "Dimensions": [
            {
                "Name": "InstanceId",
                "Value": "i-1234567890abcdef0"
            }
        ],
        "MetricName": "CPUUtilization"
    },
    {
        "Namespace": "AWS/EC2",
        "Dimensions": [
            {
                "Name": "InstanceId",
                "Value": "i-1234567890abcdef0"
            }
        ],
        "MetricName": "NetworkIn"
    },
    ...
  ]
}
```

列出指定資源的所有可用指標

以下範例指定 AWS/EC2 命名空間和 InstanceId 維度，以僅檢視指定執行個體的結果。

```
aws cloudwatch list-metrics --namespace AWS/EC2 --dimensions
 Name=InstanceId,Value=i-1234567890abcdef0
```

列出所有資源的指標

以下範例指定 AWS/EC2 命名空間和指標名稱，以僅檢視指定指標的結果。

```
aws cloudwatch list-metrics --namespace AWS/EC2 --metric-name CPUUtilization
```

從 CloudWatch 跨帳戶可觀察性的連結來源帳戶擷取指標

下列範例會在監控帳戶中執行，以從監控帳戶和所有已連結的來源帳戶擷取指標。若您未新增 --include-linked-accounts，命令僅會傳回監控帳戶的指標。

```
aws cloudwatch list-metrics --include-linked-accounts
```

從 CloudWatch 跨帳戶可觀察性的來源帳戶擷取指標

下列範例會在監控帳戶中執行，以使用 ID 111122223333 從來源帳戶擷取指標。

```
aws cloudwatch list-metrics --include-linked-accounts --owning-account "111122223333"
```

搜尋可用的指標

您可以使用目標搜尋詞彙，在您帳戶的所有指標中搜尋。將會傳回在其命名空間、指標名稱或維度中具有相符結果的指標。

如果這是 CloudWatch 跨帳戶觀察性的監控帳戶，您也可以從連結至此監視帳戶的來源帳戶中搜尋指標。

> **Note**
>
> 過去兩週內沒有任何新資料點的指標不會顯示在主控台中。當您在主控台的 All metrics (所有指標) 索引標籤的搜尋方塊中輸入公制名稱或維度名稱時，它們也不會顯示，而且在 list-metrics 命令的結果中不會傳回這些名稱。擷取這些度量的最佳方式是使用中的get-metric-data或get-metric-statistics命令 AWS CLI。

若要搜尋可用的測量結果 CloudWatch

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 指標。
3. 在 All metrics (所有指標) 索引標籤的搜尋欄位中，輸入搜尋字詞，例如指標名稱、命名空間、帳戶 ID、帳戶標籤、維度名稱或值，或是資源名稱。這會顯示有此搜尋詞彙之指標的所有命名空間。

   例如，如果您搜尋 **`volume`**，將會顯示內含具有此詞彙之指標名稱的命名空間。

   如需搜尋的詳細資訊，請參閱在圖形中使用搜尋運算式。
4. 若要建立所有搜尋結果的圖形，請選擇 Graph search (圖形搜尋)

   或

選取命名空間，檢視該命名空間中的指標。然後，您可以執行下列作業：

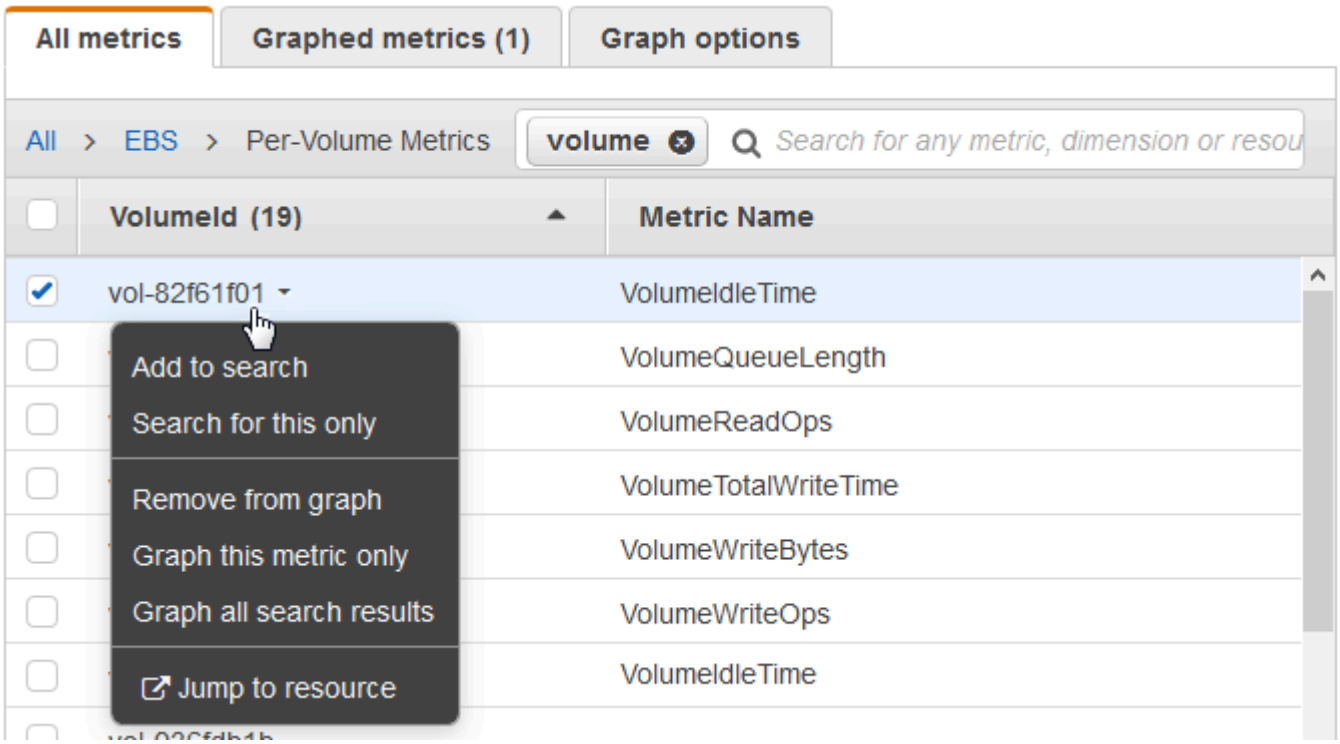
a. 若要將一或多個指標圖形化，請選取各個指標旁的核取方塊。若要選擇所有指標，請勾選表格標題列中的核取方塊。

b. 若要強化搜尋，請將滑鼠游標移到指標名稱上，然後選擇 Add to search (新增至搜尋) 或 Search for this only (僅搜尋此項)。

c. 若要在其主控台中檢視資源之一，請選擇資源 ID，然後選擇 Jump to resource (跳至資源)。

d. 若要查看指標的說明，請選取指標名稱並選擇 What is this? (這是什麼？)。

圖形中隨即顯示選取的指標。



5. (選用) 在搜尋列中選取其中一個按鈕，編輯搜尋詞彙的一部分。

建立指標圖形

使用主 CloudWatch 控台來繪製其他 AWS 服務產生的度量資料圖形。如此可讓您更有效率地查看服務上的指標活動。下列程序說明如何在中繪製度量圖形 CloudWatch。

目錄

- 將指標圖形化

- 將兩個圖形合併為一個
- 使用動態標籤
- 修改圖形的時間範圍或時區格式
- 放大折線圖或堆疊區域圖
- 修改圖形的 Y 軸
- 從圖形的指標建立警示

將指標圖形化

您可以使用 CloudWatch主控台選取指標並建立指標資料的圖形。

CloudWatch 支援下列測量結果的統計資料：`Average`、`Minimum`、`Maximum`、`Sum`、和`SampleCount`。如需詳細資訊，請參閱 統計資料。

您可以用不同的詳細資料層級檢視您的資料。例如，您可以選擇一分鐘檢視，這在疑難排解時非常有用。或者，選擇較不精細的一小時檢視。當檢視更廣泛的時間範圍 (例如 3 天) 時，您可以看到隨時間變化的趨勢。如需詳細資訊，請參閱 期間。

如果您使用的帳戶設定為 CloudWatch 跨帳戶可觀察性的監視帳戶，則可以從連結至此監視帳戶的來源帳戶繪製指標圖形。如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。

建立圖形

繪製指標圖形

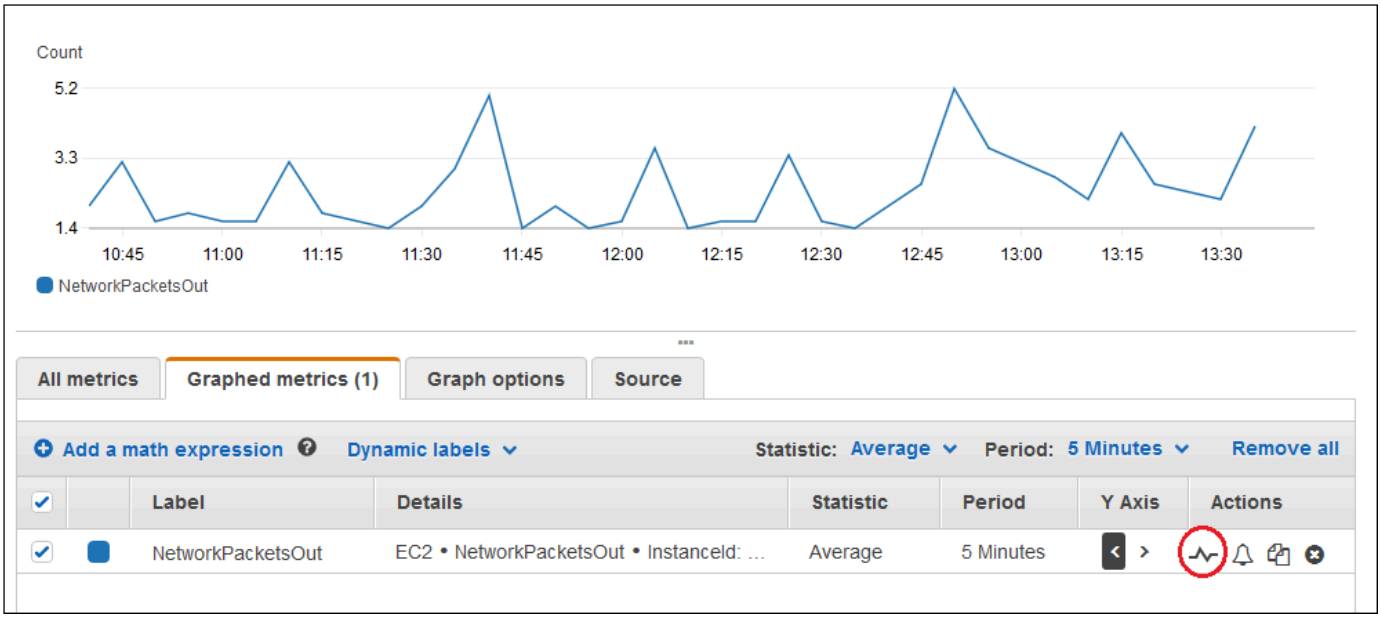
1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Metrics (指標)、All metrics (所有指標)。
3. 在瀏覽索引標籤的搜尋欄位中，輸入搜尋字詞，例如指標名稱、帳戶 ID 或資源名稱。

   例如，如果您搜尋 `CPUUtilization` 指標，您會看到具有此指標的命名空間和維度。
4. 選取您的搜尋結果之一，以檢視該指標。
5. 若要將一或多個指標圖形化，請選取各個指標旁的核取方塊。若要選擇所有指標，請勾選表格標題列中的核取方塊。
6. (選用) 若要變更圖形類型，請選擇選項索引標籤。然後，您可以在折線圖、堆疊區域圖、數字顯示、儀表圖、長條圖或圓餅圖之間進行選擇。
7. 選擇 Graphed metrics (圖表化指標) 標籤。

8. (選用) 若要變更圖形中使用的統計資料，請在指標名稱旁的 Statistic (資料) 欄位中選擇新的統計資料。

   如需 CloudWatch 統計資料的詳細資訊，請參閱CloudWatch 統計定義。如需有關 pxx 百分位數統計數字的詳細資訊，請參閱 百分位數。

9. (選用) 如果要新增顯示指標預期值的異常偵測範圍，請選擇指標旁 Actions (動作) 下的異常偵測圖示。



   CloudWatch 使用最多兩週的量度最近歷史資料來計算預期值的模型。然後，它會將此範圍的預期值顯示為圖表上的區帶。 CloudWatch 在量度下新增一列，以顯示標示為「異常 _ 偵測 _BAND」的異常偵測帶數學運算式。如果有最近歷史資料存在，您可以立即看到預覽異常偵測範圍，這是模型產生的異常偵測範圍的大約值。最多需要 15 分鐘，實際異常偵測範圍才會出現。

   根據預設， CloudWatch 會建立預期值區間的上限和下限，區帶臨界值的預設值為 2。若要變更此數字，請變更範圍 Details (詳細資訊) 下公式結尾的值。

   - (選用) 選擇 Edit model (編輯模型) 來變更異常偵測模型計算的方式。您可在訓練計算模型時，排除使用過去和未來的時間區間。請務必從訓練資料中排除不尋常的事件，例如系統中斷、部署和假日。您也可以指定用於日光節約時間變更模型的時區。

   如需詳細資訊，請參閱 使用 CloudWatch 異常偵測。

若要從圖形中隱藏模型，請從具有 ANOMALY_DETECTION_BAND 函數的行中移除核取記號，或選擇 X 圖示。若要完整刪除模型，請選擇 Edit model (編輯模型)，然後選擇 Delete model (刪除模型)。

10. (選用) 若您選擇指標圖形，請指定動態標籤，在每個指標的圖形圖例上顯示。動態標籤會顯示指標的統計資料，並在儀表板或圖形重新整理時自動更新。若要新增動態標籤，請選擇圖表化指標，然後選擇新增動態標籤。

    根據預設，您新增到標籤的動態值會出現在標籤的開頭。您可以接著選擇指標的 Label (標籤) 值來編輯標籤。如需詳細資訊，請參閱 使用動態標籤。

11. 若要檢視正在繪製圖形的指標詳細資訊，請將滑鼠游標移至圖例上。

12. 水平註釋可協助圖形使用者更有效率地查看指標何時的峰值達到特定水準，或是指標是否在預先定義的範圍內。若要新增水平註釋，請選擇選項索引標簽，然後選擇新增水平註釋：

    a. 針對 Label (標籤)，輸入註釋的標籤。

    b. 針對 Value (值)，輸入水平註釋顯示的指標值。

    c. 在 Fill (填充) 中指定此註釋是否要填充陰影。例如，選擇 Above 或 Below 作為要填入的相關區域。如果您指定 Between，將會顯示另一個 Value 欄位，這兩個值之間的圖形區域將被填充。

    d. 針對 Axis (軸)，指定 Value 中的數字是否要參照與左側 Y 軸或右側 Y 軸關聯的指標 (如果此圖形包含多個指標)。

       您可以透過選擇註釋左側欄中的顏色方塊，來變更註釋的填充顏色。

    重複這些步驟，將多個水平註釋新增到同一個圖形。

    若要隱藏註釋，請清除該註釋左側欄中的核取方塊。

    若要刪除註釋，請選擇 Actions (動作) 欄中的 x。

13. 若要取得您的圖形的 URL，請選擇 Actions (動作)、Share (分享)。複製 URL 以儲存或共用。

14. 若要將您的圖形新增至儀表板，請選擇 Actions (動作)、Add to dashboard (新增至儀表板)。

從另一個資料來源建立指標圖表

您可以建立顯示資料來源以外的資源的圖形 CloudWatch。如需有關建立其他資料來源連線的詳細資訊，請參閱 從其他資料來源中查詢指標。

從另一個資料來源繪製指標圖形

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Metrics (指標)、All metrics (所有指標)。
3. 選擇多來源查詢索引標籤。
4. 對於資料來源，選取您想要使用的資料來源。

   如果尚未建立所需資料來源的連線，請選取建立和管理資料來源，然後選擇建立和管理資料來源。如需有關此資料來源建立程序其餘部分的資訊，請參閱 使用精靈連線至預先建立的資料來源。
5. 精靈或查詢編輯器會提示您輸入查詢所需的資訊。每個資料來源的工作流程都不同，並針對資料來源量身打造。例如，對於 Amazon Managed Service for Prometheus 和 Prometheus 資料來源，則會出現一個包含查詢協助程式的 PromQL 查詢編輯器方塊。
6. 完成查詢的建構後，請選擇圖形查詢。

   圖形會填入查詢中的指標。
7. (選用) 水平註釋可協助圖形使用者更有效率地查看指標何時的峰值達到特定水準，或是指標是否在預先定義的範圍內。若要新增水平註釋，請選擇選項索引標籤，然後選擇新增水平註釋：

   a. 針對 Label (標籤)，輸入註釋的標籤。

   b. 針對 Value (值)，輸入水平註釋顯示的指標值。

   c. 在 Fill (填充) 中指定此註釋是否要填充陰影。例如，選擇 `Above` 或 `Below` 作為要填入的相關區域。如果您指定 `Between`，將會顯示另一個 `Value` 欄位，這兩個值之間的圖形區域將被填充。

   d. 針對 Axis (軸)，指定 `Value` 中的數字是否要參照與左側 Y 軸或右側 Y 軸關聯的指標 (如果此圖形包含多個指標)。

      您可以透過選擇註釋左側欄中的顏色方塊，來變更註釋的填充顏色。

   重複這些步驟，將多個水平註釋新增到同一個圖形。

   若要隱藏註釋，請清除該註釋左側欄中的核取方塊。

   若要刪除註釋，請選擇 Actions (動作) 欄中的 x。
8. (選用) 若要將圖表新增至儀表板，請依次選擇動作、新增至儀表板。

更新圖形

更新您的圖形

1. 若要變更圖形的名稱，請選擇鉛筆圖示。
2. 若要變更時間範圍，請選取一個預先定義的值，或選擇 custom (自訂)。如需詳細資訊，請參閱 修改圖形的時間範圍或時區格式。
3. 若要變更統計數字，請選擇 Graphed metrics (圖形化指標) 索引標籤。選擇欄位標題或個別的值，然後選擇統計資料之一或預先定義的百分位，或指定自訂的百分比 (例如，**p95.45**)。
4. 若要變更期間，請選擇 Graphed metrics (圖表化指標) 標籤。選擇欄位標題或個別的值，然後選擇不同的值。
5. 若要新增水平註釋，請選擇 Graph options (圖形選項)，和 Add horizontal annotation (新增水平註釋)：

   a. 針對 Label (標籤)，輸入註釋的標籤。

   b. 針對 Value (值)，輸入水平註釋顯示的指標值。

   c. 在 Fill (填充) 中指定此註釋是否要填充陰影。例如，選擇 `Above` 或 `Below` 作為要填入的相關區域。如果您指定 `Between`，將會顯示另一個 `Value` 欄位，這兩個值之間的圖形區域將被填充。

   d. 針對 Axis (軸)，指定 `Value` 中的數字是否要參照與左側 Y 軸或右側 Y 軸關聯的指標 (如果此圖形包含多個指標)。

      您可以透過選擇註釋左側欄中的顏色方塊，來變更註釋的填充顏色。

   重複這些步驟，將多個水平註釋新增到同一個圖形。

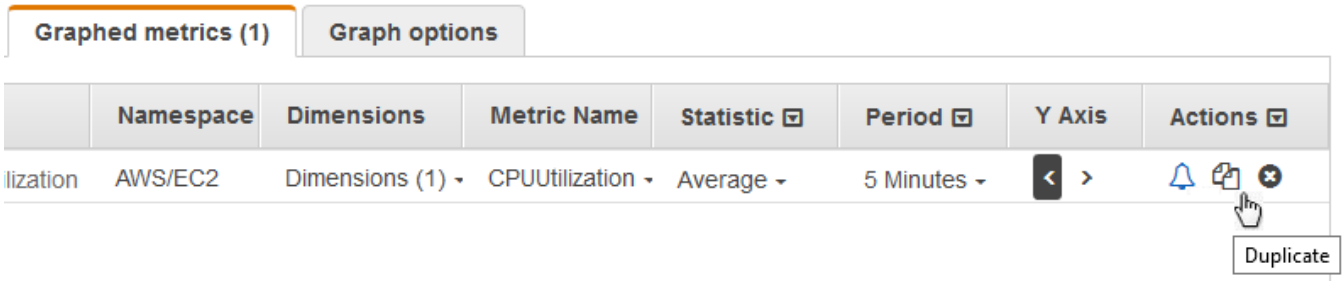
   若要隱藏註釋，請清除該註釋左側欄中的核取方塊。

   若要刪除註釋，請選擇 Actions (動作) 欄中的 x。
6. 若要變更重新整理間隔，請選擇 Refresh options (畫面更新選項)，然後選取 Auto refresh (自動畫面更新) 或選擇 1 Minute (1 分鐘)、2 Minutes (2 分鐘)、5 Minutes (5 分鐘) 或 15 Minutes (15 分鐘)。

複製指標

複製指標

1. 選擇 Graphed metrics (圖形化指標) 標籤。
2. 在 Actions (動作) 中選擇 Duplicate (複製) 圖示。



3. 視需要更新重複的指標。

將兩個圖形合併為一個

您可以將兩個不同的圖形合併為一個，然後產生的圖形會顯示兩個指標。如果您已經在不同的圖形中顯示不同的指標，並希望結合這些項目，或是想要使用來自不同區域的指標輕鬆建立單一圖形，則此功能非常有用。

若要將圖形合併到另一個圖形中，您可以使用要併入之圖形的 URL 或 JSON 來源。

將兩個圖形合併為一個

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 開啟您要合併到另一個圖形的圖形。若要這樣做，您可以選擇指標、所有指標，然後選擇要以圖形表示的指標。您也可以開啟儀表板，然後選取圖形並從圖形右上角的選單中選擇在指標中開啟，以在儀表板上開啟其中一個圖形。
3. 開啟圖形後，請執行下列其中一個動作：
   - 從瀏覽器列複製 URL。
   - 選擇來源索引標籤，然後選擇複製。
4. 開啟您要將上一個圖形併入的圖形。
5. 當您在指標檢視中開啟第二個圖形時，請選擇動作、合併圖形。
6. 輸入您先前複製的 URL 或 JSON，然後選擇合併。
7. 合併的圖形隨即顯示。左側的 Y 軸適用於原始圖形，而右側的 Y 軸則適用於併入的圖形。

> **Note**
>
> 如果您併入的圖形使用 METRINCE() 函數，則該併入圖形中的指標不會包含在合併圖形的 METRINCE() 計算中。

8. 若要將合併圖形儲存至儀表板，請依序選擇動作、新增至儀表板。

使用動態標籤

您可以搭配圖形使用動態標籤。動態標籤會將動態更新的值新增到所選取指標的標籤。您可以將各種值加入至標籤，如以下表格所示。

在標籤中顯示的動態值衍生自目前顯示在圖形上的時間範圍。標籤的這個動態部分會自動在儀表板或圖形重新整理時更新。

若您搭配搜尋表達式使用動態標籤，動態標籤便會套用到搜尋傳回的每個指標。

您可以使用 CloudWatch 主控台將動態值新增至標籤、編輯標籤、變更標籤欄中動態值的位置，以及進行其他自訂。

動態標籤

在動態標籤中，您可以使用下列與指標屬性相關的值：

動態標籤即時值	描述
${AVG}	目前圖形中所顯示時間範圍內的平均值。
${DATAPOINT_COUNT}	目前圖形中所顯示時間範圍內的資料點數目。
${FIRST}	目前圖形中所顯示時間範圍內最舊的指標值。
${FIRST_LAST_RANGE}	目前顯示在圖形中的最舊和最新資料點的指標值之間的差異。
${FIRST_LAST_TIME_RANGE}	目前顯示在圖形中的最舊和最新資料點之間的絕對時間範圍。
${FIRST_TIME}	目前圖形中所顯示時間範圍內的最舊資料點的時間戳記。

動態標籤即時值	描述
${FIRST_TIME_RELATIVE}	目前圖形中所顯示時間範圍內的現時與最舊資料點的時間戳記之間的絕對時間差異。
${LABEL}	指標預設標籤的表示法。
${LAST}	目前圖形中所顯示時間範圍內最新的指標值。
${LAST_TIME}	目前圖形中所顯示時間範圍內的最新資料點的時間戳記。
${LAST_TIME_RELATIVE}	目前圖形中所顯示時間範圍內的現時與最新資料點的時間戳記之間的絕對時間差異。
${MAX}	目前圖形中所顯示時間範圍內的最大值。
${MAX_TIME}	目前圖形中顯示的資料點中具有最高指標值的資料點的時間戳記。
${MAX_TIME_RELATIVE}	目前圖形中顯示的資料點中具有最高值的資料點的現時與時間戳記之間的絕對時間差異。
${MIN}	目前圖形中所顯示時間範圍內的最小值。
${MIN_MAX_RANGE}	目前圖形中顯示的資料點中具有最高和最低指標值的資料點之間的指標值的差異。
${MIN_MAX_TIME_RANGE}	目前圖形中顯示的資料點中具有最高和最低指標值的資料點之間的絕對時間範圍。
${MIN_TIME}	目前圖形中顯示的資料點中具有最低指標值的資料點的時間戳記。
${MIN_TIME_RELATIVE}	目前圖形中顯示的資料點中具有最低值的資料點的現時與時間戳記之間的絕對時間差異。
$ {支柱 (AccountId')}	量度的 AWS 帳戶 ID。
$ {支柱 (AccountLabel')}	針對擁有此量度之來源帳戶所指定的標籤，以 CloudWatch 跨帳戶觀察性表示。

動態標籤即時值	描述
${PROP('Dim.*dimension_name*')}	指定的維度值。使用維度的區分大小寫的名稱取代 *dimension_name* 。
$ {支柱 (MetricName')}	指標的名稱
${PROP('Namespace')}	指標的命名空間。
${PROP('Period')}	指標的期間，以秒為單位。
${PROP('Region')}	發佈量度的 AWS 區域。
${PROP('Stat')}	正在繪製圖形的指標統計資料。
${SUM}	目前圖形中所顯示時間範圍內所有值的總和。

例如，假設您有一個搜尋表達式 **`SEARCH(' {AWS/Lambda, FunctionName} Errors ', 'Sum')`**，此表達式會為您每一個 Lambda 函數尋找 `Errors`。若您將標籤設為 `[max: ${MAX} Errors for Function Name ${LABEL}]`，每個指標的標籤便是 [max: ***`number`*** Errors for Function Name ***`Name`***]。

您可向標籤新增最多六個動態值。您只能在每個標籤中使用 `${LABEL}` 預留位置一次。

修改圖形的時間範圍或時區格式

本節說明如何修改 CloudWatch 量度圖表上的日期、時間和時區格式。它還介紹了如何放大圖形以套用特定時間範圍。如需建立圖形的資訊，請參閱將指標圖形化。

> **Note**
>
> 如果儀表板的時間範圍短於儀表板圖形使用的期間，則會發生下列情況：
>
> - 即使這比儀表板時間範圍更長，圖形也會修改為顯示與該 Widget 相對應的一個完整期間的資料量。這可確保圖形上至少有一個資料點。
> - 此資料點期間的開始時間會向後調整，以確保至少可顯示一個資料點。

設定相對時間範圍

New interface

為圖形指定相對時間範圍

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Metrics (指標)，然後選擇 All metrics (所有指標)。在畫面右上角，您可以選取預先定義的時間範圍之一，範圍從 1 小時到 1 週 (1 小時、3 小時、12 小時、1 天、3 天或 1 週。或者，您也可以選擇 Custom (自訂) 來設定您自己的時間範圍。
3. 選擇 Custom (自訂)，然後選取方塊左上角中的 Relative (相對) 索引標籤。您可以在 Minutes (分鐘)、Hours (小時)、Days (天)、Weeks (週)、Months (月) 中指定時間範圍。
4. 指定時間範圍後，選擇 Apply (套用)。

Original interface

為圖形指定相對時間範圍

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Metrics (指標)，然後選擇 All metrics (所有指標)。在畫面右上角，您可以選取預先定義的時間範圍之一，範圍從 1 小時到 1 週 (1 小時、3 小時、12 小時、1 天、3 天或 1 週。或者，您也可以選擇 Custom (自訂) 來設定您自己的時間範圍。
3. 選擇 Custom (自訂)，然後方塊左上角的 Relative (相對)。您可以在 Minutes (分鐘)、Hours (小時)、Days (天)、Weeks (週) 或 Months (月) 中指定時間範圍。

設定絕對時間範圍

New interface

為圖形指定絕對時間範圍

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Metrics (指標)，然後選擇 All metrics (所有指標)。在畫面右上角，您可以選取預先定義的時間範圍之一，範圍從 1 小時到 1 週 (1 小時、3 小時、12 小時、1 天、3 天或 1 週。或者，您也可以選擇 Custom (自訂) 來設定您自己的時間範圍。
3. 選擇 Custom (自訂)，然後選取方塊左上角中的 Absolute (絕對) 索引標籤。使用行事曆選擇器或文字欄位方塊來指定時間範圍。

4. 指定時間範圍後，選擇 Apply (套用)。

Original interface

為圖形指定絕對時間範圍

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Metrics (指標)，然後選擇 All metrics (所有指標)。在畫面右上角，您可以選取預先定義的時間範圍之一，範圍從 1 小時到 1 週 (1 小時、3 小時、12 小時、1 天、3 天或 1 週。或者，您也可以選擇 Custom (自訂) 來設定您自己的時間範圍。
3. 選擇 Custom (自訂)，然後方塊左上角的 Absolute (絕對)。使用行事曆選擇器或文字欄位方塊來指定時間範圍。
4. 指定時間範圍後，選擇 Apply (套用)。

設定時區格式

New interface

指定圖形的時區

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Metrics (指標)，然後選擇 All metrics (所有指標)。在畫面右上角，您可以選取預先定義的時間範圍之一，範圍從 1 小時到 1 週 (1 小時、3 小時、12 小時、1 天、3 天或 1 週。或者，您也可以選擇 Custom (自訂) 來設定您自己的時間範圍。
3. 選擇 Custom (自訂)，然後選擇方塊右上角的下拉式選單。您可以將時區變更為 UTC 或者 Local time zone (本機時區)。
4. 做出變更後，請選擇 Apply (套用)。

Original interface

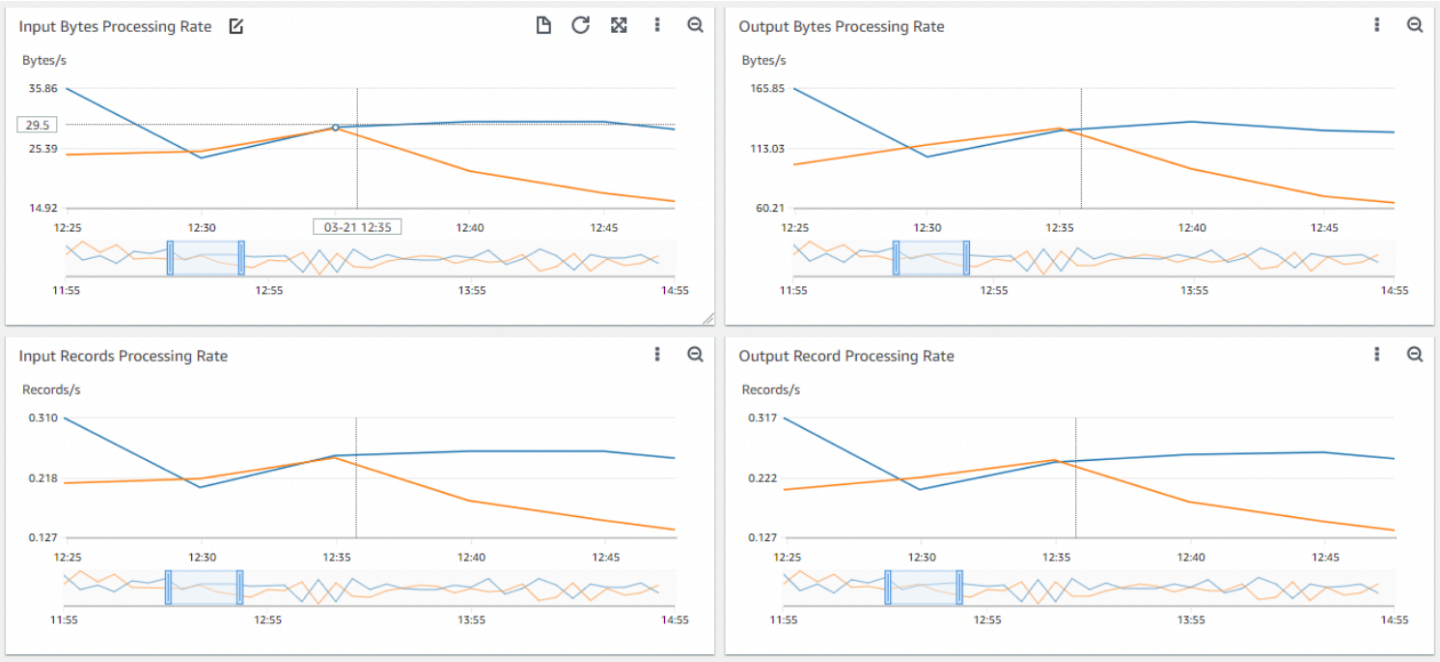
指定圖形的時區

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Metrics (指標)，然後選擇 All metrics (所有指標)。在畫面右上角，您可以選取預先定義的時間範圍之一，範圍從 1 小時到 1 週 (1 小時、3 小時、12 小時、1 天、3 天或 1 週。或者，您也可以選擇 Custom (自訂) 來設定您自己的時間範圍。

3. 選擇 Custom (自訂)，然後選擇方塊右上角的下拉式選單。您可以將時區變更為 UTC 或者 Local timezone (本機時區)。

放大折線圖或堆疊區域圖

在 CloudWatch 主控台中，您可以使用迷你地圖縮放功能來專注於線條圖和堆疊區域圖的區段，而不需要在放大和縮小檢視之間變更。例如，您可以使用迷你地圖縮放功能，將焦點放在折線圖中的峰值上，以便在同一時間軸中將尖峰與儀表板中的其他指標進行比較。本節中的程序介紹如何使用縮放功能。



在上圖中，縮放功能著重於折線圖中與輸入位元組處理速率相關的尖峰，同時還顯示儀表板中著重於同一時間軸中各區段的其他折線圖。

New interface

放大圖形

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Metrics (指標)，然後選擇 All metrics (所有指標)。
3. 選擇 Browse (瀏覽)，然後選取要繪製圖表的一或多個指標。
4. 選擇 Options (選項)，然後選取 Widget type (小工具類型) 下的 Line (線條)。
5. 選擇並拖曳要聚焦的圖表區域，然後放開拖曳。

6. 若要重設縮放，請選擇 Reset zoom (重設縮放) 圖示，這看起來像一個帶有減號 (-) 符號的放大鏡。

Original interface

放大圖形

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Metrics (指標)，然後選擇 All metrics (所有指標)。
3. 選擇 All metrics (所有指標)，然後選取要繪製圖表的指標。
4. 選擇 Graph options (圖表選項)。在 Widget type (小工具類型) 下，選取 Line (線條)。
5. 選擇並拖曳要聚焦的圖表區域，然後放開拖曳。
6. 若要重設縮放，請選擇 Reset zoom (重設縮放) 圖示，這看起來像一個帶有減號 (-) 符號的放大鏡。

> **Tip**
>
> 如果已建立包含折線圖或堆疊區域圖的儀表板，則可以前往儀表板並開始使用縮放功能。

修改圖形的 Y 軸

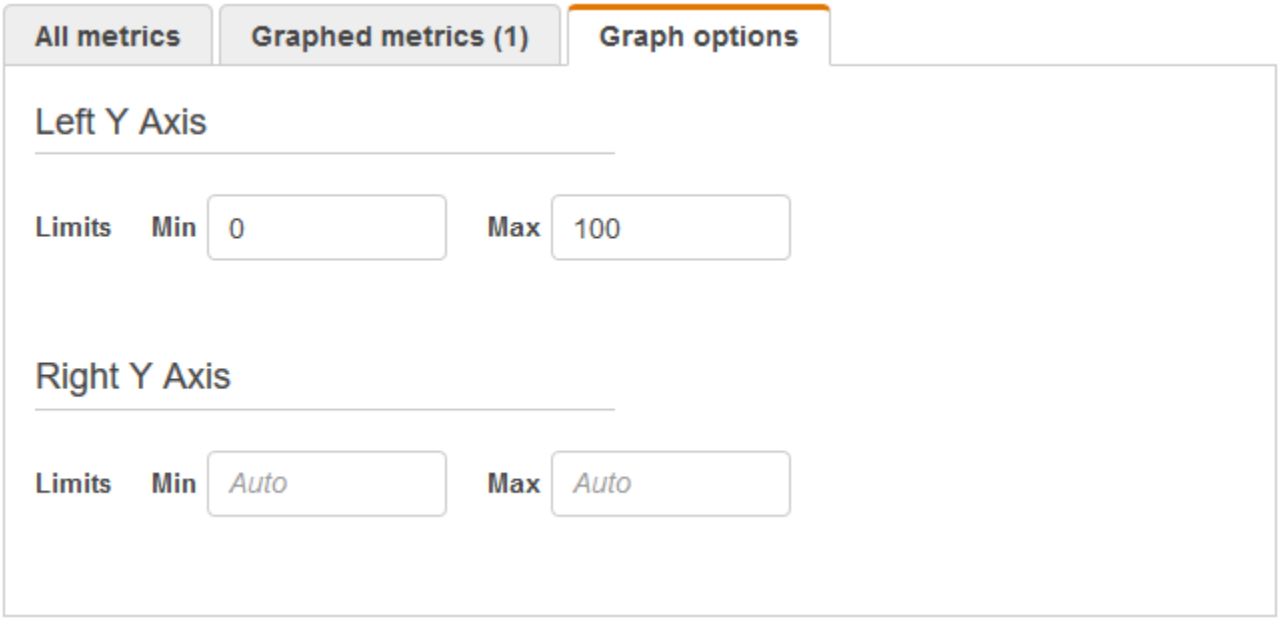
您可以設定圖形的 Y 軸自訂邊界，以利更佳查看資料。例如，您可以將 `CPUUtilization` 圖形邊界變更為 100%，以便容易查看 CPU 是低 (繪圖線接近圖形的底部) 或高 (繪圖線接近圖形的頂部)。

您可以在您圖形的兩個不同 Y 軸間切換。如果圖形包含的指標有不同的單位或指標值範圍差異很大，此功能將很有用。

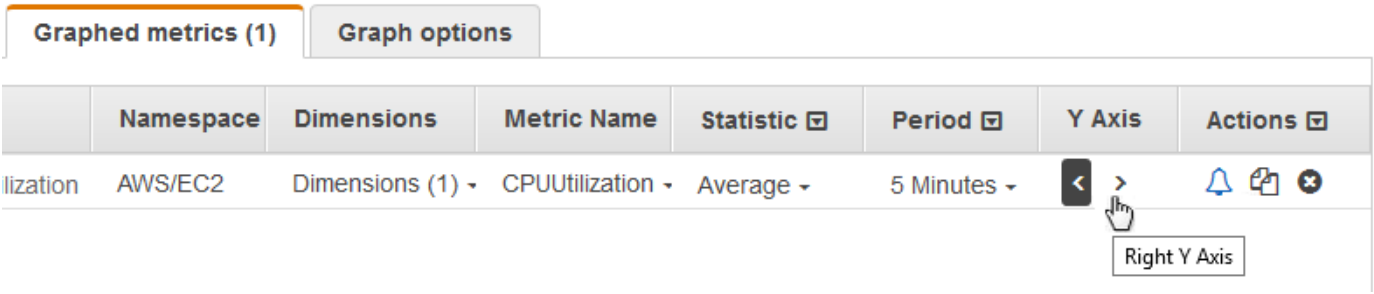
修改圖形的 Y 軸

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 指標。
3. 選取指標命名空間 (例如 EC2)，然後選取指標維度 (例如，Per-Instance Metrics (每個執行個體指標))。
4. All metrics (所有指標) 索引標籤會顯示命名空間中該維度的所有指標。若要將指標圖形化，請勾選指標旁的核取方塊。

5. 在 Graph options (圖形選項) 標籤中，指定 Left Y Axis (左 Y 軸) 的 Min (最小) 與 Max (最大) 值。Min (最小) 的值不可大於 Max (最大) 的值。



6. 若要建立第二個 Y 軸，請指定 Right Y Axis (右 Y 軸) 的 Min (最小) 和 Max (最大) 值。
7. 若要切換兩個 Y 軸，請選擇 Graphed metrics (圖表化指標) 標籤。在 Y Axis (Y 軸) 中選擇 Left Y Axis (左 Y 軸) 或 Right Y Axis (右 Y 軸)。



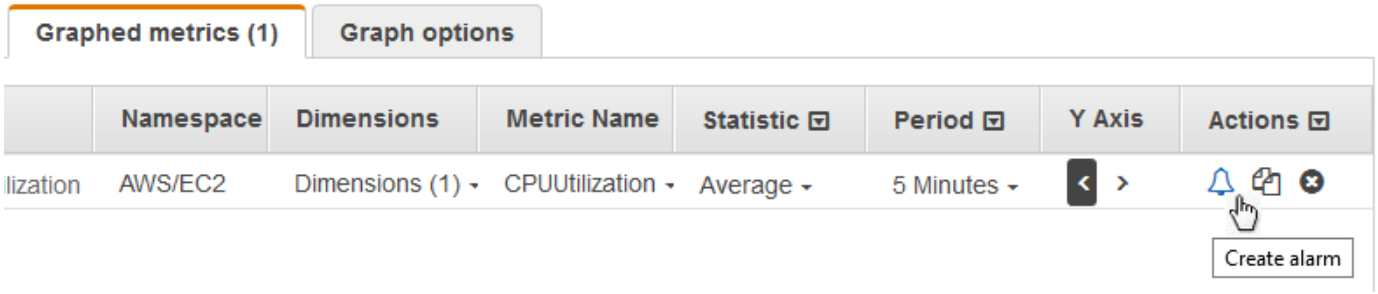
從圖形的指標建立警示

您可以繪製指標圖形，然後從圖形上的指標建立警示，它的好處是會為您填入許多警示欄位。

從圖形的指標建立警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 指標。
3. 選取指標命名空間 (例如 EC2)，然後選取指標維度 (例如，Per-Instance Metrics (每個執行個體指標))。

4. All metrics (所有指標) 索引標籤會顯示命名空間中該維度的所有指標。若要將指標圖形化，請勾選指標旁的核取方塊。
5. 若要建立指標的警示，請選擇 Graphed metrics (圖表化指標) 索引標籤。在 Actions (動作) 中選擇警示圖示。



6. 在 Conditions (條件) 下，選擇 Static (靜態) 或 Anomaly detection (異常偵測)，以指定要針對警示使用靜態臨界值或異常偵測模型。

   根據您的選擇，輸入警示條件的其餘資料。
7. 選擇 Additional configuration (其他組態)。針對 Datapoints to alarm (要警示的資料點)，請指定 (資料點) 必須處於 ALARM 狀態多少評估期間，才會觸發警示。如果此處的兩個值相符，您便可以建立警示，在許多連續期間違規時移至 ALARM 狀態。

   若要建立 N 個中有 M 個警示，請針對第一個值，指定低於您為第二個值所指定值的值。如需詳細資訊，請參閱 評估警示。
8. 針對 Missing data treatment (遺失資料處理)，選擇警示在遺失某些資料點時的行為。如需詳細資訊，請參閱 設定 CloudWatch 警示如何處理遺失的資料。
9. 選擇下一步。
10. 在 Notification (通知) 下，選取 SNS 主題來在警示處於 ALARM 狀態、OK 狀態或 INSUFFICIENT_DATA 狀態時進行通知。

    若要讓警示針對相同的警示狀態或不同警示狀態傳送多個通知，請選擇 Add notification (新增通知)。

    若要讓警示不傳送通知，請選擇 Remove (移動)。
11. 若要讓警示執行 Auto Scaling 或 EC2 動作，請選擇適當的按鈕，然後選擇警示狀態及要執行的動作。
12. 完成時，請選擇下一步。
13. 輸入警示的名稱與說明。名稱只能包含 ASCII 字元。然後選擇下一步。

14. 在 Preview and create (預覽及建立) 下，請確認資訊和條件都是您希望的內容，然後選擇 Create alarm (建立警示)。

使用 CloudWatch 異常偵測

當您啟用量度的異常偵測時，會 CloudWatch 套用統計和機器學習演算法。這些演算法會在使用者介入程度最低的情況下，持續分析系統和應用程式的指標、判斷正常基準以及表面異常情況。

演算法會產生異常偵測模型。模型會產生預期值範圍，代表正常指標行為。

您可以使用 AWS Management Console、、或 AWS SDK 啟用 AWS CLI異常偵測。 AWS CloudFormation您可以在提供的指標上啟用異常偵測，也可以針對自訂指標啟用異常偵測。 AWS 在設定為 CloudWatch 跨帳戶觀察性監視帳戶的帳戶中，除了監控帳戶中的指標之外，您還可以在來源帳戶中建立指標的異常偵測器。

您可以透過兩種方式來使用預期值的模型：

- 建立以指標預期值為基礎的異常偵測警示。這些類型的警示沒有用於判斷警示狀態的靜態臨界值。相反地，它們會根據異常偵測模型，將指標值與預期值進行比較。

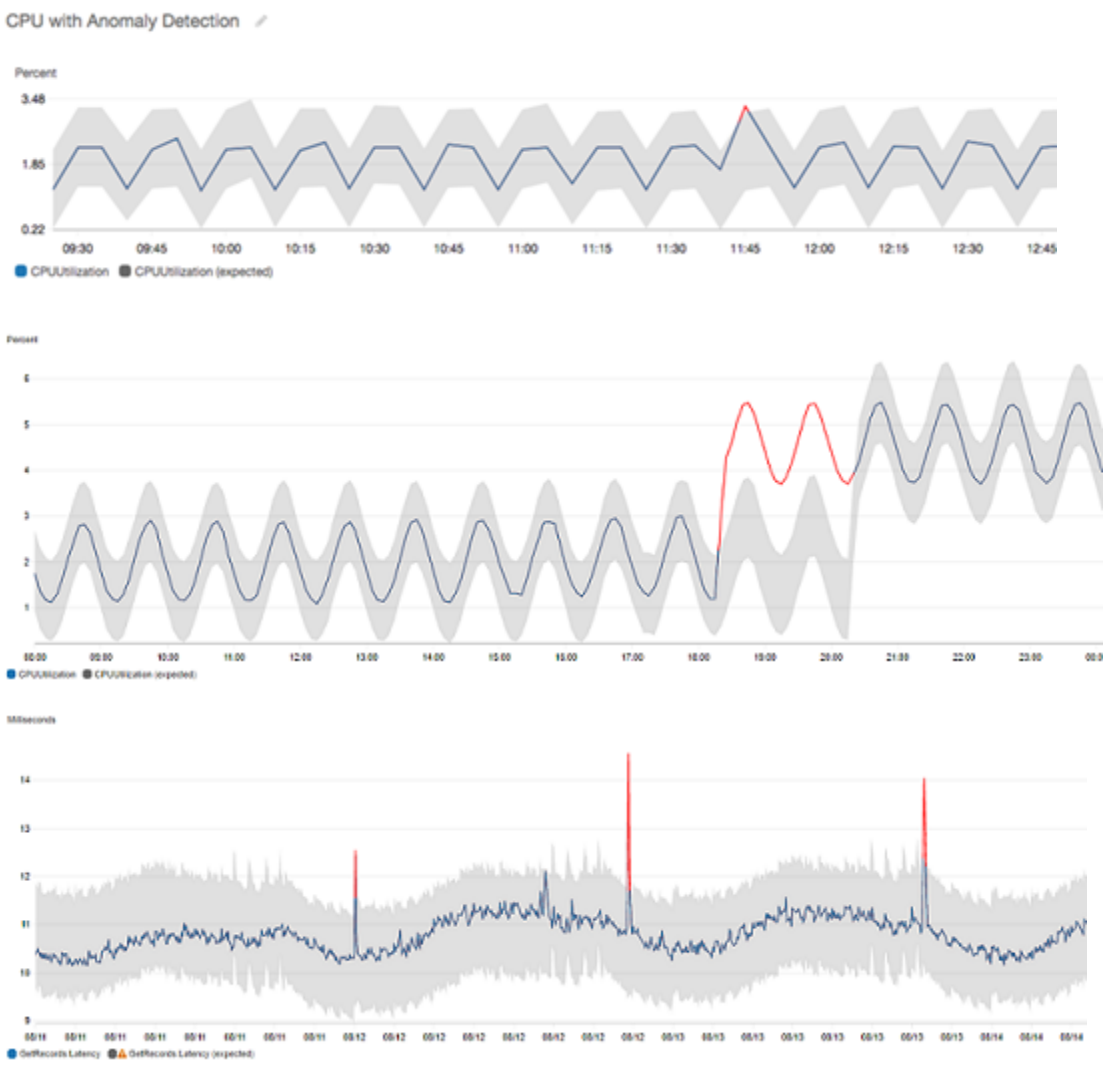
  您可以選擇警示觸發時機是在指標值超過預期值範圍、低於範圍，或者兩者同時發生時。

  如需詳細資訊，請參閱 根據異常偵測建立 CloudWatch 警示。
- 檢視指標資料圖表時，將預期值以區帶形式覆蓋到圖表上。這樣可以清楚地看出圖表中哪些值超出正常範圍。如需詳細資訊，請參閱 建立圖形。

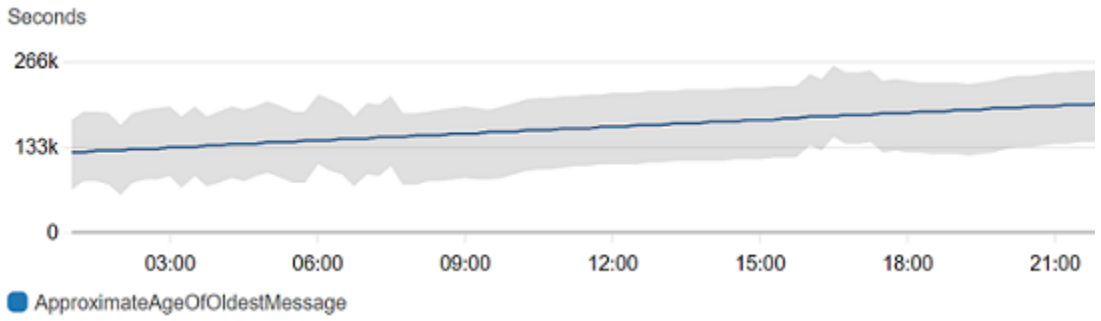
  您也可以使用含 `ANOMALY_DETECTION_BAND` 指標數學函數的 `GetMetricData` API 請求來擷取模型範圍的上下限值。如需詳細資訊，請參閱GetMetricData。

在具有異常偵測的圖表中，預期值的範圍顯示為灰色區帶。如果指標的實際值超出此區帶，則在該時間內會顯示為紅色。

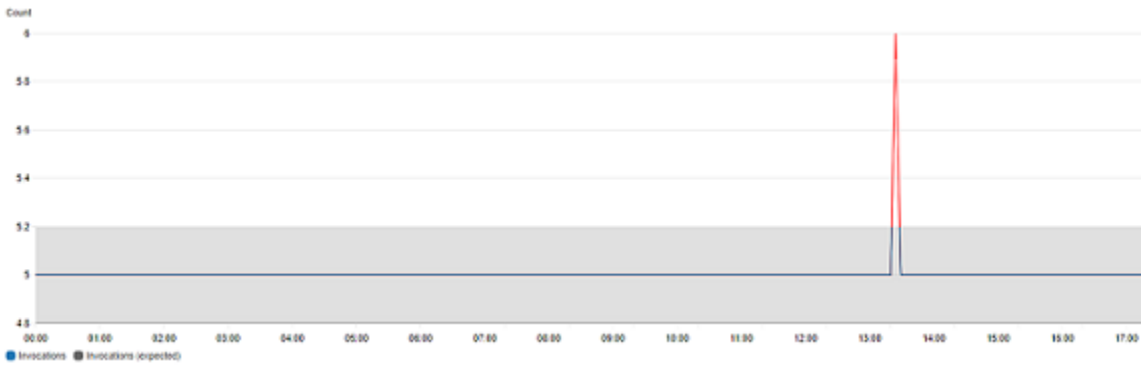
異常偵測演算法會考慮指標的季節性和趨勢變化。季節性變化可能是每小時、每日或每週，如下列範例所示。



較長範圍的趨勢可能是向下或向上。



異常偵測也適用於具有平面模式的指標。



CloudWatch 異常偵測的運作方式

當您為量度啟用異常偵測時，會將機器學習演算法 CloudWatch 套用至量度的過去資料，以建立量度預期值的模型。模型會評估指標的趨勢以及每小時、每日和每週模式。演算法可以最多兩週的指標資料為目標，但即使指標沒有完整兩週的資料，您仍可以對指標啟用異常偵測。

您可以指定異常偵測臨界值的值，該臨界值會與模型一起 CloudWatch 使用，以決定量度的「正常」值範圍。較高的異常偵測臨界值會產生較厚的「正常」值範圍。

機器學習模型是指標和統計資料所特有的。例如，如果您使用 AVG 統計資料來啟用指標的異常偵測，此模型就是 AVG 統計資料所特有的。

從 AWS 服務 CloudWatch 建立許多常見度量的模型時，可確保頻帶不會延伸到邏輯值之外。例如，EC2 實例的頻段將保持在 0 和 100 之間，並且頻段跟踪 CloudFront `Requests`（不能為`MemoryUtilization`負數）永遠不會低於零。

建立模型之後， CloudWatch 異常偵測會持續評估模型並對其進行調整，以確保其盡可能準確。這包括重新訓練模型，以在指標隨時間變化或突然變更時對其進行調整，並包含預測器，可改善季節性、尖峰或稀疏的指標模型。

對指標啟用異常偵測後，您可以選擇排除指標的指定時段，避免這些時段被用於模型的訓練。如此一來，您就可以排除部署或其他不尋常事件，避免這些事件被用於模型的訓練，確保建立的模型是最準確的。

針對警示使用異常偵測模型，會對您 AWS 的帳戶產生費用。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

指標數學上的異常偵測

指標數學上的異常偵測是一項功能，您可使用此功能根據輸出指標數學表達式建立異常偵測警示。您可以使用這些表達式來建立可視化異常偵測頻段的圖形。此功能支援基本算術函數、比較和邏輯運算子，

以及大多數其他函數。如需不受支援之函數的相關資訊，請參閱 Amazon 使用 CloudWatch 者指南中的使用指標數學運算。

您可以根據指標數學表達式建立異常偵測模型，類似於建立異常偵測模型的方式。您可以從 CloudWatch 主控台將異常偵測套用至量度數學運算式，並選取異常偵測作為這些運算式的臨界值類型。

> **Note**
>
> 只能在最新版的指標使用者介面中啟用和編輯指標數學的異常偵測。在新版介面中根據指標數學表達式建立異常偵測器時，您可以在舊版中進行檢視，但無法編輯。

如需如何為異常偵測和指標數學建立警示和模型的相關資訊，請參閱下列章節：

- 根據異常偵測建立 CloudWatch 警示
- 根據度量數學運算式建立 CloudWatch 警示

您也可以使用 CloudWatch API 搭配、和，根據度量數學運算式建立`PutAnomalyDetector`、`DeleteAnomalyDetector`刪除和`DescribeAnomalyDetectors`探索異常偵測模型。如需這些 API 動作的相關資訊，請參閱 Amazon CloudWatch API 參考中的以下各節。

- PutAnomalyDetector
- DeleteAnomalyDetector
- DescribeAnomalyDetectors

如需異常偵測警示計價方式的詳細資訊，請參閱 Amazon CloudWatch 定價。

使用指標數學

Metric math 可讓您查詢多個 CloudWatch 量度，並使用數學運算式根據這些量度建立新的時間序列。您可以在 CloudWatch 主控台上視覺化產生的時間序列，並將其新增至儀表板。使用 AWS Lambda 指標為例，您可以將指`Errors`標除以指`Invocations`標以獲得錯誤率。然後將產生的時間序列新增至 CloudWatch 儀表板上的圖表。

您也可以使用 `GetMetricData` API 操作，以程式設計方式執行指標數學。如需詳細資訊，請參閱GetMetricData。

將數學運算式新增至 CloudWatch 圖表

您可以將數學運算式新增至 CloudWatch 儀表板上的圖形。每個圖形最多可使用 500 個指標和表達式，因此您只能在圖形有 499 個以下的指標時新增數學表達式。即使圖形上未顯示所有指標， 也適用此情況。

新增數學表達式到圖形

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 建立或編輯圖形。圖形中至少需要有一個指標。
3. 選擇 Graphed metrics (圖形化指標)。
4. 選擇 Math expression (數學運算式)、Start with empty expression (以空表達式開始)。表達式出現新行。
5. 在新行的 Details (詳細資訊) 欄下，輸入數學運算式。指標數學語法和函數區段中的表格會列出您可以在運算式中使用的函數。

   若要使用指標或另一個表達式的結果做為此表達式公式的一部分，請使用 Id 欄中顯示的值：例如 m1+m2 或 e1-MIN(e1)。

   您可以變更 Id 的值。它可以包含數字、字母和底線，而且必須以小寫字母開始。將 Id 的值變更為較有意義的名稱，也可以讓圖形更好了解：例如，將 m1 和 m2 變更為 errors (錯誤) 和 requests (請求)。

   > **Tip**
   >
   > 選擇 Math Expression (數學運算式) 旁的向下箭頭，以查看建立運算式時可以使用的支援函數清單。

6. 針對表達式的 Label (標記) 欄，輸入描述表達式計算內容的名稱。

   如果表達式的結果是一系列時間序列，這些時間序列的每一個都會以不同的顏色顯示在使用不同行的圖形上。圖形底下是一個圖例代表在圖形中每一行。對於會產生多個時間序列的單一表達式，那些時間序列的圖例標題格式為 ***Expression-Label Metric-Label***。例如，若圖形包含具錯誤標籤和運算式 FILL(METRICS(), 0) 的指標，其具有標籤 Filled With 0: (填充 0)，圖例中的一行是 Filled With 0: Errors (填充 0：錯誤)。您可以將`#####`設為空白，讓圖例只顯示原始的指標標籤。

   當一個表達式在圖形上產生時間序列的陣列時，您就無法變更這些時間序列各自使用的顏色。

7. 新增所需的運算式之後，您可以隱藏一些原始指標以簡化圖形。若要隱藏指標或運算式，請清除 Id 欄位左側的核取方塊。

指標數學語法和函數

以下章節說明可用於指標數學的函數。所有函數都必須為大寫 (如 AVG)，而所有指標和數學表達式的 Id 欄位則必須以小寫字母開頭。

任何數學運算式的最終結果必須是單一時間序列，或一系列的時間序列。有些函數會產生純量數。您可以使用這些功能較大的函數，其最終會產生時間序列。例如，採用單一時間序列的 AVG 會產生純量數，所以它無法成為最終的表達式結果。但您可以在函數 m1-AVG(m1) 中使用它來顯示每一個別資料點，和該時間序列平均值之間差異的時間序列。

資料類型縮寫

有些數僅對於特定類型的資料有效。函數表會使用以下清單中的縮寫表示每個函數中支援的資料類型：

- S 代表純量數，例如 2、-5 或 50.25。
- TS 是時間序列 (一段時間內單一 CloudWatch測量結果的一系列值)：例如，`i-1234567890abcdef0`過去三天的`CPUUtilization`測量結果。
- TS [] 是一系列的時間序列，例如多個指標的時間序列。
- 字串[] 是字串陣列。

METRICS() 函數

METRICS () 函數會傳回請求中所有的指標。數學表達式不包含在內。

您可以使用較大運算式內的 METRICS()，其會產生單一時間序列，或一系列的時間序列。例如，運算式 SUM(METRICS()) 會傳回一個時間序列 (TS)，其為所有圖表化指標值的總和。METRICS()/100 會傳回一系列的時間序列，其每一個都是時間序列顯示每個除以 100 之其中一個指標資料點。

您可以使用 METRICS() 函數和字串，僅傳回圖表化指標，其中在 Id 欄位中包含該字串。例如，運算式 SUM(METRICS("errors")) 會傳回一個時間序列，其為 Id 欄位中所有含「errors」的圖表化指標值的總和。您也可以使用 SUM([METRICS(“4xx”), METRICS(“5xx”)]) 來符合多個字串。

基本算術函數

下表列出受支援的基本算術函數。時間序列的遺失值視為 0。如果資料點的值會導致函數除以零，資料點會被捨棄。

作業	引數	範例
算術運算子: + - * / ^	S, S	PERIOD(m1)/60
	S, TS	5 * m1
	TS, TS	m1 - m2
	S, TS[]	SUM(100/[m1, m2])
	TS, TS[]	AVG(METRICS())
		METRICS()*100
一元減法 -	S	-5*m1
	TS	-m1
	TS[]	SUM(-[m1, m2])

比較與邏輯運算子

您可以使用比較和邏輯運算子搭配任一對時間序列或一對單一純量值。在您使用一對時間序列的比較運算子時，運算子會傳回一個時間序列，其中每個資料點都是 0 (false) 或 1 (true)。如果您在一對純量值使用比較運算子，則傳回單一純量值 (0 或 1)。

當兩個時間序列之間使用比較運算子，且只有一個時間序列具有特定時間戳記的值時，函數會將另一個時間序列中的遺失值視為 0。

您可以將邏輯運算子與比較運算子搭配使用，以建立更複雜的函數。

下表列出支援的運算子。

運算子類型	支援的運算子
比較運算子	==
	!=
	<=

運算子類型	支援的運算子
	>= < >
邏輯運算子	AND 或 && OR 或 ||

如要查看這些運算子是如何使用的，假設我們有兩個時間序列：metric1 值為 `[30, 20, 0, 0]`，而 metric2 值為 `[20, -, 20, -]`，其中 - 表示該時間戳記沒有值。

表達式	輸出
(metric1 < metric2)	0, 0, 1, 0
(metric1 >= 30)	1, 0, 0, 0
(metric1 > 15 AND metric2 > 15)	1, 0, 0, 0

支援指標數學的函數

下表說明您可以用於數學運算式的函數。請使用大寫字母輸入所有函數。

任何數學運算式的最終結果必須是單一時間序列，或一系列的時間序列。以下各節中有些函數會產生純量數。您可以使用這些功能較大的函數，其最終會產生時間序列。例如，採用單一時間序列的 AVG 會產生純量數，所以它無法成為最終的表達式結果。但您可以在函數 m1-AVG(m1) 中使用它來顯示每一個別資料點，和該資料點平均值之間差異的時間序列。

在下表中，Examples (範例) 欄的每個範例是產生單一時間序列或一系列時間序列的運算式。這些範例顯示，傳回純量數的函數，如何當做會產生單一時間序列之有效運算式的一部分來使用。

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
ABS	TS TS[]	TS TS[]	傳回每個資料點的絕對值。	ABS(m1-m2) MIN(ABS([m1, m2])) ABS(METRICS())	✓
ANOMALY_DETECTION_BAND	TS TS、S	TS[]	傳回指定指標適用的異常偵測範圍。此範圍包含兩個時間序列，一個代表指標「正常」預期值的上限，另一個代表下限。此函數可能會使用兩個引數。第一個是要建立範圍的指標 ID。第二個引數是用於範圍的標準偏差數字。如果您沒有指定此引數，則會使用預設值 2。如需詳細資訊，請參閱 使用 CloudWatch 異常偵測。	ANOMALY_DETECTION_BAND(m1) ANOMALY_DETECTION_BAND(m1,4)	
AVG	TS TS[]	S TS	單一時間序列的 AVG 會傳回一個純量，代表指標中所有資料點的平均值。一系列的時間序列的 AVG 會傳回單一時間序列。遺失值視為 0。 Note 如果您希望函數傳回純量，建議您不要在 CloudWatc	SUM([m1,m2])/AVG(m2) AVG(METRICS())	✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
			h 警報中使用此函數。例如 AVG(m2)。每當警示評估是否要變更狀態時，都會 CloudWatch 嘗試擷取比指定為「評估期間」數目更多的資料點。當請求額外的資料時，此函數會有所不同。若要將此功能與警示搭配使用，尤其是有「自動擴展」動作的警示，建議您將警示設定為使用 N 個資料點中的 M 個，其中 M < N。		
CEIL	TS TS[]	TS TS[]	傳回每個指標的上限。上限是大於或等於每個值的最小整數。	CEIL(m1) CEIL(METRICS()) SUM(CEIL(METRICS()))	✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
DATAPOINT_COUNT	TS TS[]	S TS	傳回所報告值的資料點計數。這對於計算稀疏指標的平均值非常有用。 **Note** 我們建議您不要在 CloudWatch 鬧鐘中使用此功能。每當警示評估是否要變更狀態時，都會 CloudWatch 嘗試擷取比指定為「評估期間」數目更多的資料點。當請求額外的資料時，此函數會有所不同。	SUM(m1) / DATAPOINT_COUNT(m1) DATAPOINT_COUNT(METRICS())	✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
DB_PERF_INSIGHTS	字串, 字串, 字串 字串、字串、字串[]	TS (如果指定單一字串) TS[] (如果指定字串陣列)	傳回 Amazon Relational Database Service 和 Amazon DocumentDB (with MongoDB compatibility) 這類資料庫的 Performance Insights 計數器指標。此函數會傳回您透過直接查詢 Performance Insights API 的方式可以取得的相同資料量。您可以在中使用這些量度 CloudWatch 來繪製和建立警示。 **Important** 使用此函數時，必須指定資料庫的唯一資料庫資源 ID。這與資料庫識別碼不同。若要在 Amazon RDS 主控台中查找資料庫資源 ID，請選擇資料庫執行個體來查看其詳細資訊。然後選擇 Configuration (組態) 標籤。資	DB_PERF_INSIGHTS('RDS', 'db-ABCDEFGHIJKLMNOPQRSTUVWXY1', 'os.cpuUtilization.user.avg') DB_PERF_INSIGHTS('DOCDB, 'db-ABCDEFGHIJKLMNOPQRSTUVWXY1', ['os.cpuUtilization.idle.avg', 'os.cpuUtilization.user.max'])	

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
			源 ID 顯示在組態區段中。 DB_PERF_INSIGHTS 也引進了次分鐘間隔的 DBLoad 指標。 使用此函數擷取的 Performance Insights 指標不會儲存在中 CloudWatch。因此，某些 CloudWatch 功能 (例如跨帳戶可觀察性、異常偵測、量度資料流、指標總管和指標洞見) 無法與您使用 DB_PERF_INSIGETSIGETSION 擷取的 Performance Insights 能洞見指標搭配使用。 使用 DB_PERF_INSIGETICS 函數的單個請求可以檢索以下數量的數據點。 • 高解析度期間的 1080 個資料點 (1 秒、10 秒、30 秒) • 標準解析度期間的 1440 個資料點 (1		

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
			米、5 米、1 小時、1 維) DB_PERF_INSIGHTS 函數僅支援下列週期長度： • 1 秒鐘 • 10 秒 • 30 秒 • 1 分鐘 • 5 分鐘 • 1 小時 • 1 天 如需 Amazon RDS Performance Insights 計數器指標的更多資訊，請參閱 Performance Insights 計數器指標。 如需 Amazon DocumentDB Performance Insights 計數器指標的更多資訊，請參閱適用於計數器指標的 Performance Insights。 Note 由 DB_PERF_INSIGHTS 擷取		

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
			的次分鐘精細度高解析度指標，僅適用於 DBLoad 指標，或者如果您已以較高解析度啟用「增強型監控」，則適用於作業系統指標。如需 Amazon RDS 增強型監控的更多資訊，請參閱使用增強型監控來監控 OS 指標。 您可以使用 DB_PERF_INSIGHTS 函數建立高解析度警示，最長時間範圍為三小時。您可以使用 CloudWatch 主控台繪製任何時間範圍內使用 DB_PERF_INSIGETICS 函式擷取的度量圖形。		

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
DIFF	TS TS[]	TS TS[]	傳回時間序列中每個值與該時間序列的上述值之間的差異。	DIFF(m1)	✓
DIFF_TIME	TS TS[]	TS TS[]	傳回時間序列中每個值的時間戳記與該時間序列的上述值之間的時間戳記之間的差異，以秒為單位。	DIFF_TIME(METRICS())	✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
FILL	TS, [S | REPEA | LINEAR TS[], [TS | S | REPEA | LINEAR	TS TS[]	填入時間序列的遺失值。有幾個選項可用來做為遺失值的填充物： • 您可指定用作填充值的值。 • 您可指定用作填充值的指標。 • 您可以使用 REPEATE 關鍵字，以在遺失值之前使用指標的最新實際值來填入遺失值。 • 您可以使用 LINEAR 關鍵字來填入遺失值，這些值會在間隙開頭和結尾之間建立線性插值。 **Note** 當您在警示中使用此函數時，如果您的指標發佈略有延遲，且最近的一分鐘從來沒有資料，則可能會遇到問題。在此例中，FILL 將遺失的資料點替換為請求值。這會導致指標的	FILL(m1,10) FILL(METRICS(), 0) FILL(METRICS(), m1) FILL(m1, MIN(m1)) FILL (m1, REPEAR) FILL(METRICS(), LINEAR)	✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
			最新資料點一律是 FILL 值，這可能會導致警示卡在 OK 狀態或 ALARM 狀態。您可以使用「N 中取 M」警示來解決此問題。如需詳細資訊，請參閱 評估警示。		
FIRST LAST	TS[]	TS	傳回從時間序列陣列的第一個或最後一個時間序列。與 SORT 函數一起使用時，這非常實用。它也可以用來從 ANOMALY_DETECTION_BAND 函數取得高和低閾值。	IF(FIRST(SORT(METRICS(), AVG, DESC))>100, 1, 0) 查看陣列的頂部指標，該陣列由 AVG 排序。然後，它會針對每個資料點傳回 1 或 0 (取決於該資料點值是否大於 100)。 LAST(ANOMALY_DETECTION_BAND(m1)) 會傳回異常預測帶的上限。	✓
FLOOR	TS TS[]	TS TS[]	傳回每個指標的下限。下限是小於或等於每個值的最大整數。	FLOOR(m1) FLOOR(METRICS())	✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
IF	IF 運算式	TS	使用 IF 與比較運算子一起篩選出時間序列的資料點，或建立由多個定序時間序列組成的混合時間序列。如需詳細資訊，請參閱 使用 IF 表達式。	如需範例，請參閱 使用 IF 表達式。	✓
INSIGHT_RULE_METRIC	INSIGHT_RULE_METRIC(ruleName, metricName)	TS	使用 INSIGHT_RULE_METRIC 來從 Contributor Insights 內的規則擷取統計資料。如需詳細資訊，請參閱 繪製規則產生的指標。		
LAMBDA	拉姆達（LambdaFunctionName[，可選參數]*）	TS TS{}	呼叫 Lambda 函數，以查詢資料來源中不是指標 CloudWatch。如需詳細資訊，請參閱 如何將引數傳遞給 Lambda 函數。		
LOG	TS TS[]	TS TS[]	時間序列的 LOG 會傳回時間序列中每個值的自然對數值。	LOG(METRICS())	✓
LOG10	TS TS[]	TS TS[]	時間序列的 LOG10 會傳回時間序列中每個值的以 10 為底的對數值。	LOG10(m1)	✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
MAX	TS TS[]	S TS	單一時間序列的 MAX 會傳回一個純量，代表指標中所有資料點的最大值。 如果您輸入的時間序列的陣列，MAX 函數會建立並傳回由每個資料點的最高值組成的時間序列的時間序列的時間序列。 **Note** 如果您希望函數傳回純量，建議您不要在 CloudWatch 警報中使用此函數。例如，MAX(m2)每當警示評估是否要變更狀態時，都會 CloudWatch 嘗試擷取比指定為「評估期間」數目更多的資料點。當請求額外的資料時，此函數會有所不同。	MAX(m1)/m1 MAX(METRICS())	✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
METRIC_COUNT	TS[]	S	傳回時間序列陣列的指標數。	m1/METRIC_COUNT(METRICS())	✓
METRICS	null string	TS[]	該指標（ ）函數返回請求中的所有 CloudWatch 度量。數學表達式不包含在內。 您可以使用較大運算式內的 METRICS()，其會產生單一時間序列，或一系列的時間序列。 您可以使用 METRICS() 函數和字串，僅傳回圖表化指標，其中在 Id 欄位中包含該字串。例如，運算式 SUM(METRICS("errors")) 會傳回一個時間序列，其為 Id 欄位中所有含「errors」的圖表化指標值的總和。您也可以使用 SUM([METRICS(“4xx”), METRICS(“5xx”)]) 來符合多個字串。	AVG(METRICS()) SUM(METRICS("errors"))	✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
MIN	TS TS[]	S TS	單一時間序列的 MIN 會傳回一個純量，代表指標中所有資料點的最小值。 如果您輸入的時間序列的陣列，MIN 函數會建立並傳回由每個資料點的最低值組成的時間序列的時間序列的時間序列。 如果您輸入的時間序列的陣列，MIN 函數會建立並傳回由每個資料點的最高值組成的時間序列的時間序列的時間序列。 **Note** 如果您希望函數傳回純量，建議您不要在 CloudWatch 警報中使用此函數。例如，`MIN(m2)`每當警示評估是否要變更狀態時，都會 CloudWatch 嘗試擷取比指定為「評估期	m1-MIN(m1) MIN(METRICS())	✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
			間」數目更多的資料點。當請求額外的資料時，此函數會有所不同。		

函式	引數	傳回類型*	描述	範例	支援跨帳戶？

MINUTE
 	
HOUR
 	
DAY
 	
DATE
 	
MONTH
 	
YEAR
 	
EPOCH
	TS	TS	這些函數採用時間序列的期間和範圍，並傳回一個新的非稀疏時間序列，其中每個值均根據其時間戳記。
			
			• MINUTE 傳回 0 到 59 之間的非稀疏整數時間序列，表示原始時間序列中每個時間戳記的 UTC (分鐘)。
			• HOUR 傳回 0 到 23 之間的非稀疏整數時間序列，表示原始時間序列中每個時間戳記的 UTC (小時)。
			• DAY 傳回 1 到 7 之間的非稀疏整數時間序列，表示原始時間序列中每個時間戳記的 UTC (週幾)。1 表示週一，7 表示週日。
			• DATE 傳回 1 到 31 之間的非稀疏整數時間序列，表示原始時間序列中每個時間戳記的 UTC (號)。
			• DAY 傳回 1 到 7 之間的非稀疏整數時間序列，表示原始時間序列中每個時間戳記的
				MINUTE(m1)
				
				IF(DAY(m1)<6,m1) 只會傳回平日 (星期一至星期五) 的指標。
				
				IF(MONTH(m1) == 4,m1) 只會傳回 4 月份發佈的指標。
					✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
			UTC (月份)。1 表示 1 月，12 表示 12 月。 • YEAR 傳回非稀疏整數時間序列，表示原始時間序列中每個時間戳記的 UTC (年份)。 • EPOCH 傳回非稀疏整數時間序列，表示自原始時間序列中每個時間戳記的 Epoch 以來的 UTC 時間 (秒)。Epoch 是 1970 年 1 月 1 日。		
PERIOD	TS	S	傳回指標的期間，以秒為單位。有效輸入為指標，而不是其他運算式的結果。	m1/PERIOD(m1)	✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
RATE	TS TS[]	TS TS[]	傳回每秒的指標變更速率。此計算為最新資料點值和之前的資料點值之間的差異值，再除以兩個值之間的時間差異 (以秒為單位)。 **Important** 在具有稀疏資料的量度上使用 RATE 函數的運算式上設定警示可能會出現不可預測的行為，因為評估警示時擷取的資料點範圍可能會因上次發佈資料點的時間而有所不同。	RATE(m1) RATE(METRICS())	✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
REMOVE_EMPTY	TS[]	TS[]	從時間序列陣列中刪除沒有資料點的任何時間序列。結果是時間序列的陣列，其中每個時間序列至少包含一個資料點。 **Note** 我們建議您不要在 CloudWatch 鬧鐘中使用此功能。每當警示評估是否要變更狀態時，都會 CloudWatch 嘗試擷取比指定為「評估期間」數目更多的資料點。當請求額外的資料時，此函數會有所不同。	REMOVE_EMPTY(METRICS())	✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
RUNNING_SUM	TS TS[]	TS TS[]	傳回與原始時間序列中的值的執行總和的時間序列。 **Note** 我們建議您不要在 CloudWatch 鬧鐘中使用此功能。每當警示評估是否要變更狀態時，都會 CloudWatch 嘗試擷取比指定為「評估期間」數目更多的資料點。當請求額外的資料時，此函數會有所不同。	RUNNING_SUM([m1,m2])	✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
SEARCH	搜尋表達式	一或多個 TS	傳回一或多個符合您指定之搜尋條件的時間序列。SEARCH 函數能讓您將多個相關的時間序列新增到有一個表達式的圖形。圖形會動態更新，以包含後續新增的新指標，及符合搜尋條件。如需詳細資訊，請參閱 在圖形中使用搜尋運算式。 您無法根據 SEARCH 表達式建立警示。這是因為搜尋表達式會傳回多個時間序列，並且基於數學表達式的警示只能監看一個時間序列。 如果您以 CloudWatch 跨帳戶可觀察性登入監控帳戶，則搜尋功能會在來源帳戶和監視帳戶中尋找指標。		✓
SERVICE_QUOTA	為用量指標的 TS	TS	傳回指定用量指標的服務配額。您可以使用此項目來視覺化您目前用量與配額相較的情況，以及設定警示，在您接近配額時發出警告。如需詳細資訊，請參閱 AWS 使用量度。		✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
SLICE	(TS[], S, S) 或 (TS[], S)	TS[] TS	擷取時間序列陣列的一部分。這與 SORT 結合時特別實用。例如，您可以從時間序列陣列中排除最上層的結果。 您可以使用兩個純量引數來定義一組要傳回的時間序列。這兩個純量定義要傳回的數列開始 (包括) 和結束 (不包括)。該陣列是零索引的，因此陣列中的第一個時間序列是時間序列 0。或者，您可以只指定一個值，並 CloudWatch 返回以該值開始的所有時間序列。	SLICE(SORT(METRICS(), SUM, DESC), 0, 10) 從要求中具有最高 SUM 值的指標陣列傳回 10 個指標。 SLICE(SORT(METRICS(), AVG, ASC), 5) 按 AVG 統計資料排序指標的陣列，然後傳回除了具有最低 AVG 的 5 之外的所有時間序列。	✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
SORT	(TS[], FUNCT SORT_(R) (TS[], FUNCT SORT_(R, S)	TS[]	根據您指定的函數排序時間序列的陣列。您使用的函數可以是 AVG、MIN、MAX 或 SUM。排序順序可以是 ASC 升冪 (最低值為先排序) 或 DESC 以先排序較高的值。您可以選擇性在排序順序後指定數字，以做為限制。例如，指定限制為 5 只會傳回排序的前 5 個時間序列。 當此數學函數顯示在圖形上時，也會對圖形中每個指標的標籤進行排序和編號。	SORT(METRICS(), AVG, DESC, 10) 計算每個時間序列的平均值，在排序開始時使用最高值的時間序列進行排序，並僅傳回具有最高平均值的 10 個時間序列。 SORT(METRICS(), MAX, ASC) 由最大統計指標的陣列排序，然後按升冪排列傳回所有值。	✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
STDDEV	TS TS[]	S TS	單一時間序列的 STDDEV 會傳回一個純量，代表指標中所有資料點的標準偏差。一系列的時間序列的 STDDEV 會傳回單一時間序列。 **Note** 如果您希望函數傳回純量，建議您不要在 CloudWatch 警報中使用此函數。例如，STDDEV(m2) 每當警示評估是否要變更狀態時，都會 CloudWatch 嘗試擷取比指定為「評估期間」數目更多的資料點。當請求額外的資料時，此函數會有所不同。	m1/STDDEV(m1) STDDEV(METRICS())	✓

函式	引數	傳回類型*	描述	範例	支援跨帳戶？
SUM	TS TS[]	S TS	單一時間序列的 SUM 會傳回一個純量，代表指標中所有資料點的值總和。一系列的時間序列的 SUM 會傳回單一時間序列。 **Note** 如果您希望函數傳回純量，建議您不要在 CloudWatch 警報中使用此函數。例如 SUM(m1)。每當警示評估是否要變更狀態時，都會 CloudWatch 嘗試擷取比指定為「評估期間」數目更多的資料點。當請求額外的資料時，此函數會有所不同。	SUM(METRICS())/SUM(m1) SUM([m1,m2]) SUM(METRICS("errors"))/SUM(METRICS("requests"))*100	✓
TIME_SERIES	S	TS	傳回非稀疏時間序列，其中會將每個值設定為純量引數。	TIME_SERIES(MAX(m1)) TIME_SERIES(5*AVG(m1)) TIME_SERIES(10)	✓

*使用僅傳回純量數的函數是無效的，因為所有運算式的最終結果必須是單一時間序列或一系列的時間序列。反之，使用這些函數為傳回時間序列的較大運算式的一部分時。

使用 IF 表達式

使用 IF 與比較運算子一起篩選出時間序列的資料點，或建立由多個定序時間序列組成的混合時間序列。

IF 會使用以下引數：

```
IF(condition, trueValue, falseValue)
```

如果條件資料點的值為 0，條件判斷值為 FALSE；如果條件的值為任何其他值 (無論該值為正或負)，則判斷值為 TRUE。如果條件是一個時間序列，它會分別評估每個時間戳記。

以下列出有效的語法。對於這些每一個語法而言，輸出是一個單一的時間序列。

- IF(TS ***#####*** S, S | TS, S | TS)

> **Note**
>
> 如果TS comparison operator S是 TRUE 但metric2沒有對應的資料點，則輸出將為 0。

- IF(TS, TS, TS)
- IF(TS, S, TS)
- IF(TS, TS, S)
- IF(TS, S, S)
- IF(S, TS, TS)

下列各節提供這些語法的詳細資訊和範例。

IF(TS ***#####*** S, scalar2 | metric2, scalar3 | metric3)

對應的輸出時間序列值：

- 具有 scalar2 或 metric2 的值，若 TS ***Comparison Operator*** S 為 TRUE
- 具有 scalar3 或 metric3 的值，若 TS ***Comparison Operator*** S 為 FALSE

- 如果 TS **#####**為 TRUE，且量度 2 中的對應資料點不存在，則值為 0。
- 如果 TS **#####**為 FALSE，且量度 3 中的對應資料點不存在，則值為 0。
- 是一個空的時間序列，如果對應的資料點不存在於 metric3 中，或者如果運算式中省略了 scalar3/metric3

IF(metric1, metric2, metric3)

對於 metric1 的每個資料點，對應的輸出時間序列值為：

- 具有 metric2 的值，若 metric1 的對應資料點為 TRUE。
- 具有 metric3 的值，若 metric1 的對應資料點為 FALSE。
- 具有 0 的值，如果 metric1 的對應資料點為 TRUE，且 metric2中不存在對應的資料點。
- 如果量度 1 的對應資料點為 FALSE，且相應的資料點不存在於 metric3 中，則會捨棄
- 如果量度 1 的對應資料點為 FALSE，且表示式省略了 metric3，則會捨棄。
- 如果缺少量度 1 的對應資料點，則會捨棄。

下表顯示此語法的範例。

指標或函數	值
(metric1)	[1, 1, 0, 0, -]
(metric2)	[30, -, 0, 0, 30]
(metric3)	[0, 0, 20, -, 20]
IF(metric1, metric2, metric3)	[三十, 0, 二十, 零,-]

IF(metric1, scalar2, metric3)

對於 metric1 的每個資料點，對應的輸出時間序列值為：

- 具有 scalar2 的值，如果度 metric1 的對應資料點為 TRUE。
- 具有 metric3 的值，若 metric1 的對應資料點為 FALSE。
- 已捨棄，如果 metric1 的對應資料點為 FALSE，並且對應的資料點不存在於 metric3，或如果 metric3 已從運算式中省略。

指標或函數	值
(metric1)	[1, 1, 0, 0, -]
scalar2	5
(metric3)	[0, 0, 20, -, 20]
IF(metric1, scalar2, metric3)	[5, 5, 20, -, -]

IF(metric1, metric2, scalar3)

對於 metric1 的每個資料點，對應的輸出時間序列值為：

- 具有 metric2 的值，若 metric1 的對應資料點為 TRUE。
- 具有 scalar3 的值，若 metric1 的對應資料點為 FALSE。
- 具有 0 的值，如果 metric1 的對應資料點為 TRUE，且 metric2中不存在對應的資料點。
- 如果 metric1 中不存在對應資料點，則會捨棄。

指標或函數	值
(metric1)	[1, 1, 0, 0, -]
(metric2)	[30, -, 0, 0, 30]
scalar3	5
IF(metric1, metric2, scalar3)	[30, 0, 5, 5, -]

IF(scalar1, metric2, metric3)

對應的輸出時間序列值：

- 具有 metric2 的值，如果 scalar1 為 TRUE。
- 具有 metric3 的值，如果 scalar1 為 FALSE。
- 是一個空的時間序列，如果從運算式省略 metric3。

IF 表達式的使用案例範例

下列範例說明 IF 函數的可能用途。

- 若只要顯示指標的低值：

  IF(metric1<400, metric1)
- 若要將指標中的每個資料點變更為兩個值之一，以顯示原始指標的相對高低：

  IF(metric1<400, 10, 2)
- 若要針對延遲超過閾值的每個時間戳記顯示 1，並針對所有其他資料點顯示 0：

  IF(latency>threshold, 1, 0)

搭配 GetMetricData API 運算使用公制數學

您可以使用 `GetMetricData` 執行數學運算式的計算，也可以在一次的 API 呼叫中也擷取大批量的指標資料。如需詳細資訊，請參閱GetMetricData。

指標數學上的異常偵測

指標數學上的異常偵測是一項功能，您可以使用它根據指標數學表達式的單一指標和指標輸出來建立異常偵測警示。您可以使用這些表達式來建立可視化異常偵測頻段的圖形。此功能支援基本算術函數、比較和邏輯運算子，以及大多數其他函數。

指標數學上的異常偵測不支援下列函數：

- 在同一行中包含多個 ANOMALY_DETECTION_BAND 的表達式。
- 包含 10 個以上指標或數學表達式的表達式。
- 包含 METRICS (指標) 表達式的表達式。
- 包含 SEARCH (搜尋) 函數的表達式。
- 使用 DP_PERF_INSIGHTS 函數的運算式。
- 將指標與不同期間搭配使用的表達式。
- 使用高解析度量作為輸入的度量數學異常偵測器。

如需有關此功能的詳細資訊，請參閱 Amazon 使用 CloudWatch 者指南中的使用 CloudWatch 異常偵測。

在圖形中使用搜尋運算式

搜尋運算式是一種可新增至 CloudWatch 圖形的數學運算式類型。搜尋表達式可讓您快速在圖形中新增多個相關指標。它們還可讓您建立動態圖形，自動將適當的指標新增至其顯示畫面，即使這些指標在您第一次建立該圖形時並不存在。

例如，您可以建立會顯示區域中所有執行個體之 AWS/EC2 CPUUtilization 指標的搜尋表達式。如果稍後啟動新的執行個體，則新執行個體的 CPUUtilization 會自動新增到圖形中。

當您在圖形中使用搜尋表達式時，搜尋會在指標名稱、命名空間，維度名稱和維度值中尋找搜尋表達式。您可以使用布林值運算子達行更複雜強大的搜尋。搜尋運算式只能找到過去兩週內已報告資料的指標。

您無法根據 SEARCH 表達式建立警示。這是因為搜尋表達式會傳回多個時間序列，並且基於數學表達式的警示只能監看一個時間序列。

如果您在 CloudWatch 跨帳戶觀察能力下使用監視帳戶，則您的搜尋運算式可以在連結至該監視帳戶的來源帳戶中尋找指標。

主題

- CloudWatch 搜尋運算式語法
- CloudWatch 搜尋運算式範例
- 使用搜尋運算式建立 CloudWatch 圖表

CloudWatch 搜尋運算式語法

有效的搜尋表達式有以下格式。

```
SEARCH(' {Namespace, DimensionName1, DimensionName2, ...} SearchTerm', 'Statistic')
```

例如：

```
SEARCH('{AWS/EC2,InstanceId} MetricName="CPUUtilization"', 'Average')
```

- SEARCH 一詞後面以大括號括住的查詢第一部分，是要搜尋的指標結構描述。指標結構描述包含指標命名空間和一或多個維度名稱。搜尋查詢可選擇是否包含指標結構描述。如果指定，則指標結構描述必須包含命名空間，且可以選擇包含一或多個該命名空間中有效的維度名稱。

您不需要在指標結構描述內使用引號，除非命名空間或維度名稱包含空格或非英數字元。在這種情況下，您必須使用雙引號括住包含這些字元的名稱。

- SearchTerm 也是選用的，但有效的搜尋必須包含指標結構描述、SearchTerm 或兩者皆含。SearchTerm 通常包含一或多個帳戶 ID、指標名稱或維度值。SearchTerm 可以包含多個搜尋詞，部分相符和完全相符。也可以包含布林值運算子。

  在中使用帳戶 ID 僅SearchTerm適用於設定為監視帳戶以進行 CloudWatch 跨帳戶觀察性的帳戶。SearchTerm 中帳戶 ID 的語法為 :aws.AccountId = "*444455556666*"。您也可以使用 'LOCAL' 來指定監控帳戶本身：:aws.AccountId = 'LOCAL'

  如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。

  SearchTerm 可以包含一或多個指示項，例如本範例中的 MetricName=，但使用指示項非必要。

  指標結構描述和 SearchTerm 必須使用一對單引號括在一起。

- Statistic是任何有效 CloudWatch 統計資料的名稱。它必須用單引號括住。如需詳細資訊，請參閱 統計資料。

前一個範例搜尋有 InstanceId 維度名稱之任何指標的 AWS/EC2 命名空間。運算式會傳回找到的所有 CPUUtilization 指標，並以圖形顯示 Average 統計資料。

搜尋運算式只能找到過去兩週內已報告資料的指標。

搜尋表達式限制

搜尋表達式查詢大小上限為 1024 個字元。一個圖形最多可有 100 個搜尋表達式。圖形最多可以顯示 500 個時間序列。

CloudWatch 搜尋運算式:標記化

當您指定時SearchTerm，搜尋函數會搜尋 Token，這些記號是從完整度量名稱、維度名稱、維度值和命名空間 CloudWatch 自動產生的子字串。 CloudWatch 生成由原始字符串中駱駝大小寫區分的令牌。數值字元也可以做為新字符的開頭，然後使用英數字元做為分隔符號，在非英數字元的前後建立字符。

同類型字符分隔符號字元的連續字串會產生一個字符。

所有產生的字符都是小寫。下表顯示產生的一些字符範例。

原始字串	產生的字符
CustomCount1	customcount1 ，custom, count, 1
SDBFailure	sdbfailure ，sdb, failure
Project2-trial333	project2trial333 ，project, 2, trial, 333

CloudWatch 搜尋運算式:部分相符

當您指定一個時SearchTerm，也會標記化搜尋字詞。 CloudWatch根據部分相符項目來尋找量度，這些項目是從搜尋字詞產生的單一 Token 與量度名稱、命名空間、維度名稱或維度值產生的單一 Token 相符項目。

部分相符會搜尋不區分大小寫的相符單一字符。例如，使用下列任一搜尋詞會傳回 CustomCount1 指標：

- count
- Count
- COUNT

不過，使用 couNT 做為搜尋詞找不到 CustomCount1，因為搜尋詞 couNT 的大寫已字符化為 cou 和 NT。

搜尋也會比對複合字符，這是在原始名稱中連續出現的多個字符。為比對複合字符，搜尋會區分大小寫。例如，如果原始詞彙是 CustomCount1，則搜尋 CustomCount 或 Count1 會成功，但搜尋 customcount 或 count1 則否。

CloudWatch 搜尋運算式:完全相符

您可以使用雙引號括住搜尋詞需要完全相符的那部分，定義搜尋只尋找與搜尋詞完全相符的項目。這些雙引號是括在用來括住整個搜尋詞的單引號中。例如，**SEARCH(' {MyNamespace}, "CustomCount1" ', 'Maximum')** 會尋找確切的字串 CustomCount1，如果此字串在名為 MyNamespace 的命名空間中做為指標名稱、維度名稱或維度值。不過，搜尋 **SEARCH(' {MyNamespace}, "customcount1" ', 'Maximum')** 或 **SEARCH(' {MyNamespace}, "Custom" ', 'Maximum')** 找不到這個字串。

單一搜尋表達式可以結合部分相符詞和完全相符詞。例如，**SEARCH(' {AWS/NetworkELB, LoadBalancer} "ConsumedLCUs" OR flow ', 'Maximum')** 會傳回名為 ConsumedLCUs 的 Elastic Load Balancing 指標，以及所有包含字符 flow 的 Elastic Load Balancing 指標或維度。

使用完全相符也是尋找有特殊字元名稱的好方法，例如非英數字元或空格，如下列範例所示。

```
SEARCH(' {"My Namespace", "Dimension@Name"}, "Custom:Name[Special_Characters" ',
 'Maximum')
```

CloudWatch 搜尋表示式:排除測量結果綱要

到目前為止所示範的範例，都包含以大括號括住的指標結構描述。省略指標結構描述的搜尋也有效。

例如，**SEARCH(' "CPUUtilization" ', 'Average')** 會傳回與字串 CPUUtilization 完全相符的所有指標名稱、維度名稱、維度值和命名空間。在指 AWS 標命名空間中，這可能包括來自多個服務的指標，包括 Amazon EC2、Amazon ECS 和其他服務。 SageMaker

若要將此搜尋範圍縮小為只有一個 AWS 服務，最佳做法是在測量結果綱要中指定命名空間和任何必要的維度，如下列範例所示。雖然這會將搜尋縮小到 AWS/EC2 命名空間，但仍然會傳回其他指標的結果，如已將 CPUUtilization 定義為這些指標的維度值。

```
SEARCH(' {AWS/EC2, InstanceType} "CPUUtilization" ', 'Average')
```

或者，您可以在 SearchTerm 中新增命名空間，如以下範例所示。但在本範例中，搜尋會比對任何 AWS/EC2 字串，即使是自訂的維度名稱或值。

```
SEARCH(' "AWS/EC2" MetricName="CPUUtilization" ', 'Average')
```

CloudWatch 搜尋運算式:指定搜尋中的屬性名稱

以下 "CustomCount1" 的完全相符搜尋會傳回名稱完全一致的所有指標。

```
SEARCH(' "CustomCount1" ', 'Maximum')
```

但也會傳回具有 CustomCount1 維度名稱、維度值或命名空間的指標。若要進一步建構您的搜尋，您可以指定要在搜尋中尋找的物件類型屬性名稱。以下範例會搜尋所有命名空間，並傳回名為 CustomCount1 的指標。

```
SEARCH(' MetricName="CustomCount1" ', 'Maximum')
```

您也可以使用命名空間和維度名稱/值對做為屬性名稱，如下列範例所示。這些範例中的第一例，也示範了您可以使用屬性名稱加部分符合搜尋。

```
SEARCH(' InstanceType=micro ', 'Average')
```

```
SEARCH(' InstanceType="t2.micro" Namespace="AWS/EC2" ', 'Average')
```

CloudWatch 搜尋運算式:非英數字元

非英數字元做為分隔符號，並標記指標、維度、命名空間和搜尋詞等名稱要分隔到字符的位置。當詞彙字符化後，非英數字元會被去除，也不會出現在字符中。例如，Network-Errors_2 產生字符 network、errors 和 2。

您的搜尋詞可以包含任何非英數字元。如果這些字元出現在搜尋詞中，它們可以在部分相符中指定複合字符。例如，以下所有搜尋都會尋找名為 Network-Errors-2 或 NetworkErrors2 的指標。

```
network/errors
network+errors
network-errors
Network_Errors
```

當您正在執行完全相符搜尋時，完全相符搜尋中使用的任何非英數字元都必須是出現在要搜尋字串中的正確字元。例如，如果您想要尋找 Network-Errors-2，則搜尋 "Network-Errors-2" 會成功，但搜尋 "Network_Errors_2" 則否。

當您執行完全相符搜尋時，必須使用反斜線跳脫下列字元。

```
" \ ( )
```

例如，使用搜尋詞 **"Europe\\France Traffic\(Network\)"** 以完全相符尋找指標名稱 Europe \France Traffic(Network)

CloudWatch 搜尋運算式:布林運算子

搜尋支援在 SearchTerm 中使用布林值運算子 AND、OR 和 NOT。布林值運算子是括在您括住整個搜尋詞的單引號中。布林值運算子區分大小寫，因此 and、or 和 not 不是有效的布林值運算子。

您可以在搜尋中明確使用 AND，例如 **SEARCH('{AWS/EC2,InstanceId} network AND packets', 'Average')**。在搜尋詞之間不使用任何布林值運算子，即默示使用 AND 運算子搜尋這些詞，所以 **SEARCH(' {AWS/EC2,InstanceId} network packets ', 'Average')** 會獲得相同的搜尋結果。

使用 NOT 排除部分結果資料。例如，**SEARCH(' {AWS/EC2,InstanceId} MetricName="CPUUtilization" NOT i-1234567890123456 ', 'Average')** 傳回您所有執行個體的 CPUUtilization，但執行個體 i-1234567890123456 除外。您也可以使用 NOT 子句做為唯一的搜尋詞。例如，**SEARCH('NOT Namespace=AWS ', 'Maximum')** 會得到您所有的自訂指標 (指標與不包含 AWS 的命名空間)。

查詢中可以使用多個 NOT 詞。例如，**SEARCH(' {AWS/EC2,InstanceId} MetricName="CPUUtilization" NOT "ProjectA" NOT "ProjectB" ', 'Average')** 傳回區域中所有執行個體的 CPUUtilization，但維度值為 ProjectA 或 ProjectB 的執行個體除外。

您可以使用布林值運算子組合執行更強大詳細的搜尋，如以下範例所示。使用括號分組運算子。

接下來的兩個範例都會傳回 EC2 和 EBS 命名空間中所有包含 ReadOps 的指標。

```
SEARCH(' (EC2 OR EBS) AND MetricName=ReadOps ', 'Maximum')
```

```
SEARCH(' (EC2 OR EBS) MetricName=ReadOps ', 'Maximum')
```

以下範例會將前一項的搜尋縮小至只包含 ProjectA 的結果，這可能是維度值。

```
SEARCH(' (EC2 OR EBS) AND ReadOps AND ProjectA ', 'Maximum')
```

以下範例使用巢狀分組。它會傳回所有函數中 Errors 的 Lambda 指標，以及名稱中包含字串 ProjectA 或 ProjectB 之函數的 Invocations。

```
SEARCH(' {AWS/Lambda,FunctionName} MetricName="Errors" OR (MetricName="Invocations" AND
 (ProjectA OR ProjectB)) ', 'Average')
```

CloudWatch 搜尋運算式:使用數學運算式

您可以在圖形中的數學表達式中使用搜尋表達式。

例如，**SUM(SEARCH(' {AWS/Lambda, FunctionName} MetricName="Errors" ', 'Sum'))** 會傳回所有 Lambda 函數之 Errors 指標的總和。

搜尋表達式和數學表達式使用不同行可能會得到更多有用的結果。例如，假設您在圖形中使用以下兩個表達式。第一行分別顯示每個 Lambda 函數的 Errors 行。此表達式的 ID 是 e1。第二行會新增另一列顯示所有函數的錯誤總和。

```
SEARCH(' {AWS/Lambda, FunctionName}, MetricName="Errors" ', 'Sum')
SUM(e1)
```

CloudWatch 搜尋運算式範例

以下範例會示範更多的搜尋表達式用法和語法。讓我們開始在區域所有執行個體中搜尋 CPUUtilization，然後查看變化。

此範例一行顯示區域中一個執行個體，顯示 AWS/EC2 命名空間的 CPUUtilization 指標。

```
SEARCH(' {AWS/EC2,InstanceId} MetricName="CPUUtilization" ', 'Average')
```

將 InstanceId 變更成 InstanceType 會變更圖形，一行顯示區域中使用的一個執行個體類型。每種類型的所有執行個體資料都會彙總至該執行個體類型的一行。

```
SEARCH(' {AWS/EC2,InstanceType} MetricName="CPUUtilization" ', 'Average')
```

以下範例依執行個體類型彙總 CPUUtilization，一行顯示一種包括字串 micro 的執行個體類型。

```
SEARCH('{AWS/EC2,InstanceType} InstanceType=micro MetricName="CPUUtilization" ',
 'Average')
```

此範例會縮小前一個範例的結果，將 InstanceType 變更為 t2.micro 執行個體的完全相符搜尋。

```
SEARCH('{AWS/EC2,InstanceType} InstanceType="t2.micro" MetricName="CPUUtilization" ',
 'Average')
```

以下搜尋會移除查詢的 {metric schema} 部分，所以所有命名空間的 CPUUtilization 指標會出現在圖形中。這可能會傳回相當多的結果，因為圖表包含來自每個 AWS 服務之CPUUtilization量度的多行，並沿著不同的維度彙總。

```
SEARCH('MetricName="CPUUtilization" ', 'Average')
```

若要略微縮小這些結果，您可以指定兩個特定指標命名空間。

```
SEARCH('MetricName="CPUUtilization" AND ("AWS/ECS" OR "AWS/ES") ', 'Average')
```

前一個範例是使用一個搜尋查詢搜尋特定多個命名空間的唯一方式，因為每個查詢中只能指定一個指標結構描述。不過，您可以在圖形中使用兩個查詢以新增更多結構，如以下範例所示。此範例還可指定使用某個維度來彙總 Amazon ECS 資料，以新增更多結構。

```
SEARCH('{AWS/ECS ClusterName}, MetricName="CPUUtilization" ', 'Average')
SEARCH(' {AWS/EBS} MetricName="CPUUtilization" ', 'Average')
```

以下範例會傳回名為 ConsumedLCUs 的 Elastic Load Balancing 指標，以及所有包含字符 flow 的 Elastic Load Balancing 指標或維度。

```
SEARCH('{AWS/NetworkELB, LoadBalancer} "ConsumedLCUs" OR flow ', 'Maximum')
```

以下範例使用巢狀分組。它會傳回所有函數中 Errors 的 Lambda 指標，以及名稱中包含字串 ProjectA 或 ProjectB 之函數的 Invocations。

```
SEARCH('{AWS/Lambda,FunctionName} MetricName="Errors" OR (MetricName="Invocations" AND
 (ProjectA OR ProjectB)) ', 'Average')
```

以下範例顯示您所有的自訂指標，不包含 AWS 服務產生的指標。

```
SEARCH('NOT Namespace=AWS ', 'Average')
```

以下範例顯示的指標，其指標名稱、命名空間、維度名稱和維度值中包含字串 Errors 做為名稱的一部分。

```
SEARCH('Errors', 'Average')
```

以下範例會將搜尋縮小至完全相符搜尋。例如，此搜尋會尋找指標名稱 Errors，但不尋找名為 ConnectionErrors 或 errors 的指標。

```
SEARCH(' "Errors" ', 'Average')
```

以下範例示範如何指定在搜尋詞的指標結構描述部分中，包含空格或特殊字元彙的名稱。

```
SEARCH('{"Custom-Namespace", "Dimension Name With Spaces"}, ErrorCount ', 'Maximum')
```

CloudWatch 跨帳戶可觀察性搜尋運算式範例

CloudWatch 跨帳戶可觀察性範例

如果您登入的帳戶設定為 CloudWatch 跨帳戶可觀察性的監視帳戶，您可以使用 SEAR CH 函數從指定的來源帳戶傳回量度。如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。

下列範例使用帳戶 ID 111122223333 從帳戶擷取所有 Lambda 指標。

```
SEARCH(' AWS/Lambda :aws.AccountId = "111122223333" ', 'Average')
```

下列範例從兩個帳戶 111122223333 和 777788889999 擷取所有 AWS/EC2 指標。

```
SEARCH(' AWS/EC2 :aws.AccountId = ("111122223333" OR "777788889999") ', 'Average')
```

下列範例從來源帳戶 111122223333 和監控帳戶本身擷取所有 AWS/EC2 指標。

```
SEARCH(' AWS/EC2 :aws.AccountId = ("111122223333" OR 'LOCAL') ', 'Average')
```

下列範例從具有 InstanceId 維度的帳戶 444455556666 擷取 MetaDataToken 指標的 SUM。

```
SEARCH('{AWS/EC2,InstanceId} :aws.AccountId=444455556666 MetricName=\"MetadataNoToken
\"','Sum')
```

使用搜尋運算式建立 CloudWatch 圖表

在 CloudWatch 主控台上，您可以在將圖形新增至儀表板或使用「度量」檢視時存取搜尋功能。

您無法根據 SEARCH 表達式建立警示。這是因為搜尋表達式會傳回多個時間序列，並且基於數學表達式的警示只能監看一個時間序列。

使用搜尋表達式將圖形新增至現有的儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Dashboards (儀表板)，然後選取儀表板。
3. 選擇 Add widget (新增 widget)。
4. 選擇 Line (列) 或 Stacked area (堆疊區域)，然後選擇 Configure (設定)。

5. 在 Graphed metrics (圖表化指標) 標籤上，選擇 Add a math expression (新增數學表達式)。
6. 針對 Details (詳細資訊)，輸入您想要的搜尋表達式。例如 **`SEARCH('{AWS/EC2,InstanceId} MetricName="CPUUtilization"', 'Average')`**
7. (選用) 若要將另一個搜尋表達式或數學表達式新增到圖形，請選擇 Add a math expression (新增數學表達式)。
8. (選用) 在您新增搜尋表達式後，您可以指定動態標籤在每個指標的圖形圖例上顯示。動態標籤會顯示指標的統計資料，並在儀表板或圖形重新整理時自動更新。若要新增動態標籤，請選擇 Graphed metrics (圖表化指標)，然後選擇 Dynamic labels (動態標籤)。

   根據預設，您新增到標籤的動態值會出現在標籤的開頭。然後，您接著可以按一下指標的 Label (標籤) 值來編輯標籤。如需詳細資訊，請參閱 使用動態標籤。
9. (選用) 若要將單一指標新增至圖形，請選擇 All metrics (所有指標) 標籤，向下鑽研至您想要的指標。
10. (選用) 若要變更圖形上顯示的時間範圍，請選擇圖形頂端的 custom (自訂)，或 custom (自訂) 左側其中一個時段。
11. (選用) 水平註釋可協助儀表板使用者快速查看指標何時突增到特定水準，或指標是否在預先定義的範圍內。若要新增水平註釋，請選擇 Graph options (圖形選項)，和 Add horizontal annotation (新增水平註釋)：

    a. 針對 Label (標籤)，輸入註釋的標籤。
    b. 針對 Value (值)，輸入水平註釋顯示的指標值。
    c. 在 Fill (填充) 中指定此註釋是否要填充陰影。例如，選擇 `Above` 或 `Below` 作為要填入的相關區域。如果您指定 `Between`，將會顯示另一個 `Value` 欄位，這兩個值之間的圖形區域將被填充。
    d. 針對 Axis (軸)，指定 `Value` 中的數字是否要參照與左側 Y 軸或右側 Y 軸關聯的指標 (如果此圖形包含多個指標)。

       您可以透過選擇註釋左側欄中的顏色方塊，來變更註釋的填充顏色。

    重複這些步驟，將多個水平註釋新增到同一個圖形。

    若要隱藏註釋，請清除該註釋左側欄中的核取方塊。

    若要刪除註釋，請選擇 Actions (動作) 欄中的 x。
12. (選用) 垂直註釋協助您標記里程碑的圖形，例如操作事件或部署的開始與結束。若要新增垂直註釋，請選擇 Graph options (圖形選項)，然後選擇 Add vertical annotation (新增垂直註釋)：

a. 針對 Label (標籤)，輸入註釋的標籤。若要僅顯示註釋上的日期與時間，請保留 Label (標籤) 欄位留白。

b. 在 Date (日期)，指定垂直註釋出現的日期和時間位置。

c. 針對 Fill (填充)，指定是否在垂直註釋之前或之後，或是在兩個垂直註釋之間使用填充陰影。例如，選擇 `Before` 或 `After` 作為要填入的相關區域。如果您指定 `Between`，將會顯示另一個 `Date` 欄位，這兩個值之間的圖形區域將被填充。

重複這些步驟，將多個垂直註釋新增到同一個圖形。

若要隱藏註釋，請清除該註釋左側欄中的核取方塊。

若要刪除註釋，請選擇 Actions (動作) 欄中的 x。

13. 選擇 Create widget (建立 widget)。
14. 選擇 Save dashboard (儲存儀表板)。

使用 Metrics (指標) 檢視建立搜尋指標的圖形

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 指標。
3. 在搜尋欄位中，輸入要搜尋的字符：例如 **`cpuutilization t2.small`**。

   即會顯示符合您搜尋的結果。
4. 若要建立所有符合搜尋指標的圖形，請選擇 Graph search (圖形搜尋)。

   或

   若要強化搜尋，請選擇出現在搜尋結果中的其中一個命名空間。
5. 如已選取命名空間縮減搜尋結果，您可以執行以下操作：

   a. 若要將一或多個指標圖形化，請選取各個指標旁的核取方塊。若要選擇所有指標，請勾選表格標題列中的核取方塊。

   b. 若要強化搜尋，請將滑鼠游標移到指標名稱上，然後選擇 Add to search (新增至搜尋) 或 Search for this only (僅搜尋此項)。

   c. 若要查看指標的說明，請選取指標名稱並選擇 What is this? (這是什麼？)。

圖形中隨即顯示選取的指標。

6. (選用) 在搜尋列中選取其中一個按鈕，編輯搜尋詞彙的一部分。
7. (選用) 若要將圖形新增至儀表板，請選擇 Actions (動作)，然後選擇 Add to dashboard (新增至儀表板)。

取得指標的統計資訊

CloudWatch 統計定義

統計資料是隨著指定期間的指標資料彙總。繪製或擷取指標的統計資料時，請指定時間期間，例如五分鐘，以用來計算每個統計值。例如，如果期間是五分鐘，總和是五分鐘期間收集的所有樣本值的總和，而最小值是在五分鐘期間收集的最小值。

CloudWatch 支援下列測量結果統計資料。

- SampleCount是期間內資料點的數量。
- Sum (總和) 是指在該期間收集的所有資料點的值總和。
- 平均值是指定期間的 `Sum/SampleCount` 值
- 最小值是在指定期間內觀察到的最低值。
- 最大值是在指定期間內觀察到的最高值。
- 百分位數 (p) 會指出資料集中相關準備好的值。例如，p95 是第 95 個百分位數，亦表示 95% 的資料是低於此值且 5% 資料高於這個值。百分位數協助您更加了解您指標資料的分佈。
- 裁剪平均值 (TM) 是兩個指定邊界之間的所有值的平均值。計算平均值時，會忽略邊界以外的值。您可以將邊界定義為介於 0 到 100 之間的一個或兩個數字，最多可達 10 位小數位數。這些數字可以是絕對值或百分比。例如：tm90 會在移除具有最高值的 10% 資料點後計算平均值。TM(2%:98%) 會計算移除 2% 最低資料點和 2% 最高資料點後的平均值。TM(150:1000) 會在移除所有低於或等於 150 或高於 1000 的資料點後計算平均值。
- 四分位平均值 (IQM) 是四分位數範圍的裁剪平均值，或者中間 50% 的值。它等於 TM(25%:75%)。
- 縮尾均值 (WM) 類似於裁剪平均值。但是，對於縮尾均值，不會忽略邊界外的值，而是會將其視為等於適當邊界邊緣的值。在此正規化之後，即會計算平均值。您可以將邊界定義為介於 0 到 100 之間的一個或兩個數字，最多可達 10 位小數位數。例如：wm98 會計算平均值，同時將最高值的 2% 處理為等於 98 個百分位數的值。WM(10%:90%) 會計算平均值，同時將最高 10% 的資料點視為 90% 界限的值，並將最低 10% 的資料點視為 10% 界限的值。

- 百分位數排名 (PR)是符合固定閾值的百分比。例如，PR(:300) 會傳回值為 300 或以下的資料點百分比。PR(100:2000) 會傳回值介於 100 到 2000 之間的資料點百分比。

  百分位數排名在下界上是排斥的，並且在上界包容性。

- 裁剪計數 (TC) 是裁剪平均統計資料所選範圍內的資料點數目。例如，tc90 會傳回資料點數目，不包括落在最高 10% 值的任何資料點。TC(0.005:0.030) 會傳回值介於 0.005 (不含) 和 0.030 (含) 之間的資料點數目。
- 裁剪總和 (TS) 是裁剪平均統計資料所選範圍內的資料點的值總和。它相當於 (裁剪平均值) * (裁剪計數)。例如：ts90 會傳回資料點的總和，不包括落在最高 10% 值的任何資料點。TS(80%:) 會傳回資料點的值總和，不包括值範圍最低 80% 的任何資料點。

> **Note**
>
> 對於「裁剪平均值」、「裁剪計數」、「裁剪總和」和「縮尾均值」，如果您將兩個邊界定義為固定值而非百分比，則計算會包含等於較高邊界的值，但不包含等於下邊界的值。

語法

對於「裁剪平均值」、「裁剪計數」、「裁剪總和」和「縮尾均值」，套用下列語法規則：

- 使用帶有百分比符號的一個或兩個數字的括號，定義要用作資料集中位於您指定的兩個百分位數之間的值的邊界。例如，TM(10%:90%) 僅使用第 10 和第 90 個百分位數之間的值。TM(:95%) 會使用從資料設定最低端到第 95 個百分位數的值，忽略具有最高值的 5% 資料點。
- 使用含有一個或兩個數字的括號 (不含百分比符號) 可定義用作資料集中的值的邊界，且這些值位於您指定的明確值之間。例如，TC(80:500) 只會使用介於 80 (不含) 和 500 (含) 之間的值。TC(:0.5) 僅使用等於或小於 0.5 的值。
- 使用沒有括號的數字來計算使用百分比，忽略高於指定百分位數的資料點。例如，tm99 會計算平均值，同時忽略具有最高值的 1% 資料點。其與 TM(:99%) 相同。
- 裁剪平均值、裁剪計數、裁剪總和以及縮尾均值都可以在指定範圍時使用大寫字母縮寫，例如 TM(5%:95%)、TM(100:200) 或 TM(:95%)。當您只指定一個數字時，它們只能使用小寫字母縮寫，例如 tm99。

統計資料使用案例

- 裁剪均值對於具有較大範例大小的指標 (例如網頁延遲) 最有用。例如，tm99 忽略了網路問題或人為錯誤可能導致的極高異常值，以提供一般請求的平均延遲更準確的數字。同樣地，TM(10%:) 會忽略最低 10% 的延遲值，例如快取點擊所產生的延遲值。另外，TM(10%:99%) 不包含這兩種類型的異常值。建議您使用裁剪平均值來監視延遲。
- 每當您使用修剪平均值時，最好密切注意裁剪計數，以確保裁剪平均值計算中使用的值數量足以在統計上顯著。
- 百分位數排名可讓您將值放入範圍的「箱」中，而且您可以使用它來手動建立長條圖。若要執行此動作，請將值分解成各種不同的資料箱，例如 PR(:1)、PR(1:5)、PR(5:10) 和 PR(10:)。將這些資料箱中的每一個以長條圖形式放入視覺效果中，並且您有一個直方圖。

  百分位數排名在下界上是排斥的，並且在上界包容性。

百分位數與裁剪平均值

百分位數 (例如 p99) 和裁剪平均值 (例如 tm99) 測量相似，但值不相同。p99 和 tm99 均會忽略具有最高值的資料點的 1%，而這會被視為異常值。之後，p99 是剩餘 99% 的 最大值，而 tm99 是剩餘 99% 的平均值。如果您正在查看 Web 請求的延遲，p99 告訴您最糟糕的客戶體驗，忽略異常值；而 tm99 告訴您平均客戶體驗，忽略異常值。

裁剪平均值是一個不錯的延遲統計數字，如果您想要最佳化您的客戶體驗。

使用百分位數、裁剪平均值和一些其他統計數字的需求

CloudWatch 需要原始資料點來計算以下統計資料：

- 百分位數
- 裁剪平均值
- 四分位平均值
- 縮尾均值
- 裁剪的總和
- 裁剪
- 百分位數排名

如果您使用統計數字集而非原始資料的統計數字來為自訂統計數字發佈資料，只有在以下條件之一為 true 時，您才能擷取此資料的這些統計數字類型：

- 統計資料集的 SampleCount 值為 1，而最小值、最大值和總和都是相等的。
- 最小值和最大值相等，並且總和等於最小乘以 SampleCount。

下列 AWS 服務包含支援這些統計資料類型的測量結果。

- API Gateway
- Application Load Balancer
- Amazon EC2
- Elastic Load Balancing
- Kinesis
- Amazon RDS

此外，當任何指標值為負數時，這些統計數字類型不適用於指標。

下列範例說明如何取得資源 CloudWatch 指標 (例如 EC2 執行個體) 的統計資料。

範例

- 取得特定資源的統計資料
- 跨資源彙總統計資料
- 依據 Auto Scaling 群組彙總統計資料
- 以 Amazon Machine Image (AMI) 彙總統計資料

取得特定資源的統計資料

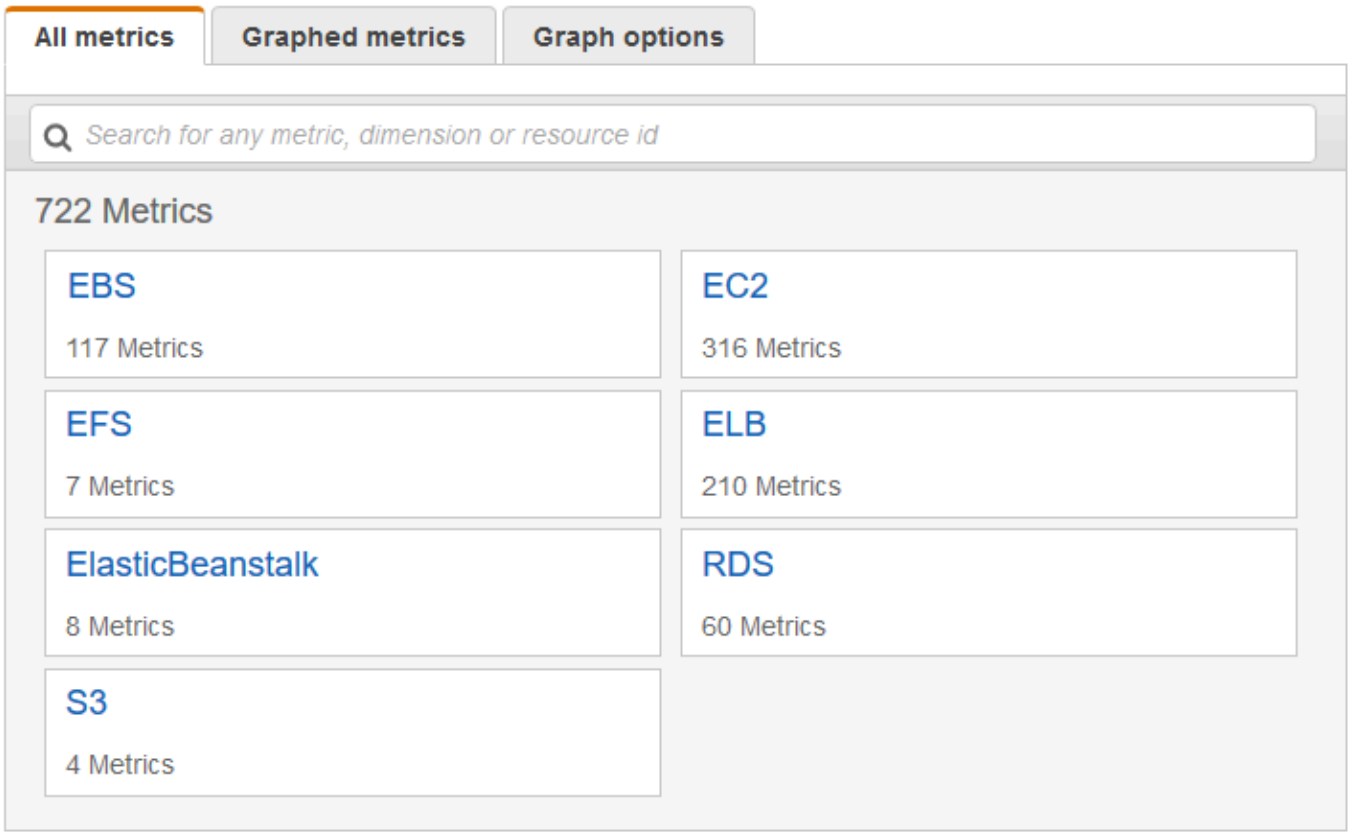
以下範例說明如何判斷特定 EC2 執行個體的最大 CPU 使用率。

要求

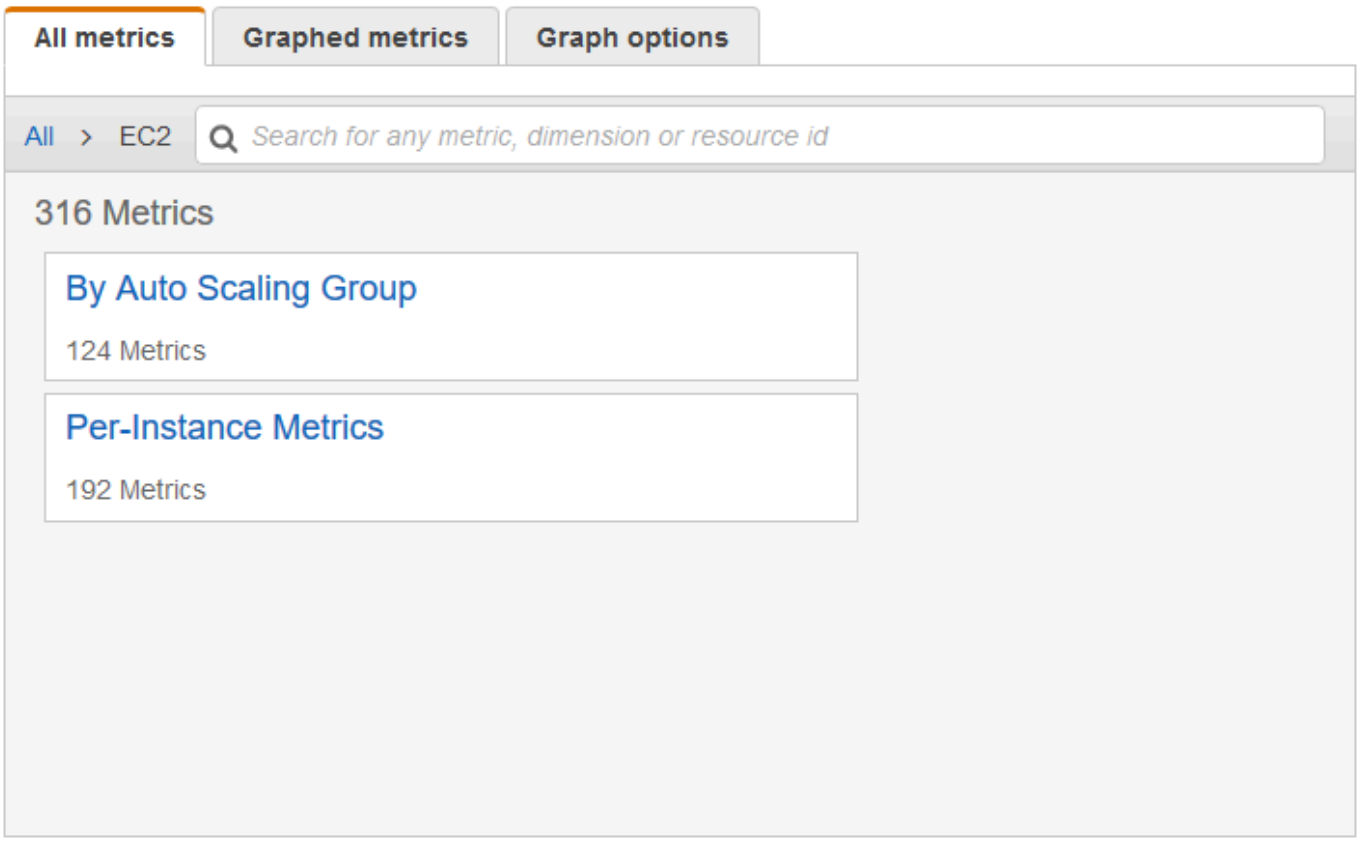
- 您必須擁有該執行個體的 ID。您可以使用 Amazon EC2 主控台或 describe-instances 命令以取得執行個體 ID。
- 預設會啟用基本監控，但您可以啟用詳細監控。如需詳細資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的啟用或停用執行個體的詳細監控。

使用主控台顯示特定執行個體的平均 CPU 使用率

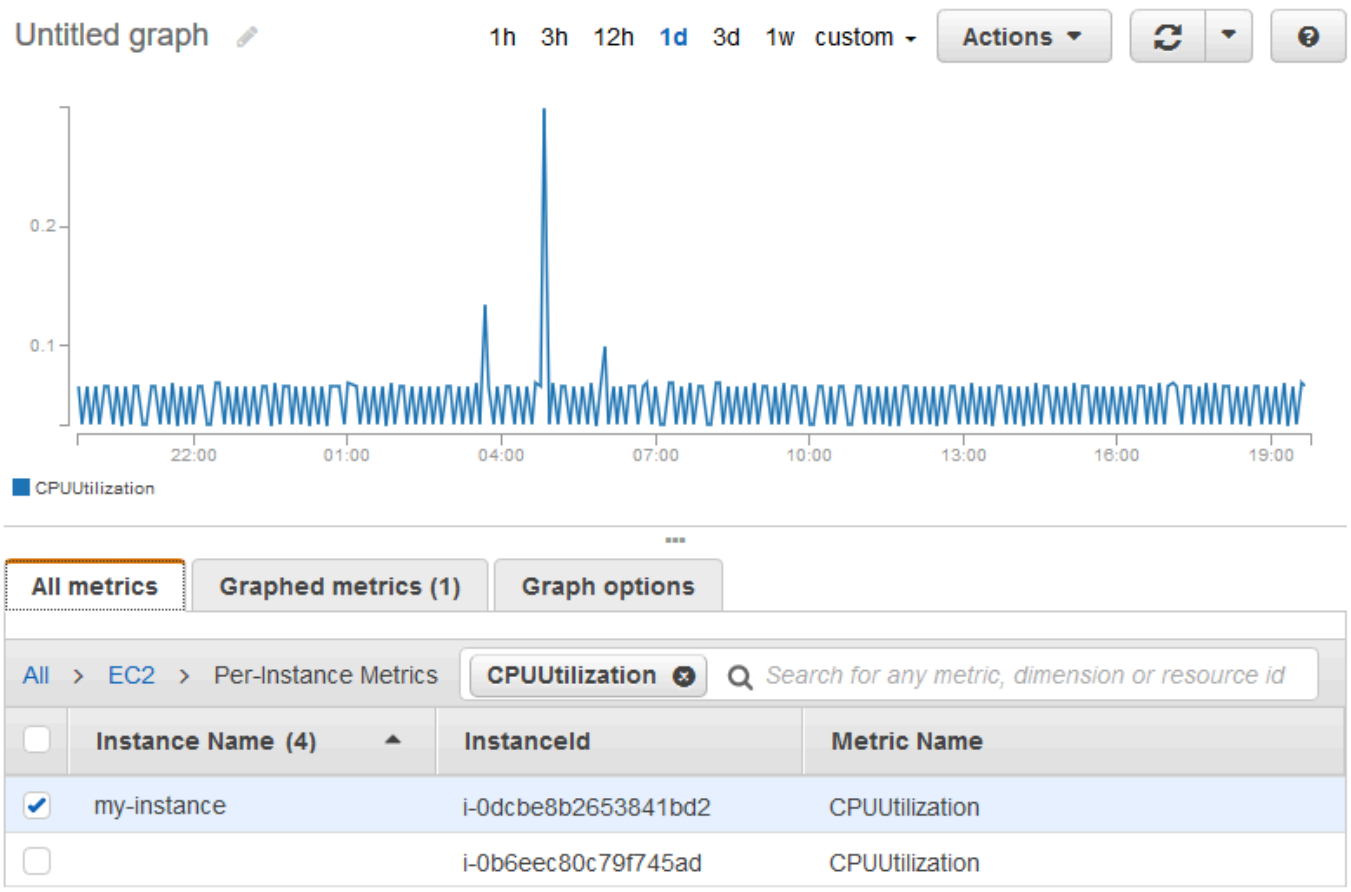
1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 指標。
3. 選取 EC2 指標命名空間。



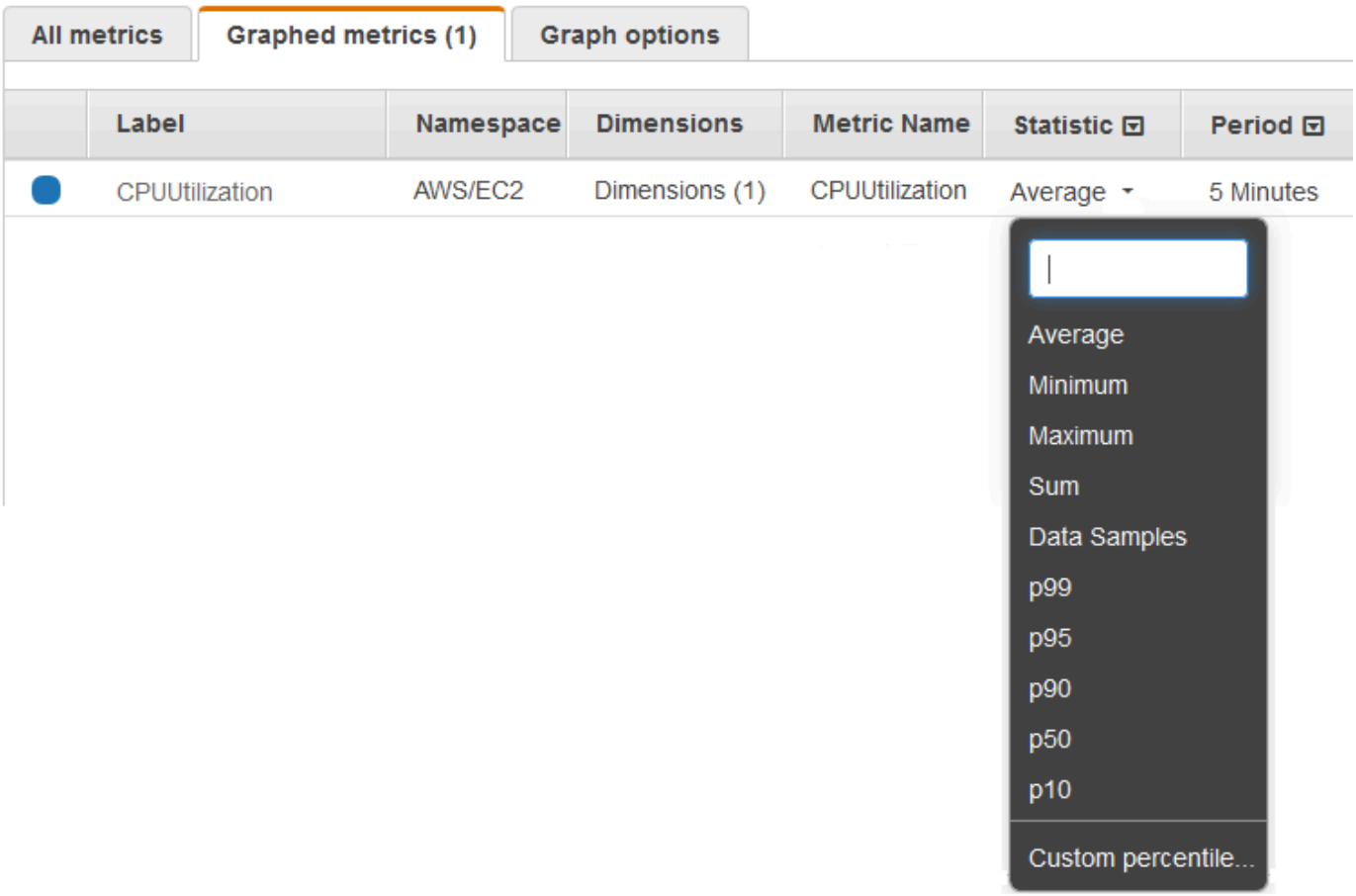
4. 選取 Per-Instance Metrics (每個執行個體指標) 維度。



5. 在搜尋欄位中，輸入 **`CPUUtilization`**，然後按 Enter 鍵。選取特定執行個體的資料列，這會顯示該執行個體 `CPUUtilization` 指標的圖形。若要變更圖形的名稱，請選擇鉛筆圖示。若要變更時間範圍，請選取一個預先定義的值，或選擇 custom (自訂)。



6. 若要變更統計數字，請選擇 Graphed metrics (圖形化指標) 索引標籤。選擇欄位標題或個別的值，然後選擇統計資料之一或預先定義的百分位，或指定自訂的百分比 (例如，**p99.999**)。



7. 若要變更期間，請選擇 Graphed metrics (圖形化指標) 標籤。選擇欄位標題或個別的值，然後選擇不同的值。

若要取得每個 EC2 執行個體的 CPU 使用率，請使用 AWS CLI

使用命get-metric-statistics令，如下所示取得CPUUtilization指定執行個體的指標。

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name CPUUtilization \
--dimensions Name=InstanceId,Value=i-1234567890abcdef0 --statistics Maximum \
--start-time 2016-10-18T23:18:00 --end-time 2016-10-19T23:18:00 --period 360
```

傳回的統計資料為請求之 24 小時時間間隔的 6 分鐘值。每個值皆代表指定執行個體在特定 6 分鐘期間的最大 CPU 使用率百分比。資料點不會依照時間順序傳回。以下顯示範例輸出的開頭 (完整輸出包含 24 小時期間之每 6 分鐘的資料點)。

```
{
    "Datapoints": [
        {
```

```
            "Timestamp": "2016-10-19T00:18:00Z",
            "Maximum": 0.33000000000000002,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2016-10-19T03:18:00Z",
            "Maximum": 99.670000000000002,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2016-10-19T07:18:00Z",
            "Maximum": 0.34000000000000002,
            "Unit": "Percent"
        },
        ...
    ],
    "Label": "CPUUtilization"
}
```

跨資源彙總統計資料

您可以跨多個資源彙總 AWS 資源的指標。指標在區域之間完全分開，但您可以使用指標數學來彙總跨區域的類似指標。如需詳細資訊，請參閱 使用指標數學。

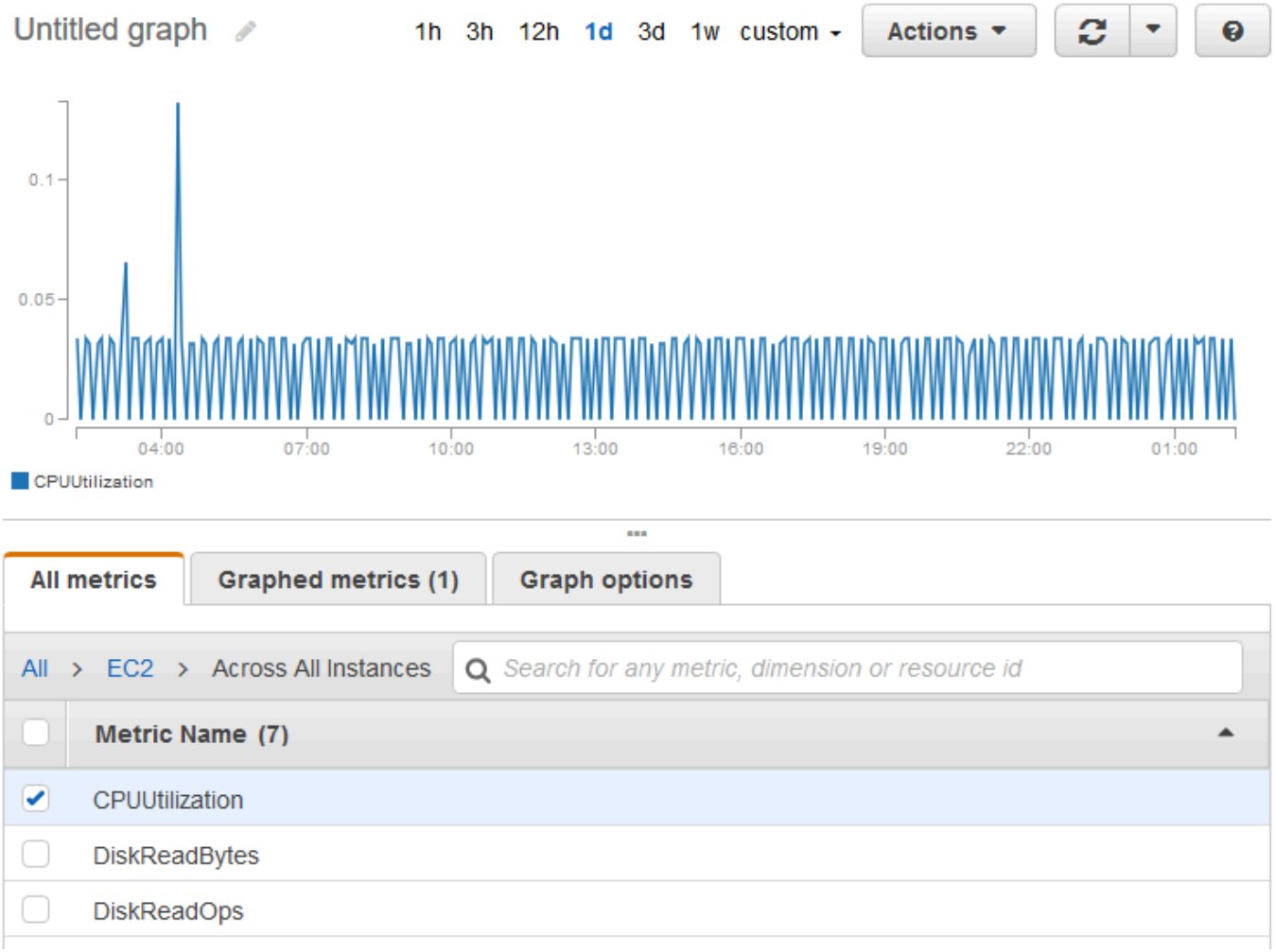
例如，您可以為您已啟用詳細監控的 EC2 執行個體彙總統計資料。使用基本監控的執行個體不包含在內。因此，您必須啟用詳細監控 (額外付費)，它提供 1 分鐘期間的資料。如需詳細資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的啟用或停用執行個體的詳細監控。

此範例顯示如何取得您的 EC2 執行個體的平均 CPU 使用量。因為未指定維度，所以會 CloudWatch 傳回AWS/EC2命名空間中所有維度的統計資料。若要取得其他指標的統計資料，請參閱 AWS 發佈指 CloudWatch標的服務。

> **Important**
>
> 這種跨 AWS 命名空間擷取所有維度的技巧不適用於您發佈到 CloudWatch的自訂命名空間。使用自訂命名空間，您必須指定與任何特定資料點建立關聯的一組完整維度，以擷取包含該資料點的統計資料。

顯示您的 EC2 執行個體的平均 CPU 使用率

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 指標。
3. 選擇 EC2 命名空間，然後選擇 Across All Instances (在所有執行個體中)。
4. 選取包含 `CPUUtilization` 的資料列，這會顯示您所有 EC2 執行個體指標的圖形。若要變更圖形的名稱，請選擇鉛筆圖示。若要變更時間範圍，請選取一個預先定義的值，或選擇 custom (自訂)。



5. 若要變更統計數字，請選擇 Graphed metrics (圖形化指標) 索引標籤。選擇欄位標題或個別的值，然後選擇統計資料之一或預先定義的百分位，或指定自訂的百分比 (例如，**`p95.45`**)。
6. 若要變更期間，請選擇 Graphed metrics (圖表化指標) 標籤。選擇欄位標題或個別的值，然後選擇不同的值。

若要取得整個 EC2 執行個體的平均 CPU 使用率，請使用 AWS CLI

使用 get-metric-statistics 命令，如下所示：

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name CPUUtilization
 --statistics "Average" "SampleCount" \
--start-time 2016-10-11T23:18:00 --end-time 2016-10-12T23:18:00 --period 3600
```

下列為範例輸出：

```
{
    "Datapoints": [
        {
            "SampleCount": 238.0,
            "Timestamp": "2016-10-12T07:18:00Z",
            "Average": 0.038235294117647062,
            "Unit": "Percent"
        },
        {
            "SampleCount": 240.0,
            "Timestamp": "2016-10-12T09:18:00Z",
            "Average": 0.16670833333333332,
            "Unit": "Percent"
        },
        {
            "SampleCount": 238.0,
            "Timestamp": "2016-10-11T23:18:00Z",
            "Average": 0.041596638655462197,
            "Unit": "Percent"
        },
        ...
    ],
    "Label": "CPUUtilization"
}
```

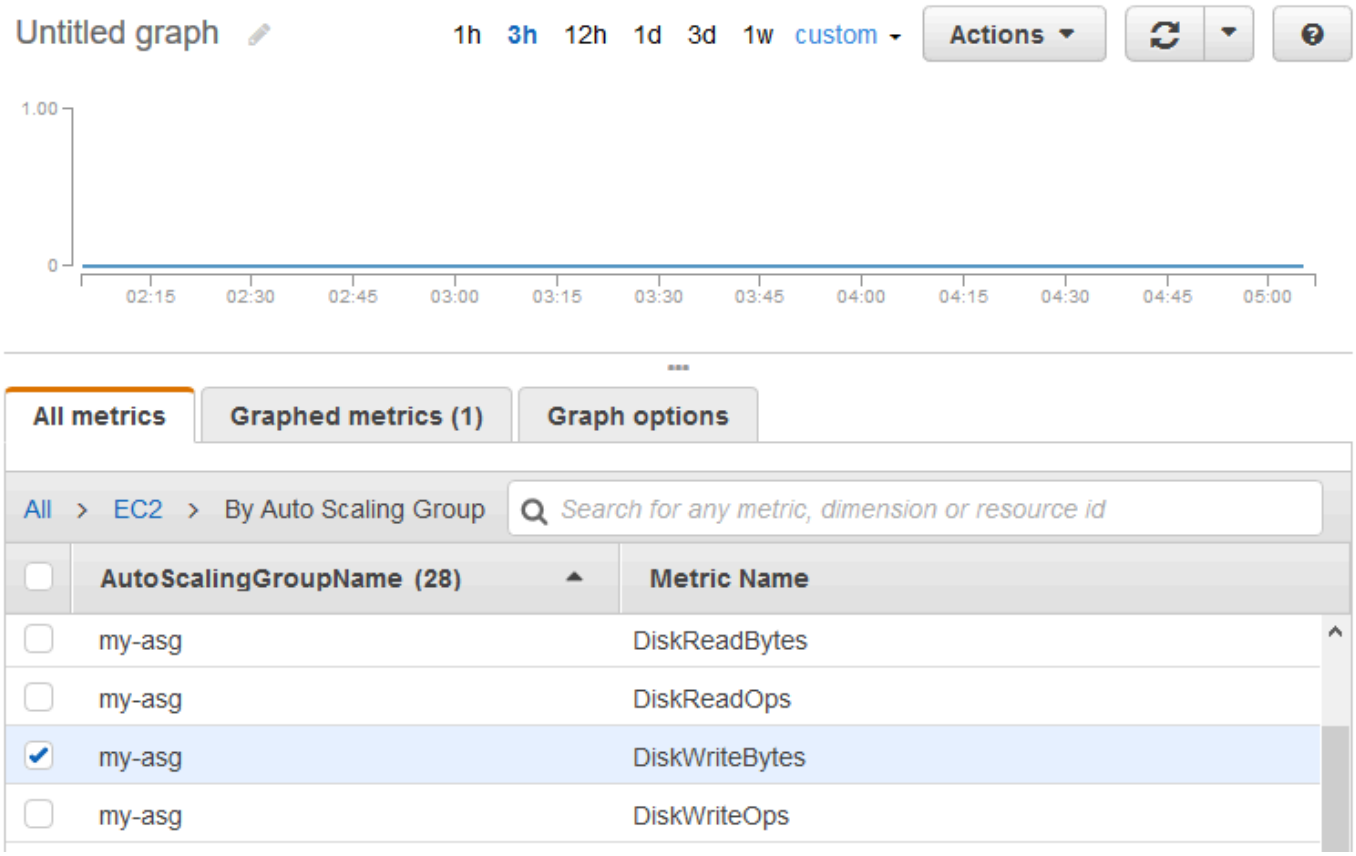
依據 Auto Scaling 群組彙總統計資料

您可以彙總 Auto Scaling 群組中 EC2 執行個體的統計資料。量度在區域之間完全分開，但您可以使用指 CloudWatch 標數學來彙總和轉換來自多個區域的量度。您也可以使用跨帳戶儀表板，針對不同帳戶的指標執行指標數學。

此範例顯示如何取得一個 Auto Scaling 群組寫入磁碟的總位元組。總計是計算指定 Auto Scaling 群組中的所有 EC2 執行個體，以 24 小時為間隔取得每隔 1 分鐘的數量。

使用主控台顯示 DiskWriteBytes Auto Scaling 群組中的執行個體

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 指標。
3. 選擇 EC2 命名空間，然後選擇 By Auto Scaling Group (依據 Auto Scaling 群組)。
4. 選取DiskWriteBytes量度和特定「Auto Scaling 例」群組的列，此群組會顯示「Auto Scaling 比例」群組中執行個體的量度圖表。若要變更圖形的名稱，請選擇鉛筆圖示。若要變更時間範圍，請選取一個預先定義的值，或選擇 custom (自訂)。



5. 若要變更統計數字，請選擇 Graphed metrics (圖形化指標) 索引標籤。選擇欄位標題或個別的值，然後選擇統計資料之一或預先定義的百分位，或指定自訂的百分比 (例如，**p95.45**)。
6. 若要變更期間，請選擇 Graphed metrics (圖表化指標) 標籤。選擇欄位標題或個別的值，然後選擇不同的值。

若要取得「Auto Scaling」群組中的執行個體， DiskWriteBytes 請使用 AWS CLI

使用 get-metric-statistics 命令，如下所示。

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name DiskWriteBytes
```

```
--dimensions Name=AutoScalingGroupName,Value=my-asg --statistics "Sum" "SampleCount" \
--start-time 2016-10-16T23:18:00 --end-time 2016-10-18T23:18:00 --period 360
```

下列為範例輸出。

```
{
    "Datapoints": [
        {
            "SampleCount": 18.0,
            "Timestamp": "2016-10-19T21:36:00Z",
            "Sum": 0.0,
            "Unit": "Bytes"
        },
        {
            "SampleCount": 5.0,
            "Timestamp": "2016-10-19T21:42:00Z",
            "Sum": 0.0,
            "Unit": "Bytes"
        }
    ],
    "Label": "DiskWriteBytes"
}
```

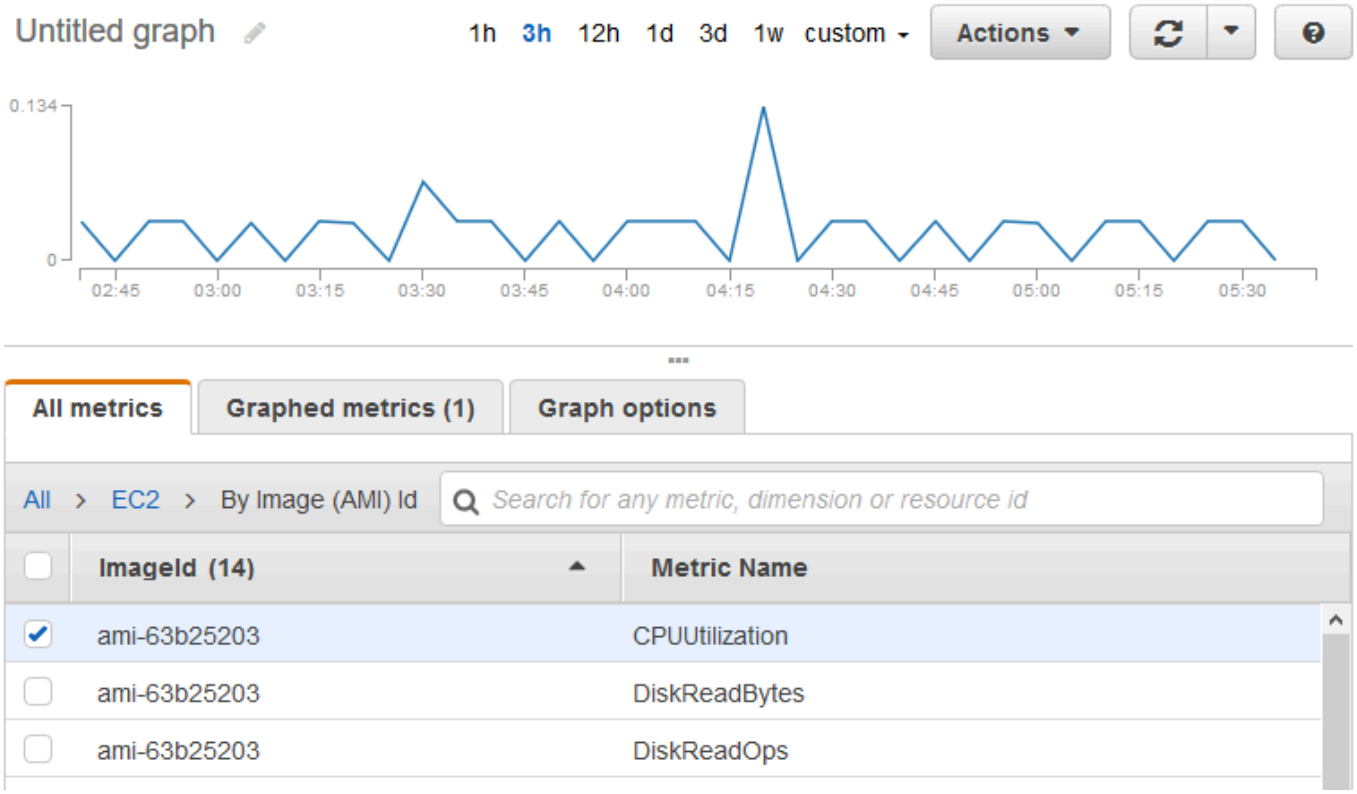
以 Amazon Machine Image (AMI) 彙總統計資料

您可以為已啟用詳細監控的 EC2 執行個體彙總統計資料。使用基本監控的執行個體不包含在內。如需詳細資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的啟用或停用執行個體的詳細監控。

此範例顯示如何判斷使用指定的 AMI 的所有執行個體的平均 CPU 使用率。此平均是以一天期間內每 60 秒的時間間隔計算。

使用主控台顯示依據 AMI 的平均 CPU 使用率

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 指標。
3. 選擇 EC2 命名空間，然後選擇 By Image (AMI) Id (依據映像 (AMI) ID)。
4. 選取 CPUUtilization 指標與特定 AMI 的資料列，這會顯示指定 AMI 的指標圖形。若要變更圖形的名稱，請選擇鉛筆圖示。若要變更時間範圍，請選取一個預先定義的值，或選擇 custom (自訂)。



5. 若要變更統計數字，請選擇 Graphed metrics (圖表化指標) 索引標籤。選擇欄位標題或個別的值，然後選擇統計資料之一或預先定義的百分位，或指定自訂的百分比 (例如，**p95.45**)。
6. 若要變更期間，請選擇 Graphed metrics (圖表化指標) 標籤。選擇欄位標題或個別的值，然後選擇不同的值。

若要取得 AMI 的平均 CPU 使用率，請使用 AWS CLI

使用 get-metric-statistics 命令，如下所示。

```
aws cloudwatch get-metric-statistics --namespace AWS/EC2 --metric-name CPUUtilization \
--dimensions Name=ImageId,Value=ami-3c47a355 --statistics Average \
--start-time 2016-10-10T00:00:00 --end-time 2016-10-11T00:00:00 --period 3600
```

此操作傳回的統計資料是以一天為間隔的每個小時值。每個值代表 EC2 執行個體執行指定的 AMI 的平均 CPU 使用率百分比。下列為範例輸出。

```
{
    "Datapoints": [
        {
            "Timestamp": "2016-10-10T07:00:00Z",
            "Average": 0.041000000000000009,
```

```
            "Unit": "Percent"
        },
        {
            "Timestamp": "2016-10-10T14:00:00Z",
            "Average": 0.079579831932773085,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2016-10-10T06:00:00Z",
            "Average": 0.036000000000000011,
            "Unit": "Percent"
        },
        ...
    ],
    "Label": "CPUUtilization"
}
```

發佈 自訂指標

您可以 CloudWatch 使用 AWS CLI 或 API 將自己的指標發佈到。您可以使用檢視已發佈量度的統計圖表 AWS Management Console。

CloudWatch 將有關量度的資料儲存為一系列資料點。每個資料點都有一個關聯的時間戳記。您甚至可以發佈彙總的一組資料點，稱為 statistic set (統計資料集)。

主題

- 高解析度指標
- 使用維度
- 發佈單一資料點
- 發佈統計資料集
- 發佈零值
- 停止發佈指標

高解析度指標

每個指標皆為下列其中一種：

- 標準解析度，包含 1 分鐘精細度的資料

- 高解析度，包含 1 秒鐘精細度的資料

依預設， AWS 服務產生的度量為標準解析度。當您發佈自訂指標時，可將它定義為標準解析度或高解析度。當您發佈高解析度量時，請以 1 秒的解析度 CloudWatch 儲存該指標，而且您可以讀取和擷取它，時間為 1 秒、5 秒、10 秒、30 秒或 60 秒的任意倍數。

高解析度的指標可讓您以高於分鐘層級的精細度，更即時地深入了解您應用程式的活動。請注意，對自訂指標進行的每個 PutMetricData 呼叫皆需付費，因此經常對高解析度指標呼叫 PutMetricData 將導致更高的費用。如需有關 CloudWatch 定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

如果您在高解析度指標設定警示，您可以指定期間為 10 秒或 30 秒的高解析度警示，或者設定期間為 60 秒的任何倍數的定期警示。期間為 10 或 30 秒的高解析度警示將產生更高的費用。

使用維度

在自訂指標中，--dimensions 參數是很常見的。維度可更進一步釐清指標內容及其存放的資料。指派給一個指標的維度最多可有 30 個，每個維度皆以名值對定義。

當您使用不同的命令時，指定維度的方式也不同。使用時 put-metric-data，您可以將每個標註指定為 *MyName*= *MyValue*，並put-metric-alarm使用get-metric-statistics或使用格式 Name= *MyName*，Value=*MyValue*。例如，以下命令會發佈包含 InstanceId 和 InstanceType 兩個維度的 Buffers 指標。

```
aws cloudwatch put-metric-data --metric-name Buffers --namespace MyNameSpace --unit
 Bytes --value 231434333 --dimensions InstanceId=1-23456789,InstanceType=m1.small
```

此命令擷取相同指標的統計資料。以逗號分隔單一維度的名稱和值部分，但如果您有多個維度，請維度與維度之間使用空格。

```
aws cloudwatch get-metric-statistics --metric-name Buffers --namespace MyNameSpace --
dimensions Name=InstanceId,Value=1-23456789 Name=InstanceType,Value=m1.small --start-
time 2016-10-15T04:00:00Z --end-time 2016-10-19T07:00:00Z --statistics Average --period
 60
```

如果單一量度包含多個維度，則必須在使用時為每個已定義的維度指定一個值get-metric-statistics。例如，Amazon S3 指標BucketSizeBytes包含維度BucketName和StorageType，因此您必須使用 get-metric-statistics.

```
aws cloudwatch get-metric-statistics --metric-name BucketSizeBytes --start-time
 2017-01-23T14:23:00Z --end-time 2017-01-26T19:30:00Z --period 3600 --namespace
```

```
AWS/S3 --statistics Maximum --dimensions Name=BucketName,Value=MyBucketName
Name=StorageType,Value=StandardStorage --output table
```

請使用 list-metrics 命令查看針對指標所定義的維度。

發佈單一資料點

若要為新的或現有的量度發佈單一資料點，請使用具有一個值和時間戳記的put-metric-data指令。例如，以下每個動作都會發佈一個資料點。

```
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService --
value 2 --timestamp 2016-10-20T12:00:00.000Z
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService --
value 4 --timestamp 2016-10-20T12:00:01.000Z
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService --
value 5 --timestamp 2016-10-20T12:00:02.000Z
```

如果您使用新的量度名稱呼叫此指令， CloudWatch 會為您建立量度。否則，請 CloudWatch 將您的資料與您指定的現有量度產生關聯。

> **Note**
>
> 建立測量結果時，最多可能需要 2 分鐘才能使用get-metric-statistics命令擷取新測量結果的統計資料。不過，使用 list-metrics 命令擷取的指標清單可能需要長達 15 分鐘，才會顯示新的指標。

雖然您可以將時間戳記的資料點發佈為千分之一秒的精細度，但是會將資料 CloudWatch 彙總為 1 秒的最小粒度。 CloudWatch 記錄每個週期收到的值的平均值 (所有項目的總和除以項目數目)，以及同一時間週期的樣本數目、最大值和最小值。例如，之前範例的 PageViewCount 指標包含三個資料點，其時間戳記僅相差幾秒。如果您的期間設定為 1 分鐘，則會 CloudWatch 彙總這三個資料點，因為這三個資料點都在 1 分鐘內有時間戳記。

您可以使用 get-metric-statistics 命令，根據您發佈的資料點來擷取統計資料。

```
aws cloudwatch get-metric-statistics --namespace MyService --metric-name PageViewCount
 \
--statistics "Sum" "Maximum" "Minimum" "Average" "SampleCount" \
--start-time 2016-10-20T12:00:00.000Z --end-time 2016-10-20T12:05:00.000Z --period 60
```

下列為範例輸出。

```
{
    "Datapoints": [
        {
            "SampleCount": 3.0,
            "Timestamp": "2016-10-20T12:00:00Z",
            "Average": 3.6666666666666665,
            "Maximum": 5.0,
            "Minimum": 2.0,
            "Sum": 11.0,
            "Unit": "None"
        }
    ],
    "Label": "PageViewCount"
}
```

發佈統計資料集

您可以在發佈到之前彙總資料 CloudWatch。如果您在每分鐘有多個資料點，彙總資料可大幅減少呼叫 put-metric-data 的次數。例如，您可以使用 `--statistic-values` 參數，將資料彙總到統計資料集，並以一次呼叫發佈此統計資料集，而非針對 3 秒內發生的三個資料點多次呼叫 put-metric-data。

```
aws cloudwatch put-metric-data --metric-name PageViewCount --namespace MyService
 --statistic-values Sum=11,Minimum=2,Maximum=5,SampleCount=3 --
timestamp 2016-10-14T12:00:00.000Z
```

CloudWatch 需要原始資料點來計算百分位數。如果已用統計資料集發佈資料，除非以下條件之一為 true，否則您就無法擷取此資料的百分位統計資料：

- 統計資料集的 `SampleCount` 設定為 1。
- 統計資料集的 `Minimum` 和 `Maximum` 相等。

發佈零值

當您的資料更為零散，而且您有一些期間沒有關聯的資料，那麼您可以選擇為該期間發佈零值 (0) 或無值。如果您使用定期呼叫來 `PutMetricData` 以監控應用程式的狀態，您可能會想要發佈零而非無值。例如，您可以設定 CloudWatch 警示，以便在應用程式無法每五分鐘發佈指標時通知您。您希望這類應用程式在無需關聯資料的期間發佈零。

如果您想追蹤資料點的總數，或希望統計資料如最低值和平均值包含 0 值的資料點，那麼您也會希望發佈零。

停止發佈指標

若要停止發佈自訂指標 CloudWatch，請變更應用程式或服務的程式碼以停止使用PutMetricData。CloudWatch 不會從應用程式提取指標，它只會接收推送到應用程式的指標，因此若要停止發佈指標，您必須在來源處停止指標。

使用 Amazon CloudWatch 警報

您可以在 Amazon 中創建指標和複合警報 CloudWatch。

- 公制警示會根據度 CloudWatch 量監視單一度量或數學運算式的 CloudWatch 結果。警示會根據在數個期間與閾值相關的指標值或表達式值來執行一或多個動作。動作可以是傳送通知給 Amazon SNS 主題、執行 Amazon EC2 動作或 Amazon EC2 自動 Auto Scaling 作，或在 Systems Manager 中建立 OpsItem 或事件。
- 複合警示包括一個規則表達式，該表達式會考慮您所建立之其他警示的警示狀態。僅在符合規則的所有條件時，複合警示才會進入 ALARM 狀態。複合警示規則表達式中指定的警示可以包括指標警示和其他複合警示。

  使用複合警示可以減少警示噪音。您可以建立多個指標警示，也可以建立一個複合警示，並僅針對複合警示設定提醒。例如，僅在所有的基礎指標警示都處於 ALARM 狀態時，複合警示才可能進入 ALARM 狀態。

  複合警示可在狀態變更時傳送 Amazon SNS 通知，並可在進入警示狀態時建立 Systems Manager OpsItems 或事件，但無法執行 EC2 動作或 Auto Scaling 動作。

> **Note**
>
> 您可以在 AWS 帳戶中創建任意數量的警報。

您可以將警示新增至儀表板，以便在多個地區監視和接收有關資 AWS 源和應用程式的警示。將警示新增至儀表板後，當警示處於 `INSUFFICIENT_DATA` 狀態時會變成灰色，當處於 `ALARM` 狀態時則會變成紅色。處於 `OK` 狀態的所示警示沒有顏色。

您也可以從 CloudWatch 主控台導覽窗格中的 [我的最愛和最近] 選項，將最近造訪過的鬧鐘加入最愛。Favorites and recents (我的最愛和最近的項目) 選項具備我的最愛警示和最近造訪過的警示資料欄。

只有在警示變更狀態時，警示才會叫用動作。此例外狀況適用於針對 Auto Scaling 動作的警示。對於 Auto Scaling 動作，警示維持在新狀態的每一分鐘，警示會持續叫用動作。

鬧鐘可以監看相同帳戶中的指標。如果您已在 CloudWatch 主機中啟用跨帳戶功能，您也可以建立警示來監視其他 AWS 帳戶中的指標。不支援建立跨帳戶複合警示。支援建立使用數學表達式的跨帳

戶警示，除了這個之外，跨帳戶警示不支援 ANOMALY_DETECTION_BAND、INSIGHT_RULE，以及 SERVICE_QUOTA 函數。

> **Note**
>
> CloudWatch 不會測試或驗證您指定的動作，也不會偵測因嘗試叫用不存在的動作而產生的任何 Amazon EC2 Auto Scaling 或 Amazon SNS 錯誤。請確定您的警示動作存在。

指標警示狀態

警示擁有以下可能的狀態：

- OK – 指標或表達式在定義的閾值內。
- ALARM – 指標或表達式在定義的閾值外。
- INSUFFICIENT_DATA – 警示剛開始無法使用指標，或資料不足無法讓指標判斷警示狀態。

評估警示

建立鬧鐘時，您可以指定三個設定來啟用， CloudWatch 以評估何時變更警示狀態：

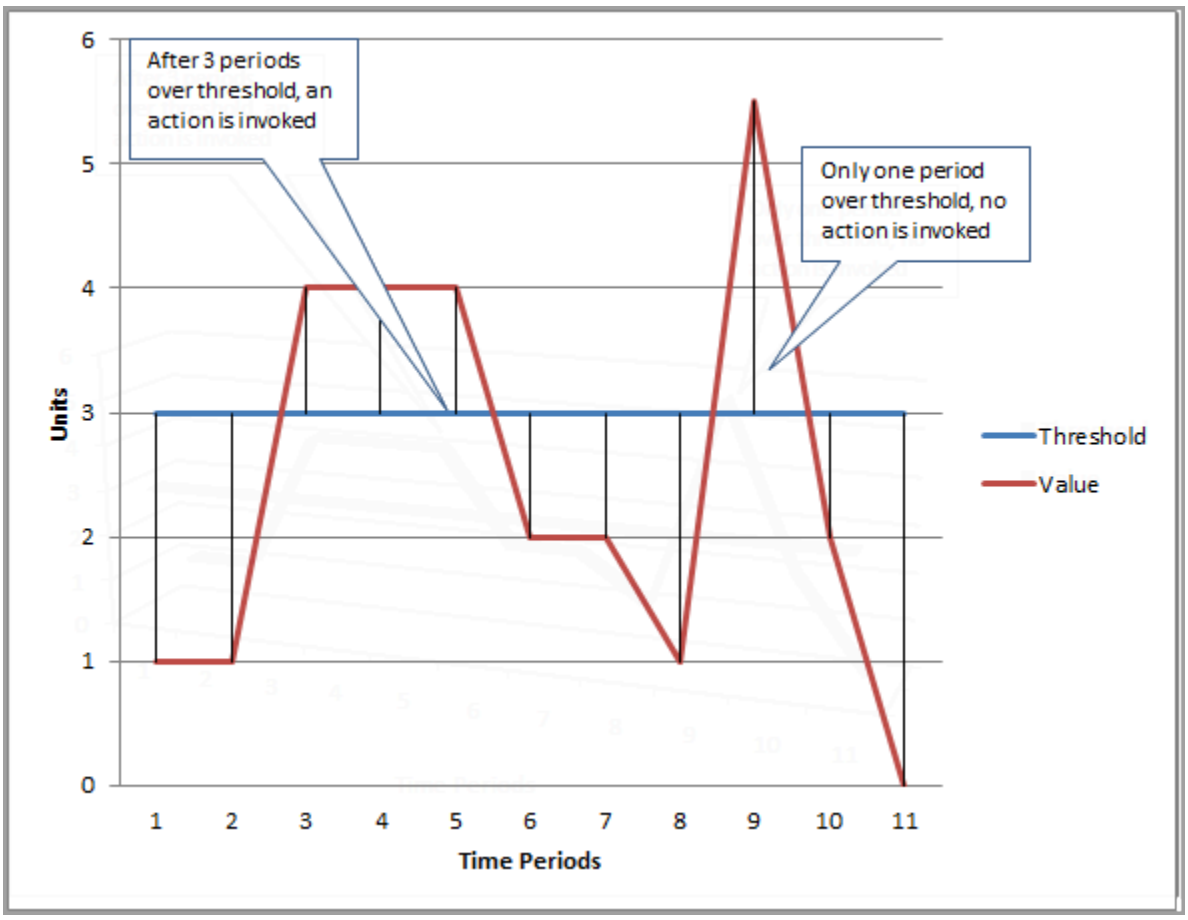
- 期間是為了建立警示的每個資料點，用來評估指標或表達式的時間長度。它會以秒表示。
- Evaluation Periods (評估期間) 是判斷警示狀態時所評估的最近期間數或資料點數目。
- Datapoints to Alarm (要警示的資料點) 是在評估期間，必須違規以導致警示進入 ALARM 狀態的資料點數目。違規的資料點不必連續，但是其必須位於等於 Evaluation Period (評估期間) 的資料點最後數字的範圍內。

對於任何一分鐘或更長的期間，系統會每分鐘評估一次警示，且此評估是以期間和評估期間所定義的時間範圍為依據。舉例來說，如果期間為 5 分鐘 (300 秒) 且評估期間的數量為 1，則在第 5 分鐘結束時，警示將會根據第 1 到第 5 分鐘的資料進行評估。接著，在第 6 分鐘結束時，系統會根據第 2 到第 6 分鐘的資料評估警示。

如果警示期間為 10 秒或 30 秒，則系統會每 10 秒評估一次警示。

下圖中，指標警示的警示閾值已設為三個單位。Evaluation Period (評估期間) 和 Datapoints to Alarm (要警示的資料點) 均為 3。也就是說，當最新三個連續期間中的所有現有資料點都超過閾值時，警示便會移至 ALARM 狀態。在此圖中，這會在第三到五個時段發生。在期間六，值 dips 低於閾值，因此評估

其中一個期間不違反，且警示狀態變更為 OK。在第九個時段，再次達到閾值，但只針對一個期間。因此，該警示狀態會維持 OK。



當您將 Evaluation Periods (評估期間) 和 Datapoints to Alarm (要警示的資料點) 設為不同值時，您會設定「N 個項目中有 M 個」警示。Datapoints to Alarm (要警示的資料點) 是 ("M")，Evaluation Periods (評估期間) 是 ("N")。評估間隔是評估時段數乘以時段長度。例如，若您設定 1 分鐘內 5 個資料點中有 4 個，則評估間隔為 5 分鐘。若您設定 10 分鐘內 3 個資料點中有 3 個，則評估間隔為 30 分鐘。

> **Note**
>
> 如果在您建立警示後不久遺失資料點，而且在您建立警示 CloudWatch 之前已報告指標，則會在評估警示時從建立警示之前 CloudWatch擷取最新的資料點。

警示動作

您可以在警示於 OK、ALARM 和 INSUFFICIENT_DATA 狀態之間變更時，指定其要採取的動作。

您可以針對上述三種狀態的轉換設定大多數動作。「自動擴展」動作除外，這些動作只有在狀態轉換時才會發生，而且如果條件持續數小時或數天，也不會再次執行。您可以利用警示允許多個動作的事實，在超出閾值時傳送電子郵件，然後在違規條件結束時傳送另一封電子郵件。這可協助您驗證擴展或復原動作是否會在預期時觸發，並且如所需正常運作。

以下是支援的警示動作。

- 透過使用 Amazon Simple Notification Service 主題，通知一個或多個訂閱用戶。訂閱用戶可以是應用程式以及個人。如需 Amazon SNS 的詳細資訊，請參閱「什麼是 Amazon SNS？」。
- 調用 Lambda 函數。這是您在警示狀態變更時自動執行自訂動作的最簡單方法。
- 以 EC2 指標為基礎的警示也可以執行 EC2 動作，例如停止、終止、重新啟動或復原 EC2 執行個體。如需詳細資訊，請參閱 建立警示以停止、終止、重新啟動或復原 EC2 執行個體。
- 警示也可以執行動作來擴展 Auto Scaling 群組。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 的步進和簡易擴展政策。
- 警示可以 OpsItems 在 Systems Manager 作業中心建立，或在 AWS 系統管理員事件管理員中建立事件。這些動作只有在警示進入 ALARM 狀態時才會執行。如需詳細資訊，請參閱設 CloudWatch 定為 OpsItems 從警示建立和事件建立。

Lambda 警示動作

CloudWatch 警報可保證指定狀態變更的 Lambda 函數非同步叫用，但下列情況除外：

- 當函數不存在時。
- 如果 CloudWatch 沒有授權調用 Lambda 函數。

如果 CloudWatch 無法連線至 Lambda 服務，或是因為其他原因而拒絕訊息，請 CloudWatch 重試直到呼叫成功為止。Lambda 會將訊息排入佇列，並處理執行重試。如需有關此執行模型的詳細資訊，包括 Lambda 如何處理錯誤的資訊，請參閱 AWS Lambda 開發人員指南中的非同步叫用。

您可以在相同帳戶或其他 AWS 帳戶中叫用 Lambda 函數。

當您指定警示來調用 Lambda 函數作為警示動作時，您可以選擇指定函數名稱、函數別名或函數的特定版本。

將 Lambda 函數指定為警示動作時，必須為函數建立資源原則，以允許 CloudWatch 服務主體叫用函數。

其中一種方法是使用 AWS CLI, 如下列範例所示：

```
aws lambda add-permission \
--function-name my-function-name \
--statement-id AlarmAction \
--action 'lambda:InvokeFunction' \
--principal lambda.alarms.cloudwatch.amazonaws.com \
--source-account 111122223333 \
--source-arn arn:aws:cloudwatch:us-east-1:111122223333:alarm:alarm-name
```

或者，您可以建立類似下列其中一個範例的政策，然後將其指派給函數。

下列範例會指定警示所在的帳戶，因此只有該帳戶 (111122223333) 中的警示才能調用函數。

```
{
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [{
        "Sid": "AlarmAction",
        "Effect": "Allow",
        "Principal": {
            "Service": "lambda.alarms.cloudwatch.amazonaws.com"
        },
        "Action": "lambda:InvokeFunction",
        "Resource": "arn:aws:lambda:us-east-1:444455556666:function:function-name",
        "Condition": {
            "StringEquals": {
                "AWS:SourceAccount": "111122223333"
            }
        }
    }]
}
```

下列範例的範圍較窄，只允許指定帳戶中的指定警示調用函數。

```
{
  "Version": "2012-10-17",
  "Id": "default",
  "Statement": [
    {
      "Sid": "AlarmAction",
      "Effect": "Allow",
      "Principal": {
```

```
        "Service": "lambda.alarms.cloudwatch.amazonaws.com"
      },
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-east-1:444455556666:function:function-name",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "111122223333",
          "AWS:SourceArn": "arn:aws:cloudwatch:us-east-1:111122223333:alarm:alarm-name"
        }
      }
    }]
}
```

不建議建立不指定來源帳戶的政策，因為這類政策容易出現混淆的代理問題。

從傳送 CloudWatch 至 Lambda 的事件物件

當您將 Lambda 函數設定為警示動作時，會在 Lambda 函數叫用函數時將 JSON 承載 CloudWatch 傳遞給該函數。此 JSON 承載用作函數的事件物件。您可以從此 JSON 物件中提取資料並在函數中使用它。以下是來自指標警示的事件物件範例。

```
{
  'source': 'aws.cloudwatch',
  'alarmArn': 'arn:aws:cloudwatch:us-east-1:444455556666:alarm:lambda-demo-metric-
alarm',
  'accountId': '444455556666',
  'time': '2023-08-04T12:36:15.490+0000',
  'region': 'us-east-1',
  'alarmData': {
    'alarmName': 'lambda-demo-metric-alarm',
    'state': {
      'value': 'ALARM',
      'reason': 'test',
      'timestamp': '2023-08-04T12:36:15.490+0000'
    },
    'previousState': {
      'value': 'INSUFFICIENT_DATA',
      'reason': 'Insufficient Data: 5 datapoints were unknown.',
      'reasonData':
 '{"version":"1.0","queryDate":"2023-08-04T12:31:29.591+0000","statistic":"Average","period":60
[],"threshold":5.0,"evaluatedDatapoints":[{"timestamp":"2023-08-04T12:30:00.000+0000"},
{"timestamp":"2023-08-04T12:29:00.000+0000"},
{"timestamp":"2023-08-04T12:28:00.000+0000"},
```

```
{"timestamp":"2023-08-04T12:27:00.000+0000"},
{"timestamp":"2023-08-04T12:26:00.000+0000"}]}',
      'timestamp': '2023-08-04T12:31:29.595+0000'
    },
    'configuration': {
      'description': 'Metric Alarm to test Lambda actions',
      'metrics': [
        {
          'id': '1234e046-06f0-a3da-9534-EXAMPLEe4c',
          'metricStat': {
            'metric': {
              'namespace': 'AWS/Logs',
              'name': 'CallCount',
              'dimensions': {
                'InstanceId': 'i-12345678'
              }
            },
            'period': 60,
            'stat': 'Average',
            'unit': 'Percent'
          },
          'returnData': True
        }
      ]
    }
  }
}
```

以下是來自複合警示的事件物件範例。

```
{
  'source': 'aws.cloudwatch',
  'alarmArn': 'arn:aws:cloudwatch:us-east-1:111122223333:alarm:SuppressionDemo.Main',
  'accountId': '111122223333',
  'time': '2023-08-04T12:56:46.138+0000',
  'region': 'us-east-1',
  'alarmData': {
    'alarmName': 'CompositeDemo.Main',
    'state': {
      'value': 'ALARM',
      'reason': 'arn:aws:cloudwatch:us-
east-1:111122223333:alarm:CompositeDemo.FirstChild transitioned to ALARM at Friday 04
 August, 2023 12:54:46 UTC',
```

```
      'reasonData': '{"triggeringAlarms":[{"arn":"arn:aws:cloudwatch:us-
east-1:111122223333:alarm:CompositeDemo.FirstChild","state":
{"value":"ALARM","timestamp":"2023-08-04T12:54:46.138+0000"}}]}',
      'timestamp': '2023-08-04T12:56:46.138+0000'
    },
    'previousState': {
      'value': 'ALARM',
      'reason': 'arn:aws:cloudwatch:us-
east-1:111122223333:alarm:CompositeDemo.FirstChild transitioned to ALARM at Friday 04
 August, 2023 12:54:46 UTC',
      'reasonData': '{"triggeringAlarms":[{"arn":"arn:aws:cloudwatch:us-
east-1:111122223333:alarm:CompositeDemo.FirstChild","state":
{"value":"ALARM","timestamp":"2023-08-04T12:54:46.138+0000"}}]}',
      'timestamp': '2023-08-04T12:54:46.138+0000',
      'actionsSuppressedBy': 'WaitPeriod',
      'actionsSuppressedReason': 'Actions suppressed by WaitPeriod'
    },
    'configuration': {
      'alarmRule': 'ALARM(CompositeDemo.FirstChild) OR
 ALARM(CompositeDemo.SecondChild)',
      'actionsSuppressor': 'CompositeDemo.ActionsSuppressor',
      'actionsSuppressorWaitPeriod': 120,
      'actionsSuppressorExtensionPeriod': 180
    }
  }
}
```

設定 CloudWatch 警示如何處理遺失的資料

有時候，並非每個指標的預期資料點都會報告給 CloudWatch。例如，發生遺失連線情況、伺服器停機或根據設計指標只會間歇性報告資料時。

CloudWatch 可讓您指定在評估警示時如何處理遺失的資料點。這可協助您設定警示，從而在適合受監控的資料類型時進入 ALARM 狀態。您可以在資料遺失並不表示有問題時避免誤報。

與每個警報始終處於三種狀態之一類似，報告的每個特定數據點 CloudWatch 屬於以下三種類別之一：

- 未違反 (在閾值中)
- 違反 (超出閾值)
- 缺少

對於每個警報，您可以指 CloudWatch 定將遺失的資料點視為下列任一項目：

- notBreaching – 將遺失的資料點視為「良好」且在閾值內
- breaching – 將遺失的資料點視為「不良」且超出閾值
- ignore – 維持目前的警示狀態
- missing – 如果所有資料點在警示評估範圍內遺失，警示會轉換為 INPOLATENT_DATA。

最佳選擇取決於度量的類型和警報的用途。例如，如果您要使用持續報告資料的量度建立應用程式回復警示，您可能會想要將遺失的資料點視為違規，因為這可能表示發生錯誤。但是，如果指標只在發生錯誤時才會產生資料點 (例如 Amazon DynamoDB 中的 ThrottledRequests)，則建議將遺失資料視為 notBreaching。預設行為是 missing。

> **Important**
>
> 如果缺少指標資料點，Amazon EC2 指標上設定的警示可以暫時進入不足的狀態。這種情況很少見，但可能會在指標報告中斷時發生，即使 Amazon EC2 執行個體正常運作良好。對於設定為停止、終止、重新開機或復原動作的 Amazon EC2 指標上的警示，我們建議您將這些警示設定為將遺失的資料視為missing，並讓這些警示僅在處於 ALARM 狀態時觸發。

為警示選擇最佳選項，可避免不必要且誤導的警示條件變更，也可以更準確指出系統的運作狀態。

> **Important**
>
> 即使您在警示處理遺漏資料的方式中選擇了其他選項，在 AWS/DynamoDB 命名空間中評估指標的警示始終會忽略遺漏的資料。如果 AWS/DynamoDB 指標有遺漏資料，則評估該指標的警示將保持其目前狀態。

資料遺失時評估警示狀態的方式

每當警示評估是否要變更狀態時，都會 CloudWatch 嘗試擷取比指定為「評估期間」數目更多的資料點。它會嘗試擷取的資料點數量取決於警示期間的長度，以及使用的是標準解析度指標還是高解析度指標。其嘗試擷取的資料點時間範圍便是評估範圍。

CloudWatch 擷取這些資料點後，會發生下列情況：

- 如果評估範圍內沒有資料點遺失，請根據最近收集的資料點來評 CloudWatch 估警示。評估的資料點數目等於警示的 Evaluation Periods (評估期間) 的值。不需要評估範圍內較久遠的額外資料點，且會予以忽略。
- 如果缺少評估範圍中的某些資料點，但成功從評估範圍擷取的現有資料點總數等於或大於警示的「評估期間」， CloudWatch 則會根據最近成功擷取的實際資料點來評估警示狀態，包括從評估範圍之後的必要額外資料點。在這種情況下，您不需要設定如何處理資料遺失的數值並可忽略。
- 如果缺少評估範圍中的某些資料點，且擷取的實際資料點數量低於警示的「評估期間」數目，請使用您為處理遺失資料的方式指定的結果 CloudWatch 填入遺失的資料點，然後評估警示。但是，評估範圍內的所有實際數據點都包含在評估中。 CloudWatch 使用缺少的資料點只有盡可能少的次數。

> **Note**
>
> 此行為的特殊情況是，在指標停止流動之後， CloudWatch 警示可能會在一段時間內重複評估最後一組資料點。此重新評估可能會導致警示變更狀態和重新執行動作 (如果在指標串流停止前便已立即變更狀態的話)。為了減少這種行為，請使用較短的期間。

下表說明範例警示評估行為。在第一個表格中，警示和評估期間的資料點都是 3。 CloudWatch 在評估警報時擷取 5 個最新的資料點，以防最近 3 個資料點遺失。5 是警報的評估範圍。

第 1 欄顯示了 5 個最新的資料點，因為評估範圍為 5。這些資料點會與右側的資料點一起顯示。0 是未違反的資料點、X 是違反的資料點，以及 - 是遺失的資料點。

欄 2 顯示遺失多少必要的 3 個資料點。即使最近 5 個資料點已完成評估，只需要 3 (Evaluation Periods (評估期間) 的設定) 個便可以評估警示狀態。欄 2 中資料點的數量，是必須使用如何處理遺失資料設定「填入」的資料點數量。

在 3-6 欄中，欄標題是如何處理遺失資料的可能值。這些欄位中的列會顯示針對處理遺失資料的每種可能方式設定的警示狀態。

資料點	必須填充的資料點數	缺少	IGNORE	BREACHING	NOT BREACHING
0 - X - X	0	OK	OK	OK	OK
- - - - 0	2	OK	OK	OK	OK

資料點	必須填充的資料點數	缺少	IGNORE	BREACHING	NOT BREACHING
- - - - -	3	INSUFFICIENT_DATA	保留目前狀態	ALARM	OK
0 X X - X	0	ALARM	ALARM	ALARM	ALARM
- - X - -	2	ALARM	保留目前狀態	ALARM	OK

在前一個表格的第二列中，該警示會保持在 OK，即使將遺失資料都視為違規也一樣，因為一個現有的資料點沒有違規，並且它是隨著兩個視為違規的遺失資料點評估的。下一次評估此警示時，若仍然遺失資料，它便會進入 ALARM，因為未違反的資料點將不再於評估範圍內。

第三列 (最近五個資料點全部遺失) 說明了如何處理遺失資料的各種設定會如何影響警示狀態。如果遺失的資料點被視為違規，則警示會進入 ALARM 狀態，而如果它們被視為未違規，則警報會進入 OK 狀態。如果忽略遺失的資料點，警示會保留遺失資料點之前的目前狀態。如果遺失的資料點被視為遺失，則警示沒有足夠的最新實際資料來進行評估，並且會進入 INSUFFICIENT_DATA。

在第四列，警示進入 ALARM 狀態，因為當警示的最近三個資料點違規時，警示的 Evaluation Periods (評估期間) 和 Datapoints to Alarm (資料點到警示) 均設為 3。於此狀況下，遺失資料點會予以忽略，且無須如何評估遺失資料的設定，因為有 3 個實際資料點要評估。

第 5 列代表警示評估的特殊情況，稱為過早警示狀態。如需詳細資訊，請參閱 避免過早轉換到警示狀態。

在下一個表格中，Period (期間) 會再次設為 5 分鐘，Datapoints to Alarm (要警示的資料點) 為 2，而 Evaluation Periods (評估期間) 則為 3。這是一個 2 來自 3、M 來自 N 警示。

評估範圍為 5。這是已擷取且可以在遺失某些資料點時使用的最近資料點數量上限。

資料點	# 遺失資料點	MISSING	IGNORE	BREACHING	NOT BREACHING
0 - X - X	0	ALARM	ALARM	ALARM	ALARM
0 0 X 0 X	0	ALARM	ALARM	ALARM	ALARM
0 - X - -	1	OK	OK	ALARM	OK

資料點	# 遺失資料點	MISSING	IGNORE	BREACHING	NOT BREACHING
- - - - 0	2	OK	OK	ALARM	OK
- - - - X	2	ALARM	保留目前狀態	ALARM	OK

在第 1 列和第 2 列中，警示始終處於 ARMARM 狀態，因為 3 個最新的資料點中有 2 個違規。在第 2 列中，不需要評估範圍中的兩個最舊的資料點，因為 3 個最新的資料點都不會遺失，因此會忽略這兩個較舊的資料點。

在第 3 列和第 4 列中，只有當遺失資料被視為違規時，警示才會進入 ALARM 狀態，在這種情況下，兩個最近的遺失資料點都會被視為違規。在第 4 列，這兩個被視為違規的遺失資料點提供兩個必要違規的資料點以觸發 ALARM 狀態。

第 5 列代表警示評估的特殊情況，稱為過早警示狀態。如需詳細資訊，請參閱下一節。

避免過早轉換到警示狀態

CloudWatch 警報評估包括嘗試避免誤報的邏輯，在此情況下，當資料間歇時，警報會過早進入 ALARM 狀態。前一節表格中第 5 列顯示的範例說明了此邏輯。在這些資料列中，以及在下列範例中，Evaluation Periods (評估期間) 是 3，而評估範圍是 5 個資料點。Datapoints to Alarm (要警示的資料點) 是 3，除了 N 個例子中的 M 之外，其中 Datapoints to Alarm (要警示的資料點) 是 2。

假設警示的最新資料是 - - - - X，其中有四個遺失的資料點，然後有一個作為最新資料點的違規資料點。由於下一個資料點可能是未違反，因此當資料為 - - - - X 或 - - - X - 且 Datapoints to Alarm (要警示的資料點) 為 3 時，警示不會立即進入 ARMARM 狀態。透過這種方式，當下一個資料點沒有違反時，可避免誤判，並導致資料成為 - - - X 0 或 - - X - 0。

但是，如果最新幾個資料點是 - - X - -，即使遺失的資料點被視為遺失，警示也會進入 ARMARM 狀態。這是因為警示旨在於 Evaluation Periods (評估期間) 最舊的可用違反資料點數目至少與 Datapoints to Alarm (要警示的資料點) 的值一樣陳舊並且所有其他最新的資料點違規或遺失時，始終能進入 ALARM 狀態。在這種情況下，即使可用的資料點總數小於 M (Datapoints to Alarm (要警示的資料點))，則警示會進入 ALARM 狀態。

此警示邏輯也適用於 N 個中的 M 個警報。如果評估範圍期間最舊的違規資料點至少與 Datapoints to Alarm (要警示的資料點) 同樣陳舊，並且所有最新的資料點均已違規或遺失，則無論 M (Datapoints to Alarm (要警示的資料點)) 的值為何，警示會進入 ALARM 狀態。

高解析度警示

如果您在高解析度指標設定警示，您可以指定期間為 10 秒或 30 秒的高解析度警示，或者設定期間為 60 秒的任何倍數的定期警示。高解析度警示費用更高。如需高解析度指標的詳細資訊，請參閱 發佈 自訂指標。

數學運算式的警示

您可以針對以一或多個 CloudWatch 量度為基礎的數學運算式結果設定警示。用於警示的數學表達式最多可包含 10 個指標。每個指標都必須使用相同的期間。

對於以數學運算式為基礎的警示，您可以指定如 CloudWatch 何處理遺失的資料點。在此情況下，如果數學表達式未傳回該資料點的值，會將該資料點視為遺失。

以數學表達式為基礎的警示無法執行 Amazon EC2 動作。

如需指標數學表達式和語法的詳細資訊，請參閱使用指標數學。

以百分比為基礎的 CloudWatch 警報和低資料樣本

當您設定一個百分位數作為警示的統計資料，您可以指定當沒有足夠的資料可進行良好的統計評估時該進行的動作。您可以選擇無論如何讓警示評估統計資料，並盡可能變更警示狀態。或者，您可以在範例大小為低時讓警示忽略指標，並等待直到有足夠的資料時再進行評估。

對於 0.5 (含) 和 1.00 (含) 之間的百分位數，會在評估期間有少於 10/(1 個百分位數) 資料點時使用此設定。例如，如果對於在 p99 百分位數的警示有少於 1000 個範例，就會使用此設定。對於 0 和 0.5 (不含) 之間的百分位數，會在評估期間有少於 10/(1 個百分位數) 資料點時使用此設定。

CloudWatch 警報的常見功能

下列功能適用於所有 CloudWatch 鬧鐘：

- 您可建立的警示數量沒有限制。若要建立或更新警示，請使用 CloudWatch 主控台、PutMetricAlarmAPI 動作或 AWS CLI. put-metric-alarm
- 警示名稱只能包含 UTF-8 字元，不能包含 ASCII 控制字元。
- 您可以列出任何或所有目前設定的警示，並列出處於特定狀態的任何警示，方法是使用 CloudWatch 主控台、DescribeAlarmsAPI 動作或中的 describe- alarm 命令。 AWS CLI

- 您可以使用和 EnableAlarmActionsAPI 動作或中的DisableAlarmActions和enable-alarm-actions命令來停用disable-alarm-actions和啟用警示 AWS CLI。
- 您可以使用 SetAlarmStateAPI 動作或中的set-alarm-state指令，將警示設定為任何狀態來測試警示 AWS CLI。此臨時狀態變更只持續直到下一個警示比較發生為止。
- 您可以在建立自訂指標之前，為自訂指標建立警示。若要讓警示有效，您必須包含自訂指標的所有維度，除了在警示定義中的指標命名空間和指標名稱。若要執行此操作，您可以使用 PutMetricAlarmAPI 動作或 AWS CLI. put-metric-alarm
- 您可以使用 CloudWatch 主控台、DescribeAlarmHistoryAPI 動作或中的describe-alarm-history指令來檢視警示的歷史記錄 AWS CLI。 CloudWatch 保留警報歷史記錄 30 天。每個狀態轉換都標示有唯一時間戳記。在極少數情況下，您歷史記錄可能會顯示一個以上的狀態變更通知。時間戳記可讓您確認獨特的狀態變更。
- 您可以在 CloudWatch 主控台導覽窗格的 [我的最愛] 和 [最近] 選項中將游標停留在您要加入最愛的鬧鐘上，並選擇旁邊的星號符號，將鬧鐘設為最愛。
- 警示的評估期間數乘以每個評估期間的長度後，不得超過一天。

> **Note**
>
> 某些 AWS 資源 CloudWatch 在特定條件下不會將指標資料傳送至。例如，Amazon EBS 可能不會傳送並未連接到 Amazon EC2 執行個體可用磁碟區的指標資料，因為針對該磁碟區沒有要監控的指標活動。若您有為這類指標設定警示，您可能會注意到其狀態變更為 `INSUFFICIENT_DATA`。這可能表示您的資源並未處在作用中，而不一定表示發生問題。您可以指定每個警示要如何處理遺失的資料。如需詳細資訊，請參閱 設定 CloudWatch 警示如何處理遺失的資料。

服務的最佳做法警示建 AWS 議

CloudWatch 提供 out-of-the 箱警報建議。我們建議您針對由其他 AWS 服務發佈的指標建立這些 CloudWatch 警示。這些建議可協助您識別應設定警示的指標，以遵循監控的最佳實務。這些建議也會建議要設定的警示閾值。遵循這些建議可以幫助您不會錯過對 AWS 基礎結構的重要監視。

若要尋找警示建議，請使用 CloudWatch 主控台的指標區段，然後選取警示建議篩選器切換。如果您在控制台中導航到建議的警報，然後創建建議的警報，則 CloudWatch 可以預先填寫某些鬧鐘設置。針對某些建議的警示，也會預先填入警示閾值。您也可以使用主控台下載建議警 infrastructure-as-code 報的警示定義，然後使用此程式碼在 AWS CloudFormation AWS CLI、或 Terraform 中建立警示。

也可以在 建議的警示 中查看建議的警示清單。

您需要為您建立的鬧鐘收費，費率與您在中建立的任何其他鬧鐘相同 CloudWatch。使用建議不會產生額外費用。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

尋找並建立建議的警示

請依照下列步驟尋找建 CloudWatch 議您設定警示的量度，並選擇性地建立其中一個警示。第一個程序說明如何尋找具有建議警示的指標，以及如何建立其中一個警示。

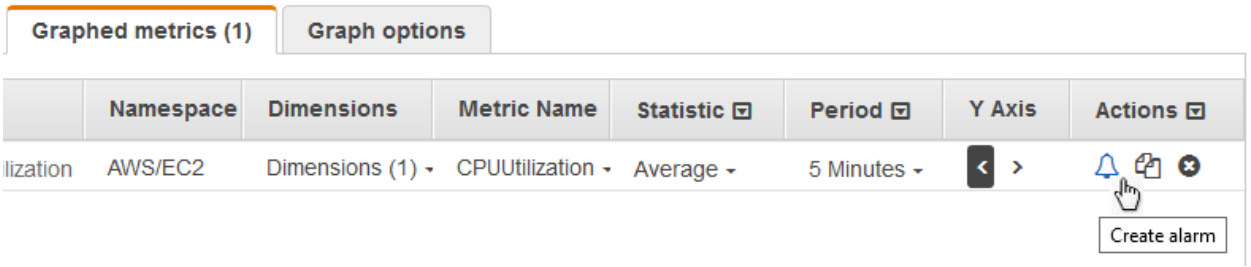
您也可以大量下載 AWS 命名空間中所有建議警示的警示定義，例如AWS/Lambda或AWS/S3。infrastructure-as-code 本主題稍後將會介紹這些指令。

尋找建議警示的指標，並建立單一建議警示

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇 Metrics (指標)、All metrics (所有指標)。
3. 在指標資料表上方，選擇警示建議。

   系統會篩選指標命名空間清單，以僅包括具有警示建議的指標，以及您帳戶中正在發布的服務。
4. 選擇服務的命名空間。

   系統會篩選此命名空間下的指標清單，以僅包含具有警示建議的指標。
5. 若要查看指標的警示目的和建議的閾值，請選擇檢視詳細資訊。
6. 若要為其中一個指標建立警示，請執行下列其中一項操作：
   - 若要使用主控台建立警示，請執行下列操作：
     a. 選取指標的核取方塊，然後選擇圖形化指標索引標籤。
     b. 選擇警示圖示。



   警示建立精靈即會出現，並根據警示建議填入指標名稱、統計資料和期間。如果建議包括特定閾值，則還會預先填入該值。

c. 選擇下一步。

d. 在通知下，選取 SNS 主題，以便在警示轉換為 ALARM 狀態、OK 狀態或 INSUFFICIENT_DATA 狀態時進行通知。

若要讓警示針對相同的警示狀態或不同警示狀態傳送多個通知，請選擇 Add notification (新增通知)。

若要讓警示不傳送通知，請選擇 Remove (移動)。

e. 若要讓警示執行 Auto Scaling 或 EC2 動作，請選擇適當的按鈕，然後選擇警示狀態及要執行的動作。

f. 完成時，請選擇下一步。

g. 輸入警示的名稱與說明。名稱只能包含 ASCII 字元。然後選擇下一步。

h. 在 Preview and create (預覽及建立) 下，請確認資訊和條件都是您希望的內容，然後選擇 Create alarm (建立警示)。

- 若要下載 infrastructure-as-code 警示定義以在、或 Terraform 中使用 AWS CloudFormation AWS CLI，請選擇 [下載鬧鐘代碼]，然後選取您想要的格式。下載的程式碼將具有指標名稱、統計資料和閾值的建議設定。

下載 infrastructure-as-code AWS 服務之所有建議警示的警示定義

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇 Metrics (指標)、All metrics (所有指標)。
3. 在指標資料表上方，選擇警示建議。

   系統會篩選指標命名空間清單，以僅包括具有警示建議的指標，以及您帳戶中正在發布的服務。
4. 選擇服務的命名空間。

   系統會篩選此命名空間下的指標清單，以僅包含具有警示建議的指標。
5. 「下載」警示代碼會顯示此命名空間中指標建議使用的警示數量。若要下載所有建議 infrastructure-as-code 鬧鐘的警示定義，請選擇 [下載鬧鐘代碼]，然後選擇您想要的程式碼格式。

建議的警示

以下各節列出了我們建議您設定最佳實務警示的指標。針對每項指標，還會顯示維度、警示目的、建議閾值、閾值對正，以及期間長度和資料點數目。

某些指標可能會在清單中出現兩次。若針對指標的不同維度組合建議不同的警示，就會發生這種情況。

警示資料點是指將警示傳送至 ALARM 狀態時必須違反的資料點數目。評估期是指評估警示時要考慮的期間數。如果這些數字相同，則僅當該連續週期數的值超出閾值時，警示才會進入 ALARM 狀態。如果警示資料點數目低於評估期數目，則為「M 超出 n」警示，並且如果在資料點設定的任何評估期內至少違反警示資料點的資料點數目，則警示會進入 ALARM 狀態。如需詳細資訊，請參閱 評估警示。

主題

- Amazon API Gateway
- Amazon EC2 Auto Scaling
- Amazon CloudFront
- Amazon Cognito
- Amazon DynamoDB
- Amazon EBS
- Amazon EC2
- Amazon ElastiCache
- Amazon EC2 (AWS/ElasticGPUs)
- Amazon ECS
- 具有 Container Insights 的 Amazon ECS
- Amazon EFS
- 具有 Container Insights 的 Amazon EKS
- Amazon Kinesis Data Streams
- Lambda
- Lambda Insights
- Amazon VPC (AWS/NATGateway)
- AWS 私人連結 (AWS/PrivateLinkEndpoints)
- AWS 私人連結 (AWS/PrivateLinkServices)
- Amazon RDS
- Amazon Route 53 Public Data Plane
- Amazon S3
- S3ObjectLambda
- Amazon SNS

- Amazon SQS
- AWS VPN

Amazon API Gateway

4XXError

尺寸:ApiName, 舞台

警示描述：此警示可偵測用戶端的高速率錯誤。這可能表示授權或用戶端請求參數中存在問題。這也可能意味著資源已移除，或用戶端正在請求不存在的資源。請考慮啟用 CloudWatch 記錄檔，並檢查是否有任何可能造成 4XX 錯誤的錯誤。此外，請考慮啟用詳細 CloudWatch 度量來檢視每個資源和方法的此量度，並縮小錯誤來源的範圍。錯誤也可能是由於超過設定的限流限制所引起。如果回應和日誌均報告較高且非預期速率的 429 錯誤，請遵循本指南以對此問題進行故障診斷。

目的：此警示可偵測 API Gateway 請求中用戶端的高速率錯誤。

統計資料：平均值

建議的閾值：0.05

閾值對正：建議的閾值會在請求總計的 5% 以上出現 4XX 錯誤時偵測到。但是，您可調整閾值以適應請求的流量以及可接受的錯誤率。您還可分析歷史資料，以確定應用程式工作負載可接受的錯誤率，然後相應地調整閾值。經常發生的 4XX 錯誤需要觸發警示。但是，將閾值設定為極低的值可能會導致警示過於敏感。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

5XXError

尺寸:ApiName, 舞台

警示描述：此警示有助於偵測用戶端的高速率錯誤。這可能表示 API 後端、網路或 API 閘道與後端 API 之間的整合存在問題。本文件可協助您對 5xx 錯誤的原因進行故障診斷。

目的：此警示可偵測 API Gateway 請求中伺服器端的高速率錯誤。

統計資料：平均值

建議的閾值：0.05

閾值對正：建議的閾值會在請求總計的 5% 以上出現 5XX 錯誤時偵測到。但是，您可調整閾值以符合請求的流量以及可接受的錯誤率。您還可以分析歷史資料，以確定應用程式工作負載可接受的錯誤率，然後相應地調整閾值。經常發生的 5XX 錯誤需要觸發警示。但是，將閾值設定為極低的值可能會導致警示過於敏感。

期間：60

警示資料點數目：3

評估期：3

比較運算子：GREATER_THAN_THRESHOLD

Count (計數)

尺寸:ApiName, 舞台

警示描述：此警示有助於偵測低流量 REST API 階段。這可能是應用程式呼叫 API (例如使用不正確的端點) 發生問題的指示器。還可能是 API 設定或許可問題的指示器，讓用戶端無法連線。

目的：此警示可偵測非預期低流量 REST API 階段。如果您的 API 在正常情況下接收可預測且一致的請求數目，則建議您建立此警示。如果您啟用了詳細的 CloudWatch 指標，並且可以預測每個方法和資源的正常流量，我們建議您建立替代警示，以更精細地監控每個資源和方法的流量下降。建議不要將此警示用於預期並非具有持續且一致流量的 API。

統計：SampleCount

建議的閾值：視乎您的情況而定

閾值對正：根據歷史資料分析來設定閾值，以確定 API 的預期基準請求計數為何。將閾值設定為極高的值，可能會導致警示在正常和預期低流量期間過於敏感。相反，將其設定為極低的值，可能會導致警示錯過流量中較小的異常下降。

期間：60

警示資料點數目：10

評估期：10

比較運算子：LESS_THAN_THRESHOLD

Count (計數)

維度：ApiName、階段、資源、方法

警示描述：此警示有助於偵測階段中 REST API 資源和方法的低流量錯誤。這可能表示應用程式呼叫 API (例如使用不正確的端點) 發生問題。還可能是 API 設定或許可問題的指示器，讓用戶端無法連線。

目的：此警示可偵測階段中 REST API 資源和方法的非預期低流量錯誤。如果您的 API 在正常情況下接收可預測且一致的請求數目，則建議您建立此警示。建議不要將此警示用於預期並非具有持續且一致流量的 API。

統計：SampleCount

建議的閾值：視乎您的情況而定

閾值對正：根據歷史資料分析來設定閾值，以確定 API 的預期基準請求計數為何。將閾值設定為極高的值，可能會導致警示在正常和預期低流量期間過於敏感。相反，將其設定為極低的值，可能會導致警示錯過流量中較小的異常下降。

期間：60

警示資料點數目：10

評估期：10

比較運算子：LESS_THAN_THRESHOLD

Count (計數)

尺寸:ApiId, 舞台

警示描述：此警示有助於偵測低流量 HTTP API 階段。這可能表示應用程式呼叫 API (例如使用不正確的端點) 發生問題。還可能是 API 設定或許可問題的指示器，讓用戶端無法連線。

目的：此警示可偵測非預期低流量 HTTP API 階段。如果您的 API 在正常情況下接收可預測且一致的請求數目，則建議您建立此警示。如果您啟用了詳細的 CloudWatch 指標，並且可以預測每條路由的正常流量，我們建議您為此創建替代警報，以便對每個路由的流量下降進行更精細的監控。建議不要將此警示用於預期並非具有持續且一致流量的 API。

統計：SampleCount

建議的閾值：視乎您的情況而定

閾值對正：根據歷史資料分析來設定閾值，以確定 API 的預期基準請求計數為何。將閾值設定為極高的值，可能會導致警示在正常和預期低流量期間過於敏感。相反，將其設定為極低的值，可能會導致警示錯過流量中較小的異常下降。

期間：60

警示資料點數目：10

評估期：10

比較運算子：LESS_THAN_THRESHOLD

Count (計數)

維度：ApiId、階段、資源、方法

警示描述：此警示有助於偵測階段中 HTTP API 路由的低流量錯誤。這可能表示應用程式呼叫 API (例如使用不正確的端點) 發生問題。還可能表示 API 設定或許可問題，使用戶端無法連線。

目的：此警示可偵測階段中 HTTP API 路由的非預期低流量錯誤。如果您的 API 在正常情況下接收可預測且一致的請求數目，則建議您建立此警示。建議不要將此警示用於預期並非具有持續且一致流量的 API。

統計：SampleCount

建議的閾值：視乎您的情況而定

閾值對正：根據歷史資料分析來設定閾值，以確定 API 的預期基準請求計數為何。將閾值設定為極高的值，可能會導致警示在正常和預期低流量期間過於敏感。相反，將其設定為極低的值，可能會導致警示錯過流量中較小的異常下降。

期間：60

警示資料點數目：10

評估期：10

比較運算子：LESS_THAN_THRESHOLD

IntegrationLatency

尺寸:ApiId, 舞台

警示描述：此警示有助於偵測階段中 API 請求是否存在較高的整合延遲。您可將 `IntegrationLatency` 指標值與後端的對應延遲指標 (例如 Lambda 整合的 `Duration` 指標) 建立關聯。這可協助您確定 API 後端是否因效能問題而需要更多時間來處理用戶端的請求，或是初始化或冷啟動是否存在某些其他額外負荷。此外，請考慮為 API 啟用 CloudWatch 日誌，並檢查日誌中是否有任何可能導致高延遲問題的錯誤。此外，請考慮啟用詳細 CloudWatch 指標以取得每個路由的此指標檢視，以協助您縮小整合延遲的來源範圍。

目的：此警示可偵測階段中的 API Gateway 請求何時具有較高的整合延遲。我們建議您針對 WebSocket API 使用此警示，而且我們認為 HTTP API 是選用的，因為這些警示已針對「延遲」度量具有個別的警示建議。如果您已啟用詳細 CloudWatch 指標，而且每個路由都有不同的整合延遲效能需求，建議您建立替代警示，以便對每個路由的整合延遲進行更精細的監控。

統計資料：p90

建議的閾值：2000.0

閾值對正：建議的閾值不適用於所有 API 工作負載。但是，您可將其用做閾值的起點。然後，您可根據 API 的工作負載和可接受的延遲、效能和 SLA 需求，選擇不同的閾值。如果一般情況下可接受 API 具有較高的延遲，請設定較高的閾值以使警示不太敏感。但是，如果 API 預期會提供近乎即時的回應，請設定較低的閾值。您還可分析歷史資料，以確定應用程式工作負載的預期基準延遲，然後用於相應地調整閾值。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_OR_EQUAL_TO_THRESHOLD

IntegrationLatency

尺寸：ApiId，舞台，路線

警示說明：此警示有助於偵測階段中路由的 WebSocket API 要求是否有很高的整合延遲。您可將 `IntegrationLatency` 指標值與後端的對應延遲指標 (例如 Lambda 整合的 `Duration` 指標) 建立關聯。這可協助您確定 API 後端是否因效能問題而需要更多時間來處理用戶端的請求，或是初始化或冷啟動是否有其他額外負荷。此外，請考慮為 API 啟用 CloudWatch 日誌，並檢查日誌中是否有任何可能導致高延遲問題的錯誤。

目的：此警示可偵測階段中路由的 API Gateway 請求何時具有較高的整合延遲。

統計資料：p90

建議的閾值：2000.0

閾值對正：建議的閾值不適用於所有 API 工作負載。但是，您可將其用做閾值的起點。然後，您可根據 API 的工作負載和可接受的延遲、效能和 SLA 需求，選擇不同的閾值。如果一般情況下可接受 API 具有較高的延遲，您可設定較高的閾值以使警示不太敏感。但是，如果 API 預期會提供近乎即時的回應，請設定較低的閾值。您還可分析歷史資料，以確定應用程式工作負載的預期基準延遲，然後用於相應地調整閾值。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Latency (延遲)

尺寸:ApiName, 舞台

警示描述：此警示會偵測階段中的高延遲錯誤。尋找 `IntegrationLatency` 指標值以檢查 API 後端延遲。如果兩個指標大多情況下是一致的，則 API 後端是較高延遲的來源，您應在此調查是否存在問題。也請考慮啟用 CloudWatch 記錄檔，並檢查是否有可能造成高延遲的錯誤。此外，請考慮啟用詳細 CloudWatch 指標以檢視每個資源和方法的量度，並縮小延遲來源的範圍。如果適用，請參閱 Lambda 故障診斷或邊緣優化的 API 端點故障診斷指南。

目的：此警示可偵測階段中的 API Gateway 請求何時具有較高的延遲。如果您已啟用詳細 CloudWatch 指標，而且每個方法和資源的延遲效能需求都有不同，建議您建立替代警示，以更精細地監控每個資源和方法的延遲。

統計資料：p90

建議的閾值：2500.0

閾值對正：建議的閾值不適用於所有 API 工作負載。但是，您可將其用做閾值的起點。然後，您可根據 API 的工作負載和可接受的延遲、效能和 SLA 需求，選擇不同的閾值。如果一般情況下可接受 API 具有較高的延遲，您可設定較高的閾值以使警示不太敏感。但是，如果 API 預期會提供近乎即時的回應，請設定較低的閾值。您還可分析歷史資料，以確定應用程式工作負載的預期基準延遲，然後相應地調整閾值。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Latency (延遲)

維度：ApiName、階段、資源、方法

警示說明：此警示可偵測階段中資源和方法的高延遲錯誤。尋找 `IntegrationLatency` 指標值以檢查 API 後端延遲。如果兩個指標大多情況下是一致的，則 API 後端是較高延遲的來源，您應在此調查是否存在效能問題。也請考慮啟用 CloudWatch 記錄檔，並檢查是否有任何可能造成高延遲的錯誤。如果適用，您還可參閱 Lambda 故障診斷或邊緣優化的 API 端點故障診斷指南。

目的：此警示可偵測階段中資源和方法的 API Gateway 請求何時具有較高的延遲。

統計資料：p90

建議的閾值：2500.0

閾值對正：建議的閾值不適用於所有 API 工作負載。但是，您可將其用做閾值的起點。然後，您可根據 API 的工作負載和可接受的延遲、效能和 SLA 需求，選擇不同的閾值。如果一般情況下可接受 API 具有較高的延遲，您可設定較高的閾值以使警示不太敏感。但是，如果 API 預期會提供近乎即時的回應，請設定較低的閾值。您還可分析歷史資料，以確定應用程式工作負載的預期基準延遲，然後相應地調整閾值。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Latency (延遲)

尺寸:ApiId, 舞台

警示描述：此警示會偵測階段中的高延遲錯誤。尋找 `IntegrationLatency` 指標值以檢查 API 後端延遲。如果兩個指標大多情況下是一致的，則 API 後端是較高延遲的來源，您應在此調查是否存在效能問題。也請考慮啟用 CloudWatch 記錄檔，並檢查是否有任何可能造成高延遲的錯誤。此外，請考慮啟用詳細 CloudWatch 指標以檢視每個路由的指標，並縮小延遲來源的範圍。如果適用，您還可參閱 Lambda 整合故障診斷指南。

目的：此警示可偵測階段中的 API Gateway 請求何時具有較高的延遲。如果您已啟用詳細的 CloudWatch 指標，而且每個路由都有不同的延遲效能需求，建議您建立替代警示，以便對每個路由的延遲進行更精細的監控。

統計資料：p90

建議的閾值：2500.0

閾值對正：建議的閾值不適用於所有 API 工作負載。但其可用做閾值的起點。然後，您可根據 API 的工作負載和可接受的延遲、效能和 SLA 需求，選擇不同的閾值。如果一般情況下可接受 API 具有較高的延遲，則您可設定較高的閾值以讓其不太敏感。但如果 API 預期會提供近乎即時的回應，則設定較低的閾值。您還可分析歷史資料，以確定應用程式工作負載的預期基準延遲，然後相應地調整閾值。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Latency (延遲)

維度：ApiId、階段、資源、方法

警示說明：此警示可偵測階段中路由的高延遲錯誤。尋找 `IntegrationLatency` 指標值以檢查 API 後端延遲。如果兩個指標大多情況下是一致的，則 API 後端是較高延遲的來源，並且應調查是否存在效能問題。也請考慮啟用 CloudWatch 記錄檔，並檢查是否有任何可能造成高延遲的錯誤。如果適用，您還可參閱 Lambda 整合故障診斷指南。

目的：此警示用於偵測階段中路由的 API Gateway 請求何時具有較高的延遲。

統計資料：p90

建議的閾值：2500.0

閾值對正：建議的閾值不適用於所有 API 工作負載。但其可用做閾值的起點。然後，您可根據 API 的工作負載和可接受的延遲、效能和 SLA 需求，選擇不同的閾值。如果一般情況下可接受 API 具有較高的延遲，您可設定較高的閾值以使警示不太敏感。但是，如果 API 預期會提供近乎即時的回應，請設定較低的閾值。您還可分析歷史資料，以確定應用程式工作負載的預期基準延遲，然後相應地調整閾值。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_OR_EQUAL_TO_THRESHOLD

4xx

尺寸:ApiId, 舞台

警示描述：此警示可偵測用戶端的高速率錯誤。這可能表示授權或用戶端請求參數中存在問題。這也可能意味著路由已移除，或用戶端正在請求 API 中不存在的資源。請考慮啟用 CloudWatch 記錄檔，並檢查是否有任何可能造成 4xx 錯誤的錯誤。此外，請考慮啟用詳細 CloudWatch 指標來檢視每個路由的量度，以協助您縮小錯誤來源的範圍。錯誤也可能是由於超過設定的限流限制所引起。如果回應和日誌均報告較高且非預期速率的 429 錯誤，請遵循本指南以對此問題進行故障診斷。

目的：此警示可偵測 API Gateway 請求中用戶端的高速率錯誤。

統計資料：平均值

建議的閾值：0.05

閾值對正：建議的閾值會在請求總計的 5% 以上出現 4xx 錯誤時偵測到。但是，您可調整閾值以適應請求的流量以及可接受的錯誤率。您還可分析歷史資料，以確定應用程式工作負載可接受的錯誤率，然後相應地調整閾值。經常發生的 4xx 錯誤需要觸發警示。但是，將閾值設定為極低的值可能會導致警示過於敏感。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

5xx

尺寸:ApiId, 舞台

警示描述：此警示有助於偵測用戶端的高速率錯誤。這可能表示 API 後端、網路或 API 閘道與後端 API 之間的整合存在問題。本文件可協助您對 5xx 錯誤的原因進行故障診斷。

目的：此警示可偵測 API Gateway 請求中伺服器端的高速率錯誤。

統計資料：平均值

建議的閾值：0.05

閾值對正：建議的閾值會在請求總數的 5% 以上出現 5xx 錯誤時偵測到。但是，您可調整閾值以適應請求的流量以及可接受的錯誤率。您還可分析歷史資料，以確定應用程式工作負載可接受的錯誤率，然後您可相應地調整閾值。經常發生的 5xx 錯誤需要觸發警示。但是，將閾值設定為極低的值可能會導致警示過於敏感。

期間：60

警示資料點數目：3

評估期：3

比較運算子：GREATER_THAN_THRESHOLD

MessageCount

尺寸:ApiId, 舞台

警報說明：此警報有助於檢測 WebSocket API 階段的低流量。這可能表示用戶端呼叫 API 時發生問題，例如使用不正確的端點，或是後端向用戶端傳送訊息時發生問題。還可能表示 API 設定或許可問題，使用戶端無法連線。

意圖：此警報可以檢測 WebSocket API 階段的意外低流量。如果您的 API 在正常情況下接收和傳送可預測且一致的訊息數目，建議您建立此警示。如果您啟用了詳細的 CloudWatch 指標，並且可以預測每條路由的正常流量，則最好為此路由創建替代警報，以便對每條路線的流量下降進行更精細的監控。建議不要將此警示用於預期並非具有持續且一致流量的 API。

統計：SampleCount

建議的閾值：視乎您的情況而定

閾值對正：根據歷史資料分析來設定閾值，以確定 API 的預期基準訊息計數為何。將閾值設定為極高的值，可能會導致警示在正常和預期低流量期間過於敏感。相反，將其設定為極低的值，可能會導致警示錯過流量中較小的異常下降。

期間：60

警示資料點數目：10

評估期：10

比較運算子：LESS_THAN_THRESHOLD

MessageCount

尺寸：ApiId，舞台，路線

警示說明：此警示有助於偵測階段中 WebSocket API 路由的低流量。這可能表示用戶端呼叫 API 時發生問題，例如使用不正確的端點，或是後端向用戶端傳送訊息時發生問題。還可能表示 API 設定或許可問題，使用戶端無法連線。

意圖：此警示可以偵測階段中 WebSocket API 路由的意外低流量。如果您的 API 在正常情況下接收和傳送可預測且一致的訊息數目，建議您建立此警示。建議不要將此警示用於預期並非具有持續且一致流量的 API。

統計：SampleCount

建議的閾值：視乎您的情況而定

閾值對正：根據歷史資料分析來設定閾值，以確定 API 的預期基準訊息計數為何。將閾值設定為極高的值，可能會導致警示在正常和預期低流量期間過於敏感。相反，將其設定為極低的值，可能會導致警示錯過流量中較小的異常下降。

期間：60

警示資料點數目：10

評估期：10

比較運算子：LESS_THAN_THRESHOLD

ClientError

尺寸:ApiId, 舞台

警示描述：此警示可偵測用戶端的高速率錯誤。這可能表示授權或訊息參數中存在問題。這也可能意味著路由已移除，或用戶端正在請求 API 中不存在的資源。請考慮啟用 CloudWatch 記錄檔，並檢查是否有任何可能造成 4xx 錯誤的錯誤。此外，請考慮啟用詳細 CloudWatch 指標來檢視每個路由的量度，以協助您縮小錯誤來源的範圍。錯誤也可能是由於超過設定的限流限制所引起。如果回應和日誌均報告較高且非預期速率的 429 錯誤，請遵循本指南以對此問題進行故障診斷。

意圖：此警示可以偵測 WebSocket API Gateway 訊息的用戶端錯誤的高速率。

統計資料：平均值

建議的閾值：0.05

閾值對正：建議的閾值會在請求總計的 5% 以上出現 4xx 錯誤時偵測到。您可調整閾值以適應請求的流量，以及調整為可接受的錯誤率。您還可分析歷史資料，以確定應用程式工作負載可接受的錯誤率，然後相應地調整閾值。經常發生的 4xx 錯誤需要觸發警示。但是，將閾值設定為極低的值可能會導致警示過於敏感。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

ExecutionError

尺寸:ApiId, 舞台

警示描述：此警示有助於偵測執行的高速率錯誤。這可能是由於您的整合發生 5xx 錯誤、許可問題或其他阻止成功調用整合的因素導致，例如限流或刪除整合。考慮為 API 啟用 CloudWatch 日誌，並檢查日誌的類型和錯誤原因。此外，請考慮啟用詳細 CloudWatch 指標，以取得每個路由的此指標檢視，以協助您縮小錯誤的來源範圍。本文件可協助您對任何連線錯誤的原因進行故障診斷。

意圖：此警示可偵測 WebSocket API Gateway 訊息的高速執行錯誤。

統計資料：平均值

建議的閾值：0.05

閾值對正：建議的閾值會在請求總計的 5% 以上出現執行錯誤時偵測到。您可調整閾值以適應請求的流量，以及調整為可接受的錯誤率。您可分析歷史資料，以確定應用程式工作負載可接受的錯誤率，然後相應地調整閾值。經常發生的執行錯誤需要觸發警示。但是，將閾值設定為極低的值可能會導致警示過於敏感。

期間：60

警示資料點數目：3

評估期：3

比較運算子：GREATER_THAN_THRESHOLD

Amazon EC2 Auto Scaling

GroupInServiceCapacity

尺寸:AutoScalingGroupName

警示描述：此警示有助於偵測群組中的容量何時低於工作負載所需的容量。若要進行故障診斷，請檢查擴展活動是否存在啟動失敗，並確認所需的容量組態正確無誤。

目的：此警示可偵測由於啟動失敗或暫停啟動，Auto Scaling 群組中的低可用性問題。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：閾值應為執行工作負載所需的最小容量。在大多數情況下，您可以將其設定為符合 GroupDesiredCapacity 量度。

期間：60

警示資料點數目：10

評估期：10

比較運算子：LESS_THAN_THRESHOLD

Amazon CloudFront

5 xxErrorRate

尺寸：DistributionId，區域 = 全局

警示說明：此警示會監控來自原始伺服器的 5xx 錯誤回應百分比，以協助您偵測 CloudFront 服務是否有問題。如需可協助您了解伺服器問題的相關資訊，請參閱對原始伺服器的錯誤回應進行故障診斷。此外，開啟額外指標，以取得詳細的錯誤指標。

意圖：此警示用於偵測來自原始伺服器提供要求的問題，或與原始伺服器之間 CloudFront 的通訊問題。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：此警示的建議閾值很大程度上取決於 5xx 回應的容錯值。您可分析歷史資料和趨勢，然後相應地設定閾值。由於 5xx 錯誤可能是由於暫時性問題引起，因此建議您將閾值設定為大於 0 的值，以便警示不會過於敏感。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

OriginLatency

尺寸：DistributionId，區域 = 全局

警示描述：警示有助於監控原始伺服器是否花費太長時間來回應。如果伺服器需要太長時間來回應，可能會導致逾時。如果出現持續較高的 `OriginLatency` 值，請參閱尋找並修正原始伺服器上應用程式的延遲回應。

目的：此警示用於偵測原始伺服器回應時間太長的問題。

統計資料：p90

建議的閾值：視乎您的情況而定

閾值對正：您應計算約 80% 的來源回應逾時值，並使用結果做為閾值。如果此指標持續接近來源伺服器回應逾時值，則可能會開始出現 504 錯誤。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

FunctionValidationErrors

尺寸：DistributionId、、區域 FunctionName = 全域

警示說明：此警示可協助您監控 CloudFront 功能的驗證錯誤，以便您可以採取措施來解決這些錯誤。分析 CloudWatch 函數日誌並查看函數代碼以查找並解決問題的根本原因。請參閱邊緣函數的限制以了解 CloudFront 函數的常見錯誤配置。

意圖：此警報用於檢測 CloudFront 功能的驗證錯誤。

統計資料：總和

建議的閾值：0.0

閾值對正：大於 0 的值表示驗證錯誤。我們建議將閾值設置為 0，因為驗證錯誤意味著 CloudFront 函數交回時出現問題。 CloudFront例如， CloudFront 需要 HTTP 主機標頭才能處理請求。沒有什麼可以阻止用戶刪除其 CloudFront 函數代碼中的 Host 頭文件。但是，當 CloudFront 獲取響應並且 Host 頭丟失時，會 CloudFront 拋出驗證錯誤。

期間：60

警示資料點數目：2

評估期：2

比較運算子：GREATER_THAN_THRESHOLD

FunctionExecutionErrors

尺寸：DistributionId、、區域 FunctionName = 全域

警報說明：此警報可幫助您監控 CloudFront 功能的執行錯誤，以便您可以採取措施來解決這些錯誤。分析 CloudWatch 函數日誌並查看函數代碼以查找並解決問題的根本原因。

意圖：此警報用於檢測 CloudFront 功能執行錯誤。

統計資料：總和

建議的閾值：0.0

閾值對正：建議將閾值設定為 0，因為執行錯誤表示在執行期發生程式碼問題。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

FunctionThrottles

尺寸：DistributionId、、區域 FunctionName = 全域

警報說明：此警報可幫助您監控 CloudFront 功能是否節流。如果函數受到限流，這意味著執行時間太長。若要避免函數限流，請考慮優化函數程式碼。

意圖：此警報可以檢測您的 CloudFront 功能何時被限制，以便您可以做出反應並解決問題，以獲得流暢的客戶體驗。

統計資料：總和

建議的閾值：0.0

閾值對正：建議將閾值設定為 0，以便更快地解析函數限流。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

Amazon Cognito

SignUpThrottles

尺寸:UserPool, UserPoolClient

警示描述：此警示可監控限流請求的計數。如果使用者持續受到限流，則您應透過請求增加服務配額來提高限制。請參閱 Amazon Cognito 中的配額，了解如何請求增加配額。若要主動執行動作，請考慮追蹤用量配額。

目的：此警示有助於監控限流註冊請求的發生。這可協助您知道何時執行動作，以緩解註冊體驗出現任何降級。請求持續限流是一種負面的使用者註冊體驗。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：佈建良好的使用者集區不應遇到跨越多個資料點的任何限流。因此，預期工作負載的典型閾值應為零。對於經常爆量的不規則工作負載，您可分析歷史資料以確定應用程式工作負載可接受的限流，然後您可相應地調整閾值。應重試限流的請求，以將對應用程式的影響降至最低。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

SignInThrottles

尺寸:UserPool, UserPoolClient

警示描述：此警示可監控限流使用者驗證請求的計數。如果使用者持續受到限流，您可能需要透過請求增加服務配額來提高限制。請參閱 Amazon Cognito 中的配額，了解如何請求增加配額。若要主動執行動作，請考慮追蹤用量配額。

目的：此警示有助於監控限流登入請求的發生。這可協助您知道何時執行動作，以緩解登入體驗出現任何降級。請求持續限流是一種糟糕的使用者身分驗證體驗。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：佈建良好的使用者集區不應遇到跨越多個資料點的任何限流。因此，預期工作負載的典型閾值應為零。對於經常爆量的不規則工作負載，您可分析歷史資料以確定應用程式工作負載可接受的限流，然後您可相應地調整閾值。應重試限流的請求，以將對應用程式的影響降至最低。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

TokenRefreshThrottles

尺寸:UserPool, UserPoolClient

警示描述：您可設定符合請求流量的閾值，以及符合權杖重新整理請求的可接受限流。限流用於保護您的系統免受太多請求的影響。但是，監控正常流量是否佈建不足也非常重要。您可分析歷史資料，尋找應用程式工作負載可接受的限流，然後再將警示閾值調整為高於可接受的限流層級。應用程式/服務應重試限流請求，因為它們是暫時的。因此，將閾值設定為極低的值可能會導致警示較為敏感。

目的：此警示有助於監控權杖重新整理請求的發生。這可協助您知道何時執行動作來緩解任何潛在問題，以確保順暢的使用者體驗，以及身分驗證系統的良好運作狀態與可靠性。請求持續限流是一種糟糕的使用者身分驗證體驗。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：還可設定/調整閾值以適應請求的流量，以及權杖重新整理請求的可接受限流。限流可保護您的系統免受太多請求的影響，但是，監控正常流量是否佈建不足並查看是否造成影響也非常重要。還可分析歷史資料，以了解應用程式工作負載可接受的限流，並且可將閾值調整為高於通常可接受的限流層級。應用程式/服務應重試限流請求，因為它們是暫時的。因此，將閾值設定為極低的值可能會導致警示較為敏感。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

FederationThrottles

尺寸:UserPool UserPoolClient, IdentityProvider

警示描述：此警示可監控限流聯合身分請求的計數。如果您持續發現受到限流，可能表示您需要透過請求增加服務配額來提高限制。請參閱 Amazon Cognito 中的配額，了解如何請求增加配額。

目的：此警示有助於監控限流聯合身分請求的發生。這可協助您主動回應效能瓶頸或設定錯誤，並確保為使用者提供順暢的身分驗證體驗。請求持續限流是一種糟糕的使用者身分驗證體驗。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：您可設定閾值以符合請求的流量，以及符合聯合身分請求的可接受限流。限流用於保護您的系統免受太多請求的影響。但是，監控正常流量是否佈建不足也非常重要。您可分析歷史資料，尋找應用程式工作負載可接受的節流，然後將閾值設定為高於可接受限流層級的值。應用程式/服務應重試限流請求，因為它們是暫時的。因此，將閾值設定為極低的值可能會導致警示較為敏感。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

Amazon DynamoDB

AccountProvisionedReadCapacityUtilization

維度：無

警示描述：此警示可偵測帳戶的讀取容量是否達到其佈建限制。如果發生此情況，您可提高讀取容量使用率的帳戶配額。您可使用 Service Quotas 來檢視目前讀取容量單位的配額並請求增加。

目的：該警示可偵測帳戶的讀取容量使用率是否接近其佈建的讀取容量使用率。如果使用率達到其最大限制，DynamoDB 會開始對讀取請求限流。

統計資料：最大值

建議的閾值：80.0

閾值對正：將閾值設定為 80%，以便在達到容量已滿之前可執行動作 (例如提高帳戶限制) 以避免限流。

期間：300

警示資料點數目：2

評估期：2

比較運算子：GREATER_THAN_THRESHOLD

AccountProvisionedWriteCapacityUtilization

維度：無

警示描述：此警示可偵測帳戶的寫入容量是否達到其佈建限制。如果發生此情況，您可提高寫入容量使用率的帳戶配額。您可使用 Service Quotas 來檢視目前寫入容量單位的配額並請求增加。

目的：此警示可偵測帳戶的寫入容量使用率是否接近其佈建的寫入容量使用率。如果使用率達到其最大限制，DynamoDB 會開始對寫入請求限流。

統計資料：最大值

建議的閾值：80.0

閾值對正：將閾值設定為 80%，以便在達到容量已滿之前可執行動作 (例如提高帳戶限制) 以避免限流。

期間：300

警示資料點數目：2

評估期：2

比較運算子：GREATER_THAN_THRESHOLD

AgeOfOldestUnreplicatedRecord

尺寸:TableName, DelegatedOperation

警示描述：此警示可偵測複寫到 Kinesis 資料串流的延遲。在正常的操作下，`AgeOfOldestUnreplicatedRecord` 應僅以毫秒為單位。若不成功的複寫嘗試是因客戶控制的組態選擇所引起，此數字會隨著不成功複寫嘗試的增加而增加。可能導致複寫嘗試失敗的客戶控制組態示例包括，佈建的 Kinesis 資料串流容量不足導致過度限流，或是手動更新 Kinesis 資料串流的存取原則因而拒絕 DynamoDB 新增資料至資料串流。為儘可能降低此指標，您需要確保妥善佈建 Kinesis 資料串流容量，並確保 DynamoDB 的許可保持不變。

目的：此警示可監控複寫嘗試失敗，以及複寫到 Kinesis 資料串流產生的延遲。

統計資料：最大值

建議的閾值：視乎您的情況而定

閾值對正：根據所需的複寫延遲 (以毫秒為單位) 設定閾值。此值取決於工作負載需求和預期效能。

期間：300

警示資料點數目：3

評估期：3

比較運算子：GREATER_THAN_THRESHOLD

FailedToReplicateRecordCount

尺寸:TableName, DelegatedOperation

警示描述：此警示可偵測 DynamoDB 無法複寫到您的 Kinesis 資料串流的記錄數目。大於 34KB 的某些項目可能會擴充大小，以變更大於 Kinesis Data Streams 1MB 項目大小限制的資料記錄。當這些大於 34KB 的項目包含大量的布林值或空白屬性值時，就會發生此大小擴充。布林值和空白屬性值會以 1 位元組形式儲存在 DynamoDB 中，但是在使用用於 Kinesis Data Streams 複寫的標準 JSON 將其序列化時，最多可擴充至 5 個位元組。DynamoDB 無法將這類變更記錄複寫到您的 Kinesis 資料串流。DynamoDB 會略過這些變更資料記錄，並自動繼續複寫後續記錄。

目的：此警示可監控 DynamoDB 因 Kinesis Data Streams 的項目大小限制而無法複寫到 Kinesis 資料串流的記錄數目。

統計資料：總和

建議的閾值：0.0

閾值對正：將閾值設定為 0，以偵測 DynamoDB 無法複寫的任何記錄。

期間：60

警示資料點數目：1

評估期：1

比較運算子：GREATER_THAN_THRESHOLD

ReadThrottleEvents

尺寸:TableName

警示描述：此警示可偵測 DynamoDB 資料表是否存在大量讀取請求限制。若要對該問題進行故障診斷，請參閱對 Amazon DynamoDB 中的限流問題進行故障診斷。

目的：此警示可偵測 DynamoDB 資料表讀取請求的持續限流問題。持續限流讀取請求可能會對工作負載讀取操作產生負面影響，並降低系統的整體效率。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：根據 DynamoDB 資料表的預期讀取流量設定閾值，並以可接受的限流層級為基準。務必要監控是否佈建不足，並且不會導致持續限流。您還可分析歷史資料，尋找應用程式工作負載可接受的限流層級，然後將閾值調整為高於通常的限流層級。應用程式或服務應重試限流請求，因為它們是暫時的。因此，極低的閾值可能會導致警示過於敏感，從而引起不必要的狀態轉換。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

ReadThrottleEvents

尺寸:TableName, GlobalSecondaryIndexName

警示描述：此警示可偵測 DynamoDB 資料表的全域次要索引是否存在大量讀取請求限制。若要對該問題進行故障診斷，請參閱對 Amazon DynamoDB 中的限流問題進行故障診斷。

目的：該警示可偵測 DynamoDB 資料表全域次要索引的讀取請求是否持續限流。持續限流讀取請求可能會對工作負載讀取操作產生負面影響，並降低系統的整體效率。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：根據 DynamoDB 資料表的預期讀取流量設定閾值，並以可接受的限流層級為基準。務必要監控是否佈建不足，並且不會導致持續限流。您還可分析歷史資料，尋找應用程式工作負載可接受的限流層級，然後將閾值調整為高於通常可接受的限流層級。應用程式或服務應重試限流請求，因為它們是暫時的。因此，極低的閾值可能會導致警示過於敏感，從而引起不必要的狀態轉換。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

ReplicationLatency

尺寸:TableName, ReceivingRegion

警示描述：警示可偵測全域資料表「區域」中的複本是否落後於「來源區域」。如果某個 AWS 區域降級，且您在該區域中有複本表格，則延遲可能會增加。在此情況下，您可以暫時將應用程式的讀取和寫入活動重新導向至不同的 AWS 區域。如果您正在使用 2017.11.29 (舊式) 全域資料表，您應確認每個複本資料表的寫入容量單位 (WCU) 是否相同。您還可確定遵循管理容量的最佳實務和需求中的建議。

目的：該警示可偵測區域中的複本資料表是否落後於另一個區域的複寫變更。這可能會導致您的複本與其他複本發生偏離。瞭解每個 AWS 區域的複寫延遲，並在複寫延遲持續增加時發出警示非常有用。資料表的複寫僅適用於全域資料表。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：此警示的建議閾值很大程度上取決於您的使用案例。複寫延遲超過 3 分鐘通常需要進行調查。檢閱複寫延遲的重要性和需求，並分析歷史趨勢，然後相應地選取閾值。

期間：60

警示資料點數目：15

評估期：15

比較運算子：GREATER_THAN_THRESHOLD

SuccessfulRequestLatency

尺寸：TableName，操作

警示描述：此警示可偵測 DynamoDB 資料表操作的高延遲問題 (由警示中的維度值 `Operation` 表示)。如需對 Amazon DynamoDB 中的延遲問題進行故障診斷，請參閱此故障診斷文件。

目的：此警示可偵測 DynamoDB 資料表操作的高延遲問題。較高的操作延遲可能會對系統的整體效率產生負面影響。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值理由：DynamoDB 針對單一作業 (例如 GetItem、 PutItem等) 提供平均 10 毫秒的延遲。但是，您可針對工作負載涉及的操作類型和資料表，根據其可接受的延遲容錯值來設定閾值。您可分析此指標的歷史資料，以尋找資料表操作的一般延遲問題，然後將閾值設定為代表操作嚴重延遲的數字。

期間：60

警示資料點數目：10

評估期：10

比較運算子：GREATER_THAN_THRESHOLD

SystemErrors

尺寸:TableName

警示描述：此警示可偵測 DynamoDB 資料表請求的大量持續性系統錯誤。如果您繼續收到 5xx 錯誤，請開啟 AWS 服務運作狀態儀表板，以檢查服務的操作問題。如果 DynamoDB 發生長時間內部服務問題，您可使用此警示獲得通知，並協助您關聯用戶端應用程式面臨的問題。如需詳細資訊，請參閱 DynamoDB 的錯誤處理。

目的：此警示可偵測 DynamoDB 資料表請求的持續性系統問題。系統錯誤表示 DynamoDB 發生的內部服務錯誤，並有助於與用戶端發生的問題建立關聯。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：根據預期流量設定閾值，並以可接受的系統錯誤層級為基準。您還可分析歷史資料，以尋找應用程式工作負載可接受的錯誤計數，然後相應地調整閾值。應用程式/服務應重試系統錯誤，因為它們是暫時的。因此，極低的閾值可能會導致警示過於敏感，從而引起不必要的狀態轉換。

期間：60

警示資料點數目：15

評估期：15

比較運算子：GREATER_THAN_THRESHOLD

ThrottledPutRecordCount

尺寸:TableName, DelegatedOperation

警示描述：此警示可偵測在將變更資料擷取複寫到 Kinesis 期間，Kinesis 資料串流受到限流的記錄。之所以發生此限流，是因為 Kinesis 資料串流容量不足。如果遇到過多且規律的調節，您可能需要按照觀察到的資料表寫入輸送量按比例增加 Kinesis 串流碎片的數量。若要進一步了解如何判斷 Kinesis 資料串流的大小，請參閱判斷 Kinesis Data Stream 的初始大小。

目的：此警示可監控因 Kinesis 資料串流容量不足而受到 Kinesis Data Streams 限流的記錄數目。

統計資料：最大值

建議的閾值：視乎您的情況而定

閾值對正：在特殊用量尖峰期間，您可能會遇到限流的情況，但限流記錄應儘可能低以避免較高的複寫延遲 (DynamoDB 會重試將限流記錄傳送至 Kinesis 資料串流)。將閾值設定為可協助您擷取常規過度限流的數字。您還可分析此指標的歷史資料，以尋找應用程式工作負載可接受的限流速率。根據您的使用案例，將閾值調整為應用程式可容忍的值。

期間：60

警示資料點數目：10

評估期：10

比較運算子：GREATER_THAN_THRESHOLD

UserErrors

維度：無

警示描述：此警示可偵測 DynamoDB 資料表請求的大量持續性使用者錯誤。您可在問題時間範圍內，檢查用戶端應用程式日誌，以了解請求無效的原因。您可檢查 HTTP 狀態碼 400，以查看收到的錯誤類型，並相應地執行動作。您可能必須修正應用程式邏輯，才能建立有效的請求。

目的：此警示可偵測 DynamoDB 資料表請求的持續性使用者問題。使用者請求操作錯誤意味著用戶端正在產生無效的請求，並且將會失敗。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：將閾值設定為零以偵測任何用戶端錯誤。或者，如果您想要避免因極少的錯誤而觸發警示，可將其設定為更高的值。根據您的使用案例和請求的流量來決定。

期間：60

警示資料點數目：10

評估期：10

比較運算子：GREATER_THAN_THRESHOLD

WriteThrottleEvents

尺寸:TableName

警示描述：此警示可偵測 DynamoDB 資料表是否存在大量寫入請求限制。若要對該問題進行故障診斷，請參閱對 Amazon DynamoDB 中的限流問題進行故障診斷。

目的：此警示可偵測 DynamoDB 資料表讀取寫入的持續限流問題。持續限流寫入請求可能會對工作負載寫入操作產生負面影響，並降低系統的整體效率。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：根據 DynamoDB 資料表的預期寫入流量設定閾值，並以可接受的限流層級為基準。務必要監控是否佈建不足，並且不會導致持續限流。您還可分析歷史資料，尋找應用程式工作負載可接受的限流層級，然後將閾值調整為高於通常可接受的限流層級。應用程式/服務應重試限流請求，因為它們是暫時的。因此，極低的閾值可能會導致警示過於敏感，從而引起不必要的狀態轉換。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

WriteThrottleEvents

尺寸:TableName, GlobalSecondaryIndexName

警示描述：此警示可偵測 DynamoDB 資料表的全域次要索引是否存在大量寫入請求限制。若要對該問題進行故障診斷，請參閱對 Amazon DynamoDB 中的限流問題進行故障診斷。

目的：該警示可偵測 DynamoDB 資料表全域次要索引的寫入請求是否持續限流。持續限流寫入請求可能會對工作負載寫入操作產生負面影響，並降低系統的整體效率。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：根據 DynamoDB 資料表的預期寫入流量設定閾值，並以可接受的限流層級為基準。務必要監控是否佈建不足，並且不會導致持續限流。您還可分析歷史資料，尋找應用程式工作負載可接受的限流層級，然後將閾值調整為高於通常可接受的限流層級。應用程式/服務應重試限流請求，因為它們是暫時的。因此，極低的值可能會導致警示過於敏感，從而引起不必要的狀態轉換。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

Amazon EBS

VolumeStalledIOCHECK

尺寸:VolumeId, InstanceId

警示說明：此警示可協助您監控 Amazon EBS 磁碟區的 IO 效能。此檢查可偵測 Amazon EBS 基礎設施的基本問題，例如 Amazon EBS 磁碟區底層儲存子系統上的硬體或軟體問題、影響 Amazon EBS 磁碟區從 Amazon EC2 執行個體可達性的實體主機上的硬體問題，以及偵測執行個體和 Amazon EBS 磁碟區之間的連線問題。如果「停止的 IO 檢查」失敗，您可以等待 AWS 解決問題，也可以採取動作，例如更換受影響的磁碟區，或停止並重新啟動磁碟區所連接的執行個體。在大多數情況下，當此指標失敗時，Amazon EBS 會在幾分鐘內自動診斷並恢復您的磁碟區。

意圖：此警示可偵測 Amazon EBS 磁碟區的狀態，以判斷這些磁碟區何時受損且無法完成 I/O 作業。

統計資料：最大值

建議的閾值：1.0

閾值對正：當狀態檢查失敗時，此指標的值為 1。設定閾值，以便每當狀態檢查失敗時，警示都會處於 ALARM 狀態。

期間：60

警示資料點數目：10

評估期：10

比較運算子：GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Amazon EC2

CPUUtilization

尺寸:InstanceId

警示描述：此警示有助於監控 EC2 執行個體的 CPU 使用率。視乎應用程式而定，持續的高使用率層級可能是正常的。但是，如果效能降級，且應用程式不受磁碟 I/O、記憶體或網路資源的限制，

則 CPU 上限可能表示資源瓶頸或應用程式效能問題。高 CPU 使用率可能表示需要升級至 CPU 更密集的執行個體。如果已啟用詳細監控，您可將期間變更為 60 秒，而不是 300 秒。如需詳細資訊，請參閱啟用或關閉詳細監控您的執行個體。

目的：此警示用於偵測高 CPU 使用率。

統計資料：平均值

建議的閾值：80.0

閾值對正：通常，您可將 CPU 使用率的閾值設定為 70-80%。但是，您可根據可接受的效能層級和工作負載特性來調整此值。對於某些系統來說，持續的高 CPU 使用率可能是正常的，並非表示存在問題，而對於其他系統來說，則可能是需要關注的問題。分析歷史 CPU 使用率資料以識別用量、尋找系統可接受的 CPU 使用率，並相應地設定閾值。

期間：300

警示資料點數目：3

評估期：3

比較運算子：GREATER_THAN_THRESHOLD

StatusCheckFailed

尺寸:InstanceId

警示描述：此警示有助於同時監控系統狀態檢查和執行個體狀態檢查。如果任一類型的狀態檢查失敗，則此警示應處於 ALARM 狀態。

目的：此警示用於偵測執行個體的基本問題，包括系統狀態檢查失敗和執行個體狀態檢查失敗。

統計資料：最大值

建議的閾值：1.0

閾值對正：當狀態檢查失敗時，此指標的值為 1。設定閾值，以便每當狀態檢查失敗時，警示都會處於 ALARM 狀態。

期間：300

警示資料點數目：2

評估期：2

比較運算子：GREATER_THAN_OR_EQUAL_TO_THRESHOLD

StatusCheckFailed附件 (_)

尺寸:InstanceId

警示說明：此警示可協助您監控連接至執行個體的 Amazon EBS 磁碟區是否可連線，以及是否能夠完成 I/O 操作。此狀態檢查可偵測運算或 Amazon EBS 基礎設施的基本問題，如下所示：

- Amazon EBS 磁碟區底層儲存子系統上的硬體或軟體問題
- 影響 Amazon EBS 磁碟區可連接性的實體主機上的硬體問題
- 執行個體和 Amazon EBS 磁碟區之間的連線問題

當連接的 EBS 狀態檢查失敗時，您可以等待 Amazon 解決問題，也可以採取動作，例如更換受影響的磁碟區或停止並重新啟動執行個體。

意圖：此警示用於偵測連接至執行個體的無法連接的 Amazon EBS 磁碟區。這些可能會導致 I/O 操作失敗。

統計資料：最大值

建議的閾值：1.0

閾值對正：當狀態檢查失敗時，此指標的值為 1。設定閾值，以便每當狀態檢查失敗時，警示都會處於 ALARM 狀態。

期間：60

警示資料點數目：10

評估期：10

比較運算子：GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Amazon ElastiCache

CPUUtilization

尺寸:CacheClusterId, CacheNodeId

警示說明：此警示可協助監督整個 ElastiCache 執行處理的 CPU 使用率，包括資料庫引擎處理序和執行處理上執行的其他處理作業。 AWS 彈性疼支持兩種引擎類型：內存緩存和 Redis。當您

在 Memcached 節點上達到高 CPU 使用率時，您應考慮擴展執行個體類型或新增快取節點。針對 Redis，如果您的主要工作負載來自讀取請求，則應考慮將更多僅供讀取複本新增至快取叢集。如果您的主要工作負載來自寫入請求，則應考慮以下方面：在叢集模式下執行時，新增更多碎片以將工作負載分散到更多主節點；或在非叢集模式下執行 Redis 時，則擴展執行個體類型。

意圖：此警報用於檢測 ElastiCache 主機的 CPU 使用率高。能夠全面檢視整個執行個體 (包括非引擎程序) 的 CPU 使用率非常有幫助。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：將閾值設定為反映應用程式臨界 CPU 使用率層級的百分比。若是 Memcached，引擎可使用最多 num_threads 個核心。若是 Redis，引擎大部分是單一執行緒，但如果適用，可能會使用額外的核心來加速 I/O。在大多數情況下，您可將閾值設定為可用 CPU 的 90% 左右。因為 Redis 為單執行緒，實際閾值應以節點總容量的分數計算。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

CurrConnections

尺寸:CacheClusterId, CacheNodeId

警示描述：此警示可偵測高連線計數，這可能表示負載過重或效能問題。持續增加 `CurrConnections` 可能會導致 65,000 個可用連線用盡。這可能表示應用程式端連線關閉不正確，並在伺服器端建立連線。您應考慮使用連線集區或閒置連線逾時，來限制與叢集建立連線的數目，或者針對 Redis，考慮調整叢集上的 tcp-keepalive，以偵測並終止潛在的失效對等。

意圖：警示可協助您識別可能影響 ElastiCache 叢集效能和穩定性的高連線計數。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：此警示的建議閾值很大程度上取決於叢集可接受的連線範圍。檢閱 ElastiCache 叢集的容量和預期的工作負載，並分析一般使用期間的歷史連線計數以建立基準，然後選取臨界值。請記住，每個節點可支援最多 65,000 個並行連線。

期間：60

警示資料點數目：10

評估期：10

比較運算子：GREATER_THAN_THRESHOLD

DatabaseMemoryUsagePercentage

尺寸:CacheClusterId

警示描述：此警示可協助您監控叢集的記憶體使用率。當您的 `DatabaseMemoryUsagePercentage` 達到 100% 時，會觸發 Redis maxmemory 政策，而且可能會根據選取的政策發生移出。如果快取中沒有物件符合移出政策，寫入操作會失敗。某些工作負載預期或依賴於移出，但如果沒有，您將需要增加叢集的記憶體容量。您可新增更多主節點來擴展叢集，或使用較大的節點類型來擴展叢集。如需詳細資訊，ElastiCache 請參閱 Redis 叢集的縮放。

目的：此警示用於偵測叢集的高記憶體使用率，以免在寫入叢集時出現失敗。如果您的應用程式預期不會移出，知道何時需要縱向擴展叢集會很有幫助。

統計資料：平均值

建議的閾值：視乎您的情況而定

臨界值理由：根據應用程式的記憶體需求和 ElastiCache 叢集的記憶體容量而定，您應該將臨界值設定為反映叢集臨界記憶體使用量層級的百分比。您可使用歷史記憶體用量資料，做為可接受記憶體用量閾值的參考。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

EngineCPUUtilization

尺寸:CacheClusterId

警示說明：此警示可協助監控執行個體內 Redis 引擎 ElastiCache 執行緒的 CPU 使用率。引擎 CPU 較高的常見原因包括長時間執行的命令，這些命令會取用較高的 CPU、具有大量請求、在短

時間內增加新的用戶端連線請求，以及在快取沒有足夠的記憶體來保存新資料時的大量移出。您應該考慮新增更多節點或擴展 ElastiCache 執行個體類型，以擴展 Redis 叢集。

目的：此警示用於偵測 Redis 引擎執行緒的高 CPU 使用率。如果您想要監控資料庫引擎本身的 CPU 使用率，這會很有幫助。

統計資料：平均值

建議的閾值：90.0

閾值對正：將閾值設定為反映應用程式臨界引擎 CPU 使用率層級的百分比。您可使用應用程式和預期工作負載來對叢集進行基準測試，以將 EngineCPUUtilization 和效能做為參考來關聯，然後相應地設定閾值。在大多數情況下，您可將閾值設定為可用 CPU 的 90% 左右。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

ReplicationLag

尺寸:CacheClusterId

警示說明：此警示有助於監視 ElastiCache 叢集的複寫健全狀況。高複寫延遲表示主節點或複本無法跟上複寫的速度。如果您的寫入活動太多，請考慮新增更多主節點來擴展叢集，或使用較大的節點類型來擴展叢集。如需詳細資訊，ElastiCache 請參閱 Redis 叢集的縮放。如果根據讀取請求數量，僅供讀取複本過載，則考慮新增更多僅供讀取複本。

目的：此警示用於偵測主節點上的資料更新與其同步處理複本節點之間的延遲。它有助於確保僅供讀取複本叢集節點的資料一致性。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：根據應用程式的需求和複寫延遲的潛在影響來設定閾值。針對可接受的複寫延遲，您應考慮應用程式的預期寫入速率和網路條件。

期間：60

警示資料點數目：15

評估期：15

比較運算子：GREATER_THAN_THRESHOLD

Amazon EC2 (**AWS/ElasticGPUs**)

顯示卡 ConnectivityCheckFailed

尺寸:InstanceId, 輸出

警示描述：此警示有助於偵測執行個體與 Elastic Graphics 加速器之間的連線失敗。Elastic Graphics 使用執行個體網路將 OpenGL 命令傳送到遠端連接的顯示卡。此外，執行使用 Elastic Graphics 加速器之 OpenGL 應用程式的桌面，通常會使用遠端存取技術存取。區分 OpenGL 轉譯與桌面遠端存取技術相關的效能問題非常重要。若要進一步了解該問題，請參閱調查應用程式效能問題。

目的：此警示用於偵測從執行個體到 Elastic Graphics 加速器的連線問題。

統計資料：最大值

建議的閾值：0.0

閾值對正：閾值 1 表示連線失敗。

期間：300

警示資料點數目：3

評估期：3

比較運算子：GREATER_THAN_THRESHOLD

顯示卡 HealthCheckFailed

尺寸:InstanceId, 輸出

警示描述：此警示可協助您了解 Elastic Graphics 加速器的狀態何時運作狀態不良。如果加速器運作狀態不良，請參閱解決運作狀態不良問題中的故障診斷步驟。

目的：此警示用於偵測 Elastic Graphics 加速器運作狀態是否良好。

統計資料：最大值

建議的閾值：0.0

閾值對正：閾值 1 表示狀態檢查失敗。

期間：300

警示資料點數目：3

評估期：3

比較運算子：GREATER_THAN_THRESHOLD

Amazon ECS

CPUReservation

尺寸:ClusterName

警示描述：此警示可協助您偵測 ECS 叢集的高 CPU 保留。高 CPU 保留可能表示叢集已用盡為任務註冊的 CPU。若要進行故障診斷，您可新增更多容量、擴展叢集，或是設定自動擴展。

目的：該警示用於偵測叢集上任務預留的 CPU 單元總數是否達到為叢集註冊的 CPU 單元總計。這有助於您了解何時縱向擴展叢集。達到叢集的 CPU 單元總計可能會導致任務的 CPU 用盡。如果您已開啟 EC2 容量供應商受管擴展，或者已將 Fargate 與容量供應商建立關聯，則不建議使用此警示。

統計資料：平均值

建議的閾值：90.0

閾值對正：將 CPU 預留的閾值設定為 90%。或者，您可根據叢集特性來選擇較低的值。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

CPUUtilization

尺寸:ClusterName, ServiceName

警示描述：此警示可協助您偵測 ECS 服務的高 CPU 使用率。如果沒有進行中的 ECS 部署，CPU 使用率上限可能表示資源瓶頸或應用程式效能問題。若要進行故障診斷，您可增加 CPU 限制。

目的：此警示用於偵測 ECS 服務的高 CPU 使用率。持續的高 CPU 使用率可能表示資源瓶頸或應用程式效能問題。

統計資料：平均值

建議的閾值：90.0

閾值對正：CPU 使用率的服務指標可能超過 100% 的使用率。但是，建議您監控高 CPU 使用率的指標，以避免影響其他服務。將閾值設定為約 90-95%。建議您更新任務定義來反映實際用量，以避免未來其他服務發生問題。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

MemoryReservation

尺寸:ClusterName

警示描述：此警示可協助您偵測 ECS 叢集的高記憶體保留。高記憶體保留可能表示叢集的資源瓶頸。若要進行故障診斷，請分析服務任務的效能，以查看是否可優化任務的記憶體使用率。此外，您可註冊更多記憶體或設定自動擴展比例。

目的：該警示用於偵測叢集上任務預留的記憶體單元總計是否達到為叢集註冊的記憶體單元總計。這可協助您了解何時縱向擴展叢集。達到叢集的記憶體單元總計可能會導致叢集無法啟動新的任務。如果您已開啟 EC2 容量供應商受管擴展，或者已將 Fargate 與容量供應商建立關聯，不建議使用此警示。

統計資料：平均值

建議的閾值：90.0

閾值對正：將記憶體預留的閾值設定為 90%。您可根據叢集特性，將其調整為較低的值。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

HTTPCode_Target_5XX_Count

尺寸:ClusterName, ServiceName

警示描述：此警示可協助您偵測 ECS 服務的高伺服器端錯誤計數。這可能表示存在導致服務無法發出請求的錯誤。若要進行故障診斷，請檢查應用程式日誌。

目的：此警示用於偵測 ECS 服務的高伺服器端錯誤計數。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：計算平均流量約 5% 的值，並使用此值作為閾值的起點。您可使用 RequestCount 指標尋找平均流量。您還可分析歷史資料，以確定應用程式工作負載可接受的錯誤率，然後相應地調整閾值。經常發生的 5XX 錯誤需要觸發警示。但是，將閾值設定為極低的值可能會導致警示過於敏感。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

TargetResponseTime

尺寸:ClusterName, ServiceName

警示描述：此警示可協助您偵測 ECS 服務請求的較長目標回應時間。這可能表示存在導致服務無法及時發出請求的問題。若要進行故障診斷，請檢查 CPUUtilization 指標，以查看服務是否用盡 CPU，或檢查服務所依賴的其他下游服務的 CPU 使用率。

目的：此警示用於偵測 ECS 服務請求較常的目標回應時間。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：此警示的建議閾值很大程度上取決於您的使用案例。請檢閱服務目標回應時間的重要性和要求，並分析此指標的歷史行為，以確定合理的閾值層級。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

具有 Container Insights 的 Amazon ECS

EphemeralStorageUtilized

尺寸:ClusterName, ServiceName

警示說明：此警示可協助您偵測 Fargate 叢集使用的高暫時性儲存。如果暫時性儲存持續很高，您可以檢查暫時性儲存的使用量並增加暫時性儲存空間。

意圖：此警示用於偵測 Fargate 叢集的較高暫時性儲存空間使用量。使用持續的高暫時性儲存可能表示磁碟已滿，並可能導致容器故障。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：將閾值設定為臨時儲存大小的 90%。可以根據 Fargate 叢集的可接受暫時性儲存使用率來調整此值。對於某些系統，持續使用高暫時性儲存可能是正常的，而對於其他系統，則可能會導致容器故障。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

RunningTaskCount

尺寸:ClusterName, ServiceName

警示說明：此警示可協助您偵測 ECS 服務的執行任務計數不足。如果執行中的任務計數太低，可能表示應用程式無法處理服務負載，並可能導致效能問題。如果沒有執行中的任務，則 Amazon ECS 服務可能無法使用，或者可能發生部署問題。

意圖：此警示用於檢測正在執行的任務數量是否過低。持續的低執行任務計數可能表示 ECS 服務部署或效能問題。

統計資料：平均值

建議的閾值：0.0

閾值對正：您可以根據 ECS 服務的最小執行中任務計數來調整閾值。如果執行中的任務計數為 0，則 Amazon ECS 服務將無法使用。

期間：60

警示資料點數目：5

評估期：5

比較運算子：LESS_THAN_OR_EQUAL_TO_THRESHOLD

instance_filesystem_utilization

尺寸:InstanceId ContainerInstanceId, ClusterName

警示描述：此警示可協助您偵測 ECS 叢集的高檔案系統使用率。如果檔案系統使用率持續很高，請檢查磁碟使用率。

意圖：此警示用於偵測 Amazon ECS 叢集的高檔案系統使用率。持續的高檔案系統使用率可能表示資源瓶頸或應用程式效能問題，而且可能無法執行新任務。

統計資料：平均值

建議的閾值：90.0

閾值對正：可將檔案系統使用率的閾值設定為 90-95%。您可以根據 Amazon ECS 叢集的可接受檔案系統容量級別來調整此值。對於某些系統而言，持續的高檔案系統使用率可能是正常的，並不表示有問題，而對其他系統而言，這可能是令人擔憂的原因，並且可能導致效能問題並阻止執行新任務。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

Amazon EFS

PercentIOLimit

尺寸:FileSystemId

警示描述：此警示有助於確保工作負載維持在檔案系統可用的 I/O 限制內。如果指標持續達到 I/O 限制，考慮將應用程式移至使用「I/O 效能上限」模式的檔案系統。如需進行故障診斷，請檢查連線至檔案系統的用戶端，以及對檔案系統限流的用戶端應用程式。

目的：此警示用於偵測檔案系統有多接近一般用途效能模式的 I/O 限制。持續較高的 I/O 百分比可能是檔案系統無法根據 I/O 請求進行足夠擴展的指標，而且檔案系統可能會成為使用檔案系統之應用程式的資源瓶頸。

統計資料：平均值

建議的閾值：100.0

閾值對正：當檔案系統達到其 I/O 限制時，可能會減慢讀取和寫入請求的回應。因此，建議您監控該指標，以免影響使用該檔案系統的應用程式。閾值可設定為 100% 左右。但是，可根據檔案系統特性將此值調整為較低的值。

期間：60

警示資料點數目：15

評估期：15

比較運算子：GREATER_THAN_OR_EQUAL_TO_THRESHOLD

BurstCreditBalance

尺寸:FileSystemId

警示描述：此警示有助於確保檔案系統用量有可用的爆量額度餘額。若沒有可用的爆量額度，由於輸送量較低，應用程式對檔案系統的存取將會受到限制。如果指標持續降至 0，考慮將輸送量模式變更為彈性或佈建輸送量模式。

目的：此警示用於偵測檔案系統的低爆量額度餘額。持續較低的爆量額度餘額可能是輸送量減慢和 I/O 延遲增加的指標。

統計資料：平均值

建議的閾值：0.0

臨界值理由：當檔案系統用完了突發積分，而且即使基準輸送量率較低，EFS 仍會繼續為所有檔案系統提供 1 MiBps 的計量輸送量。但是，建議監控指標是否為低爆量額度餘額，以避免檔案系統成為應用程式的資源瓶頸。閾值可設定為 0 位元組左右。

期間：60

警示資料點數目：15

評估期：15

比較運算子：LESS_THAN_OR_EQUAL_TO_THRESHOLD

具有 Container Insights 的 Amazon EKS

node_cpu_utilization

尺寸:ClusterName

警示描述：此警示有助於偵測 EKS 叢集工作節點中的高 CPU 使用率。如果使用率持續較高，則可能表示需要將工作節點取代為具有更大 CPU 或需要水平擴展系統的執行個體。

目的：此警示有助於監控 EKS 叢集中工作節點的 CPU 使用率，使系統效能不會降級。

統計資料：最大值

建議的閾值：80.0

閾值對正：建議將閾值設定為小於或等於 80%，以便在系統開始看到影響之前有足夠的時間對問題進行偵錯。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

node_filesystem_utilization

尺寸:ClusterName

警示描述：此警示有助於偵測 EKS 叢集中工作節點的高檔案系統使用率。如果使用率持續較高，則您可能需要更新工作節點以擁有更大的磁碟區，或者可能需要水平擴展。

目的：此警示有助於監控 EKS 叢集中工作節點的檔案系統使用率。如果使用率達到 100%，則可能導致應用程式失敗、磁碟 I/O 瓶頸、Pod 移出或節點完全無回應。

統計資料：最大值

建議的閾值：視乎您的情況而定

閾值對正：如果磁碟壓力足夠 (表示磁碟將變滿)，則節點會標示為運作狀態不佳，並從節點移出 Pod。如果可用的檔案系統低於 kubelet 上設定的移出閾值，則會移出節點上具有磁碟壓力的 Pod。設定警示閾值，以便在從叢集中移出節點之前有足夠的回應時間。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

node_memory_utilization

尺寸:ClusterName

警示描述：此警示有助於偵測 EKS 叢集工作節點中的高記憶體使用率。如果使用率持續較高，可能表示需要擴展 Pod 複本的數目或優化您的應用程式。

目的：此警示有助於監控 EKS 叢集中工作節點的記憶體使用率，使系統效能不會降級。

統計資料：最大值

建議的閾值：80.0

閾值對正：建議將閾值設定為小於或等於 80%，以便在系統開始看到影響之前有足夠的時間對問題進行偵錯。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

pod_cpu_utilization_over_pod_limit

維度：ClusterName，命名空間，服務

警示描述：此警示有助於偵測 EKS 叢集 Pod 中的高 CPU 使用率。如果使用率持續較高，可能表示需要增加受影響 Pod 的 CPU 限制。

目的：此警示有助於監控屬於 EKS 叢集中 Kubernetes 服務的 Pod 的 CPU 使用率，以便您快速識別服務的 Pod 取用的 CPU 是否高於預期。

統計資料：最大值

建議的閾值：80.0

閾值對正：建議將閾值設定為小於或等於 80%，以便在系統開始看到影響之前有足夠的時間對問題進行偵錯。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

pod_memory_utilization_over_pod_limit

維度：ClusterName，命名空間，服務

警示描述：此警示有助於偵測 EKS 叢集 Pod 中的高記憶體使用率。如果使用率持續較高，可能表示需要增加受影響 Pod 的記憶體限制。

目的：此警示有助於監控 EKS 叢集中 Pod 的記憶體使用率，使系統效能不會降級。

統計資料：最大值

建議的閾值：80.0

閾值對正：建議將閾值設定為小於或等於 80%，以便在系統開始看到影響之前有足夠的時間對問題進行偵錯。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

Amazon Kinesis Data Streams

GetRecords.IteratorAgeMilliseconds

尺寸:StreamName

警示描述：此警示可偵測反覆運算器的存留期上限是否太長。針對即時資料處理應用程式，請根據延遲容容錯值設定資料保留。這通常在幾分鐘內即可完成。針對處理歷史資料的應用程式，請使用此指標來監控追趕速度。防止資料遺失的快速解決方案是，在診斷問題時增加保留期。您還可增加取用者應用程式中處理記錄的工作者數目。反覆運算器逐步存留期最常見的原因是實體資源不足，或者記錄處理邏輯沒有隨著串流輸送量的增加而進行擴展。如需詳細資訊，請參閱 連結。

目的：此警示用於偵測串流中的資料是否因預留時間太長或記錄處理太慢而到期。它可協助您避免在達到 100% 串流保留時間之後遺失資料。

統計資料：最大值

建議的閾值：視乎您的情況而定

閾值對正：此警示的建議閾值很大程度上取決於串流保留期間和記錄處理延遲的容錯值。檢閱您的需求並分析歷史趨勢，然後將閾值設定為代表處理嚴重延遲的毫秒數。如果反覆運算器的存留期超過保留期的 50% (預設為 24 小時，最多可設定為 365 天)，會有因記錄過期而遺失資料的風險。您可監控指標，以確保沒有任何碎片達到此限制。

期間：60

警示資料點數目：15

評估期：15

比較運算子：GREATER_THAN_THRESHOLD

GetRecords. 成功

尺寸:StreamName

警示描述：每當您的取用者成功從串流中讀取資料時，此指標會增加。GetRecords 在擲出例外狀況時，不會傳回任何資料。最常見的例外狀況是 ProvisionedThroughputExceededException，因為串流的請求速率太高，或是因為在指定秒數已經提供可用的輸送量。請減少請求的頻率或大小。如需詳細資訊，請參閱《Amazon Kinesis Data Streams 開發人員指南》中的串流限制，以及 AWS中的錯誤重試和指數退避。

目的：此警示可偵測取用者從串流中擷取記錄是否失敗。藉由在此指標上設定警示，您可主動偵測任何與資料取用相關的問題，例如提高錯誤率或拒絕成功擷取。這讓您能夠及時執行動作，以解決潛在問題並保持順從的資料處理管道。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：視乎從串流擷取記錄的重要性，根據應用程式失敗記錄的容錯值來設定閾值。閾值應與成功操作百分比相對應。您可以使用歷史 GetRecords 測量結果資料作為可接受失敗率的參考。您還應在設定閾值時考慮重試，因為失敗的記錄可以重試。這有助於防止暫時性尖峰觸發不必要的警示。

期間：60

警示資料點數目：5

評估期：5

比較運算子：LESS_THAN_THRESHOLD

PutRecord. 成功

尺寸:StreamName

警示描述：此警示可偵測失敗的 PutRecord 操作次數何時違反閾值。調查資料生產者日誌，以尋找失敗的根本原因。最常見的原因是，碎片上的輸送量佈建不足，從而導致 ProvisionedThroughputExceededException。之所以發生這種情況，是因為串流的請求速率太高，或是嘗試擷取到碎片中的輸送量太高。請減少請求的頻率或大小。如需詳細資訊，請參閱. AWS

目的：此警示可偵測將記錄擷取到串流中是否失敗。它可協助您識別將資料寫入串流的問題。藉由在此指標上設定警示，您可主動偵測生產者在將資料發布到串流時的任何問題，例如提高錯誤率或減少成功發布的記錄。這讓您能夠及時執行動作，以解決潛在問題並確保可靠的資料擷取程序。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：根據資料擷取和處理服務的重要性，根據應用程式失敗記錄的容錯值來設定閾值。閾值應與成功操作百分比相對應。您可以使用歷史 PutRecord 測量結果資料作為可接受失敗率的參考。您還應在設定閾值時考慮重試，因為失敗的記錄可以重試。

期間：60

警示資料點數目：5

評估期：5

比較運算子：LESS_THAN_THRESHOLD

PutRecords.FailedRecords

尺寸:StreamName

警示描述：此警示可偵測失敗的 `PutRecords` 何時超出閾值。Kinesis Data Streams 會嘗試處理每個 `PutRecords` 請求中的所有記錄，但單一記錄失敗不會停止處理後續記錄。這些失敗的主要原因是，超出串流或個別碎片的輸送量。常見的原因是，流量尖峰和網路延遲會導致記錄到達串流不均勻。您應偵測未成功處理的記錄，並在後續呼叫中重試這些記錄。如需詳細資訊，請參閱使 PutRecords用時處理失敗。

目的：此警示可偵測在使用批次操作將記錄置於串流中時是否出現持續失敗。藉由在此指標上設定警示，您可主動偵測失敗記錄的增加，從而能夠及時執行動作來解決潛在問題，並確保順暢、可靠的資料擷取程序。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：將閾值設定為反映應用程式失敗記錄容錯值的失敗記錄數目。您可使用歷史資料，做為可接受失敗值的參考。您也應該在設定臨界值時考慮重試，因為失敗的記錄可以在後續 PutRecords 呼叫中重試。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

ReadProvisionedThroughputExceeded

尺寸:StreamName

警示描述：警示可追蹤導致讀取輸送容量限流的記錄數目。如果您發現持續受到限流，應考慮在串流中新增更多碎片，以增加佈建讀取輸送量。如果有多個取用者應用程式在串流中執行，並且共用 GetRecords 限制，建議您透過強化廣發功能來註冊新的取用者應用程式。如果新增更多碎片並不會降低限流數目，則可能相較於其他碎片，您可能正在讀取一個「熱」碎片。啟用增強型監控，尋找「熱」碎片，然後將其分割。

目的：此警示可偵測取用者在超過佈建讀取輸送量時 (由您擁有的碎片數目決定) 是否受到限流。在此情況下，您將無法從串流中讀取，並且串流可開始備份。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：通常限流請求可重試，因此將閾值設定為零會使警示過於敏感。但是，持續限流可能會影響從串流中讀取，並且應會觸發警示。根據應用程式的限流請求，將閾值設定為百分比，然後重試組態。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

SubscribeToShardEvent.MillisBehindLatest

尺寸:StreamName, ConsumerName

警示描述：此警示可偵測應用程式中的記錄處理延遲何時違反閾值。暫時性問題 (例如下游應用程式的 API 操作失敗) 可能會導指標突增。您應調查其是否持續發生。一個常見的原因是，取用者處理記錄的速度不夠快，因為隨著串流輸送量的增加，實體資源不足或是未擴展記錄處理邏輯。封鎖關鍵路徑中的呼叫，通常是記錄處理速度變慢的原因。您可增加碎片數目以提高並行處理程度。您還應確認基礎處理節點在尖峰需求期間擁有足夠的實體資源。

目的：此警示可偵測串流碎片事件訂閱中的延遲。這表示處理延遲，並且可協助識別取用者應用程式效能或整體串流運作狀態的潛在問題。若處理延遲變得顯著，您應調查並解決任何瓶頸或取用者應用程式效率問題，以確保即時處理資料，以及最大限度地減少資料積壓。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：此警示的建議閾值很大程度上取決於應用程式的延遲容錯值。檢閱應用程式需求並分析歷史趨勢，然後相應地選取閾值。 SubscribeToShard 通話成功時，您的消費者開始透過持續連線接收 SubscribeToShardEvent 事件最多 5 分鐘，在此之後，如果您想要繼續接收記錄，則需要 SubscribeToShard 再次呼叫以續訂訂閱。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

WriteProvisionedThroughputExceeded

尺寸:StreamName

警示描述：此警示可偵測導致寫入輸送容量限流的記錄數目何時達到閾值。若您的生產者超過佈建寫入輸送量 (由您擁有的碎片數目確定) 時，則會對其限流，並且您將無法將記錄放入串流。為了解決持續限流問題，您應考慮將碎片新增至串流。這會提高您的佈建寫入輸送量，並防止未來限流。擷取記錄時，您還應考慮選擇磁碟分割區索引鍵。隨機分割區索引鍵是首選，因為在可能的情況下它會將記錄平均分散到串流碎片中。

目的：此警示可偵測您的生產者是否因串流或碎片限流而被拒絕寫入記錄。如果您的串流處於佈建模式，則設定此警示可協助您在資料串流達到限制時主動執行動作，從而讓您優化佈建容量或執行適當的擴展動作，以避免資料遺失並確保順暢的資料處理。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：通常限流請求可重試，因此將閾值設定為零會使警示過於敏感。但是，持續限流可能會影響串流的寫入，因此您應設定警示閾值來偵測此問題。根據應用程式的限流請求，將閾值設定為百分比，然後重試組態。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

Lambda

ClaimedAccountConcurrency

維度：無

警示說明：此警示有助於監控 Lambda 函數的並行是否已接近帳戶的區域層級並行限制。如果函數達到並行限制，則會開始對函數限流。您可執行下列動作以避免限流。

1. 要求此區域的並行增加。
2. 識別並減少任何未使用的保留並行或佈建並行。
3. 識別功能中的效能問題，以提高處理速度，從而提高輸送量。
4. 增加函數的批處理大小，以便每次函數調用處理更多消息。

意圖：此警示可以主動偵測 Lambda 函數的並行性是否接近帳戶的區域層級並行配額，以便您可以採取行動。如果ClaimedAccountConcurrency達到帳戶的區域層級並行配額，則會限制函數。如果您使用的是保留並行 (RC) 或佈建並行 (PC)，此警示可讓您更清楚地瞭解並行使用率，而不是開啟的警示。ConcurrentExecutions

統計資料：最大值

建議的閾值：視乎您的情況而定

閾值理由：您應該計算為區域中帳戶設定的並行配額約 90% 的值，並使用結果作為閾值。根據預設，您的帳戶設有一個區域中所有函數共計 1,000 的並行配額。不過，您應該從「Service Quotas」控制面板檢查帳戶的配額。

期間：60

警示資料點數目：10

評估期：10

比較運算子：GREATER_THAN_THRESHOLD

錯誤

尺寸:FunctionName

警示描述：此警示可偵測高錯誤計數。錯誤包括程式碼擲回的例外狀況，以及 Lambda 執行期擲回的例外狀況。您可檢查與該函數相關的日誌以診斷問題。

目的：警示有助於偵測函數調用中的高錯誤計數。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：將閾值設定為大於零的數字。準確值可能取決於應用程式中的容錯值。了解函數正在處理的調用重要性。針對某些應用程式，任何錯誤可能都是不可接受的，而其他應用程式可能會允許一定的錯誤餘度。

期間：60

警示資料點數目：3

評估期：3

比較運算子：GREATER_THAN_THRESHOLD

限流

尺寸:FunctionName

警示描述：此警示可偵測大量限流調用請求。若沒有可用於縱向擴展的並行，就會發生限流。有幾種方法可解決此問題。1) 要求並行增加本地區的 Sup AWS port。2) 識別函數中的效能問題，以提高處理速度，從而改善輸送量。3) 增加函數的批次大小，以便每次函數調用處理更多訊息。

目的：警示有助於偵測 Lambda 函數的大量限流調用請求。務必要了解請求是否因限流而持續遭拒，以及是否需要改善 Lambda 函數效能或增加並行容量來避免持續限流。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：將閾值設定為大於零的數字。閾值的準確值可取決於應用程式容錯值。根據函數的用量和擴展需求來設定閾值。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_OR_EQUAL_TO_THRESHOLD

Duration (持續時間)

尺寸:FunctionName

警示描述：此警示可偵測 Lambda 函數處理事件較常的持續時間。較長的持續時間可能是因為函數程式碼的變更使得函數執行時間延長，或者函數的相依項需要更長的處理時間。

目的：此警示可偵測 Lambda 函數的長時間執行持續時間。較長的執行期持續時間表示函數調用時間延長，而且如果 Lambda 處理的事件數目增多，也會影響調用的並行容量。務必要了解 Lambda 函數是否持續花費比預期更長的執行時間。

統計資料：p90

建議的閾值：視乎您的情況而定

閾值對正：持續時間閾值取決於應用程式和工作負載以及您的效能需求。針對高效能需求，將閾值設定為較短的時間，以查看函數是否符合預期。您還可分析持續時間指標的歷史資料，以查看花費的時間是否符合函數的效能預期，然後將閾值設定為比歷史平均值更長的時間。確定將閾值設定為低於設定的函數逾時值。

期間：60

警示資料點數目：15

評估期：15

比較運算子：GREATER_THAN_THRESHOLD

ConcurrentExecutions

尺寸:FunctionName

警示描述：此警示有助於監控函數的並行性是否接近您帳戶的區域層級並行限制。如果函數達到並行限制，則會開始對函數限流。您可執行下列動作以避免限流。

1. 要求此區域的並行增加。
2. 識別功能中的效能問題，以提高處理速度，從而提高輸送量。
3. 增加函數的批處理大小，以便每次函數調用處理更多消息。

若要更好地瞭解保留並行使用率和佈建的並行使用率，請改為在新指標`ClaimedAccountConcurrency`上設定警示。

目的：此警示可主動偵測函數的並行性是否接近您帳戶的區域級並行配額，以便您可對其執行動作。如果函數達到帳戶的區域層級並行配額，則會對該函數限流。

統計資料：最大值

建議的閾值：視乎您的情況而定

閾值對正：將閾值設定為在區域帳戶中設定的約 90% 的並行配額。根據預設，您的帳戶設有一個區域中所有函數共計 1,000 的並行配額。但是，您可以檢查帳戶的配額，因為它可以通過聯繫 AWS 支持來增加。

期間：60

警示資料點數目：10

評估期：10

比較運算子：GREATER_THAN_THRESHOLD

Lambda Insights

建議針對下列 Lambda Insights 指標設定最佳實務警示。

memory_utilization

維度：function_name

警示描述：此警示用於偵測 Lambda 函數的記憶體使用率是否接近設定的限制。如需進行故障診斷，您可嘗試 1) 優化您的程式碼。2) 準確估計記憶體需求，藉此來正確調整記憶體配置大小。您可參考 Lambda 功能調校的相同內容。3) 使用連線集區。如需了解 RDS 資料庫的連線集區，請參閱搭配 Lambda 使用 Amazon RDS Proxy。4) 您還可考慮設計函數，以避免在調用之間記憶體中存放大量資料。

目的：此警示用於偵測 Lambda 函數的記憶體使用率是否接近設定的限制。

統計資料：平均值

建議閾值：90.0

閾值對正：將閾值設定為 90%，以便在記憶體使用率超過所配置記憶體的 90% 時收到警示。如果您為工作負載的記憶體使用率感到擔憂，可將其調整為較低的值。您還可檢查此指標的歷史資料，並相應地設定閾值。

期間：60

警示資料點數目：10

評估期：10

ComparisonOperator：大於 _ 臨界值

Amazon VPC (**AWS/NATGateway**)

ErrorPortAllocation

尺寸:NatGatewayId

警示描述：此警示有助於偵測 NAT Gateway 何時無法將連接埠配置給新的連線。若要解決此問題，請參閱解決 NAT Gateway 上的連接埠配置錯誤。

目的：此警示用於偵測 NAT Gateway 是否無法配置來源連接埠。

統計資料：總和

建議的閾值：0.0

閾值理由：如果的值大 ErrorPortAllocation 於零，則表示透過 NatGateway 開啟與單一熱門目的地的同時連線過多。

期間：60

警示資料點數目：15

評估期：15

比較運算子：GREATER_THAN_THRESHOLD

PacketsDropCount

尺寸:NatGatewayId

警示描述：此警示有助於偵測 NAT Gateway 何時捨棄封包。這可能是因為 NAT 閘道發生問題，因此請檢查AWS 服務健康狀態儀表板以瞭解您所在地區的 AWS NAT 閘道狀態。這可協助您關聯與使用 NAT Gateway 的流量相關的網路問題。

目的：此警示用於偵測 NAT Gateway 是否會捨棄封包。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：您應計算 NAT Gateway 上總流量 0.01% 的值，並使用該結果作為閾值。使用 NAT Gateway 上的流量歷史資料來確定閾值。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

AWS 私人連結 (**AWS/PrivateLinkEndpoints**)

PacketsDropped

維度：VPC ID、VPC 端點 ID、端點類型、子網路 ID、服務名稱

警示描述：此警示可監控端點捨棄的封包數目，藉此來協助偵測端點或端點服務是否運作狀態不良。請注意，封包大於 8,500 位元組且到達 VPC 端點的封包會被捨棄。如需進行故障診斷，請參閱介面 VPC 端點與端點服務之間的連線問題。

目的：此警示用於偵測端點或端點服務是否運作狀態不良。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：根據使用案例設定閾值。如果您想要了解端點或端點服務的運作狀態不良情況，應將閾值設定為低，以便在大量資料遺失之前有機會修復問題。您可使用歷史資料來了解捨棄封包的容錯值，並相應地設定閾值。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

AWS 私人連結 (**AWS/PrivateLinkServices**)

RstPacketsSent

維度：服務 ID、Load Balancer Arn、Az

警示描述：此警示可協助您根據傳送至端點的重設封包數目，偵測端點服務運作狀態不良的目標。當您對服務的用戶偵錯連線錯誤時，您可以驗證服務是否正在重設與 RstPacketsSent 指標的連線，或是網路路徑上是否有其他失敗。

目的：此警示用於偵測端點服務運作狀態不良的目標。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：閾值取決於使用案例。如果您的使用案例可容忍運作狀態不良的目標，您可將閾值設定為較高。如果使用案例無法容忍運作狀態不良的目標，您可將閾值設定得極低。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

Amazon RDS

CPUUtilization

尺寸:分貝 InstanceIdentifier

警示描述：此警示有助於監控一致的高 CPU 使用率。CPU 使用率會測量非閒置時間。請考慮使用 Enhanced Monitoring 或 Performance Insights 來檢閱 MariaDB、MySQL、Oracle 以及 PostgreSQL 的哪個等待時間正在耗用大部分 CPU 時間 (`guest`、`irq`、`wait`、`nice` 等)。然後評估哪些查詢消耗的 CPU 量最高。如果無法調整工作負載，請考慮移至較大的資料庫執行個體類別。

意圖：此警示用於檢測一致的高 CPU 使用率，以防止非常高的回應時間和逾時。如果要檢查 CPU 使用率的微爆量，可以設定較低的警示評估時間。

統計資料：平均值

建議的閾值：90.0

閾值對正：CPU 消耗的隨機峰值可能不會影響資料庫效能，但持續的高 CPU 使用率可能會阻礙即將到來的資料庫請求。視整體資料庫工作負載而定，RDS/Aurora 執行個體的高 CPU 使用率可能會降低整體效能。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

DatabaseConnections

尺寸:分貝 InstanceIdentifier

警示說明：此警示會偵測大量連線。檢閱現有連線，並終止處於「休眠」狀態或未正確關閉的連線。請考慮使用連線集區來限制新連線的數目。或者，增加資料庫執行個體大小，以使用具有更多記憶體的類別，以及較高的 `max_connections` 預設值，或在 RDS、Aurora MySQL 以及 PostgreSQL 中增加目前類別的 `max_connections` 值 (如果它支援您的工作負載)。

意圖：當達到資料庫連線數量上限時，此警示可協助防止被拒絕的連線。如果您經常變更資料庫執行個體類別，則不建議使用此警示，因為這樣做會變更記憶體和預設連線數目上限。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：允許的連線數目取決於資料庫執行個體類別的大小，以及與程序/連線相關的資料庫引擎特定參數。應該計算資料庫連線數目上限 90-95% 之間的值，並使用該結果作為閾值。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

EBS 百分比 ByteBalance

尺寸:分貝 InstanceIdentifier

警示說明：此警示有助於監控剩餘輸送量額度的低百分比。如需疑難排解，請檢查 RDS 中的延遲問題。

意圖：此警示用於偵測爆量儲存貯體中剩餘輸送量額度的低百分比。低位元組平衡百分比可能會造成輸送量瓶頸問題。不建議 Aurora PostgreSQL 執行個體使用此警示。

統計資料：平均值

建議閾值：10.0

閾值對正：低於 10% 的輸送量額度餘額被認為很差，應該相應地設定閾值。如果應用程式可以容忍較低的工作負載輸送量，也可以設定較低的閾值。

期間：60

警示資料點數目：3

評估期：3

比較運算子：LESS_THAN_THRESHOLD

EBSIOBalance%

尺寸:分貝 InstanceIdentifier

警示說明：此警示有助於監控剩餘 IOPS 額度的低百分比。如需疑難排解，請參閱 RDS 中的延遲問題。

意圖：此警示用於偵測爆量儲存貯體中剩餘 I/O 額度的低百分比。低 IOPS 平衡百分比可能會造成 IOPS 瓶頸問題。不建議 Aurora 執行個體使用此警示。

統計資料：平均值

建議閾值：10.0

閾值對正：低於 10% 的 IOPS 額度餘額被認為很差，可相應地設定閾值。如果應用程式可以容忍較低的工作負載 IOPS，也可以設定較低的閾值。

期間：60

警示資料點數目：3

評估期：3

比較運算子：LESS_THAN_THRESHOLD

FreeableMemory

尺寸:分貝 InstanceIdentifier

警示說明：此警示有助於監控低可用記憶體，這可能表示資料庫連線出現峰值，或執行個體可能面臨高記憶體壓力。檢查內存壓力通過監測 CloudWatch 指標 SwapUsage「除了FreeableMemory。如果執行個體記憶體的耗用量經常太高，這表示您應該檢查工作負載或升級執行個體類別。對於 Aurora 讀取器資料庫執行個體，請考慮將其他讀取器資料庫執行個體新增至叢集。如需有關 Aurora 疑難排解的資訊，請參閱可用記憶體問題。

意圖：此警示用於協助防止記憶體不足，從而導致連線被拒絕。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：視工作負載和執行個體類別而定，閾值的不同值可能適當。理想情況下，可用記憶體不應長時間低於總記憶體的 25%。對於 Aurora，可將此閾值設為接近 5%，因為接近 0 的指標意味著資料庫執行個體已盡可能擴展。您可以分析此指標的歷史行為，以判斷合理的閾值級別。

期間：60

警示資料點數目：15

評估期：15

比較運算子：LESS_THAN_THRESHOLD

FreeLocalStorage

尺寸:分貝 InstanceIdentifier

警示說明：此警示有助於監控免費本機儲存空間是否不足。Aurora PostgreSQL 相容版本使用本機儲存體來存放錯誤日誌和臨時檔案。Aurora MySQL 會使用本機儲存體，存放錯誤日誌、一般日誌、慢速查詢日誌、稽核日誌，以及非 InnoDB 暫存資料表。這些本機儲存磁碟區由 Amazon EBS Store 支援，且可透過使用更大的資料庫執行個體類別來擴充。如需疑難排解，請查看 Aurora PostgreSQL 相容版本和 MySQL 相容版本。

意圖：如果您未使用 Aurora Serverless v2 或更高版本，則此警示用於偵測 Aurora 資料庫執行個體是否快達到本機儲存限制。當您將非持續性資料 (例如暫存資料表和日誌檔案) 儲存在本機儲存體時，本機儲存體可能會達到容量。此警示可防止資料庫執行個體在本機儲存空間用完時發生 out-of-space 錯誤。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：您應該根據磁碟區使用量的速度和趨勢計算大約 10%-20% 的可用儲存體容量，然後使用該結果作為閾值，以便在磁碟區達到限制之前主動採取行動。

期間：60

警示資料點數目：5

評估期：5

比較運算子：LESS_THAN_THRESHOLD

FreeStorageSpace

尺寸:分貝 InstanceIdentifier

警示說明：此警示會監測可用儲存空間是否不足。如果經常接近儲存容量限制，請考慮擴充資料庫儲存體。應加入一些緩衝，以應對應用程式中無法預見的需求增長。或者，請考慮啟用 RDS 儲存體自動擴展。此外，請考慮刪除未使用或過時的資料和日誌來釋放更多空間。如需進一步資訊，請參閱 RDS 耗盡儲存體文件和 PostgreSQL 儲存體問題文件。

意圖：此警示有助於防止儲存體已滿問題。當資料庫執行個體耗盡儲存體時，這可防止出現停機。如果已啟用儲存體自動擴展，或經常變更資料庫執行個體的儲存容量，則不建議使用此警示。

統計資料：最小值

建議的閾值：視乎您的情況而定

閾值對正：閾值將取決於目前配置的儲存空間。通常，應該計算已配置儲存空間 10% 的值，並使用該結果作為閾值。

期間：60

警示資料點數目：5

評估期：5

比較運算子：LESS_THAN_THRESHOLD

MaximumUsedTransaction身份證

尺寸:分貝 InstanceIdentifier

警示說明：此警示有助於防止 PostgreSQL 的交易 ID 回卷。請參閱此部落格中的疑難排解步驟，以調查並解決問題。也可以參考此部落格，進一步熟悉自動清空概念、常見問題和最佳實務。

意圖：此警示可用於協助防止 PostgreSQL 的交易 ID 回卷。

統計資料：平均值

建議閾值：1.0E9

閾值對正：將此閾值設定為 10 億，讓您有時間調查問題。預設的 autovacuum_freeze_max_age 值為 2 億。如果最舊的交易存留期為 10 億，則自動清空將此閾值保持在 2 億個交易 ID 目標以下時遇到問題。

期間：60

警示資料點數目：1

評估期：1

比較運算子：GREATER_THAN_THRESHOLD

ReadLatency

尺寸:分貝 InstanceIdentifier

警示說明：此警示有助於監控高讀取延遲。如果儲存體延遲很高，這是因為工作負載超出資源限制。可以檢閱與執行個體相關的 I/O 使用率並配置儲存體組態。請參閱疑難排解 IOPS 瓶頸造成的 Amazon EBS 磁碟區延遲。對於 Aurora，可以切換到具有 I/O 優化儲存組態的執行個體類別。如需指引，請參閱規劃 Aurora 中的 I/O。

意圖：此警示用於偵測高讀取延遲。資料庫磁碟通常具有較低的讀取/寫入延遲，它們可能存在會導致高延遲操作的問題。

統計資料：p90

建議的閾值：視乎您的情況而定

閾值對正：此警示的建議閾值很大程度上取決於您的使用案例。讀取延遲超過 20 毫秒可能是需要調查的原因。如果應用程式具有較高的讀取操作延遲，也可以設定較高的閾值。請檢閱讀取延遲的重要性和要求，並分析此指標的歷史行為，以確定合理的閾值級別。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

ReplicaLag

尺寸:分貝 InstanceIdentifier

警示說明：此警示協助您了解複本落後於主要執行個體的秒數。若在來源資料庫執行個體上未發生使用者交易，PostgreSQL 僅供讀取複本會回報最多五分鐘的複寫延遲。當 ReplicaLag 指標達到 0 時，複本已追上主要資料庫執行個體。如果 ReplicaLag 測量結果傳回 -1，則表示複寫目前未處於作用中狀態。如需 RDS PostgreSQL 的相關指引，請參閱複寫最佳做法和疑難排解ReplicaLag和相關錯誤，請參閱疑難排解。 ReplicaLag

意圖：此警示可以偵測複本延遲，這反映了在主要執行個體出現故障時可能發生資料丟失。如果複本遠遠落後於主要執行個體且主要執行個體失敗，則複本將遺失主要執行個體中的資料。

統計資料：最大值

建議閾值：60.0

閾值對正：通常情況下，可接受的延遲取決於應用程式。建議不要超過 60 秒。

期間：60

警示資料點數目：10

評估期：10

比較運算子：GREATER_THAN_THRESHOLD

WriteLatency

尺寸:分貝 InstanceIdentifier

警示說明：此警示有助於監控高寫入延遲。如果儲存體延遲很高，這是因為工作負載超出資源限制。可以檢閱與執行個體相關的 I/O 使用率並配置儲存體組態。請參閱疑難排解 IOPS 瓶頸造成的

Amazon EBS 磁碟區延遲。對於 Aurora，可以切換到具有 I/O 優化儲存組態的執行個體類別。如需指引，請參閱規劃 Aurora 中的 I/O。

意圖：此警示用於偵測高寫入延遲。雖然資料庫磁碟通常具有低讀取/寫入延遲，但可能會遇到導致高延遲操作的問題。進行監控將確保磁碟延遲與預期一樣低。

統計資料：p90

建議的閾值：視乎您的情況而定

閾值對正：此警示的建議閾值很大程度上取決於您的使用案例。寫入延遲超過 20 毫秒可能是需要調查的原因。如果應用程式具有較高的寫入操作延遲，也可以設定較高的閾值。請檢閱寫入延遲的重要性和要求，並分析此指標的歷史行為，以確定合理的閾值級別。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

DBLoad

尺寸:分貝 InstanceIdentifier

警示說明：此警示有助於監控高資料庫負載。如果程序數目超過 vCPUs 數目，則程序會開始排入佇列。佇列增加時，效能會受到影響。若資料庫負載通常高於最大 vCPU，而主要等待狀態為 CPU，則 CPU 會超過負載。在此情況下，可以在 Performance Insights/Enhanced Monitoring 中監控 CPUUtilization、DBLoadCPU 和排入佇列的任務。您可能會想要節制與執行個體間的連線、以高 CPU 負載來微調任何 SQL 查詢、或者考慮使用較大的執行個體類別。處於任何等待狀態的密集且穩定的執行個體表示可能有您應解決的瓶頸或資源爭用問題。

意圖：此警示用於偵測高資料庫負載。高資料庫負載可能會導致資料庫執行個體發生效能問題。此警示不適用於無伺服器資料庫執行個體。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：最大 vCPU 數值由資料庫執行個體的 vCPU (虛擬 CPU) 核心數目決定。根據最大 vCPU 的不同，閾值有不同的值。理想情況下，資料庫負載不應超過 vCPU 數線。

期間：60

警示資料點數目：15

評估期：15

比較運算子：GREATER_THAN_THRESHOLD

AuroraVolumeBytesLeftTotal

尺寸:分貝 ClusterIdentifier

警示說明：此警示有助於監控低剩餘總容量。當剩餘的總磁碟區達到大小限制時，叢集會報告錯 out-of-space 誤。Aurora 儲存會根據叢集磁碟區中的資料自動擴展，並根據資料庫引擎版本擴充高達至 128 TiB 或 64 TiB。因此，可捨棄不再需要的資料表和資料庫，以減少儲存空間。如需詳細資訊，請參閱儲存體擴展。

意圖：此警示用於偵測 Aurora 叢集與磁碟區大小限制的差距。此警示可防止叢集空間不足時發生 out-of-space 錯誤。建議此警示僅用於 Aurora MySQL。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：應該根據磁碟區使用量增加的速度和趨勢來計算 10%-20% 的實際大小限制，然後使用該結果作為閾值，以便在磁碟區達到限制之前主動採取行動。

期間：60

警示資料點數目：5

評估期：5

比較運算子：LESS_THAN_THRESHOLD

AuroraBinlogReplicaLag

尺寸：DBClusterIdentifier，角色 = 寫入器

警示說明：此警示有助於監控 Aurora 寫入器執行個體複寫的錯誤狀態。如需詳細資訊，請參閱跨 AWS 區域複寫 Aurora MySQL 資料庫叢集。如需疑難排解，請參閱 Aurora MySQL 複寫問題。

意圖：此警示用於偵測寫入器執行個體是否處於錯誤狀態，並且無法複寫來源。建議此警示僅用於 Aurora MySQL。

統計資料：平均值

建議閾值：-1.0

閾值對正：建議使用 -1 作為閾值，因為如果複本處於錯誤狀態，Aurora MySQL 會發佈此值。

期間：60

警示資料點數目：2

評估期：2

比較運算子：LESS_THAN_OR_EQUAL_TO_THRESHOLD

BlockedTransactions

尺寸:分貝 InstanceIdentifier

警示說明：此警示有助於監控 Aurora 資料庫執行個體中較高的封鎖交易計數。被封鎖的交易可以在轉返或遞交中結束。高並行性、交易閒置或長時間執行的交易都可能導致交易遭到封鎖。如需疑難排解，請參閱 Aurora MySQL 文件。

意圖：此警示用於偵測 Aurora 資料庫執行個體中較高的封鎖交易計數，以防止交易轉返和效能降低。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：應使用 `ActiveTransactions` 指標來計算執行個體所有交易的 5%，並使用該結果作為閾值。也可檢閱被封鎖交易的重要性和要求，並分析此指標的歷史行為，以確定合理的閾值級別。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

BufferCacheHitRatio

尺寸:分貝 InstanceIdentifier

警示說明：此警示可協助您監控 Aurora 叢集的持續較低的快取命中率。低命中率表示您對此資料庫執行個體的查詢經常移至磁碟。如需進行疑難排解，請調查工作負載，查看哪些查詢導致了此行為，並參閱資料庫執行個體 RAM 建議文件。

意圖：此警示用於偵測持續較低的快取命中率，以防止 Aurora 執行個體中的持續效能降低。

統計資料：平均值

建議的閾值：80.0

閾值對正：可以將緩衝區快取命中率的閾值設定為 80%。但是，您可根據可接受的效能層級和工作負載特性來調整此值。

期間：60

警示資料點數目：10

評估期：10

比較運算子：LESS_THAN_THRESHOLD

EngineUptime

尺寸：DBClusterIdentifier，角色 = 寫入器

警示說明：此警示有助於監控寫入器資料庫執行個體的低停機時間。寫入器資料庫執行個體可能會因為重新開機、維護、升級或容錯移轉而關閉。當執行時間由於叢集中的容錯移轉而達到 0 並且叢集擁有一個或多個 Aurora 複本時，Aurora 複本會在失敗事件期間提升為主要寫入器執行個體。若要提高資料庫叢集的可用性，可考慮在兩個或更多不同的可用區域建立一個或多個 Aurora 複本。如需詳細資訊，請參閱影響 Aurora 停機時間的因素。

意圖：此警示用於偵測 Aurora 寫入器資料庫執行個體是否處於停機狀態。這可以防止因為當機或容錯移轉而在寫入器執行個體中發生長時間故障。

統計資料：平均值

建議的閾值：0.0

閾值對正：失敗事件會導致短暫中斷，在此期間，讀取和寫入操作會失敗，並引發例外狀況。不過，服務通常會在 60 秒之內還原，往往不超過 30 秒。

期間：60

警示資料點數目：2

評估期：2

比較運算子：LESS_THAN_OR_EQUAL_TO_THRESHOLD

RollbackSegmentHistoryListLength

尺寸:分貝 InstanceIdentifier

警示說明：此警示有助於監控 Aurora 執行個體的持續較高的轉返區段歷史記錄長度。若 InnoDB 歷史記錄清單長度較長，表示該清單具有過多舊資料列版本，導致查詢和資料庫關閉變慢。如需詳細資訊和疑難排解，請參閱 InnoDB 歷史記錄清單長度顯著增加文件。

意圖：此警示用於偵測持續較高的轉返區段歷史記錄長度。這可協助您避免 Aurora 執行個體中的持續效能降低和 CPU 使用率過高。建議此警示僅用於 Aurora MySQL。

統計資料：平均值

建議閾值：1000000.0

閾值對正：將此閾值設定為 100 萬，讓您有時間調查問題。但是，您可根據可接受的效能層級和工作負載特性來調整此值。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

StorageNetworkThroughput

尺寸：DBClusterIdentifier，角色 = 寫入器

警示說明：此警示有助於監控高儲存網路輸送量。如果儲存網路輸送量超過 EC2 執行個體的總網路頻寬，可能會導致較高的讀取和寫入延遲，進而導致效能降低。您可以從 AWS 主控台檢查 EC2 執行個體類型。若要進行疑難排解，請檢查有關寫入/讀取延遲的任何變更，並評估您是否也對此指標發出警示。如果是這種情況，請在觸發警示時評估您的工作負載模式。這可以協助您確定是否可以優化工作負載以減少網路流量的總量。如果這是不可能的，可能需要考慮擴展您的執行個體。

意圖：此警示用於偵測高儲存網路輸送量。偵測高輸送量可防止網路封包丟失和效能降低。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：應該計算 EC2 執行個體類型總網路頻寬的 80%-90% 左右，然後使用該結果作為閾值，以便在網路封包受到影響之前主動採取動作。也可檢閱儲存網路輸送量的重要性和要求，並分析此指標的歷史行為，以確定合理的閾值級別。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

Amazon Route 53 Public Data Plane

HealthCheckStatus

尺寸:HealthCheckId

警示描述：此警示有助於根據運作狀態檢程式來偵測運作狀態不良的端點。若要了解導致運作狀態不良的失敗原因，使用 Route 53 運作狀態檢查主控台中的「運作狀態檢查程式」索引標籤來檢視每個區域的狀態，以及運作狀態檢查的上一次失敗。狀態索引標籤也會顯示端點報告為運作狀態不良的原因。請參閱故障診斷步驟。

目的：此警示使用 Route53 運作狀態檢查程式來偵測運作狀態不良的端點。

統計資料：平均值

建議的閾值：1.0

閾值對正：端點運作狀態良好時，會將其狀態報告為 1。小於 1 的一切數值都為運作狀態不良。

期間：60

警示資料點數目：3

評估期：3

比較運算子：LESS_THAN_THRESHOLD

Amazon S3

4xxErrors

尺寸:BucketName, FilterId

警示描述：此警示可協助我們報告對用戶端請求做出回應時產生的 4xx 錯誤狀態碼總數。例如，403 錯誤碼可能表示 IAM 政策不正確，而 404 錯誤碼可能表示用戶端應用程式操作不正確。暫時啟用 S3 伺服器存取日誌記錄可協助您使用 HTTP 狀態和錯誤碼欄位，來找出問題的來源。若要了解有關錯誤碼的詳細資訊，請參閱錯誤回應。

目的：此警示用於建立典型 4xx 錯誤率的基準，以便您查看任何可能表示設定問題的異常狀況。

統計資料：平均值

建議的閾值：0.05

閾值對正：建議的閾值會在請求總數的 5% 以上出現 4XX 錯誤時偵測到。經常發生的 4XX 錯誤應觸發警示。但是，將閾值設定為極低的值可能會導致警示過於敏感。您還可調整閾值以適應請求的負載，以可接受的 4XX 錯誤層級為基準。您還可分析歷史資料，以尋找應用程式工作負載可接受的錯誤率，然後相應地調整閾值。

期間：60

警示資料點數目：15

評估期：15

比較運算子：GREATER_THAN_THRESHOLD

5xxErrors

尺寸:BucketName, FilterId

警示描述：此警示有助於偵測用戶端的大量錯誤。這些錯誤表示用戶端發出伺服器無法完成的請求。這可協助您關聯應用程式因 S3 而將面臨的問題。如需協助您有效處理或減少錯誤的詳細資訊，請參閱優化效能設計模式。錯誤也可能是由 S3 問題引起，請檢查 AWS 服務運作狀態儀表板，了解您所在區域中 Amazon S3 的狀態。

目的：此警示可協助偵測應用程式是否因 5xx 錯誤而出現問題。

統計資料：平均值

建議的閾值：0.05

閾值對正：建議將閾值設定為在請求總數的 5% 以上出現 5XXError 時偵測到。但是，您可調整閾值以適應請求的流量以及可接受的錯誤率。您還可分析歷史資料，以查看應用程式工作負載可接受的錯誤率，並相應地調整閾值。

期間：60

警示資料點數目：15

評估期：15

比較運算子：GREATER_THAN_THRESHOLD

OperationsFailedReplication

尺寸:SourceBucket DestinationBucket, RuleId

警示描述：此警示有助於了解複寫失敗。此指標可追蹤使用 S3 CRR 或 S3 SRR 複寫的新物件的狀態，還可追蹤使用 S3 批次複寫來複寫的現有物件。如需詳細資訊，請參閱複寫故障診斷。

目的：此警示用於偵測複寫操作是否失敗。

統計資料：最大值

建議的閾值：0.0

閾值對正：此指標針對成功的操作發出值 0，而此時若沒有執行複寫操作，則不會發出任何值。當指標發出大於 0 的值時，複寫操作就會失敗。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

S3ObjectLambda

4xxErrors

尺寸：AccessPointName DataSource

警示描述：此警示可協助我們報告對用戶端請求做出回應時產生的 4xx 錯誤狀態碼總數。暫時啟用 S3 伺服器存取日誌記錄可協助您使用 HTTP 狀態和錯誤碼欄位，來找出問題的來源。

目的：此警示用於建立典型 4xx 錯誤率的基準，以便您查看任何可能表示設定問題的異常狀況。

統計資料：平均值

建議的閾值：0.05

閾值對正：建議將閾值設定為在請求總數的 5% 以上出現 4XXError 時偵測到。經常發生的 4XX 錯誤應觸發警示。但是，將閾值設定為極低的值可能會導致警示過於敏感。您還可調整閾值以適應請求的負載，以可接受的 4XX 錯誤層級為基準。您還可分析歷史資料，以尋找應用程式工作負載可接受的錯誤率，然後相應地調整閾值。

期間：60

警示資料點數目：15

評估期：15

比較運算子：GREATER_THAN_THRESHOLD

5xxErrors

尺寸：AccessPointName DataSource

警示描述：此警示有助於偵測用戶端的大量錯誤。這些錯誤表示用戶端發出伺服器無法完成的請求。這些錯誤可能是由 S3 問題引起，請檢查 AWS 服務運作狀態儀表板，了解您所在區域中 Amazon S3 的狀態。這可協助您關聯應用程式因 S3 而將面臨的問題。如需協助您有效處理或減少這些錯誤的資訊，請參閱優化效能設計模式。

目的：此警示可協助偵測應用程式是否因 5xx 錯誤而出現問題。

統計資料：平均值

建議的閾值：0.05

閾值對正：建議將閾值設定為在請求總數的 5% 以上出現 5XX 錯誤時偵測到。但是，您可調整閾值以適應請求的流量以及可接受的錯誤率。您還可分析歷史資料，以查看應用程式工作負載可接受的錯誤率，並相應地調整閾值。

期間：60

警示資料點數目：15

評估期：15

比較運算子：GREATER_THAN_THRESHOLD

LambdaResponse4xx

尺寸：AccessPointName DataSource

警示描述：此警示可協助您偵測並診斷呼叫 S3 Object Lambda 時的失敗 (500)。這些錯誤可能是由負責回應請求的 Lambda 函數中的錯誤或設定錯誤導致。調查與物件 Lambda 存取點關聯之 Lambda 函數的 CloudWatch 日誌串流，可協助您根據 S3 物件 Lambda 的回應，精確找出問題的來源。

意圖：此警報用於檢測 WriteGetObjectResponse 呼叫 4xx 客戶端錯誤。

統計資料：平均值

建議的閾值：0.05

閾值對正：建議將閾值設定為在請求總數的 5% 以上出現 4XXError 時偵測到。經常發生的 4XX 錯誤應觸發警示。但是，將閾值設定為極低的值可能會導致警示過於敏感。您還可調整閾值以適應請求的負載，以可接受的 4XX 錯誤層級為基準。您還可分析歷史資料，以尋找應用程式工作負載可接受的錯誤率，然後相應地調整閾值。

期間：60

警示資料點數目：15

評估期：15

比較運算子：GREATER_THAN_THRESHOLD

Amazon SNS

NumberOfMessagesPublished

尺寸:TopicName

警示描述：此警示可偵測何時發布的 SNS 訊息數目太少。如需進行疑難排解，請檢查發布者傳送較少流量的原因。

目的：此警示可協助您主動監控並偵測通知發布中的重大捨棄。這可協助您識別應用程式或商務程序的潛在問題，以便您執行適當的動作來確保預期的通知流程。如果您期望系統提供的流量達到最低，您應建立此警示。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：發布的訊息數目應與應用程式預期發布的訊息數目一致。您還可分析歷史資料、趨勢和流量以尋找合適的閾值。

期間：60

警示資料點數目：5

評估期：5

比較運算子：LESS_THAN_THRESHOLD

NumberOfNotificationsDelivered

尺寸:TopicName

警示描述：此警示可偵測何時傳遞的 SNS 訊息數目太少。這可能是因為無意中取消訂閱端點，或是因為導致訊息發生延遲的 SNS 事件。

目的：此警示可協助您偵測傳遞的數量下降。如果您期望系統提供的流量達到最低，您應建立此警示。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：傳送的訊息數目應與預期產生的訊息數目和取用者數目一致。您還可分析歷史資料、趨勢和流量以尋找合適的閾值。

期間：60

警示資料點數目：5

評估期：5

比較運算子：LESS_THAN_THRESHOLD

NumberOfNotificationsFailed

尺寸:TopicName

警示描述：此警示可偵測何時的失敗 SNS 訊息數目太多。如果要疑難排解失敗的通知，請啟用記 CloudWatch 錄到記錄 檢查日誌可協助您尋找哪些訂閱用戶失敗，及其正在傳回的狀態碼。

目的：此警示可協助您主動尋找傳遞通知時發生的問題，並執行適當的動作來解決這些問題。

統計資料：總和

建議的閾值：視乎您的情況而定

閾值對正：此警示的建議閾值很大程度上取決於失敗通知的影響。檢閱提供給最終使用者的 SLA、容錯和通知的重要性，並分析歷史資料，然後相應地選取閾值。針對只有 SQS、Lambda 或 Firehose 訂閱的主題，失敗的通知數目應為 0。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

NumberOfNotificationsFilteredOut-InvalidAttributes

尺寸:TopicName

警示描述：此警示有助於監控和解決發布者或訂閱用戶的潛在問題。檢查發布者是否發布具有無效屬性的訊息，或者是否將不適當的篩選條件套用至訂閱用戶。您也可以分析 CloudWatch 記錄檔，協助尋找問題的根本原因。

目的：該警示用於偵測發布的訊息是否無效，或是否已將不適當的篩選條件套用至訂閱用戶。

統計資料：總和

建議的閾值：0.0

閾值對正：無效的屬性幾乎總是發布者的錯誤。建議將閾值設定為 0，因為運作狀態良好的系統中不需要無效的屬性。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

NumberOfNotificationsFilteredOut-InvalidMessageBody

尺寸:TopicName

警示描述：此警示有助於監控和解決發布者或訂閱用戶的潛在問題。檢查發布者是否發布具有無效訊息內文的訊息，或者是否將不適當的篩選條件套用至訂閱用戶。您也可以分析 CloudWatch 記錄檔，協助尋找問題的根本原因。

目的：該警示用於偵測發布的訊息是否無效，或是否已將不適當的篩選條件套用至訂閱用戶。

統計資料：總和

建議的閾值：0.0

閾值對正：無效的郵件內文幾乎總是發布者的錯誤。建議將閾值設定為 0，因為運作狀態良好的系統中不需要無效的訊息內文。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

NumberOfNotificationsRedrivenToDlq

尺寸:TopicName

警示描述：此警示有助於監控移至無效字母佇列的訊息數目。

目的：該警示用於偵測移至無效字母佇列的訊息。建議您在將 SNS 與 SQS、Lambda 或 Firehose 搭配使用時建立此警示。

統計資料：總和

建議的閾值：0.0

閾值對正：在任何訂閱用戶類型的運作狀態良好的系統中，訊息都不應移至無效字母佇列。建議在有任何訊息置於佇列時收到通知，以便識別並解決根本原因，並可能重新驅動無效字母佇列中的訊息，進而避免資料遺失。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

NumberOfNotificationsFailedToRedriveToDlq

尺寸:TopicName

警示描述：此警示有助於監控無法移至無效字母佇列的訊息。檢查您的無效字母佇列是否存在，以及設定是否正確。此外，請確認 SNS 具有存取無效字母佇列的許可。如需進一步了解，請參閱無效字母佇列文件。

目的：該警示用於偵測無法移至無法移至無效字母佇列的訊息。

統計資料：總和

建議的閾值：0.0

閾值對正：如果消息無法移至無效字母佇列，則幾乎總是出現錯誤。建議閾值為 0，意味著所有處理失敗的訊息必須在設定無效字母佇列後，才能到達無效字母佇列。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

MonthToDateSpent美元短信

尺寸:TopicName

警示描述：警示有助於監控您的帳戶中是否有足夠的配額，以便讓 SNS 能夠傳遞訊息。如果達到配額，SNS 將無法傳遞 SMS 訊息。如需設定每月 SMS 支出配額的相關資訊，或要求使用提高支出配額的相關資訊 AWS，請參閱設定 SMS 訊息偏好設定。

目的：此警示用於偵測您的帳戶中是否有足夠的配額，以便成功傳遞 SMS 訊息。

統計資料：最大值

建議的閾值：視乎您的情況而定

閾值對正：根據帳戶的配額 (帳戶支出限制) 設定閾值。選擇一個閾值，在達到配額限制時儘早通知您，以便您有時間請求增加配額。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

短信 SuccessRate

尺寸:TopicName

警示描述：此警示有助於監控 SMS 訊息傳遞失敗的速率。您可設定 Cloudwatch Logs 來了解失敗的本質，並據此執行動作。

目的：此警示用於偵測傳遞失敗的 SMS 訊息。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：根據 SMS 訊息傳送失敗的容錯值來設定警示的閾值。

期間：60

警示資料點數目：5

評估期：5

比較運算子：GREATER_THAN_THRESHOLD

Amazon SQS

ApproximateAgeOfOldestMessage

尺寸:QueueName

警示描述：此警示可監控佇列中最舊訊息的存留期。您可使用此警示來監控您的取用者是否以所需速度來處理 SQS 訊息。考慮增加取用者計數或取用者輸送量，以減少訊息存留期。此指標可與 `ApproximateNumberOfMessagesVisible` 結合使用，以確定佇列積壓程度，以及處理訊息的速度。若要防止郵件在處理之前遭到刪除，請考慮將無效字母佇列設定為旁邊潛在的毒藥訊息。

意圖：此警報用於檢測 QueueName 隊列中最舊消息的年齡是否過高。較長的存留期可能表示訊息的處理速度不夠快，或者有一些毒丸訊息停滯在佇列中且無法處理。

統計資料：最大值

建議的閾值：視乎您的情況而定

閾值對正：此警示的建議閾值很大程度上取決於預期的訊息處理時間。您可使用歷史資料來計算平均訊息處理時間，然後將閾值設定為高於佇列取用者預期 SQS 訊息處理時間上限的 50%。

期間：60

警示資料點數目：15

評估期：15

比較運算子：GREATER_THAN_OR_EQUAL_TO_THRESHOLD

ApproximateNumberOfMessagesNotVisible

尺寸:QueueName

警示描述：此警示有助於偵測與 QueueName 相關的大量傳輸中訊息。如需進行疑難排解，請檢查訊息積壓減少。

目的：此警示用於偵測佇列中的大量傳輸中訊息。如果取用者未在可視性逾時期間內刪除訊息，則輪詢佇列時，訊息會重新出現在佇列中。若是 FIFO 佇列，最多可有 20,000 則傳輸中訊息。如果達到此配額，SQS 不會傳回錯誤訊息。FIFO 佇列會查看前 20,000 則訊息，以確定可用的訊息群組。這意味著，如果您在單一訊息群組中有積壓訊息，您將無法取用稍後傳送至佇列的其他訊息群組中的訊息，直至您成功取用積壓訊息為止。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：此警示的建議閾值很大程度上取決於預期的傳輸中訊息數目。您可使用歷史資料來計算預期的傳送中訊息數目上限，並將閾值設定為超過此值的 50%。如果佇列取用者正在處理，但並未刪除佇列中的訊息，則此數目會突增。

期間：60

警示資料點數目：15

評估期：15

比較運算子：GREATER_THAN_OR_EQUAL_TO_THRESHOLD

ApproximateNumberOfMessagesVisible

尺寸:QueueName

警示描述：此警示會監視訊息佇列積壓是否超出預期，表示取用者太慢或沒有足夠的取用者。如果此警示進入 ALARM 狀態，考慮增加取用者計數或加快取用者速度。

目的：此警示用於偵測作用中佇列的訊息計數是否太高，取用者處理訊息較慢，或者沒有足夠的取用者來處理這些訊息。

統計資料：平均值

建議的閾值：視乎您的情況而定

閾值對正：顯示非預期的較高訊息數目，表示取用者未以預期的速率處理訊息。設定此閾值時，您應考慮歷史資料。

期間：60

警示資料點數目：15

評估期：15

比較運算子：GREATER_THAN_OR_EQUAL_TO_THRESHOLD

NumberOfMessagesSent

尺寸:QueueName

警示描述：此警示有助於偵測是否沒有從生產者傳送與 QueueName 相關的訊息。如需進行疑難排解，請檢查生產者未傳送訊息的原因。

目的：此警示用於偵測生產者何時停止傳送訊息。

統計資料：總和

建議的閾值：0.0

閾值對正：如果傳送的訊息數目為 0，則生產者不會傳送任何訊息。如果此佇列的 TPS 較低，請 EvaluationPeriods 相應地增加的數目。

期間：60

警示資料點數目：15

評估期：15

比較運算子：LESS_THAN_OR_EQUAL_TO_THRESHOLD

AWS VPN

TunnelState

尺寸:VpnId

警示描述：此警示可協助您了解一個或多個通道的狀態是否為 DOWN。如需進行疑難排解，請參閱 VPN 通道故障診斷。

目的：此警示用於偵測此 VPN 是否至少有一個通道處於 DOWN 狀態，以便您對受影響的 VPN 進行故障診斷。對於只設定單一通道的網路，此警示將一律處於 ALARM 狀態。

統計資料：最小值

建議的閾值：1.0

閾值對正：值小於 1 表示至少有一個通道處於 DOWN 狀態。

期間：300

警示資料點數目：3

評估期：3

比較運算子：LESS_THAN_THRESHOLD

TunnelState

尺寸:TunnelIpAddress

警示描述：此警示可協助您了解此通道的狀態是否為 DOWN。如需進行疑難排解，請參閱 VPN 通道故障診斷。

目的：此警示用於偵測通道是否處於 DOWN 狀態，以便您對受影響的 VPN 進行故障診斷。對於只設定單一通道的網路，此警示將一律處於 ALARM 狀態。

統計資料：最小值

建議的閾值：1.0

閾值對正：值小於 1 表示通道處於 DOWN 狀態。

期間：300

警示資料點數目：3

評估期：3

比較運算子：LESS_THAN_THRESHOLD

針對指標的警示

以下各節中的步驟說明如何在指標上建立 CloudWatch 警示。

根據靜態閾值建立 CloudWatch 警示

您可以選擇要監視的警示 CloudWatch 量度，以及該量度的臨界值。當指標超過閾值達到指定的評估期間數，警示便會進入 ALARM 狀態。

如果您要在設定為 CloudWatch 跨帳戶觀察性監控帳戶的帳戶中建立警示，則可以設定警示，以便在連結至此監視帳戶的來源帳戶中觀看指標。如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。

根據單一指標建立警示

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇 Alarms (警示)、All alarms (所有警示)。
3. 選擇 Create alarm (建立警示)。
4. 選擇 Select Metric (選取指標)。
5. 執行以下任意一項：
   - 選擇包含您所需指標的服務命名空間。繼續選擇出現的選項以縮減選擇。出現指標清單時，請選取您所需指標旁的核取方塊。
   - 在搜尋方塊中，輸入指標名稱、帳戶 ID、帳戶標籤、維度或資源 ID。然後，選擇其中一個結果，並繼續進行直到指標清單出現為止。選取您所需指標旁的核取方塊。
6. 選擇 Graphed metrics (圖表化指標) 標籤。
   a. 在 Statistic (統計) 下，選擇其中一個統計資料或預先定義的百分位數，或指定自訂百分位數 (例如 **p95.45**)。

b. 在期間下，選擇警示的評估期間。評估警示時，每個期間都會彙整為一個資料點。

您也可以在建立警示時選擇 Y 軸圖例要顯示在左側或右側。只有在建立警示時才會使用此喜好設定。

c. 選擇選取指標。

Specify metric and conditions (指定指標和條件) 頁面隨即出現，顯示您所選取指標和統計資料的圖形及其他資訊。

7. 在 Conditions (條件) 下，指定以下內容：

a. 針對 Whenever ***metric*** is (指標為...時)，請指定指標是否必須大於、小於，或等於閾值。在 than... (於...) 下，指定閾值。

b. 選擇 Additional configuration (其他組態)。針對 Datapoints to alarm (要警示的資料點)，請指定 (資料點) 必須處於 ALARM 狀態多少評估期間，才會觸發警示。如果此處的兩個值相符，您便可以建立警示，在許多連續期間違規時移至 ALARM 狀態。

若要建立 N 個中有 M 個警示，請針對第一個值，指定低於您為第二個值所指定值的值。如需詳細資訊，請參閱 評估警示。

c. 針對 Missing data treatment (遺失資料處理)，選擇警示在遺失某些資料點時的行為。如需詳細資訊，請參閱 設定 CloudWatch 警示如何處理遺失的資料。

d. 若警示使用百分位數作為監控統計資料，則會出現一個 Percentiles with low samples (低樣本的百分位數) 方塊。請使用它來選擇是要評估還是忽略具有低抽樣率的案例。若您選擇 ignore (maintain alarm state) (忽略 (維持警示狀態))，則會在抽樣大小過低時一律維持目前的警示狀態。如需詳細資訊，請參閱 以百分比為基礎的 CloudWatch 警報和低資料樣本。

8. 選擇下一步。

9. 在 Notification (通知) 下，選取 SNS 主題來在警示處於 ALARM 狀態、OK 狀態或 INSUFFICIENT_DATA 狀態時進行通知。

若要讓警示針對相同的警示狀態或不同警示狀態傳送多個通知，請選擇 Add notification (新增通知)。

在 CloudWatch 跨帳戶可觀察性中，您可以選擇將通知發送到多個 AWS 帳戶。例如，傳送給監控帳戶和來源帳戶。

若要讓警示不傳送通知，請選擇 Remove (移動)。

10. 若要讓警示執行 Auto Scaling、 EC2、Lambda 或 Systems Manager 動作，請選擇適當的按鈕，然後選擇警示狀態及要執行的動作。警示只能在進入 ALARM 狀態時執行 Systems Manager 動

作。如需有關「Systems Manager」動作的詳細資訊，請參 CloudWatch 閱設定為 OpsItems 從警示建立和事件建立。

> **Note**
>
> 若要建立執行 SSM Incident Manager 動作的警示，您必須具備特定許可。如需詳細資訊，請參閱AWS 系統管理員事件管理員的身分識別原則範例。

11. 完成時，請選擇下一步。
12. 輸入警示的名稱與說明。此名稱只能包含 UTF-8 字元，不能包含 ASCII 控制字元。說明可以包括降價格式，僅顯示在 CloudWatch 控制台的警報詳細信息選項卡中。Markdown 對於將連結新增至執行手冊或其他內部資源很實用。然後選擇下一步。
13. 在 Preview and create (預覽及建立) 下，請確認資訊和條件都是您希望的內容，然後選擇 Create alarm (建立警示)。

您也可以將警示新增至儀表板。如需詳細資訊，請參閱 從 CloudWatch 儀表板新增或移除警報小工具。

根據度量數學運算式建立 CloudWatch 警示

若要根據度量數學運算式建立警示，請選擇一或多個要在運算式中使用的 CloudWatch 量度。然後，請指定表達式、閾值和評估期間。

您無法根據 SEARCH 表達式建立警示。這是因為搜尋表達式會傳回多個時間序列，並且基於數學表達式的警示只能監看一個時間序列。

根據指標數學表達式建立警示

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇 Alarms (警示)，然後選擇 All alarms (所有警示)。
3. 選擇 Create alarm (建立警示)。
4. 選擇 Select Metric (選取指標)，然後執行以下其中一項動作：
   - 從 AWS namespaces (命名空間) 下拉式清單或 Custom namespaces (自訂命名空間) 下拉式清單中選取命名空間。選取命名空間後，您可以持續選擇選項直到出現指標清單，然後勾選正確指標旁邊的核取方塊。
   - 使用搜尋方塊尋找指標、帳戶 ID、維度或資源 ID。輸入指標、維度或資源 ID 後，您可以持續選擇選項直到出現指標清單，然後勾選正確指標旁邊的核取方塊。

5. (選用) 如果要向指標數學表達式新增另一個指標，則可以使用搜尋方塊尋找特定指標。您可以向指標數學表達式新增最多 10 個指標。
6. 選取 Graphed metrics (圖形指標) 索引標籤。對於您之前新增的每個指標，執行下列動作：

   a. 在 Statistic (統計數字) 資料欄下方，選取下拉式選單。在下拉式選單中，選擇其中一個預先定義的統計數字或百分位數。使用下拉式選單中的搜尋方塊，指定自訂百分位數。

   b. 在 Period (期間) 資料欄下方，選取下拉式選單。在下拉式選單中，選擇其中一個預先定義的評估期間。

   您也可以在建立警示時指定 Y 軸圖例是否要顯示在圖形的左側或右側。

   > **Note**
   >
   > CloudWatch 評估警示時，週期會彙總為單一資料點。

7. 選擇 Add math (新增數學) 下拉式清單，然後從預先定義的指標數學表達式清單中選取 Start with an empty expression (從空的表達式開始)。

   在您選擇 Start with an empty expression (從空的表達式開始) 後，將顯示一個數學表達式方塊，您可以在其中套用或編輯數學表達式。

8. 在數學表達式方塊中，輸入數學表達式，然後選擇 Apply (套用)。

   選擇 Apply (套用) 後，在 Label (標籤) 資料欄旁邊將顯示 ID 資料欄。

   若要使用指標或另一個指標數學表達式的結果作為目前數學表達式公式的一部分，請使用 ID 資料欄下顯示的值。若要變更 ID 的值，請選取目前值 pen-and-paper 旁邊的圖示。新值必須以小寫字母開始，且可以包含數字、字母和底線符號。將 ID 的值變更為更有意義的名稱，有助於更容易了解警示圖形。

   如需指標數學可用函數的資訊，請參閱 指標數學語法和函數。

9. (選用) 在新數學表達式的公式中，使用其他數學表達式的指標和結果，以新增更多數學表達式。
10. 具有要用於警示的表達式之後，請清除頁面上其他每個表達式和每個指標左側的核取方塊。您應只選取要用於警示表達式旁的核取方塊。您為警示選擇的表達式必須產生單一時間序列，而且在圖形上僅顯示一條線。然後選擇 Select metric (選取指標)。

    Specify metric and conditions (指定指標與條件) 頁面隨即出現，顯示您已選取數學表達式的圖形和其他資訊。

11. 針對 Whenever ***expression*** is (表達式為...時)，指定表達式是否必須大於、小於或等於閾值。在 than... (於...) 下，指定閾值。
12. 選擇 Additional configuration (其他組態)。針對 Datapoints to alarm (要警示的資料點)，請指定 (資料點) 必須處於 ALARM 狀態多少評估期間，才會觸發警示。如果此處的兩個值相符，您便可以建立警示，在許多連續期間違規時移至 ALARM 狀態。

    若要建立 N 個中有 M 個警示，請針對第一個值，指定低於您為第二個值所指定值的值。如需詳細資訊，請參閱 評估警示。
13. 針對 Missing data treatment (遺失資料處理)，選擇警示在遺失某些資料點時的行為。如需詳細資訊，請參閱 設定 CloudWatch 警示如何處理遺失的資料。
14. 選擇下一步。
15. 在 Notification (通知) 下，選取 SNS 主題來在警示處於 ALARM 狀態、OK 狀態或 INSUFFICIENT_DATA 狀態時進行通知。

    若要讓警示針對相同的警示狀態或不同警示狀態傳送多個通知，請選擇 Add notification (新增通知)。

    若要讓警示不傳送通知，請選擇 Remove (移動)。
16. 若要讓警示執行 Auto Scaling、 EC2、Lambda 或 Systems Manager 動作，請選擇適當的按鈕，然後選擇警示狀態及要執行的動作。如果選擇 Lambda 函數作為警示動作，則可以指定函數名稱或 ARN，並且可以選擇性地選擇函數的特定版本。

    警示只能在進入 ALARM 狀態時執行 Systems Manager 動作。如需有關 Systems Manager 動作的詳細資訊，請參閱設定 CloudWatch 為 OpsItems 從警示建立與事件建立。

    > **Note**
    >
    > 若要建立執行 SSM Incident Manager 動作的警示，您必須具備特定許可。如需詳細資訊，請參閱AWS 系統管理員事件管理員的身分識別原則範例。
17. 完成時，請選擇下一步。
18. 輸入警示的名稱與說明。然後選擇下一步。

    此名稱只能包含 UTF-8 字元，不能包含 ASCII 控制字元。說明可以包括降價格式，僅顯示在 CloudWatch 控制台的警報詳細信息選項卡中。Markdown 對於將連結新增至執行手冊或其他內部資源很實用。

19. 在 Preview and create (預覽及建立) 下，請確認資訊和條件都是您希望的內容，然後選擇 Create alarm (建立警示)。

您也可以將警示新增至儀表板。如需詳細資訊，請參閱 從 CloudWatch 儀表板新增或移除警報小工具。

根據指標洞察查詢建立 CloudWatch 警示

您可以針對傳回單一時間序列的 Metrics Insights 查詢建立警示。這對於建立動態警示來監看基礎結構或應用程式機群中的彙總指標特別有用。建立警示後，在資源新增至機群或從機群中移除時，它會進行調整。例如，您可以建立警示，以監看機群的 CPU 使用率，該警示會在您新增或移除執行個體時，進行動態調整。

如需完整說明，請參閱 在 Metrics Insights 查詢上建立警示。

根據連線的資料來源建立警示

您可以建立警示，以監視不在資料來源的指標 CloudWatch。如需有關建立其他資料來源連線的詳細資訊，請參閱 從其他資料來源中查詢指標。

若要從已連線的資料來源建立指標警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Metrics (指標)、All metrics (所有指標)。
3. 選擇多來源查詢索引標籤。
4. 對於資料來源，選取您想要使用的資料來源。
5. 查詢建置器會提示您輸入查詢所需的資訊，以擷取用於警示的指標。每個資料來源的工作流程都不同，並針對資料來源量身打造。例如，對於 Amazon Managed Service for Prometheus 和 Prometheus 資料來源，則會出現一個包含查詢協助程式的 PromQL 查詢編輯器方塊。
6. 完成查詢的建構後，請選擇圖形查詢。
7. 如果範例圖表看起來符合您的預期，請選擇建立警示。
8. 指定指標和條件頁面出現。如果使用的查詢產生多個時間序列，將會在頁面頂部看到警告橫幅。如果您這樣做，請選取一個函數，用於彙總彙總函數中的時間序列。
9. (選用) 新增警示標籤。
10. 對於無論何時如***your-metric-name***此。 ，選擇「大於」、「大於/等於」、「低/等於」或「小於」。對於相比...，為閾值指定一個數字。

11. 選擇 Additional configuration (其他組態)。針對 Datapoints to alarm (要警示的資料點)，請指定 (資料點) 必須處於 ALARM 狀態多少評估期間，才會觸發警示。如果此處的兩個值相符，您便可以建立警示，在許多連續期間違規時移至 ALARM 狀態。

    若要建立 N 個中有 M 個警示，請針對小於第二個值之數字的第一個值指定數字。如需詳細資訊，請參閱 評估警示。

12. 對於 Missing data treatment (遺失資料處理方式)，選擇警示在遺失某些資料點時的行為。如需詳細資訊，請參閱 設定 CloudWatch 警示如何處理遺失的資料。

13. 選擇下一步。

14. 對於通知，請指定當警示轉換為 ALARM、OK 或 INSUFFICIENT_DATA 狀態時要通知的 Amazon SNS 主題。

    a. (選用) 若要針對相同警示狀態或不同警示狀態傳送多個通知，請選擇 Add notification (新增通知)。

    > **Note**
    >
    > 建議您設定警示，以便除了在進入警示狀態外，進入資料不足狀態時應採取動作。這是因為連線至資料來源的 Lambda 函數有許多問題可能會導致警示轉換為資料不足。

    b. (選用) 若不傳送 Amazon SNS 通知，請選擇移除。

15. 若要讓警示執行 Auto Scaling、 EC2、Lambda 或 Systems Manager 動作，請選擇適當的按鈕，然後選擇警示狀態及要執行的動作。如果選擇 Lambda 函數作為警示動作，則可以指定函數名稱或 ARN，並且可以選擇性地選擇函數的特定版本。

    警示只能在進入 ALARM 狀態時執行 Systems Manager 動作。如需有關 Systems Manager 動作的詳細資訊，請參閱設定 CloudWatch 為 OpsItems 從警示建立與事件建立。

    > **Note**
    >
    > 若要建立執行 SSM Incident Manager 動作的警示，您必須具備特定許可。如需詳細資訊，請參閱AWS 系統管理員事件管理員的身分識別原則範例。

16. 選擇下一步。

17. 在 Name and description (名稱和描述) 下，輸入警示的名稱和描述，然後選擇 Next (下一步)。此名稱只能包含 UTF-8 字元，不能包含 ASCII 控制字元。說明可以包括降價格式，僅顯示在

CloudWatch 控制台的警報詳細信息選項卡中。Markdown 對於將連結新增至執行手冊或其他內部資源很實用。

> **Tip**
>
> 警示名稱只能包含 UTF-8 字元。它不能包含 ASCII 控制字元。

18. 在 Preview and create (預覽及建立) 下，請確認警示資訊和條件都是正確的，然後選擇 Create alarm (建立警示)。

已連線資料來源的警示詳細資訊

- CloudWatch 評估警報時，即使警報的時間長度超過一分鐘，它也會每分鐘執行一次。若要讓警示運作，Lambda 函數必須能夠傳回從任何一分鐘開始的時間戳記清單，而不僅是週期長度的倍數。這些時間戳記必須相隔一個週期長度。

  因此，如果 Lambda 查詢的資料來源只能傳回週期長度倍數的時間戳記，則函數應「重新取樣」擷取的資料，以符合 `GetMetricData` 請求所預期的時間戳記。

  例如，使用每次偏移一分鐘的五分鐘時段，每分鐘評估一次週期為五分鐘的警示。在此案例中：

  - 對於 12:15:00 的`12:00:00`警示評估， CloudWatch 預期資料點的時間戳記為、和。`12:05:00` `12:10:00`
  - 然後，對於 12:16:00 的`12:01:00`警報評估， CloudWatch 預期的資料點的時間戳記為、和。`12:06:00` `12:11:00`

- 評估警示時，Lambda 函數傳回的任何與預期時間戳記不符合的資料點都 CloudWatch 會捨棄，並使用剩餘的預期資料點評估警示。例如，在 `12:15:00` 評估警示時，它預期資料具有 `12:00:00`、`12:05:00` 和 `12:10:00` 時間戳記。如果接收時間戳記為`12:00:00`、、和的資料 `12:05:00`、`12:06:00`，則會捨棄來自`12:06:00`的資料`12:10:00`，並使用其他時間戳記 CloudWatch 評估警示。

  然後，對於在 `12:16:00` 進行的下一次評估，它預期資料具有 `12:01:00`、`12:06:00` 和 `12:11:00` 時間戳記。如果它只有時間戳記為 `12:00:00`、`12:05:00` 和 `12:10:00` 的資料，則所有這些資料點都會在 12:16:00 被忽略，並且警示會根據您指定該警示來處理遺失資料的方式轉換為相應狀態。如需詳細資訊，請參閱 評估警示。

- 建議您建立這些警示，以便在它們轉換為 `INSUFFICIENT_DATA` 狀態時採取動作，因為多個 Lambda 函數失敗使用案例都會將警示轉換為 `INSUFFICIENT_DATA`，無論您設定警示以何種方式處理遺失的資料。

- 如果 Lambda 函數傳回錯誤或傳回部分資料：
  - 如果呼叫 Lambda 函數時發生許可問題，警示會開始遺失資料轉換，其依據為您指定該警示在建立時處理遺失資料的方式。
  - 如果 Lambda 函數傳回 `'StatusCode' = 'PartialData'`，則警示評估失敗，並且警示在嘗試三次後轉換為 `INSUFFICIENT_DATA`，這大約需要三分鐘。
  - 任何來自 Lambda 函數的其他錯誤都會導致警示轉換為 `INSUFFICIENT_DATA`。
- 如果 Lambda 函數請求的指標有一些延遲，從而導致最後一個資料點永遠遺失，您應採取因應措施。可以建立「N 中取 M」警示，或增加警示的評估時間。如需「N 中取 M」警示的詳細資訊，請參閱 評估警示。

根據異常偵測建立 CloudWatch 警示

您可以根據 CloudWatch 異常偵測建立警示，以分析過去的指標資料並建立預期值模型。預期值會將指標中每小時、每日和每週模式列入考慮。

您可以設定異常偵測閾值的值，並將此臨界值與模型搭配 CloudWatch 使用，以決定量度值的「正常」範圍。較高的臨界值會產生較厚的「正常」值範圍。

您可以選擇警示觸發時機是在指標值超過預期值範圍、低於範圍，或者兩者擇一。

您也可以使用它根據指標數學表達式的單一指標和指標輸出來建立異常偵測警示。您可以使用這些表達式來建立可視化異常偵測頻段的圖形。

在設定為 CloudWatch 跨帳戶觀察性監視帳戶的帳戶中，除了監控帳戶中的指標之外，您還可以在來源帳戶中建立指標的異常偵測器。

如需詳細資訊，請參閱 使用 CloudWatch 異常偵測。

> **Note**
>
> 如果您在指標主控台中，因視覺化目的正在對指標使用異常偵測，並對該相同指標建立異常偵測警示，則您為警示設定的閾值不會改變您已為視覺化設定的閾值。如需詳細資訊，請參閱 建立圖形。

建立以異常偵測為基礎的警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/

2. 在導覽窗格中，選擇 Alarms (警示)、All alarms (所有警示)。
3. 選擇 Create alarm (建立警示)。
4. 選擇 Select Metric (選取指標)。
5. 執行以下任意一項：
   - 選擇包含指標的服務命名空間，然後繼續選擇顯示的選項以縮小選項範圍。出現指標清單時，請選取指標旁的核取方塊。
   - 在搜尋方塊中，輸入指標名稱、維度或資源 ID。選取其中一個結果，然後繼續選擇出現的選項，直到出現指標清單為止。選取指標旁的核取方塊。
6. 選擇「圖形度量」。
   a. (選擇性) 在統計資料中，選擇下拉式清單，然後選取其中一個預先定義的統計資料或百分位數。您可以使用下拉式清單中的搜尋方塊，指定自訂百分位數，例如 **p95.45**。
   b. (選擇性) 對於期間，選擇下拉式清單，然後選取其中一個預先定義的評估期間。

   > **Note**
   >
   > 當 CloudWatch 評估您的警報時，它會將週期加重為單個資料點。若是異常偵測警示，評估期間必須為一分鐘或更久。

7. 選擇下一步。
8. 在 Conditions (條件) 下，指定以下內容：
   a. 選擇 Anomaly detection (異常偵測)。

   如果此量度和統計資料的模型已存在，則會在畫面頂端的圖形中 CloudWatch 顯示異常偵測區間的預覽。建立警示後，最多需要 15 分鐘，實際異常偵測範圍才會出現在圖表中。在這之前，您看到的範圍只是異常偵測範圍的大約值。

   > **Tip**
   >
   > 若要在畫面上方查看較長時間範圍的圖表，請選擇頁面右上角的 Edit (編輯)。

   如果此量度和統計資料的模型尚未存在，請在您完成建立鬧鐘後 CloudWatch 產生異常偵測頻帶。如果是新模型，最多可能需要 3 小時，實際異常偵測範圍才會出現在圖表中。訓練新模型可能需要長達兩週的時間，這樣異常偵測範圍才能顯示更準確的預期值。

b. 無論**##**是何時的，請指定何時觸發警示。例如，當指標大於、小於頻帶或位於頻帶範圍的外部 (任一方向)。

c. 對於 Anomaly detection threshold (異常偵測臨界值)，請選擇用於異常偵測臨界值的數字。數字越大，「正常」值範圍越寬，對指標變化的容忍度越大。數字越小，範圍越窄，因此將以較小的指標偏差進入 ALARM 狀態。該數字不一定是整數。

d. 選擇 Additional configuration (其他組態)。針對 Datapoints to alarm (要警示的資料點)，請指定 (資料點) 必須處於 ALARM 狀態多少評估期間，才會觸發警示。如果此處的兩個值相符，您便可以建立警示，在許多連續期間違規時移至 ALARM 狀態。

   若要建立 N 個中有 M 個警示，請針對小於第二個值之數字的第一個值指定數字。如需詳細資訊，請參閱 評估警示。

e. 對於 Missing data treatment (遺失資料處理方式)，選擇警示在遺失某些資料點時的行為。如需詳細資訊，請參閱 設定 CloudWatch 警示如何處理遺失的資料。

f. 若警示使用百分位數作為監控統計資料，則會出現一個 Percentiles with low samples (低樣本的百分位數) 方塊。請使用它來選擇是要評估還是忽略具有低抽樣率的案例。若您選擇 Ignore (maintain alarm state) (忽略 (維持警示狀態))，則會在抽樣大小過低時一律維持目前的警示狀態。如需詳細資訊，請參閱 以百分比為基礎的 CloudWatch 警報和低資料樣本。

9. 選擇下一步。

10. 在 Notification (通知) 下，選取 SNS 主題來在警示處於 ALARM 狀態、OK 狀態或 INSUFFICIENT_DATA 狀態時進行通知。

    若要針對相同的警示狀態或不同警示狀態傳送多個通知，請選擇 Add notification (新增通知)。

    如果您不想要警示傳送通知，則請選擇 Remove (移除)。

11. 您可以設定警示以執行 EC2 動作或在 Lambda 函數變更狀態時叫用，或在進入警報狀態時建立 Systems Manager OpsItem 或事件。若要進行此操作，請選擇適當的按鈕，然後選擇警示狀態及要執行的動作。

    如果選擇 Lambda 函數作為警示動作，則可以指定函數名稱或 ARN，並且可以選擇性地選擇函數的特定版本。

    如需有關「Systems Manager」動作的詳細資訊，請參 CloudWatch 閱設定為 OpsItems 從警示建立和事件建立。

> **Note**
>
> 若要建立執行 AWS Systems Manager Incident Manager 動作的警示，您必須具備特定許可。如需詳細資訊，請參閱AWS 系統管理員事件管理員的身分識別原則範例。

12. 選擇下一步。
13. 在 Name and description (名稱和描述) 下，輸入警示的名稱和描述，然後選擇 Next (下一步)。此名稱只能包含 UTF-8 字元，不能包含 ASCII 控制字元。說明可以包括降價格式，僅顯示在 CloudWatch 控制台的警報詳細信息選項卡中。Markdown 對於將連結新增至執行手冊或其他內部資源很實用。

> **Tip**
>
> 此警示名稱只能包含 UTF-8 字元，不能包含 ASCII 控制字元。

14. 在 Preview and create (預覽及建立) 下，請確認警示資訊和條件都是正確的，然後選擇 Create alarm (建立警示)。

修改異常偵測模型

建立警示後，您就可以調整異常偵測模型。您可以在建立模型時排除某些期間。請務必從訓練資料中排除不尋常的事件，例如系統中斷、部署和假日。您也可以指定是否針對日光節約時間變更來調整模型。

調整警示的異常偵測模型

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Alarms (警示)、All alarms (所有警示)。
3. 選擇警示的名稱。如有需要，請使用搜尋方塊尋找警示。
4. 選擇分析、在指標中。
5. 在詳細資訊欄位中，選擇 ANOMALY_DETECTION_BAND、編輯異常偵測模型。
6. 若要排除用於產生模型的時段，請依結束日期選擇行事曆圖示。然後選取或輸入要排除的訓練天數和時間，然後選擇 Apply (套用)。
7. 如果指標容易受到日光節約時間變更的影響，在 Metric timezone (指標時區) 方塊中選取適當的時區。
8. 選擇更新。

刪除異常偵測模型

對警示使用異常偵測會累積 費用。作為最佳實務，如果警示不再需要異常偵測模型，請先刪除警示，然後再刪除模型。評估異常偵測警示時，我們會代表您建立任何缺失的異常偵測器。如果您刪除模型，而未刪除警示，警示會自動重新建立模型。

刪除警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Alarms (警示)、All alarms (所有警示)。
3. 選擇警示的名稱。
4. 選擇 動作、刪除。
5. 在確認方塊中，選擇「刪除」。

刪除已用於警示的異常偵測模型

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Metrics (指標)，然後選擇 All metrics (所有指標)。
3. 選擇 Browse (瀏覽)，然後選取包含異常偵測模型的指標。您可以在搜尋方塊中搜尋指標，也可透過選擇選項來選取指標。
   - (選用) 如果您使用原始介面，請選擇 All metrics (所有指標)，然後選擇包含異常偵測模型的指標。您可以在搜尋方塊中搜尋指標，也可透過選擇選項來選取指標。
4. 選擇 Graphed metrics (圖表化指標)。
5. 在 Graphed metrics (圖表化指標) 索引標籤中，選擇您想要移除之異常偵測模型的名稱，然後選擇 Delete anomaly detection model (刪除異常偵測模型)。
   - (選用) 如果您使用原始介面，請選擇 Edit model (編輯模型)。您將被導向至新的畫面。在新畫面上，選擇 Delete model (刪除模型)，然後選擇 Delete (刪除)。

針對日誌的警示

下列各節中的步驟說明如何在記錄檔上建立 CloudWatch 警示。

根據記錄群組量度篩選器建立 CloudWatch 警示

本節中的程序說明如何根據日誌群組指標篩選條件建立警示。使用指標篩選器，您可以在資料傳送至記錄檔資料時尋找術語和模式 CloudWatch。如需詳細資訊，請參閱 Amazon CloudWatch Logs 使用者

指南中的使用篩選器從日誌事件建立指標。根據日誌群組指標篩選條件建立警示之前，必須先完成下列動作：

- 建立 日誌群組 如需詳細資訊，請參閱 Amazon CloudWatch 日誌使用者指南中的使用日誌群組和日誌串流。
- 建立指標篩選條件。如需詳細資訊，請參閱 Amazon CloudWatch 日誌使用者指南中的為日誌群組建立指標篩選器。

根據日誌群組指標篩選條件建立警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Logs (日誌)，然後選擇 Log groups (日誌群組)。
3. 選擇包含指標篩選條件的日誌群組。
4. 選擇 Metric filters (指標篩選條件)。
5. 在指標篩選條件索引標籤中，選取要作為警示依據之指標篩選條件的方塊。
6. 選擇 Create alarm (建立警示)。
7. (選用) 在 Metric (指標) 下，編輯 Metric name (指標名稱)、Statistic (統計資料) 以及 Period (期間)。
8. 在 Conditions (條件) 下，指定以下內容：
   a. 對於 Threshold type (閾值類型)，選擇 Static (靜態) 或 Anomaly detection (異常偵測)。
   b. 對於無論何時如***your-metric-name***此。 ，選擇「大於」、「大於/等於」、「低/等於」或「小於」。
   c. 對於 than . . . (相比…)，為閾值指定一個數字。
9. 選擇 Additional configuration (其他組態)。
   a. 對於 Data points to alarm (要警示的資料點)，指定多少資料點會觸發警示進入 ALARM 狀態。如果您指定相符的值，則在許多連續期間違規時，警示會進入 ALARM 狀態。若要建立 N 個中有 M 個警示，請為第一個值指定一個數字，該數字應小於您為第二個值指定的數字。如需詳細資訊，請參閱使用 Amazon CloudWatch 警示。
   b. 對於 Missing data treatment (遺失資料處理方式)，請選取一個選項，以指定評估警示時如何處理遺失資料。
10. 選擇下一步。
11. 對於 Notification (通知)，請指定當警示處於 ALARM、OK 或 INSUFFICIENT_DATA 狀態時要通知的 Amazon SNS 主題。

a. (選用) 若要針對相同警示狀態或不同警示狀態傳送多個通知，請選擇 Add notification (新增通知)。

b. (選用) 若不傳送通知，請選擇 Remove (移除)。

12. 若要讓警示執行 Auto Scaling、 EC2、Lambda 或 Systems Manager 動作，請選擇適當的按鈕，然後選擇警示狀態及要執行的動作。如果選擇 Lambda 函數作為警示動作，則可以指定函數名稱或 ARN，並且可以選擇性地選擇函數的特定版本。

    警示只能在進入 ALARM 狀態時執行 Systems Manager 動作。如需有關 Systems Manager 動作的詳細資訊，請參閱設定 CloudWatch 為 OpsItems 從警示建立與事件建立。

    > **Note**
    >
    > 若要建立執行 SSM Incident Manager 動作的警示，您必須具備特定許可。如需詳細資訊，請參閱AWS 系統管理員事件管理員的身分識別原則範例。

13. 選擇下一步。

14. 在 Name and description (名稱和描述) 中，輸入警示的名稱和描述。此名稱只能包含 UTF-8 字元，不能包含 ASCII 控制字元。說明可以包括降價格式，僅顯示在 CloudWatch 控制台的警報詳細信息選項卡中。Markdown 對於將連結新增至執行手冊或其他內部資源很實用。

15. 對於 Preview and create (預覽並建立)，請檢查組態是否正確無誤，然後選擇 Create alarm (建立警示)。

合併警示

使用時 CloudWatch，您可以將多個警示合併為一個複合警示，以針對整個應用程式或資源群組建立摘要、彙總的健全狀況指標。複合警示可監控其他警示的狀態來判斷其狀態。您可使用布林邏輯來定義規則，以合併這些受監控警示的狀態。

您可使用複合警示來減少警示雜訊，但只能在彙總層級執行動作。例如，您可建立一個複合警示，以在與 Web 伺服器相關的任何警示觸發時，向 Web 伺服器團隊傳送通知。若其中任何警示進入 ALARM 狀態，複合警示會自行進入 ALARM 狀態，並向您的團隊傳送通知。如果與您的 Web 伺服器相關的其他警示也進入 ALARM 狀態，您的團隊不會因新的通知而超載，因為複合警示已通知其現有情況。

您還可使用複合警示來建立複雜的警示條件，並且只有在滿足許多不同條件時才會執行動作。例如，您可建立合併 CPU 警示和記憶體警示的複合警示，並且只有在 CPU 警示和記憶體警示都觸發時才會通知您的團隊。

使用複合警示

您在使用複合警示時有兩個選項：

- 設定您只想在複合警示層級執行的動作，並建立不含處理動作的基礎監控警示
- 在複合警示層級設定不同的動作集。例如，在出現普遍問題的情況下，複合警示動作可能會讓不同的團隊參與。

複合警示僅可執行下列動作：

- 通知 Amazon SNS 主題
- 調用 Lambda 函數
- OpsItems 在 Systems Manager 作業中心建立
- 在 Systems Manager Incident Manager 中建立事件

> **Note**
>
> 複合警示中的所有基礎警示皆必須與您的複合警示處於相同帳戶和相同區域。但是，如果您在 CloudWatch 跨帳戶可觀察性監視帳戶中設置複合警報，則基礎警報可以監視不同來源帳戶和監視帳戶本身中的指標。如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。
> 每個複合警示可以監控 100 個基礎警示，150 個複合警示可以監控單一個基礎警示。

規則表達式

所有複合警示都包含規則表達式。規則表達式會告訴複合警示要監控哪些其他警示，並從何判斷其狀態。規則表達式可以參照指標警示和複合警示。當您參照規則表達式中的警示時，您可以為警示指定一個函數，以決定警示將處於下列三種狀態中的哪一種：

- ALARM (警示)

  如果警示處於 ALARM 狀態，則 ALARM (“alarm-name 或 alarm-ARN”) 為 TRUE。
- OK

  如果警示處於 OK 狀態，則 OK (“alarm-name 或 alarm-ARN”) 為 TRUE。
- INSUFFICIENT_DATA

如果給定的警示處於 INSUFFICIENT_DATA 狀態，則 INSUFFICIENT_DATA (“alarm-name 或 alarm-ARN”) 為 TRUE。

> **Note**
>
> TRUE 永遠評估為 TRUE，FALSE 永遠評估為 FALSE。

範例表達式

請求參數 AlarmRule 支援使用邏輯運算子 AND、OR、NOT，讓您可以將多個函數組合成單一運算式。下列範例運算式顯示如何在複合警示中設定基礎警示：

- `ALARM(CPUUtilizationTooHigh) AND ALARM(DiskReadOpsTooHigh)`

  此表達式指定複合警示僅在 CPUUtilizationTooHigh 和 DiskReadOpsTooHigh 處於 ALARM 時進入 ALARM。
- `ALARM(CPUUtilizationTooHigh) AND NOT ALARM(DeploymentInProgress)`

  此表達式指定複合警示在 CPUUtilizationTooHigh 處於 ALARM 且 DeploymentInProgress 不處於 ALARM 時進入 ALARM。這是一個複合警示的範例，用來減少部署期間的警示干擾。
- `(ALARM(CPUUtilizationTooHigh) OR ALARM(DiskReadOpsTooHigh)) AND OK(NetworkOutTooHigh)`

  此表達式指定複合警示在 (ALARM(CPUUtilizationTooHigh) 或 (DiskReadOpsTooHigh) 處於 ALARM 且 (NetworkOutTooHigh) 處於 OK 時進入 ALARM。這是一個複合警示的範例，每當發生網路問題時，若有任何基礎警示不處於 ALARM 便不會向您發送通知，藉此減少警示干擾。

主題

- 建立複合警示
- 抑制複合警示動作

建立複合警示

本節中的步驟說明如何使用主 CloudWatch 控台建立複合警示。您也可以使用 API 或 AWS CLI 建立複合警示。如需詳細資訊，請參閱PutCompositeAlarm或 put-composite-alarm

建立複合警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Alarms (警示)，然後選擇 All alarms (所有警示)。
3. 從警示列表選取您要在規則表達式中參照的每個現有警示旁的核取方塊，然後選擇 Create composite alarm (建立複合警示)。
4. 在 Specify composite alarm conditions (指定複合警示條件) 中，指定新複合警示的規則表達式。

   > **Note**
   >
   > 您從警示清單中選取的警示會自動列在 Conditions (條件) 方框中。在預設情況下，ALARM 函數已指定給您的每個警示，且每個警示都由邏輯運算子 OR 串聯。

   您可以使用下列子步驟來調整您的規則表達式：

   a. 您可以從以下位置將每個警示的必要狀態從 ALARM 變更至 OK 或 INSUFFICENT_DATA。
   b. 您可以將規則表達式中的邏輯運算子從 OR 變更至 AND 或 NOT，也可以加入括號來分組函數。
   c. 您可以在規則表達式中加入其他警示，或從規則表達式中刪除警示。

   範例：具有條件的規則表達式

   ```
   (ALARM("CPUUtilizationTooHigh") OR
   ALARM("DiskReadOpsTooHigh")) AND
   OK("NetworkOutTooHigh")
   ```

   在複合警示進入的範例規則運算式中，ALARM當 ALARM (「CPUUtilizationTooHigh」或 ALARM (」DiskReadOpsTooHigh「) 同時ALARM位於 OK (」NetworkOutTooHigh「) 中時，複合警示會進入OK。
5. 完成時，請選擇下一步。
6. 在 Configure actions (設定動作) 中，您可以從下列項目選擇：

   對於 Notification (通知)

   - Select an exisiting SNS topic (選取現有的 SNS 主題)、Create a new SNS topic (建立新 SNS 主題)，或是 Use a topic ARN (使用主題 ARN) 來定義將接收通知的 SNS 主題。

- Add notification (新增通知) 以針對相同的警示狀態或不同警示狀態傳送多個通知。
- Remove (移除) 以停止警示傳送通知或採取動作。

(選用) 若要讓警示在變更狀態時調用 Lambda 函數，請選擇新增 Lambda 動作。然後指定函數名稱或 ARN，並選擇性地選擇函數的特定版本。

對於 Systems Manager action (Systems Manager 動作)

- Add Systems Manager action (新增 Systems Manager 動作)，以讓您的警示在進入 ALARM (警示) 狀態時可以執行 SSM 動作。

若要深入瞭解 Systems Manager 動作，請參閱AWS Systems Manager 使用指南中的設定 CloudWatch 為 OpsItems 從警示建立和事件管理員使用指南中的建立事件。若要建立執行 SSM Incident Manager 動作的警示，您必須具備正確的許可。如需詳細資訊，請參閱事件管理員使用指南中的 AWS Systems Manager 事件管理員身分識別原則範例。

7. 完成時，請選擇下一步。
8. 在 Add name and description (新增名稱和描述) 中，輸入警示名稱，並選擇是否為您的新複合警示加入描述。此名稱只能包含 UTF-8 字元，不能包含 ASCII 控制字元。說明可以包括降價格式，僅顯示在 CloudWatch 控制台的警報詳細信息選項卡中。Markdown 對於將連結新增至執行手冊或其他內部資源很實用。
9. 完成時，請選擇下一步。
10. 在 Preview and create (預覽並建立) 中確認您的資訊，然後選擇 Create composite alarm (建立複合警示)。

> **Note**
>
> 您可能會建立複合警示循環，使一個複合警示和另一個複合警示彼此參照。如果您發現自己處於這種情況，您的複合警示會停止接受評估，而且您會無法刪除複合警示，因為兩者彼此參照。要打破複合警示彼此參照的循環，最簡單的方法是將您其中一個複合警示中的函數 `AlarmRule` 變更為 `False`。

抑制複合警示動作

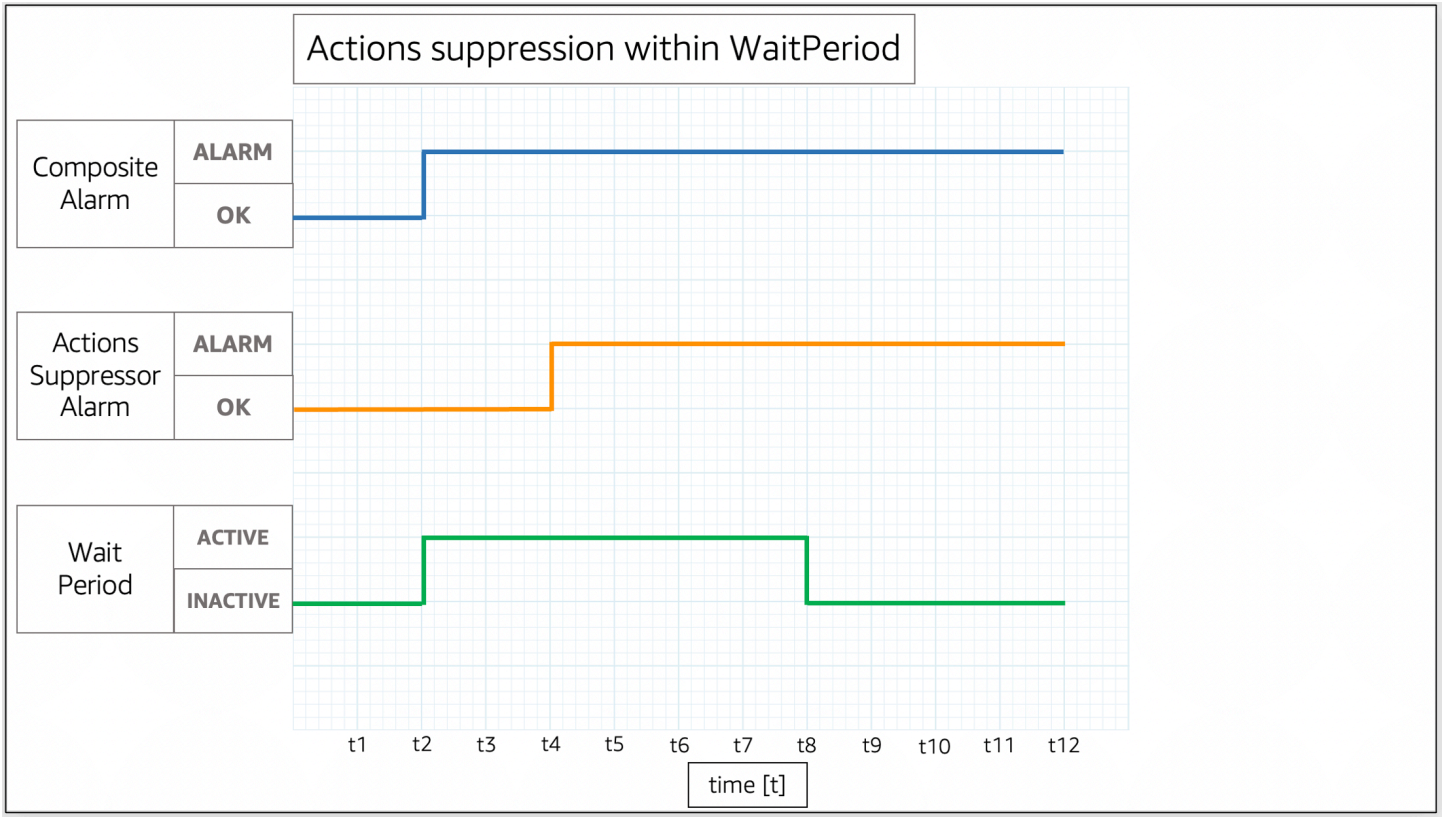
由於複合警示可讓您跨多個警示取得運作狀態的彙總檢視，因此，在常見情況下，預期會觸發這些警示。例如，在應用程式維護時段期間，或您在調查進行中事件時。在這類情況下，您可能想要抑制複合警示的動作，以防止不必要的通知或建立新的事件工單

藉由「複合警示動作抑制」功能，您可以將警示定義為抑制器警示。抑制器警示可阻止複合警示採取動作。例如，您可以指定代表支援資源狀態的抑制器警示。如果支援資源關閉，抑制器警示會阻止複合警示傳送通知。複合警示動作抑制有助於減少警示干擾，為您節省管理警示的時間，將更多時間專注於操作上。

您可以在設定複合警示時指定抑制器警示。任何警示都可以用作抑制器警示。當抑制器警示狀態從 OK 變更至 ALARM，其複合警示會停止採取動作。當抑制器警示狀態從 ALARM 變更至 OK，其複合警示會恢復採取動作。

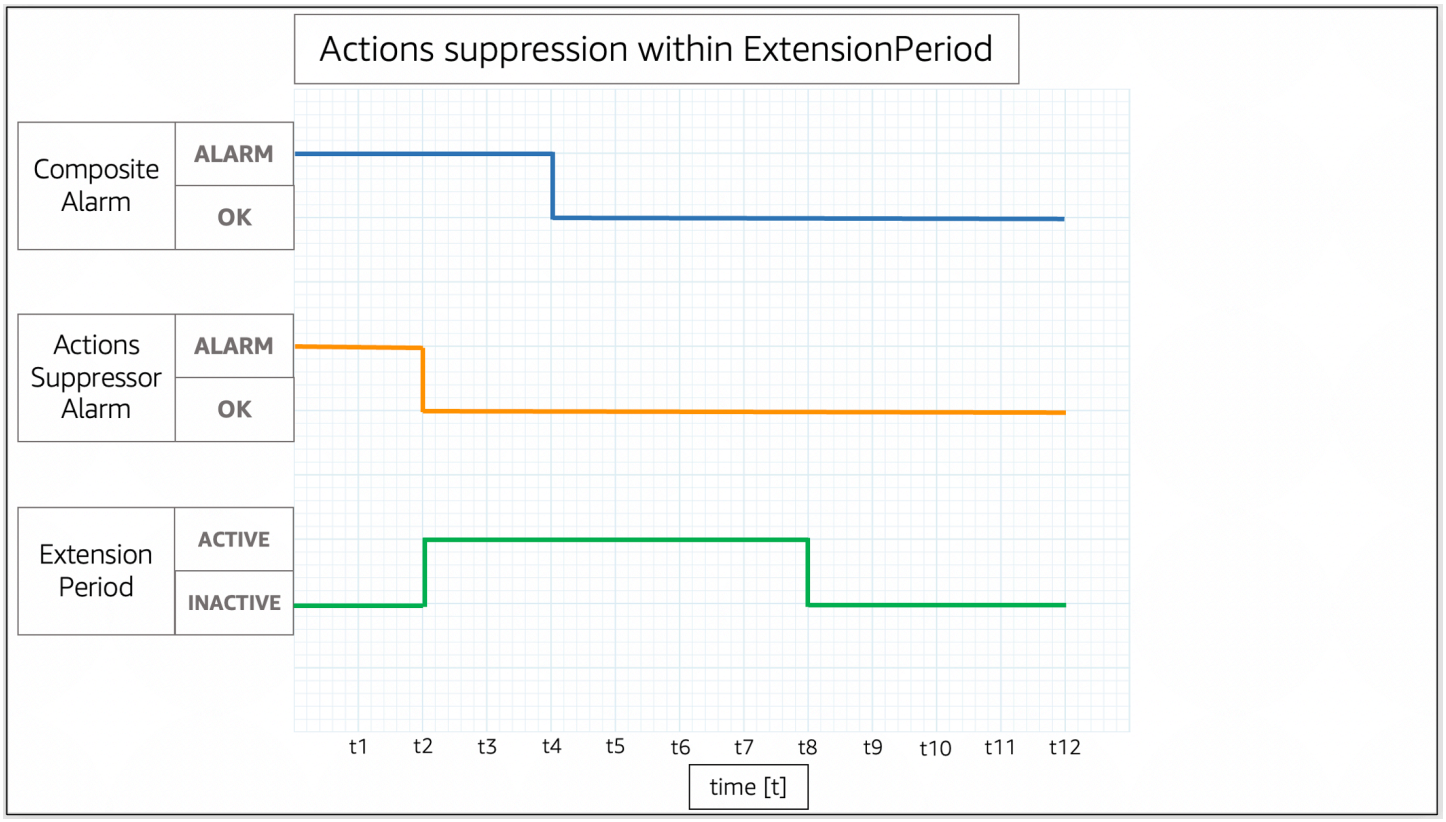
`WaitPeriod` 和 `ExtensionPeriod`

當您指定抑制器警示時，您可以設定參數 `WaitPeriod` 和 `ExtensionPeriod`。這些參數可防止複合警示在抑制器警示狀態變更時意外採取動作。利用 `WaitPeriod` 來補償抑制器警示從 OK 變更至 ALARM 時可能發生的任何延遲。例如，若抑制器警示在 60 秒內從 OK 變更至 ALARM，請將 `WaitPeriod` 設為 60 秒。



在圖片中，複合警示會在 t2 從 OK 變更為 ALARM。一個 WaitPeriod 從 t2 開始，在 t8 結束。這使得抑制器警示有時間可以在 t4 將狀態從 OK 變更為 ALARM，然後當 WaitPeriod 在 t8 到期時抑制複合警示的動作。

當複合警示變為 OK 後抑制器警示變為 OK 時，利用 ExtensionPeriod 來補償可能發生的任何延遲。例如，若複合警示在抑制器警報變為 OK 的 60 秒內變更為 OK，請將 ExtensionPeriod 設為 60 秒。



在圖片中，抑制器警示會在 t2 從 ALARM 變更至 OK。一個 ExtensionPeriod 從 t2 開始，在 t8 結束。這使得複合警示有時間可以在 ExtensionPeriod 於 t8 到期之前，從 ALARM 變更為 OK。

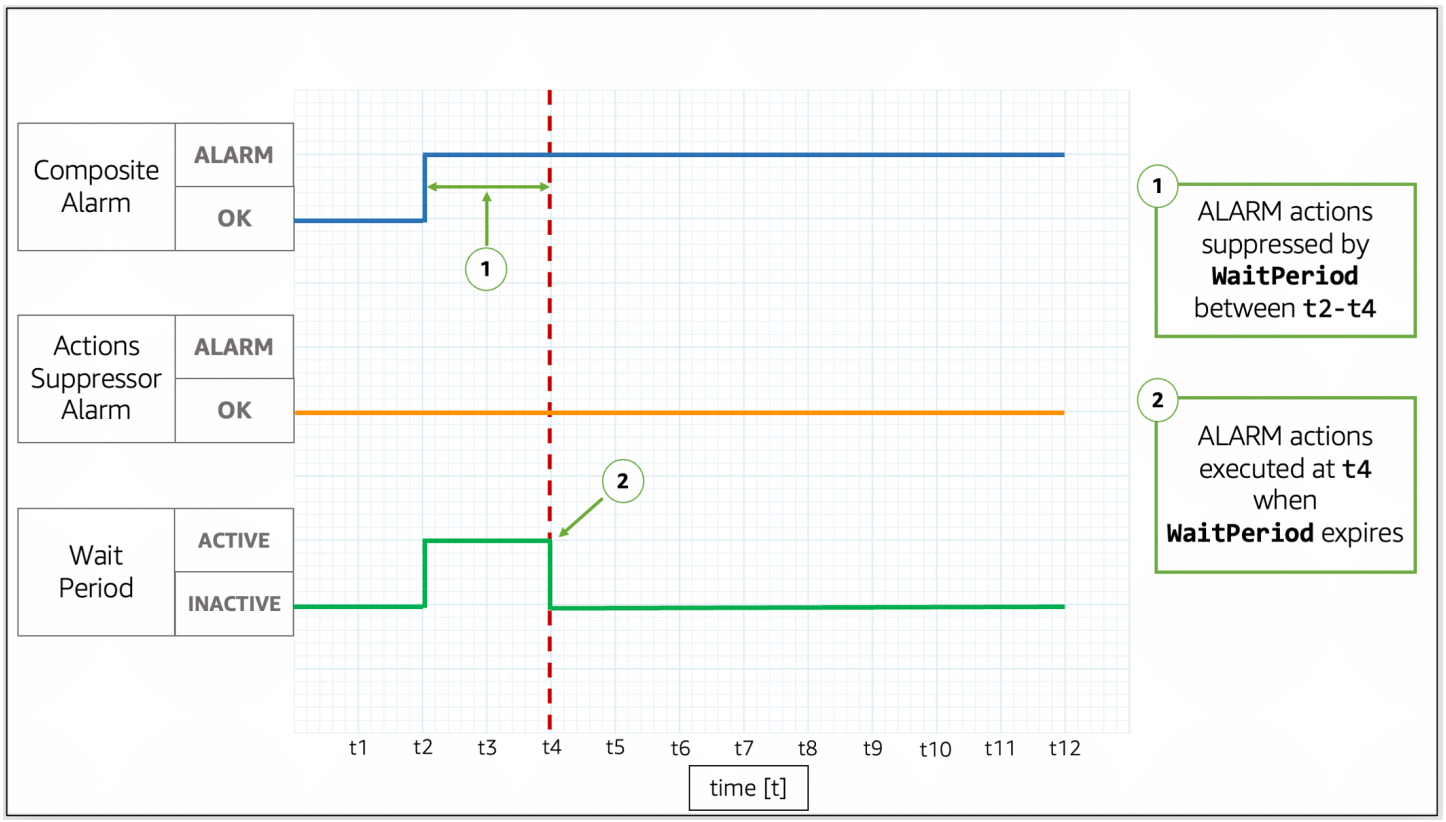
當 WaitPeriod 和 ExtensionPeriod 成為作用中時，複合警示不會採取動作。當 ExtensionPeriod 和 WaitPeriod 成為非作用中時，複合警示會根據其當下狀態採取動作。我們建議您將每個參數的值設定為 60 秒，以便每分鐘 CloudWatch 評估一次公制警示。您可以將參數設置為任何整數 (單位為秒)。

下面的例子更詳細地描述了 WaitPeriod 和 ExtensionPeriod 如何防止複合警報意外採取動作。

> **Note**
>
> 在下列範例中，WaitPeriod 配置為 2 個時間單位，而 ExtensionPeriod 配置為 3 個時間單位。

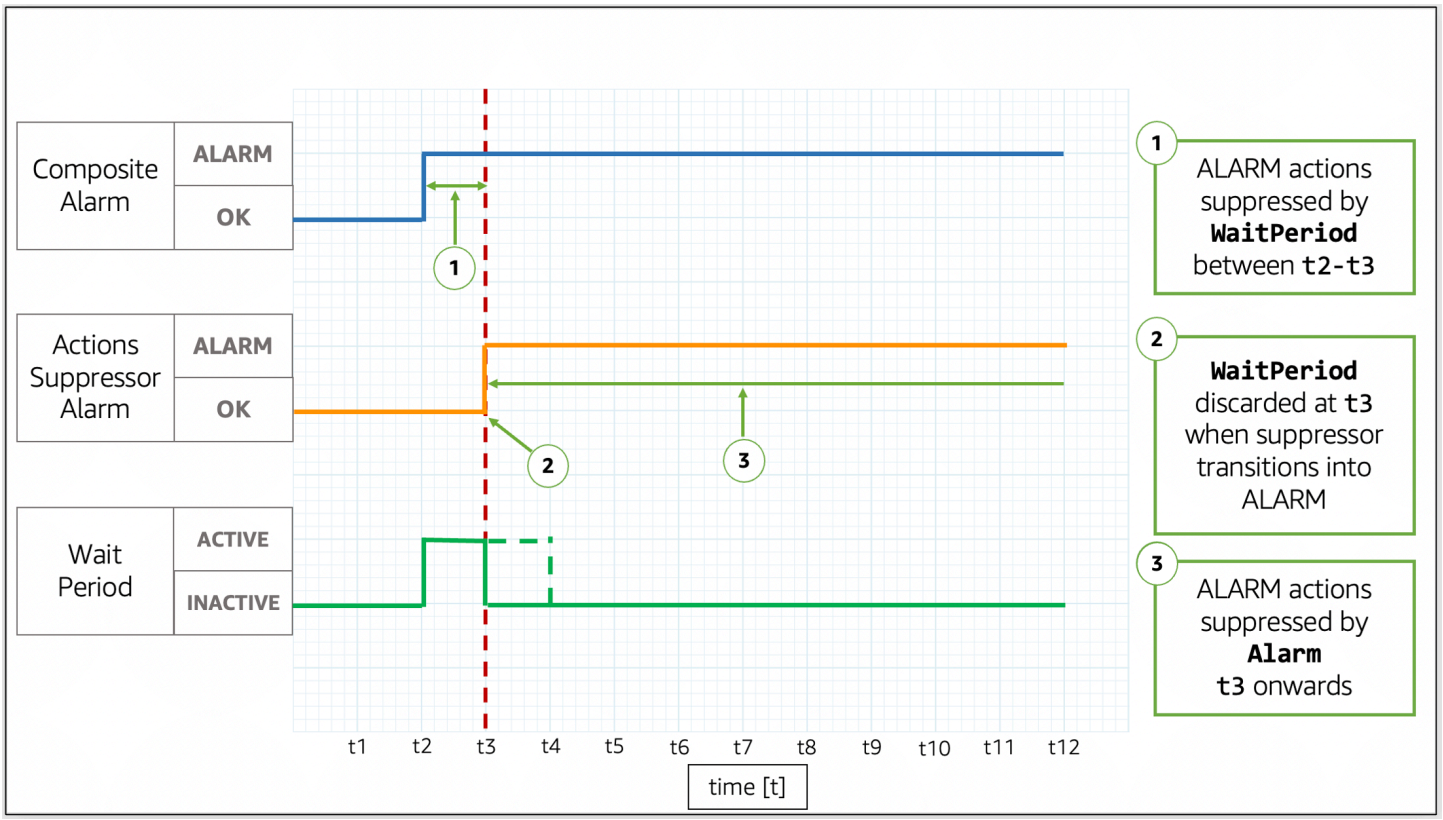
範例

範例 1：動作於 **WaitPeriod** 之後不會被抑制



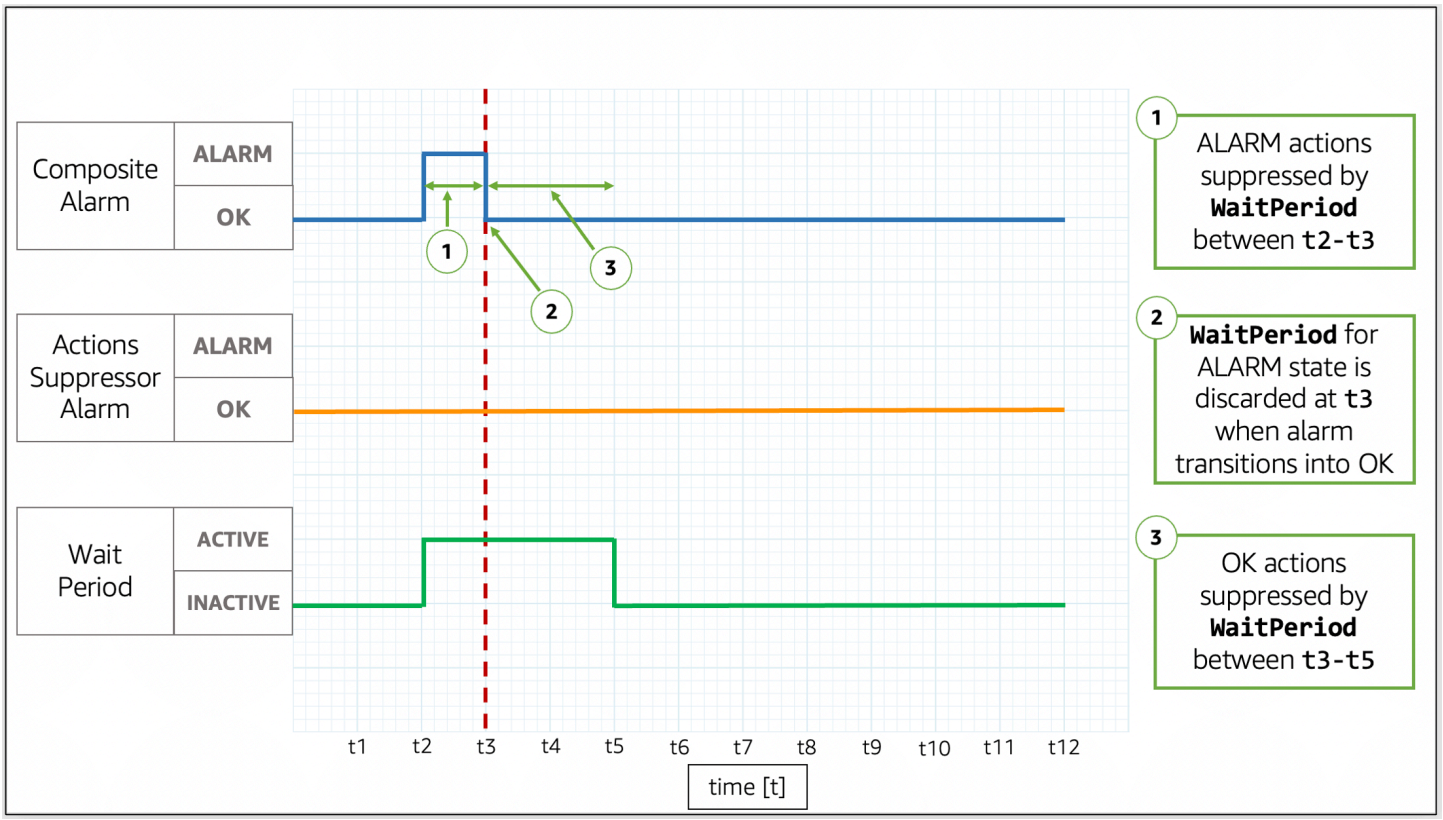
在圖片中，複合警示的狀態會在 t2 從 OK 變更至 ALARM。一個 WaitPeriod 從 t2 開始，在 t4 結束，使它可以防止複合警示採取動作。在 WaitPeriod 於 t4 到期後，複合警示採取其動作，因為抑制器警示仍處於 OK。

範例 2：動作於 **WaitPeriod** 到期前被警示抑制



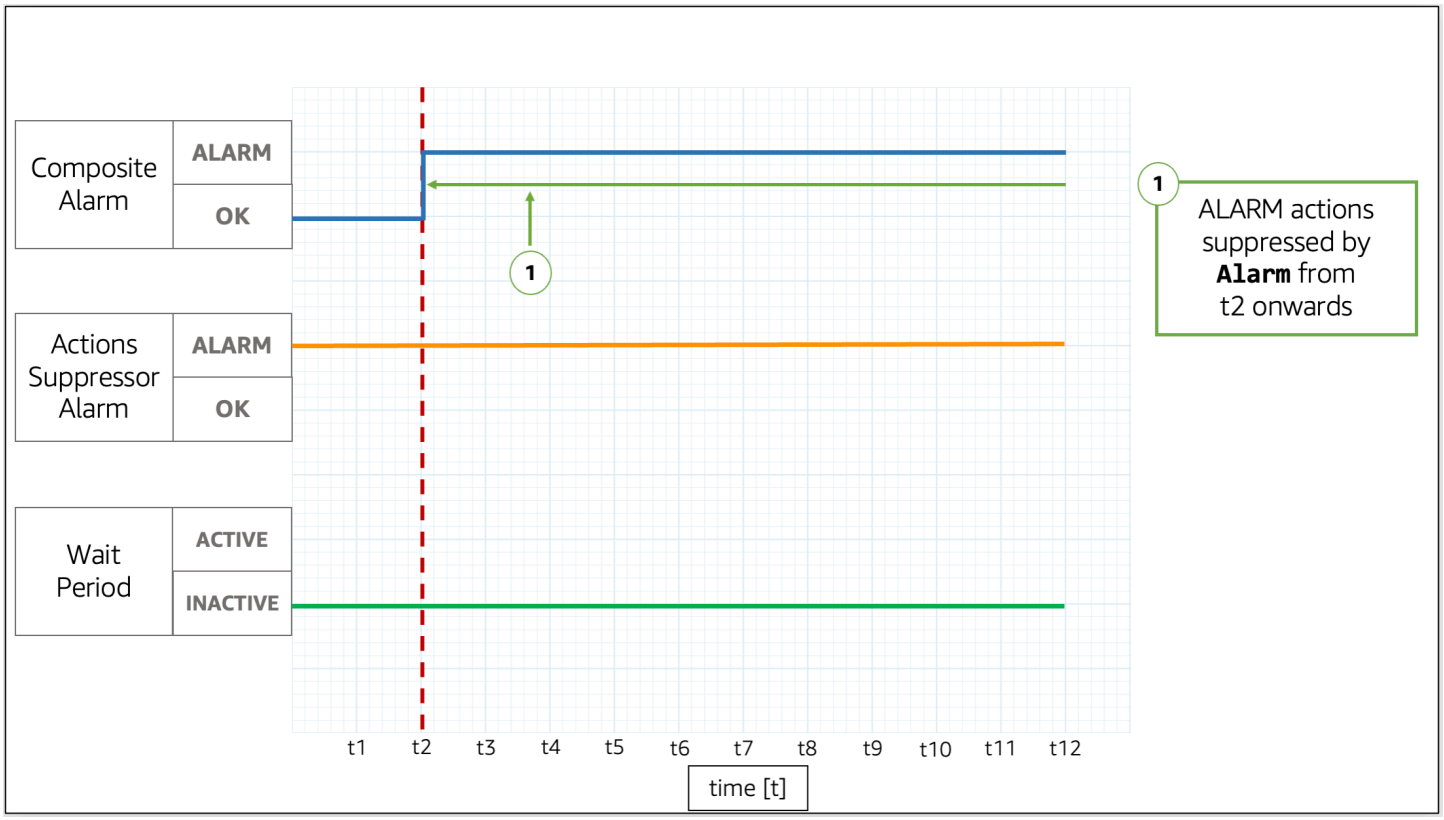
在圖片中，複合警示的狀態會在 t2 從 OK 變更至 ALARM。一個 WaitPeriod 從 t2 開始，在 t4 結束。這使得抑制器警示有時間可以在 t3 將狀態從 OK 變更為 ALARM。由於抑制器警示在 t3 將狀態從 OK 變更為 ALARM，導致從 t2 開始的 WaitPeriod 被捨棄，而抑制器警示現在會阻止複合警示採取動作。

範例 3：動作被 **WaitPeriod** 抑制時的狀態轉換



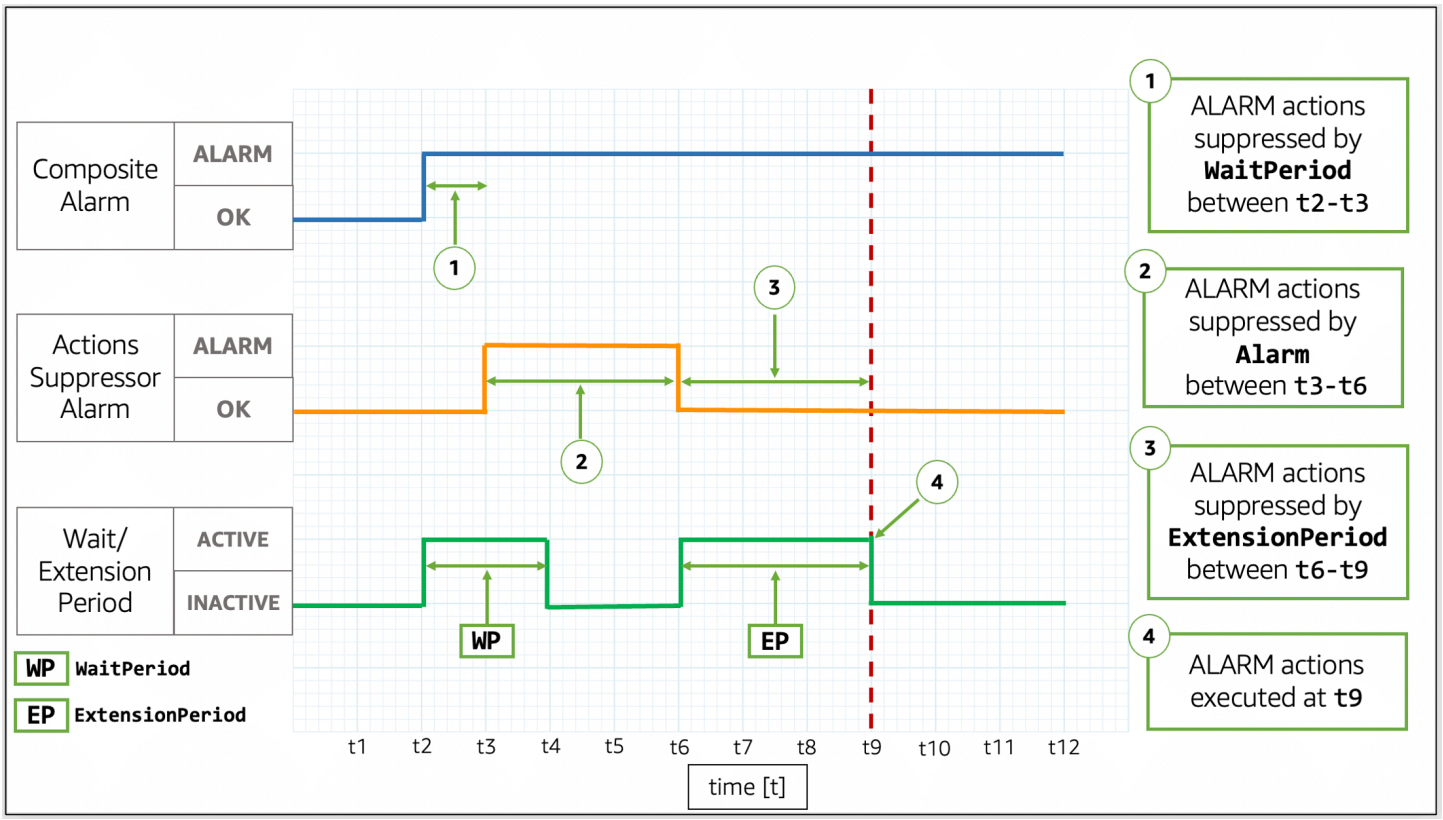
在圖片中，複合警示的狀態會在 t2 從 OK 變更至 ALARM。一個 WaitPeriod 從 t2 開始，在 t4 結束。這使得抑制器警示有時間可以更改狀態。複合警示在 t3 變回 OK，因此從 t2 開始的 WaitPeriod 被捨棄。一個新的 WaitPeriod 從 t3 開始，在 t5 結束。等到新的 WaitPeriod 在 t5 到期後，複合警示便採取其動作。

範例 4：動作被警示抑制時的狀態轉換



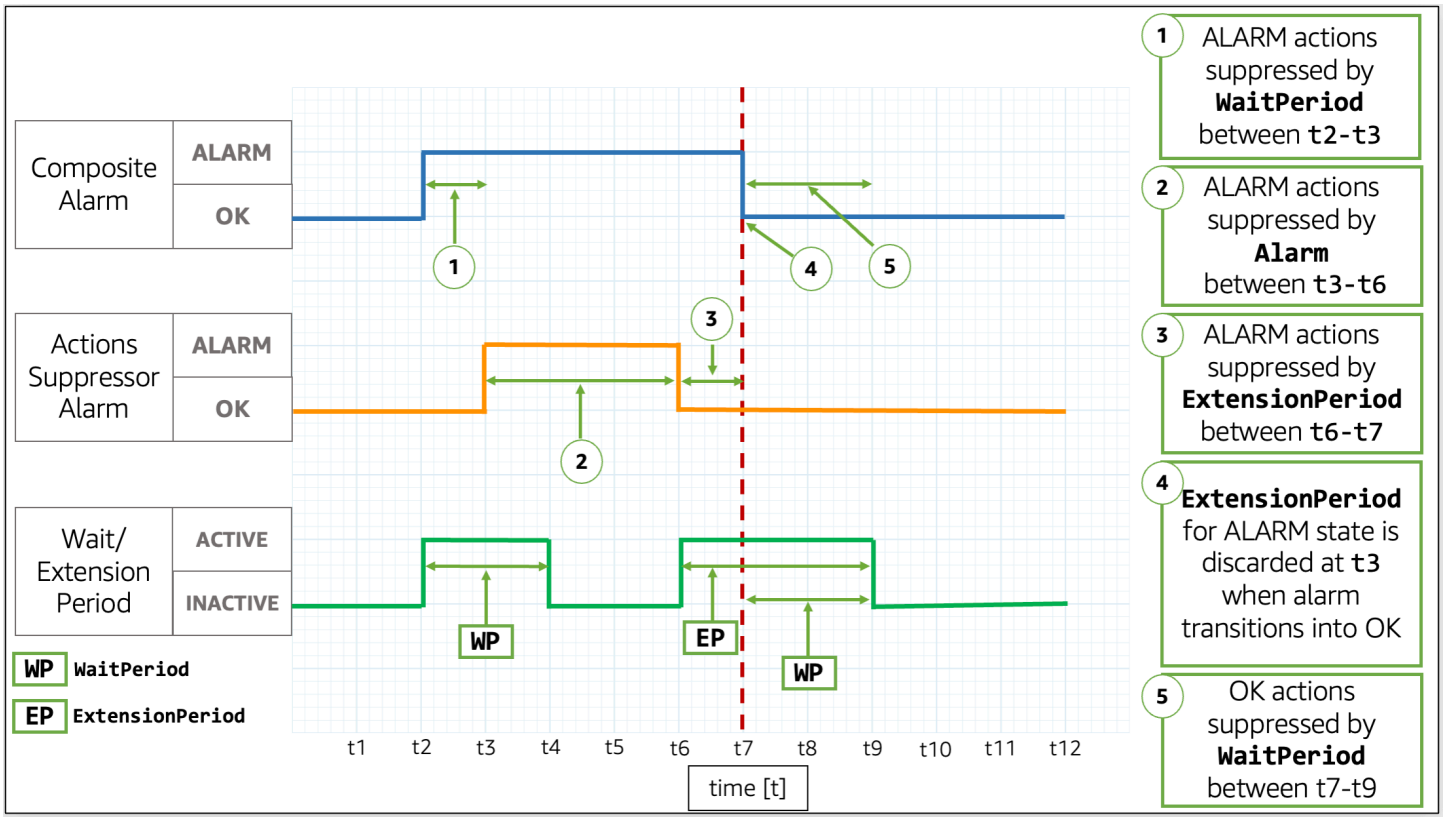
在圖片中，複合警示的狀態會在 t2 從 OK 變更至 ALARM。抑制器警示已經處於 ALARM。抑制器警示會阻止複合警示採取動作。

範例 5：動作於 **`ExtensionPeriod`** 之後不會被抑制



在圖片中，複合警示的狀態會在 t2 從 OK 變更至 ALARM。一個 WaitPeriod 從 t2 開始，在 t4 結束。這使得抑制器警示有時間可以在 t3 將狀態從 OK 變更為 ALARM，然後才抑制複合警示的動作直到 t6。由於抑制器警示在 t3 將狀態從 OK 變更為 ALARM，導致從 t2 開始的 WaitPeriod 被捨棄。在 t6 時，抑制器警示轉為 OK。一個 ExtensionPeriod 從 t6 開始，在 t9 結束。ExtensionPeriod 到期後，複合警示便會採取動作。

範例 6：動作被 **ExtensionPeriod** 抑制時的狀態轉換



在圖片中，複合警示的狀態會在 t2 從 OK 變更至 ALARM。一個 WaitPeriod 從 t2 開始，在 t4 結束。這使得抑制器警示有時間可以在 t3 將狀態從 OK 變更為 ALARM，然後才抑制複合警示的動作直到 t6。由於抑制器警示在 t3 將狀態從 OK 變更為 ALARM，導致從 t2 開始的 WaitPeriod 被捨棄。在 t6，抑制器警示變回 OK。一個 ExtensionPeriod 從 t6 開始，在 t9 結束。當複合警示在 t7 變回 OK，此時 ExtensionPeriod 被捨棄，且一個新的 WaitPeriod 從 t7 開始，在 t9 結束。

> **Tip**
>
> 若您替換了動作抑制器警示，則任何作用中的 WaitPeriod 或 ExtensionPeriod 都會被捨棄。

針對警示變更採取行動

CloudWatch 可以在兩種類型的警報變更時通知使用者：警示變更狀態時，以及更新警示組態時。

當警示評估時，它可能會從一種狀態變更為另一個狀態，例如「ALARM」、「OK」或「INSUFFICIENT_DATA」。這些警示狀態變更可能表示可能發生的事件、恢復正常或無法取得指標。在此情況下，您可以使用下列其中一個選項與使用者互動或通知使用者：

- 可以設定警示以將通知傳送至 SNS 主題 (做為警示動作的一部分)。然後可以為 application-to-application (A2A) 訊息以及 application-to-person (A2P) 通知設定 SNS 主題，包括電子郵件通知和 SMS 等通道。您為 SNS 主題定義的所有目的地都會收到警示通知。如需詳細資訊，請參閱 Amazon SNS 事件目的地。
- 您可以設定警示狀態變更事件的通知。 AWS 使用者通知提供設定此類通知的原生方式，並且是建議使用的方法。

此外， EventBridge 每當警示狀態變更，以及建立、刪除或更新警示時，都 CloudWatch 會傳送事件至 Amazon。您可以編寫 EventBridge 規則以採取行動或在 EventBridge 收到這些事件時收到通知。

主題

- 在警示變更時通知使用者
- 警報事件和 EventBridge

在警示變更時通知使用者

本節說明如何使用使用 AWS 者通知或 Amazon 簡單通知服務，讓使用者收到警示變更的通知。

設定 AWS 使用者通知

您可以使用使用AWS 者通知來設定傳遞管道，以接收 CloudWatch 警示狀態變更和組態變更事件的通知。當事件符合您指定的規則時，便會收到通知。您可以透過多個管道接收事件通知，包括電子郵件、AWS Chatbot 聊天通知或 AWS 主控台行動應用程式推送通知。您也可以在主控台通知中心查看通知。使用者通知支援彙總，可減少您在特定事件期間收到的通知數目。

您使用「 AWS 使用者通知」建立的通知組態不會計入您可以針對每個目標警示狀態設定的動作數目上限。當 AWS 使用者通知符合發送給 Amazon 的事件時 EventBridge，它會傳送帳戶和選定區域中所有警示的通知，除非您指定允許清單或拒絕清單特定警示或模式的進階篩選器。

下列進階篩選條件範例會在名為 `ServerCpuTooHigh` 的警示上比對從 OK 到 ALARM 的警示狀態變更。

```
{
"detail": {
    "alarmName": ["ServerCpuTooHigh"],
    "previousState": { "value": ["OK"] },
    "state": { "value": ["ALARM"] }
  }
```

```
}
```

您可以使用 EventBridge 事件中警示所發佈的任何屬性來建立篩選器。如需詳細資訊，請參閱 警報事件和 EventBridge。

設定 Amazon SNS 通知

您可以使用 Amazon 簡易通知服務來傳送 application-to-application (A2A) 簡訊和 application-to-person (A2P) 簡訊，包括行動文字簡訊 (SMS) 和電子郵件訊息。如需詳細資訊，請參閱 Amazon SNS 事件目的地。

對於警示可能發生的每個狀態，您可以設定警示以將訊息傳送至 SNS 主題。針對指定警示上的某個狀態設定的每個 Amazon SNS 主題都會計入您可為該警示和狀態設定的動作數目上限。您可以從帳戶中的任何警示傳送訊息至相同的 Amazon SNS 主題，並針對應用程式 (A2A) 和個人 (A2P) 消費者使用相同的 Amazon SNS 主題。由於此組態是在警示層級完成，因此只有您設定的警示會傳送訊息至選取的 Amazon SNS 主題。

首先，請建立主題，然後訂閱它。您可以選擇性將測試訊息發佈到主題。如需範例，請參閱使用設定 Amazon SNS 主題 AWS Management Console。或者，如需詳細資訊，請參閱 Amazon SNS 入門。

或者，如果您打算使用建立 CloudWatch 警示，您可以略過此程序，因為您可以在建立警示時建立主題。 AWS Management Console

建立警示時，您可以針對 CloudWatch 警示進入的任何目標狀態新增動作。針對您要收到通知的狀態新增 Amazon SNS 通知，然後選取您在上一個步驟中建立的 Amazon SNS 主題，以在警示進入所選狀態時傳送電子郵件通知。

> **Note**
>
> 建立 Amazon SNS 主題時，您可以選擇將其設為標準主題或 FIFO 主題。 CloudWatch 保證將所有警示通知發佈至這兩種類型的主題。但是，即使您使用 FIFO 主題，在極少數情況下也會按順序將通知 CloudWatch 發送到主題。如果您使用 FIFO 主題，則警示會將警示通知的訊息群組 ID 設定為警示 ARN 的雜湊。

防止混淆代理人問題

為了避免跨服務混淆的副安全問題，建議您使用 Amazon SNS 資源政策中的`aws:SourceArn`和`aws:SourceAccount`全域條件金鑰，以授 CloudWatch 予存取 Amazon SNS 資源的權限。

下列範例資源策略使用aws:SourceArn條件索引鍵來縮小僅供指定帳號中 CloudWatch 警示使用的SNS:Publish權限範圍。

```
{
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "cloudwatch.amazonaws.com"
        },
        "Action": "SNS:Publish",
        "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
        "Condition": {
            "ArnLike": {
                "aws:SourceArn": "arn:aws:cloudwatch:us-east-2:111122223333:alarm:*"
            },
            "StringEquals": {
                "aws:SourceAccount": "111122223333"
            }
        }
    }]
}
```

如果警示 ARN 包含任何非 ASCII 字元，請僅使用 aws:SourceAccount 全域條件鍵來限制許可。

使用設定 Amazon SNS 主題 AWS Management Console

首先，請建立主題，然後訂閱它。您可以選擇性將測試訊息發佈到主題。

建立 SNS 主題

1. 在 https://console.aws.amazon.com/sns/v3/home 開啟 Amazon SNS 主控台。
2. 在 Amazon SNS 儀表板上，在 Common actions (常見的動作) 下，選擇 Create Topic (建立主題)。
3. 在 Create new topic (建立新主題) 對話方塊中，針對 Topic name (主題名稱)，輸入主題的名稱 (例如 **my-topic**)。
4. 請選擇建立主題。
5. 複製下一個任務的 Topic ARN (主題 ARN) (例如，arn:aws:sns:us-east-1:111122223333:my-topic)。

若要訂閱 SNS 主題

1. 在 https://console.aws.amazon.com/sns/v3/home 開啟 Amazon SNS 主控台。
2. 在導覽面板中依序選擇 Subscriptions (訂閱) 與 Create subscription (建立訂閱)。
3. 在 Create subscription (建立訂閱) 對話方塊中，對於 Topic ARN (主題 ARN)，貼上在之前工作所建立的主題 ARN。
4. 對於通訊協定，選擇電子郵件。
5. 針對 Endpoint (端點)，輸入您可以用來接收通知的電子郵件地址，然後選擇 Create subscription (建立訂閱)。
6. 在您的電子郵件應用程式中，開啟「 AWS 通知」中的訊息並確認您的訂閱。

   您的 Web 瀏覽器顯示自 Amazon SNS 的確認回覆。

若要將文字訊息發佈至 SNS 主題

1. 在 https://console.aws.amazon.com/sns/v3/home 開啟 Amazon SNS 主控台。
2. 在導覽窗格中，選擇主題。
3. 在 Topics (主題) 頁面上，選取主題然後選擇 Publish to topic (發佈到主題)。
4. 在 Publish a message (發佈訊息) 頁面中，針對 Subject (主旨)，輸入訊息的主旨行，然後針對 Message (訊息)，輸入簡短訊息。
5. 選擇 Publish Message (發佈訊息)。
6. 檢查電子郵件以確認您已收到訊息。

使用設定 SNS 主題 AWS CLI

首先，您可以建立一個 SNS 主題，然後直接將訊息發佈到主題來測試您已正確設定。

設定 SNS 主題

1. 使用 create-topic 命令建立主題，如下所示。

   ```
   aws sns create-topic --name my-topic
   ```

   Amazon SNS 會以下列格式傳回主題 ARN：

   ```
   {
   ```

```
    "TopicArn": "arn:aws:sns:us-east-1:111122223333:my-topic"
}
```

2. 使用 subscribe 命令來訂閱您的電子郵件地址至該主題。如果訂閱請求成功，您會收到一封確認電子郵件訊息。

```
aws sns subscribe --topic-arn arn:aws:sns:us-east-1:111122223333:my-topic --
protocol email --notification-endpoint my-email-address
```

Amazon SNS 傳回下列回應：

```
{
    "SubscriptionArn": "pending confirmation"
}
```

3. 在您的電子郵件應用程式中，開啟「AWS 通知」中的訊息並確認您的訂閱。

您的 Web 瀏覽器顯示自 Amazon Simple Notification Service 的確認回覆。

4. 使用list-subscriptions-by-topic命令檢查訂閱。

```
aws sns list-subscriptions-by-topic --topic-arn arn:aws:sns:us-
east-1:111122223333:my-topic
```

Amazon SNS 傳回下列回應：

```
{
  "Subscriptions": [
    {
        "Owner": "111122223333",
        "Endpoint": "me@mycompany.com",
        "Protocol": "email",
        "TopicArn": "arn:aws:sns:us-east-1:111122223333:my-topic",
        "SubscriptionArn": "arn:aws:sns:us-east-1:111122223333:my-topic:64886986-
bf10-48fb-a2f1-dab033aa67a3"
    }
  ]
}
```

5. (選用) 使用 publish 命令將測試訊息發佈到該主題。

```
aws sns publish --message "Verification" --topic arn:aws:sns:us-
east-1:111122223333:my-topic
```

Amazon SNS 傳回下列回應：

```
{
    "MessageId": "42f189a0-3094-5cf6-8fd7-c2dde61a4d7d"
}
```

6. 檢查電子郵件以確認您已收到訊息。

警報事件和 EventBridge

CloudWatch EventBridge 每當建立、更新、刪除或變更 CloudWatch 警示狀態時，都會傳送事件至 Amazon。您可以使用 EventBridge和這些事件來撰寫規則，以便在警示變更狀態時採取動作 (例如通知您)。如需詳細資訊，請參閱什麼是 Amazon EventBridge？

CloudWatch 保證警報狀態變更事件傳遞給 EventBridge。

範例事件來源 CloudWatch

本節包含來自的範例事件 CloudWatch。

單一指標警示的狀態變更

```
{
    "version": "0",
    "id": "c4c1c1c9-6542-e61b-6ef0-8c4d36933a92",
    "detail-type": "CloudWatch Alarm State Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2019-10-02T17:04:40Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServerCpuTooHigh"
    ],
    "detail": {
        "alarmName": "ServerCpuTooHigh",
        "configuration": {
            "description": "Goes into alarm when server CPU utilization is too high!",
            "metrics": [
```

```
                {
                    "id": "30b6c6b2-a864-43a2-4877-c09a1afc3b87",
                    "metricStat": {
                        "metric": {
                            "dimensions": {
                                "InstanceId": "i-12345678901234567"
                            },
                            "name": "CPUUtilization",
                            "namespace": "AWS/EC2"
                        },
                        "period": 300,
                        "stat": "Average"
                    },
                    "returnData": true
                }
            ]
        },
        "previousState": {
            "reason": "Threshold Crossed: 1 out of the last 1 datapoints
 [0.0666851903306472 (01/10/19 13:46:00)] was not greater than the threshold (50.0)
 (minimum 1 datapoint for ALARM -> OK transition).",
            "reasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2019-10-01T13:56:40.985+0000\",\"startDate\":\"2019-10-01T13:46:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[0.0666851903306472],
\"threshold\":50.0}",
            "timestamp": "2019-10-01T13:56:40.987+0000",
            "value": "OK"
        },
        "state": {
            "reason": "Threshold Crossed: 1 out of the last 1 datapoints
 [99.50160229693434 (02/10/19 16:59:00)] was greater than the threshold (50.0) (minimum
 1 datapoint for OK -> ALARM transition).",
            "reasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2019-10-02T17:04:40.985+0000\",\"startDate\":\"2019-10-02T16:59:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[99.50160229693434],
\"threshold\":50.0}",
            "timestamp": "2019-10-02T17:04:40.989+0000",
            "value": "ALARM"
        }
    }
}
```

指標數學警示的狀態變更

```
{
    "version": "0",
    "id": "2dde0eb1-528b-d2d5-9ca6-6d590caf2329",
    "detail-type": "CloudWatch Alarm State Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2019-10-02T17:20:48Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:TotalNetworkTrafficTooHigh"
    ],
    "detail": {
        "alarmName": "TotalNetworkTrafficTooHigh",
        "configuration": {
            "description": "Goes into alarm if total network traffic exceeds 10Kb",
            "metrics": [
                {
                    "expression": "SUM(METRICS())",
                    "id": "e1",
                    "label": "Total Network Traffic",
                    "returnData": true
                },
                {
                    "id": "m1",
                    "metricStat": {
                        "metric": {
                            "dimensions": {
                                "InstanceId": "i-12345678901234567"
                            },
                            "name": "NetworkIn",
                            "namespace": "AWS/EC2"
                        },
                        "period": 300,
                        "stat": "Maximum"
                    },
                    "returnData": false
                },
                {
                    "id": "m2",
                    "metricStat": {
                        "metric": {
                            "dimensions": {
                                "InstanceId": "i-12345678901234567"
```

```
                            },
                            "name": "NetworkOut",
                            "namespace": "AWS/EC2"
                        },
                        "period": 300,
                        "stat": "Maximum"
                    },
                    "returnData": false
                }
            ]
        },
        "previousState": {
            "reason": "Unchecked: Initial alarm creation",
            "timestamp": "2019-10-02T17:20:03.642+0000",
            "value": "INSUFFICIENT_DATA"
        },
        "state": {
            "reason": "Threshold Crossed: 1 out of the last 1 datapoints [45628.0
 (02/10/19 17:10:00)] was greater than the threshold (10000.0) (minimum 1 datapoint for
 OK -> ALARM transition).",
            "reasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2019-10-02T17:20:48.551+0000\",\"startDate\":\"2019-10-02T17:10:00.000+0000\",
\"period\":300,\"recentDatapoints\":[45628.0],\"threshold\":10000.0}",
            "timestamp": "2019-10-02T17:20:48.554+0000",
            "value": "ALARM"
        }
    }
}
```

異常偵測警示的狀態變更

```
{
    "version": "0",
    "id": "daafc9f1-bddd-c6c9-83af-74971fcfc4ef",
    "detail-type": "CloudWatch Alarm State Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2019-10-03T16:00:04Z",
    "region": "us-east-1",
    "resources": ["arn:aws:cloudwatch:us-east-1:123456789012:alarm:EC2 CPU Utilization
 Anomaly"],
    "detail": {
        "alarmName": "EC2 CPU Utilization Anomaly",
```

```
        "state": {
            "value": "ALARM",
            "reason": "Thresholds Crossed: 1 out of the last 1 datapoints [0.0
 (03/10/19 15:58:00)] was less than the lower thresholds [0.020599444741798756] or
 greater than the upper thresholds [0.3006915352732461] (minimum 1 datapoint for OK ->
 ALARM transition).",
            "reasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2019-10-03T16:00:04.650+0000\",\"startDate\":\"2019-10-03T15:58:00.000+0000\",
\"period\":60,\"recentDatapoints\":[0.0],\"recentLowerThresholds\":
[0.020599444741798756],\"recentUpperThresholds\":[0.3006915352732461]}",
            "timestamp": "2019-10-03T16:00:04.653+0000"
        },
        "previousState": {
            "value": "OK",
            "reason": "Thresholds Crossed: 1 out of the last 1 datapoints
 [0.166666666664241 (03/10/19 15:57:00)] was not less than the lower thresholds
 [0.0206719426210418] or not greater than the upper thresholds [0.30076870222143803]
 (minimum 1 datapoint for ALARM -> OK transition).",
            "reasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2019-10-03T15:59:04.670+0000\",\"startDate\":\"2019-10-03T15:57:00.000+0000\",
\"period\":60,\"recentDatapoints\":[0.166666666664241],\"recentLowerThresholds\":
[0.0206719426210418],\"recentUpperThresholds\":[0.30076870222143803]}",
            "timestamp": "2019-10-03T15:59:04.672+0000"
        },
        "configuration": {
            "description": "Goes into alarm if CPU Utilization is out of band",
            "metrics": [{
                "id": "m1",
                "metricStat": {
                    "metric": {
                        "namespace": "AWS/EC2",
                        "name": "CPUUtilization",
                        "dimensions": {
                            "InstanceId": "i-12345678901234567"
                        }
                    },
                    "period": 60,
                    "stat": "Average"
                },
                "returnData": true
            }, {
                "id": "ad1",
                "expression": "ANOMALY_DETECTION_BAND(m1, 0.8)",
                "label": "CPUUtilization (expected)",
```

```
                "returnData": true
            }]
        }
    }
}
```

帶有抑制器警示的複合警示的狀態變化

```
{
    "version": "0",
    "id": "d3dfc86d-384d-24c8-0345-9f7986db0b80",
    "detail-type": "CloudWatch Alarm State Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-07-22T15:57:45Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
    ],
    "detail": {
        "alarmName": "ServiceAggregatedAlarm",
        "state": {
            "actionsSuppressedBy": "WaitPeriod",
            "actionsSuppressedReason": "Actions suppressed by WaitPeriod",
            "value": "ALARM",
            "reason": "arn:aws:cloudwatch:us-
east-1:123456789012:alarm:SuppressionDemo.EventBridge.FirstChild transitioned to ALARM
 at Friday 22 July, 2022 15:57:45 UTC",
            "reasonData": "{\"triggeringAlarms\":[{\"arn\":\"arn:aws:cloudwatch:us-
east-1:123456789012:alarm:ServerCpuTooHigh\",\"state\":{\"value\":\"ALARM\",\"timestamp
\":\"2022-07-22T15:57:45.394+0000\"}}]}",
            "timestamp": "2022-07-22T15:57:45.394+0000"
        },
        "previousState": {
            "value": "OK",
            "reason": "arn:aws:cloudwatch:us-
east-1:123456789012:alarm:SuppressionDemo.EventBridge.Main was created and its alarm
 rule evaluates to OK",
            "reasonData": "{\"triggeringAlarms\":[{\"arn\":\"arn:aws:cloudwatch:us-
east-1:123456789012:alarm:TotalNetworkTrafficTooHigh\",\"state\":{\"value\":\"OK\",
\"timestamp\":\"2022-07-14T16:28:57.770+0000\"}},{\"arn\":\"arn:aws:cloudwatch:us-
east-1:123456789012:alarm:ServerCpuTooHigh\",\"state\":{\"value\":\"OK\",\"timestamp\":
\"2022-07-14T16:28:54.191+0000\"}}]}",
```

```
            "timestamp": "2022-07-22T15:56:14.552+0000"
        },
        "configuration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR
 ALARM(TotalNetworkTrafficTooHigh)",
            "actionsSuppressor": "ServiceMaintenanceAlarm",
            "actionsSuppressorWaitPeriod": 120,
            "actionsSuppressorExtensionPeriod": 180
        }
    }
}
```

複合警示的建立

```
{

    "version": "0",
    "id": "91535fdd-1e9c-849d-624b-9a9f2b1d09d0",
    "detail-type": "CloudWatch Alarm Configuration Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-03-03T17:06:22Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
    ],
    "detail": {
        "alarmName": "ServiceAggregatedAlarm",
        "operation": "create",
        "state": {
            "value": "INSUFFICIENT_DATA",
            "timestamp": "2022-03-03T17:06:22.289+0000"
        },
        "configuration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR
 ALARM(TotalNetworkTrafficTooHigh)",
            "alarmName": "ServiceAggregatedAlarm",
            "description": "Aggregated monitor for instance",
            "actionsEnabled": true,
            "timestamp": "2022-03-03T17:06:22.289+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
```

```
        }
    }
}
```

建立帶有抑制器警示的複合警示

```
{
    "version": "0",
    "id": "454773e1-09f7-945b-aa2c-590af1c3f8e0",
    "detail-type": "CloudWatch Alarm Configuration Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-07-14T13:59:46Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
    ],
    "detail": {
        "alarmName": "ServiceAggregatedAlarm",
        "operation": "create",
        "state": {
            "value": "INSUFFICIENT_DATA",
            "timestamp": "2022-07-14T13:59:46.425+0000"
        },
        "configuration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR
 ALARM(TotalNetworkTrafficTooHigh)",
            "actionsSuppressor": "ServiceMaintenanceAlarm",
            "actionsSuppressorWaitPeriod": 120,
            "actionsSuppressorExtensionPeriod": 180,
            "alarmName": "ServiceAggregatedAlarm",
            "actionsEnabled": true,
            "timestamp": "2022-07-14T13:59:46.425+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        }
    }
}
```

指標警示的更新

```
{
```

```
"version": "0",
"id": "bc7d3391-47f8-ae47-f457-1b4d06118d50",
"detail-type": "CloudWatch Alarm Configuration Change",
"source": "aws.cloudwatch",
"account": "123456789012",
"time": "2022-03-03T17:06:34Z",
"region": "us-east-1",
"resources": [
    "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServerCpuTooHigh"
],
"detail": {
    "alarmName": "ServerCpuTooHigh",
    "operation": "update",
    "state": {
        "value": "INSUFFICIENT_DATA",
        "timestamp": "2022-03-03T17:06:13.757+0000"
    },
    "configuration": {
        "evaluationPeriods": 1,
        "threshold": 80,
        "comparisonOperator": "GreaterThanThreshold",
        "treatMissingData": "ignore",
        "metrics": [
            {
                "id": "86bfa85f-b14c-ebf7-8916-7da014ce23c0",
                "metricStat": {
                    "metric": {
                        "namespace": "AWS/EC2",
                        "name": "CPUUtilization",
                        "dimensions": {
                            "InstanceId": "i-12345678901234567"
                        }
                    },
                    "period": 300,
                    "stat": "Average"
                },
                "returnData": true
            }
        ],
        "alarmName": "ServerCpuTooHigh",
        "description": "Goes into alarm when server CPU utilization is too high!",
        "actionsEnabled": true,
        "timestamp": "2022-03-03T17:06:34.267+0000",
```

```
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        },
        "previousConfiguration": {
            "evaluationPeriods": 1,
            "threshold": 70,
            "comparisonOperator": "GreaterThanThreshold",
            "treatMissingData": "ignore",
            "metrics": [
                {
                    "id": "d6bfa85f-893e-b052-a58b-4f9295c9111a",
                    "metricStat": {
                        "metric": {
                            "namespace": "AWS/EC2",
                            "name": "CPUUtilization",
                            "dimensions": {
                                "InstanceId": "i-12345678901234567"
                            }
                        },
                        "period": 300,
                        "stat": "Average"
                    },
                    "returnData": true
                }
            ],
            "alarmName": "ServerCpuTooHigh",
            "description": "Goes into alarm when server CPU utilization is too high!",
            "actionsEnabled": true,
            "timestamp": "2022-03-03T17:06:13.757+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        }
    }
}
```

更新帶有抑制器警示的複合警示

```
{
    "version": "0",
    "id": "4c6f4177-6bd5-c0ca-9f05-b4151c54568b",
    "detail-type": "CloudWatch Alarm Configuration Change",
```

```
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-07-14T13:59:56Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
    ],
    "detail": {
        "alarmName": "ServiceAggregatedAlarm",
        "operation": "update",
        "state": {
            "actionsSuppressedBy": "WaitPeriod",
            "value": "ALARM",
            "timestamp": "2022-07-14T13:59:46.425+0000"
        },
        "configuration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR
 ALARM(TotalNetworkTrafficTooHigh)",
            "actionsSuppressor": "ServiceMaintenanceAlarm",
            "actionsSuppressorWaitPeriod": 120,
            "actionsSuppressorExtensionPeriod": 360,
            "alarmName": "ServiceAggregatedAlarm",
            "actionsEnabled": true,
            "timestamp": "2022-07-14T13:59:56.290+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        },
        "previousConfiguration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR
 ALARM(TotalNetworkTrafficTooHigh)",
            "actionsSuppressor": "ServiceMaintenanceAlarm",
            "actionsSuppressorWaitPeriod": 120,
            "actionsSuppressorExtensionPeriod": 180,
            "alarmName": "ServiceAggregatedAlarm",
            "actionsEnabled": true,
            "timestamp": "2022-07-14T13:59:46.425+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        }
    }
}
```

刪除指標數學警示

```
{

    "version": "0",
    "id": "f171d220-9e1c-c252-5042-2677347a83ed",
    "detail-type": "CloudWatch Alarm Configuration Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-03-03T17:07:13Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:TotalNetworkTrafficTooHigh"
    ],
    "detail": {
        "alarmName": "TotalNetworkTrafficTooHigh",
        "operation": "delete",
        "state": {
            "value": "INSUFFICIENT_DATA",
            "timestamp": "2022-03-03T17:06:17.672+0000"
        },
        "configuration": {
            "evaluationPeriods": 1,
            "threshold": 10000,
            "comparisonOperator": "GreaterThanThreshold",
            "treatMissingData": "ignore",
            "metrics": [{
                    "id": "m1",
                    "metricStat": {
                        "metric": {
                            "namespace": "AWS/EC2",
                            "name": "NetworkIn",
                            "dimensions": {
                                "InstanceId": "i-12345678901234567"
                            }
                        },
                        "period": 300,
                        "stat": "Maximum"
                    },
                    "returnData": false
                },
                {
                    "id": "m2",
                    "metricStat": {
```

```
                        "metric": {
                            "namespace": "AWS/EC2",
                            "name": "NetworkOut",
                            "dimensions": {
                                "InstanceId": "i-12345678901234567"
                            }
                        },
                        "period": 300,
                        "stat": "Maximum"
                    },
                    "returnData": false
                },
                {
                    "id": "e1",
                    "expression": "SUM(METRICS())",
                    "label": "Total Network Traffic",
                    "returnData": true
                }
            ],
            "alarmName": "TotalNetworkTrafficTooHigh",
            "description": "Goes into alarm if total network traffic exceeds 10Kb",
            "actionsEnabled": true,
            "timestamp": "2022-03-03T17:06:17.672+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []

        }
    }
}
```

刪除帶有抑制器警示的複合警示

```
{
    "version": "0",
    "id": "e34592a1-46c0-b316-f614-1b17a87be9dc",
    "detail-type": "CloudWatch Alarm Configuration Change",
    "source": "aws.cloudwatch",
    "account": "123456789012",
    "time": "2022-07-14T14:00:01Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:cloudwatch:us-east-1:123456789012:alarm:ServiceAggregatedAlarm"
```

```
    ],
    "detail": {
        "alarmName": "ServiceAggregatedAlarm",
        "operation": "delete",
        "state": {
            "actionsSuppressedBy": "WaitPeriod",
            "value": "ALARM",
            "timestamp": "2022-07-14T13:59:46.425+0000"
        },
        "configuration": {
            "alarmRule": "ALARM(ServerCpuTooHigh) OR
 ALARM(TotalNetworkTrafficTooHigh)",
            "actionsSuppressor": "ServiceMaintenanceAlarm",
            "actionsSuppressorWaitPeriod": 120,
            "actionsSuppressorExtensionPeriod": 360,
            "alarmName": "ServiceAggregatedAlarm",
            "actionsEnabled": true,
            "timestamp": "2022-07-14T13:59:56.290+0000",
            "okActions": [],
            "alarmActions": [],
            "insufficientDataActions": []
        }
    }
}
```

管理警示

編輯或刪除 CloudWatch 鬧鐘

您可以編輯或刪除現有警示。

您無法變更現有警示的名稱。您可以複製警示並給予新的警示不同名稱。若要複製警示，請在警示清單中選取警示名稱旁的核取方塊，然後選擇 Action (動作)、Copy (複製)。

編輯警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Alarms (警示)、All alarms (所有警示)。
3. 選擇警示的名稱。
4. 若要新增或移除標籤，請選擇標籤索引標籤，然後選擇管理標籤。

5. 若要編輯警示的其他部分，請選擇動作、編輯。

   Specify metric and conditions (指定指標和條件) 頁面隨即出現，顯示您所選取指標和統計資料的圖形及其他資訊。

6. 若要變更指標，請選擇 Edit (編輯)、All metrics (所有指標) 標籤，然後執行以下其中一項作業：

   - 選擇包含您所需指標的服務命名空間。繼續選擇出現的選項以縮減選擇。出現指標清單時，請選取您所需指標旁的核取方塊。
   - 在搜尋方塊中，輸入指標名稱、維度或資源 ID，然後按 Enter。然後，選擇其中一個結果，並繼續進行直到出現指標清單為止。選取您所需指標旁的核取方塊。

   選擇選取指標。

7. 若要變更警示的其他方面，請選擇適當的選項。若要變更必須要有多少資料點違規，才會使警示移至 ALARM 狀態，或是變更處理遺失資料的方式，請選擇 Additional configuration (其他組態)。

8. 選擇下一步。

9. 在 Notification (通知)、Auto Scaling action (Auto Scaling 動作) 及 EC2 action (EC2 動作) 下，選擇性地編輯觸發警示時的動作。然後選擇下一步。

10. 您也可以選用地變更警示描述。

    您無法變更現有警示的名稱。您可以複製警示並給予新的警示不同名稱。若要複製警示，請在警示清單中選取警示名稱旁的核取方塊，然後選擇 Action (動作)、Copy (複製)。

11. 選擇下一步。

12. 在 Preview and create (預覽並建立) 下，確認資訊和條件是您希望的內容，然後選擇 Update alarm (更新警示)。

若要更新使用 Amazon SNS 主控台建立的電子郵件通知清單

1. 在 https://console.aws.amazon.com/sns/v3/home 開啟 Amazon SNS 主控台。
2. 在導覽窗格中，選擇 Topics (主題)，然後選取通知清單 (主題) 的 ARN。
3. 執行以下任意一項：

   - 若要新增電子郵件地址，選擇 Create subscription (建立訂閱)。對於通訊協定，選擇電子郵件。針對 Endpoint (端點)，輸入新收件人的電子郵件地址。選擇建立訂閱。
   - 若要移除電子郵件地址，選擇 Subscription ID (訂閱 ID)。選擇 Other subscription actions (其他訂閱動作)、Delete subscriptions (刪除訂閱)。

4. 選擇 Publish to topic (發佈至主題)。

刪除警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇警示。
3. 選取警示名稱左側的核取方塊，然後選擇 Actions (動作)、Delete (刪除)。
4. 選擇刪除。

隱藏 Auto Scaling 鬧鐘

當您在中檢視警示時 AWS Management Console，可以隱藏與 Amazon EC2 自動擴展和 Application Auto Scaling 自動擴展相關的警示。這項功能只在 AWS Management Console中可供使用。

暫時隱藏 Auto Scaling 警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中選擇 Alarms (警示)、All alarms (所有警示)，然後選取 Hide all AutoScaling alarms (隱藏所有 AutoScaling 警示)。

警示使用案例和範例

以下各節提供常見使用案例的警示範例和教學課程。

建立帳單警示以監控您的估計 AWS 費用

您可以通過使用 Amazon 監控您的估計 AWS 費用 CloudWatch。當您啟用 AWS 帳戶的預估費用監視時，系統會計算估計費用，並以指標資料的 CloudWatch 形式每天多次傳送至。

帳單指標資料存放在美國東部 (維吉尼亞北部) 區域，並且會呈現全球費用。此資料包括您使用的每項服務 AWS 的估計費用，以及估計的總 AWS 費用。

當您的帳戶帳單超過指定閾值時，便會觸發警示。其只會在目前帳單超過閾值時觸發。它不會使用根據您目前一個月用量所進行的預測。

若您建立帳單警示時，費用已經超過閾值，則警示會立即移至 ALARM 狀態。

> **Note**
>
> 如需分析已收取 CloudWatch 費用的相關資訊，請參閱CloudWatch 帳單和成本。

任務

- 啟用帳單提醒
- 建立帳單警示
- 刪除帳單警示

啟用帳單提醒

您必須先啟用帳單警示，才能建立預估費用警示，以便您可以使用帳單指標資料監控預估 AWS 費用並建立警示。在啟用帳單提醒之後，將無法停用資料收集，但您可以刪除已建立的任何帳單警示。

在您第一次啟用帳單提醒之後，大約需要 15 分鐘才能查看帳單資料和設定帳單警示。

要求

- 您必須使用帳戶根使用者憑證登入，或以已獲授予檢視帳單資訊許可的 IAM 使用者身分登入。
- 針對合併帳單帳戶，以付款帳戶身分登入，即可看到各個連結帳戶的帳單資料。除了整合帳戶，您可以檢視各個連結帳戶的總計預估費用及各項服務預估費用。
- 在合併帳單帳戶中，只有在付款人帳戶啟用 Receive Billing Alers (接收帳單提醒) 喜好設定時，才會擷取成員連結的帳戶指標。若您變更管理/付款人帳戶的帳戶，則必須在新的管理/付款人帳戶中啟用帳單提醒。
- 該帳戶不得是 Amazon 合作夥伴網路 (APN) 的一部分，因為 APN 帳戶的計費指標並未發佈到 CloudWatch 。如需詳細資訊，請參閱 AWS 合作夥伴網路。

啟用監控預估費用

1. 開啟主 AWS Billing 控台，網址為 https://console.aws.amazon.com/billing/。
2. 在導覽窗格中，選擇 Billing preferences (帳單偏好設定)。
3. 依警示偏好設定選擇編輯。
4. 選擇 [接收 CloudWatch 帳單通知]。
5. 選擇 Save preferences (儲存喜好設定)。

建立帳單警示

> **Important**
>
> 建立帳單警示前，您必須將區域設定為美國東部 (維吉尼亞北部)。帳單指標資料會存放在此區域，並且會呈現全球費用。此外，您還須為帳戶或在管理/付款人帳戶中啟用帳單提醒 (若您使用的是合併帳單)。如需詳細資訊，請參閱 啟用帳單提醒。

在此程序中，您會建立警示，當您的預估費用 AWS 超過定義的臨界值時傳送通知。

使用 CloudWatch 主控台建立帳單警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Alarms (警示)，然後選擇 All alarms (所有警示)。
3. 選擇 Create alarm (建立警示)。
4. 選擇 Select metric (選取指標)。在 Browse (瀏覽) 中，選擇 Billing (計費)，然後選擇 Total Estimated Charge (預估費用總金額)。

   > **Note**
   >
   > 如果您未看到帳單/預估費用總金額指標，請啟用帳單警示，並將區域變更為美國東部 (維吉尼亞北部)。如需詳細資訊，請參閱 啟用帳單提醒。

5. 選取EstimatedCharges測量結果的方塊，然後選擇 「選取測量結果」。
6. 對於 Statistic (統計數字)，選擇 Maximum (最大值)。
7. 在 Period (時段) 中，選擇 6 hours (6 小時)。
8. 對於閾值類型，選擇靜態。
9. 對於無論何時如 EstimatedCharges 此。 ，選擇 [更大]。
10. 針對於...，定義您要讓警示觸發的值。例如，**200** 美元。

    EstimatedCharges指標值僅以美元 (USD) 為單位，貨幣轉換由 Amazon 服務有限責任公司提供。如需詳細資訊，請參閱什麼是 AWS Billing？ 。

> **Note**
>
> 定義閾值後，預覽圖表會顯示當月的預估費用。

11. 選擇其他組態並執行下列操作：
    - 在 Datapoints to alarm (要警示的資料點) 中，指定 1 out of 1 (1 傳出 1)。
    - 在 Missing data treatment (遺失資料處理) 中，選擇 Treat missing data as missing (將遺失資料視為遺失)。
12. 選擇下一步。
13. 在通知下，確定已選取警示中。接著，指定當警示處於 ALARM 狀態時要通知的 Amazon SNS 主題。Amazon SNS 主題可以包含您的電子郵件地址，以便您在帳單金額超過自己指定的閾值時收到電子郵件。

    您可以選取現有的 Amazon SNS 主題、建立新的 Amazon SNS 主題，或使用主題 ARN 來通知其他帳戶。若希望警報針對相同警示狀態或不同警示狀態傳送多個通知，請選擇 Add notification (新增通知)。
14. 選擇下一步。
15. 在 Name and description (名稱和描述) 下，輸入警示的名稱。此名稱只能包含 UTF-8 字元，不能包含 ASCII 控制字元。
    - (選用) 輸入警示的描述。說明可以包括降價格式，僅顯示在 CloudWatch 控制台的警報詳細信息選項卡中。Markdown 對於將連結新增至執行手冊或其他內部資源很實用。
16. 選擇下一步。
17. 在 Preview and create (預覽並建立) 下，請檢查組態是否正確無誤，然後選擇 Create alarm (建立警示)。

刪除帳單警示

當您不再需要帳單警示時，可以將它刪除。

刪除帳單警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 如有必要，請將 Region (區域) 變更為美國東部 (維吉尼亞北部)。帳單指標資料存放於此區域，而且會反映全球費用。

3. 在導覽窗格中，選擇 Alarms (警示)、All alarms (所有警示)。
4. 選取警示旁的核取方塊，然後選擇 Actions (動作)、Delete (刪除)。
5. 出現確認提示時，選擇 Yes, Delete (是，刪除)。

建立 CPU 使用量警示

您可以建立 CloudWatch 警示，在警示狀態從變更OK為時，使用 Amazon SNS 傳送通知ALARM。

警示會在 EC2 執行個體的平均 CPU 用量超過連續指定期間的指定閾值時變更為 ALARM 狀態。

使用設定 CPU 使用率警示 AWS Management Console

使用下列步驟 AWS Management Console 來建立 CPU 使用率警示。

根據 CPU 使用率建立警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Alarms (警示)、All alarms (所有警示)。
3. 選擇 Create alarm (建立警示)。
4. 選擇 Select metric (選取指標)。
5. 在 All metrics (所有指標) 標籤中，選擇 EC2 metrics (EC2 指標)。
6. 選擇指標類別 (例如，Per-Instance Metrics (每個執行個體指標))。
7. 在 InstanceId「測量結果名稱」資料欄中找到您要列示之執行處理的資料列，並在「測量結果名稱」資料欄中找到「CPU 選取此資料列旁的核取方塊，然後選擇 Select metric (選取指標)。
8. 在 Specify metric and conditions (指定指標和條件) 下，針對 Statistic (統計資訊)，選擇 Average (平均)，選擇其中一個預先定義的百分位數，或指定自訂的百分位數 (例如 **p95.45**)。
9. 選擇期間 (例如，**5 minutes**)。
10. 在 Conditions (條件) 下，指定以下內容：

    a. 對於閾值類型，選擇靜態。

    b. 針對 Whenever CPUUtilization is (每當 CPUUtilization 為)，指定 Greater (大於)。在 than... (比...) 下，指定當 CPU 使用率超過此百分比時，觸發警示進入 ARARM 狀態的閾值。例如：70。

    c. 選擇 Additional configuration (其他組態)。針對 Datapoints to alarm (要警示的資料點)，請指定 (資料點) 必須處於 ALARM 狀態多少評估期間，才會觸發警示。如果此處的兩個值相符，您便可以建立警示，在許多連續期間違規時移至 ALARM 狀態。

若要建立 N 個中有 M 個警示，請針對第一個值，指定低於您為第二個值所指定值的值。如需詳細資訊，請參閱 評估警示。

d. 針對 Missing data treatment (遺失資料處理)，選擇警示在遺失某些資料點時的行為。如需詳細資訊，請參閱 設定 CloudWatch 警示如何處理遺失的資料。

e. 若警示使用百分位數作為監控統計資料，則會出現一個 Percentiles with low samples (低樣本的百分位數) 方塊。請使用它來選擇是要評估還是忽略具有低抽樣率的案例。若您選擇 ignore (maintain alarm state) (忽略 (維持警示狀態))，則會在抽樣大小過低時一律維持目前的警示狀態。如需詳細資訊，請參閱 以百分比為基礎的 CloudWatch 警報和低資料樣本。

11. 選擇下一步。
12. 在 Notification (通知) 下，選擇 In alarm (在警示中)，然後選取當警示處於 ALARM 狀態時要通知的 SNS 主題。

    若要讓警示針對相同的警示狀態或不同警示狀態傳送多個通知，請選擇 Add notification (新增通知)。

    若要讓警示不傳送通知，請選擇 Remove (移動)。
13. 完成時，請選擇下一步。
14. 輸入警示的名稱與說明。然後選擇下一步。

    此名稱只能包含 UTF-8 字元，不能包含 ASCII 控制字元。說明可以包括降價格式，僅顯示在 CloudWatch 控制台的警報詳細信息選項卡中。Markdown 對於將連結新增至執行手冊或其他內部資源很實用。
15. 在 Preview and create (預覽及建立) 下，請確認資訊和條件都是您希望的內容，然後選擇 Create alarm (建立警示)。

使用設定 CPU 使用率警示 AWS CLI

使用下列步驟 AWS CLI 來建立 CPU 使用率警示。

根據 CPU 使用率建立警示

1. 設定 SNS 主題。如需詳細資訊，請參閱 設定 Amazon SNS 通知。
2. 使用put-metric-alarm指令建立警示，如下所示。

```
aws cloudwatch put-metric-alarm --alarm-name cpu-mon --alarm-description "Alarm
 when CPU exceeds 70%" --metric-name CPUUtilization --namespace AWS/EC2 --statistic
```

```
 Average --period 300 --threshold 70 --comparison-operator GreaterThanThreshold --
dimensions  Name=InstanceId,Value=i-12345678 --evaluation-periods 2 --alarm-actions
 arn:aws:sns:us-east-1:111122223333:my-topic --unit Percent
```

3. 透過使用set-alarm-state指令強制變更警示狀態來測試警報。

   a. 將警示的狀態從 INSUFFICIENT_DATA 變更為 OK。

   ```
   aws cloudwatch set-alarm-state --alarm-name cpu-mon --state-reason
    "initializing" --state-value OK
   ```

   b. 將警示的狀態從 OK 變更為 ALARM。

   ```
   aws cloudwatch set-alarm-state --alarm-name cpu-mon --state-reason
    "initializing" --state-value ALARM
   ```

   c. 確認您已收到一封與警示相關的通知。

建立會傳送電子郵件的負載平衡器延遲警示

您可以設定一個 Amazon SNS 通知和設定警示來監控 Classic Load Balancer 中超過 100 毫秒的延遲。

使用設定延遲警示 AWS Management Console

使用下列步驟 AWS Management Console 來建立負載平衡器延遲警示。

若要建立負載平衡器延遲警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Alarms (警示)、All alarms (所有警示)。
3. 選擇 Create alarm (建立警示)。
4. 在「依類別的CloudWatch 量度」下，選擇「ELB 測量結果」類別。
5. 選取 Classic Load Balancer 的列以及 Latency (延遲) 指標。
6. 針對統計資訊，選擇 Average (平均)，選擇其中一個預先定義百分位數，或指定自訂的百分位數 (例如 **p95.45**)。
7. 針對期間，請選擇 1 Minute (1 分鐘)。
8. 選擇下一步。

9. 在 Alarm Threshold (警示閾值) 下，請輸入警示的唯一名稱 (例如，**myHighCpuAlarm**) 和警示的說明 (例如，**Alarm when Latency exceeds 100s**)。警示名稱只能包含 UTF-8 字元，不能包含 ASCII 控制字元。

   此名稱只能包含 UTF-8 字元，不能包含 ASCII 控制字元。說明可以包括降價格式，僅顯示在 CloudWatch 控制台的警報詳細信息選項卡中。Markdown 對於將連結新增至執行手冊或其他內部資源很實用。

10. 在 Whenever (無論何時) 下，針對 is (是)，選擇 **>**，然後輸入 **0.1**。針對 for (期間)，輸入 **3**。

11. 在 Additional settings (其他設定) 下，針對 Treat missing data as (將遺失資料視為)，請選擇 ignore (maintain alarm state) (忽略 (維持警示狀態))，讓遺失資料點不會觸發警示狀態變更。

    針對 Percentiles with low samples (低樣本的百分位數)，選擇 ignore (maintain the alarm state) (忽略 (維持警示狀態))，讓警示僅評估具足夠資料樣本數的狀況。

12. 在動作下的 每當此警示，選擇狀態為警示。在 Send notification to (傳送通知至) 中，選擇現有 SNS 主題或建立新的主題。

    若要建立 SNS 主題，請選擇 New list (新增清單)。針對 Send notification to (傳送通知至)，輸入 SNS 主題的名稱 (例如，**myHighCpuAlarm**)，並針對 Email list (電子郵件清單)，輸入以逗號分隔的電子郵件地址清單，當警示變更為 ALARM 狀態時，這些電子郵件地址將會收到通知。每個電子郵件地址都會收到主題訂閱確認電子郵件。您必須確認訂閱，才會傳送通知。

13. 選擇建立警示 。

使用設定延遲警示 AWS CLI

使用下列步驟 AWS CLI 來建立負載平衡器延遲警示。

若要建立負載平衡器延遲警示

1. 設定 SNS 主題。如需詳細資訊，請參閱 設定 Amazon SNS 通知。
2. 使用put-metric-alarm指令建立警示，如下所示：

```
aws cloudwatch put-metric-alarm --alarm-name lb-mon --alarm-description "Alarm
 when Latency exceeds 100s" --metric-name Latency --namespace AWS/ELB --statistic
 Average --period 60 --threshold 100 --comparison-operator GreaterThanThreshold --
dimensions Name=LoadBalancerName,Value=my-server --evaluation-periods 3 --alarm-
actions arn:aws:sns:us-east-1:111122223333:my-topic --unit Seconds
```

3. 透過使用set-alarm-state指令強制變更警示狀態來測試警報。

a. 將警示的狀態從 INSUFFICIENT_DATA 變更為 OK。

```
aws cloudwatch set-alarm-state --alarm-name lb-mon --state-reason
 "initializing" --state-value OK
```

b. 將警示的狀態從 OK 變更為 ALARM。

```
aws cloudwatch set-alarm-state --alarm-name lb-mon --state-reason
 "initializing" --state-value ALARM
```

c. 確認您已收到一封與警示相關的電子郵件通知。

建立會傳送電子郵件的儲存體輸送量警示

您可以設定 SNS 通知和設定當 Amazon EBS 超過 100 MB 輸送量時傳送電子郵件的警示。

使用 AWS Management Console設定儲存體輸送量警示

使用這些步驟 AWS Management Console 來根據 Amazon EBS 輸送量建立警示。

若要建立儲存輸送量警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Alarms (警示)、All alarms (所有警示)。
3. 選擇 Create alarm (建立警示)。
4. 在 EBS Metrics (EBS 指標) 下，選擇指標類別。
5. 選取含有體積和量VolumeWriteBytes度的資料列。
6. 在統計資料中選擇 Average (平均)。對於期間，選擇 5 Minutes (5 分鐘)。選擇下一步。
7. 在 Alarm Threshold (警示閾值) 下，請輸入警示的唯一名稱 (例如，**myHighWriteAlarm**) 和警示的說明 (例如，**VolumeWriteBytes exceeds 100,000 KiB/s**)。此名稱只能包含 UTF-8 字元，不能包含 ASCII 控制字元。說明可以包括降價格式，僅顯示在 CloudWatch 控制台的警報詳細信息選項卡中。Markdown 對於將連結新增至執行手冊或其他內部資源很實用。
8. 在 Whenever (無論何時) 下，針對 is (是)，選擇 **>**，然後輸入 **100000**。針對 for (期間)，輸入 **15** 個連續期間。

   以圖像方式呈現的閥值會顯示在 Alarm Preview (警示預覽) 下方。

9. 在 Additional settings (其他設定) 下，針對 Treat missing data as (將遺失資料視為)，請選擇 ignore (maintain alarm state) (忽略 (維持警示狀態))，讓遺失資料點不會觸發警示狀態變更。
10. 在動作下的 每當此警示，選擇狀態為警示。在 Send notification to (傳送通知至) 中選擇現有 SNS 主題或建立主題。

    若要建立 SNS 主題，請選擇 New list (新增清單)。針對 Send notification to (傳送通知至)，輸入 SNS 主題的名稱 (例如，**myHighCpuAlarm**)，並針對 Email list (電子郵件清單)，輸入以逗號分隔的電子郵件地址清單，當警示變更為 ALARM 狀態時，這些電子郵件地址將會收到通知。每個電子郵件地址都會收到主題訂閱確認電子郵件。您必須確認訂閱，通知才會傳送到您的電子郵件地址。
11. 選擇建立警示 。

使用設定儲存區輸送量警示 AWS CLI

使用這些步驟 AWS CLI 來根據 Amazon EBS 輸送量建立警示。

若要建立儲存輸送量警示

1. 建立 SNS 主題。如需詳細資訊，請參閱 設定 Amazon SNS 通知。
2. 建立警示。

```
aws cloudwatch put-metric-alarm --alarm-name ebs-mon --alarm-description "Alarm
 when EBS volume exceeds 100MB throughput" --metric-name VolumeReadBytes --
namespace AWS/EBS --statistic Average --period 300 --threshold 100000000 --
comparison-operator GreaterThanThreshold --dimensions Name=VolumeId,Value=my-
volume-id --evaluation-periods 3 --alarm-actions arn:aws:sns:us-
east-1:111122223333:my-alarm-topic --insufficient-data-actions arn:aws:sns:us-
east-1:111122223333:my-insufficient-data-topic
```

3. 透過使用set-alarm-state指令強制變更警示狀態來測試警報。

   a. 將警示的狀態從 INSUFFICIENT_DATA 變更為 OK。

   ```
   aws cloudwatch set-alarm-state --alarm-name ebs-mon --state-reason
    "initializing" --state-value OK
   ```

   b. 將警示的狀態從 OK 變更為 ALARM。

   ```
   aws cloudwatch set-alarm-state --alarm-name ebs-mon --state-reason
    "initializing" --state-value ALARM
   ```

c. 將警示的狀態從 ALARM 變更為 INSUFFICIENT_DATA。

```
aws cloudwatch set-alarm-state --alarm-name ebs-mon --state-reason
 "initializing" --state-value INSUFFICIENT_DATA
```

d. 確認您已收到一封與警示相關的電子郵件通知。

從資料庫建立 Performance Insights 計 AWS 數器指標的警示

CloudWatch 包含 DB_PERF_INSISIG ETS 指標數學函數，您可以使用此函數將 Performance Insights 計數器指標 CloudWatch 從 Amazon Relational Database Service 服務和 Amazon DocumentDB (與 MongoDB 相容性) 引入。DB_PERF_INSIGHTS 也引進了次分鐘間隔的 DBLoad 指標。您可以在這些指標上設定 CloudWatch 警示。

如需有關 Amazon RDS Performance Insights 的更多資訊，請參閱在 Amazon RDS 上使用 Performance Insights 監控資料庫負載。

如需有關 Amazon DocumentDB Performance Insights 的更多資訊，請參閱使用 Performance Insights 監控。

根據 DB_PERF_INSIGHTS 函數的警示不支援異常偵測。

> **Note**
>
> 由 DB_PERF_INSIGHTS 擷取的次分鐘精細度高解析度指標，僅適用於 DBLoad 指標，或者如果您已以較高解析度啟用「增強型監控」，則適用於作業系統指標。如需 Amazon RDS 增強型監控的更多資訊，請參閱使用增強型監控來監控 OS 指標。
> 您可以使用 DB_PERF_INSITE XT 函數建立高解析度警示。高解析度警報的最大估算範圍為三小時。您可以使用 CloudWatch 主控台繪製任何時間範圍內使用 DB_PERF_INSIGET ICS 函式擷取的度量圖形。

若要建立根據 Performance Insights 指標的警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Alarms (警示)，然後選擇 All alarms (所有警示)。
3. 選擇 Create alarm (建立警示)。
4. 選擇 Select Metric (選取指標)。

5. 選擇新增數學下拉式清單，然後從清單中選取資料庫效能指標、DB_PERF_INSIGHTS。

   在您選擇 DB_PERF_INSIGHTS 後，畫面上會顯示一個數學表達式方塊，您可以在其中套用或編輯數學表達式。

6. 在數學表達式方塊中，輸入您的 DB_PERF_INSIGHTS 數學表達式，然後選擇套用。

   例如：**DB_PERF_INSIGHTS('RDS', 'db-ABCDEFGHIJKLMNOPQRSTUVWXY1', 'os.cpuUtilization.user.avg')**

   > **Important**
   >
   > 當您使用 DB_PERF_INSIGHTS數學運算式時，必須指定資料庫的唯一資料庫資源 ID。這與資料庫識別碼不同。若要在 Amazon RDS 主控台中查找資料庫資源 ID，請選擇資料庫執行個體來查看其詳細資訊。然後選擇 Configuration (組態) 標籤。資源 ID 顯示在組態區段中。

   如需有關 DB_PERF_INSIGHTS 函數和其他指標數學可用函數的資訊，請參閱 指標數學語法和函數。

7. 選擇選取指標。

   Specify metric and conditions (指定指標與條件) 頁面隨即出現，顯示您已選取數學表達式的圖形和其他資訊。

8. 針對 Whenever ***expression*** is (表達式為...時)，指定表達式是否必須大於、小於或等於閾值。在 than... (於...) 下，指定閾值。

9. 選擇 Additional configuration (其他組態)。針對 Datapoints to alarm (要警示的資料點)，請指定 (資料點) 必須處於 ALARM 狀態多少評估期間，才會觸發警示。如果此處的兩個值相符，您便可以建立警示，在許多連續期間違規時移至 ALARM 狀態。

   若要建立 N 個中有 M 個警示，請針對第一個值，指定低於您為第二個值所指定值的值。如需詳細資訊，請參閱 評估警示。

10. 針對 Missing data treatment (遺失資料處理)，選擇警示在遺失某些資料點時的行為。如需詳細資訊，請參閱 設定 CloudWatch 警示如何處理遺失的資料。

11. 選擇下一步。

12. 在 Notification (通知) 下，選取 SNS 主題來在警示處於 ALARM 狀態、OK 狀態或 INSUFFICIENT_DATA 狀態時進行通知。

若要讓警示針對相同的警示狀態或不同警示狀態傳送多個通知，請選擇 Add notification (新增通知)。

若要讓警示不傳送通知，請選擇 Remove (移動)。

13. 若要讓警示執行 Auto Scaling、 EC2、Lambda 或 Systems Manager 動作，請選擇適當的按鈕，然後選擇警示狀態及要執行的動作。如果選擇 Lambda 函數作為警示動作，則可以指定函數名稱或 ARN，並且可以選擇性地選擇函數的特定版本。

    警示只能在進入 ALARM 狀態時執行 Systems Manager 動作。如需有關 Systems Manager 動作的詳細資訊，請參閱設定 CloudWatch 為 OpsItems 從警示建立與事件建立。

    > **Note**
    >
    > 若要建立執行 SSM Incident Manager 動作的警示，您必須具備特定許可。如需詳細資訊，請參閱AWS 系統管理員事件管理員的身分識別原則範例。

14. 完成時，請選擇下一步。
15. 輸入警示的名稱與說明。然後選擇下一步。

    此名稱只能包含 UTF-8 字元，不能包含 ASCII 控制字元。說明可以包括降價格式，僅顯示在 CloudWatch 控制台的警報詳細信息選項卡中。Markdown 對於將連結新增至執行手冊或其他內部資源很實用。

16. 在 Preview and create (預覽及建立) 下，請確認資訊和條件都是您希望的內容，然後選擇 Create alarm (建立警示)。

建立警示以停止、終止、重新啟動或復原 EC2 執行個體

使用 Amazon CloudWatch 警示動作，您可以建立自動停止、終止、重新開機或復原 EC2 執行個體的警示。當執行個體不再需要執行，您可以使用停止或終止動作以協助您節省成本。如果發生系統受損，您可以使用重新啟動和復原動作，自動重新啟動這些執行個體或將它們復原到新的硬體。

在許多情況下，您可能想要自動停止或終止您的執行個體。例如，您可能有專門批次處理薪資作業或科學運算任務的執行個體，它們在執行一段時間後完成工作。不要讓這些執行個體閒置 (及累積費用)，您可以停止或終止它們，以協助您節省成本。使用停止和終止警示動作的主要差別在於，如果您後來需要再次執行已停止的執行個體，則可以輕鬆地重新啟動它。您也可以保留相同的執行個體 ID 和根磁碟區。不過，您不能重新啟動已終止的執行個體。相反地，您必須啟動新的執行個體。

除了包含 "InstanceId=" 維度的任何自訂指標之外，您還可以將停止、終止或重新啟動動作新增到 Amazon EC2 每個執行個體指標上設定的任何警示，包括 Amazon 提供的基本和詳細監控指標 CloudWatch (在 AWS/EC2 命名空間中)，只要該 InstanceId 值參考有效的執行中 Amazon EC2 執行個體即可。您還可將復原動作新增至設定於任何 Amazon EC2 每執行個體指標上的警示，`StatusCheckFailed_Instance` 除外。

若要設定可以重新啟動、停止或終止執行個體的 CloudWatch 警示動作，您必須使用服務連結的 IAM 角色AWSServiceRoleForCloudWatchEvents。 AWSServiceRoleForCloudWatchEvents IAM 角色可 AWS 讓您代表執行警示動作。

若要為 CloudWatch 事件建立服務連結角色，請使用下列命令：

```
aws iam create-service-linked-role --aws-service-name events.amazonaws.com
```

主控台支援

您可以使用 CloudWatch 主控台或 Amazon EC2 主控台建立警示。本文件中的程序使用主 CloudWatch 控台。如需使用 Amazon EC2 主控台的程序，請參閱《Amazon EC2 Linux 執行個體使用者指南中的建立警示，來停止、終止、重新啟動復原或恢復執行個體。

許可

如果您使用 AWS Identity and Access Management (IAM) 帳戶建立或修改執行 EC2 動作或 Systems Manager OpsItem 動作的警示，您必須擁有該`iam:CreateServiceLinkedRole`權限。

目錄

- 向 Amazon CloudWatch 警報添加停止操作
- 向 Amazon CloudWatch 警報添加終止動作
- 將重新開機動作新增至 Amazon CloudWatch 警示
- 將恢復動作添加到 Amazon CloudWatch 警報
- 檢視已觸發警示和動作的歷史記錄

向 Amazon CloudWatch 警報添加停止操作

您可以建立警示，在符合特定閾值時停止 Amazon EC2 執行個體。例如，您可以執行開發或測試執行個體，並偶爾忘記關閉它們。您可以建立警示，在平均 CPU 使用率百分比已低於 10% 達 24 小時時觸發，以通知其為閒置且不再使用。您可以調整閾值、持續時間和期間以符合您的需求，而且您可以新增 SNS 通知，當觸發警示時，您將會收到電子郵件。

使用 Amazon Elastic Block Store 磁碟區作為根裝置的 Amazon EC2 執行個體可以停止或終止，而使用執行個體存放區作為根裝置的執行個體只能終止。

使用 Amazon CloudWatch 主控台建立警示以停止閒置執行個體

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Alarms (警示)、All alarms (所有警示)。
3. 選擇 Create alarm (建立警示)。
4. 選擇 Select Metric (選取指標)。
5. 在 AWS 命名空間中，選擇 EC2。
6. 請執行下列操作：
   a. 選擇 Per-Instance Metrics (每個執行個體指標)。
   b. 勾選正確執行個體的資料列中的核取方塊以及 CPUUtilization 指標。
   c. 選擇 Graphed metrics (圖表化指標) 標籤。
   d. 在統計資料中選擇 Average (平均)。
   e. 選擇期間 (例如，**1 Hour**)。
   f. 選擇選取指標。
7. 在 Define Alarm (定義警示) 步驟，執行下列動作：
   a. 在 Conditions (條件) 下選擇 Static (靜態)。
   b. 在 Whenever CPUUtilization is (每當 CPU 使用率為) 下方，選擇 Lower (較低)。
   c. 針對 than (比較)，輸入 **10**。
   d. 選擇下一步。
   e. 在 Notification (通知) 之下的 Send notification to (傳送通知至) 中，選擇現有的 SNS 主題或建立新的主題。

      若要建立 SNS 主題，請選擇 New list (新增清單)。在 Send notification to (傳送通知至) 中，輸入 SNS 主題的名稱 (例如，Stop_EC2_Instance)。在 Email list (電子郵件清單) 中輸入以逗號分隔的電子郵件地址清單，當警示變更為 ALARM 狀態時，這些電子郵件地址將會收到通知。每個電子郵件地址都會收到主題訂閱確認電子郵件。您必須確認訂閱，通知才會傳送到您的電子郵件地址。
   f. 選擇 Add EC2 Action (新增 EC2 動作)。
   g. 針對 Alarm state trigger (警示狀態觸發)，選擇 In Alarm (警示中)。針對 Take the action (採取動作)，選擇 Stop this instance (停止此執行個體)。

h. 選擇下一步。

i. 輸入警示的名稱與說明。名稱只能包含 ASCII 字元。然後選擇下一步。

j. 在 Preview and create (預覽及建立) 下，請確認資訊和條件都是您希望的內容，然後選擇 Create alarm (建立警示)。

向 Amazon CloudWatch 警報添加終止動作

您可以建立警示，在符合特定閾值時自動終止 EC2 執行個體 (前提是執行個體未啟用終止保護)。例如，您可能想要在執行個體完成作業時予以終止，而且不再需要該執行個體。如果您之後還要使用該執行個體，您應該停止而非終止執行個體。如需有關啟用和停用執行個體終止保護的詳細資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的啟用執行個體終止保護。

若要使用 Amazon CloudWatch 主控台建立警示以終止閒置執行個體

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Alarms (警示)、Create Alarm (建立警示)。
3. 在 Select Metric (選取指標) 步驟，執行下列動作：

   a. 在 EC2 Metrics (EC2 指標) 之下選擇 Per-Instance Metrics (每個執行個體指標)。

   b. 選取該執行個體的資料列以及 CPUUtilization 指標。

   c. 在統計資料中選擇 Average (平均)。

   d. 選擇期間 (例如，**1 Hour**)。

   e. 選擇下一步。
4. 在 Define Alarm (定義警示) 步驟，執行下列動作：

   a. 在 Alarm Threshold (警示閾值) 之下輸入唯一的名稱 (例如，終止 EC2 執行個體)，以及警示的描述 (例如，當 CPU 閒置太久時終止 EC2 執行個體)。警示名稱只能包含 ASCII 字元。

   b. 在 Whenever (無論何時) 之下的 is (是) 中，選擇 **<** 並輸入 **10**。在 for (持續) 中，輸入 **24** 個連續期間。

      以圖像方式呈現的閥值會顯示在 Alarm Preview (警示預覽) 下方。

   c. 在 Notification (通知) 之下的 Send notification to (傳送通知至) 中，選擇現有的 SNS 主題或建立新的主題。

      若要建立 SNS 主題，請選擇 New list (新增清單)。在 Send notification to (傳送通知至) 中，輸入 SNS 主題的名稱 (例如，Terminate_EC2_Instance)。在 Email list (電子郵件清單) 中輸

入以逗號分隔的電子郵件地址清單，當警示變更為 ALARM 狀態時，這些電子郵件地址將會收到通知。每個電子郵件地址都會收到主題訂閱確認電子郵件。您必須確認訂閱，通知才會傳送到您的電子郵件地址。

d. 選擇 EC2 Action (EC2 動作)。

e. 在 Whenever this alarm (每當此警示) 中選擇 State is ALARM (狀態為警示)。在 Take this action (採取此動作) 中選擇 Terminate this instance (終止此執行個體)。

f. 選擇建立警示 。

將重新開機動作新增至 Amazon CloudWatch 警示

您可以建立 Amazon CloudWatch 警示來監控 Amazon EC2 執行個體並自動重新啟動執行個體。重新啟動警示動作建議用於執行個體運作狀態檢查失敗 (相對的，復原警示動作則適用於系統運作狀態檢查失敗)。重新啟動執行個體等同於重新啟動作業系統。在大多數情況下，將執行個體重新開機只需要幾分鐘的時間。當您重新啟動執行個體時，它會維持在相同的實體主機上，所以您的執行個體會保有其公有 DNS 名稱、私有 IP 地址和執行個體存放區磁碟區上的任何資料。

不同於停用和重新啟動您的執行個體，重新啟動執行個體不會啟動新執行個體計費小時。如需重新啟動執行個體的詳細資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的啟動執行個體。

> **Important**
>
> 為了避免重新啟動和復原動作之間的競爭情況，請避免為重新啟動警示和復原警示設定相同的評估期間。我們建議您將重新開機警示設定為三個各一分鐘的評估期間。

若要使用 Amazon CloudWatch 主控台建立警示以重新啟動執行個體

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Alarms (警示)、Create Alarm (建立警示)。
3. 在 Select Metric (選取指標) 步驟，執行下列動作：

   a. 在 EC2 Metrics (EC2 指標) 之下選擇 Per-Instance Metrics (每個執行個體指標)。

   b. 選取含有執行處理和 StatusCheckFailed_Instance 測量結果的資料列。

   c. 在統計資料中選擇 Minimum (最小值)。

   d. 選擇期間 (例如，**1 Minute**)。

   e. 選擇下一步。

4. 在 Define Alarm (定義警示) 步驟，執行下列動作：

   a. 在 Alarm Threshold (警示閾值) 之下輸入唯一的名稱 (例如，重新啟動 EC2 執行個體)，以及警示的描述 (例如，當運作狀態檢查失敗時重新啟動 EC2 執行個體)。警示名稱只能包含 ASCII 字元。

   b. 在 Whenever (無論何時) 之下的 is (是) 中，選擇 **>** 並輸入 **0**。在 for (持續) 中，輸入 **3** 個連續期間。

      以圖像方式呈現的閥值會顯示在 Alarm Preview (警示預覽) 下方。

   c. 在 Notification (通知) 之下的 Send notification to (傳送通知至) 中，選擇現有的 SNS 主題或建立新的主題。

      若要建立 SNS 主題，請選擇 New list (新增清單)。在 Send notification to (傳送通知至) 中，輸入 SNS 主題的名稱 (例如，Reboot_EC2_Instance)。在 Email list (電子郵件清單) 中輸入以逗號分隔的電子郵件地址清單，當警示變更為 ALARM 狀態時，這些電子郵件地址將會收到通知。每個電子郵件地址都會收到主題訂閱確認電子郵件。您必須確認訂閱，通知才會傳送到您的電子郵件地址。

   d. 選擇 EC2 Action (EC2 動作)。

   e. 在 Whenever this alarm (每當此警示) 中選擇 State is ALARM (狀態為警示)。在 Take this action (採取此動作) 中選擇 Reboot this instance (重新啟動此執行個體)。

   f. 選擇建立警示 。

將恢復動作添加到 Amazon CloudWatch 警報

您可以建立 Amazon CloudWatch 警示來監控 Amazon EC2 執行個體，並在執行個體因基礎硬體故障或需要 AWS 參與修復的問題而受損時，自動復原執行個體。已終止的執行個體無法復原。復原後的執行個體與原始執行個體相同，包括執行個體 ID、私有 IP 地址、彈性 IP 地址及所有執行個體中繼資料。

當 StatusCheckFailed_System 警示觸發且復原動作啟動時，您將收到 Amazon SNS 主題通知，這是您在建立警示時選擇並與復原動作關聯的通知。在執行個體復原期間，執行個體會在重新啟動期間遷移，記憶體內的任何資料都將遺失。當程序完成時，會將資訊發佈到您為此警示設定的 SNS 主題。訂閱此 SNS 主題的所有使用者都將會收到電子郵件通知，其中包含復原嘗試的狀態和進一步的說明。您將會發現執行個體在已復原的執行個體上重新啟動。

復原動作只能用於 StatusCheckFailed_System，而非 StatusCheckFailed_Instance。

導致系統狀態檢查失敗的問題範例包括：

- 網路連線中斷
- 系統電力中斷
- 實體主機的軟體問題
- 實體主機上會影響網路連線的硬體問題

僅某些執行個體類型上支援復原動作。如需有關受支援的執行個體類型和其他需求的詳細資訊，請參閱復原您的執行個體和要求。

> **Important**
>
> 為了避免重新啟動和復原動作之間的競爭情況，請避免為重新啟動警示和復原警示設定相同的評估期間。我們建議您將復原警示設定為兩個各一分鐘的評估期間，將重新啟動警示設定為三個各一分鐘的評估期間。

使用 Amazon CloudWatch 主控台建立警示以復原執行個體

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Alarms (警示)、Create Alarm (建立警示)。
3. 在 Select Metric (選取指標) 步驟，執行下列動作：
   a. 在 EC2 Metrics (EC2 指標) 之下選擇 Per-Instance Metrics (每個執行個體指標)。
   b. 選取含有執行處理和 StatusCheckFailed_System 測量結果的資料列。
   c. 在統計資料中選擇 Minimum (最小值)。
   d. 選擇期間 (例如，**`1 Minute`**)。

      > **Important**
      >
      > 為了避免重新啟動和復原動作之間的競爭情況，請避免為重新啟動警示和復原警示設定相同的評估期間。我們建議您將復原警示設定為兩個各一分鐘的評估期間。

   e. 選擇下一步。
4. 在 Define Alarm (定義警示) 步驟，執行下列動作：
   a. 在 Alarm Threshold (警示閾值) 之下輸入唯一的名稱 (例如，復原 EC2 執行個體)，以及警示的描述 (例如，當運作狀態檢查失敗時復原 EC2 執行個體)。警示名稱只能包含 ASCII 字元。

b. 在 Whenever (無論何時) 之下的 is (是) 中，選擇 **>** 並輸入 **0**。在 for (持續) 中，輸入 **2** 個連續期間。

c. 在 Notification (通知) 之下的 Send notification to (傳送通知至) 中，選擇現有的 SNS 主題或建立新的主題。

   若要建立 SNS 主題，請選擇 New list (新增清單)。在 Send notification to (傳送通知至) 中，輸入 SNS 主題的名稱 (例如，Recover_EC2_Instance)。在 Email list (電子郵件清單) 中輸入以逗號分隔的電子郵件地址清單，當警示變更為 ALARM 狀態時，這些電子郵件地址將會收到通知。每個電子郵件地址都會收到主題訂閱確認電子郵件。您必須確認訂閱，通知才會傳送到您的電子郵件地址。

d. 選擇 EC2 Action (EC2 動作)。

e. 在 Whenever this alarm (每當此警示) 中選擇 State is ALARM (狀態為警示)。在 Take this action (採取此動作) 中選擇 Recover this instance (復原此執行個體)。

f. 選擇建立警示 。

檢視已觸發警示和動作的歷史記錄

您可以在 Amazon CloudWatch 主控台中檢視警示和動作歷史記錄。Amazon CloudWatch 保留了最近 30 天的警報和行動歷史記錄。

檢視已觸發警示和動作的歷史記錄

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Alarms (警示)，並選取警示。
3. 若要查看隨著時間轉換的最新狀態及指標值，請選擇 Details (詳細資訊)。
4. 若要查看最新的歷史記錄項目，請選擇 History (歷史記錄)。

警報和標記

標籤是索引鍵值配對，可協助您組織和分類資源。您也可以使用這些標籤，僅授予使用者存取或變更具備特定標籤值資源的許可，來設定使用者許可的範圍。有關標記資源的更多一般信息，請參閱標記資 AWS 源

下列清單說明標籤如何與 CloudWatch 警示搭配運作的一些詳細資料。

- 若要能夠設定或更新 CloudWatch 資源的標籤，您必須登入具有cloudwatch:TagResource權限的帳號。例如，若要建立警示並為其設定標籤，除了cloudwatch:TagResource權限之外，您還必須擁有該the cloudwatch:PutMetricAlarm 權限。我們建議您確保組織中將建立或更新 CloudWatch 資源的任何人都有cloudwatch:TagResource權限。
- 標籤可用於基於標籤的授權控制。例如，IAM 使用者或角色許可可可包含條件，以根據其標記限制對特定資源的 CloudWatch 呼叫。但是，請記住以下幾點
  - 名稱開頭為的標籤無法用於aws:以標籤為基礎的授權控制。
  - 複合警報不支援以標籤為基礎的授權控制。

Application Signals

⚠ Application Signals 為預覽版本。如果您對此功能有任何意見，可以透app-signals-feedback過 @amazon .com 與我們聯絡。

使用 CloudWatch 應用程式信號自動檢測您的應用程式，以 AWS 便您監控目前的應用程式健康狀態，並根據業務目標追蹤長期應用程式效能。Application Signals 為您提供應用程式、服務和相依性的統一、以應用程式為中心的檢視，並協助您監控和分類應用程式運作狀態。

- 啟用 Application Signals 可自動收集應用程式的指標和追蹤，並顯示呼叫量、可用性、延遲、故障和錯誤等關鍵指標。快速查看和分類目前的操作運作狀態，以及您的應用程式是否能實現其長期效能目標，無需撰寫自訂程式碼或建立儀表板。
- 使用 Application Signals 建立和監控服務水準目標 (SLO)。輕鬆建立並追蹤與 CloudWatch 指標相關的 SLO 狀態，包括應用程式訊號收集的新標準應用程式指標。在服務清單和拓撲地圖中查看並追蹤應用程式服務的服務水準指標 (SLI) 狀態。建立警示以追蹤您的 SLO，並追蹤 Application Signals 收集的新標準應用程式指標。
- 查看 Application Signals 自動探索的應用程式拓撲地圖，以視覺化方式呈現應用程式、相依性及其連線能力。
- 應用程序信號與 CloudWatch RUM，CloudWatchSynthetics 金絲雀 AWS Service Catalog AppRegistry，並 Amazon EC2 Auto Scaling 顯示您的客戶端頁面，Synthetics 金絲雀和儀表板和地圖中的應用程序名稱。

使用 Application Signals 進行日常應用程式監控

在 CloudWatch 控制台中使用應用程序信號，作為日常應用程序監控的一部分：

1. 如果已為您的服務建立服務水準目標 (SLO)，請從服務水準目標 (SLO) 頁面開始。這可讓您立即檢視最重要的服務和操作的運作狀態。選擇 SLO 的服務或操作名稱，開啟服務詳細資訊頁面，並在疑難排解問題時查看詳細的服務資訊。
2. 開啟服務頁面以查看所有服務的摘要，並快速查看故障率或延遲最高的服務。如果已建立 SLO，請查看「服務」資料表，了解哪些服務具有運作狀態不佳的服務水準指標 (SLI)。如果特定服務的運作狀態不佳，請選取該服務以開啟服務詳細資訊頁面，並查看服務操作、相依性、Synthetics Canary 和用戶端請求。在圖表中選取一個點以查看相關的追蹤，以便可以疑難排解並識別操作問題的根本原因。

3. 如果已部署新服務或相依性已變更，請開啟 Service Map 以檢查您的應用程式拓撲。查看應用程式的地圖，它可顯示用戶端、Synthetics Canary、服務和相依性之間的關係。快速查看 SLI 運作狀態，檢視呼叫量、故障率和延遲等關鍵指標，並深入查看服務詳細資訊頁面中的更多詳細資訊。

使用 Application Signals 會產生費用。如需 CloudWatch 定價的相關資訊，請參閱 Amazon CloudWatch 定價。

> **Note**
>
> 沒有必要使應用程序信號使用 CloudWatch Synthetics， CloudWatch RUM 或 CloudWatch顯然。但是，Synthetics 和 CloudWatch RUM 與應用程序信號一起使用時提供好處，當您一起使用這些功能。

支援的語言和架構

目前，應用程序信號支持 Java 和 Python 應用程序。

Application Signals 在 Amazon EKS、Amazon ECS 和 Amazon EC2 上受到支援和測試。在 Amazon EKS 叢集上，它會自動探索服務和叢集的名稱。在其他架構上，當您為 Application Signals 啟用這些服務時，必須提供服務和環境的名稱。

在 Amazon EC2 上啟用應用程式訊號的說明應適用於任何支援 CloudWatch 代理程式和發行 AWS 版的 OpenTelemetry架構。但是，這些指示尚未在 Amazon ECS 和 Amazon EC2 以外的架構上進行測試。

支援的區域

對於此預覽版本，下列區域支援 Application Signals。

- 美國東部 (維吉尼亞北部)
- 美國東部 (俄亥俄)
- 美國西部 (奧勒岡)
- 亞太區域 (悉尼)
- 亞太區域 (東京)
- 歐洲 (愛爾蘭)

預覽 SDK

SDK 的預覽版本可供下載。

> **Warning**
>
> API 操作和參數可能會在 Application Signals 正式推出之前變更。這些變更可能是重大變更。請勿將 SDK 的預覽版本用於生產目的。

若要安裝預覽 SDK，請先安裝或更新第 2 版的最新 AWS CLI 版本。如需詳細資訊，請參閱安裝或更新最新版本的 AWS CLI。

然後使用下列命令從 Amazon S3 儲存貯體下載 SDK 壓縮檔案，然後解壓縮其內容。每個 SDK 壓縮檔案都包含 SDK 指示和 API 文件。

> **Note**
>
> SDK 以多種編程語言提供，因此您可以將應用程序信號 API 與任何這些編程語言一起使用。不過，只有 Java 和 Python 應用程式才支援自動檢測應用程式以將資料傳送至應用程式訊號。

- Java V2 SDK: `aws s3 cp s3://application-signals-preview-sdk/awsJavaSdkV2.zip ./`
- JavaScript V3 開發套件：`aws s3 cp s3://application-signals-preview-sdk/jsSdkV3.zip ./`
- JavaScript 軟體開發套件：`aws s3 cp s3://application-signals-preview-sdk/jsSdkV2.zip ./`
- Python SDK: `aws s3 cp s3://application-signals-preview-sdk/pythonSdk.zip ./`
- Kotlin SDK: `aws s3 cp s3://application-signals-preview-sdk/kotlin.zip ./`
- Android SDK: `aws s3 cp s3://application-signals-preview-sdk/andriod.zip ./`
- C++ SDK: `aws s3 cp s3://application-signals-preview-sdk/awsCppSdk.zip ./`
- PHP SDK: `aws s3 cp s3://application-signals-preview-sdk/awsSdkPhp.zip ./`
- Ruby SDK: `aws s3 cp s3://application-signals-preview-sdk/awsSdkRuby.zip ./`
- Go V2 SDK: `aws s3 cp s3://application-signals-preview-sdk/awsSdkGoV2.zip ./`
- Go V1 SDK: `aws s3 cp s3://application-signals-preview-sdk/go.zip ./`

- iOS SDK: `aws s3 cp s3://application-signals-preview-sdk/iOS.zip ./`

主題

- Application Signals 所需的許可
- 啟用 Application Signals
- 服務水準目標 (SLO)
- 使用 Application Signals 監控應用程式的運作狀態
- 收集的標準應用程式指標
- 使用綜合監控
- CloudWatch 明顯地執行發射和 A/B 實驗
- 使用 CloudWatch 朗姆酒

Application Signals 所需的許可

⚠ 應用程式訊號正在適用於 Amazon 的預覽版本中， CloudWatch 且可能會變更。

本節說明啟用、管理和操作 Application Signals 所需的許可。

啟用和管理 Application Signals 的許可

若要管理應用程式訊號，您必須使用下列權限登入：

```
{
 "Version": "2012-10-17",
 "Statement": [
    {
        "Sid": "CloudWatchApplicationSignalsFullAccessPermissions",
        "Effect": "Allow",
        "Action": "application-signals:*",
        "Resource": "*"
    },
    {
        "Sid": "CloudWatchApplicationSignalsAlarmsPermissions",
        "Effect": "Allow",
        "Action": [
```

```
            "cloudwatch:DescribeAlarms"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CloudWatchApplicationSignalsMetricsPermissions",
        "Effect": "Allow",
        "Action": [
            "cloudwatch:GetMetricData",
            "cloudwatch:ListMetrics"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CloudWatchApplicationSignalsLogGroupPermissions",
        "Effect": "Allow",
        "Action": [
            "logs:StartQuery",
            "logs:DescribeMetricFilters"
        ],
        "Resource": "arn:aws:logs:*:*:log-group:/aws/application-signals/data:*"
    },
    {
        "Sid": "CloudWatchApplicationSignalsLogsPermissions",
        "Effect": "Allow",
        "Action": [
            "logs:GetQueryResults",
            "logs:StopQuery"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CloudWatchApplicationSignalsSyntheticsPermissions",
        "Effect": "Allow",
        "Action": [
            "synthetics:DescribeCanaries",
            "synthetics:DescribeCanariesLastRun",
            "synthetics:GetCanaryRuns"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CloudWatchApplicationSignalsRumPermissions",
        "Effect": "Allow",
```

```
        "Action": [
            "rum:BatchCreateRumMetricDefinitions",
            "rum:BatchDeleteRumMetricDefinitions",
            "rum:BatchGetRumMetricDefinitions",
            "rum:GetAppMonitor",
            "rum:GetAppMonitorData",
            "rum:ListAppMonitors",
            "rum:PutRumMetricsDestination",
            "rum:UpdateRumMetricDefinition"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CloudWatchApplicationSignalsXrayPermissions",
        "Effect": "Allow",
        "Action": [
            "xray:GetTraceSummaries"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CloudWatchApplicationSignalsPutMetricAlarmPermissions",
        "Effect": "Allow",
        "Action": "cloudwatch:PutMetricAlarm",
        "Resource": [
            "arn:aws:cloudwatch:*:*:alarm:SLO-AttainmentGoalAlarm-*",
            "arn:aws:cloudwatch:*:*:alarm:SLO-WarningAlarm-*",
            "arn:aws:cloudwatch:*:*:alarm:SLI-HealthAlarm-*"
        ]
    },
    {
        "Sid": "CloudWatchApplicationSignalsCreateServiceLinkedRolePermissions",
        "Effect": "Allow",
        "Action": "iam:CreateServiceLinkedRole",
        "Resource": "arn:aws:iam::*:role/aws-service-role/application-
signals.cloudwatch.amazonaws.com/AWSServiceRoleForCloudWatchApplicationSignals",
        "Condition": {
            "StringLike": {
                "iam:AWSServiceName": "application-signals.cloudwatch.amazonaws.com"
            }
        }
    },
    {
        "Sid": "CloudWatchApplicationSignalsGetRolePermissions",
```

```
        "Effect": "Allow",
        "Action": "iam:GetRole",
        "Resource": "arn:aws:iam::*:role/aws-service-role/application-
signals.cloudwatch.amazonaws.com/AWSServiceRoleForCloudWatchApplicationSignals"
    },
    {
        "Sid": "CloudWatchApplicationSignalsSnsWritePermissions",
        "Effect": "Allow",
        "Action": [
            "sns:CreateTopic",
            "sns:Subscribe"
        ],
        "Resource": "arn:aws:sns:*:*:cloudwatch-application-signals-*"
    },
    {
        "Sid": "CloudWatchApplicationSignalsSnsReadPermissions",
        "Effect": "Allow",
        "Action": "sns:ListTopics",
        "Resource": "*"
    }
 ]
}
```

若要在 Amazon EC2 或自訂架構上啟用應用程式訊號，請參閱使用自訂設定在其他平台上啟用應用程式訊號。Kubernetes若要使用 Amazon 可 CloudWatch 觀測性 EKS 附加元件在 Amazon EKS 上啟用和管理應用程式訊號，您需要下列許可。

> **Important**
>
> 這些許可包括具有 Resource "*" 的 iam:PassRole 和具有 Resource "*" 的 eks:CreateAddon。這些都是強大的許可，授予它們時應小心。

```
{
"Version": "2012-10-17",
    "Statement": [
        {
"Sid": "CloudWatchApplicationSignalsEksAddonManagementPermissions",
            "Effect": "Allow",
            "Action": [
                "eks:AccessKubernetesApi",
                "eks:CreateAddon",
```

```
                "eks:DescribeAddon",
                "eks:DescribeAddonConfiguration",
                "eks:DescribeAddonVersions",
                "eks:DescribeCluster",
                "eks:DescribeUpdate",
                "eks:ListAddons",
                "eks:ListClusters",
                "eks:ListUpdates",
                "iam:ListRoles",
                "iam:PassRole"
            ],
            "Resource": "*"
        },
        {
"Sid":
 "CloudWatchApplicationSignalsEksCloudWatchObservabilityAddonManagementPermissions",
            "Effect": "Allow",
            "Action": [
                "eks:DeleteAddon",
                "eks:UpdateAddon"
            ],
            "Resource": "arn:aws:eks:*:*:addon/*/amazon-cloudwatch-observability/*"
        }
    ]
}
```

「應用程式訊號」儀表板會顯示 SLO 相關聯的 AWS Service Catalog AppRegistry應用程式。若要在 SLO 頁面中查看這些應用程式，您必須具備下列權限：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchApplicationSignalsTaggingReadPermissions",
            "Effect": "Allow",
            "Action": "tag:GetResources",
            "Resource": "*"
        }
    ]
}
```

正在運作的 Application Signals

使用應用程式訊號監控服務和 SLO 的服務營運商必須以下列唯讀權限登入帳戶：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchApplicationSignalsReadOnlyAccessPermissions",
            "Effect": "Allow",
            "Action": [
                "application-signals:BatchGet*",
                "application-signals:Get*",
                "application-signals:List*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsGetRolePermissions",
            "Effect": "Allow",
            "Action": "iam:GetRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/application-
signals.cloudwatch.amazonaws.com/AWSServiceRoleForCloudWatchApplicationSignals"
        },
        {
        "Sid": "CloudWatchApplicationSignalsLogGroupPermissions",
        "Effect": "Allow",
        "Action": [
            "logs:StartQuery",
            "logs:DescribeMetricFilters"
        ],
        "Resource": "arn:aws:logs:*:*:log-group:/aws/application-signals/data:*"
    },
    {
        "Sid": "CloudWatchApplicationSignalsLogsPermissions",
        "Effect": "Allow",
        "Action": [
            "logs:GetQueryResults",
            "logs:StopQuery"
        ],
        "Resource": "*"
    },
        {
```

```
            "Sid": "CloudWatchApplicationSignalsAlarmsReadPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsMetricsReadPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:ListMetrics"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsSyntheticsReadPermissions",
            "Effect": "Allow",
            "Action": [
                "synthetics:DescribeCanaries",
                "synthetics:DescribeCanariesLastRun",
                "synthetics:GetCanaryRuns"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsRumReadPermissions",
            "Effect": "Allow",
            "Action": [
                "rum:BatchGetRumMetricDefinitions",
                "rum:GetAppMonitor",
                "rum:GetAppMonitorData",
                "rum:ListAppMonitors"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsXrayReadPermissions",
            "Effect": "Allow",
            "Action": [
                "xray:GetTraceSummaries"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

若要查看 SLO 在「 AWS Service Catalog AppRegistry 應用程式訊號」儀表板中關聯的應用程式，您必須具備下列權限：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchApplicationSignalsTaggingReadPermissions",
            "Effect": "Allow",
            "Action": "tag:GetResources",
            "Resource": "*"
        }
    ]
}
```

若要檢查使用 Amazon 可 CloudWatch 觀測 EKS 附加元件的 Amazon EKS 上的應用程式訊號是否已啟用，您需要具有下列許可：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchApplicationSignalsEksReadPermissions",
            "Effect": "Allow",
            "Action": [
                "eks:ListAddons",
                "eks:ListClusters"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsEksDescribeAddonReadPermissions",
            "Effect": "Allow",
            "Action": [
                "eks:DescribeAddon"
            ],
            "Resource": "arn:aws:eks:*:*:addon/*/amazon-cloudwatch-observability/*"
        }
```

```
    ]
}
```

啟用 Application Signals

⚠ Application Signals 為預覽版本。如果您對此功能有任何意見，可以透app-signals-feedback過 @amazon .com 與我們聯絡。

本節中的主題說明如何在您的環境中啟用 CloudWatch 應用程式訊號。Amazon EKS 叢集支援 Application Signals，並使用主控台設定工作流程。其他平台 (包括 Amazon EC2) 上也支援，並具有自訂設定程序。

主題

- Application Signals 支援的系統
- OpenTelemetry 相容性考量
- 在 Amazon EKS 叢集上啟用 Application Signals
- 使用自訂設定在其他平台上啟用 Application Signals
- 疑難排解 Application Signals 安裝
- 設定 Application Signals

Application Signals 支援的系統

⚠ Application Signals 為預覽版本。如果您對此功能有任何意見，可以透app-signals-feedback過 @amazon .com 與我們聯絡。

Application Signals 在 Amazon EKS、Amazon ECS 和 Amazon EC2 上受到支援和測試。在 Amazon EC2 上啟用應用程式訊號的說明應該可以在任何支援 CloudWatch 代理程式和發行 AWS 版的平台上運作 OpenTelemetry，但這些指令尚未在其他平台上進行測試。

Java 相容性

應用程序信號支持 Java 應用程序，並支持與發行 AWS 版相同的 Java 庫和框架。 OpenTelemetry 如需詳細資訊，請參閱支援的程式庫、架構、應用程式伺服器和 JVM。

支援 JVM 版本 8、11 和 17。

Python 相容性

應用程序信號支持與 AWS 發行版相同的庫和框架。 OpenTelemetry 如需詳細資訊，請參閱 opentelemetry-python-contrib。

Python 援 3.8 及更新版本。

在您啟用 Python 應用程式的應用程式訊號之前，請注意下列考量事項。

- 在某些容器化應用程式中，遺失的PYTHONPATH環境變數有時可能會導致應用程式無法啟動。若要解決此問題，請務必將PYTHONPATH環境變數設定為應用程式工作目錄的位置。這是由於 OpenTelemetry 自動檢測的已知問題所致。有關此問題的更多信息，請參閱 PYTHONPATH 的 Python 自動檢測設置不兼容。
- 對於 Django 應用程序，還有其他必需的配置，這些配置在 OpenTelemetry Python 文檔中概述。
  - 使用`--noreload`旗標可防止自動重新載入。
  - 將`DJANGO_SETTINGS_MODULE`環境變量設置為 Django 應用程序`settings.py`文件的位置。這確保了可 OpenTelemetry 以正確訪問並與您的 Django 設置集成。

OpenTelemetry 相容性考量

⚠ Application Signals 為預覽版本。如果您對此功能有任何意見，可以透app-signals-feedback過 @amazon .com 與我們聯絡。

若要使用「應用程式訊號」啟動應用 CloudWatch 程式，建議您事先從應用程式中完全移除任何現有的應用程式效能監控解決方案 這包括刪除任何檢測程式碼和組態。

即使「應用程式訊號」使用 OpenTelemetry 儀器，也不能保證與您現有的 OpenTelemetry 儀器或組態相容。在最佳情況下，您可能可以保留某些 OpenTelemetry 功能，例如自訂指標。但是，請務必閱讀以下小節以獲取詳細資訊。

已使用的考量事項 OpenTelemetry

如果您已經與應用程式 OpenTelemetry 搭配使用，則本節的其餘部分包含重要資訊，以達到與應用程式訊號相容性的重要資訊。

- 在 OpenTelemetry 為應用程式啟用應用程式 Signals 之前，您必須根據應用程式移除任何其他自動檢測代理程式的插入。這有助於避免組態衝突。您可以繼續使用手動檢測，使用相容的 OpenTelemetry API 以及應用程式訊號。
- 如果使用手動檢測功能從應用程式中產生自訂範圍或指標，則根據檢測的複雜性而定，啟用 Application Signals 可能會導致它們停止產生資料或其他不良行為。您可能可以使用中的某些可用組態 OpenTelemetry (本節稍後表格中提到的組態除外)，以保留現有量度或範圍的所需行為。如需有關這些設定的詳細資訊，請參閱 OpenTelemetry 文件中的 SDK 組態。

  例如，透過使用OTEL_EXPORTER_OTLP_METRICS_ENDPOINT組態和自我管理的 OpenTelemetry Collector 執行個體，您可以繼續將自訂指標傳送到您想要的目的地。
- 某些環境變數或系統屬性不得搭配 Application Signals 使用，而只要遵循表格中的指引，就可以使用其他變數或系統屬性。請參閱下列資料表，了解詳細資訊。

環境變數	Application Signals 建議
一般環境變數	
OTEL_SDK_DISABLED	不得設定為 true。
OTEL_TRACES_EXPORTER	必須設定為 otlp。
OTEL_EXPORTER_OTLP_ENDPOINT	不得使用。
OTEL_EXPORTER_OTLP_TRACES_ENDPOINT	不得使用。
OTEL_ATTRIBUTE_COUNT_LIMIT	如果設定，則必須設定足夠高，以包含由 CloudWatch 應用程式訊號新增的大約 10 個跨度屬性。
OTEL_PROPAGATORS	如果設定，則必須包含用於結束追蹤的 xray。
OTEL_TRACES_SAMPLER	如果設定，必須設為 xray 以使用 X-Ray 集中抽樣。 若要使用本機取樣，請將此項設定為 parentbased_traceidratio ，並在

環境變數	Application Signals 建議
	OTEL_TRACES_SAMPLER_ARG 中指定取樣率。
OTEL_TRACES_SAMPLER_ARG	如果您使用的是 X-Ray 集中式追蹤樣本的預設值，則不得使用此變數。 如果您使用的是本機取樣，請在此變數中設定取樣率。例如，對於 5% 的取樣率，為 0.05。
特定於 Java 的環境變量	
OTEL_JAVA_ENABLED_RESOURCE_PROVIDERS	如果設定，則必須包含 AWS 資源偵測器。
蟒蛇特定的環境變量	
OTEL_PYTHON_CONFIGURATOR	如果使用，則必須設定為 aws_configurator
OTEL_PYTHON_DISTRO	如果使用，則必須設定為 aws_distro

在 Amazon EKS 叢集上啟用 Application Signals

⚠ Application Signals 為預覽版本。如果您對此功能有任何意見，可以透app-signals-feedback過 @amazon .com 與我們聯絡。

CloudWatch 在 Amazon EKS 叢集中執行的 Java 和 Python 應用程式支援應用程式訊號。若要在 Amazon EKS 叢集中啟用 Application Signals，您有下列兩種選擇：

- 若要在現有 Amazon EKS 叢集上為您的應用程式啟用 Application Signals，請使用 使用您的服務，在 Amazon EKS 叢集上啟用 Application Signals 中的步驟。
- 若要在非生產環境中使用範例應用程式來試用 Application Signals，請使用 使用範例應用程式在新的 Amazon EKS 叢集上啟用 Application Signals 中的指示。此工作流程使用 AWS 提供的指令碼來建立新的 Amazon EKS 叢集，並安裝針對 Application Signals 啟用的範例應用程式。這使您可以查看和測試應用程序信號的 end-to-end 功能。

主題

- 使用您的服務，在 Amazon EKS 叢集上啟用 Application Signals
- 使用範例應用程式在新的 Amazon EKS 叢集上啟用 Application Signals

使用您的服務，在 Amazon EKS 叢集上啟用 Application Signals

⚠ Application Signals 為預覽版本。如果您對此功能有任何意見，可以透app-signals-feedback過 @amazon .com 與我們聯絡。

若要在現有 Amazon EKS 叢集上的應用程式上啟用應用程式訊號， CloudWatch 請使用本節中的指示。

⚠ Important

如果您已經 OpenTelemetry 與要啟用「應用程式訊號」的應用程式搭配使用，請參閱啟用應用程式訊號OpenTelemetry 相容性考量之前。

在現有 Amazon EKS 叢集上為您的應用程式啟用 Application Signals

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇服務。
3. 如果尚未在此帳戶中啟用 Application Signals，則必須授予 Application Signals 所需的許可，以探索您的服務。為此，請執行下列操作。您的帳戶只需執行一次此操作。

   a. 選擇開始探索您的服務。

   b. 選取核取方塊，然後選擇開始探索服務。

      第一次在您的帳戶中完成此步驟會建立AWSServiceRoleForCloudWatchApplicationSignals服務連結角色。此角色會授予 Application Signals 下列許可：

      - `xray:GetServiceGraph`
      - `logs:StartQuery`
      - `logs:GetQueryResults`
      - `cloudwatch:GetMetricData`

- `cloudwatch:ListMetrics`
- `tag:GetResources`

如需有關此角色的詳細資訊，請參閱 CloudWatch 應用程式訊號的服務連結角色權限。

4. 選擇啟用 Application Signals。
5. 針對指定平台，選擇 EKS。
6. 針對選取 EKS 叢集，選取要啟用 Application Signals 的叢集。
7. 如果此叢集尚未啟用 Amazon 可 CloudWatch 觀測性 EKS 附加元件，系統會提示您啟用它。在此情況下，請執行下列操作：

   a. 選擇添加 CloudWatch 可觀察性 EKS 附加組件。Amazon EKS 主控台會出現。

   b. 選取 Amazon CloudWatch 可觀測性的核取方塊，然後選擇「下一步」。

   可 CloudWatch 觀測性 EKS 附加元件可提供應用程式訊號和 CloudWatch 容器洞見，並增強 Amazon EKS 的可觀察性。如需更多 Container Insights 的相關資訊，請參閱 Container Insights。

   c. 選取最新版本的附加元件進行安裝。

   d. 選取要用於附加元件的 IAM 角色。如果選擇從節點繼承，請將正確的許可附接至工作節點使用的 IAM 角色。取代*my-worker-node-role*為您的 Kubernetes 工作者節點所使用的 IAM 角色。

   ```
   aws iam attach-role-policy \
   --role-name my-worker-node-role \
   --policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
   --policy-arn arn:aws:iam::aws:policy/AWSXRayWriteOnlyAccess
   ```

   e. 如果想要建立服務角色以使用附加元件，請參閱 使用 Amazon CloudWatch 可觀測 EKS 附加元件安裝 CloudWatch 代理程式。

   f. 選擇下一步，確認畫面上的資訊，然後選擇建立。

   g. 在下一個屏幕中，選擇啟用 CloudWatch 應用程序信號返回 CloudWatch 控制台並完成該過程。

8. 啟用應用程式訊號的應用程式有兩個選項。為了保持一致性，建議您為每個叢集選擇一個選項。

   - 控制台選項更簡單。使用此方法會導致您的網繭立即重新啟動。
   - 註釋清單文件方法使您可以更好地控制網繭何時重新啟動，並且如果您不想集中化，還可以幫助您以更分散的方式管理監視。

Console

主控台選項使用 Amazon 可 CloudWatch 觀測性 EKS 附加元件的進階組態，為您的服務設定應用程式訊號。如需附加元件的詳細資訊，請參閱(選用) 額外組態。

如果您沒有看到工作負載和命名空間的清單，請確定您擁有檢視此叢集的正確權限。如需詳細資訊，請參閱必要權限。

您可以監視單一工作負載或整個命名空間。

監視單一工作負載：

1. 選取您要監督的工作負載旁的核取方塊。
2. 選取工作負載的語言。對於 Python 應用程式，請確定您的應用程式符合必要的先決條件，然後 如需詳細資訊，請參閱 啟用應用程序信號後，Python 應用程序不會啟動。
3. 選擇完成。Amazon 可 CloudWatch 觀測性 EKS 附加元件會立即將用於 OpenTelemetry 自動檢測 (ADOT) SDK 的發行 AWS 版插入到您的網繭中，並觸發網繭重新啟動以啟用應用程式指標和追蹤的收集。

監視整個命名空間：

1. 選取您要監視的命名空間旁邊的核取方塊。
2. 選取工作負載的語言。這會套用至此命名空間中的所有工作負載，無論它們目前已部署或將在 future 部署。對於 Python 應用程式，請確定您的應用程式符合必要的先決條件，然後 如需詳細資訊，請參閱 啟用應用程序信號後，Python 應用程序不會啟動。
3. 選擇完成。Amazon 可 CloudWatch 觀測性 EKS 附加元件會立即將用於 OpenTelemetry 自動檢測 (ADOT) SDK 的發行 AWS 版插入到您的網繭中，並觸發網繭重新啟動以啟用應用程式指標和追蹤的收集。

若要在其他 Amazon EKS 叢集中啟用 Application Signals，請從服務畫面中選擇啟用 Application Signals。

Annotate manifest file

在 CloudWatch 主控台中，[監視服務] 區段說明您必須將註解新增至叢集中的資訊清單 YAML。新增此注釋會自動檢測應用程式，以便將指標、追蹤和日誌傳送至 Application Signals。

有兩個注釋選項：

- 標註工作負載會自動檢測叢集中的單一工作負載。
- 標註命名空間會自動檢測所選命名空間中部署的所有工作負載。

選擇其中一個選項，然後遵循適當的步驟：

- 若要註解單一工作負載：
  1. 選擇標註工作負載。
  2. 將下列其中一行貼到工作負載資訊清單檔案的PodTemplate區段中。
     - 對於 Java 工作負載：`annotations: instrumentation.opentelemetry.io/inject-java: "true"`
     - 對於 Python 工作負載：`annotations: instrumentation.opentelemetry.io/inject-python: "true"`

       對於 Python 應用程序，還有其他必需的配置。如需詳細資訊，請參閱 啟用應用程序信號後，Python 應用程序不會啟動。
  3. 在終端中，輸入 `kubectl apply -f` *`your_deployment_yaml`* 以套用變更。
- 若要註解命名空間中的所有工作負載：
  1. 選擇標註命名空間。
  2. 將下列其中一行貼到命名空間資訊清單檔案的中繼資料區段中。如果命名空間同時包含 Java 和 Python 工作負載，請將這兩行貼到命名空間資訊清單檔案中。
     - 如果命名空間中有 Java 工作負載：`annotations: instrumentation.opentelemetry.io/inject-java: "true"`
     - 如果命名空間中有 Python 工作負載：`annotations: instrumentation.opentelemetry.io/inject-python: "true"`

       對於 Python 應用程序，還有其他必需的配置。如需詳細資訊，請參閱 啟用應用程序信號後，Python 應用程序不會啟動。
  3. 在終端中，輸入 `kubectl apply -f` *`your_namespace_yaml`* 以套用變更。
  4. 在終端中，輸入命令以重新啟動命名空間中的所有 Pod。重新啟動部署工作負載的範例命令為 `kubectl rollout restart deployment -n` *`namespace_name`*

9. 選擇完成後檢視服務。這會帶您前往 Application Signals Services 檢視，您可以在其中查看 Application Signals 正在收集的資料。可能需要幾分鐘的時間才會顯示資料。

若要在其他 Amazon EKS 叢集中啟用 Application Signals，請從服務畫面中選擇啟用 Application Signals。

如需服務檢視的詳細資訊，請參閱 使用 Application Signals 監控應用程式的運作狀態。

> **Note**
>
> 我們已經確定了一些考量，你應該記住時啟用 Python 應用程序的應用程序信號。如需詳細資訊，請參閱 啟用應用程序信號後，Python 應用程序不會啟動。

使用範例應用程式在新的 Amazon EKS 叢集上啟用 Application Signals

> **Application Signals 為預覽版本。如果您對此功能有任何意見，可以透app-signals-feedback過 @amazon .com 與我們聯絡。**

若要在使用範例 CloudWatch 應用程式進行檢測之前，先在範例應用程式上試用應用程式訊號，請遵循本節中的指示。這些指示使用指令碼來協助您建立 Amazon EKS 叢集、安裝範例應用程式，以及檢測範例應用程式以使用 Application Signals。

範例應用程式是由四個微服務組成的 Spring「寵物診所」應用程式。這些服務在 Amazon Amazon EC2 上的 Amazon EKS 上執行，並利用應用程式訊號啟用指令碼，透過 Java 或 Python 自動檢測代理程式啟用叢集。

需求

- 目前，應用程序信號僅監視 Java 和 Python 應用程序。
- 您必須在執行個體上 AWS CLI 安裝。我們建議使用 AWS CLI 版本 2，但版本 1 也應該可以使用。如需有關安裝的詳細資訊 AWS CLI，請參閱安裝或更新 AWS CLI.
- 本節中的指令碼旨在執行於 Linux 和 macOS 環境中。對於 Windows 執行個體，建議您使用 AWS Cloud9 環境來執行這些指令碼。如需有關的詳細資訊 AWS Cloud9，請參閱什麼是 AWS Cloud9？。
- 安裝支援的 `kubectl` 版本。您所使用的 `kubectl` 版本，必須與 Amazon EKS 叢集控制平面的版本差距在一個版本以內。例如，1.26 `kubectl` 用戶端可搭配使用 Kubernetes 1.25、1.26 和 1.27

版叢集。如果您已經有 Amazon EKS 叢集，則可能需要 AWS 為kubectl. 如需詳細資訊，請參閱針對 Amazon EKS 叢集建立或更新 kubeconfig 檔案。

- 安裝eksctl。 eksctl使 AWS CLI 用與互動 AWS，這意味著它使用與 AWS AWS CLI. 如需詳細資訊，請參閱安裝或更新 eksctl。
- 安裝 jq。需要 jq 才能執行 Application Signals 啟用指令碼。如需詳細資訊，請參閱下載 jq。

步驟 1：下載指令碼

若要使用範例應用程 CloudWatch 式下載指令碼以設定應用程式 Signals，您可以將壓縮的 GitHub 專案檔案下載並解壓縮到本機磁碟機，或者您也可以複製 GitHub 專案。

若要複製專案，請開啟終端視窗，並在指定工作目錄中輸入以下 Git 命令。

```
git clone https://github.com/aws-observability/application-signals-demo.git
```

步驟 2：建置並部署範例應用程式

若要建置和推播範例應用程式映像，請遵循下列指示。

步驟 3：部署並啟用 Application Signals 和範例應用程式

在完成下列步驟之前，請確定已完成 使用範例應用程式在新的 Amazon EKS 叢集上啟用 Application Signals 中列出的要求。

部署並啟用 Application Signals 和範例應用程式

1. 在您解壓縮啟動指令碼的本機終端中，輸入下列命令。以您要用於新叢集的名稱取*new-cluster-name*代。將區*###*替換為 AWS 區域的名稱，例如。us-west-1

   此命令會設定在已啟用 Application Signals 的新 Amazon EKS 叢集中執行的範例應用程式。

   ```
   # assuming the current working directory is 'onboarding'
   # this script sets up a new cluster, enables Application Signals, and deploys the
   # sample application
   cd application-signals-demo/scripts/eks/appsignals/one-step && ./setup.sh new-cluster-name region-name
   ```

   安裝指令碼大約需要 30 分鐘才能執行，並執行下列動作：

   - 在指定區域中建立新的 Amazon EKS 叢集。

- 為 Application Signals 建立必要的 IAM 許可 (`arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess` 和 `arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy`)。
- 透過安裝 CloudWatch 代理程式並自動檢 CloudWatch 測測量度和 X-Ray 追蹤的範例應用程式，以啟用應用程式訊號。
- 在相同的 Amazon EKS 叢集中部署 S PetClinic pring 範例應用程式。
- 創建五個 CloudWatch Synthetics 金絲雀，命名為`pc-add-vist`，，`pc-create-owners`，`pc-visit-pet`，`pc-visit-vet`。`pc-clinic-traffic`這些 canary 將以一分鐘的頻率執行，以產生範例應用程式的合成流量，並演示 Synthetics canary 如何出現在 Application Signals 中。
- 使用下列名稱為 PetClinic 應用程式建立四個服務等級目標 (SLO)：
  - 搜尋擁有者的可用性
  - 搜尋擁有者的延遲
  - 註冊擁有者的可用性
  - 註冊擁有者的延遲
- 使用自訂信任政策建立必要的 IAM 角色，並授予 Application Signals 下列許可：
  - `cloudwatch:PutMetricData`
  - `cloudwatch:GetMetricData`
  - `xray:GetServiceGraph`
  - `logs:StartQuery`
  - `logs:GetQueryResults`

2. (選擇性) 如果您要檢閱 PetClinic 範例應用程式的原始程式碼，您可以在根資料夾下找到它們。

```
- application-signals-demo
  - spring-petclinic-admin-server
  - spring-petclinic-api-gateway
  - spring-petclinic-config-server
  - spring-petclinic-customers-service
  - spring-petclinic-discovery-server
  - spring-petclinic-vets-service
  - spring-petclinic-visits-service
```

3. 若要檢視已部署的 PetClinic 範例應用程式，請執行下列命令以尋找 URL：

```
kubectl get ingress
```

步驟 4：監控範例應用程式

完成上一節中建立 Amazon EKS 叢集和部署範例應用程式的步驟後，可以使用 Application Signals 來監控應用程式。

> **Note**
>
> 若要讓 Application Signals 主控台開始填入，一些流量必須到達範例應用程式。先前步驟的一部分創建了 CloudWatch Synthetics 金絲雀，以生成流量到示例應用程序。

服務運作狀態監控

啟用之後，「CloudWatch 應用程式訊號」會自動探索並填入服務清單，而不需要任何額外的設定。

若要檢視探索到的服務清單並監控其運作狀態

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，依次選擇 Application Signals、服務。
3. 若要檢視服務、其操作及其相依性，請在清單中選擇其中一個服務的名稱。

   這個以應用程式為中心的統一檢視有助於全面了解使用者如何與您的服務互動。如果發生效能異常，這可協助您分類問題。如需有關服務檢視的完整詳細資訊，請參閱 使用 Application Signals 監控應用程式的運作狀態。
4. 選擇服務操作索引標簽，即可查看該服務操作的標準應用程式指標。例如，操作就是服務呼叫的 API 操作。

   然後，若要檢視該服務的單個操作的圖表，請選擇該操作名稱。
5. 選擇相依性索引標簽，查看應用程式具有的相依性，以及每個相依性的重要應用程式指標。相依性包括 AWS 應用程式呼叫的服務和協力廠商服務。
6. 若要從服務詳細資訊頁面中檢視相關追蹤，請在表格上方的三個圖表之一中選擇一個資料點。這會使用時間週期中篩選的追蹤填入新窗格。系統會根據您選擇的圖表對這些追蹤進行排序和篩選。例如，如果選擇延遲圖表，則會按照服務回應時間對追蹤排序。
7. 在 CloudWatch 主控台瀏覽窗格中，選擇 SLO。您會看到指令碼為範例應用程式建立的 SLO。如需 SLO 的詳細資訊，請參閱 服務水準目標 (SLO)。

(選用) 步驟 5：清除

完成測試 Application Signals 後，可以使用 Amazon 提供的指令碼來清除和刪除帳戶中為範例應用程式建立的成品。若要執行清除，請輸入下列命令。*new-cluster-name*替換為您為示例應用程序創建的集群的名稱，並將 *region*-name 替換為 AWS 區域的名稱，例如us-west-1。

```
cd application-signals-demo/scripts/eks/appsignals/one-step && ./cleanup.sh new-
cluster-name region-name
```

使用自訂設定在其他平台上啟用 Application Signals

⚠ Application Signals 為預覽版本。如果您對此功能有任何意見，可以透app-signals-feedback過 @amazon .com 與我們聯絡。

使用這些章節中的自訂設定步驟，在 Amazon EKS 以外的平台上啟用應用 CloudWatch 程式訊號。在這些架構上，您可以 OpenTelemetry 自行安裝和設定 CloudWatch 代理程式和 AWS 發行版。

在這些架構上，Application Signals 不會自動探索您的服務或其叢集或主機的名稱。您必須在自訂設定期間指定這些名稱，而您指定的名稱是顯示在 Application Signals 儀表板中的名稱。

主題

- 使用自訂設定在 Amazon ECS 上啟用 Application Signals
- 使用自訂設定在 Amazon EC2 和其他平台上啟用 Application Signals

使用自訂設定在 Amazon ECS 上啟用 Application Signals

⚠ Application Signals 為預覽版本。如果您對此功能有任何意見，可以透app-signals-feedback過 @amazon .com 與我們聯絡。

使用這些自訂設定說明，將 Amazon ECS 上的應用程式上架到 CloudWatch 應用程式訊號。您可以 OpenTelemetry 自行安裝並設定 CloudWatch 代理程式和 AWS 發行版。

在 Amazon ECS 叢集上，Application Signals 不會自動探索服務名稱或執行服務的叢集。您必須在自訂設定期間指定這些名稱，而您指定的名稱是顯示在 Application Signals 儀表板中的名稱。

> **Important**
>
> 僅支援 awsvpc 網路模式。

步驟 1：在您的帳戶中啟用 Application Signals

如果尚未在此帳戶中啟用 Application Signals，則必須授予 Application Signals 所需的許可，以探索您的服務。為此，請執行下列操作。您的帳戶只需執行一次此操作。

為您的應用程式啟用 Application Signals

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇服務。
3. 選擇開始探索您的服務。
4. 選取核取方塊，然後選擇開始探索服務。

   第一次在您的帳戶中完成此步驟會建立AWSServiceRoleForCloudWatchApplicationSignals服務連結角色。此角色會授予 Application Signals 下列許可：

   - xray:GetServiceGraph
   - logs:StartQuery
   - logs:GetQueryResults
   - cloudwatch:GetMetricData
   - cloudwatch:ListMetrics
   - tag:GetResources

   如需有關此角色的詳細資訊，請參閱 CloudWatch 應用程式訊號的服務連結角色權限。

步驟 2：建立 IAM 角色

您必須建立兩個 IAM 角色。如果您已經建立了這些角色，則可能需要為其新增許可。

- ECS 任務角色 — 容器使用此角色來執行。權限應該是您的應用程序需要的任何內容，再加上CloudWatchAgentServerPolicy和AWSXRayWriteOnlyAccess。

- ECS 任務執行角色 – Amazon ECS 使用此角色來啟動和執行容器。如果您已經創建了這個角色，請將 AmazonECS SSM ReadOnlyAccess，亞馬遜TaskExecutionRolePolicy和政策附加到它。CloudWatchAgentServerPolicy

  如果您需要存放更敏感的資料以供 Amazon ECS 使用，請參閱指定敏感資料以取得詳細資訊。

如需建立 IAM 角色的詳細資訊，請參閱建立 IAM 角色。

步驟 3：準備 CloudWatch 代理程式組態

首先，在啟用 Application Signals 的情況下準備代理程式組態。若要執行此操作，請建立名為 `/tmp/ecs-cwagent.json` 的本機檔案。

```
{
  "traces": {
    "traces_collected": {
      "app_signals": {}
    }
  },
  "logs": {
    "metrics_collected": {
      "app_signals": {}
    }
  }
}
```

將此組態上傳至 SSM 參數存放區。若要進行這項動作，請輸入下列指令。在檔案中，將 *$REGION* 取代為實際區域名稱。

```
aws ssm put-parameter \
--name "ecs-cwagent" \
--type "String" \
--value "`cat /tmp/ecs-cwagent.json`" \
--region "$REGION"
```

步驟 4：向 CloudWatch 代理商檢測您的申請

下一步是檢測您的應用信號 CloudWatch 應用程序。

Java

透過代理程式在 Amazon ECS 上測試您的應用程式 CloudWatch

1. 首先，指定綁定掛載。在接下來的步驟中，該磁碟區將用於跨容器共用檔案。將在此程序的後續步驟中使用此綁定掛載。

```
"volumes": [
  {
    "name": "opentelemetry-auto-instrumentation"
  }
]
```

2. 新增 CloudWatch 代理程式並行定義。因此，請將名為 `ecs-cwagent` 的新容器附加到應用程式的任務定義。將 *$REGION* 取代為實際區域名稱。取代為 Amazon 彈性容器登錄上最新 CloudWatch 容器映像的路徑。如需詳細資訊，請參閱 Amazon ECR 上的 cloudwatch-agent。

```
{
  "name": "ecs-cwagent",
  "image": "$IMAGE",
  "essential": true,
  "secrets": [
    {
      "name": "CW_CONFIG_CONTENT",
      "valueFrom": "ecs-cwagent"
    }
  ],
  "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
      "awslogs-create-group": "true",
      "awslogs-group": "/ecs/ecs-cwagent",
      "awslogs-region": "$REGION",
      "awslogs-stream-prefix": "ecs"
    }
  }
}
```

3. 將新容器 `init` 附加到應用程式的任務定義。用 OpenTelemetry Amazon ECR 映像庫發行 AWS 版######## $ IMAGE。

```
{
```

```
  "name": "init",
  "image": "$IMAGE",
  "essential": false,
  "command": [
    "cp",
    "/javaagent.jar",
    "/otel-auto-instrumentation/javaagent.jar"
  ],
  "mountPoints": [
    {
      "sourceVolume": "opentelemetry-auto-instrumentation",
      "containerPath": "/otel-auto-instrumentation",
      "readOnly": false
    }
  ]
}
```

4. 將下列環境變數新增至應用程式容器。如需詳細資訊，請參閱

環境變數	啟用 Application Signals 的設定
OTEL_RESOURCE_ATTRIBUTES	將 $SVC_NAME 取代為應用程式名稱。這將顯示為 Application Signals 儀表板中的應用程式名稱。 將 $HOST_ENV 取代為執行應用程式的主機環境。這將顯示為 Application Signals 儀表板中應用程式的 Hosted In 環境。
OTEL_AWS_APP_SIGNALS_ENABLED	設定為true以啟用應用程式訊號 SpanMetricsProcessor。
OTEL_METRICS_EXPORTER	設定為 none 以停用其他指標匯出工具。
OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT	設定為http://127.0.0.1:4315 將量度傳送至並 CloudWatch 行。
OTEL_EXPORTER_OTLP_TRACES_ENDPOINT	設定為http://127.0.0.1:4315 將軌跡傳送至 CloudWatch 邊車。
OTEL_TRACES_SAMPLER	將 X-Ray 定義為追蹤取樣器。

環境變數	啟用 Application Signals 的設定
OTEL_PROPAGATORS	將 X-Ray 新增為其中一個傳播者。
JAVA_TOOL_OPTIONS	為 OpenTelemetry Java 代理程式注入發 AWS 行版。

5. 掛載您在此程序步驟 1 中定義的磁碟區 opentelemetry-auto-instrumentation。

   對於 Java 應用程序，請使用以下內容。

```
{
  "name": "app",
   ...
  "environment": [
    {
      "name": "OTEL_RESOURCE_ATTRIBUTES",
      "value": "aws.hostedin.environment=$HOST_ENV,service.name=$SVC_NAME"
    },
    {
      "name": "OTEL_AWS_APP_SIGNALS_ENABLED",
      "value": "true"
    },
    {
      "name": "OTEL_METRICS_EXPORTER",
      "value": "none"
    },
    {
      "name": "JAVA_TOOL_OPTIONS",
      "value": " -javaagent:/otel-auto-instrumentation/javaagent.jar"
    },
    {
      "name": "OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT",
      "value": "http://127.0.0.1:4315"
    },
    {
      "name": "OTEL_TRACES_SAMPLER",
      "value": "xray"
    },
    {
      "name": "OTEL_EXPORTER_OTLP_TRACES_ENDPOINT",
      "value": "http://127.0.0.1:4315"
    },
```

```
    {
      "name": "OTEL_PROPAGATORS",
      "value": "tracecontext,baggage,b3,xray"
    }
  ],
  "mountPoints": [
    {
      "sourceVolume": "opentelemetry-auto-instrumentation",
      "containerPath": "/otel-auto-instrumentation",
      "readOnly": false
    }
  ]
}
```

Python

在您啟用 Python 應用程式的應用程式訊號之前，請注意下列考量事項。

- 在某些容器化應用程式中，遺失的PYTHONPATH環境變數有時可能會導致應用程式無法啟動。若要解決此問題，請務必將PYTHONPATH環境變數設定為應用程式工作目錄的位置。這是由於 OpenTelemetry 自動檢測的已知問題所致。有關此問題的更多信息，請參閱 PYTHONPATH 的 Python 自動檢測設置不兼容。
- 對於 Django 應用程序，還有其他必需的配置，這些配置在 OpenTelemetry Python 文檔中概述。
  - 使用`--noreload`旗標可防止自動重新載入。
  - 將`DJANGO_SETTINGS_MODULE`環境變量設置為 Django 應用程序`settings.py`文件的位置。這確保了可 OpenTelemetry 以正確訪問並與您的 Django 設置集成。

使用代理程式在 Amazon ECS 上檢測您的 Python 應用程式 CloudWatch

1. 首先，指定綁定掛載。在接下來的步驟中，該磁碟區將用於跨容器共用檔案。將在此程序的後續步驟中使用此綁定掛載。

```
"volumes": [
  {
    "name": "opentelemetry-auto-instrumentation-python"
  }
]
```

2. 新增 CloudWatch 代理程式並行定義。因此，請將名為 `ecs-cwagent` 的新容器附加到應用程式的任務定義。將 *$REGION* 取代為實際區域名稱。取代為 Amazon 彈性容器登錄上最新 CloudWatch 容器映像的路徑。如需詳細資訊，請參閱 Amazon ECR 上的 cloudwatch-agent。

```
{
  "name": "ecs-cwagent",
  "image": "$IMAGE",
  "essential": true,
  "secrets": [
    {
      "name": "CW_CONFIG_CONTENT",
      "valueFrom": "ecs-cwagent"
    }
  ],
  "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
      "awslogs-create-group": "true",
      "awslogs-group": "/ecs/ecs-cwagent",
      "awslogs-region": "$REGION",
      "awslogs-stream-prefix": "ecs"
    }
  }
}
```

3. 將新容器 `init` 附加到應用程式的任務定義。用 OpenTelemetry Amazon ECR 映像庫發行 AWS 版######## $ IMAGE。

```
{
    "name": "init",
    "image": "$IMAGE",
    "essential": false,
    "command": [
        "cp",
        "-a",
        "/autoinstrumentation/.",
        "/otel-auto-instrumentation-python"
    ],
    "mountPoints": [
        {
            "sourceVolume": "opentelemetry-auto-instrumentation-python",
            "containerPath": "/otel-auto-instrumentation-python",
            "readOnly": false
```

```
        }
    ]
}
```

4. 將下列環境變數新增至應用程式容器。如需詳細資訊，請參閱

環境變數	啟用 Application Signals 的設定
OTEL_RESOURCE_ATTRIBUTES	將 $SVC_NAME 取代為應用程式名稱。這將顯示為 Application Signals 儀表板中的應用程式名稱。 將 $HOST_ENV 取代為執行應用程式的主機環境。這將顯示為 Application Signals 儀表板中應用程式的 Hosted In 環境。
OTEL_AWS_APP_SIGNALS_ENABLED	設定為true以啟用應用程式訊號 SpanMetricsProcessor。
OTEL_METRICS_EXPORTER	設定為 none 以停用其他指標匯出工具。
OTEL_EXPORTER_OTLP_PROTOCOL	設定為http/protobuf 可使 CloudWatch 用 HTTP 將度量和追蹤傳送至。
OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT	設定為http://127.0.0.1:4316/v1/metrics 可將量度傳送至並 CloudWatch 行。
OTEL_EXPORTER_OTLP_TRACES_ENDPOINT	設定為http://127.0.0.1:4316/v1/traces 將軌跡傳送至 CloudWatch 邊車。
OTEL_TRACES_SAMPLER	將 X-Ray 定義為追蹤取樣器。
OTEL_PROPAGATORS	將 X-Ray 新增為其中一個傳播者。
OTEL_PYTHON_DISTRO	設定為aws_distro 以使用亞多 Python 分析儀器。
OTEL_PYTHON_CONFIGURATOR	設定aws_configuration 為使用亞多 Python 組態。

環境變數	啟用 Application Signals 的設定
PYTHONPATH	取代$APP_PATH 為容器內應用程式工作目錄的位置。Python 解釋器需要這樣才能找到您的應用程序模塊。
DJANGO_SETTINGS_MODULE	僅適用於 Django 應用程式。將其設置為 Django 應用程序settings.py 文件的位置。取代$PATH_TO_SETTINGS 。

5. 掛載您在此程序步驟 1 中定義的磁碟區 opentelemetry-auto-instrumentation-python。

   對於 Python 應用程序，請使用以下內容。

```
{
    "name": "app",
    ...
    "environment": [
        {
            "name": "PYTHONPATH",
            "value": "/otel-auto-instrumentation-python/opentelemetry/
instrumentation/auto_instrumentation:$APP_PATH:/otel-auto-instrumentation-
python"
        },
        {
            "name": "OTEL_EXPORTER_OTLP_PROTOCOL",
            "value": "http/protobuf"
        },
        {
            "name": "OTEL_TRACES_SAMPLER",
            "value": "xray"
        },
        {
            "name": "OTEL_TRACES_SAMPLER_ARG",
            "value": "endpoint=http://localhost:2000"
        },
        {
            "name": "OTEL_LOGS_EXPORTER",
            "value": "none"
        },
        {
```

```
            "name": "OTEL_PYTHON_DISTRO",
            "value": "aws_distro"
        },
        {
            "name": "OTEL_PYTHON_CONFIGURATOR",
            "value": "aws_configurator"
        },
        {
            "name": "OTEL_EXPORTER_OTLP_TRACES_ENDPOINT",
            "value": "http://localhost:4316/v1/traces"
        },
        {
            "name": "OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT",
            "value": "http://localhost:4316/v1/metrics"
        },
        {
            "name": "OTEL_METRICS_EXPORTER",
            "value": "none"
        },
        {
            "name": "OTEL_AWS_APP_SIGNALS_ENABLED",
            "value": "true"
        },
        {
            "name": "OTEL_RESOURCE_ATTRIBUTES",
            "value": "aws.hostedin.environment=$HOST_ENV,service.name=$SVC_NAM"
        },
        {
            "name": "DJANGO_SETTINGS_MODULE",
            "value": "$PATH_TO_SETTINGS.settings"
        }
    ],
    "mountPoints": [
        {
            "sourceVolume": "opentelemetry-auto-instrumentation-python",
            "containerPath": "/otel-auto-instrumentation-python",
            "readOnly": false
        }
    ]
}
```

步驟 5：部署應用程式

建立任務定義的新修訂版本，並將其部署到應用程式叢集。應該會在新建立的任務中看到三個容器：

- `init`
- `ecs-cwagent`
- `app`

使用自訂設定在 Amazon EC2 和其他平台上啟用 Application Signals

⚠ Application Signals 為預覽版本。如果您對此功能有任何意見，可以透app-signals-feedback過 @amazon .com 與我們聯絡。

對於在 Amazon EC2 上執行的應用程式和其他非 Amazon EKS 架構上執行的應用程式，您可以自行安裝和設定 CloudWatch 代理程式和 AWS 發行 OpenTelemetry 版。在使用自訂 Application Signals 設定啟用的這些架構上，Application Signals 不會自動探索您的服務或其執行所在的叢集或主機的名稱。您必須在自訂設定期間指定這些名稱，而您指定的名稱是顯示在 Application Signals 儀表板中的名稱。

以下步驟已在 Amazon EC2 執行個體上進行了測試，但也預計可以在支援發行 AWS 版的其他架構上運作。 OpenTelemetry

需求

- 若要取得應用程式訊號的支援，您必須同時使用最新版本的 CloudWatch代理程式和代理程 OpenTelemetry 式 AWS 發行版。
- 您必須在執行個體上 AWS CLI 安裝。我們建議使用 AWS CLI 版本 2，但版本 1 也應該可以使用。如需有關安裝的詳細資訊 AWS CLI，請參閱安裝或更新 AWS CLI.

⚠ Important

如果您已經 OpenTelemetry 與要啟用「應用程式訊號」的應用程式搭配使用，請參閱啟用應用程式訊號OpenTelemetry 相容性考量之前。

步驟 1：在您的帳戶中啟用 Application Signals

如果尚未在此帳戶中啟用 Application Signals，則必須授予 Application Signals 所需的許可，以探索您的服務。為此，請執行下列操作。您的帳戶只需執行一次此操作。

為您的應用程式啟用 Application Signals

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇服務。
3. 選擇開始探索您的服務。
4. 選取核取方塊，然後選擇開始探索服務。

   第一次在您的帳戶中完成此步驟會建立AWSServiceRoleForCloudWatchApplicationSignals服務連結角色。此角色會授予 Application Signals 下列許可：

   - `xray:GetServiceGraph`
   - `logs:StartQuery`
   - `logs:GetQueryResults`
   - `cloudwatch:GetMetricData`
   - `cloudwatch:ListMetrics`
   - `tag:GetResources`

   如需有關此角色的詳細資訊，請參閱 CloudWatch 應用程式訊號的服務連結角色權限。

步驟 2：下載並啟動 CloudWatch 代理程式

在 Amazon EC2 執行個體上啟用應用程 CloudWatch 式訊號時安裝代理程式

1. 將最新版本的 CloudWatch 代理程式下載至執行個體。如果執行個體已安裝 CloudWatch 代理程式，您可能需要更新它。只有 2023 年 11 月 30 日或更新版本發行的代理程式支援 CloudWatch 應用程式訊號。

   如需有關下載 CloudWatch 代理程式的資訊，請參閱下載 CloudWatch 代理程式套件。
2. 在您啟動 CloudWatch 代理程式之前，請先將其設定為啟用應用程式訊號。下列範例是 CloudWatch 代理程式組態，可針對 EC2 主機上的指標和追蹤啟用應用程式訊號。

   可以輸入以下命令來建立此檔案：

```
vim amazon-cloudwatch-agent.json
```

新增以下內容作為此檔案的內容。

```
{
  "traces": {
    "traces_collected": {
      "app_signals": {}
    }
  },
  "logs": {
    "metrics_collected": {
      "app_signals": {}
    }
  }
}
```

3. 將CloudWatchAgentServerPolicy和 AWSXrayWriteOnlyAccessIAM 政策附加到 Amazon EC2 執行個體的 IAM 角色。

   a. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。

   b. 選擇角色並查找 Amazon EC2 執行個體使用的角色。然後選擇該角色的名稱。

   c. 在許可索引標籤中，依序選擇新增許可、附接政策。

   d. 尋找CloudWatchAgentServerPolicy。如有需要，請使用搜尋方塊。然後，選取該政策的核取方塊，並選擇新增許可。

   e. 尋找AWSXrayWriteOnlyAccess。如有需要，請使用搜尋方塊。然後，選取該政策的核取方塊，並選擇新增許可。

4. 輸入下列命令以啟動 CloudWatch 代理程式。以 CloudWatch 代理程式組態檔的路徑取*agent-config-file-path*代，例如`./amazon-cloudwatch-agent.json`。必須包含如下所示的`file:`字首。

```
export CONFIG_FILE_PATH=./amazon-cloudwatch-agent.json
```

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl \
-a fetch-config \
-m ec2 -s -c file:$CONFIG_FILE_PATH
```

步驟 3：檢測您的應用程式並啟動它

下一步是檢測您的應用信號 CloudWatch 應用程序。

Java

在 Amazon EC2 執行個體上啟用應用程式訊號時，檢測 Java 應用程式

1. 下載適用於 OpenTelemetry Java 自動檢測 AWS 代理程式的最新版本。可以使用此連結來下載最新版本。您可以在「發行」中檢視有關所有已發行 aws-otel-java-instrumentation 版本的資訊
2. 若要最佳化 Application Signals 優勢，請在啟動應用程式之前，使用環境變數提供其他資訊。此資訊將顯示在 Application Signals 儀表板中。

   a. 對於 OTEL_RESOURCE_ATTRIBUTES 變數，請將下列資訊指定為索引鍵/值對：

      - aws.hostedin.environment 會設定執行應用程式的環境。這將顯示為 Application Signals 儀表板中應用程式的 Hosted In 環境。此屬性金鑰僅供應用程式訊號使用，並會轉換為 X-Ray 追蹤註解和度 CloudWatch 量維度。如果您不提供此索引鍵的值，則會使用預設值 Generic。
      - service.name 會設定服務名稱。這將顯示為 Application Signals 儀表板中應用程式的服務名稱。如果您不提供此索引鍵的值，則會使用預設值 unknown_service。

   b. 對於 OTEL_EXPORTER_OTLP_TRACES_ENDPOINT 變數，請指定要接收追蹤匯出的基礎端點 URL。 CloudWatch 代理程式將 4315 公開為其 OLTP 連接埠。在 Amazon EC2 上，因為應用程式會與本機 CloudWatch 代理程式通訊，所以您應將此值設為 OTEL_EXPORTER_OTLP_TRACES_ENDPOINT=http://localhost:4315

   c. 對於 OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT 變數，請指定要接收指標匯出的基礎端點 URL。 CloudWatch 代理程式將 4315 公開為其 OLTP 連接埠。在 Amazon EC2 上，因為應用程式會與本機 CloudWatch 代理程式通訊，所以您應將此值設為 OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT=http://localhost:4315

   d. 對於JAVA_TOOL_OPTIONS變數，請指定儲存 OpenTelemetry Java 自動檢測代理程式的發行 AWS 版的路徑。

      ```
      export JAVA_TOOL_OPTIONS=' -javaagent:$ADOT_AGENT_PATH'
      ```

      例如：

```
export ADOT_AGENT_PATH=./aws-opentelemetry-agent.jar
```

e. 對於 OTEL_METRICS_EXPORTER 變數，建議將值設定為 none。這會停用其他指標匯出工具，以便只使用 Application Signals 匯出器。

f. 對於OTEL_AWS_APP_SIGNALS_ENABLED變數，透過OTEL_AWS_APP_SIGNALS_ENABLED將true設定為啟用 SpanMetricProcessor (SMP)。這會從追蹤中產生 Application Signals 指標。

3. 使用上一個步驟中討論的環境變數來啟動應用程式。以下是啟動指令碼的範例。

```
JAVA_TOOL_OPTIONS=' -javaagent:$ADOT_AGENT_PATH' \
OTEL_METRICS_EXPORTER=none \
OTEL_AWS_APP_SIGNALS_ENABLED=true \
OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT=http://localhost:4315 \
OTEL_EXPORTER_OTLP_TRACES_ENDPOINT=http://localhost:4315 \
OTEL_RESOURCE_ATTRIBUTES=aws.hostedin.environment=$YOUR_HOST_ENV,service.name=
$YOUR_SVC_NAME \
java -jar $MY_JAVA_APP.jar
```

Python

在 Amazon EC2 執行個體上啟用應用程式訊號時，測試您的 Python 應用程式

1. 下載最新版本的 OpenTelemetry Python 自動檢測 AWS 代理程式的發行版。請執行下列 命令進行安裝。

```
pip install aws-opentelemetry-distro
```

您可以在 AWS OpenTelemetry Python 儀器的發行版中查看有關所有已發行版本的信息。

2. 若要最佳化 Application Signals 優勢，請在啟動應用程式之前，使用環境變數提供其他資訊。此資訊將顯示在 Application Signals 儀表板中。

a. 對於 OTEL_RESOURCE_ATTRIBUTES 變數，請將下列資訊指定為索引鍵/值對：

- aws.hostedin.environment 會設定執行應用程式的環境。這將顯示為 Application Signals 儀表板中應用程式的 Hosted In 環境。此屬性金鑰僅供應用程式訊號使用，並會轉換為 X-Ray 追蹤註解和度 CloudWatch 量維度。如果您不提供此索引鍵的值，則會使用預設值 Generic。

- `service.name` 會設定服務名稱。這將顯示為 Application Signals 儀表板中應用程式的服務名稱。如果您不提供此索引鍵的值，則會使用預設值 `unknown_service`。

b. 對於`OTEL_EXPORTER_OTLP_PROTOCOL`變數，請指`http/protobuf`定透過 HTTP 將遙測資料匯出至下列步驟中列出的 CloudWatch 代理程式端點。

c. 對於 `OTEL_EXPORTER_OTLP_TRACES_ENDPOINT` 變數，請指定要接收追蹤匯出的基礎端點 URL。 CloudWatch 代理程式會透過 HTTP 公開 4316 做為其 OLTP 連接埠。在 Amazon EC2 上，因為應用程式會與本機 CloudWatch 代理程式通訊，所以您應將此值設為 `OTEL_EXPORTER_OTLP_TRACES_ENDPOINT=http://localhost:4316/v1/traces`

d. 對於 `OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT` 變數，請指定要接收指標匯出的基礎端點 URL。 CloudWatch 代理程式會透過 HTTP 公開 4316 做為其 OLTP 連接埠。在 Amazon EC2 上，因為應用程式會與本機 CloudWatch 代理程式通訊，所以您應將此值設為 `OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT=http://localhost:4316/v1/metrics`

e. 對於 `OTEL_METRICS_EXPORTER` 變數，建議將值設定為 none。這會停用其他指標匯出工具，以便只使用 Application Signals 匯出器。

f. 對於`OTEL_AWS_APP_SIGNALS_ENABLED`變數，啟用 SpanMetricProcessor by 設定`OTEL_AWS_APP_SIGNALS_ENABLED`為`true`。這會從追蹤中產生 Application Signals 指標。

3. 使用上一個步驟中討論的環境變數來啟動應用程式。以下是啟動指令碼的範例。

- 將 `$HOST_ENV` 取代為執行應用程式的主機環境。這會在應用程式訊號儀表板中顯示為應用程式的託管環境。
- 將 `$SVC_NAME` 取代為應用程式名稱。這將在應用程序信號儀表板中顯示為應用程序的名稱。
- 替換為您`$PYTHON_APP`的應用程序的位置和名稱。

```
OTEL_METRICS_EXPORTER=none \
OTEL_LOGS_EXPORTER=none \
OTEL_AWS_APP_SIGNALS_ENABLED=true \
OTEL_PYTHON_DISTRO=aws_distro \
OTEL_PYTHON_CONFIGURATOR=aws_configurator \
OTEL_EXPORTER_OTLP_PROTOCOL=http/protobuf \
OTEL_TRACES_SAMPLER=xray \
OTEL_TRACES_SAMPLER_ARG="endpoint=http://localhost:2000" \
OTEL_AWS_APP_SIGNALS_EXPORTER_ENDPOINT=http://localhost:4316/v1/metrics \
```

```
OTEL_EXPORTER_OTLP_TRACES_ENDPOINT=http://localhost:4316/v1/traces \
OTEL_RESOURCE_ATTRIBUTES=aws.hostedin.environment=$HOST_ENV,service.name=
$SVC_NAME \
opentelemetry-instrument python $PYTHON_APP.py
```

在您啟用 Python 應用程式的應用程式訊號之前，請注意下列考量事項。

- 在某些容器化應用程式中，遺失的PYTHONPATH環境變數有時可能會導致應用程式無法啟動。若要解決此問題，請務必將PYTHONPATH環境變數設定為應用程式工作目錄的位置。這是由於 OpenTelemetry 自動檢測的已知問題所致。有關此問題的更多信息，請參閱 PYTHONPATH 的 Python 自動檢測設置不兼容。
- 對於 Django 應用程序，還有其他必需的配置，這些配置在 OpenTelemetry Python 文檔中概述。
  - 使用`--noreload`旗標可防止自動重新載入。
  - 將`DJANGO_SETTINGS_MODULE`環境變量設置為 Django 應用程序`settings.py`文件的位置。這確保了可 OpenTelemetry 以正確訪問並與您的 Django 設置集成。

疑難排解 Application Signals 安裝

⚠ Application Signals 為預覽版本。如果您對此功能有任何意見，可以透app-signals-feedback過 @amazon .com 與我們聯絡。

本節包含 CloudWatch 應用程式訊號的疑難排解秘訣。

主題

- 啟用 Application Signals 後，應用程式無法啟動。
- 啟用應用程序信號後，Python 應用程序不會啟動
- 遙測數據丟失 CloudWatch 和 X-Ray
- 相依性指標具有未知的值
- 管理 Amazon CloudWatch 觀察 EKS 附加組件 ConfigurationConflict 時處理

啟用 Application Signals 後，應用程式無法啟動。

如果在叢集上啟用 Application Signals 後，Amazon EKS 叢集上的應用程式並未啟動，請檢查下列事項：

- 檢查應用程式是否已由另一個監控解決方案進行檢測。Application Signals 不支援與其他檢測解決方案共存。
- 確認您的應用程式符合使用 Application Signals 的相容性要求。如需詳細資訊，請參閱 Application Signals 支援的系統 。
- 如果您的應用程式無法提取應用程式訊號構件，例如 OpenTelemetery Java AWS 版或 Python 代理程式和 CloudWatch 代理程式映像，則可能是網路問題。

若要緩解此問題，請從應用程式部署資訊清單`instrumentation.opentelemetry.io/inject-python: "true"`中移除註解`instrumentation.opentelemetry.io/inject-java: "true"`，然後重新部署應用程式。然後檢查應用程式是否正常運作。

啟用應用程序信號後，Python 應用程序不會啟動

OpenTelemetry 自動檢測中的一個已知問題，遺漏的`PYTHONPATH`環境變數有時會導致應用程式無法啟動。若要解決此問題，請確定您將`PYTHONPATH`環境變數設定為應用程式工作目錄的位置。有關此問題的更多信息，請參閱 PYTHONPATH 的 Python 自動檢測設置與 Python 的模塊解析行為不兼容，從而破壞了 Django 應用程序。

對於 Django 應用程序，還有其他必需的配置，這些配置在 OpenTelemetry Python 文檔中概述。

- 使用`--noreload`旗標可防止自動重新載入。
- 將`DJANGO_SETTINGS_MODULE`環境變量設置為 Django 應用程序`settings.py`文件的位置。這確保了可 OpenTelemetry 以正確訪問並與您的 Django 設置集成。

遙測數據丟失 CloudWatch 和 X-Ray

如果 Application Signals 儀表板中遺失指標或追蹤，則可能是如下原因。只有在上次更新後等待 Application Signals 收集並顯示資料 15 分鐘後，才會調查這些原因。

- 請確定您使用的程式庫和架構受到 ADOT Java 代理程式的支援。如需詳細資訊，請參閱程式庫/框架。
- 確定 CloudWatch 代理程式正在執行。首先檢查 CloudWatch 代理程式網繭的狀態，並確定它們都處於`Running`狀態。

```
kubectl -n amazon-cloudwatch get pods.
```

將下列項目新增至 CloudWatch 代理程式組態檔以啟用偵錯記錄檔，然後重新啟動代理程式。

```
"agent": {
>>>>>>> streams
  "region": "${REGION}",
  "debug": true
},
```

然後檢查 CloudWatch 代理程式網繭中是否有錯誤。

- 檢查代 CloudWatch 理程式的組態問題。確認下列項目仍在 CloudWatch 代理程式組態檔中，且代理程式新增後已重新啟動。

```
"agent": {
  "region": "${REGION}",
  "debug": true
},
```

然後檢查 OpenTelemetry 調試日誌中的錯誤消息，例如`ERROR io.opentelemetry.exporter.internal.grpc.OkHttpGrpcExporter - Failed to export ....` 這些訊息可能表示有問題。

如果不能解決問題，請透過使用 `kubectl describe pod` 命令描述 pod，轉儲並檢查名稱以 `OTEL_` 開頭的環境變數。

- 若要啟用 OpenTelemetry Python 偵錯記錄，請將環境變數設定`OTEL_PYTHON_LOG_LEVEL`為`debug`並重新部署應用程式。
- 檢查從 CloudWatch代理程式匯出資料的權限是否有錯誤或不足。如果您在 CloudWatch 代理程式記錄檔中看到`Access Denied`訊息，這可能是問題所在。您安裝 CloudWatch 代理程式時套用的權限可能稍後變更或撤銷。
- 產生遙測資料時，請檢查發 AWS 行版是否有 OpenTelemetry (ADOT) 問題。

  請確定檢測註解 `instrumentation.opentelemetry.io/inject-java` 和 `sidecar.opentelemetry.io/inject-java` 套用至應用程式部署，且值為 `true`。如果沒有這些註解，即使 ADOT 附加元件已正確安裝，也不會檢測應用程式 Pod。

接下來，檢查 Init 容器是否已套用於應用程式，並且 Ready 狀態為 True。如果 init 容器尚未就緒，請參閱原因狀態。

如果問題仍然存在，請執行下列動作以在 OpenTelemetry Java SDK 上啟用偵錯記錄。然後尋找以 ERROR io.telemetry 開頭的訊息。

若要啟用偵錯日誌，請將環境變數 OTEL_JAVAAGENT_DEBUG 設定為 true，然後重新部署應用程式。

- 指標/範圍匯出程式可能正在捨棄資料。要找出答案，請檢查應用程式日誌中包含 Failed to export... 的訊息
- 將度量或跨度傳送至應用程式訊號時， CloudWatch 代理程式可能會受到限制。檢查 CloudWatch 代理程式記錄檔中是否有指示節流的訊息。

相依性指標具有未知的值

如果您在 Application Signals 儀表板中看到UnknownOperationUnknownRemoteService、或UnknownRemoteOperation相依性名稱或作業，請檢查未知遠端服務和未知遠端作業的資料點是否與其部署重合。這是 Application Signals 上的已知問題，並計劃在未來版本中進行更正。

管理 Amazon CloudWatch 觀察 EKS 附加組件 ConfigurationConflict 時處理

當您安裝或更新 Amazon CloudWatch Observability EKS 附加元件時，如果您注意到某種類型Health IssueConfigurationConflict的失敗，說明開頭為Conflicts found when trying to apply. Will not continue due to resolve conflicts mode，很可能是因為您已經在叢集上安裝了 CloudWatch 代理程式及其相關元件，例如 ServiceAccount、 ClusterRole 和安 ClusterRoleBinding 裝在叢集上。當附加元件嘗試安裝 CloudWatch 代理程式及其相關元件時，如果偵測到內容有任何變更，依預設會無法安裝或更新，以避免覆寫叢集上的資源狀態。

如果您嘗試上載 Amazon CloudWatch Observability EKS 附加元件，但發現此失敗，建議您刪除先前安裝在叢集上的現有 CloudWatch 代理程式設定，然後安裝 EKS 附加元件。請務必備份您對原始 CloudWatch 代理程式設定所做的任何自訂 (例如自訂代理程式組態)，並在下次安裝或更新時將這些自訂提供給 Amazon CloudWatch Observability EKS 附加元件。如果您之前已安裝 CloudWatch 代理程式以便上線至容器深入解析，請參閱以取得刪除容器見解的 CloudWatch 代理程式和流利位元詳細資訊。

或者，附加元件也支援衝突解決組態選項，該選項可指定 OVERWRITE。您可以使用此選項覆寫叢集上的衝突來繼續安裝或更新附加元件。如果您使用 Amazon EKS 主控台，則在建立或更新附加元件時選

擇可選組態設定，可找到衝突解決方法。如果您使用的是 AWS CLI，您可以`--resolve-conflicts OVERWRITE`將指令提供給以建立或更新附加元件。

設定 Application Signals

> ⚠ Application Signals 為預覽版本。如果您對此功能有任何意見，可以透app-signals-feedback過 @amazon .com 與我們聯絡。

本節包含設定 CloudWatch 應用程式訊號的相關資訊。

追蹤取樣率

根據預設，當您啟用 Application Signals 時，會使用 `reservoir=1/s` 和 `fixed_rate=5%` 預設取樣率設定來啟用 X-Ray 集中取樣。 OpenTelemetry (ADOT) SDK 代理程式發行 AWS 版的環境變數設定如下。

環境變數	Value	注意
`OTEL_TRACES_SAMPLER`	`xray`	
`OTEL_TRACES_SAMPLER_ARG`	`endpoint=http://cloudwatch-agent.amazon-cloudwatch:2000`	CloudWatch 代理程式的端點

如需有關變更取樣組態的資訊，請參閱下列各項：

- 若要變更 X-Ray 取樣，請參閱自訂取樣規則
- 若要變更 ADOT 取樣，請參閱設定 X-Ray 遠端取樣的 OpenTelemetry 收集器

如果想要停用 X-Ray 集中式取樣並改用本機取樣，請為 ADOT SDK Java 代理程式設定如下所示的值。下列範例將取樣率設定為 5%。

環境變數	Value
`OTEL_TRACES_SAMPLER`	`parentbased_traceidratio`

環境變數	Value
OTEL_TRACES_SAMPLER_ARG	0.05

如需有關更多進階取樣設定的資訊，請參閱 OTEL_TRACES_SAMPLER。

管理高基數作業

應用程式信號包括 CloudWatch 代理程式中的設定，您可以使用這些設定來管理作業的基數，以及管理指標匯出以最佳化成本。根據預設，當服務在一段時間內的不同作業數目超過預設臨界值 500 時，測量結果限制函數會變成作用中。您可以透過調整組態設定來調整行為。

判斷是否已啟動量度限制

您可以使用下列方法來判斷是否發生預設量度限制。如果是這樣，您應該考慮按照下一節中的步驟來優化基數控制。

- 在 CloudWatch 主控台中，選擇應用程式訊號、服務。如果您看到名為AllOtherOperations或具名的作業 AllOtherRemoteOperations，則會發生量度限制。RemoteOperation
- 如果應用程式信號收集的任何度量具有其Operation維度的值AllOtherOperations，則會發生量度限制。
- 如果應用程式信號收集的任何度量具有其RemoteOperation維度的值AllOtherRemoteOperations，則會發生量度限制。

優化基數控制

若要最佳化基數控制，您可以執行下列動作：

- 建立自訂規則以彙總作業。
- 設定指標限制原則。

建立自訂規則以彙總作業

高基數操作有時可能是由從上下文中提取的不適當唯一值引起的。例如，傳送在路徑中包含使用者 ID 或工作階段 ID 的 HTTP/S 要求可能會導致數百個不同的作業。若要解決此類問題，建議您使用自訂規則設定 CloudWatch 代理程式，以重新撰寫這些作業。

如果透過個別RemoteOperation呼叫 (例如、和類似要求) 產生許多不同的指標 PUT /api/customer/owners/123PUT /api/customer/owners/456，我們建議您將這些作業合併為單一作業RemoteOperation。一種方法是將以統一格式開頭的所有RemoteOperation呼叫標準化，特別PUT /api/customer/owners/{ownerId}是。PUT /api/customer/owners/下列的範例示範了這一點。如需其他自訂規則的資訊，請參閱〈〉啟用 CloudWatch 應用程式信。

```
{
   "logs":{
      "metrics_collected":{
         "app_signals":{
            "rules":[
               {
                  "selectors":[
                     {
                        "dimension":"RemoteOperation",
                        "match":"PUT /api/customer/owners/*"
                     }
                  ],
                  "replacements":[
                     {
                        "target_dimension":"RemoteOperation",
                        "value":"PUT /api/customer/owners/{ownerId}"
                     }
                  ],
                  "action":"replace"
               }
            ]
         }
      }
   }
}
```

在其他情況下，高基數量度量可能已彙總至AllOtherRemoteOperations，而且可能還不清楚包含哪些特定量度。 CloudWatch 代理程式能夠記錄捨棄的作業。若要識別捨棄的作業，請使用下列範例中的設定來啟動記錄，直到問題再次出現為止。然後檢查 CloudWatch 代理程式記錄 (可透過容器stdout或 EC2 記錄檔存取) 並搜尋關鍵字drop metric data。

```
{
  "agent": {
    "config": {
      "agent": {
        "debug": true
```

```
      },
      "traces": {
        "traces_collected": {
          "app_signals": {
          }
        }
      },
      "logs": {
        "metrics_collected": {
          "app_signals": {
            "limiter": {
              "log_dropped_metrics": true
            }
          }
        }
      }
    }
  }
}
```

建立指標限制政策

如果預設量度限制組態未解決服務的基數，您可以自訂指標限制器組態。若要這麼做，請在 CloudWatch 代理程式組態檔案的`logs/metrics_collected/app_signals`區`limiter`段下新增區段。

下列範例會將量度限制的臨界值從 500 個不同的量度降低為 100。

```
{
  "logs": {
    "metrics_collected": {
      "app_signals": {
        "limiter": {
          "drop_threshold": 100
        }
      }
    }
  }
}
```

服務水準目標 (SLO)

⚠ Application Signals 為預覽版本。如果您對此功能有任何意見，可以透app-signals-feedback過 @amazon .com 與我們聯絡。

可以使用 Application Signals，為關鍵業務營運服務建立服務水準目標。通過在這些服務上創建 SLO，您將能夠在 SLO 儀表板上跟踪它們，從而可以 at-a-glance 查看最重要的操作。

除了建立操作員可用來查看關鍵操作目前狀態的快速檢視之外，您還可以使用 SLO 來追蹤服務的長期效能，以確保它們符合您的期望。如果您與客戶達成服務水準協議，SLO 是確保滿足客戶的絕佳工具。

使用 SLO 評估服務的運作狀態首先要根據關鍵效能指標 (服務水準指標 (SLI)) 來設定明確、可衡量的目標。SLO 會根據您設定的閾值和目標來追蹤 SLI 效能，並報告應用程式效能與閾值之間的距離。

Application Signals 可協助您在關鍵效能指標上設定 SLO。Application Signals 會自動收集它發現的每個服務和操作的 `Latency` 和 `Availability` 指標，而且這些指標通常非常適合用作 SLI。透過 SLO 建立精靈，您可以將這些指標用於 SLO。然後，您可以使用 Application Signals 儀表板來追蹤所有 SLO 的狀態。

可以針對服務呼叫或使用的特定操作設定 SLO。除了使用`Latency`和度 CloudWatch 量之外，您還可以使用任何量度或`Availability`量度表示式做為 SLI。

創建 SLO 對於從 CloudWatch應用程序信號中獲得最大收益非常重要。建立 SLO 之後，可以在 Application Signals 主控台中檢視其狀態，以快速查看哪些重要服務和操作執行良好以及哪些運作狀態不佳。使用 SLO 進行追蹤具有下列主要好處：

- 您的服務營運商可以更輕鬆地查看根據 SLI 所測量的關鍵服務的當前運行狀況。然後，他們可以快速分類和識別運作狀態不佳的服務和操作。
- 您可以在較長時間內針對可衡量的業務目標來追蹤服務績效。

透過選擇要設定 SLO 的內容，可以優先考慮對您重要的內容。Application Signals 儀表板會自動顯示您優先選擇的內容資訊。

建立 SLO 時，您也可以選擇同時建立 CloudWatch 警示來監視 SLO。可以設定警示來監控閾值違規情況以及警告等級。如果 SLO 指標超出您設定的閾值，或者如果它們接近警告閾值，這些警示會自動通

知您。例如，接近其警告閾值的 SLO 會通知您，您的團隊可能需要減慢應用程式中的流失速度，以確保實現長期效能目標。

主題

- SLO 概念
- 建立 SLO
- 檢視和分類 SLO 狀態
- 編輯現有 SLO
- 刪除 SLO

SLO 概念

SLO 包含下列要素：

- 服務水準指標 (SLI)，這是您指定的主要效能指標。它表示應用程式所需的效能水準。Application Signals 會自動收集它發現的服務和操作的關鍵指標 `Latency` 和 `Availability`，而且這些指標通常非常適合用作 SLO。

  可以選擇用於 SLI 的閾值。例如，200 毫秒的延遲。
- 目標或達成目標，也就是 SLI 預期在每個時間間隔內達到閾值的時間百分比。時間間隔可以短至幾小時或長達一年。

  間隔可以是行事曆間隔或滾動間隔。
  - 行事曆間隔會與行事曆對齊，例如每月追蹤的 SLO。 CloudWatch 根據一個月中的天數，自動調整健康狀況、預算和成就數。行事曆間隔更適合按行事曆進行衡量的商業目標。
  - 滾動間隔是在滾動基礎上計算。滾動間隔更適合追蹤應用程式的最新使用者體驗。
- 該時段長度較短，許多時段構成一個間隔。將應用程式的效能與間隔內每個時段的 SLI 進行比較。在每個時段，確定應用程式是否已達到必要效能。

例如，行事曆間隔為一天且週期為 1 分鐘的 99% 目標，表示應用程式必須在一天的 1 分鐘週期的 99% 內達到或實現成功閾值。如果是這樣，則當天實現 SLO。第二天是新的評估間隔，而且應用程式必須在第二天 1 分鐘週期的 99% 內達到或實現成功閾值，才能實現第二天的 SLO。

SLI 可以基於 Application Signals 收集的新標準應用程式指標之一。或者，它可以是任何 CloudWatch 量度或量度表示式。可用於 SLI 的標準應用程式指標為 `Latency` 和

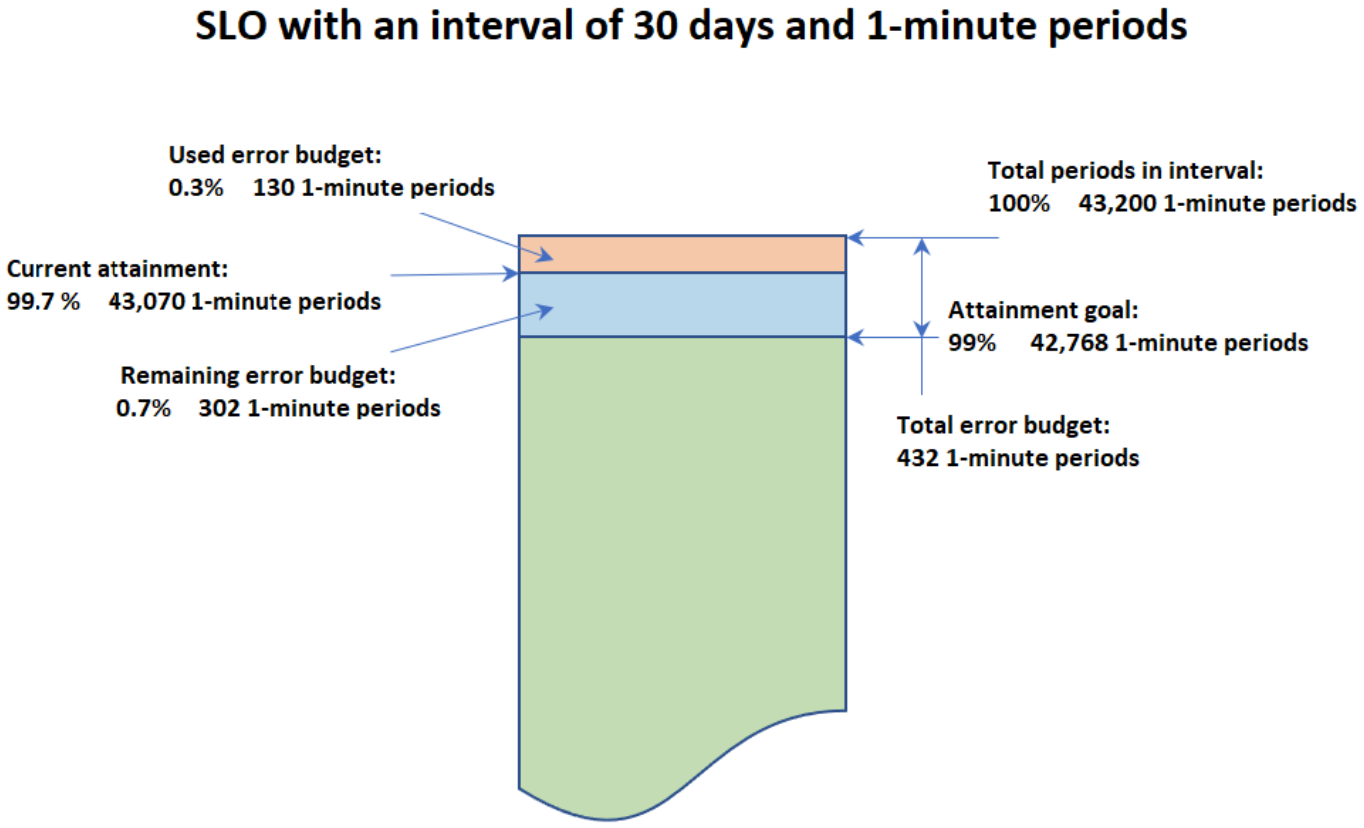
Availability。Availability 表示成功回應除以請求總數。它的計算方式為 (1 - 故障率)*100，其中故障回應為 5xx 錯誤。成功回應是沒有 5XX 錯誤的回應。4XX 回應會被視為成功。

> **Note**
>
> 目前僅支援基於時段的計算。計劃在未來版本中支援數量型或請求型計算。

計算錯誤預算和實現

當您檢視 SLO 的相關資訊時，會看到其目前運作狀態及其錯誤預算。錯誤預算是突破閾值但仍可滿足 SLO 的間隔內的時間量。總誤差預算是在整個間隔內可以容忍的違規時間總量。剩餘錯誤預算是目前間隔期間可以容忍的剩餘違規時間量。這是在從總錯誤預算中減去已經發生的違規時間量之後。

下圖說明間隔為 30 天、週期為 1 分鐘且達成目標為 99% 的目標的達成與錯誤預算概念。30 天包括 43,200 個 1 分鐘。43,200 的 99% 是 42,768，因此該月中的 42,768 分鐘必須運作正常，才能實現 SLO。到目前為止，在目前的間隔中，有 130 個 1 分鐘運作不佳。



確定每個週期內的成功

在每個週期內，SLI 資料會根據用於 SLI 的統計資料彙總為單一資料點。此資料點表示週期的整個長度。系統會將該單一資料點與 SLI 閾值進行比較，以判斷週期是否正常。在儀表板上查看目前時間範圍內的運作不佳週期，可能會提醒您的服務營運商需要對服務進行分類。

如果確定週期運作狀態不佳，則整個週期長度將根據錯誤預算計算為失敗。追蹤錯誤預算可讓您了解服務是否在較長時間內達到您想要的效能。

建立 SLO

建議在關鍵應用程式中同時設定延遲和可用性 SLO。Application Signals 收集的這些指標符合共同的業務目標。

您也可以在產生單一時間序列的任何 CloudWatch 量度或任何公制數學運算式上設定 SLO。

第一次在帳戶中建立 SLO 時， CloudWatch 會自動在您的帳戶中建立AWSServiceRoleForCloudWatchApplicationSignals服務連結角色 (如果尚未存在)。此服務連結角色可 CloudWatch 讓您從帳戶中的應用程式收集記 CloudWatch 錄資料、X-Ray 追蹤資料、 CloudWatch 指標資料，以及標記資料。如需 CloudWatch 服務連結角色的詳細資訊，請參閱使用 CloudWatch 的服務連結角色。

若要建立 SLO

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇服務水準目標 (SLO)。
3. 選擇建立 SLO。
4. 輸入 SLO 的名稱。包括服務或操作的名稱，以及適當的關鍵字 (例如延遲或可用性)，可協助您快速識別分類期間 SLO 狀態所指示的內容。
5. 對於設定服務水準指標 (SLI)，執行下列其中一項操作：
   - 在標準應用程式指標 `Latency` 或 `Availability` 中設定 SLO：

     a. 選擇服務操作。

     b. 選取此 SLO 將監控的服務。

     c. 選取此 SLO 將監控的操作。

     選取服務與選取操作下拉式清單由過去 24 小時內的作用中服務與操作所填入。

     d. 選擇可用性或延遲，然後設定閾值。

- 若要在任何 CloudWatch 度量或公 CloudWatch 制數學運算式上設定 SLO，請執行下列動作：

  a. 選擇「CloudWatch 量度」。

  b. 選擇 「選取 CloudWatch 量度」。

     選取指標畫面會出現。使用瀏覽或查詢索引標籤來查找所需的指標，或建立指標數學運算式。

     選取想要的指標之後，請選擇圖形化指標索引標籤，然後選取要用於 SLO 的統計資料和週期。然後選擇 Select metric (選取指標)。

     如需這些畫面的詳細資訊，請參閱 將指標圖形化 和 將數學運算式新增至 CloudWatch 圖表。

  c. 對於設定條件，請選取 SLO 的比較運算子和閾值，以用作成功指標。

6. 如果在步驟 5 中選取服務操作，可以選擇性地選擇其他設定，然後調整此 SLO 的週期長度。
7. 設定 SLO 的間隔和達成目標。如需有關間隔與達成目標及其如何同時運作的相關資訊，請參閱 SLO 概念。
8. (選擇性) 為 SLO 設定一或多個警 CloudWatch 示或警告臨界值。

   a. CloudWatch 警示可以使用 Amazon SNS 在應用程式運作狀態不佳時根據其 SLI 效能主動通知您。

      若要建立警示，請選取其中一個警示核取方塊，然後輸入或建立 Amazon SNS 主題，以便在警示進入 ALARM 狀態時用於通知。如需 CloudWatch 警示的詳細資訊，請參閱 使用 Amazon CloudWatch 警報。建立警示會產生費用。如需有關 CloudWatch 定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

   b. 如果設定警告閾值，它會出現在 Application Signals 畫面中，以協助您識別尚未實現的 SLO，即使它們目前運作良好。

      若要設定警告閾值，請在警告閾值中輸入閾值。當 SLO 的錯誤預算低於警告閾值時，SLO 會在多個 Application Signals 畫面中標記為警告。警告閾值也會出現在錯誤預算圖表中。也可以建立基於警告閾值的 SLO 警告警示。

9. 若要將標籤新增至此 SLO，請選擇標籤索引標籤，然後選擇新增新標籤。標籤可協助您管理、識別、組織、搜尋及篩選資源。如需有關標記的詳細資訊，請參閱標記 AWS 資源。

> **Note**
>
> 如果此 SLO 相關的應用程式已在中註冊 AWS Service Catalog AppRegistry，您可以使用`awsApplication`標籤將此 SLO 與中的應用程式建立關聯。 AppRegistry如需詳細資訊，請參閱什麼是 AppRegistry？

10. 選擇建立 SLO。如果也選擇建立一個或多個警示，按鈕名稱會變更以進行反映。

檢視和分類 SLO 狀態

您可以使用 CloudWatch 主控台中的服務等級目標或服務選項，快速查看 SLO 的健全狀況。[服務] 檢視可提供狀況不良服務比率的 at-a-glance 檢視，根據您所設定的 SLO 計算。如需有關使用服務選項的詳細資訊，請參閱 使用 Application Signals 監控應用程式的運作狀態。

服務水準目標檢視可宏觀了解您的組織。可總體上查看已實現和未實現的 SLO。根據您選擇的 SLI，這可讓您了解在較長時間內，有多少服務和操作符合您的期望。

若要使用「服務水準目標」檢視來檢視所有 SLO

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇服務水準目標 (SLO)。

   服務水準目標 (SLO) 清單隨即出現。

   可以在 SLI 狀態欄中快速查看 SLO 的目前狀態。若要排序 SLO，讓所有狀況不佳的 SLO 都位於清單頂端，請選擇 SLI 狀態欄，直到狀態不佳的 SLO 全部位於最上方。

   SLO 資料表包含以下預設資料欄。可以選擇清單上方的齒輪圖示來調整要顯示的資料欄。如需有關目標、SLI、達成目標及間隔的詳細資訊，請參閱 SLO 概念。

   - SLO 的名稱。
   - 目標資料欄會顯示每個間隔內必須順利達到 SLI 閾值才能實現 SLI 目標的週期百分比。它也會顯示 SLO 的間隔長度。
   - SLI 狀態會顯示應用程式目前的操作狀態是否正常。如果目前所選時間範圍內的任何週期對於 SLO 而言狀況不佳，則 SLI 狀態會顯示狀況不佳。
   - 最終成就是指截至所選時間範圍結束時達到的成就水準。依此資料欄排序，查看最有可能無法實現的 SLO。

- 成就差異是所選時間範圍的開始與結束之間的成就水準差異。負差值表示指標呈下降趨勢。依此資料欄排序，查看 SLO 的最新趨勢。
- 結束錯誤預算 (%) 是指週期內可能有狀態不佳週期但仍可順利實現 SLO 的總時間百分比。如果將此值設定為 5%，且間隔中剩餘週期的 5% 或更少的 SLI 狀況不佳，則仍會成功實現 SLO。
- 錯誤預算差異是指所選時間範圍開始與結束之間的錯誤預算差異。負差值表示指標呈下降趨勢。
- 結束錯誤預算 (時間) 是指運作狀態不佳但仍可成功實現 SLO 的間隔中的實際時間量。例如，如果為 14 分鐘，則若 SLI 在剩餘間隔期間運作不佳的時間少於 14 分鐘，仍然可以成功實現 SLO。
- 服務、操作和類型資料欄會顯示設定為此 SLO 之服務與操作的相關資訊。

3. 若要查看 SLO 的達成與錯誤預算圖，請選擇 SLO 名稱旁的選項按鈕。

   頁面頂端的圖形會顯示 SLO 達成與錯誤預算狀態。也會顯示與此 SLO 相關聯之 SLI 指標的圖形。

4. 若要進一步分類不符合其目標的 SLO，請選擇與該 SLO 相關聯的服務名稱或操作名稱。將轉至詳細資訊頁面，可以在其中進一步分類。如需詳細資訊，請參閱 使用服務詳細資訊頁面檢視詳細的服務活動和作業狀態。
5. 若要變更頁面中圖表和資料表的時間範圍，請選擇靠近畫面頂端的新時間範圍。

編輯現有 SLO

請依照下列步驟編輯現有 SLO。編輯 SLO 時，只能變更閾值、間隔、達成目標和標籤。若要變更其他方面 (例如服務、操作或指標)，請建立新的 SLO，而非編輯現有 SLO。

變更 SLO 部分核心組態 (例如週期或閾值) 會使先前所有資料點以及有關達成效果與運作狀態的評估失效。它可有效刪除並重新建立 SLO。

> **Note**
>
> 如果編輯 SLO，與該 SLO 相關聯的警示不會自動更新。可能需要更新警示，以使與 SLO 保持同步。

編輯現有 SLO

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇服務水準目標 (SLO)。

3. 選擇您要編輯之 SLO 旁的選項按鈕，然後選擇動作 > 編輯 SLO。
4. 進行變更，然後選擇儲存變更。

刪除 SLO

請依照下列步驟刪除現有 SLO。

> **Note**
>
> 刪除 SLO 時，與該 SLO 相關聯的警示不會自動刪除。您需要自行刪除它們。如需詳細資訊，請參閱 管理警示。

刪除 SLO

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇服務水準目標 (SLO)。
3. 選擇您要編輯之 SLO 旁的選項按鈕，然後選擇動作 > 刪除 SLO。
4. 選擇確認。

使用 Application Signals 監控應用程式的運作狀態

> **Warning**
>
> Application Signals 為預覽版本。如果您對此功能有任何意見，可以透app-signals-feedback過@amazon .com 與我們聯絡。

在CloudWatch 主控台中使用「應用程式訊號」來監視和疑難排解應用程式的運作狀態：

- 監控應用程式服務 – 作為日常營運監控的一部分，請使用服務頁面查看所有服務的摘要。查看故障率或延遲最高的服務，並查看哪些服務的服務水準指標 (SLI) 不良。選取服務以開啟服務詳細資訊頁面，並查看詳細指標、服務操作、Synthetics Canaries 和用戶端請求。這可協助您疑難排解並確定操作問題的根本原因。
- 檢查應用程式拓撲 – 使用服務地圖來了解和監控一段時間內的應用程式拓撲，包括用戶端、Synthetics Canaries、服務和相依性之間的關係。立即查看服務水準指標 (SLI) 狀況，並檢視呼叫量、故障率和延遲等關鍵指標。深入查看服務詳細資訊頁面中的更多詳細資訊。

探索範例案例，示範如何使用這些頁面快速疑難排解操作服務運作狀態問題，從初始偵測到識別根本原因。

Application Signals 如何啟用操作運作狀態監控

啟用 Application Signals 的應用程式之後，會自動探索您的應用程式服務、API 及其相依性，並顯示在服務、服務詳細資訊和服務地圖頁面中。Application Signals 會從多個來源收集資訊，以啟用服務探索和操作運作狀態監控：

- AWS 適用於 OpenTelemetry (ADOT) 的發行版 — 作為啟用應用程序信號的一部分，OpenTelemetry Java 自動檢測庫配置為發出代理程序收集的指標和跟踪。 CloudWatch 指標和追蹤可用來探索服務、操作、相依性和其他服務資訊。
- 服務水準目標 (SLO) – 在您為服務建立服務水準目標後，「服務」、「服務詳細資訊」和「服務地圖」頁面會顯示服務水準指標 (SLI) 運作狀態。SLI 可監控延遲、可用性和其他操作指標。
- CloudWatch Synthetics 金絲雀 — 當您在加那利群島上設定 X-Ray 追蹤時，從初期測試指令碼對服務的呼叫會與您的服務相關聯，並顯示在「服務詳細資料」頁面中。
- CloudWatch 真實使用者監視 (RUM) — 在 R CloudWatch UM Web 用戶端上啟用 X-Ray 追蹤時，對服務的要求會自動關聯並顯示在服務詳細資料頁面中。
- AWS Service Catalog AppRegistry— 應用程序信號會自動發現您帳戶中的 AWS 資源，並允許您將它們分組到中創建的邏輯應用程序中。 AppRegistry「服務」頁面中顯示的應用程式名稱是基於執行服務的基礎運算資源。

> **Note**
>
> Application Signals 會根據您選擇的目前時間篩選器中發出的指標和追蹤來顯示您的服務和操作。(根據預設，這是過去三個小時。) 如果服務、操作、相依性、Synthetics Canary 或用戶端頁面的目前時間篩選條件內沒有任何活動，則不會顯示。
> 目前最多可以顯示 1,000 條服務。探索服務和服務拓撲最多可能會延遲 10 分鐘。評估服務水準指標 (SLI) 運作狀態可能會延遲 15 分鐘。

使用「服務」頁面檢視整體服務活動和操作運作狀態

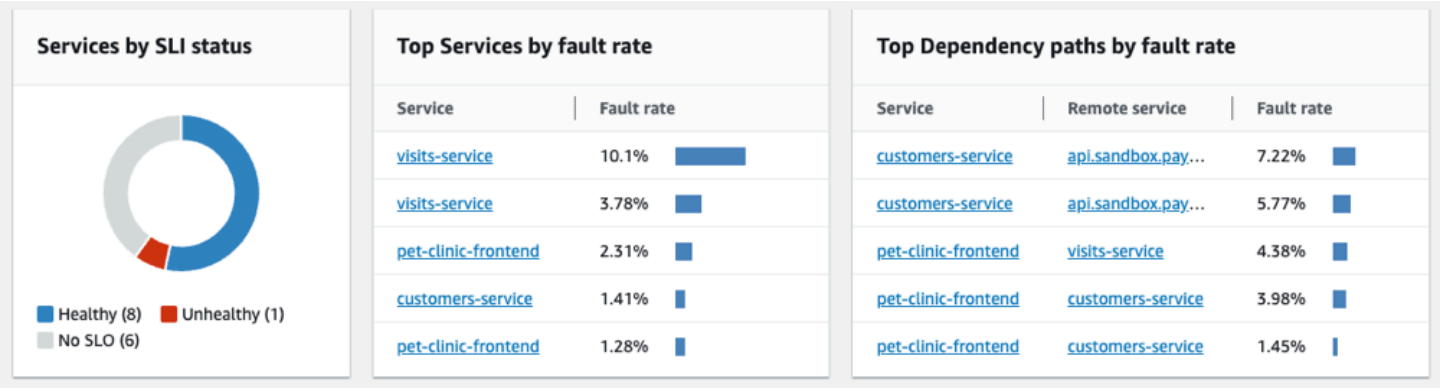
⚠ 應用程式訊號正在適用於 Amazon 的預覽版本中， CloudWatch 且可能會有所變更。

可以使用「服務」頁面來查看已針對 Application Signals 啟用的服務清單。也可以檢視操作指標，並快速查看哪些服務具有狀態不佳的服務水準指標 (SLI)。在確定操作問題的根本原因時，深入查找效能異常。若要檢視此頁面，請開啟CloudWatch 主控台，然後在左側導覽窗格的「應用程式訊號」區段下選擇「服務」。

探索服務的操作狀態指標

「服務」頁面頂端包含整體服務操作狀態圖表，以及數個顯示故障率最高的服務和服務相依性的資料表。左側的「服務」圖表顯示目前頁面層級時間篩選條件中，具有狀態良好或不良服務水準指標 (SLI) 的服務數目明細。SLI 可監控延遲、可用性和其他操作指標。

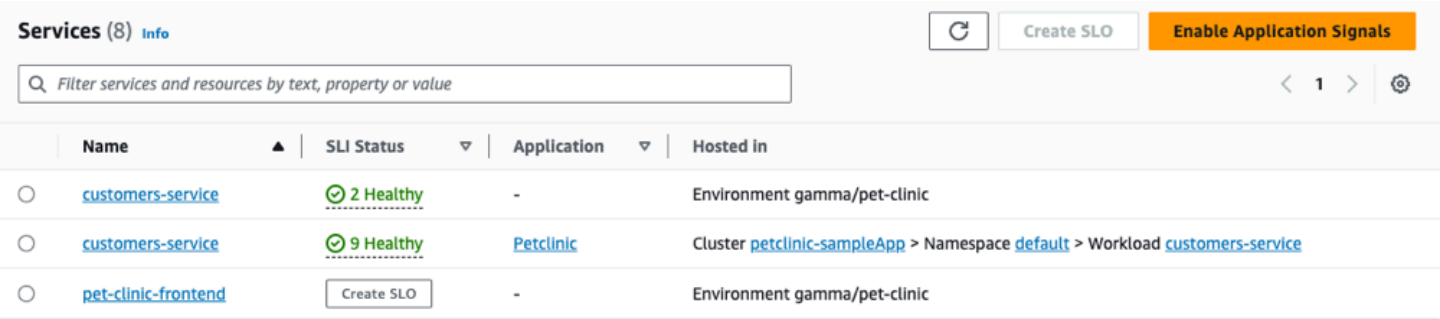
圖表旁邊的兩個資料表會顯示故障率最高的服務清單。選擇任一資料表中的服務名稱，即可開啟服務詳細資訊頁面，並查看詳細的服務操作詳細資訊。選擇相依性路徑以開啟詳細資訊頁面，並查看服務相依性詳細資訊。這兩個表格都會顯示過去最多 3 小時的資訊，即使在頁面右上方選擇了較長的週期篩選條件亦如此。



使用「服務」資料表監控操作運作狀態

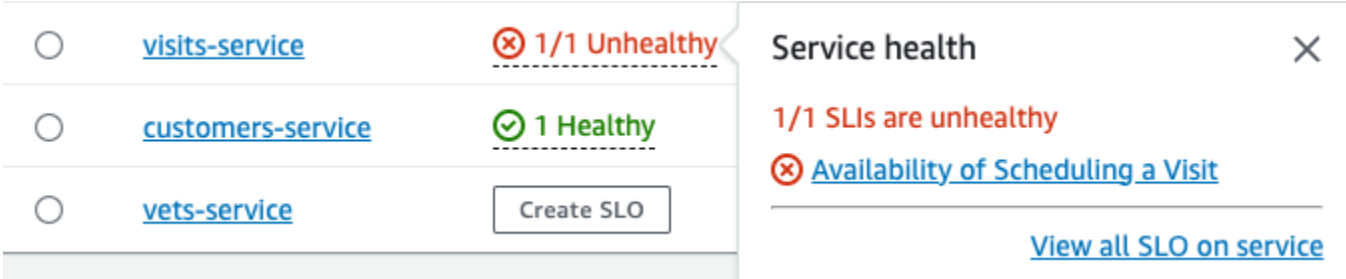
「服務」資料表會顯示已針對 Application Signals 啟用的服務清單。選擇啟用 Application Signals 以開啟設定頁面並開始設定服務。如需詳細資訊，請參閱啟用 Application Signals。

篩選「服務」資料表，可讓您更容易找到要尋找的內容，方法是從篩選文字方塊中選擇一個或多個屬性。當您選擇每個屬性時，系統會引導您完成篩選條件。您將在篩選文字方塊下方看到完整的篩選條件。可隨時選擇清除篩選條件以移除資料表篩選條件。



選擇資料表中任何服務的名稱，即可檢視包含服務水準指標、操作及其他詳細資訊的服務詳細資訊頁面。如果您已將服務的基礎運算資源與中的應用程式 AppRegistry 或 AWS Management Console 首頁上的應用程式卡產生關聯，請選擇應用程式名稱，以在「我的應用程式」主控台頁面中顯示應用程式詳細資訊。對於 Amazon EKS 中託管的服務，請選擇託管於欄中的任何連結，以檢視 CloudWatch 容器洞見中的叢集、命名空間或工作負載。對於在 Amazon ECS 或 Amazon EC2 中執行的服務，會顯示「環境」值。

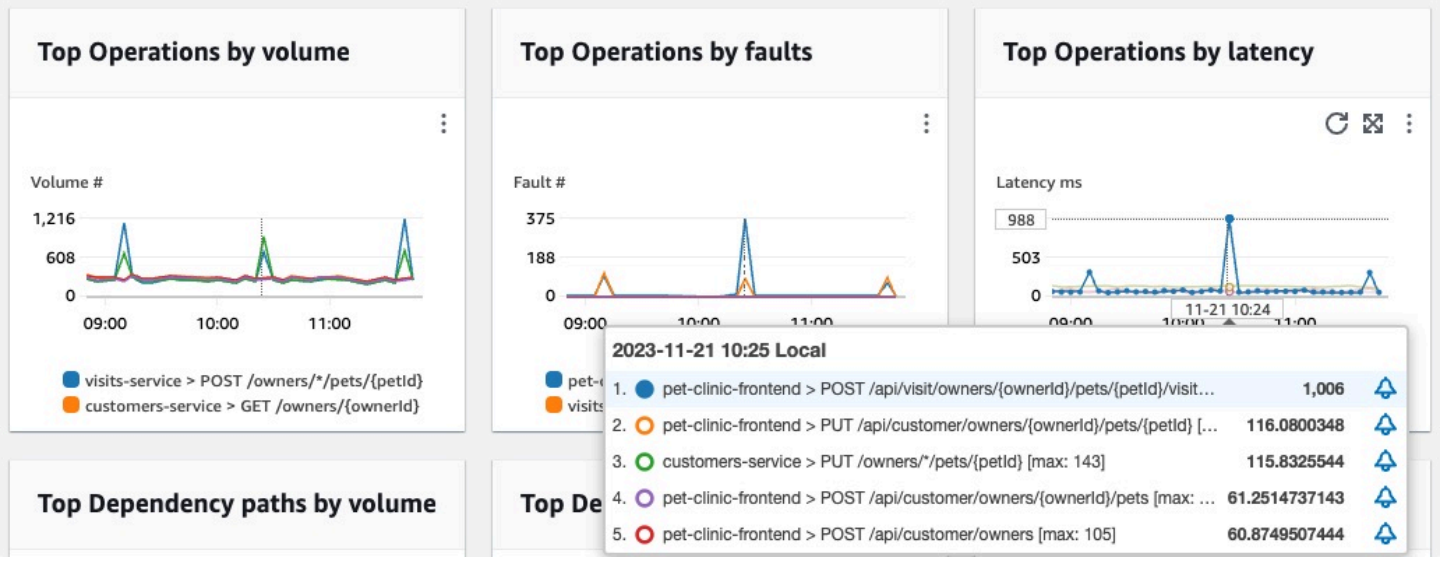
資料表中會顯示每個服務的服務水準指標 (SLI) 狀態。選擇服務的 SLI 狀態以顯示快顯視窗，其中包含任何狀態不良 SLI 的連結，以及可查看所有操作之 SLO 的連結。



如果尚未為服務建立 SLO，請在 SLI 狀態欄中選擇建立 SLO 按鈕。若要為任何服務建立額外的 SLO，請選取服務名稱旁的選項按鈕，然後在資料表右上角選擇建立 SLO。建立 SLO 時，可以一目了然地看到哪些服務和操作執行良好，哪些運作狀況不佳。如需詳細資訊，請參閱服務水準目標 (SLO)。

檢視常用的操作和相依性指標

在「服務」資料表下方，可以檢視所有服務中呼叫量、故障率和延遲最高的操作和相依性。這組圖表提供重要資訊，說明哪些操作或相依性可能在所有服務中運作狀態不佳。選擇圖表中的任何一個點，即可看到包含更詳細系列資訊的快顯視窗。將滑鼠移至圖表底部的系列說明上，即可看到包含特定操作或相依性路徑詳細指標的快顯視窗。選取圖表右上角的內容功能表按鈕，以查看其他選項，包括檢視 CloudWatch 量度或記錄頁面。



使用服務詳細資訊頁面檢視詳細的服務活動和作業狀態

⚠ 應用程式訊號正在適用於 Amazon 的預覽版本中， CloudWatch 且可能會有所變更。

當您檢測應用程式時，Amazon CloudWatch 應用程式信號會對應應用程式發現的所有服務。您可以在「服務詳細資訊」頁面查看單一服務的服務、作業、相依性、Canary 以及從屬端要求的簡介。若要檢視服務詳細資訊頁面，請執行下列動作：

- 開啟 CloudWatch 主控台。
- 在左側導覽窗格的「應用程式訊號」區段下選擇「服務」。
- 從「服務」、「常用服務」或「相依性」表格中選擇任何服務的名稱。

服務詳細資訊頁面會組織成下列索引標籤：

- 概觀 — 使用此標籤可查看單一服務的概觀，包括作業數目、相依性、合成和用戶端頁面。此索引標籤會顯示整個服務、常用作業和相依性的關鍵指標。這些指標包括該服務所有服務作業的延遲、錯誤和錯誤的時間序列資料。
- 服務作業 — 使用此索引標籤可查看服務呼叫的作業清單，以及可測量每項作業健全狀況的關鍵指標的互動式圖形。您可以在圖形中選取資料點，以取得與該資料點相關聯的追蹤、記錄或指標的相關資訊。
- 相依性 — 使用此頁籤可查看服務呼叫的相依性清單，以及這些相依性的測量結果清單。

- Synthetics 金絲雀 — 使用此選項卡可查看模擬用戶對服務調用的合成金絲雀列表，以及有關這些金絲雀的關鍵性能指標。
- 用戶端頁面 — 使用此索引標籤可查看呼叫服務的用戶端頁面清單，以及衡量用戶端與應用程式之間互動品質的度量。

檢視您的服務概述

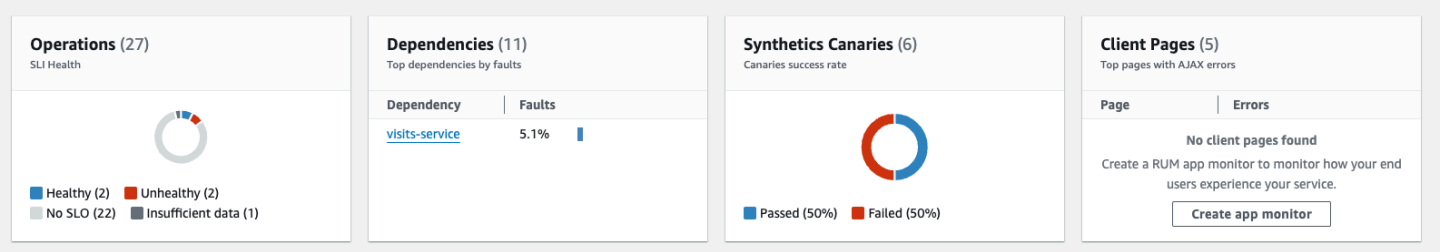
您可以使用「服務簡介」頁面，檢視單一位置中所有服務作業的測量結果高階摘要。檢查與您的應用程序交互的所有操作，依賴關係，客戶端頁面和合成金絲雀的性能。使用此資訊可協助您決定要將精力集中在何處，以識別問題、疑難排解錯誤，以及尋找最佳化的機會。

選擇「服務詳細資料」中的任何連結，即可檢視與特定服務相關的資訊。例如，對於 Amazon EKS 中託管的服務，服務詳細資料頁面會顯示叢集、命名空間和工作負載資訊。對於在 Amazon ECS 或 Amazon EC2 中託管的服務，服務詳細資訊頁面會顯示環境值。

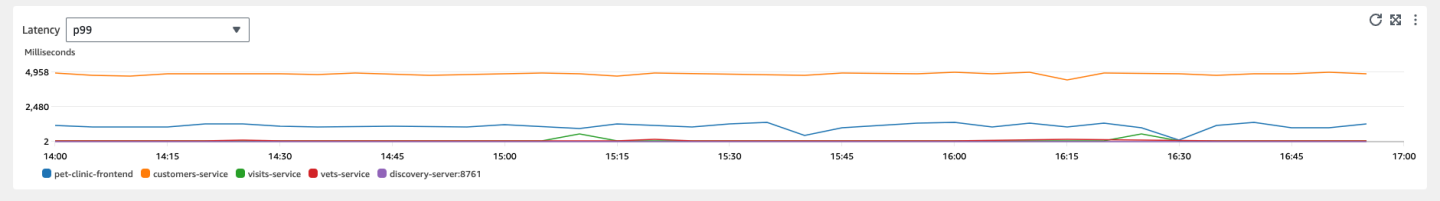
在 [服務] 底下，[概觀] 索引標籤會顯示下列項目的摘要：

- 作業 — 使用此索引標籤可查看服務作業的健全狀況。健全狀況狀態是由定義為服務等級目標 (SLO) 一部份的服務層次指標 (SLI) 決定。
- 相依性 — 使用此標籤可查看應用程式所呼叫之服務的前幾項相依性 (依故障率列出)。
- Synthetics 金絲雀 — 使用此索引標籤可查看對與服務相關聯的端點或 API 進行模擬呼叫的結果，以及失敗的金絲雀數量。
- 用戶端頁面 — 使用此索引標籤可查看發生非同步 JavaScript 和 XML (AJAX) 錯誤的用戶端呼叫的首頁。

下圖顯示您的服務概述：



[概觀] 索引標籤也會顯示所有服務之間延遲最高的相依性圖表。使用 p99、p 90 和 p 50 延遲指標快速評估哪些相依性會導致您的總服務延遲，如下所示：

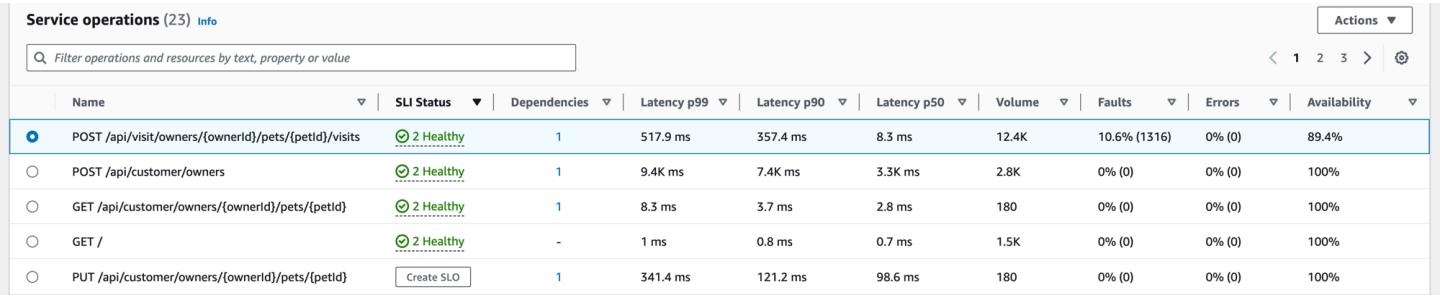


例如，先前的圖表顯示 99% 對客戶服務相依性所做的要求已在大約 4,950 毫秒內完成。其他依賴關係花費的時間更少。

按延遲顯示前四個服務作業的圖形顯示這些服務的要求數量、使用狀態、錯誤率和錯誤率，如下圖所示：

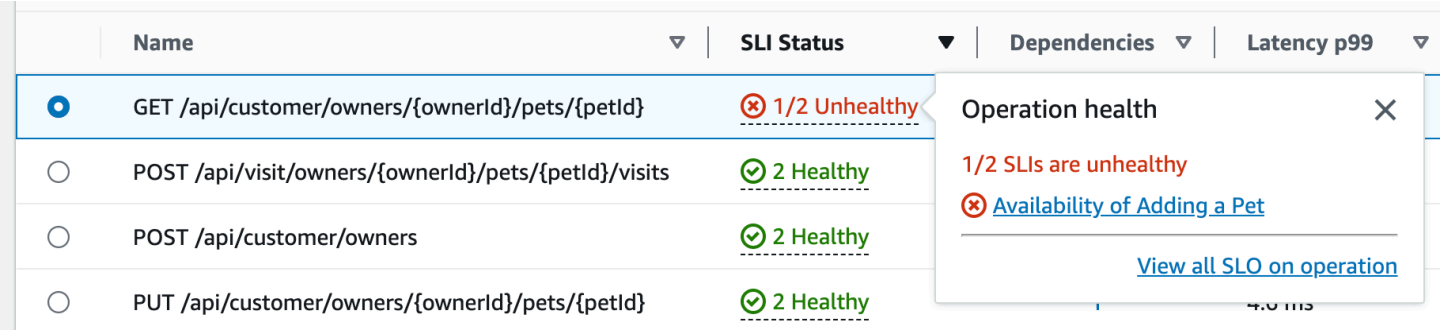
檢視服務操作

當您檢測應用程式時，「應用程式信號」會探索應用程式呼叫的所有服務作業。您可以使用「服務作業」頁籤來查看包含服務作業的表格，以及一組測量所選作業效能的測量結果。這些測量結果包括 SLI 狀態、相依性數目、延遲、磁碟區、錯誤、錯誤和可用性，如下圖所示：



從篩選文字方塊中選擇一或多個屬性，篩選表格以便更輕鬆地尋找服務作業。當您選擇每個屬性時，系統會引導您完成篩選條件，並在篩選文字方塊下方看到完整的篩選條件。可隨時選擇清除篩選條件以移除資料表篩選條件。

選擇作業的 SLI 狀態以顯示快顯視窗，其中包含任何狀況不良的 SLI 連結，以及查看作業所有 SLO 的連結，如下表所示：



服務作業表格會列出 SLI 狀態、狀況良好或狀況不良的 SLI 數目，以及每個作業的 SLO 總數。

使用 SLI 監控延遲、可用性和其他用於測量服務運作狀態的操作指標。使用 SLO 來檢查服務和作業的效能和健全狀況狀態。

若要建立 SLO，請執行下列動作：

- 如果作業沒有 SLO，請在「SLI 狀態」欄中選擇「建立 SL O」按鈕。
- 如果作業已有 SLO，請執行下列動作：
  - 選取作業名稱旁邊的選項按鈕。
  - 從表格右上角的「動作」 向下箭頭選擇「建立 SLO」。

如需詳細資訊，請參閱服務水準目標 (SLO)。

相依性欄會顯示此操作所呼叫的相依性數目。選擇此數字可開啟已根據所選操作篩選的相依性索引標籤。

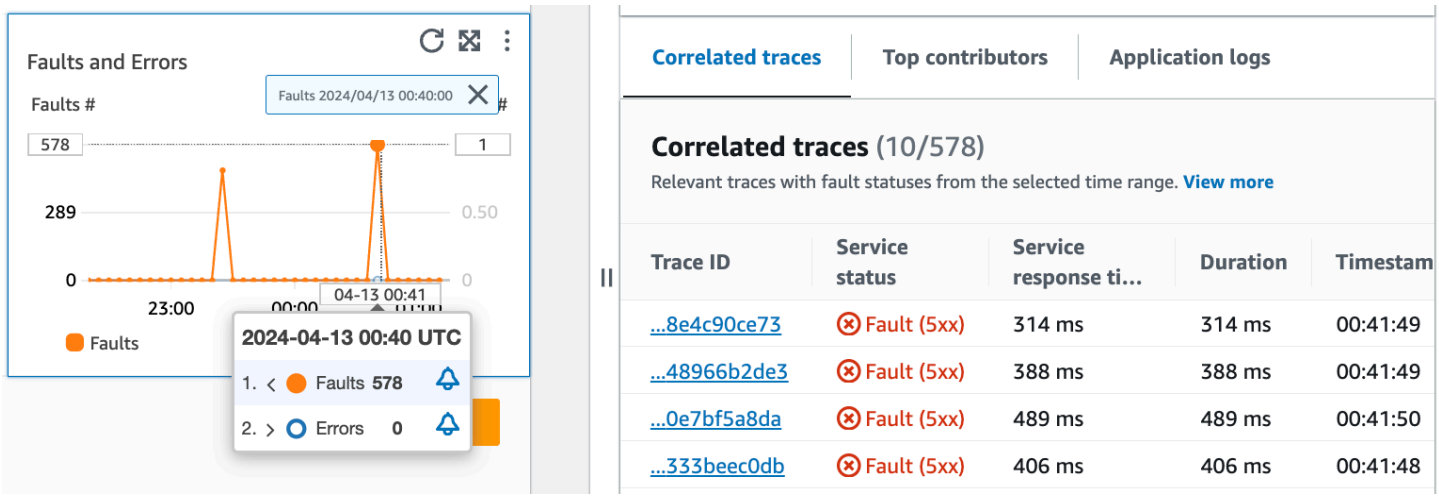
檢視服務作業指標、相關追蹤和應用程式記錄

應用程式訊號會將服務作業指標與 AWS X-Ray 追蹤、 CloudWatch容器深入解析和應用程式記錄相關聯。使用這些指標疑難排解作業健康狀態問題。若要以圖形資訊的形式檢視量度，請執行下列動作：

1. 在「服務作業」表格中選取服務作業，即可在表格上方查看所選作業的一組圖形，其中包含「磁碟區和使用狀態」、「延遲」以及「錯誤和錯誤」的測量結果。

2. 將游標暫留在圖表中的某個點上可檢視更多資訊。
3. 選取點以開啟診斷窗格，其中顯示圖形中所選點的相關追蹤、測量結果和應用程式記錄。

下圖展示了將游標懸停在圖形中某個點上後出現的工具提示，以及按一下點後顯示的診斷窗格。工具提示包含「錯誤與錯誤」圖表中關聯資料點的相關資訊。窗格包含與所選點相關聯的相關追蹤、主要貢獻者和應用程式記錄檔。



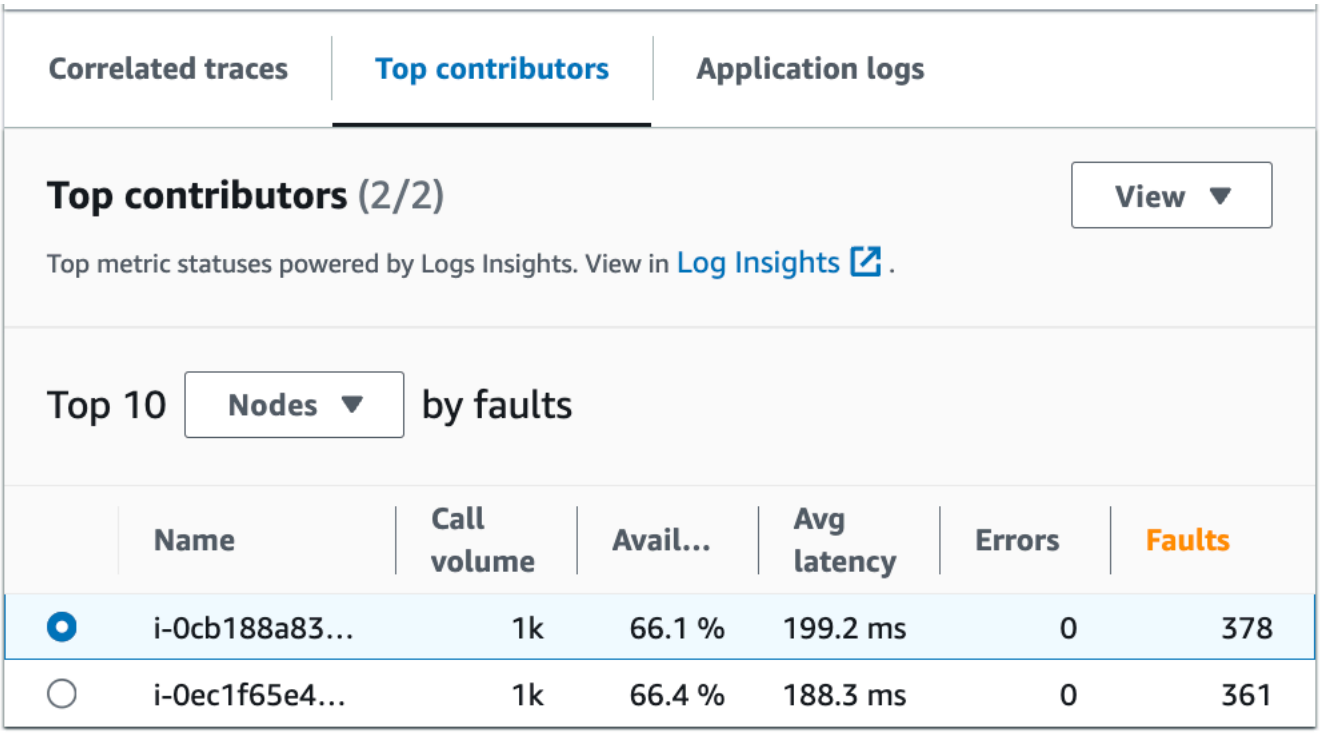
相關痕跡

查看相關跟踪以了解帶有跟踪的基本問題。您可以檢查相關聯的跟踪或與它們關聯的任何服務節點的行為是否類似。若要檢查相關的追蹤，請從「相關的追蹤」表格中選擇追蹤 ID，以開啟所選追蹤的「X-Ray」追蹤詳細資訊頁面。追蹤詳細資訊頁面包含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。

頂尖貢獻者

檢視頂尖的貢獻者，以尋找量度的主要輸入來源。依不同元件將貢獻者分組，以尋找群組內的相似性，並瞭解它們之間的追蹤行為有何不同。

[排名最高的貢獻者] 索引標籤會提供每個群組的通話量、可用性、平均延遲、錯誤和故障等指標。下列範例影像顯示 Amazon EKS 平台上部署之應用程式指標套件的主要貢獻者：



頂級貢獻者包含以下指標：

- 通話磁碟區-使用通話磁碟區來瞭解群組每個時間間隔的要求數目。
- 可用性-使用可用性來查看群組未偵測到任何錯誤的時間百分比。
- 平均延遲-使用延遲來檢查在一段時間間隔內為群組執行要求的平均時間，此時間間隔取決於您正在調查的要求是多久之前而定。在 15 天之前提出的要求會以 1 分鐘的間隔進行評估。在 15 到 30 天之前提出的要求 (含) 會以 5 分鐘的間隔進行評估。例如，如果您正在調查 15 天前造成錯誤的要求，則呼叫量度等於每 5 分鐘間隔的要求數目。
- 錯誤-在一段時間間隔內測量的每個群組的錯誤數。
- 錯誤-每個群組在一段時間間隔內的錯誤數目。

頂級貢獻者使用 Amazon EKS 或 Kubernetes

使用 Amazon EKS 上部署應用程式的主Kubernetes要貢獻者相關資訊，或查看依節點、Pod和PodTemplateHash分組的操作健康狀態指標。下列定義適用：

- 網繭是一或多個共用儲存區和資源的Docker容器所組成的群組。網繭是可在Kubernetes平台上部署的最小單位。依網繭分組，以檢查錯誤是否與網繭特定限制相關。
- 節點是執行 Pod 的伺服器。按節點分組以檢查錯誤是否與節點特定限制有關。
- 網繭範本雜湊可用來尋找部署的特定版本。依網繭範本雜湊分組，以檢查錯誤是否與特定部署相關。

頂級貢獻者使用 Amazon EC2

使用 Amazon EKS 上部署應用程式的主要貢獻者相關資訊，查看依執行個體 ID 和 auto 擴展群組分組的操作狀態指標。下列定義適用：

- 執行個體 ID 是您服務執行之 Amazon EC2 執行個體的唯一識別碼。依執行個體 ID 分組，以檢查錯誤是否與特定 Amazon EC2 執行個體相關。
- auto 擴展群組是 Amazon EC2 執行個體的集合，可讓您擴展或縮減處理應用程式請求所需的資源。如果您想要檢查群組內的執行個體範圍是否有錯誤限制，請依照 auto 調整資源調整群組進行分組。

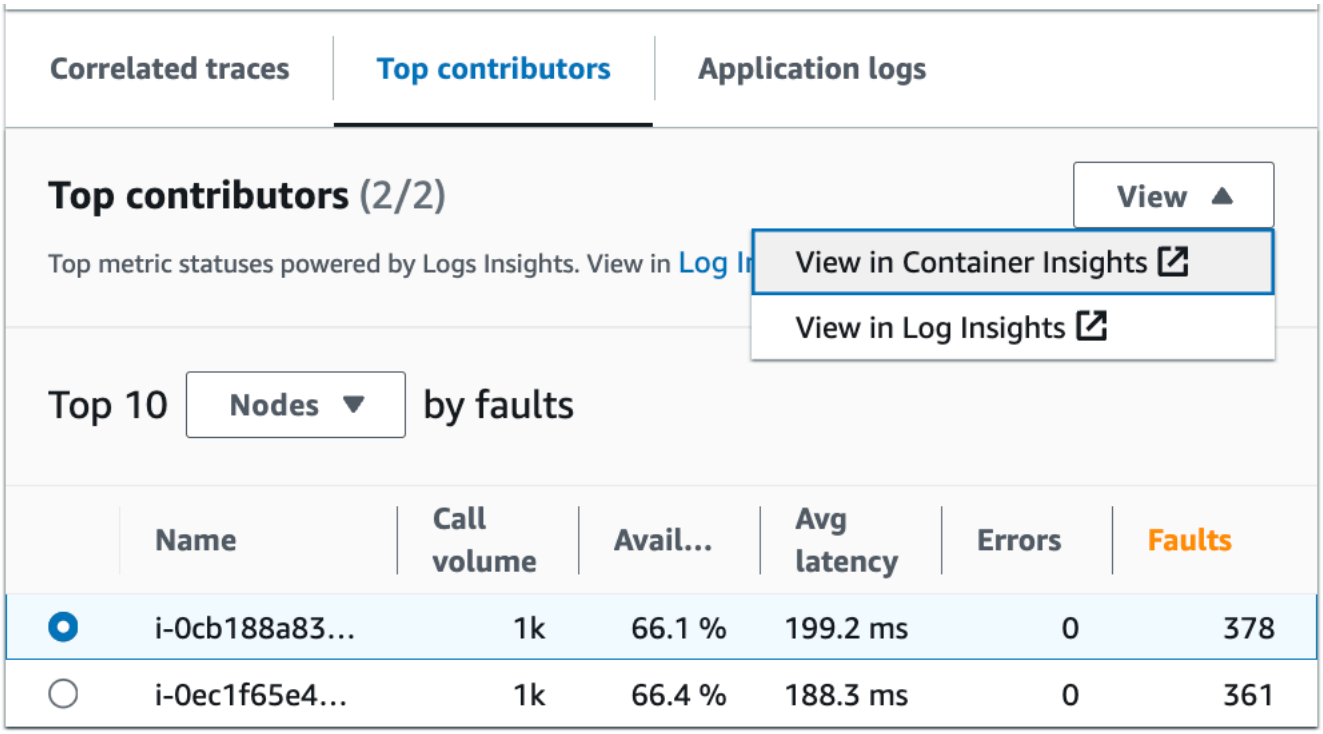
使用自訂平台的頂尖貢獻者

使用有關使用自訂檢測部署之應用程式的主要貢獻者資訊，查看依主機名稱分組的作業健全狀況指標。下列定義適用：

- 主機名稱可識別連線到網路的裝置，例如端點或 Amazon EC2 執行個體。按主機名稱分組，以檢查您的錯誤是否與特定的物理或虛擬設備有關。

檢視Log Insights和中的頂尖貢獻者 Container Insights

在 Log Insights 中檢視和修改為頂尖貢獻者產生指標的自動查詢。依特定群組 (例如容器深入解析中的網繭或節點) 檢視基礎結構效能指標。您可以按資源消耗對叢集、節點或工作負載進行排序，並在最終使用者體驗受到影響之前快速識別異常情況或主動降低風險。顯示如何選取這些選項的影像如下：



在容器深入解析中，您可以檢視 Amazon EKS 或 Amazon ECS 容器的指標，這些指標是專門針對頂尖貢獻者的群組。例如，如果您針對 EKS 容器依網繭分組以產生最高貢獻者，容器深入解析將會顯示針對網繭篩選的度量和統計資料。

在 Log Insights 中，您可以使用下列步驟修改「頂尖貢獻者」下產生量度的查詢：

1. 在日誌深入解析中選取檢視。開啟的「記錄見解」頁面包含自動產生的查詢，其中包含下列資訊：

   - 記錄叢集群組名稱。
   - 你正在調查的行 CloudWatch動
   - 與圖表上互動的作業健康度量彙總。

   系統會自動篩選記錄結果，以顯示您在服務圖表上選取資料點前五分鐘的資料。
2. 若要編輯查詢，請以您的變更取代產生的文字。您也可以使用「查詢產生器」來協助您產生新查詢或更新既有查詢。

應用程式記錄

使用 [應用程式記錄] 索引標籤中的查詢來產生目前記錄群組、服務的記錄資訊，並插入時間戳記。記錄群組是一組日誌串流，您可以在設定應用程式時定義這些串流。

使用記錄群組來組織具有類似特性的記錄檔，包括下列各項：

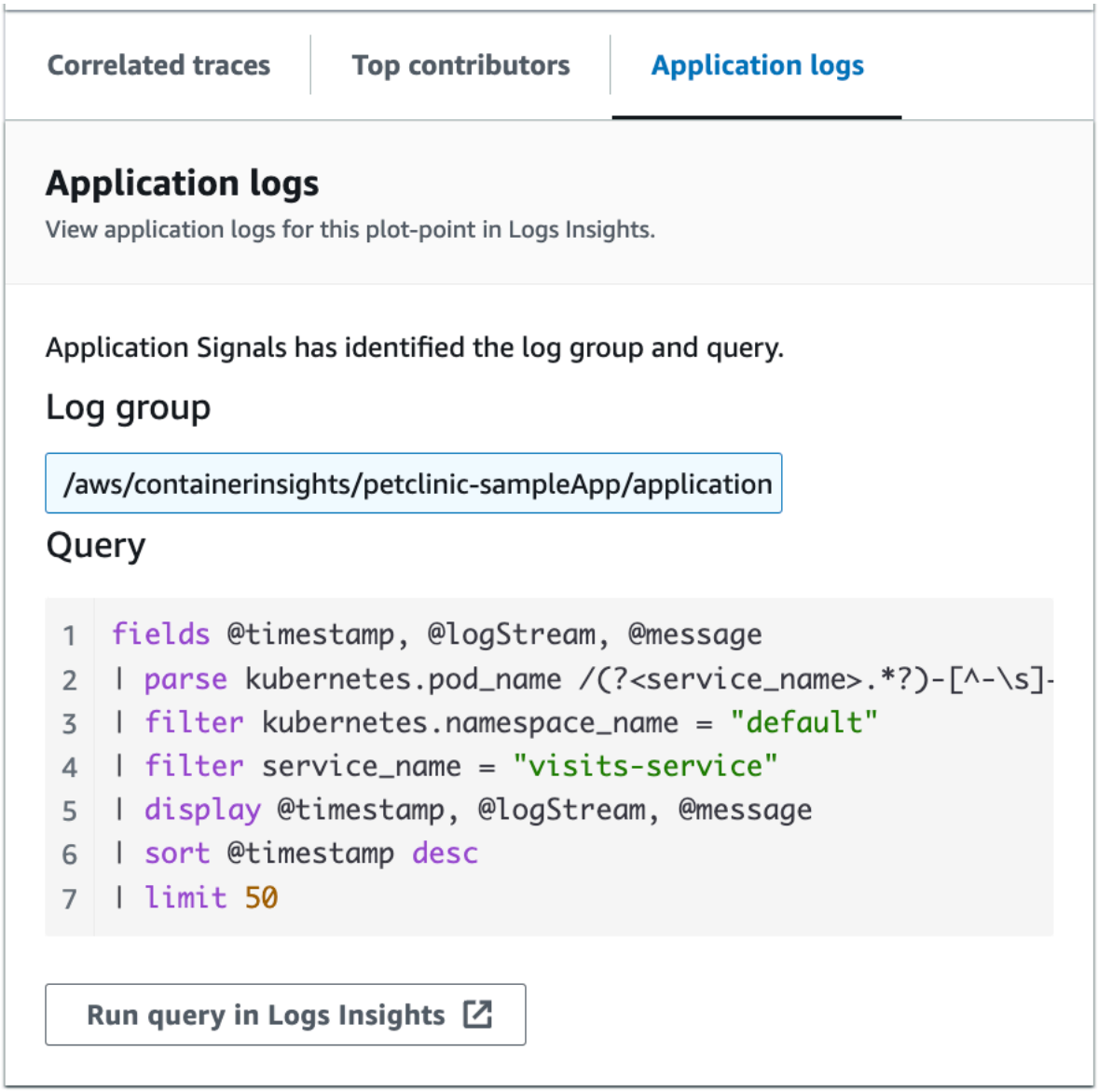
- 從特定組織、來源或功能擷取記錄。
- 擷取特定使用者存取的日誌。
- 擷取特定時段的記錄檔。

您可以使用這些記錄串流來追蹤特定群組或時間範圍。您也可以設定這些記錄群組的監視規則、警示和通知。如需有關記錄群組的詳細資訊，請參閱使用記錄群組和記錄資料流。

應用程式記錄查詢會傳回記錄檔、週期性文字模式和日誌群組的圖形視覺效果。

若要執行查詢，請選取 [在記錄檔見解中執行查詢]，以執行自動產生的查詢或修改查詢。若要編輯查詢，請以您的變更取代自動產生的文字。您也可以使用「查詢產生器」來協助您產生新查詢或更新既有查詢。

下圖顯示了根據服務操作圖中選取的點自動產生的查詢範例：



在前面的影像中， CloudWatch 已自動偵測到與所選點相關聯的記錄群組，並將其包含在產生的查詢中。

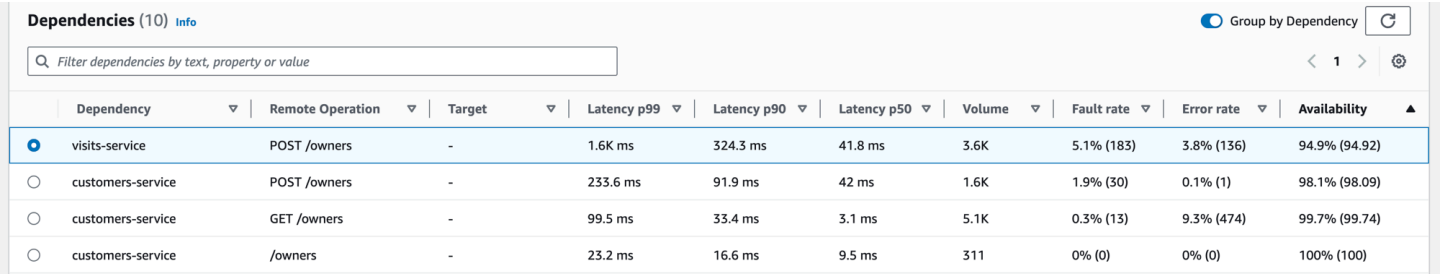
檢視服務相依性

選擇相依性索引標籤，即可顯示相依性資料表，以及所有服務操作或單個操作之相依性的一組指標。此資料表包含 Application Signals 發現的相依性清單，包括延遲、呼叫量、故障率、錯誤率和可用性的指標。

在頁面頂端，從向下箭號清單選擇作業以檢視其相依性，或選擇「全部」以查看所有作業的相依性。

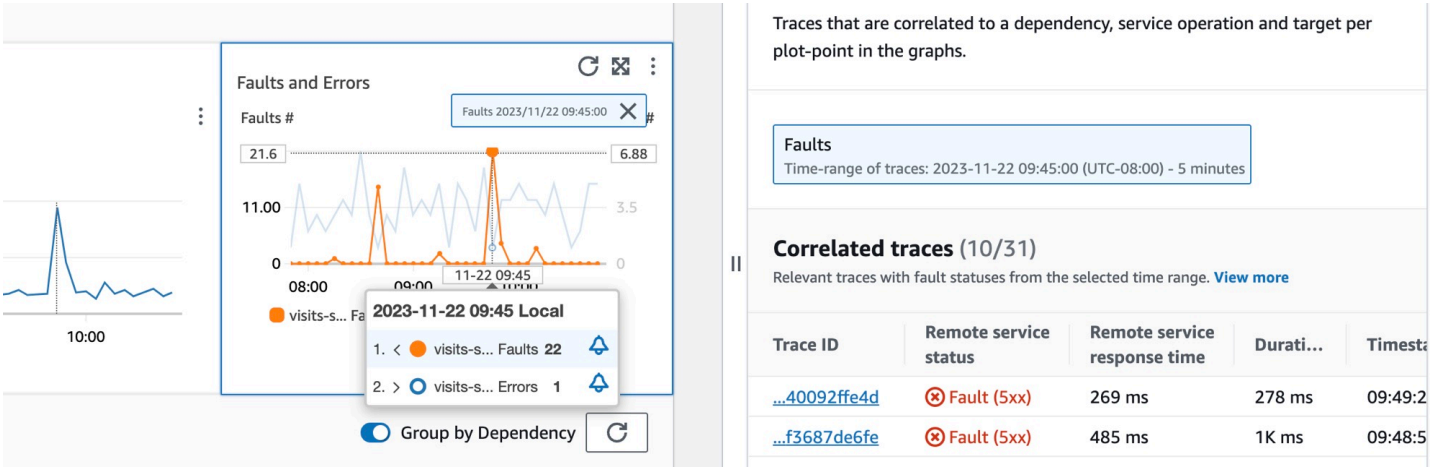
篩選資料表，可讓您更容易找到要尋找的內容，方法是從篩選文字方塊中選擇一個或多個屬性。當您選擇每個屬性時，系統會引導您完成篩選條件，並在篩選文字方塊下方看到完整的篩選條件。可隨時選

擇清除篩選條件以移除資料表篩選條件。選取資料表右上角的按相依性分組，可按服務和操作名稱對相依性分組。開啟分組時，使用相依性名稱旁邊的 + 圖示來展開或摺疊相依性群組。



相依性資料欄會顯示相依性服務名稱，而遠端操作資料欄則顯示服務操作名稱。呼叫 AWS 服務時，「目標」欄會顯示 AWS 資源，例如 DynamoDB 表格或 Amazon SNS 佇列。

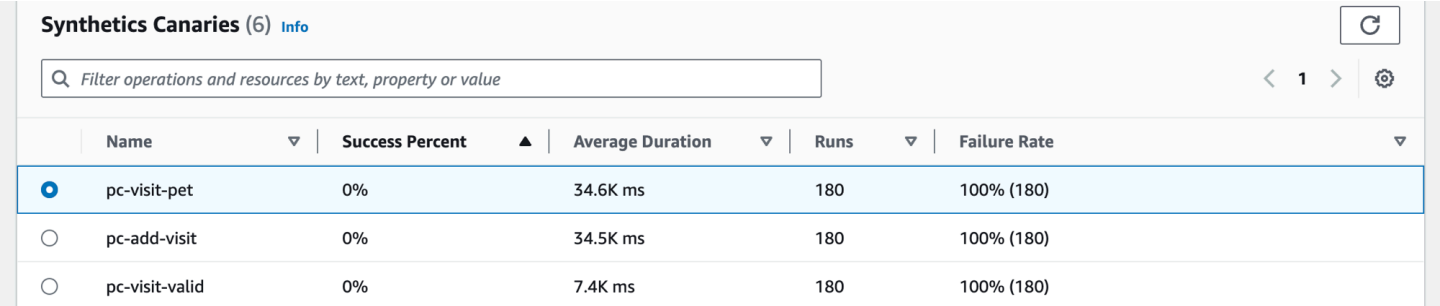
若要選取相依性，請選取相依性資料表中某個相依性旁邊的選項。這會顯示一組圖形，其中顯示通話量、使用狀態、錯誤和錯誤的詳細測量結果。將游標暫留在圖表中的某個點上，即可看到包含更多資訊的快顯視窗。在圖形中選取一個點以開啟診斷窗格，該窗格顯示圖形中所選點的相關繪線。從「相關聯」追蹤表格中選擇追蹤 ID，以開啟所選追蹤的「X-Ray 追蹤詳細資訊」頁面。



檢視 Synthetics Canaries

選擇 Synthetics Canaries 索引標籤以顯示 Synthetics Canaries 資料表，以及資料表中每個 Canary 的一組指標。此表格包含成功百分比、平均持續時間、執行次數和失敗率的指標。只會顯示已啟用 AWS X-Ray 追蹤的金絲雀。

使用合成金絲雀表格中的篩選文字方塊，尋找您感興趣的金絲雀。您建立的每個篩選器都會顯示在篩選文字方塊下方。可隨時選擇清除篩選條件以移除資料表篩選條件。



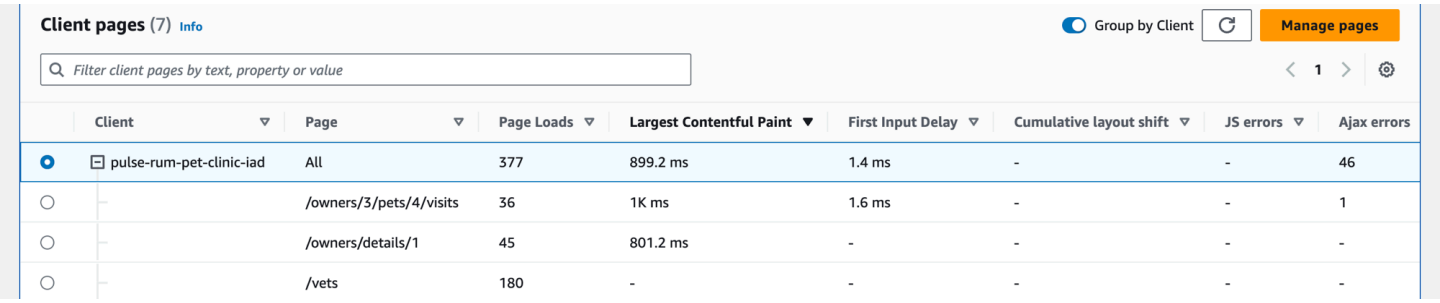
選取初期測試名稱旁邊的圓鈕，即可查看一組標籤，其中包含詳細量度的圖表，包括成功百分比、錯誤和持續時間。將游標暫留在圖表中的某個點上，即可看到包含更多資訊的快顯視窗。在圖形中選取一個點以開啟診斷窗格，該窗格顯示與所選點相關的初期測試執行。選取初期測試執行，然後選擇 [執行時間] 以查看所選初期測試執行的成品，包括記錄HTTP檔、封存 (HAR) 檔案、螢幕擷取畫面和建議步驟，以協助您疑難排解問題。選擇更多拉恩以打開金絲雀跑步旁邊的 CloudWatch Synthetics 金絲雀頁面。

檢視您的用戶端頁面

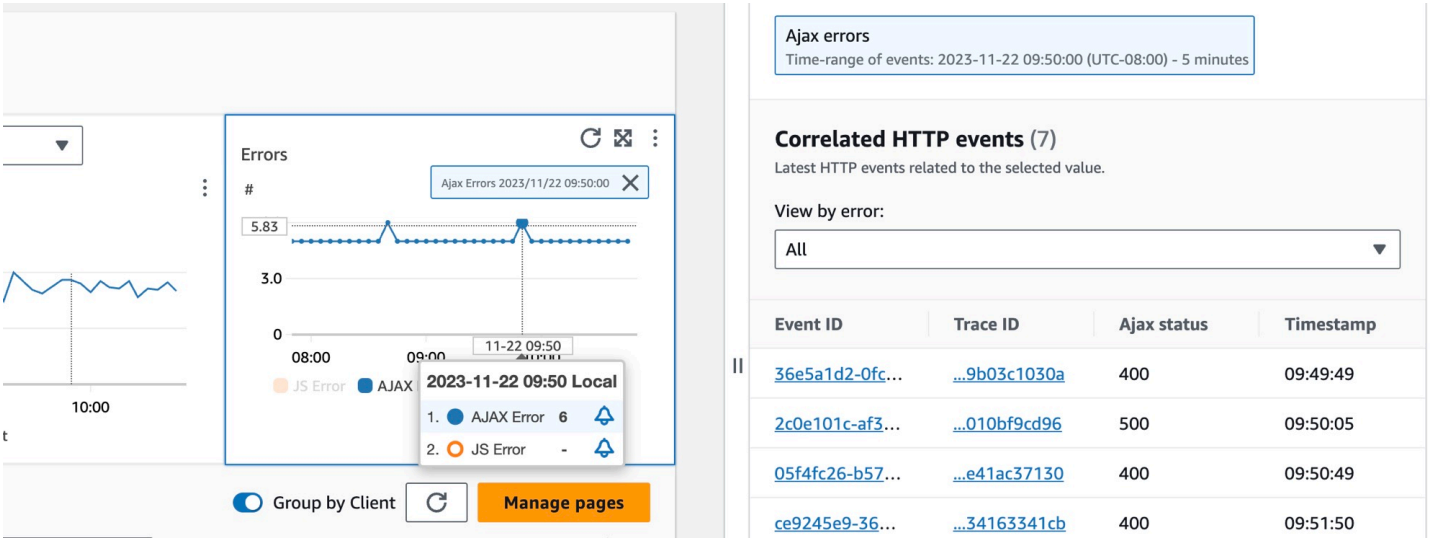
選擇「客戶端頁面」標簽以顯示調用服務的客戶端網頁列表。使用所選用戶端頁面的一組指標來衡量客戶與服務或應用程式互動時的體驗品質。這些量度包括頁面載入、Web 重要資料和錯誤。

若要在表格中顯示用戶端頁面，您必須設定 R CloudWatch UM Web 用戶端的 X-Ray 追蹤，並為用戶端頁面開啟「應用程式訊號」度量。選擇「管理頁面」以選取啟用「應用程式訊號」測量結果的頁面。

使用篩選器文字方塊，在篩選器文字方塊下方尋找您感興趣的用戶端頁面或應用程式監視器。選擇 [清除篩選器] 以移除表格篩選器。選取按用戶端分組，可按用戶端對用戶端頁面進行分組。分組後，選擇用戶端名稱旁邊的 + 圖示以展開該列，並查看該用戶端的所有頁面。



若要選取用戶端頁面，請在用戶端頁面資料表中選取用戶端頁面旁邊的選項。您將看到一組顯示詳細指標的圖表。將游標暫留在圖表中的某個點上，即可看到包含更多資訊的快顯視窗。在圖形中選取一個點以開啟診斷窗格，其中顯示圖形中所選點的相關效能導覽事件。從導覽事件清單中選擇事件 ID，以開啟所選事件的 CloudWatch RUM 頁面檢視。



> **Note**
>
> 若要查看用戶端頁面中的 AJAX 錯誤，請使用 CloudWatch RUM 網頁用戶端版本 1.15 或更新版本。
>
> 目前，每個服務最多可顯示 100 個操作、canary 和用戶端頁面，以及最多 250 個相依性。

使用 CloudWatch 服務對應檢視您的應用程式拓撲並監控作業健康狀態

> ⚠ 應用程式訊號正在適用於 Amazon 的預覽版本中， CloudWatch 且可能會有所變更。

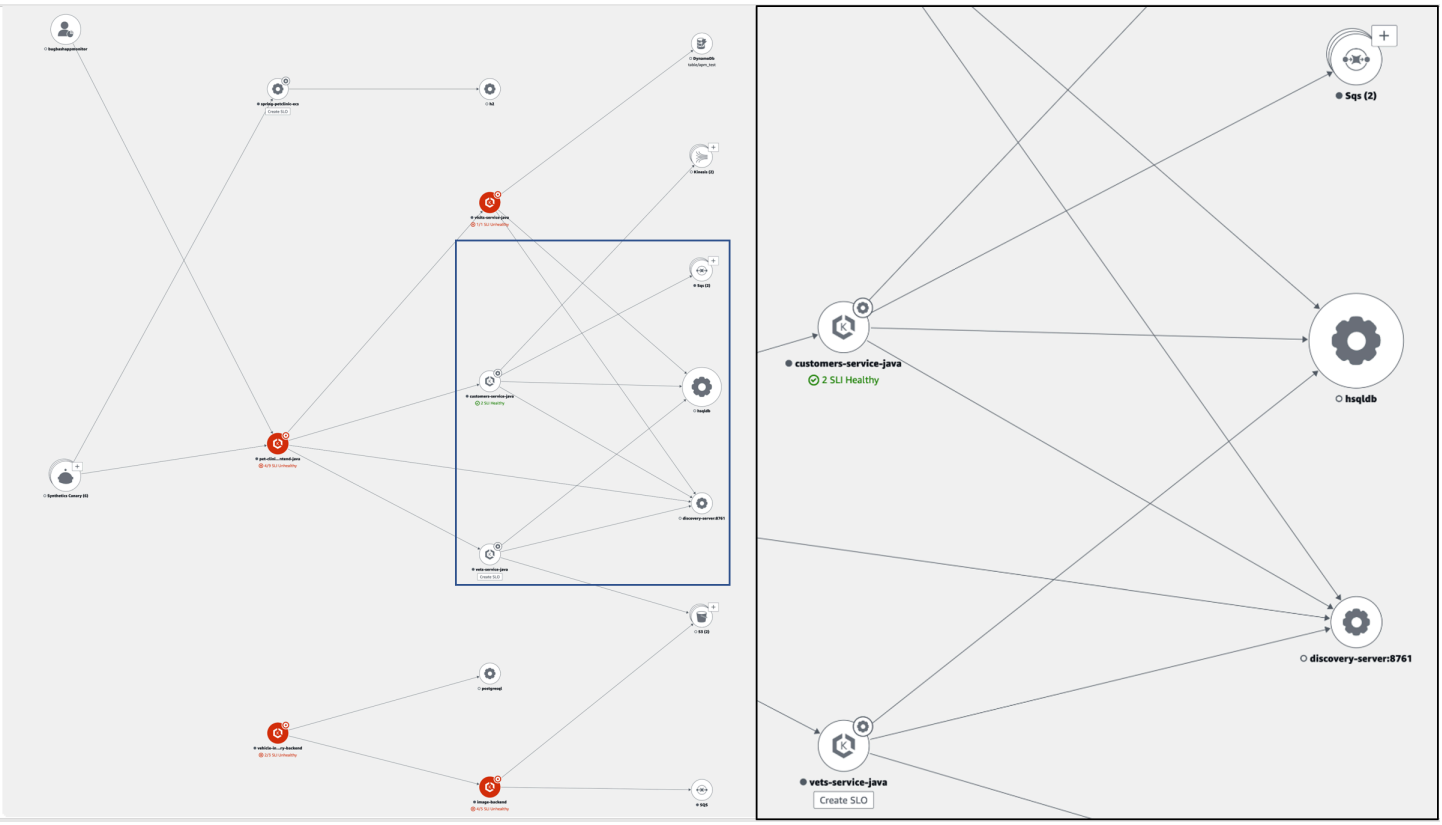
> **Note**
>
> 服 CloudWatch 務對應會取代 ServiceLens 對應。若要根據 AWS X-Ray 軌跡查看應用程式的地圖，請開啟 X-Ray 追蹤地圖。在 CloudWatch 主控台左側導覽窗格的 [X-Ray] 區段下選擇 [追蹤對應]。

使用服務對應來檢視應用程式用戶端的拓撲、合成金絲雀、服務和相依性，以及監視作業健康狀態。若要檢視服務對應，請開啟CloudWatch 主控台，然後在左側導覽窗格的「應用程式訊號」區段下選擇「服務對應」。

在您啟用應用程式的應用程式訊號之後，請使用服務對應來更輕鬆地監控應用程式的運作狀態：

- 檢視用戶端、canary、服務和相依性節點之間的連線，以協助您了解應用程式拓撲和執行流程。如果您的服務營運商不是您的開發團隊，這將特別有用。
- 查看哪些服務符合或不符合您的服務水準目標 (SLO)。當服務不符合您的 SLO 時，您可以快速識別下游服務或相依性是否可能導致問題或影響多個上游服務。
- 選取個別用戶端、合成初期測試、服務或相依性節點，以查看相關指標。[服務詳細資料] 頁面會顯示有關作業、相依性、合成金絲雀和用戶端頁面的更多詳細資訊。
- 篩選和縮放服務對應，以便更輕鬆地專注於應用程式拓撲的一部分，或查看整個地圖。從篩選文字方塊中選擇一個或多個屬性來建立篩選條件。當您選擇每個屬性時，系統會引導您完成篩選條件。您將在篩選文字方塊下方看到完整的篩選條件。可隨時選擇清除篩選條件以移除篩選條件。

下列範例服務對應會顯示邊緣將這些服務連接到它們互動的元件。如果已定義 SLO，服務對應也會顯示健全狀況狀態。

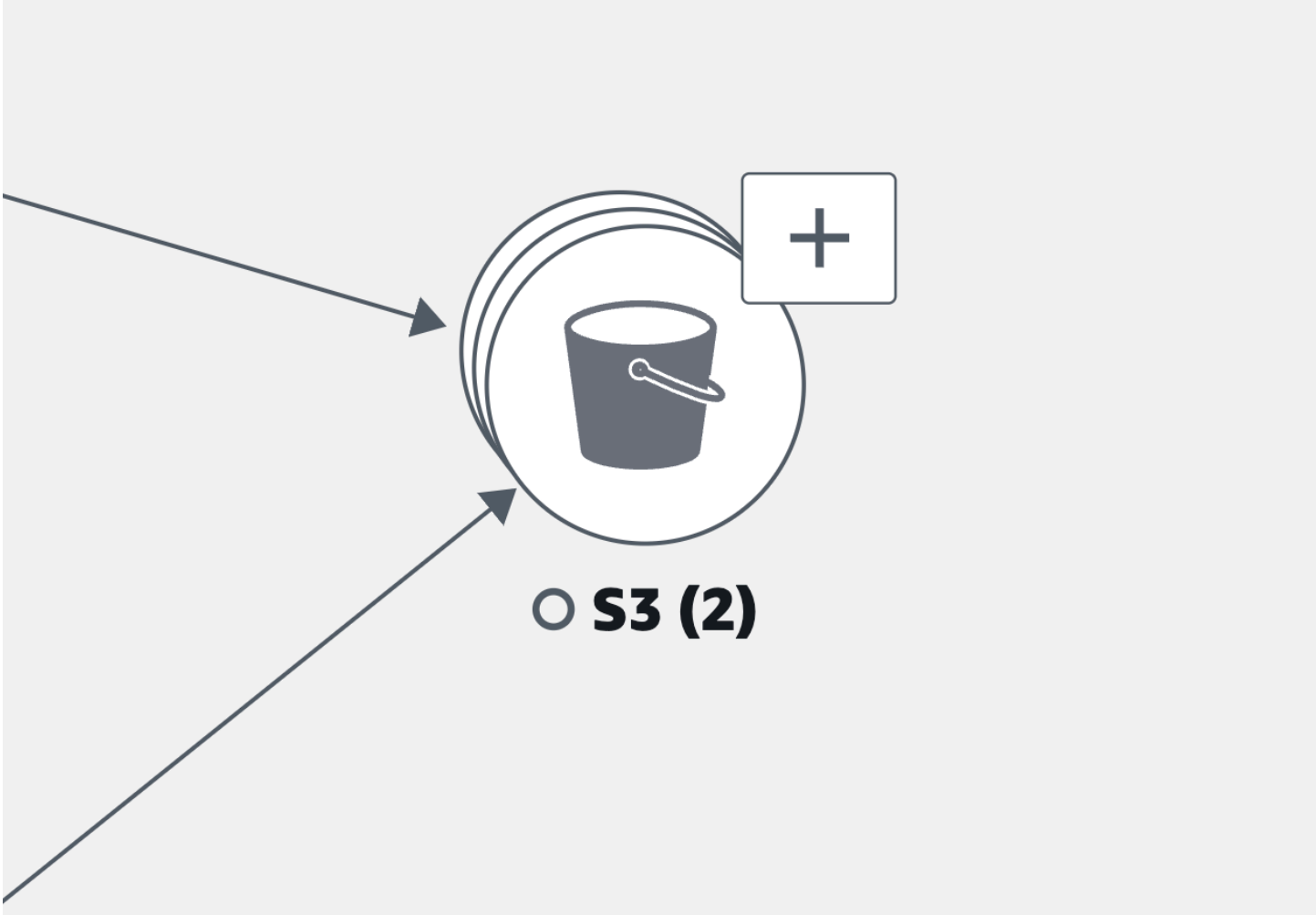


探索服務地圖

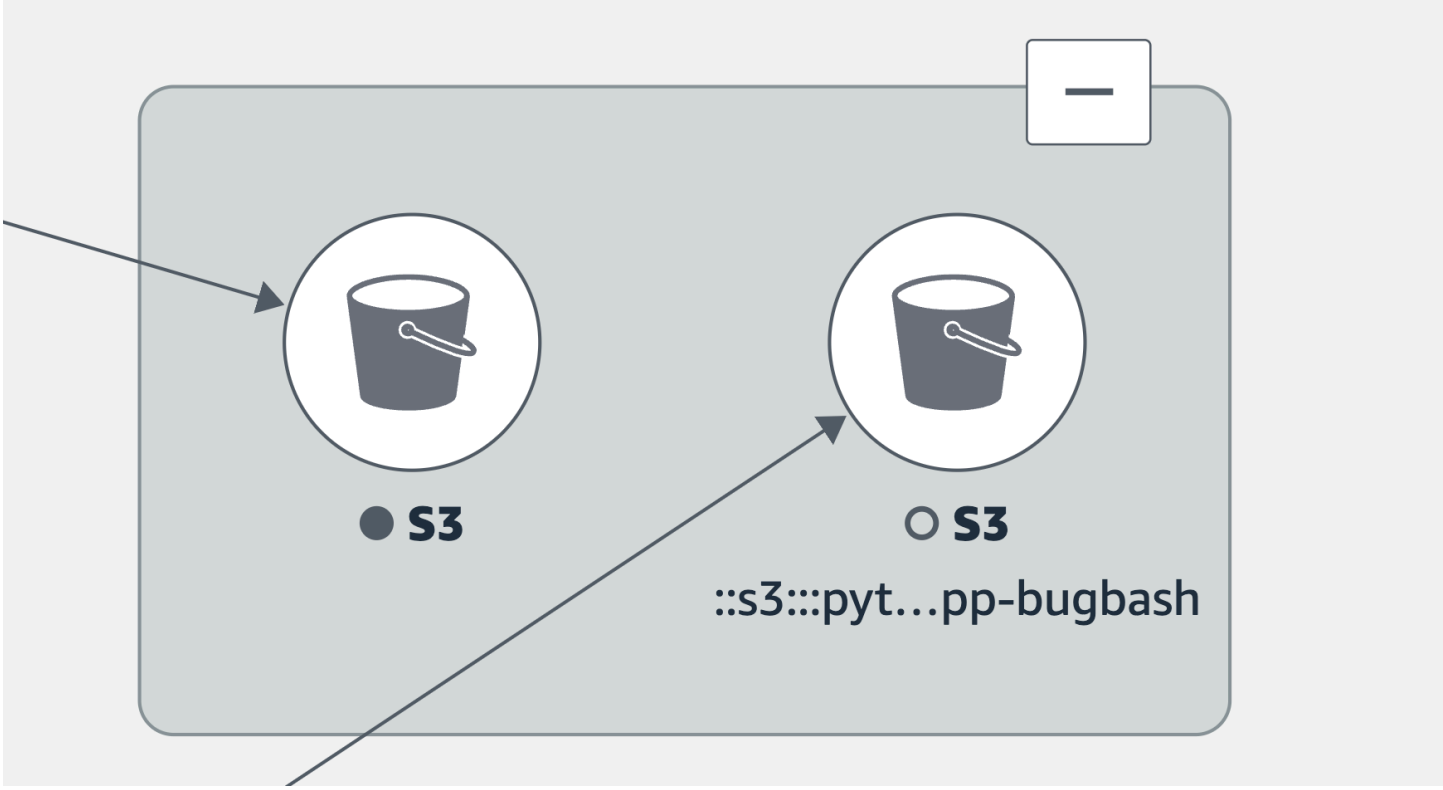
啟用應用程式的應用程式訊號之後，服務對應會顯示代表您的服務及其相依性的節點。

為您的 CloudWatch RUM 用戶端和合成金絲雀開啟主動追蹤，以在地圖上查看用戶端和金絲雀節點。

根據預設，Canary、RUM 用戶端和相同類型的 AWS 服務相依性會分組成服務對應中的單一可擴充圖示。依預設，不會將之外 AWS 的服務相依性群組在一起。例如，在下圖中，所有 Amazon S3 儲存貯體都分組在一個可擴充圖示下：



在上一張圖片中，Amazon S3 分組和原始服務之間的標籤會在相依性圖示下的括號中顯示群組的邊緣數目。選取 (+) 圖示以展開群組並查看其個別元素，如下圖所示：

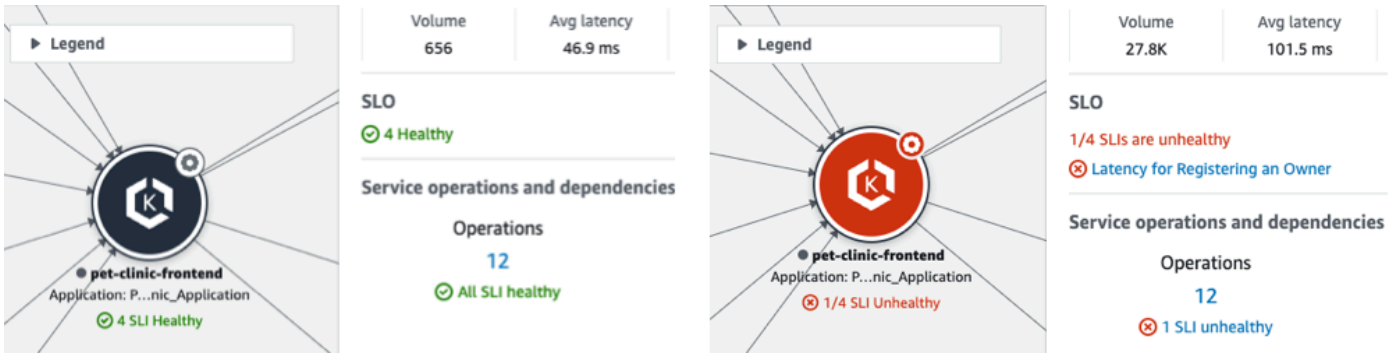


選擇索引標籤，以取得有關探索每種節點種類及其間邊緣 (連接) 的資訊。

View your application services

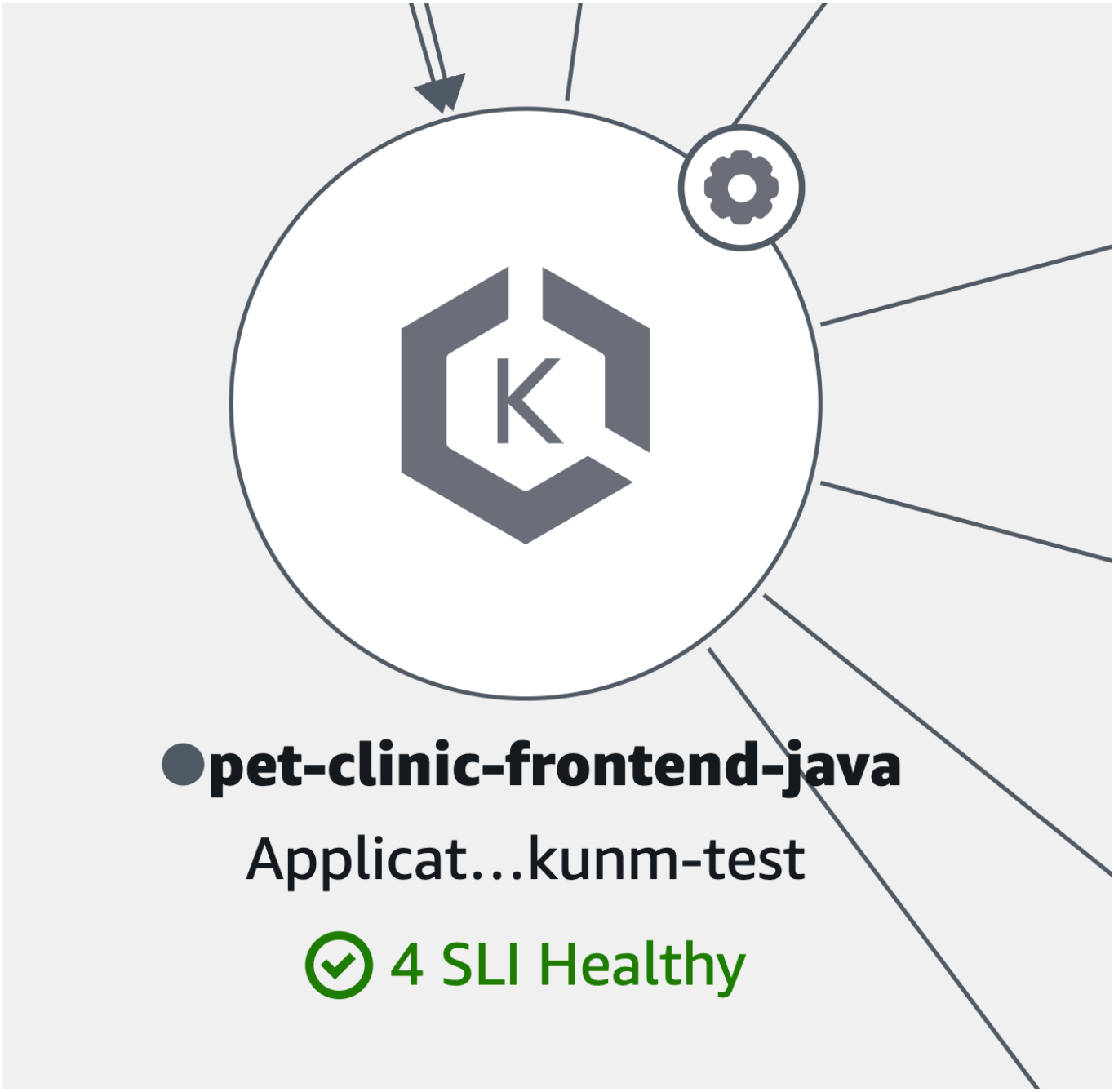
您可以在「服務對映」中檢視應用程式服務及其 SLO 和服務層級指示器 (SLI) 的狀態。如果您沒有為服務建立 SLO，請選擇服務節點下方的 [建立 SLO] 按鈕。

「服務對應」會顯示您所有的服務。它也會顯示使用服務的客戶和金絲雀，以及您的服務呼叫的相依性，如下圖所示：



下列圖示代表服務對應中應用程式服務的範例：

- Amazon Elastic Kubernetes Service:



- 一個庫伯尼特人容器：

- Amazon Elastic Compute Cloud (Amazon EC2)：

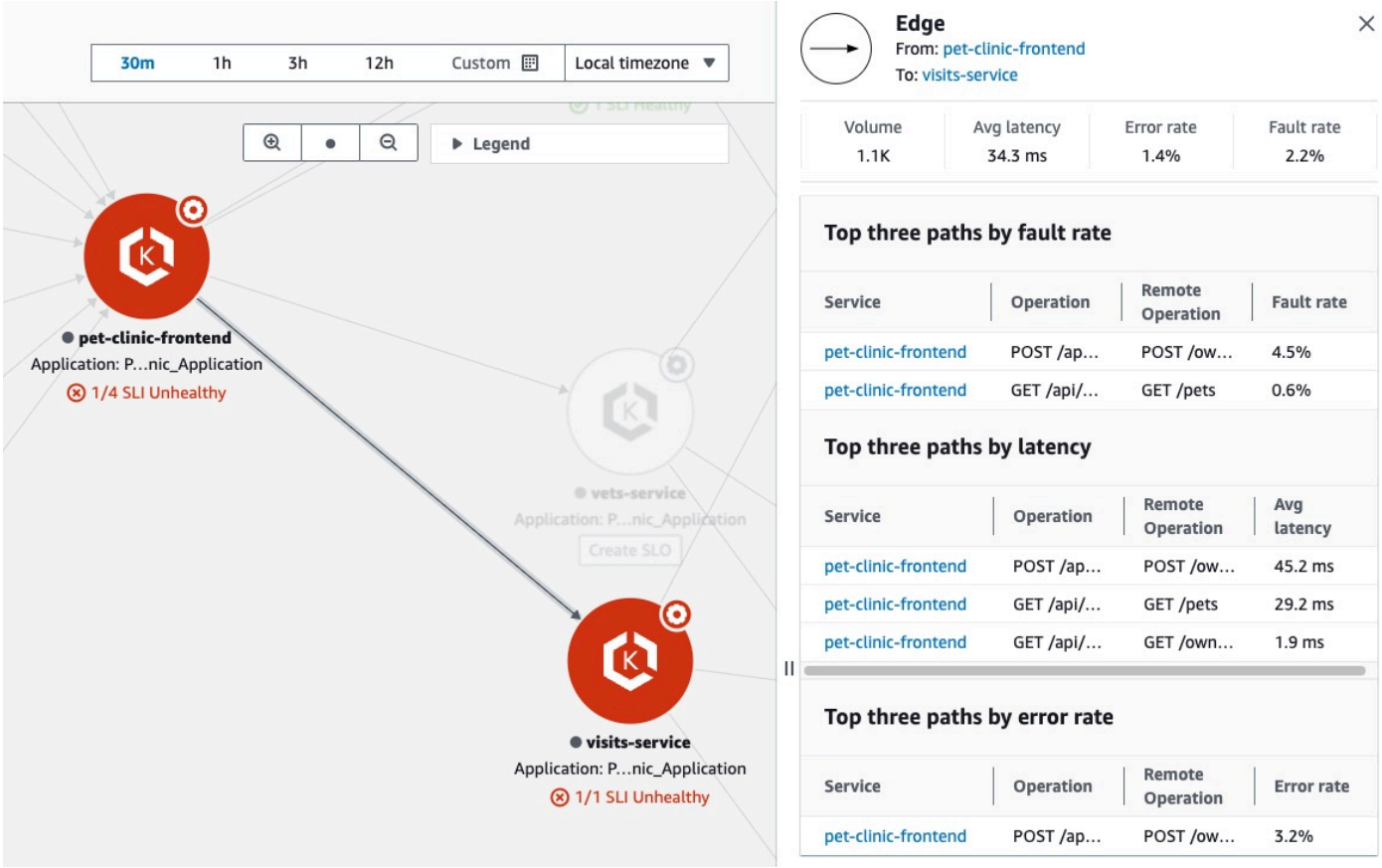
- 先前未列出的其他應用程式服務類型：

當您選取服務節點時，會開啟一個窗格，顯示詳細的服務資訊：

- 呼叫量、延遲、錯誤和故障率的指標。
- 這是或的 SLI 和 SLO 的數目`healthy`。`unhealthy`
- 檢視 SLO 相關詳細資訊的選項。
- 服務操作，依賴關係，合成金絲雀和客戶端頁面的數量。
- 選取每個號碼以開啟其 [服務詳細資訊] 頁面的選項。
- 應用程式名稱 (如果您已使用 AppRegistry 或 AWS Management Console 首頁上的應用程式卡片將基礎運算資源與應用程式相關聯)。
  - 選擇應用程式名稱，在 myApplications 主控台頁面中顯示應用程式詳細資訊。

- `Cluster`、`Namespace`、和`Workload`用於在 Amazon EKS 中託管的服務，或 `Environment` Amazon ECS 或亞馬 Amazon EC2 託管的服務。對於 Amazon EKS 託管的服務，請選擇任何連結以開啟 CloudWatch 容器見解。

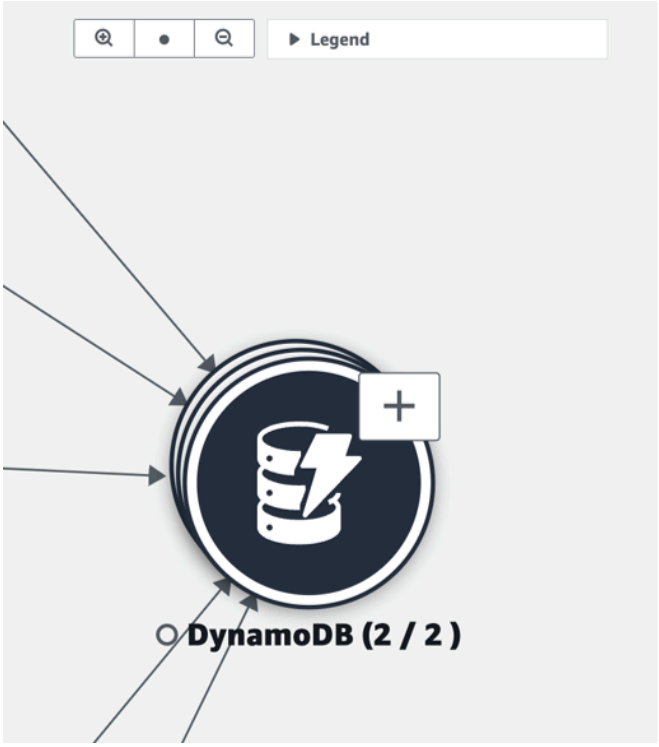
選取服務節點與下游服務或相依性節點之間的邊緣或連接。這會開啟一個窗格，其中包含按故障率、延遲和錯誤率排列的最上層路徑，如下列範例影像所示。選擇窗格中的任何連結，以開啟 [服務詳細資訊] 頁面，並查看所選服務或相依性的詳細資訊。



View dependencies

您的應用程式相依性會顯示在服務對應表上，並連接至呼叫它們的服務。

選擇相依性節點以開啟依據錯誤率、延遲和錯誤率排列的最上層路徑的窗格。選擇任何服務或目標連結，開啟「服務詳細資訊」頁面，並查看有關所選服務或相依性目標的詳細資訊，如下列範例影像所示：



Volume	Avg latency	Error rate	Fault rate
-	-	-	-

Top three paths by fault rate

Service	Remote operation	Fault rate
No paths with faults		

Top three paths by latency

Service	Remote operation	Avg latency
billing-service-ec2-python	PutItem	282.8 ms
billing-service-python	PutItem	75.6 ms
visits-service-java	PutItem	64.9 ms

Top three paths by error rate

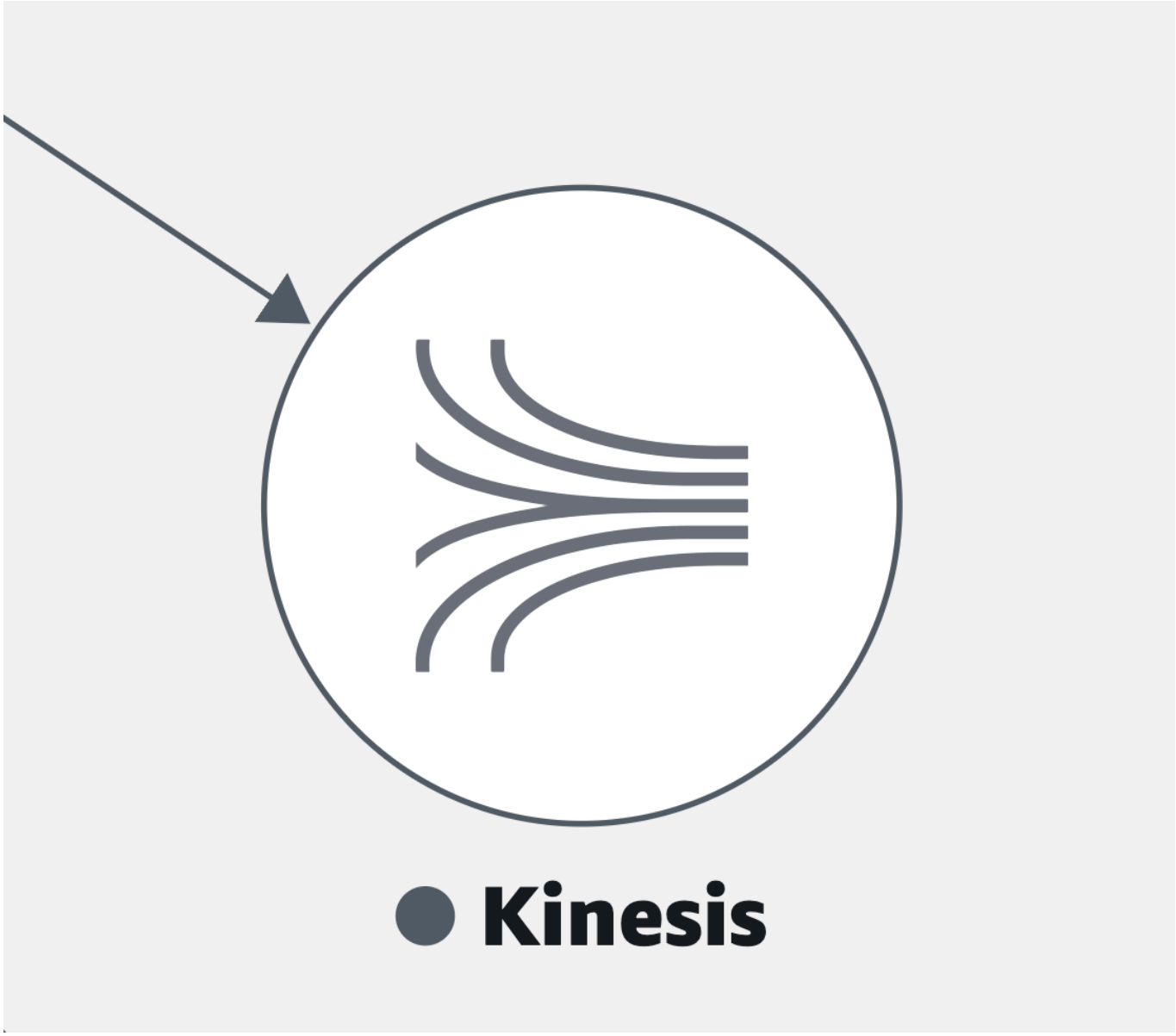
Service	Remote operation	Error rate
visits-service-java	PutItem	9.6%

依預設，服務相依性會群組成單一可展開圖示。選取 (+) 圖示 (如上圖所示) 以展開群組並查看其個別元素。

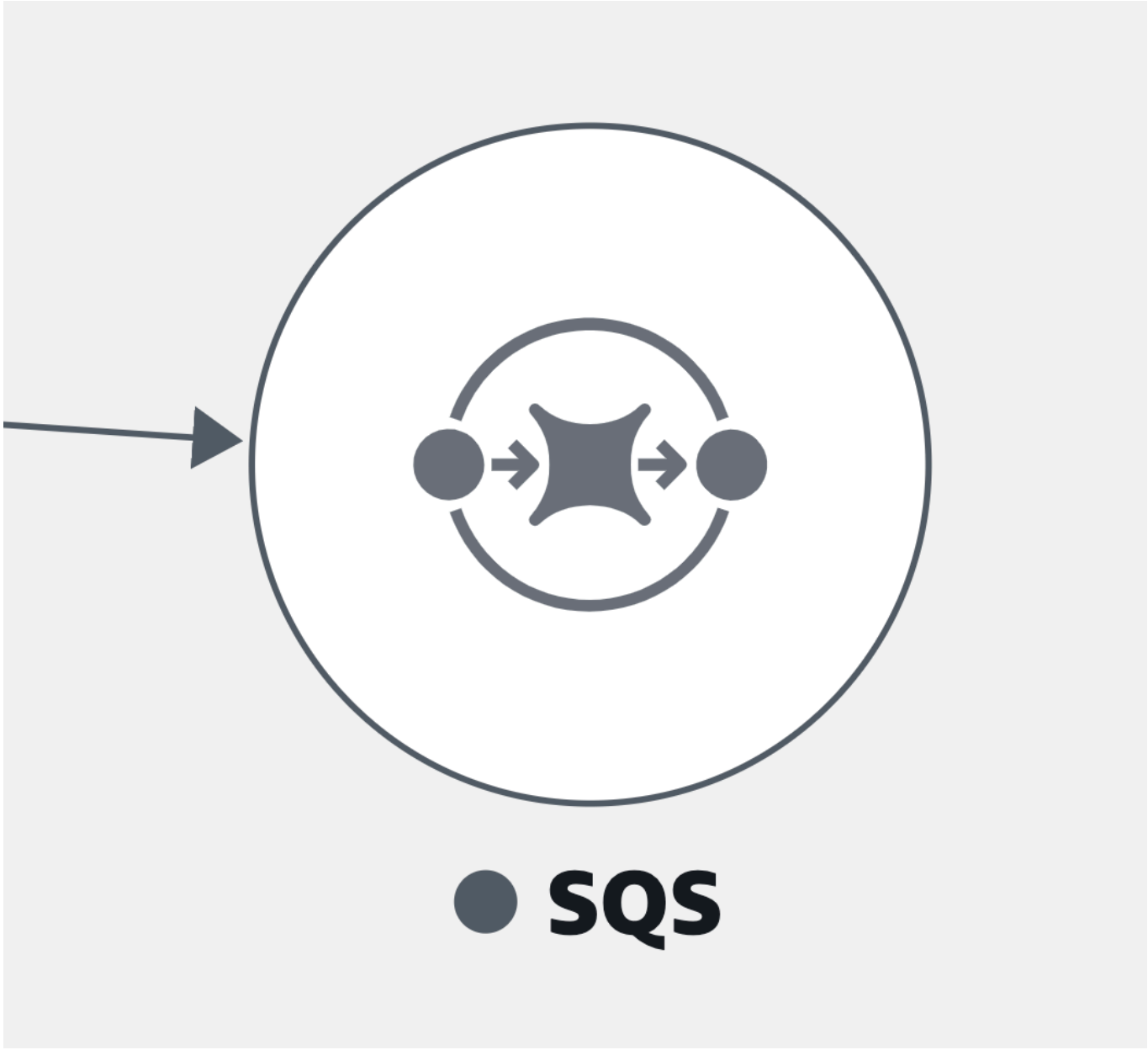
下列圖示代表服務對應中相依性節點的範例：

- Amazon S3 桶：

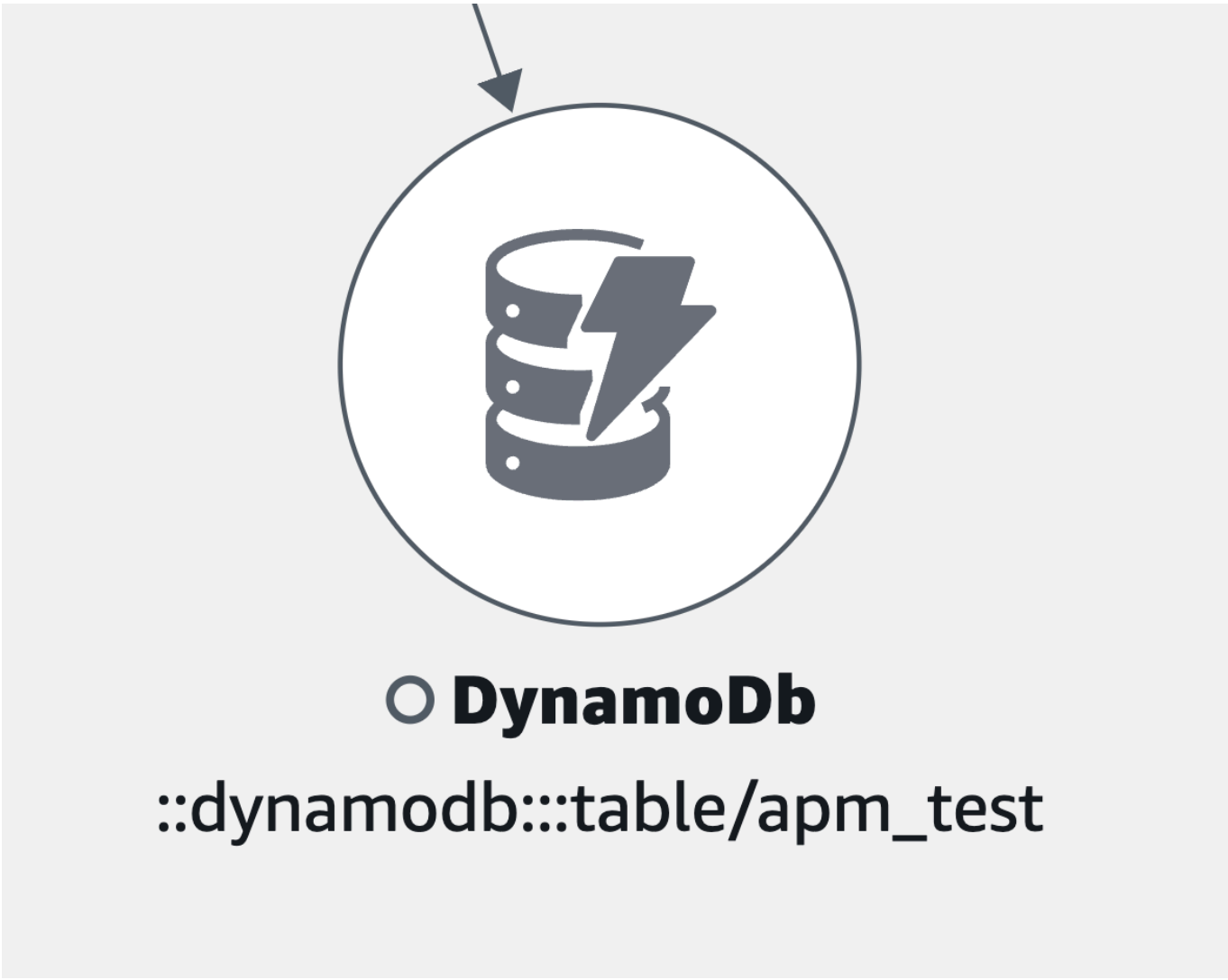
- Amazon Kinesis 流：



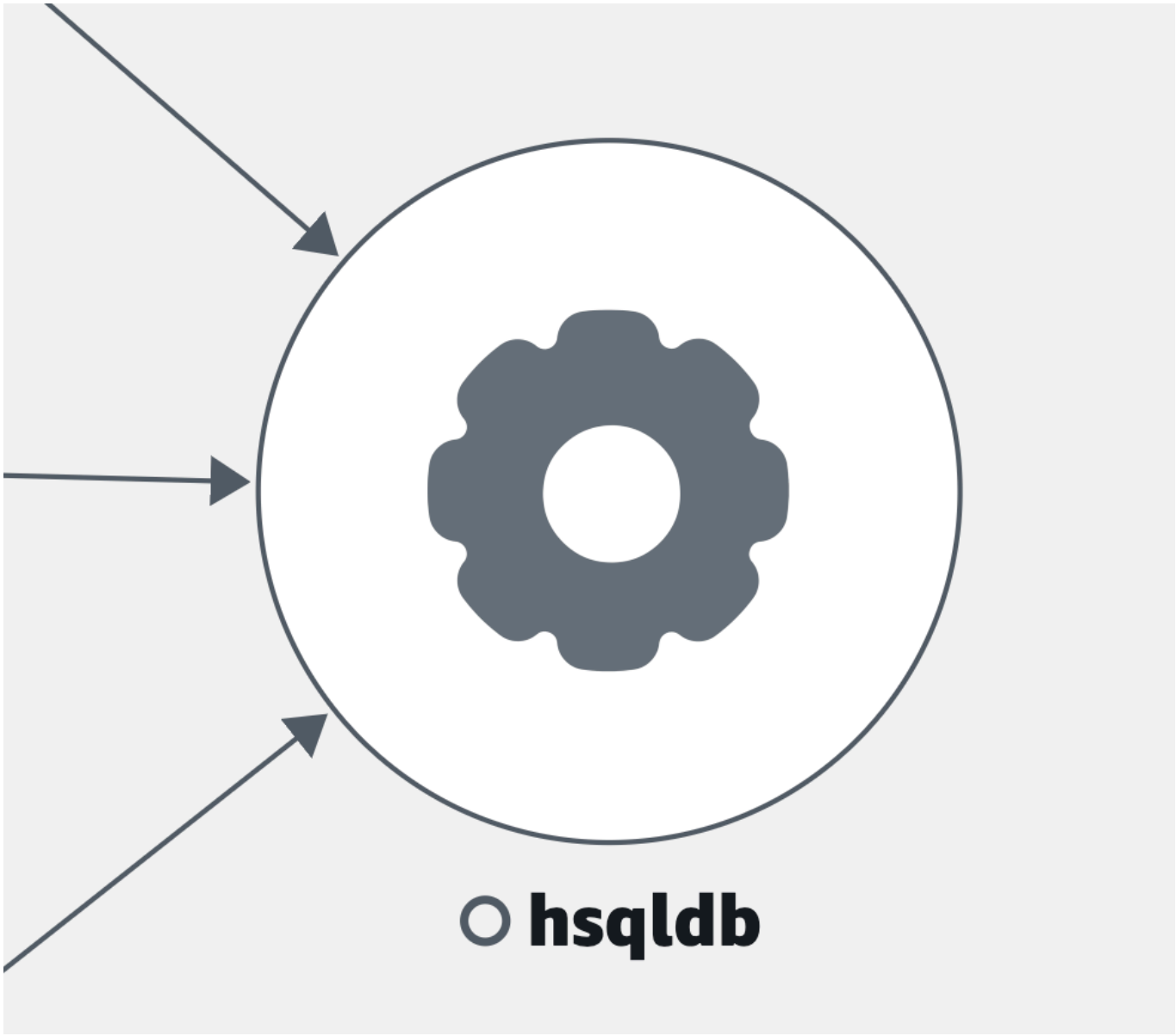
- Amazon 簡單隊列服務（Amazon SQS）：



- 亞 Amazon DynamoDB 表格：



- 先前未列出的其他相依性類型：



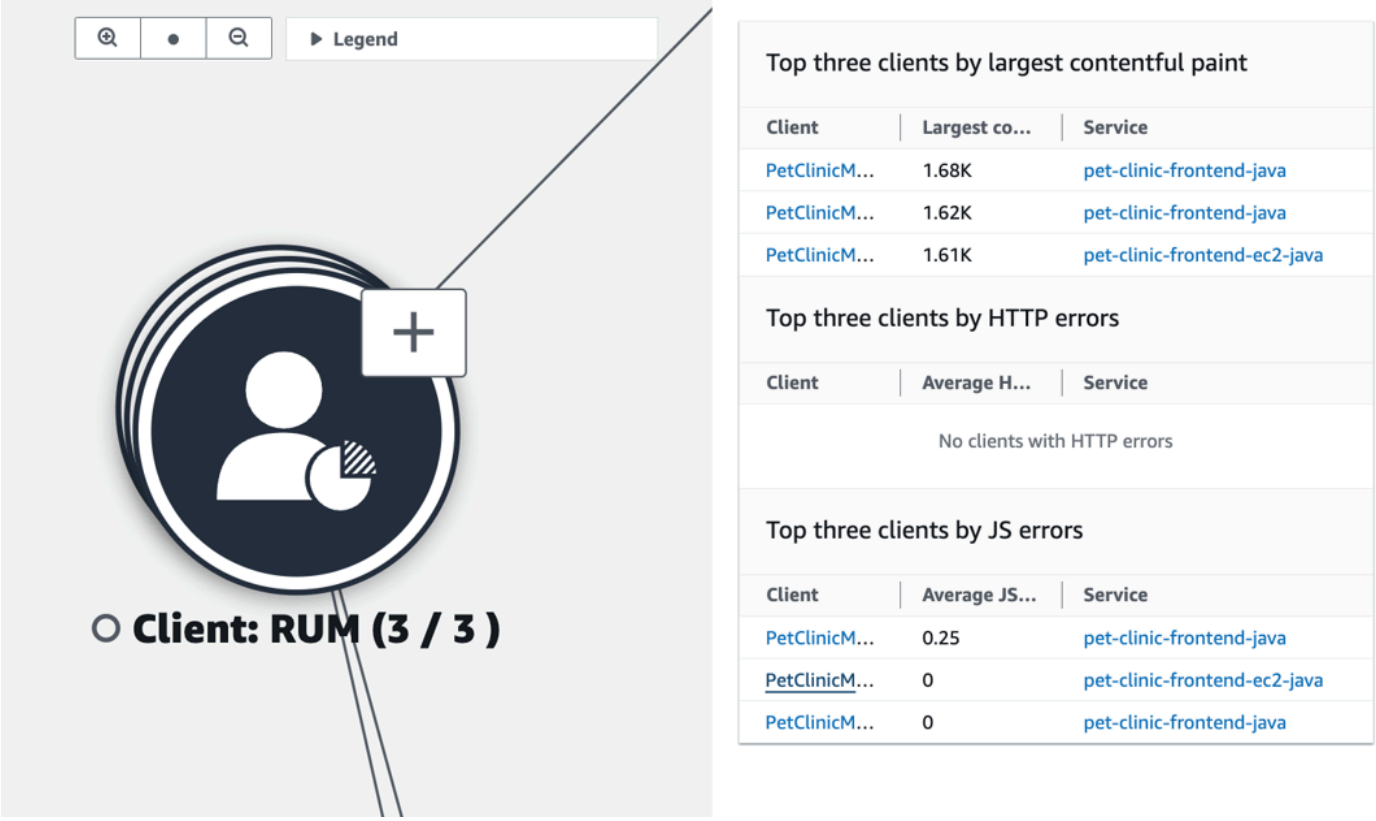
View clients

在您為 R CloudWatch UM 網頁用戶端開啟 X-Ray 追蹤之後，這些用戶端會顯示在連線到他們呼叫之服務的服務對應上。

選擇用戶端節點以開啟顯示詳細用戶端資訊的窗格：

- 頁面載入、平均載入時間、錯誤和平均 Web 關鍵數值的指標。
- 顯示錯誤明細的圖表。
- 在 CloudWatch RUM 中顯示用戶端詳細資料的連結。

根據預設，RUM 用戶端會群組成單一的可擴充圖示。選取 (+) 圖示 (如下圖所示) 以展開群組並查看其個別元素。



下列圖示代表服務對應中 RUM 用戶端的範例：

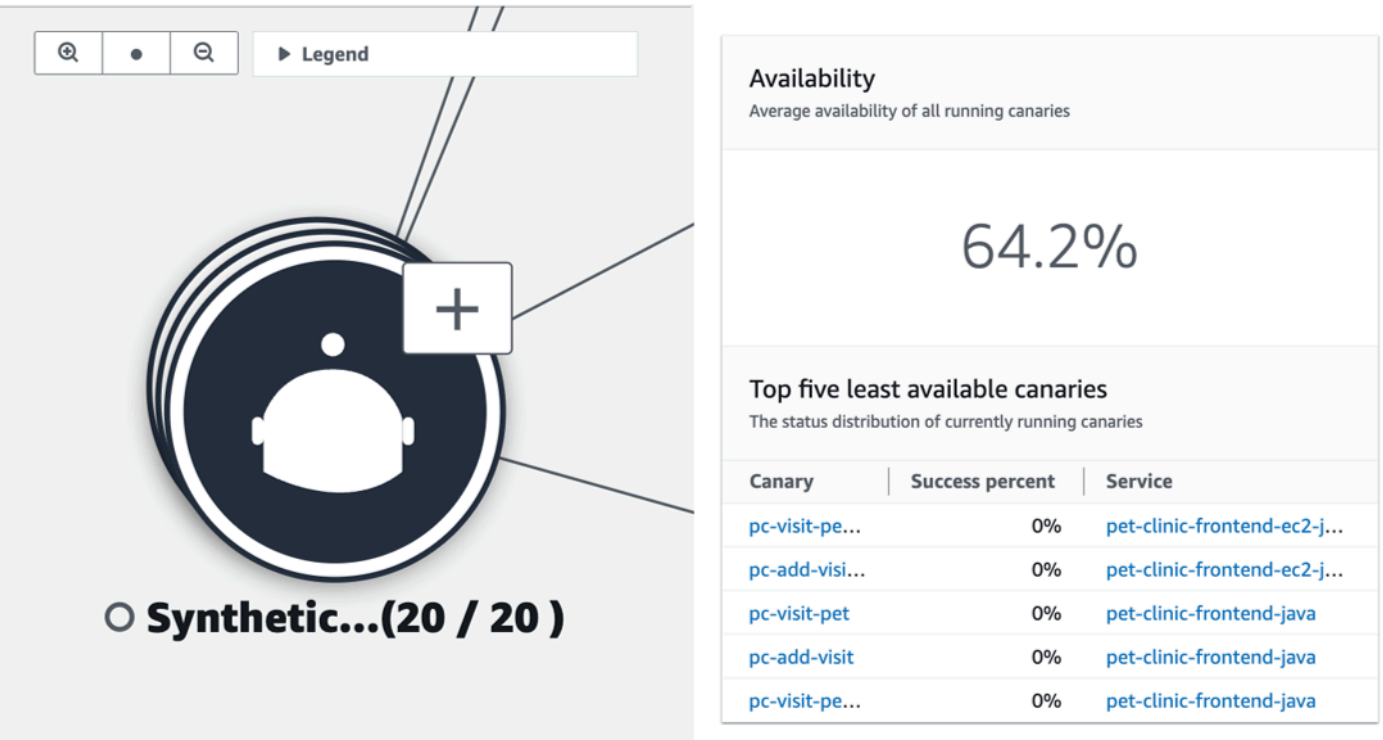
- 朗姆酒客戶端 —

> **Note**
>
> 若要查看用戶端頁面中的 AJAX 錯誤，請使用 CloudWatch RUM 網頁用戶端版本 1.15 或更新版本。

View synthetics canaries

開啟 CloudWatch Synthetics 金絲雀的 AWS X-Ray 追蹤後，它們會顯示在連接至其所呼叫服務的服務對應上，如下列範例影像所示：

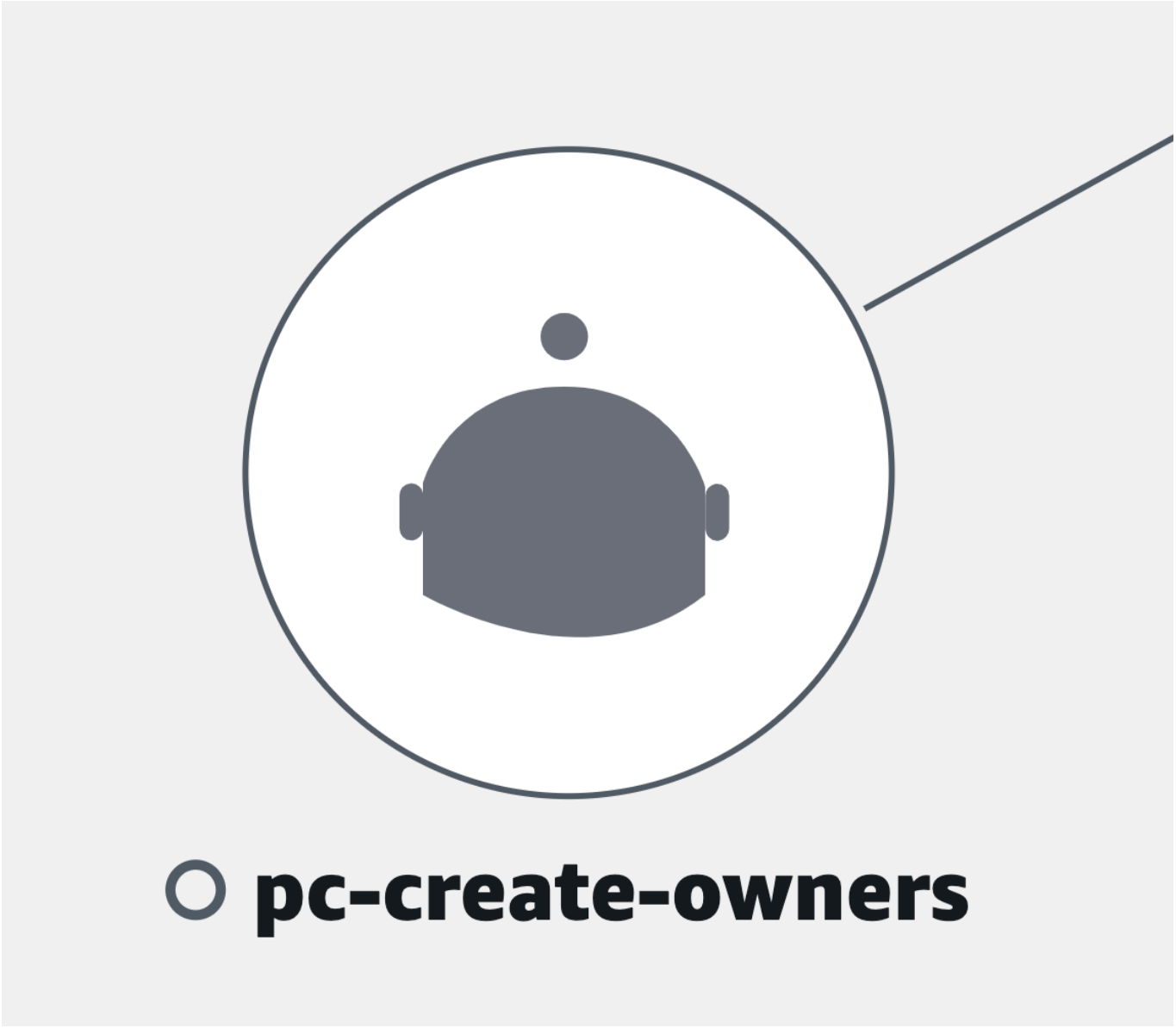
選擇初期測試節點以開啟顯示詳細測試資訊的窗格，如下圖所示：

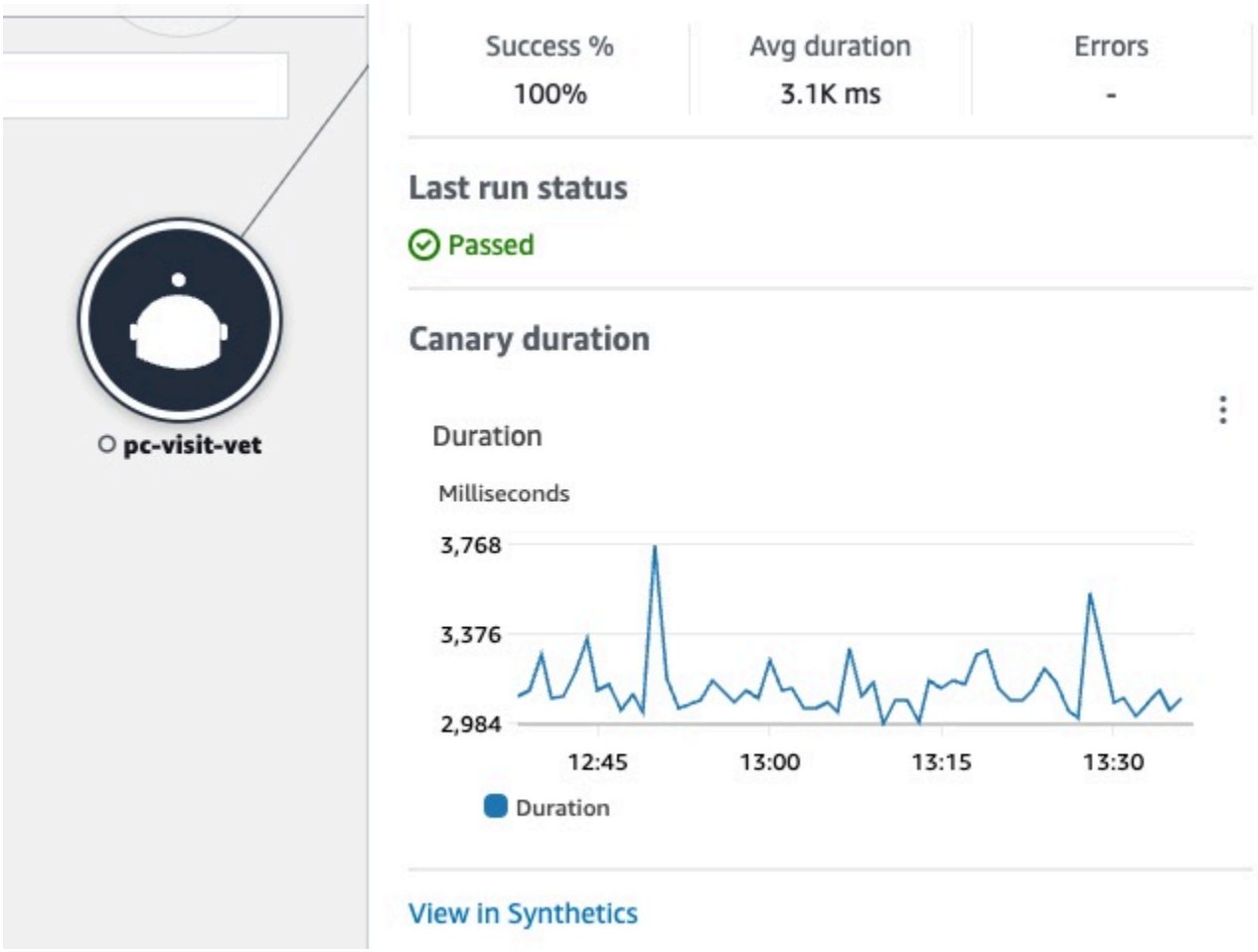


金絲雀群組在一起默認情況下成為一個單一的可擴展圖標。選取 (+) 圖示 (如上圖所示) 以展開群組並查看其個別元素。

下列圖示代表服務對應中用戶端的範例：

- 合成金絲雀 —


pc-create-owners



在初期測試節點的窗格中，您可以看到以下內容：

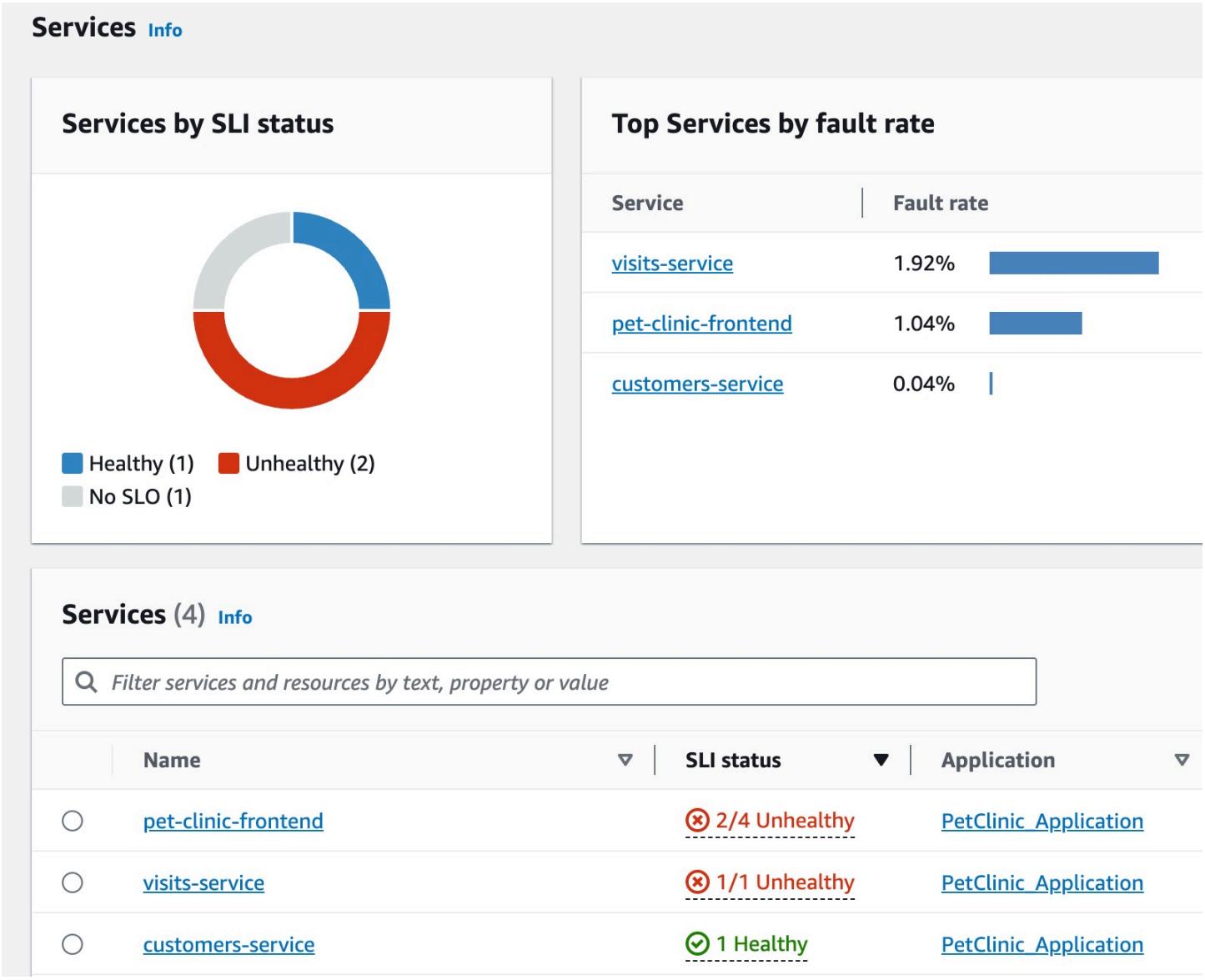
- 成功百分比、平均持續時間和錯誤的指標。
- 上次 canary 執行的狀態。
- 顯示 canary 執行持續時間的圖表。將游標暫留在圖表序列上，即可看到包含更多資訊的快顯視窗
- 在 CloudWatch Synthetics 中顯示金絲雀細節的鏈接。

範例：使用 Application Signals 來解決操作運作狀態問題

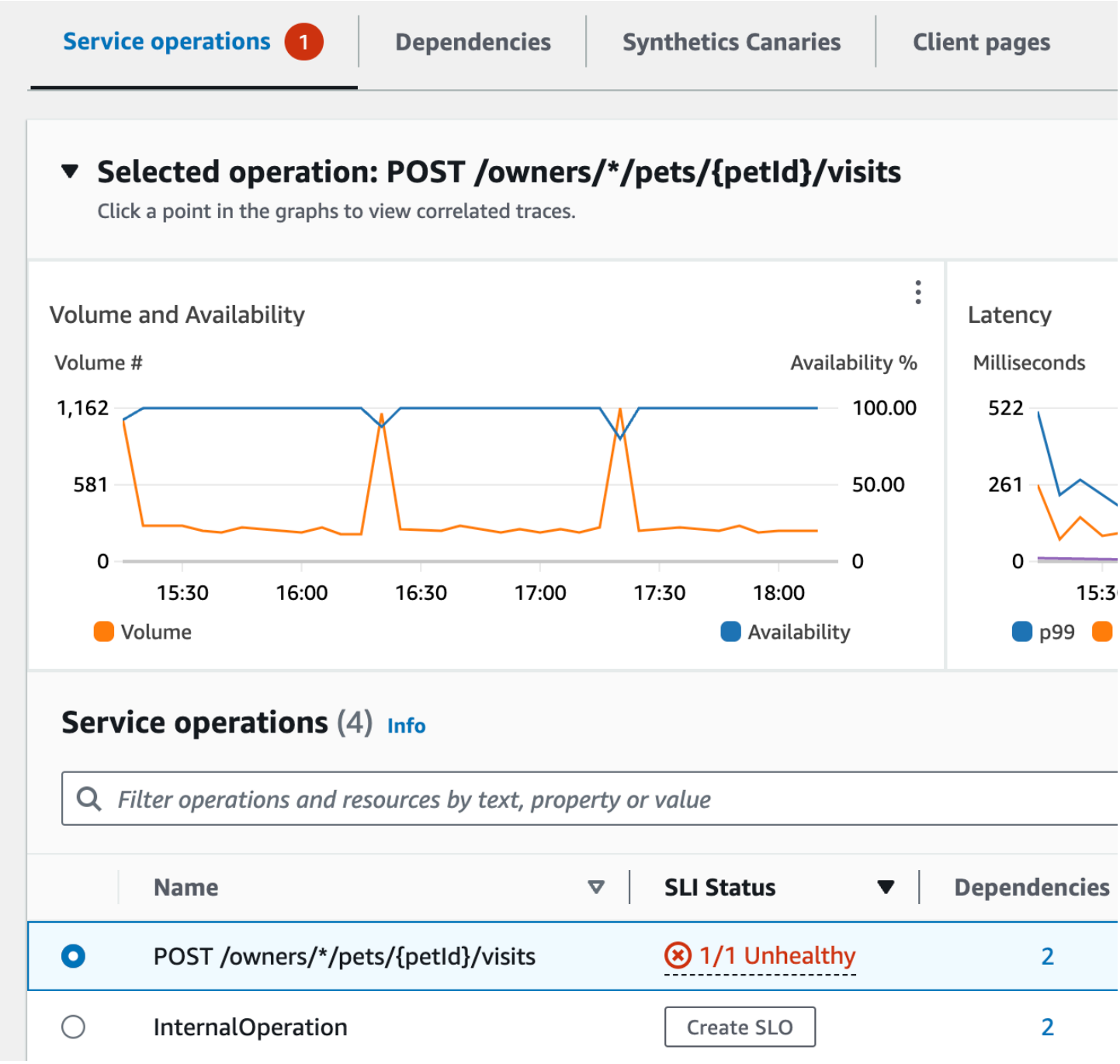
⚠ 應用程式訊號正在適用於 Amazon 的預覽版本中， CloudWatch 且可能會有所變更。

下列案例提供如何使用 Application Signals 來監控服務及識別服務品質問題的範例。深入分析以找出潛在的根本原因，並採取行動來解決問題。此範例著重於寵物診所應用程式，該應用程式由多個呼叫的微服務 AWS 服務 (例如 DynamoDB) 組成。

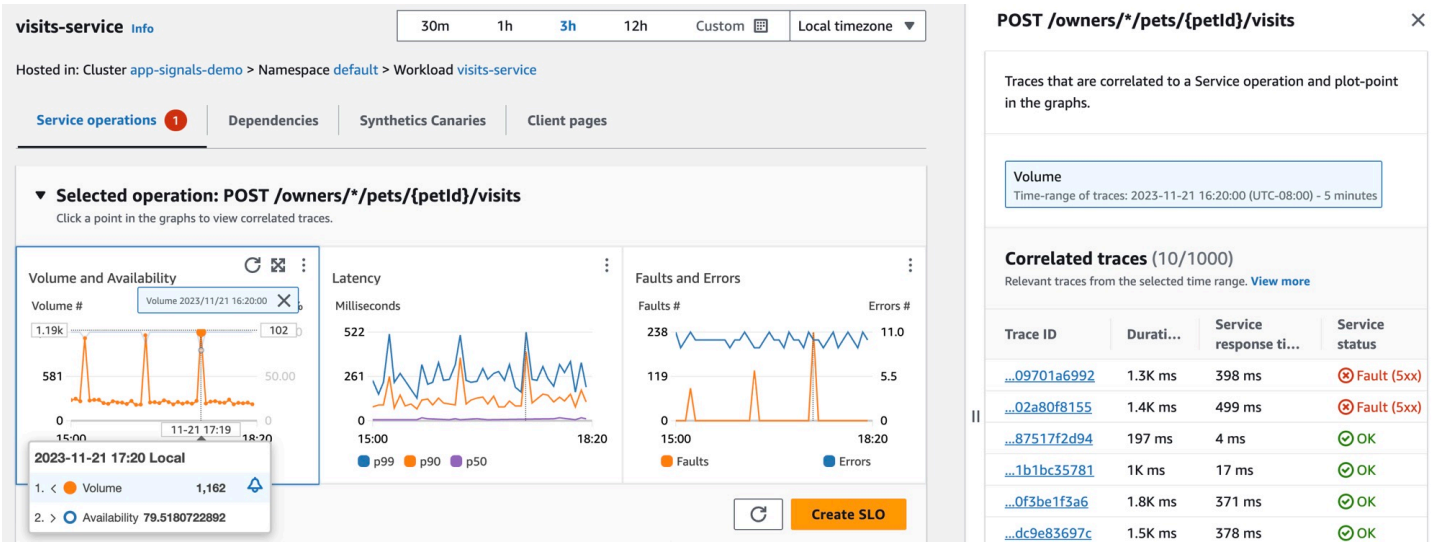
Jane 是一個 DevOps 團隊的成員，負責監督寵物診所申請的運營健康狀況。Jane 的團隊致力於確保應用程式具有高可用性和高回應速度。他們使用服務水準目標 (SLO)，根據這些業務承諾來衡量應用程式效能。她收到有關數個不良服務水準指標 (SLI) 的警示。她開啟 CloudWatch 主控台並導覽至「服務」頁面，在此頁面上看到數個服務處於不健康狀態。



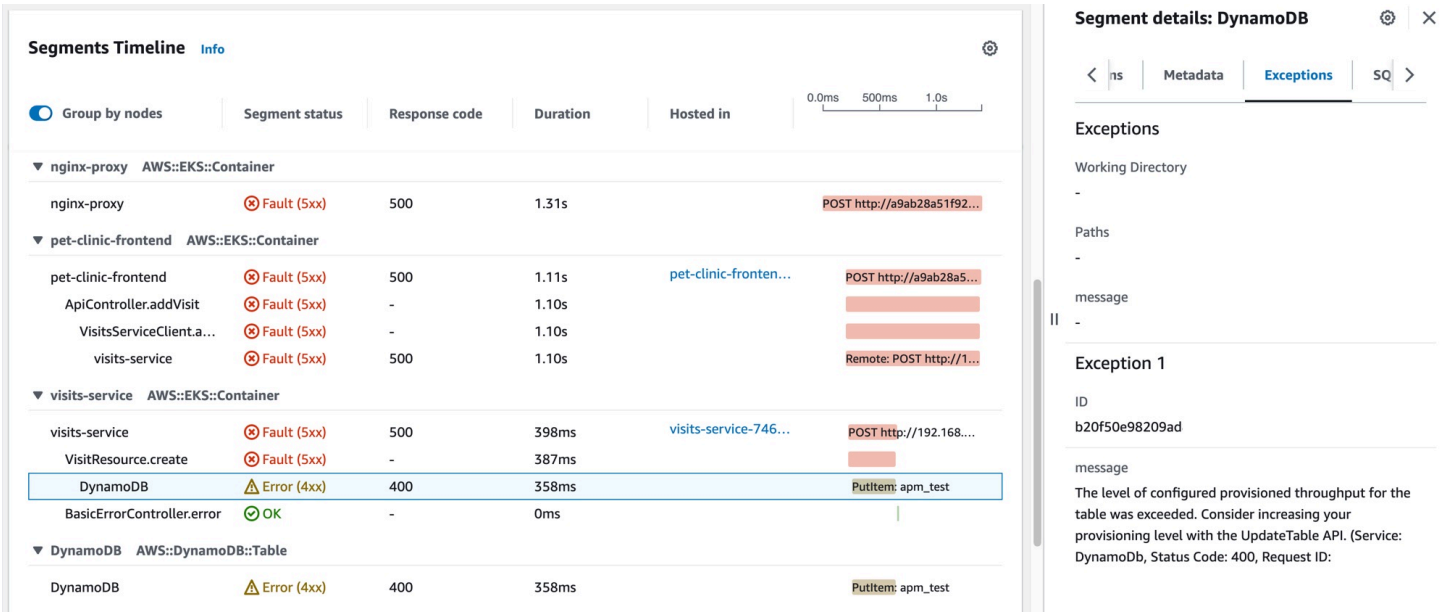
在頁面頂端，Jane 看到 `visits-service` 是故障率最高的服務。她選取圖表中的連結，開啟該服務的「服務」詳細資訊頁面。她看到服務操作資料表中有狀態不佳的操作。她選取此操作，並在「磁碟區與可用性」圖表中看到有定期呼叫量尖峰，這似乎與可用性下降相關。



為了更深入了解服務可用性的下降，Jane 會在圖表中選取其中一個可用性資料點。隨即開啟一個選單，其中顯示與所選資料點相關的 X-Ray 追蹤。她看到有多個包含故障的追蹤。



Jane 會選取其中一個具有錯誤狀態的相關追蹤，這會開啟所選追蹤的「X-Ray 追蹤」詳細資訊頁面。Jane 向下捲動至「區段時間軸」部分，並遵循呼叫路徑，直到看到對 DynamoDB 資料表的呼叫傳回錯誤為止。她選取 DynamoDB 區段，並導覽至右側抽屜的「例外狀況」索引標籤。



Jane 發現 DynamoDB 資源設定錯誤，導致客戶請求尖峰期間發生錯誤。DynamoDB 資料表的佈建輸送量水平會定期超出範圍，導致服務可用性問題和運作狀態不佳的 SLI。根據此資訊，她的團隊能夠設定更高層級的佈建輸送量，並確保應用程式的高可用性。

收集的標準應用程式指標

⚠ Application Signals 為預覽版本。如果您對此功能有任何意見，可以透app-signals-feedback過 @amazon .com 與我們聯絡。

Application Signals 會從它發現的服務中收集標準應用程式指標。這些指標與服務效能的最重要方面有關：延遲、故障和錯誤。它們可協助您識別問題、監控效能趨勢並最佳化資源，以改善整體使用者體驗。

下表列出 Application Signals 收集的指標。這些量度會在`AppSignals`命名空間 CloudWatch 中傳送至。

指標	描述
`Latency`	提出請求後，資料傳輸開始之前的延遲時間。 單位：毫秒
`Faults`	HTTP 5XX 伺服器端錯誤和 OpenTelemetry 跨度狀態錯誤的計數。 單位：無
`Errors`	HTTP 4XX 用戶端錯誤的計數。這些錯誤被認為是並非由服務問題引起的請求錯誤。因此，Application Signals 儀表板上顯示的 `Availability` 指標不會將這些錯誤視為服務錯誤。 單位：無

顯示在「應用程式訊號」儀表板上的`Availability`量度計算為 (1-**`Faults`**/總計) * 100。成功的回應是沒有 5XX 錯誤的所有回應。當 Application Signals 計算 `Availability` 時，4XX 回應會被視為成功。

收集的維度與維度組合

以下是針對每個標準應用程式指標定義的維度。如需維度的詳細資訊，請參閱 維度。

會針對服務指標和相依性指標收集不同的維度。在 Application Signals 發現的服務中，當微服務 A 呼叫微服務 B 時，微服務 B 正在提供請求。在此情況下，微服務 A 會發出相依性指標，而微服務 B 會發出服務指標。當用戶端呼叫微服務 A 時，微服務 A 會提供請求並發出服務指標。

服務指標的維度

為服務指標收集以下維度。

維度	描述
`Service`	服務的名稱。
`Operation`	API 操作或其他活動的名稱。
`HostedIn.EKS.Cluster`	執行服務的 Amazon EKS 叢集的名稱。 只有在 Amazon EKS 上執行服務時，才會收集此維度。
`HostedIn.K8s.Namespace`	執行服務之 Abernetes 命名空間的名稱。 只有在 Amazon EKS 上執行服務時，才會收集此維度。
`HostedIn.Environment`	執行服務之環境的使用者定義名稱。 只有在並非 Amazon EKS 的環境中執行服務時，才會收集此維度。

在 CloudWatch 主控台中檢視這些測量結果時，您可以選擇使用下列維度組合來檢視它們。

- `Service`, `Operation`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`
- `Service`, `HostedIn.EKS.Cluster`, `HostedIn.K8s.Namespace`

對於非 Amazon EKS 平台，也可以檢視具有下列維度組合的服務指標。

- `Service`, `Operation`, `HostedIn.Environment`
- `Service`, `HostedIn.Environment`

相依性指標的維度

為相依性指標收集以下維度。

維度	描述
Service	服務的名稱。
Operation	API 操作或其他活動的名稱。
RemoteService	正在調用之遠端服務的名稱。
RemoteOperation	正在調用之 API 操作的名稱。
HostedIn.EKS.Cluster	執行服務的 Amazon EKS 叢集的名稱。 只有在 Amazon EKS 上執行服務時，才會收集此維度。
HostedIn.K8s.Namespace	執行服務之 Abernetes 命名空間的名稱。 只有在 Amazon EKS 上執行服務時，才會收集此維度。
K8s.RemoteNamespace	正在執行相依性服務之 Kubernetes 命名空間的名稱。 只有在 Amazon EKS 上執行服務時，才會收集此維度。
RemoteTarget	遠端呼叫所調用之資源的名稱。如果遠端呼叫不是針對任何特定資源，則此維度沒有任何值。 只有在 Amazon EKS 上執行服務時，才會收集此維度。
HostedIn.Environment	執行服務之環境的使用者定義名稱。 只有在並非 Amazon EKS 的環境中執行服務時，才會收集此維度。

在 CloudWatch 主控台中檢視這些測量結果時，您可以選擇使用下列維度組合來檢視它們。

在任何平台上執行

- RemoteService

在 Amazon EKS 叢集上執行

- Service, Operation, HostedIn.EKS.Cluster, HostedIn.K8s.Namespace, RemoteService, RemoteOperation, K8s.RemoteNamespace, RemoteTarget
- Service, Operation, HostedIn.EKS.Cluster, HostedIn.K8s.Namespace, RemoteService, RemoteOperation, K8s.RemoteNamespace
- Service, Operation, HostedIn.EKS.Cluster, HostedIn.K8s.Namespace, RemoteService, RemoteOperation, RemoteTarget
- Service, Operation, HostedIn.EKS.Cluster, RemoteService, RemoteOperation,
- Service, HostedIn.EKS.Cluster, HostedIn.K8s.Namespace, RemoteService, K8s.RemoteNamespace
- Service, HostedIn.EKS.Cluster, RemoteService, K8s.RemoteNamespace
- Service, HostedIn.EKS.Cluster, HostedIn.K8s.Namespace, RemoteService, RemoteOperation, K8s.RemoteNamespace, RemoteTarget
- Service, HostedIn.EKS.Cluster, HostedIn.K8s.Namespace, RemoteService, RemoteOperation, K8s.RemoteNamespace
- Service, HostedIn.EKS.Cluster, HostedIn.K8s.Namespace, RemoteService, RemoteOperation, RemoteTarget
- Service, HostedIn.EKS.Cluster, HostedIn.K8s.Namespace, RemoteService, RemoteOperation

在 Amazon EKS 叢集以外的平台上執行

- Service, Operation, HostedIn.Environment
- Service, HostedIn.Environment
- Service, Operation, HostedIn.Environment, RemoteService, RemoteOperation, RemoteTarget
- Service, Operation, HostedIn.Environment, RemoteService, RemoteOperation,
- Service, HostedIn.Environment, RemoteService
- Service, HostedIn.Environment, RemoteService, RemoteOperation, RemoteTarget
- Service, HostedIn.Environment, RemoteService, RemoteOperation,

使用綜合監控

您可以使用 Amazon CloudWatch Synthetics 建立按排程執行的金絲雀、可設定的指令碼，以監控您的端點和 API。Canary 遵循相同的路由並執行與客戶相同的動作，即使您的應用程式沒有任何客戶流量，也能持續驗證您的客戶體驗。透過使用 Canary，您可以在客戶之前發現問題。

Canary 是以 Node.js 或 Python 撰寫的指令碼。Canary 會使用 Node.js 或 Python 作為架構，在您的帳戶中建立 Lambda 函數。Canary 透過 HTTP 和 HTTPS 通訊協定運作。金絲雀使用包含 CloudWatch Synthetics 庫的 Lambda 層。該庫包含用於 NodeJS 金絲雀的 CloudWatch Synthetics 的 NodeJS 版本和 Python 版本的 Synthetics 金絲雀。 CloudWatch 這些圖層屬於 S CloudWatch ynthetics 服務帳戶。圖書館永遠不會傳輸或存儲客戶信息。所有客戶數據僅存儲在客戶帳戶中。

Canary 透過 Puppeteer 或 Selenium Webdriver 提供對無周邊 Google Chrome 瀏覽器的程式設計存取。如需 Puppeteer 的詳細資訊，請參閱 Puppeteer。如需 Selenium 的詳細資訊，請參閱 www.selenium.dev/。

Canary 會檢查端點的可用性和延遲，並可儲存 UI 的載入時間資料和螢幕擷取畫面。它們會監控您的 REST API、URL 和網站內容，並且可以檢查來自網路釣魚、程式碼插入和跨網站指令碼的未經授權變更。

CloudWatch Synthetics 與應用程序信號集成在一起，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。若要在 Application Signals 中查看您的 Canaries，請開啟 X-Ray 作用中追蹤。您的 Canary 會顯示在與您的服務相連的服務地圖以及它們呼叫之服務的服務詳細資訊頁面中。

如需 Canary 的影片示範，請參閱以下內容：

- Amazon CloudWatch Synthetics 簡介
- Amazon CloudWatch Synthetics 演示
- 使用 Amazon S CloudWatch ynthetics 建金絲雀
- 使用 Amazon CloudWatch Synthetics 進行視覺監控

您可以只執行一次 Canary，也可以定期執行。Canary 可以每分鐘執行一次。您可以使用 Cron 與 Rate 表達式來排程 Canary。

如需建立和執行 Canary 之前要考量之安全性問題的相關資訊，請參閱Synthetics Canary 的安全考量。

默認情況下，金絲雀會在CloudWatchSynthetics命名空間中創建多個 CloudWatch 指標。這些指標具有 CanaryName 的維度。使用函數庫中 executeStep() 或 executeHttpStep() 函數的 Canary 也具有 StepName 的維度。如需 Canary 函數庫的詳細資訊，請參閱適用於 Canary 指令碼的程式庫函數。

CloudWatch Synthetics 與 X-Ray 跟踪地圖很好地集成在一起，該映射可 CloudWatch 用 AWS X-Ray 於提供服務的 end-to-end 視圖，以幫助您更有效地查明性能瓶頸並識別受影響的用戶。您使用 CloudWatch Synthetics 建立的金絲雀會出現在軌跡地圖上。如需詳細資訊，請參閱 X-Ray 追蹤地圖。

CloudWatch Synthetics 目前已在所有商業 AWS 地區和地區提供。 GovCloud

> **Note**
>
> 在亞太區域 (大阪)， AWS PrivateLink 不受支援。在亞太區域 (雅加達), AWS PrivateLink 與 X-Ray 不受支援。

主題

- CloudWatch加那利群島所需的角色和權限
- 建立 Canary
- 群組
- 在本地測試金絲雀
- 對失敗的 Canary 進行故障診斷
- Canary 指令碼的範本程式碼
- Canary 和 X-Ray 追蹤
- 在 VPC 上執行 Canary
- 加密 Canary 成品
- 檢視 Canary 統計資料和詳細資訊
- CloudWatch 加那利群島發布的指標
- 編輯或刪除 Canary
- 啟動、停止、刪除或更新多個 Canary 的執行階段
- 用 Amazon 監控金絲雀事件 EventBridge

CloudWatch加那利群島所需的角色和權限

建立和管理 Canary 的使用者以及 Canary 本身都必須具有特定許可。

管理 CloudWatch 加那利群島的使用者所需的角色和權限

若要檢視 Canary 詳細資訊和 Canary 執行的結果，您必須以已連接 `CloudWatchSyntheticsFullAccess` 或 `CloudWatchSyntheticsReadOnlyAccess` 政策的使用者身分登入。若要讀取主控台中的所有 Synthetics 資料，您還需要 `AmazonS3ReadOnlyAccess` 和 `CloudWatchReadOnlyAccess` 政策。若要檢視 Canary 使用的原始程式碼，您也需要 `AWSLambda_ReadOnlyAccess` 政策。

若要建立 Canary，您必須以具有 `CloudWatchSyntheticsFullAccess` 政策或類似許可集合的使用者身分登入。若要為 Canary 建立 IAM 角色，您還需要下列內嵌政策陳述式：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": [
                "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*",
                "arn:aws:iam::*:policy/service-role/CloudWatchSyntheticsPolicy*"
            ]
        }
    ]
}
```

> **Important**
>
> 授與使用者`iam:CreateRole`、`iam:CreatePolicy`、和`iam:AttachRolePolicy`權限可授與該使用者對您 AWS 帳戶的完整管理存取權。例如，擁有這些許可的使用者，可以建立具有所有資源完整許可的政策，並可將該政策連接至任何角色。對於您授與這些許可的對象，請務必謹慎。

如需連接政策和授與許可給使用者的資訊，請參閱變更 IAM 使用者的許可和嵌入使用者或角色的內嵌政策。

Canary 的必要角色和許可

每個 Canary 必須與連接特定許可的 IAM 角色相關聯。使用 CloudWatch 主控台建立初期測試時，您可以選擇 CloudWatch Synthetics 為初期測試建立 IAM 角色。若您這樣做，則角色將具備所需要的許可。

如果您想要自己建立 IAM 角色，或者建立在使用 AWS CLI 或 API 建立 Canary 時可以使用的 IAM 角色，該角色必須包含本節中列出的許可。

Canary 的所有 IAM 角色必須包含以下信任政策陳述式。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

此外，Canary 的 IAM 角色需要以下陳述式之一。

不使用 AWS KMS 或不需要 Amazon VPC 訪問的基本金絲雀

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::path/to/your/s3/bucket/canary/results/folder"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::name/of/the/s3/bucket/that/contains/canary/results"

            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:canary_region_name:canary_account_id:log-group:/aws/
lambda/cwsyn-canary_name-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "xray:PutTraceSegments"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": "cloudwatch:PutMetricData",
            "Condition": {
               "StringEquals": {
                "cloudwatch:namespace": "CloudWatchSynthetics"
               }
            }
        }
    ]
```

```
}
```

金絲雀用於 AWS KMS 加密初期測試成品，但不需要 Amazon VPC 存取權

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::path/to/your/S3/bucket/canary/results/folder"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::name/of/the/S3/bucket/that/contains/canary/results"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:canary_region_name:canary_account_id:log-group:/aws/
lambda/cwsyn-canary_name-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "xray:PutTraceSegments"
            ],
```

```
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": "cloudwatch:PutMetricData",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "CloudWatchSynthetics"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource":
 "arn:aws:kms:KMS_key_region_name:KMS_key_account_id:key/KMS_key_id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": [
                        "s3.region_name_of_the_canary_results_S3_bucket.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

金絲雀不使用 AWS KMS 但確實需要 Amazon VPC 訪問

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::path/to/your/S3/bucket/canary/results/folder"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::name/of/the/S3/bucket/that/contains/canary/results"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:canary_region_name:canary_account_id:log-group:/aws/
lambda/cwsyn-canary_name-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "xray:PutTraceSegments"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": "cloudwatch:PutMetricData",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "CloudWatchSynthetics"
                }
            }
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DeleteNetworkInterface"
            ],
            "Resource": [
                "*"
            ]
       }
    ]
}
```

金絲雀用於 AWS KMS 加密初期測試成品，並且還需要 Amazon VPC 訪問

如果您更新非 VPC Canary 以開始使用 VPC，則需要更新 Canary 的角色以包含以下政策中列出的網路介面許可。

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::path/to/your/S3/bucket/canary/results/folder"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::name/of/the/S3/bucket/that/contains/canary/results"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup"
            ],
            "Resource": [
                "arn:aws:logs:canary_region_name:canary_account_id:log-group:/aws/
lambda/cwsyn-canary_name-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "xray:PutTraceSegments"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": "cloudwatch:PutMetricData",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "CloudWatchSynthetics"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DeleteNetworkInterface"
            ],
            "Resource": [
                "*"
            ]
       },
       {
            "Effect": "Allow",
            "Action": [
```

```
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource":
 "arn:aws:kms:KMS_key_region_name:KMS_key_account_id:key/KMS_key_id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": [
                        "s3.region_name_of_the_canary_results_S3_bucket.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

AWS CloudWatch Synthetics 管理政策

若要新增許可給使用者、群組和角色，使用 AWS 受管政策比自己撰寫政策更容易。建立 IAM 客戶受管政策需要時間和專業知識，而受管政策可為您的團隊提供其所需的許可。若要快速開始使用，您可以使用我們的 AWS 受管政策。這些政策涵蓋常見使用案例，並可在您的 AWS 帳戶中使用。如需 AWS 受管政策的詳細資訊，請參閱《IAM 使用者指南》中的AWS 受管政策 AWS 受管政策。

AWS 服務會維護和更新 AWS 受管理的策略。您無法變更 AWS 受管理原則中的權限。服務偶爾會變更 AWS 受管政策中的許可。此類型的更新會影響已連接政策的所有身分識別 (使用者、群組和角色)。

CloudWatch AWS 受管理政策的 Synthetics 更新

檢視有關 CloudWatch Synthetics AWS 管理政策更新的詳細資料，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「 CloudWatch 文件歷史記錄」頁面上的 RSS 摘要。

變更	描述	日期
從中移除多餘動作 CloudWatchSyntheticsFullAccess	CloudWatch Synthetics 移除CloudWatchSyntheticsFullAccess原則中的`s3:PutBucketEncryption` 和`lambda:GetLayerVersionByArn` 動作，因為這些動作與原則中的其他權限是多餘的。移除的動	2021 年 3 月 12 日

變更	描述	日期
	作並未提供任何許可，而且政策授予的許可沒有網路變更。	
CloudWatch Synthetics 開始追蹤變化	CloudWatch Synthetics 開始追蹤其 AWS 管理政策的變更。	2021 年 3 月 10 日

CloudWatchSyntheticsFullAccess

以下是 CloudWatchSyntheticsFullAccess 政策的內容：

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "synthetics:*"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:CreateBucket",
            "s3:PutEncryptionConfiguration"
         ],
         "Resource":[
            "arn:aws:s3:::cw-syn-results-*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "iam:ListRoles",
            "s3:ListAllMyBuckets",
            "s3:GetBucketLocation",
            "xray:GetTraceSummaries",
            "xray:BatchGetTraces",
            "apigateway:GET"
         ],
```

```
            "Resource":"*"
        },
        {
            "Effect":"Allow",
            "Action":[
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource":"arn:aws:s3:::cw-syn-*"
        },
        {
            "Effect":"Allow",
            "Action":[
                "s3:GetObjectVersion"
            ],
            "Resource":"arn:aws:s3:::aws-synthetics-library-*"
        },
        {
            "Effect":"Allow",
            "Action":[
                "iam:PassRole"
            ],
            "Resource":[
                "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*"
            ],
            "Condition":{
                "StringEquals":{
                    "iam:PassedToService":[
                        "lambda.amazonaws.com",
                        "synthetics.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect":"Allow",
            "Action":[
                "iam:GetRole"
            ],
            "Resource":[
                "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*"
            ]
        },
        {
```

```
         "Effect":"Allow",
         "Action":[
            "cloudwatch:GetMetricData",
            "cloudwatch:GetMetricStatistics"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "cloudwatch:PutMetricAlarm",
            "cloudwatch:DeleteAlarms"
         ],
         "Resource":[
            "arn:aws:cloudwatch:*:*:alarm:Synthetics-*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "cloudwatch:DescribeAlarms"
         ],
         "Resource":[
            "arn:aws:cloudwatch:*:*:alarm:*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "lambda:CreateFunction",
            "lambda:AddPermission",
            "lambda:PublishVersion",
            "lambda:UpdateFunctionConfiguration",
            "lambda:GetFunctionConfiguration"
         ],
         "Resource":[
            "arn:aws:lambda:*:*:function:cwsyn-*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "lambda:GetLayerVersion",
            "lambda:PublishLayerVersion"
```

```
         ],
         "Resource":[
            "arn:aws:lambda:*:*:layer:cwsyn-*",
            "arn:aws:lambda:*:*:layer:Synthetics:*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "ec2:DescribeVpcs",
            "ec2:DescribeSubnets",
            "ec2:DescribeSecurityGroups"
         ],
         "Resource":[
            "*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "sns:ListTopics"
         ],
         "Resource":[
            "*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "sns:CreateTopic",
            "sns:Subscribe",
            "sns:ListSubscriptionsByTopic"
         ],
         "Resource":[
            "arn:*:sns:*:*:Synthetics-*"
         ]
      }
   ]
}
```

CloudWatchSyntheticsReadOnlyAccess

以下是 CloudWatchSyntheticsReadOnlyAccess 政策的內容：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "synthetics:Describe*",
                "synthetics:Get*",
                "synthetics:List*"
            ],
            "Resource": "*"
        }
    ]
}
```

限制使用者檢視特定的 canary

您可限制使用者檢視有關 canary 資訊的能力，讓他們只能查看您指定的 canary相關資訊。如要執行此作業，請使用具有類似下列內容之 Condition 陳述式的 IAM 政策，並將此政策連接至使用者或 IAM 角色。

下列範例限制使用者僅可查看 name-of-allowed-canary-1 和 name-of-allowed-canary-2 的相關資訊。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "synthetics:DescribeCanaries",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "synthetics:Names": [
                        "name-of-allowed-canary-1",
                        "name-of-allowed-canary-2"
                    ]
                }
            }
        }
    ]
}
```

CloudWatch Synthetics 支持列出多達五個項目在`synthetics:Names`數組中。

您還可建立於允許的 canary 名稱中使用 * 作為萬用字元的政策，如下列範例所示：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "synthetics:DescribeCanaries",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringLike": {
                    "synthetics:Names": [
                        "my-team-canary-*"
                    ]
                }
            }
        }
    ]
}
```

使用其中一個附加原則登入的任何使用者都無法使用 CloudWatch 主控台檢視任何初期測試資訊。他們只能檢視由政策授權的加那利群島的初期測試資訊，而且只能使用 DescribeCanariesAPI 或描述 AWS CLI 金絲雀命令。

建立 Canary

> **Important**
>
> 請確認您使用 Synthetics Canary 來監控只有您擁有許可 (或您是擁有者) 的端點和 API。取決於 Canary 頻率設定，這些端點可能會遭遇增加的流量。

當您使用 CloudWatch 主控台建立初期測試時，您可以使用提供的藍圖 CloudWatch 來建立初期測試，也可以撰寫自己的指令碼。如需詳細資訊，請參閱 使用 Canary 藍圖。

AWS CloudFormation 如果您使用自己的腳本進行金絲雀，則還可以使用創建金絲雀。若要取得更多資訊，請參閱《AWS CloudFormation 使用指南》AWS::Synthetics::Canary中的。

如果您正在編寫自己的腳本，則可以使用 CloudWatch Synthetics 庫中內置的幾個函數。如需詳細資訊，請參閱 Synthetics 執行時間版本。

> **Note**
>
> 建立金絲雀時，其中一個建立的圖層是附加在前面加上的「Synthetics」圖層。`Synthetics`此圖層由 Synthetics 服務帳戶擁有，並包含運行時代碼。

建立 Canary

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，Synthetics 金絲雀。
3. 選擇 Create Canary (建立 Canary)。
4. 選擇下列其中一項：
   - 若要讓 Canary 以藍圖指令碼為基礎，請選擇 Use a blueprint (使用藍圖)，然後選擇您要建立的 Canary 類型。。如需每種藍圖類型之用途的詳細資訊，請參閱使用 Canary 藍圖。
   - 若要上傳您自己的 Node.js 指令碼以建立自訂 Canary，請選擇 Upload a script (上傳指令碼)。

     然後，您可以將指令碼拖曳到 Script (指令碼) 中，或選擇 Browse files (瀏覽檔案) 以瀏覽至檔案系統中的指令碼。
   - 若要從 S3 儲存貯體匯入指令碼，請選擇 Import from S3 (從 S3 匯入)。在 Sourse location (來源位置) 下方輸入 Canary 的完整路徑，或選擇 Browse S3 (瀏覽 S3)。

     您必須擁有您使用之 S3 儲存貯體的 `s3:GetObject` 和 `s3:GetObjectVersion` 許可。值區必須位於您要建立金絲雀的相同 AWS 區域。
5. 在 Name (名稱) 下方輸入 Canary 的名稱。此名稱會用於許多頁面，因此我們建議您使用描述性的名稱，以與其他 Canary 區別。
6. 在 Application or endpoint URL (應用程式或端點 URL) 下方，輸入您要 Canary 進行測試的 URL。此 URL 必須包含通訊協定 (例如 https://)。

   如果您想要讓 Canary 在 VPC 上測試端點，您稍後在本程序中也必須輸入 VPC 的相關資訊。
7. 如果您為 Canary 使用您自己的指令碼，則請在 Lamdba handler (Lambda 處理常式) 下方輸入您希望 Canary 開始的進入點。如果您使用的執行時間早於 `syn-nodejs-puppeteer-3.4` 或 `syn-python-selenium-1.1`，則您輸入的字串必須以 `.handler` 結尾。如果您使用 `syn-`

`nodejs-puppeteer-3.4` 或者 `syn-python-selenium-1.1` 或更高版本的執行時間，則此限制不適用。

8. 如果您在指令碼中使用環境變數，選擇 Environment variables (環境變數)，然後為指令碼中定義的每個環境變數指定一個數值。如需詳細資訊，請參閱 環境變數。
9. 在 Schedule (排程) 下，選擇只執行此 Canary 一次、使用 Rate 表達式連續執行，或使用 Cron 表達式對其進行排程。

   - 當您使用 CloudWatch 控制台創建連續運行的初期測試時，您可以在每分鐘一次到每小時一次之間的任何位置選擇費率。

   - 如需撰寫 Canary 排程的 Cron 表達式的詳細資訊，請參閱 使用 Cron 排程 Canary 執行。

10. (選用) 如要設定 canary 的逾時值，請選擇 Additional configuration (其他組態)，然後指定逾時值。使其不短於 15 秒，以允許 Lambda 冷啟動和啟動 canary 儀器所需的時間。
11. 在 Data retention (資料保留) 下方，指定要將 Canary 執行失敗和成功的資訊保留多長時間。範圍是 1-455 天。

    此設定只會影響 CloudWatch Synthetics 在主控台中儲存和顯示的資料。它不會影響存放在 Amazon S3 儲存貯體中的資料，也不會影響金絲雀發佈的日誌或指標。
12. 在 Data Storage (資料儲存體) 下方，選取 S3 儲存貯體以用來儲存來自 Canary 執行的資料。儲存貯體名稱不能包含句點 (.)。如果您將此保留為空白，則會使用或建立預設的 S3 儲存貯體。

    如果您是使用 `syn-nodejs-puppeteer-3.0` 或更新版本的執行時間，當您在文字方塊中輸入儲存貯體的 URL 時，您可以在目前區域或其他區域中指定儲存貯體。如果您使用的是較早的執行時間版本，則儲存貯體必須位於當前的區域。
13. (選擇性) 根據預設，金絲雀會將其成品存放在 Amazon S3 上，並使用 AWS受管金鑰靜態加密成品 AWS KMS 。您可以使用不同的加密選項，方法是選擇 Data Storage (資料儲存) 區段中的 Additional configuration (其他組態)。然後您可以選擇要用於加密的金鑰類型。如需詳細資訊，請參閱 加密 Canary 成品。
14. 在 Access permissions (存取許可) 下方，選擇是要建立 IAM 角色來執行 Canary，還是使用現有角色。

    如果您有 CloudWatch Synthetics 創建角色，它會自動包含所有必要的權限。如果您想要自行建立角色，請參閱 Canary 的必要角色和許可 以瞭解有關必要許可的資訊。

如果您在建立初期測試時使用 CloudWatch 主控台為初期測試建立角色，則無法將該角色重複用於其他加那利群島，因為這些角色僅適用於一個初期測試。如果您已手動建立可供多個 Canary 使用的角色，則可使用現有的角色。

若要使用現有角色，您必須具有將該角色傳遞給 Synthetics 和 Lambda 的 `iam:PassRole` 許可。您也必須擁有 `iam:GetRole` 許可。

15. (選擇性) 在 [警報] 下，選擇是否要為此初期測試建立預設 CloudWatch 鬧鐘。如果您選擇建立警示，則會以下列名稱慣例建立警示：`Synthetics-Alarm-`*`canaryName`*`-`*`index`*

    `index` 是代表為此 Canary 建立的每個不同警示的數字。第一個警示的索引為 1，第二個警示的索引為 2，以此類推。

16. (選用) 若要在 VPC 上的端點執行此 Canary 測試，請選擇 VPC settings (VPC 設定)，然後執行下列操作：

    a. 選取託管端點的 VPC。

    b. 在 VPC 上選取一或多個子網路。您必須選取私有子網路，因為當 IP 地址無法在執行期間指派給 Lambda 執行個體時，則 Lambda 執行個體無法設定為在公有子網路中執行。如需詳細資訊，請參閱設定 Lambda 函數以存取 VPC 中的資源。

    c. 在 VPC 上選取一或多個安全群組。

    如果端點位於 VPC 上，則必須啟用初期測試將資訊傳送到 CloudWatch 和 Amazon S3。如需詳細資訊，請參閱 在 VPC 上執行 Canary。

17. (選用) 在 Tags (標籤) 下方，新增一或多個鍵/值對，作為此 Canary 的標籤。標籤可以幫助您識別和組織 AWS 資源並追蹤 AWS 成本。如需詳細資訊，請參閱 標記您的 Amazon CloudWatch 資源。

18. (選用) 在 Active tracing (主動追蹤) 中，選擇是否為此 Canary 啟用 X-Ray 主動追蹤。只有 Canary 使用 syn-nodejs-2.0 或更新版本的執行時間版本時，才能使用此選項。如需詳細資訊，請參閱 Canary 和 X-Ray 追蹤。

為 Canary 建立的資源

當您建立 Canary 時，會建立下列資源：

- 具有名稱的 IAM 角色`CloudWatchSyntheticsRole-`*`canary-name`*`-`*`uuid`*（如果您使用 CloudWatch 控制台建立初期測試並指定要為初期測試建立的新角色）

- 具有 CloudWatchSyntheticsPolicy-*canary-name*-*uuid* 名稱的 IAM 政策。
- 具有 cw-syn-results-*accountID*-*region* 名稱的 S3 儲存貯體。
- 具有 Synthetics-Alarm-*MyCanaryName* 名稱的警示 (如果您希望為 Canary 建立警示)
- Lambda 函數和圖層 (如果您使用藍圖來建立 Canary)。這些資源具有字首 cwsyn-*MyCanaryName*。
- CloudWatch 以名稱記錄記錄檔群組/aws/lambda/cwsyn-*MyCanaryName*-*randomId*。

使用 Canary 藍圖

本節提供有關每個 Canary 藍圖以及每個藍圖最適合工作的詳細資訊。提供的藍圖適用於下列 Canary 類型：

- 活動訊號監控
- API Canary
- 中斷的連結檢查程式
- 視覺化監控
- Canary 記錄器
- GUI 工作流程

當您使用藍圖建立初期測試時，當您填寫 CloudWatch主控台中的欄位時，頁面的指令碼編輯器區域會顯示您正在建立為 Node.js 指令碼的初期測試。您也可以在此區域編輯您的 Canary 以進一步自訂。

活動訊號監控

活動訊號指令碼會載入指定的 URL，並儲存頁面的螢幕擷取畫面和 HTTP 封存檔案 (HAR 檔案)。它們也會儲存被存取 URL 的日誌。

您可以使用 HAR 檔案來檢視有關網頁的詳細效能資料。您可以分析 Web 請求的清單，並掌握效能問題，例如項目載入的時間。

若您的 Canary 使用 syn-nodejs-puppeteer-3.1 或更新的執行時間版本，您可使用活動訊號監控藍圖來監控多個 URL，並查看 Canary 執行報告之步驟摘要中每個 URL 的狀態、持續時間、相關聯的螢幕擷取畫面和失敗原因。

API Canary

API Canary 可以測試 REST API 的基本讀取和寫入功能。REST 代表具象狀態傳輸，是開發人員在建立 API 時遵循的一組規則。其中一個規則指出，指向特定 URL 的連結應該傳回一段資料。

Canary 可以使用任何 API 並測試所有類型的功能。每個 Canary 可以進行多個 API 呼叫。

在使用 `syn-nodejs-2.2` 或更新的執行時間版本的 Canary 中，API Canary 藍圖支援多步驟 Canary，將 API 作為 HTTP 步驟進行監控。您可以在單一 Canary 中測試多個 API。每個步驟都是個別請求，可存取不同的 URL、使用不同的標頭，並使用不同的規則來確定是否擷取標頭和回應內文。藉由不擷取標頭和回應內文，您可以防止記錄敏感資料。

API Canary 中的每個請求都包含下列資訊：

- 端點，也就是您請求的 URL。
- 方法，這是傳送到伺服器的請求類型。REST API 支援 GET (讀取)、POST (寫入)、PUT (更新)、PATCH (更新) 和 DELETE (刪除) 操作。
- 標頭，同時提供資訊給用戶端和伺服器。它們用於身分驗證並提供內文內容的相關資訊。如需有效標頭的清單，請參閱 HTTP 標頭。
- 資料 (或內文)，包含要傳送到伺服器的資訊。這僅用於 POST、PUT、PATCH 或 DELETE 請求。

API Canary 藍圖支援 GET 和 POST 方法。當您使用此藍圖時，必須指定標頭。例如，您可以指定 **`Authorization`** 作為 Key (鍵)，並將必要的授權資料指定為該鍵的 Value (值)。

如果您正在測試 POST 請求，也可以在 Data (資料) 欄位中指定要發佈的內容。

與 API Gateway 的整合

API 藍圖與 Amazon API Gateway 整合。這可讓您從與初期測試相同的 AWS 帳戶和區域中選取 API Gateway API，或從 API Gateway 上傳 Swagger 範本以進行跨帳戶和跨區域 API 監控。然後，您可以在主控台中選擇其餘的詳細資訊來建立 Canary，而不是從頭開始輸入。如需 API Gateway 的詳細資訊，請參閱什麼是 Amazon API Gateway？

使用私有 API

您可以在 Amazon API Gateway 中建立使用私有 API 的 Canary。如需詳細資訊，請參閱是否在 Amazon API Gateway 中建立私有 API？

中斷的連結檢查程式

中斷連結檢查程式會使用 document.getElementsByTagName('a') 收集您正在測試之 URL 內的所有連結。它最多只會測試您指定的連結數目，而 URL 本身可計為第一個連結。例如，如果您想要檢查包含五個連結之頁面上的所有連結，您必須指定要讓 Canary 跟隨六個連結。

中斷連結檢查程式 Canary 使用 syn-nodejs-2.0-beta 執行時間或更新版本支援下列額外的功能：

- 提供一份報告，其中包含已檢查的連結、狀態碼、失敗原因 (如果有的話)，以及來源和目的地頁面螢幕擷取畫面。
- 檢視 Canary 結果時，您可以篩選僅查看中斷的連結，然後根據失敗的原因修正連結。
- 此版本會擷取每個連結的附註來源頁面螢幕擷取畫面，並反白顯示找到連結的錨點。不會對隱藏的元件對象標註。
- 您可以將此版本設定為擷取來源和目的地頁面、僅來源頁面或僅目的地頁面的螢幕擷取畫面。
- 此版本修正了舊版中的問題，亦即，即使從第一頁湊集了更多連結，Canary 指令碼也會在第一個中斷的連結後停止。

如果您想使用 syn-1.0 更新現有 Canary，以使用新的執行時間，您必須刪除並重新建立 Canary。將現有的 Canary 更新到新的執行時間不會使這些功能可用。

中斷連結檢查程式 Canary 會偵測下列類型的連結錯誤：

- 404 找不到網頁
- 無效的主機名稱
- URL 錯誤。例如，URL 缺少一個括號、有額外的斜線，或者使用錯誤的通訊協定。
- 無效的 HTTP 回應代碼。
- 主機伺服器傳回沒有內容也沒有回應代碼的空回應。
- 在 Canary 執行期間，HTTP 請求不斷逾時。
- 主機持續中斷連線，因為它設定錯誤或太忙碌。

視覺化監控藍圖

視覺化監控藍圖包含可比較在 Canary 執行期間擷取的螢幕擷取畫面與基準 Canary 執行期間擷取的螢幕擷取畫面的程式碼。如果兩個螢幕擷取畫面之間的差異超過閾值百分比，則 Canary 失敗。運行 syn-puppeteer-node-3.2 及更高版本的金絲雀支持可視化監視。目前，執行 Python 和 Selenium 的 Canary 不支援它。

視覺化監控藍圖在預設藍圖 Canary 指令碼中包含下列程式碼行，可啟用視覺化監控。

```
syntheticsConfiguration.withVisualCompareWithBaseRun(true);
```

將此行新增至指令碼之後，Canary 第一次成功執行時，它會使用該執行期間擷取的螢幕擷取畫面作為比較基準。在第一次初次測試執行之後，您可以使用 CloudWatch 主控台編輯初期測試，以執行下列任一項作業：

- 將 Canary 的下一個執行設定為新基準。
- 在目前的基準螢幕擷取畫面上繪製邊界，以指定要在視覺化比較期間忽略的螢幕擷取畫面區域。
- 移除用於視覺化監控的螢幕擷取畫面。

如需有關使用 CloudWatch 主控台編輯初期測試的詳細資訊，請參閱編輯或刪除 Canary。

您也可以使用`nextrun`或`lastrun`參數或在 UpdateCanaryAPI 中指定初期測試執行 ID，來變更用作基準線的初期測試執行。

當您使用視覺化監控藍圖時，請輸入要擷取螢幕擷取畫面的 URL，並將差異閾值指定為百分比。執行基準後，未來執行 Canary 可偵測大於該閾值的視覺化差異，進而觸發 Canary 失敗。執行基線後，您也可以編輯 Canary，在基線螢幕擷取畫面上「繪製」邊界，這些邊界要在視覺化監控期間忽略。

視覺監控功能由 ImageMagick 開放原始碼軟體工具組提供支援。如需詳細資訊，請參閱ImageMagick。

Canary 記錄器

使用初期測試記錄器藍圖，您可以使用 CloudWatch Synthetics 記錄器在網站上記錄您的點擊和輸入動作，並自動生成 Node.js 腳本，該腳本可用於創建遵循相同步驟的初期測試。S CloudWatch ynthetics 記錄器是由 Amazon 提供的谷歌瀏覽器擴展。

學分：S CloudWatch ynthetics 記錄器基於無頭記錄器。

如需詳細資訊，請參閱 使用 S CloudWatch ynthetics 記錄器的谷歌瀏覽器。

GUI 工作流程建置器

GUI 工作流程建置器藍圖會驗證是否可以在您的網頁上採取動作。例如，如果您有一個包含登入表單的網頁，Canary 可以填入使用者和密碼欄位並提交表單，以確認網頁是否正常運作。

當您使用藍圖建立此類型的 Canary 時，請指定希望 Canary 在網頁上採取的動作。您可以使用的動作如下：

- 按一下— 選取您指定的元素，並模擬使用者按一下或選擇元素。

  若要指定 Node.js 指令碼中的元素，請使用 `[id=]` 或 `a[class=]`。

  若要指定 Python 指令碼中的元素，請使用 `xpath //*[@id=]` 或 `//*[@class=]`。
- 驗證選擇器— 驗證指定的元素是否存在於網頁上。此測試有助於驗證先前的動作是否導致正確的元素填入頁面。

  若要指定要在 Node.js 指令碼中驗證的元素，請使用 `[id=]` 或 `a[class=]`。

  若要指定要在 Python 指令碼中驗證的元素，請使用 `xpath //*[@id=]` 或 `//*[class=]`。
- 驗證文字— 驗證指定的字串是否包含在目標元素中。此測試有助於驗證先前的動作是否導致顯示正確的文字。

  若要指定 Node.js 指令碼中的元素，請使用 `div[@id=]//h1` 之類的格式，因為此動作使用 Puppeteer 中的 `waitForXPath` 函數。

  若要指定 Python 指令碼中的元素，請使用 `//*[@id=]` 或 //*[@class=] 之類的 xpath 格式，因為此動作使用 Selenium 中的 `implicitly_wait` 函數。
- 輸入文字— 在目標元素中寫入指定的文字。

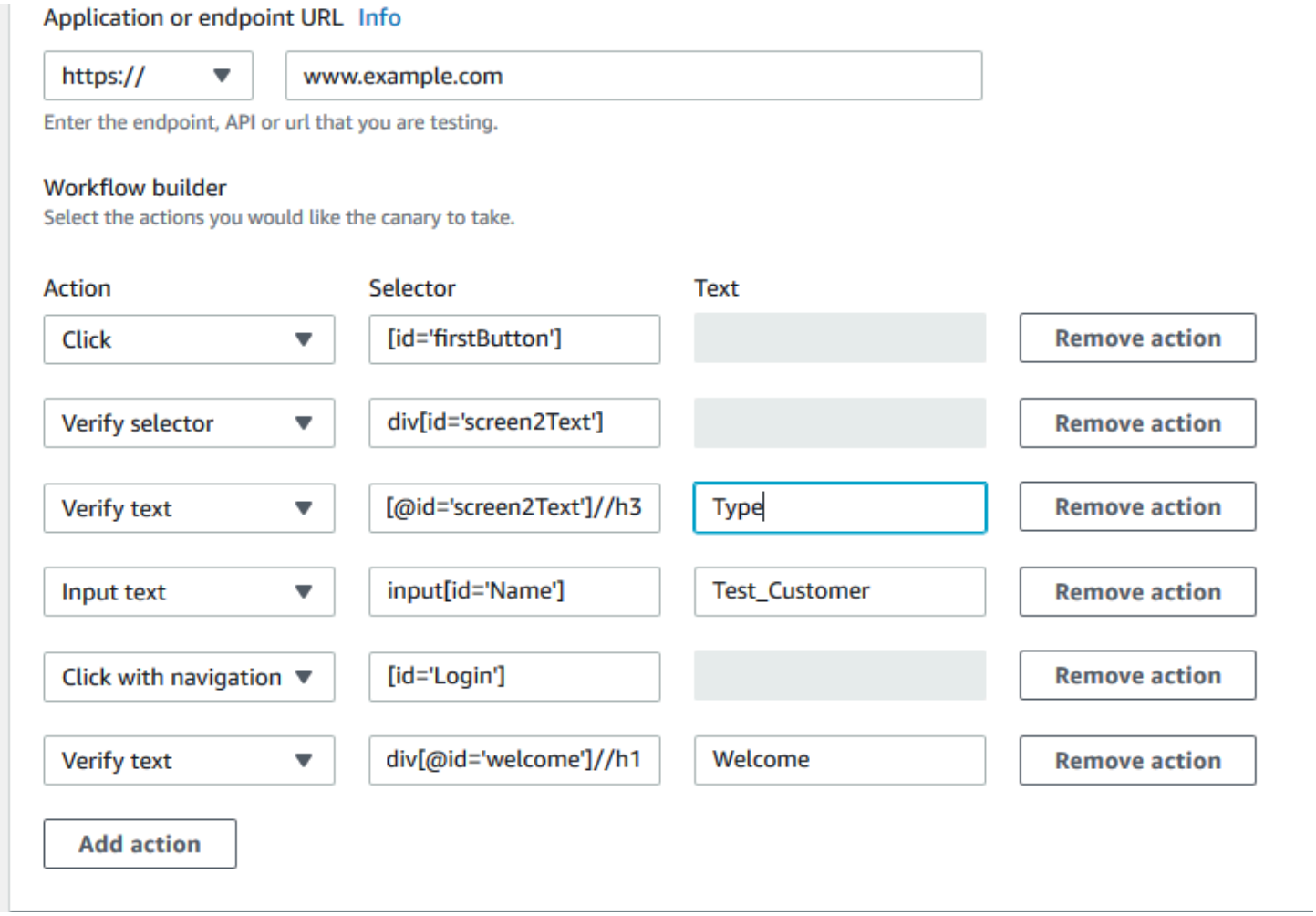
  若要指定要在 Node.js 指令碼中驗證的元素，請使用 `[id=]` 或 `a[class=]`。

  若要指定要在 Python 指令碼中驗證的元素，請使用 `xpath //*[@id=]` 或 `//*[@class=]`。
- 按一下並導覽— 選擇指定的元素後等待整個頁面載入。這很適合用於需要重新載入頁面時。

  若要指定 Node.js 指令碼中的元素，請使用 `[id=]` 或 `a[class=]`。

  若要指定 Python 指令碼中的元素，請使用 `xpath //*[@id=]` 或 `//*[@class=]`。

例如，以下藍圖使用 Node.js。其會按一下指定 URL 上的 firstButton，驗證是否出現了具有預期文字的預期選擇器，在 Name (名稱) 欄位中輸入名稱 `Test_Customer`，按一下 Login (登入) 按鈕，然後檢查下一頁的 Welcome (歡迎) 文字，驗證是否成功登入。



使用下列執行時間的 GUI 工作流程 Canary 也會提供針對每個 Canary 執行而執行的步驟摘要。您可以使用與每個步驟相關聯的螢幕擷取畫面和錯誤訊息，以尋找失敗的根本原因。

- `syn-nodejs-2.0` 或更新版本
- `syn-python-selenium-1.0` 或更新版本

使用 S CloudWatch ynthetics 記錄器的谷歌瀏覽器

Amazon 提供了 CloudWatch Synthetics 記錄器，以幫助您更輕鬆地創建金絲雀。記錄器是 Google Chrome 延伸。

記錄器記錄了您在網站上的點按和輸入動作，並自動產生 Node.js 指令碼，而該指令碼可用於建立遵循相同步驟的 Canary。

開始錄製後， CloudWatch Synthetics 記錄器會在瀏覽器中檢測到您的操作並將其轉換為腳本。您可以視需要暫停和繼續記錄。當您停止記錄時，記錄器會產生動作的 Node.js 指令碼，而您可以使用 Copy

to Clipboard (複製到剪貼簿) 按鈕輕鬆進行複製。然後，您可以使用此腳本在 CloudWatch Synthetics 中創建金絲雀。

學分：S CloudWatch ynthetics 記錄器基於無頭記錄器。

為谷歌瀏覽器安裝 CloudWatch Synthetics 記錄器擴展

要使用 S CloudWatch ynthetics 記錄器，您可以開始創建金絲雀並選擇 Can ary 記錄器藍圖。如果您在尚未下載記錄器時執行此操作，則 S CloudWatch ynthetics 控制台會提供下載鏈接。

或者，您可以依照這些步驟直接下載及安裝記錄器。

安裝 CloudWatch Synthetics 記錄器

1. 使用谷歌瀏覽器，請訪問這個網站：https://chrome.google.com/webstore/detail/cloudwatch-synthetics-rec/bhdn.com 網站轉換器
2. 選擇 Add to Chrome (新增至 Chrome)，然後選擇 Add extension (新增延伸)。

使用 S CloudWatch ynthetics 記錄器的谷歌瀏覽器

要使用 CloudWatch Synthetics 記錄器幫助您創建金絲雀，您可以在 CloudWatch 控制台中選擇「創建初期測試」，然後選擇「使用藍圖」，「Canary 記錄器」。如需詳細資訊，請參閱 建立 Canary。

或者，您可以使用記錄器來記錄步驟，而不需立即使用它們來建立 Canary。

使用 S CloudWatch ynthetics 記錄器記錄您在網站上的動作

1. 導覽至您想要監控的頁面。
2. 選擇 Chrome 擴展程序圖標，然後選擇 CloudWatchSynthetics 記錄器。
3. 選擇 Start Recording (開始記錄)。
4. 執行您想要記錄的步驟。若要暫停記錄，請選擇 Pause (暫停)。
5. 完成記錄工作流程時，選擇 Stop recording (停止錄製)。
6. 選擇 Copy to clipboard (複製到剪貼簿)，將產生的指令碼複製到剪貼簿。或者，如果您想要重新開始，請選擇 New recording (新的記錄)。
7. 若要使用複製的指令碼建立 Canary，您可以將複製的指令碼貼到記錄器藍圖內置編輯器中，或將其儲存到 Simple Storage Service (Amazon S3) 儲存貯體中並從其中匯入。
8. 如果您不是要立即建立 Canary，則可以將記錄的指令碼儲存到檔案中。

CloudWatch Synthetics 記錄器的已知限制

谷歌瀏覽器的 CloudWatch Synthetics 記錄器目前具有以下限制。

- 沒有 ID 的 HTML 元素將使用 CSS 選擇器。如果網頁結構稍後發生變更，這可能會破壞 Canary。我們計劃在未來版本的記錄器中提供一些組態選項 (例如使用 data-id)。
- 記錄器不支援按兩下或複製/貼上等動作，也不支援 CMD+0 之類的按鍵組合。
- 若要驗證頁面上是否存在元素或文字，使用者必須在產生指令碼之後新增聲明。記錄器不支援驗證元素而不對該元素執行任何動作。這類似於 Canary 工作流程建置器中的「驗證文字」或「驗證元素」選項。我們計劃在未來版本的記錄器中新增一些聲明支援。
- 記錄器會在啟動記錄的標籤中記錄所有動作。它不會記錄快顯 (例如，允許位置追蹤)，也不會從快顯導覽至不同的頁面。

Synthetics 執行時間版本

當您建立或更新 Canary 時，您需要為 Canary 選擇 Synthetics 執行時間版本。Synthetics 執行時間是 Synthetics 程式碼的組合，可呼叫指令碼處理常式，以及組合依存項目的 Lambda 層。

CloudWatch Synthetics 目前支持將 Node.js 用於腳本和木偶框架的運行時，以及使用 Python 進行腳本編寫和硒 Web 驅動程序的運行時。

建議您一律使用最新版的 Canary 執行時間，這樣才能使用最新功能和 Synthetics 程式庫的最新更新。

建立金絲雀時，其中一個建立的圖層是附加在前面加上的「Synthetics」圖層。`Synthetics`此圖層由 Synthetics 服務帳戶擁有，並包含運行時代碼。

> **Note**
>
> 每當您升級金絲雀以使用新版本的 Synthetics 執行期時，您的金絲雀使用的所有 Synthetics 程式庫函數也會自動升級到 Synthetics 執行期支援的相同版本的 NodeJS。

主題

- CloudWatch Synthetics 運行時支持策略
- 使用 Node.js 和 Puppeteer 的執行時間版本
- 執行時間版本使用 Python 和 Selenium Webdriver

CloudWatch Synthetics 運行時支持策略

Synthetics 執行時間版本受維護和安全性更新所拘束。當不再支援執行時間版本的任何元件時，該 Synthetics 執行時間版本則將遭到取代。

您無法使用已取代的執行時間版本來建立 Canary。使用已取代執行時間的 Canary 將繼續執行。您可以停止、開始和刪除這些 Canary。您可以將 Canary 更新為使用支援的執行時間版本，藉此更新使用已取代的執行時間版本的現有 Canary。

CloudWatch 如果您的金絲雀使用運行時計劃在接下來的 60 天內棄用，則 Synthetics 會通過電子郵件通知您。我們建議您將 Canary 遷移至支援的執行時間版本，以便從更新版本中包含的新功能、安全性和效能增強功能受益。

如何將 Canary 更新為新的執行時間版本？

您可以使用 CloudWatch 主控台、 AWS CloudFormation AWS CLI 或 AWS SDK 來更新初期測試的執行階段版本。使用 CloudWatch 主控台時，您可以一次更新最多五個金絲雀，方法是在初期測試清單頁面中選取它們，然後選擇 [動作] > [更新執行階段]。

您可以先使用 CloudWatch 主控台複製初期測試並更新其執行階段版本，以驗證升級。這樣一來，會建立另一個 Canary，它就是原來的 Canary 的複製。一旦您使用新的執行時間版本驗證了您的 Canary，您可以更新原始 Canary 的執行時間版本並刪除複製的 Canary。

您也可以使用升級指令碼更新多個 Canary。如需詳細資訊，請參閱 Canary 執行時間升級指令碼。

如果您升級 Canary 但失敗了，請參閱 對失敗的 Canary 進行故障診斷。

執行時間取代日期

執行時間版本	取代日期
`syn-nodejs-puppeteer-6.1`	2024年3月8日
`syn-nodejs-puppeteer-6.0`	2024年3月8日
`syn-nodejs-puppeteer-5.1`	2024年3月8日

執行時間版本	取代日期
syn-nodejs-puppeteer-5.0	2024年3月8日
syn-nodejs-puppeteer-4.0	2024年3月8日
syn-nodejs-puppeteer-3.9	2024 年 1 月 8 日
syn-nodejs-puppeteer-3.8	2024 年 1 月 8 日
syn-python-selenium-2.0	2024年3月8日
syn-python-selenium-1.3	2024年3月8日
syn-python-selenium-1.2	2024年3月8日
syn-python-selenium-1.1	2024年3月8日
syn-python-selenium-1.0	2024年3月8日
syn-nodejs-puppeteer-3.7	2024 年 1 月 8 日
syn-nodejs-puppeteer-3.6	2024 年 1 月 8 日
syn-nodejs-puppeteer-3.5	2024 年 1 月 8 日
syn-nodejs-puppeteer-3.4	2022 年 11 月 13 日

執行時間版本	取代日期
syn-nodejs-puppeteer-3.3	2022 年 11 月 13 日
syn-nodejs-puppeteer-3.2	2022 年 11 月 13 日
syn-nodejs-puppeteer-3.1	2022 年 11 月 13 日
syn-nodejs-puppeteer-3.0	2022 年 11 月 13 日
syn-nodejs-2.2	2021 年 5 月 28 日
syn-nodejs-2.1	2021 年 5 月 28 日
syn-nodejs-2.0	2021 年 5 月 28 日
syn-nodejs-2.0-beta	2021 年 2 月 8 日
syn-1.0	2021 年 5 月 28 日

Canary 執行時間升級指令碼

若要將 Canary 指令碼升級為支援的執行時間版本，請使用下列指令碼。

```
const AWS = require('aws-sdk');

// You need to configure your AWS credentials and Region.
//   https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/setting-credentials-node.html
//   https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/setting-region.html

const synthetics = new AWS.Synthetics();

const DEFAULT_OPTIONS = {
  /**
```

```
    * The number of canaries to upgrade during a single run of this script.
    */
  count: 10,
  /**
   * No canaries are upgraded unless force is specified.
   */
  force: false
};

/**
 * The number of milliseconds to sleep between GetCanary calls when
 * verifying that an update succeeded.
 */
const SLEEP_TIME = 5000;

(async () => {
  try {
    const options = getOptions();

    const versions = await getRuntimeVersions();
    const canaries = await getAllCanaries();
    const upgrades = canaries
      .filter(canary => !versions.isLatestVersion(canary.RuntimeVersion))
      .map(canary => {
        return {
          Name: canary.Name,
          FromVersion: canary.RuntimeVersion,
          ToVersion: versions.getLatestVersion(canary.RuntimeVersion)
        };
      });

    if (options.force) {
      const promises = [];

      for (const upgrade of upgrades.slice(0, options.count)) {
        const promise = upgradeCanary(upgrade);
        promises.push(promise);
        // Sleep for 100 milliseconds to avoid throttling.
        await usleep(100);
      }

      const succeeded = [];
      const failed = [];
      for (let i = 0; i < upgrades.slice(0, options.count).length; i++) {
```

```
        const upgrade = upgrades[i];
        const promise = promises[i];
        try {
          await promise;
          console.log(`The update of ${upgrade.Name} succeeded.`);
          succeeded.push(upgrade.Name);
        } catch (e) {
          console.log(`The update of ${upgrade.Name} failed with error: ${e}`);
          failed.push({
            Name: upgrade.Name,
            Reason: e
          });
        }
      }

      if (succeeded.length) {
        console.group('The following canaries were upgraded successfully.');
        for (const name of succeeded) {
          console.log(name);
        }
        console.groupEnd()
      } else {
        console.log('No canaries were upgraded successfully.');
      }

      if (failed.length) {
        console.group('The following canaries were not upgraded successfully.');
        for (const failure of failed) {
          console.log('\x1b[31m', `${failure.Name}: ${failure.Reason}`, '\x1b[0m');
        }
        console.groupEnd();
      }
    } else {
      console.log('Run with --force [--count <count>] to perform the first <count>
 upgrades shown. The default value of <count> is 10.')
      console.table(upgrades);
    }
  } catch (e) {
    console.error(e);
  }
})();

function getOptions() {
  const force = getFlag('--force', DEFAULT_OPTIONS.force);
```

```
  const count = getOption('--count', DEFAULT_OPTIONS.count);
  return { force, count };

  function getFlag(key, defaultValue) {
    return process.argv.includes(key) || defaultValue;
  }
  function getOption(key, defaultValue) {
    const index = process.argv.indexOf(key);
    if (index < 0) {
      return defaultValue;
    }
    const value = process.argv[index + 1];
    if (typeof value === 'undefined' || value.startsWith('-')) {
      throw `The ${key} option requires a value.`;
    }
    return value;
  }
}

function getAllCanaries() {
  return new Promise((resolve, reject) => {
    const canaries = [];

    synthetics.describeCanaries().eachPage((err, data) => {
      if (err) {
        reject(err);
      } else {
        if (data === null) {
          resolve(canaries);
        } else {
          canaries.push(...data.Canaries);
        }
      }
    });
  });
}

function getRuntimeVersions() {
  return new Promise((resolve, reject) => {
    const jsVersions = [];
    const pythonVersions = [];
    synthetics.describeRuntimeVersions().eachPage((err, data) => {
      if (err) {
        reject(err);
```

```
      } else {
        if (data === null) {
          jsVersions.sort((a, b) => a.ReleaseDate - b.ReleaseDate);
          pythonVersions.sort((a, b) => a.ReleaseDate - b.ReleaseDate);
          resolve({
            isLatestVersion(version) {
              const latest = this.getLatestVersion(version);
              return latest === version;
            },
            getLatestVersion(version) {
              if (jsVersions.some(v => v.VersionName === version)) {
                return jsVersions[jsVersions.length - 1].VersionName;
              } else if (pythonVersions.some(v => v.VersionName === version)) {
                return pythonVersions[pythonVersions.length - 1].VersionName;
              } else {
                throw Error(`Unknown version ${version}`);
              }
            }
          });
        } else {
          for (const version of data.RuntimeVersions) {
            if (version.VersionName === 'syn-1.0') {
              jsVersions.push(version);
            } else if (version.VersionName.startsWith('syn-nodejs-2.')) {
              jsVersions.push(version);
            } else if (version.VersionName.startsWith('syn-nodejs-puppeteer-')) {
              jsVersions.push(version);
            } else if (version.VersionName.startsWith('syn-python-selenium-')) {
              pythonVersions.push(version);
            } else {
              throw Error(`Unknown version ${version.VersionName}`);
            }
          }
        }
      }
    });
  });
}

async function upgradeCanary(upgrade) {
  console.log(`Upgrading canary ${upgrade.Name} from ${upgrade.FromVersion} to
 ${upgrade.ToVersion}`);
  await synthetics.updateCanary({ Name: upgrade.Name, RuntimeVersion:
 upgrade.ToVersion }).promise();
```

```
  while (true) {
    await usleep(SLEEP_TIME);
    console.log(`Getting the state of canary ${upgrade.Name}`);
    const response = await synthetics.getCanary({ Name: upgrade.Name }).promise();
    const state = response.Canary.Status.State;
    console.log(`The state of canary ${upgrade.Name} is ${state}`);
    if (state === 'ERROR' || response.Canary.Status.StateReason) {
      throw response.Canary.Status.StateReason;
    }
    if (state !== 'UPDATING') {
      return;
    }
  }
}

function usleep(ms) {
  return new Promise(resolve => setTimeout(resolve, ms));
}
```

使用 Node.js 和 Puppeteer 的執行時間版本

Node.js 和 Puppeteer 的第一個執行時間版本名稱為 `syn-1.0`。較新的執行時間版本具有命名慣例 `syn-`*`language`*`-`*`majorversion`*`.`*`minorversion`*。從 `syn-nodejs-puppeteer-3.0` 開始，命名慣例為 `syn-`*`language`*`-`*`framework`*`-`*`majorversion`*`.`*`minorversion`*

額外 `-beta` 尾碼顯示執行時間版本目前處於 Beta 預覽版。

具有相同主要版本編號的執行時間版本可回溯相容。

> **Important**
>
> 下列 CloudWatch Synthetics 執行階段版本排定於 2024 年 3 月 8 日淘汰。
>
> - `syn-nodejs-puppeteer-6.1`
> - `syn-nodejs-puppeteer-6.0`
> - `syn-nodejs-puppeteer-5.1`
> - `syn-nodejs-puppeteer-5.0`
> - `syn-nodejs-puppeteer-4.0`
>
> 如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

> **Important**
>
> 重要提示：隨附的 JavaScript v2 依賴性 AWS SDK 將被刪除並更新，以在 future 的運行時發行版本中為 JavaScript v3 使用 AWS SDK。發生這種情況時，您可以更新 Canary 程式碼引用。或者，您可以通過將其作為依賴項添加到源代碼 zip 文件中，繼續引用並使用隨附的 AWS SDK 進行 JavaScript v2 依賴項。

所有執行時間版本備註

當使用 `syn-nodejs-puppeteer-3.0` 執行時間版本中，確定 Canary 指令碼與 Node.js 12.x 相容。如果您使用舊版的 `syn-nodejs` 執行時間版本，確定指令碼與 Node.js 10.x 相容。

Canary 中 Lambda 程式碼的記憶體上限會設定為 1 GB。Canary 的每次執行都會在設定的逾時值後逾時。如果未指定初期測試的逾時值，請根據初期測試的頻率 CloudWatch 選擇逾時值。若您設定逾時值，請使其不短於 15 秒，以允許 Lambda 冷啟動和啟動 canary 儀器所需的時間。

> **Note**
>
> 以下 CloudWatch Synthetics 執行階段版本已於 2024 年 1 月 8 日棄用。這是因為在 2023 年 12 月 4 日 AWS Lambda 棄用了 Lambda Node.js 14 執行階段。
>
> - `syn-nodejs-puppeteer-3.9`
> - `syn-nodejs-puppeteer-3.8`
> - `syn-nodejs-puppeteer-3.7`
> - `syn-nodejs-puppeteer-3.6`
> - `syn-nodejs-puppeteer-3.5`
>
> 以下 CloudWatch Synthetics 執行階段版本已於 2022 年 11 月 13 日棄用。這是因為在 2022 年 11 月 14 日 AWS Lambda 棄用了 Lambda Node.js 12 執行階段。
>
> - `syn-nodejs-puppeteer-3.4`
> - `syn-nodejs-puppeteer-3.3`
> - `syn-nodejs-puppeteer-3.2`
> - `syn-nodejs-puppeteer-3.1`
> - `syn-nodejs-puppeteer-3.0`

如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

syn-nodejs-puppeteer-7.0

`syn-nodejs-puppeteer-7.0`執行階段是 Lambda 執行階段 Node.js 18.x 的最新執行階段版本。它使用 Node.js 和木偶工具。

主要相依性：

- Lambda 執行時間 Node.js 18.x
- 木偶核心版本 21.9.0
- 鉻的版本

代碼大小：

您可以封裝到此執行階段中的程式碼和相依性大小為 80 MB。

syn-nodejs-puppeteer-7.0 中的新功能：

- 木偶和鉻中捆綁的庫的更新版本-木偶器和鉻依賴項已更新為新版本。

> **Important**
>
> 從木偶師 19.7.0 移至木偶師 21.9.0 引入了有關測試和過濾器的重大變化。如需詳細資訊，請參閱操偶工具：v20.0.0 和傀儡核心：v21.0.0 中的「突破變更」區段。

建議升級至 AWS SDK v3

Lambda 節點 18.x 執行階段不支援 SDK v2。 AWS 我們強烈建議您移轉至 AWS SDK v3。

syn-nodejs-puppeteer-6.2

主要相依性：

- Lambda 執行時間 Node.js 18.x
- Puppeteer-core 19.7.0 版

- Chromium 111.0.5563.146 版

syn-nodejs-puppeteer-6.2 中的新功能：

- 銘捆綁庫的更新版本
- 暫時性儲存監控 — 此執行階段會在客戶帳戶中新增暫時儲存監控。
- 錯誤修正

syn-nodejs-puppeteer-5.2

`syn-nodejs-puppeteer-5.2`執行階段是 Lambda 執行階段 Node.js 16.x 的最新執行階段版本。它使用 Node.js 和木偶工具。

主要相依性：

- Lambda 執行階段 Node.js 16.x
- Puppeteer-core 19.7.0 版
- Chromium 111.0.5563.146 版

syn-nodejs-puppeteer-5.2 中的新功能：

- 銘捆綁庫的更新版本
- 錯誤修正

syn-nodejs-puppeteer-6.1

> **Important**
>
> 此執行階段版本排定於 2024 年 3 月 8 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行時間 Node.js 18.x
- Puppeteer-core 19.7.0 版
- Chromium 111.0.5563.146 版

syn-nodejs-puppeteer-6.1 中的新功能：

- 穩定性改進 – 新增用於處理間歇性 Puppeteer 啟動錯誤的自動重試邏輯。
- 相依性升級 – 升級某些第三方相依套件。
- 沒有 Amazon S3 許可的 Canaries – 錯誤修正，使得沒有任何 Amazon S3 許可的 Canary 仍然可以執行。這些沒有 Amazon S3 許可的 Canary 將無法將螢幕擷取畫面或其他成品上傳到 Amazon S3。如需有關 Canaries 的許可詳細資訊，請參閱 Canary 的必要角色和許可。

> **Important**
>
> 重要提示：隨附的 JavaScript v2 依賴性 AWS SDK 將被刪除並更新，以在 future 的運行時發行版本中為 JavaScript v3 使用 AWS SDK。發生這種情況時，您可以更新 Canary 程式碼引用。或者，您可以通過將其作為依賴項添加到源代碼 zip 文件中，繼續引用並使用隨附的 AWS SDK 進行 JavaScript v2 依賴項。

syn-nodejs-puppeteer-6.0

> **Important**
>
> 此執行階段版本排定於 2024 年 3 月 8 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行時間 Node.js 18.x
- Puppeteer-core 19.7.0 版
- Chromium 111.0.5563.146 版

syn-nodejs-puppeteer-6.0 中的新功能：

- 相依性升級 — Node.js 相依性已更新至 18.x。
- 攔截模式支援 – Puppeteer 合作攔截模式支援已新增至 Synthetics 金絲雀執行期程式庫。
- 追蹤行為變更 – 已變更預設追蹤行為，僅追蹤擷取和 xhr 請求，而不追蹤資源請求。您可設定 `traceResourceRequests` 選項來啟用資源請求追蹤。

- 精細化持續時間量度 — 指標現在會排除初期`Duration`測試用來上傳成品、擷取螢幕擷取畫面和產生 CloudWatch 指標的作業時間。 `Duration`度量值報告給 CloudWatch，您也可以在 Synthetics 控制台中看到它們。
- 錯誤修復 – 清理 Chromium 在金絲雀部署執行期間崩潰時產生的核心傾印。

> **Important**
>
> 重要提示：隨附的 JavaScript v2 依賴性 AWS SDK 將被刪除並更新，以在 future 的運行時發行版本中為 JavaScript v3 使用 AWS SDK。發生這種情況時，您可以更新 Canary 程式碼引用。或者，您可以通過將其作為依賴項添加到源代碼 zip 文件中，繼續引用並使用隨附的 AWS SDK 進行 JavaScript v2 依賴項。

syn-nodejs-puppeteer-5.1

> **Important**
>
> 此執行階段版本排定於 2024 年 3 月 8 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行階段 Node.js 16.x
- Puppeteer-core 19.7.0 版
- Chromium 111.0.5563.146 版

syn-nodejs-puppeteer-5.1 中的錯誤修正：

- 錯誤修復 - 此執行期修復了 `syn-nodejs-puppeteer-5.0` 中的一個錯誤，即 Canary 建立的 HAR 檔案缺少請求標頭。

syn-nodejs-puppeteer-5.0

> **Important**
>
> 此執行階段版本排定於 2024 年 3 月 8 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行階段 Node.js 16.x
- Puppeteer-core 19.7.0 版
- Chromium 111.0.5563.146 版

syn-nodejs-puppeteer-5.0 中的新功能：

- 相依性升級 — Puppeteer-core 版本已更新至 19.7.0。Chromium 版本已升級至 111.0.5563.146。

> **Important**
>
> 新 Puppeteer-core 版本與舊版 Puppeteer 不完全回溯相容。此版本中的部分變更可能會導致使用已移除 Puppeteer 函數的現有 Canary 失敗。如需詳細資訊，請在 Puppeteer change logs 中，參閱 Puppeteer-core 19.7.0 至 6.0 版變更日誌中的重大變更。

syn-nodejs-puppeteer-4.0

> **Important**
>
> 此執行階段版本排定於 2024 年 3 月 8 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行階段 Node.js 16.x
- Puppeteer-core 5.5.0 版

- Chromium 92.0.4512 版

syn-nodejs-puppeteer-4.0 中的新功能：

- 相依性升級 — Node.js 相依性已更新至 16.x。

Node.js 和木偶工具已淘汰的執行階段

Node.js 和木偶工具的下列執行階段已被取代。

syn-nodejs-puppeteer-3.9

> **Important**
> 此執行階段版本已於 2024 年 1 月 8 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行時間 Node.js 14.x
- Puppeteer-core 5.5.0 版
- Chromium 92.0.4512 版

syn-nodejs-puppeteer-3.9 中的新功能：

- 相依性升級 — 升級某些第三方相依套件。

syn-nodejs-puppeteer-3.8

> **Important**
> 此執行階段版本已於 2024 年 1 月 8 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行時間 Node.js 14.x
- Puppeteer-core 5.5.0 版
- Chromium 92.0.4512 版

syn-nodejs-puppeteer-3.8 中的新功能：

- 設定檔清理：現在每次 Canary 執行後，都會清理 Chromium 設定檔。

syn-nodejs-puppeteer-3.8 中的錯誤修正：

- 錯誤修正：以前，在沒有螢幕擷取畫面的情況下，視覺化監控 Canary 有時會在執行後停止正常工作。現在已修正。

syn-nodejs-puppeteer-3.7

> **Important**
>
> 此執行階段版本已於 2024 年 1 月 8 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行時間 Node.js 14.x
- Puppeteer-core 5.5.0 版
- Chromium 92.0.4512 版

syn-nodejs-puppeteer-3.7 中的新功能：

- 記錄增強功能：即使逾時或當機，Canary 也會將日誌上傳到 Amazon S3。
- Lambda 層大小減少：用於 Canary 的 Lambda 層大小減少 34%。

syn-nodejs-puppeteer-3.7 中的錯誤修正：

- 錯誤修正：日文、簡體中文和繁體中文字型會正確顯示。

syn-nodejs-puppeteer-3.6

> **Important**
>
> 此執行階段版本已於 2024 年 1 月 8 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行時間 Node.js 14.x
- Puppeteer-core 5.5.0 版
- Chromium 92.0.4512 版

syn-nodejs-puppeteer-3.6 中的新功能：

- 更精確的時間戳記：金絲雀執行的開始時間和停止時間現在已精確至毫秒。

syn-nodejs-puppeteer-3.5

> **Important**
>
> 此執行階段版本已於 2024 年 1 月 8 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行時間 Node.js 14.x
- Puppeteer-core 5.5.0 版
- Chromium 92.0.4512 版

syn-nodejs-puppeteer-3.5 中的新功能：

- 更新相依項 – 此執行時間中唯一的新功能是更新的相依項。

syn-nodejs-puppeteer-3.4

> **Important**
>
> 此執行時間版本已於 2022 年 11 月 13 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行時間 Node.js 12.x
- Puppeteer-core 5.5.0 版
- Chromium 88.0.4298.0 版

syn-nodejs-puppeteer-3.4 的新功能：

- 自訂處理常式函數 – 您現在可以為 Canary 指令碼使用自訂處理常式函數。先前的執行時間要求指令碼進入點包含 `.handler`。

  您還可以將 Canary 指令碼放在任何資料夾中，並將資料夾名稱作為處理常式的一部分傳遞。例如：`MyFolder/MyScriptFile.functionname` 可用作進入點。
- 擴展的 HAR 檔案資訊 – 您現在可以在 Canary 產生的 HAR 檔案中看到錯誤、待處理和不完整的請求。

syn-nodejs-puppeteer-3.3

> **Important**
>
> 此執行時間版本已於 2022 年 11 月 13 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行時間 Node.js 12.x
- Puppeteer-core 5.5.0 版
- Chromium 88.0.4298.0 版

syn-nodejs-puppeteer-3.3 的新功能：

- 更多成品加密選項 — 對於使用此執行階段或更新版本的金絲雀，您可以選擇使用 AWS KMS 客戶 AWS 受管金鑰或 Amazon S3 受管金鑰，而不是使用受管金鑰來加密 Canary 存放在 Amazon S3 中的成品。如需詳細資訊，請參閱 加密 Canary 成品。

syn-nodejs-puppeteer-3.2

> **Important**
>
> 此執行時間版本已於 2022 年 11 月 13 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行時間 Node.js 12.x
- Puppeteer-core 5.5.0 版
- Chromium 88.0.4298.0 版

syn-nodejs-puppeteer-3.2 中的新功能：

- 使用螢幕截圖進行視覺化監控— 使用此執行時間或更新版本的 Canary 可以將執行期間擷取的螢幕擷取畫面與相同螢幕擷取畫面的基準版本相比較。如果螢幕擷取畫面比指定的百分比閾值更大，則 Canary 失敗。如需詳細資訊，請參閱 視覺化監控 或 視覺化監控藍圖 。
- 關於敏感資料的新功能您可以防止敏感資料出現在 Canary 日誌和報告中。如需詳細資訊，請參閱 SyntheticsLogHelper 類。
- 已取代的函數 為支援其他組態選項，`RequestResponseLogHelper` 類別已被取代。如需詳細資訊，請參閱 RequestResponseLogHelper 類。

syn-nodejs-puppeteer-3.1

> **Important**
>
> 此執行時間版本已於 2022 年 11 月 13 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行時間 Node.js 12.x
- Puppeteer-core 5.5.0 版
- Chromium 88.0.4298.0 版

syn-nodejs-puppeteer-3.1 中的新功能：

- 設定 CloudWatch 度量的能力 — 使用此執行階段，您可以停用不需要的指標。否則，金絲雀會為每次金絲雀運行發布各種 CloudWatch 指標。
- 螢幕擷取畫面連結— 您可以在步驟完成後將螢幕擷取畫面連結至 Canary 步驟。為此，您可以使用 takeScreenshot 方法擷取螢幕擷取畫面，並使用您想要與螢幕擷取畫面關聯的步驟名稱。例如，您可能想要執行步驟、新增等待時間，然後擷取螢幕擷取畫面。
- 活動訊號監視器藍圖可以監視多個 URL — 您可以使用 CloudWatch 主控台中的活動訊號監視藍圖來監視多個 URL，並在初期測試執行報告的步驟摘要中查看每個 URL 的狀態、持續時間、相關螢幕擷取畫面和失敗原因。

syn-nodejs-puppeteer-3.0

> **Important**
>
> 此執行時間版本已於 2022 年 11 月 13 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行時間 Node.js 12.x
- Puppeteer-core 5.5.0 版
- Chromium 88.0.4298.0 版

syn-nodejs-puppeteer-3.0 中的新功能：

- 升級的相依性— 此執行時間版本使用 Puppeteer 5.5.0 版、Node.js 12.x 和 Chromium 88.0.4298.0 版

- 跨區域儲存貯體存取— 您現在可以將另一個區域中 S3 儲存貯體指定為 Canary 存放其日誌檔案、螢幕擷取畫面和 HAR 檔案的儲存貯體。
- 可用的新函數— 此版本會新增程式庫函數，以擷取 Canary 名稱和 Synthetics 執行時間版本。

  如需詳細資訊，請參閱 Synthetics 類別。

syn-nodejs-2.2

本節包含 `syn-nodejs-2.2` 執行時間版本的相關資訊。

> **Important**
>
> 此執行時間版本已於 2021 年 5 月 28 日被棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行時間 Node.js 10.x
- Puppeteer-core 3.3.0 版
- Chromium 83.0.4103.0 版

syn-nodejs-2.2 中的新功能：

- 將您的 Canary 作為 HTTP 步驟進行監控— 您現在可以在單一 Canary 中測試多個 API。每個 API 都作為一個單獨的 HTTP 步驟進行測試， CloudWatch Synthetics 使用步驟指標和 S CloudWatch ynthetics 步驟報告監控每個步驟的狀態。 CloudWatch Synthetics 為每個 HTTP 步驟創建`SuccessPercent`和`Duration`指標。

  此功能是由 executeHttpStep（stepName，請求選項，回調，步驟配置）函數來實現。如需詳細資訊，請參閱 executeHttpStep（stepName，請求選項，[回調]，[步驟配置]）。

  API Canary 藍圖已更新以使用此新功能。
- HTTP 請求報告— 您現在可以檢視詳細的 HTTP 請求報告，這些報告會擷取詳細資訊，例如請求/回應標頭、回應內文、狀態碼、錯誤和效能計時、TCP 連線時間、TLS 交握時間、第一個位元組時間和內容傳輸時間。此處可擷取所有實際使用 HTTP/HTTPS 模組的 HTTP 請求。預設情況下不會擷取標頭和回應正文，但可以透過設定組態選項來將其啟用。

- 全局和步驟級配置-您可以在全局級別設置 CloudWatch Synthetics 配置，該配置應用於金絲雀的所有步驟。您也可以透過傳遞組態鍵/值對來啟用或停用某些選項，在步驟層級覆寫這些組態。

  如需詳細資訊，請參閱 SyntheticsConfiguration 類。
- 繼續步驟失敗組態— 您可以選擇在步驟失敗時繼續 Canary 執行。對於 `executeHttpStep` 函數，預設為開啟。您可以在全域層級設定此選項一次，也可以針對每個步驟設定不同的選項。

syn-nodejs-2.1

> **Important**
>
> 此執行時間版本已於 2021 年 5 月 28 日被棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行時間 Node.js 10.x
- Puppeteer-core 3.3.0 版
- Chromium 83.0.4103.0 版

syn-nodejs-2.1 中的新功能：

- 可設定的螢幕擷取畫面行為— 提供關閉 UI Canary 擷取螢幕擷取畫面的能力。在使用舊版執行時間的 Canary 中，UI Canary 總會在每個步驟之前和之後擷取螢幕擷取畫面。利用 `syn-nodejs-2.1`，這是可設定的。關閉螢幕擷取畫面可降低 Simple Storage Service (Amazon S3) 儲存成本，並可協助您遵守 HIPAA 法規。如需詳細資訊，請參閱 SyntheticsConfiguration 類。
- 自訂 Google Chrome 啟動參數 您現在可以設定當 Canary 啟動 Google Chrome 瀏覽器視窗時使用的引數。如需詳細資訊，請參閱 launch(options)。

相較於舊版的 Canary 執行時間，使用 syn-nodejs-2.0 或更新版本時，Canary 持續時間可能會稍微增加。

syn-nodejs-2.0

> **Important**
>
> 此執行時間版本已於 2021 年 5 月 28 日被棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行時間 Node.js 10.x
- Puppeteer-core 3.3.0 版
- Chromium 83.0.4103.0 版

syn-nodejs-2.0 中的新功能：

- 升級的相依性— 此執行時間版本使用 Puppeteer-core 3.3.0 版和 Chromium 83.0.4103.0 版
- 支援 X-Ray 主動追蹤 當初期測試已啟用追蹤時，系統會針對使用瀏覽器、 AWS SDK 或 HTTP 或 HTTPS 模組的初期測試所發出的所有呼叫傳送 X-Ray 追蹤。已啟用追蹤的 Canary 會出現在 X-Ray 追蹤地圖中，即使它們未傳送請求至已啟用追蹤的其他服務或應用程式。如需詳細資訊，請參閱 Canary 和 X-Ray 追蹤。
- 合成報告 — 對於每次初期測試運行， CloudWatch Synthetics 都會創建一個名為`SyntheticsReport-PASSED.json`或`SyntheticsReport-FAILED.json`記錄諸如開始時間，結束時間，狀態和故障之類的數據的報告。它還記錄了 Canary 指令碼的每個步驟的通過/失敗狀態，以及每個步驟捕獲的失敗和螢幕擷取畫面。
- 中斷連接檢查程式報告— 此執行時間中包含的新版中斷連結檢查程式會建立一份報告，其中包含已檢查的連結、狀態碼、失敗原因 (如有)，以及來源和目的地頁面螢幕擷取畫面。
- 新 CloudWatch 量度 — Synthetics 會在命名`CloudWatchSynthetics`空間中發佈名為`2xx`、`4xx`、`5xx`、和`RequestFailed`的量度。這些指標顯示 Canary 執行中的 200 秒、400 秒、500 秒和請求失敗次數。使用此執行時間版本時，這些指標只會針對 UI Canary 報告，而不會針對 API Canary 報告。它們也報告了從執行時間版本開始的 API Canary`syn-nodejs-puppeteeer-2.2`。
- 可排序的 HAR 檔案— 您現在可以依狀態碼、請求大小和持續時間來排序 HAR 檔案。
- 指標時間戳記 — 現在會根據 Lambda 叫用時間而非初期 CloudWatch 測試執行結束時間來報告指標。

syn-nodejs-2.0 中的錯誤修正：

- 修正 Canary Artifact 上傳錯誤無法回報的問題。這些錯誤現在顯示為執行錯誤。
- 修正重新引導請求 (3xx) 被錯誤記錄為錯誤的問題。
- 修正螢幕擷取畫面從 0 開始編號的問題。他們現在應該從 1 開始。
- 修正中文和日文字體螢幕擷取畫面亂碼的問題。

相較於舊版的 Canary 執行時間，使用 syn-nodejs-2.0 或更新版本時，Canary 持續時間可能會稍微增加。

syn-nodejs-2.0-beta

> **Important**
>
> 此執行時間版本已於 2021 年 2 月 8 日被取代。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Lambda 執行時間 Node.js 10.x
- Puppeteer-core 3.3.0 版
- Chromium 83.0.4103.0 版

syn-nodejs-2.0-beta 中的新功能：

- 升級的相依性— 此執行時間版本使用 Puppeteer-core 3.3.0 版和 Chromium 83.0.4103.0 版
- 合成報告 — 對於每次初期測試運行， CloudWatch Synthetics 都會創建一個名為`SyntheticsReport-PASSED.json`或`SyntheticsReport-FAILED.json`記錄諸如開始時間，結束時間，狀態和故障之類的數據的報告。它還記錄了 Canary 指令碼的每個步驟的通過/失敗狀態，以及每個步驟捕獲的失敗和螢幕擷取畫面。
- 中斷連接檢查程式報告— 此執行時間中包含的新版中斷連結檢查程式會建立一份報告，其中包含已檢查的連結、狀態碼、失敗原因 (如有)，以及來源和目的地頁面螢幕擷取畫面。
- 新 CloudWatch 量度 — Synthetics 會在命名`CloudWatchSynthetics`空間中發佈名為`2xx`、`4xx`、`5xx`和`RequestFailed`的量度。這些指標顯示 Canary 執行中的 200 秒、400 秒、500 秒和請求失敗次數。這些指標只會針對 UI Canary 報告，而不會針對 API Canary 報告。

- 可排序的 HAR 檔案— 您現在可以依狀態碼、請求大小和持續時間來排序 HAR 檔案。
- 指標時間戳記 — 現在會根據 Lambda 叫用時間而非初期 CloudWatch 測試執行結束時間來報告指標。

syn-nodejs-2.0-beta 中的錯誤修正：

- 修正 Canary Artifact 上傳錯誤無法回報的問題。這些錯誤現在顯示為執行錯誤。
- 修正重新引導請求 (3xx) 被錯誤記錄為錯誤的問題。
- 修正螢幕擷取畫面從 0 開始編號的問題。他們現在應該從 1 開始。
- 修正中文和日文字體螢幕擷取畫面亂碼的問題。

syn-1.0

> **Important**
>
> 此執行時間版本計劃於 2021 年 5 月 28 日被取代。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

第一個 Synthetics 執行時間版本是 `syn-1.0`。

主要相依性：

- Lambda 執行時間 Node.js 10.x
- Puppeteer-core 1.14.0 版
- 符合 Puppeteer-core 1.14.0 的 Chromium 版本

執行時間版本使用 Python 和 Selenium Webdriver

以下部分包含有關 CloudWatch Synthetics 料運行時版本的 Python 和硒的網絡驅動程序的信息。Selenium 是一種開放原始碼瀏覽器自動化工具。如需 Selenium 的詳細資訊，請參閱 www.selenium.dev/

這些執行時間版本的命名慣例為 `syn-`*`language`*`-`*`framework`*`-`*`majorversion`*`.`*`minorversion`*。

> **Important**
>
> 下列 CloudWatch Synthetics 執行階段版本排定於 2024 年 3 月 8 日淘汰。
>
> - `syn-python-selenium-2.0`
> - `syn-python-selenium-1.3`
> - `syn-python-selenium-1.2`
> - `syn-python-selenium-1.1`
> - `syn-python-selenium-1.0`
>
> 如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

syn-python-selenium-3.0

3.0 版本是 Python 和硒的最新 CloudWatch Synthetics 運行時間。

主要相依性：

- Python 3.8
- 硒
- 鉻的版本

syn-python-selenium-3.0 中的新功能：

- 鉻捆綁庫的更新版本-鉻依賴關係更新為新版本。

syn-python-selenium-2.1

主要相依性：

- Python 3.8
- 硒
- Chromium 111.0.5563.146 版

syn-python-selenium-2.1 的新功能：

- 鉻捆綁庫的更新版本-鉻和硒的依賴關係更新為新版本。

syn-python-selenium-2.0

> **Important**
> 此執行階段版本排定於 2024 年 3 月 8 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Python 3.8
- Selenium 4.10.0
- Chromium 111.0.5563.146 版

syn-python-selenium-2.0 中的新功能：

- 更新的依賴關係— Chromium 和 Selenium 的依賴關係更新為新版本。

syn-python-selenium-2.0 中的錯誤修正：

- 加入了時間戳記 — 時間戳記已新增至金絲雀日誌。
- 工作階段重複使用 — 修正了一個錯誤，因此金絲雀現在無法重複使用先前的金絲雀執行中的工作階段。

syn-python-selenium-1.3

> **Important**
> 此執行階段版本排定於 2024 年 3 月 8 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Python 3.8

- Selenium 3.141.0
- Chromium 92.0.4512.0 版

syn-python-selenium-1.3 中的新功能：

- 更精確的時間戳記：金絲雀執行的開始時間和停止時間現在已精確至毫秒。

syn-python-selenium-1.2

> **Important**
>
> 此執行階段版本排定於 2024 年 3 月 8 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Python 3.8
- Selenium 3.141.0
- Chromium 92.0.4512.0 版

- 更新相依項 – 此執行時間中唯一的新功能是更新的相依項。

syn-python-selenium-1.1

> **Important**
>
> 此執行階段版本排定於 2024 年 3 月 8 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Python 3.8
- Selenium 3.141.0
- Chromium 83.0.4103.0 版

功能：

- 自訂處理常式函數 – 您現在可以為 Canary 指令碼使用自訂處理常式函數。先前的執行時間要求指令碼進入點包含 `.handler`。

  您還可以將 Canary 指令碼放在任何資料夾中，並將資料夾名稱作為處理常式的一部分傳遞。例如：`MyFolder/MyScriptFile.functionname` 可用作進入點。
- 用於新增指標和步驟失敗組態的組態選項 – 這些選項已在 Node.js Canary 的執行時間中提供。如需詳細資訊，請參閱 SyntheticsConfiguration 類。
- Chrome 中的自訂引數 – 您現在可以在無痕式模式下開啟瀏覽器或傳入代理伺服器組態。如需詳細資訊，請參閱 Chrome()。
- 跨區域成品儲存貯體 – Canary 可以將其成品存放在不同區域的 Simple Storage Service (Amazon S3) 儲存貯體中。
- 錯誤修正，包括 **`index.py`** 問題的修正 – 對於以前的執行時間，名為 `index.py` 的 Canary 檔案導致異常，因為其與程式庫檔案的名稱發生衝突。現在已修正此問題。

syn-python-selenium-1.0

> **Important**
>
> 此執行階段版本排定於 2024 年 3 月 8 日棄用。如需詳細資訊，請參閱 CloudWatch Synthetics 運行時支持策略。

主要相依性：

- Python 3.8
- Selenium 3.141.0
- Chromium 83.0.4103.0 版

功能：

- Selenium 支援— 您可以使用 Selenium 測試架構撰寫 Canary 指令碼。您可以通過最少的更改將 Selenium 腳本從其他地方帶入 CloudWatch Synthetics，並且它們將與服務一起使用 AWS 。

撰寫 Canary 指令碼

以下各節說明如何撰寫初期測試指令碼，以及如何將 Canary 與其他 AWS 服務以及外部相依性和程式庫整合。

主題

- 撰寫 Node.js Canary 指令碼
- 撰寫 Python Canary 指令碼
- 變更現有的 Selenium 指令碼以作為 Synthetics Canary 使用
- 更改現有的木偶工 Synthetics 腳本以驗證非標準證書

撰寫 Node.js Canary 指令碼

主題

- 從頭開始創建 CloudWatch Synthetics 金絲雀
- 打包您的 Node.js 金絲雀文件
- 變更現有的 Puppeteer 指令碼以作為 Synthetics Canary 使用
- 環境變數
- 將您的金絲雀與其他 AWS 服務集成
- 強制您的 Canary 使用靜態 IP 地址

從頭開始創建 CloudWatch Synthetics 金絲雀

下面是基本 Synthetics Canary 指令碼的範例。此指令碼作為成功執行傳遞，並傳回一個字串。若想查看失敗的 Canary 看起來是什麼樣子，請將 `let fail = false;` 變更為 `let fail = true;`。

您必須定義 Canary 指令碼的進入點函數。若要查看檔案如何上傳到指定為 Canary `ArtifactS3Location` 的 Simple Storage Service (Amazon S3) 位置，請在 /tmp 資料夾下建立這些檔案。指令碼執行之後，通過/失敗狀態和持續時間量度會發佈到， CloudWatch 而 /tmp 下的檔案會上傳到 S3。

```
const basicCustomEntryPoint = async function () {

    // Insert your code here

    // Perform multi-step pass/fail check
```

```
    // Log decisions made and results to /tmp

    // Be sure to wait for all your code paths to complete
    // before returning control back to Synthetics.
    // In that way, your canary will not finish and report success
    // before your code has finished executing

    // Throw to fail, return to succeed
    let fail = false;
    if (fail) {
        throw "Failed basicCanary check.";
    }

    return "Successfully completed basicCanary checks.";
};

exports.handler = async () => {
    return await basicCustomEntryPoint();
};
```

接下來，我們將擴展腳本以使用 Synthetics 日誌記錄並使用 AWS SDK 進行調用。為了示範，此指令碼會建立一個 Amazon DynamoDB 用戶端，並呼叫 DynamoDB listTables API。它會記錄對請求的回應，並根據請求是否成功，記錄通過或是失敗。

```
const log = require('SyntheticsLogger');
const AWS = require('aws-sdk');
// Require any dependencies that your script needs
// Bundle additional files and dependencies into a .zip file with folder structure
// nodejs/node_modules/additional files and folders

const basicCustomEntryPoint = async function () {

    log.info("Starting DynamoDB:listTables canary.");

    let dynamodb = new AWS.DynamoDB();
    var params = {};
    let request = await dynamodb.listTables(params);
    try {
        let response = await request.promise();
        log.info("listTables response: " + JSON.stringify(response));
    } catch (err) {
        log.error("listTables error: " + JSON.stringify(err), err.stack);
```

```
        throw err;
    }

    return "Successfully completed DynamoDB:listTables canary.";
};

exports.handler = async () => {
    return await basicCustomEntryPoint();
};
```

打包您的 Node.js 金絲雀文件

如果您使用 Simple Storage Service (Amazon S3) 位置上傳 Canary 指令碼，則 zip 檔案應在此資料夾結構下包含指令碼：nodejs/node_modules/*myCanaryFilename.js file*。

如果您有多個 .js 檔案，或者您的指令碼有依賴的相依性，則可將這些項目全部封裝到包含資料夾結構 nodejs/node_modules/*myCanaryFilename.js file and other folders and files* 的單一 ZIP 檔案中。如果您是使用 syn-nodejs-puppeteer-3.4 或更高版本，您可以選擇將 Canary 檔案放在另一個資料夾中，並像這樣建立資料夾結構：nodejs/node_modules/*myFolder*/*myCanaryFilename.js file and other folders and files*。

處理常式名稱

請務必將 Canary 的指令碼進入點 (處理常式) 設定為 myCanaryFilename.functionName，以符合指令碼進入點的檔案名稱。如果您使用的執行時間早於 syn-nodejs-puppeteer-3.4，則 functionName 必須為 handler。如果您使用的是 syn-nodejs-puppeteer-3.4 或更高版本，您可以選擇任何函數名稱作為處理常式。如果您使用的是 syn-nodejs-puppeteer-3.4 或更高版本，您還可以選擇將 Canary 存放在單獨的資料夾 (例如 nodejs/node_modules/myFolder/my_canary_filename) 中。如果將其存放在單獨的資料夾中，請在指令碼進入點中指定該路徑，例如 myFolder/my_canary_filename.functionName。

變更現有的 Puppeteer 指令碼以作為 Synthetics Canary 使用

本節介紹如何採用 Puppeteer 指令碼和修改它們以作為 Synthetics Canary 指令碼執行。如需 Puppeteer 的詳細資訊，請參閱 Puppeteer API v1.14.0。

我們將從這個 Puppeteer 指令碼範例開始：

```
const puppeteer = require('puppeteer');
```

```
(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();
  await page.goto('https://example.com');
  await page.screenshot({path: 'example.png'});

  await browser.close();
})();
```

轉換步驟如下：

- 建立和匯出 handler 函數。處理常式是指令碼的進入點函數。如果您使用的執行時間早於 syn-nodejs-puppeteer-3.4，則處理常式函數必須命名為 handler。如果您使用的是 syn-nodejs-puppeteer-3.4 或更高版本，函數可以具有任何名稱，但其必須與指令碼中使用的名稱相同。另外，如果您使用的是 syn-nodejs-puppeteer-3.4 或更高版本，您可以將指令碼存放在任何資料夾下，並將該資料夾指定為處理常式名稱的一部分。

  ```
  const basicPuppeteerExample = async function () {};

  exports.handler = async () => {
      return await basicPuppeteerExample();
  };
  ```

- 使用 Synthetics 相依性。

  ```
  var synthetics = require('Synthetics');
  ```

- 使用 Synthetics.getPage 函數來取得 Puppeteer Page 物件。

  ```
  const page = await synthetics.getPage();
  ```

  Synthetics.getPage 函數傳回的頁面物件具有 page.on request，以及檢測用於記錄的 response 和 requestfailed 事件。Synthetics 也會針對頁面上的請求和回應設定 HAR 檔案產生，並將 Canary ARN 新增至頁面上傳出請求的 user-agent 標頭。

指令碼現在已準備好作為 Synthetics Canary 執行。這是更新後的指令碼：

```
var synthetics = require('Synthetics');  // Synthetics dependency

const basicPuppeteerExample = async function () {
```

```
    const page = await synthetics.getPage(); // Get instrumented page from Synthetics
    await page.goto('https://example.com');
    await page.screenshot({path: '/tmp/example.png'}); // Write screenshot to /tmp
 folder
};

exports.handler = async () => {  // Exported handler function
    return await basicPuppeteerExample();
};
```

環境變數

建立 Canary 時，您可以使用環境變數。這允許您編寫單一 Canary 指令碼，然後使用具有不同數值的指令碼來快速建立具有類似任務的多個 Canary。

例如，假設您的組織在軟體開發的不同階段擁有諸如 `prod`、`dev` 和 `pre-release` 之類的端點，並且您需要建立 Canary 來測試這些端點。您可以撰寫測試軟體的單一 Canary 指令碼，然後在建立三個 Canary 的每一個 Canary 時，為端點環境變數指定不同的數值。然後，當您建立 Canary 時，您可以指定要用於環境變數的指令碼和數值。

環境變數名稱可包含字母、數字和底線字元。其必須以字母開頭，且至少有兩個字元。環境變數的總大小不能超過 4 KB。您無法指定任何 Lambda 保留環境變數作為環境變數的名稱。如需有關保留環境變數的詳細資訊，請參閱執行時間環境變數。

> **Important**
>
> 環境變數索引鍵和值未加密。請勿在其中存放敏感資訊。

以下範例指令碼使用了兩個環境變數。這個指令碼可用於檢查網頁是否可用的 Canary。它使用環境變量來參數化它檢查的 URL 和它使用的 CloudWatch Synthetics 日誌級別。

以下函數會將 `LogLevel` 設定為 `LOG_LEVEL` 環境變數的數值。

```
 synthetics.setLogLevel(process.env.LOG_LEVEL);
```

此函數會將 URL 設定為 URL 環境變數的數值。

```
const URL = process.env.URL;
```

這是完整的指令碼。當您使用此指令碼建立 Canary 時，您可以指定要用於 LOG_LEVEL 和 URL 環境變數的數值。

```
var synthetics = require('Synthetics');
const log = require('SyntheticsLogger');

const pageLoadEnvironmentVariable = async function () {

    // Setting the log level (0-3)
    synthetics.setLogLevel(process.env.LOG_LEVEL);
    // INSERT URL here
    const URL = process.env.URL;

    let page = await synthetics.getPage();
    //You can customize the wait condition here. For instance,
    //using 'networkidle2' may be less restrictive.
    const response = await page.goto(URL, {waitUntil: 'domcontentloaded', timeout:
 30000});
    if (!response) {
        throw "Failed to load page!";
    }
    //Wait for page to render.
    //Increase or decrease wait time based on endpoint being monitored.
    await page.waitFor(15000);
    await synthetics.takeScreenshot('loaded', 'loaded');
    let pageTitle = await page.title();
    log.info('Page title: ' + pageTitle);
    log.debug('Environment variable:' + process.env.URL);

    //If the response status code is not a 2xx success code
    if (response.status() < 200 || response.status() > 299) {
        throw "Failed to load page!";
    }
};

exports.handler = async () => {
    return await pageLoadEnvironmentVariable();
};
```

將環境變數傳遞給指令碼

若要在主控台中建立 Canary 時將環境變數傳遞至指令碼，請在主控台的 Environment variables (環境變數) 區段中指定環境變數的金鑰和數值。如需詳細資訊，請參閱 建立 Canary。

若要透過 API 傳遞環境變數 AWS CLI，或使用RunConfig區段中的EnvironmentVariables參數。下列範例 AWS CLI 命令會建立使用兩個環境變數與 and 的Environment金鑰建立初期測試Region。

```
aws synthetics create-canary --cli-input-json '{
   "Name":"nameofCanary",
   "ExecutionRoleArn":"roleArn",
   "ArtifactS3Location":"s3://cw-syn-results-123456789012-us-west-2",
   "Schedule":{
      "Expression":"rate(0 minute)",
      "DurationInSeconds":604800
   },
   "Code":{
      "S3Bucket": "canarycreation",
      "S3Key": "cwsyn-mycanaryheartbeat-12345678-d1bd-1234-
abcd-123456789012-12345678-6a1f-47c3-b291-123456789012.zip",
      "Handler":"pageLoadBlueprint.handler"
   },
   "RunConfig": {
      "TimeoutInSeconds":60,
      "EnvironmentVariables": {
         "Environment":"Production",
         "Region": "us-west-1"
      }
   },
   "SuccessRetentionPeriodInDays":13,
   "FailureRetentionPeriodInDays":13,
   "RuntimeVersion":"syn-nodejs-2.0"
}'
```

將您的金絲雀與其他 AWS 服務集成

所有金絲雀都可以使用 AWS SDK 庫。當您編寫金絲雀時，您可以使用此庫將金絲雀與其他 AWS 服務集成在一起。

要這麼做，您需要新增下面的程式碼到您的 Canary。在這些範例中 AWS Secrets Manager ，會用作初期測試與之整合的服務。

- 匯入 AWS SDK。

```
const AWS = require('aws-sdk');
```

- 為您要整合的 AWS 服務建立用戶端。

```
const secretsManager = new AWS.SecretsManager();
```

- 使用用戶端對該服務進行 API 呼叫。

```
var params = {
  SecretId: secretName
};
return await secretsManager.getSecretValue(params).promise();
```

下面的 Canary 指令碼程式碼片段示範了與 Secrets Manager 整合的詳細範例。

```
var synthetics = require('Synthetics');
const log = require('SyntheticsLogger');

const AWS = require('aws-sdk');
const secretsManager = new AWS.SecretsManager();

const getSecrets = async (secretName) => {
    var params = {
        SecretId: secretName
    };
    return await secretsManager.getSecretValue(params).promise();
}

const secretsExample = async function () {
    let URL = "<URL>";
    let page = await synthetics.getPage();

    log.info(`Navigating to URL: ${URL}`);
    const response = await page.goto(URL, {waitUntil: 'domcontentloaded', timeout:
 30000});

    // Fetch secrets
    let secrets = await getSecrets("secretname")

    /**
    * Use secrets to login.
    *
    * Assuming secrets are stored in a JSON format like:
    * {
```

```
    *   "username": "<USERNAME>",
    *   "password": "<PASSWORD>"
    * }
    **/
    let secretsObj = JSON.parse(secrets.SecretString);
    await synthetics.executeStep('login', async function () {
        await page.type(">USERNAME-INPUT-SELECTOR<", secretsObj.username);
        await page.type(">PASSWORD-INPUT-SELECTOR<", secretsObj.password);

        await Promise.all([
          page.waitForNavigation({ timeout: 30000 }),
          await page.click(">SUBMIT-BUTTON-SELECTOR<")
        ]);
    });

    // Verify login was successful
    await synthetics.executeStep('verify', async function () {
        await page.waitForXPath(">SELECTOR<", { timeout: 30000 });
    });
};

exports.handler = async () => {
    return await secretsExample();
};
```

強制您的 Canary 使用靜態 IP 地址

您可以設定 Canary，以便使用靜態 IP 地址。

若要強制 Canary 使用靜態 IP 地址

1. 建立新 VPC 如需詳細資訊，請參閱使用 DNS 與您的 VPC 搭配。
2. 建立新的網際網路閘道。如需詳細資訊，請參閱將網際網路閘道新增至您的 VPC。
3. 在您的新 VPC 內部建立一個公有子網路。
4. 將新的路由表新增到 VPC。
5. 在新的路由表中新增一個路由，而該路由從 `0.0.0.0/0` 移至網際網路閘道。
6. 將新的路由表與公有子網路建立關聯。
7. 建立彈性 IP 地址。如需詳細資訊，請參閱彈性 IP 地址。
8. 建立新的 NAT 閘道，並將其指派給公有子網路和彈性 IP 地址。
9. 在 VPC 內部建立私有子網路。

10. 將路由新增至 VPC 預設路由表，即從 0.0.0.0/0 移至 NAT 閘道

11. 建立 Canary。

撰寫 Python Canary 指令碼

此指令碼作為成功執行傳遞，並傳回一個字串。若想查看失敗的 Canary 看起來是什麼樣子，請將 fail = False 變更為 fail = True

```
def basic_custom_script():
    # Insert your code here
    # Perform multi-step pass/fail check
    # Log decisions made and results to /tmp
    # Be sure to wait for all your code paths to complete
    # before returning control back to Synthetics.
    # In that way, your canary will not finish and report success
    # before your code has finished executing
    fail = False
    if fail:
        raise Exception("Failed basicCanary check.")
    return "Successfully completed basicCanary checks."
def handler(event, context):
    return basic_custom_script()
```

打包你的 Python 金絲雀文件

如果您有多個 .py 檔案，或者指令碼具有相依性，則可將其全部封裝到單一 ZIP 檔案中。如果您使用 syn-python-selenium-1.1 執行時間，ZIP 檔案在 python 資料夾中必須包含主要 Canary .py 檔案，例如 python/my_canary_filename.py。如果您使用的是 syn-python-selenium-1.1 或更高版本，您可以選擇使用其他資料夾，例如 python/myFolder/my_canary_filename.py。

此 ZIP 檔案應包含所有必要的資料夾和檔案，但其他檔案不需要位於 python 資料夾中。

請務必將 Canary 的指令碼進入點設定為 my_canary_filename.functionName，以符合指令碼進入點的檔案名稱和函數名稱。如果您使用的是 syn-python-selenium-1.0 執行時間，則 functionName 必須為 handler。如果您使用的是 syn-python-selenium-1.1 或更高版本，此處理常式名稱限制就不適用，您還可以選擇將 Canary 存放在單獨的資料夾 (例如 python/myFolder/my_canary_filename.py) 中。如果將其存放在單獨的資料夾中，請在指令碼進入點中指定該路徑，例如 myFolder/my_canary_filename.functionName。

變更現有的 Selenium 指令碼以作為 Synthetics Canary 使用

您可以快速修改 Python 和 Selenium 用作 Canary 的現有指令碼。如需 Selenium 的詳細資訊，請參閱 www.selenium.dev/。

在此範例中，我們會從以下 Selenium 指令碼開始：

```
from selenium import webdriver

def basic_selenium_script():
    browser = webdriver.Chrome()
    browser.get('https://example.com')
    browser.save_screenshot('loaded.png')

basic_selenium_script()
```

轉換步驟如下。

若要將 Selenium 指令碼轉換可用作 Canary

1. 將變更 `import` 陳述式，以使用來自 `aws_synthetics` 模組的 Selenium：

   ```
   from aws_synthetics.selenium import synthetics_webdriver as webdriver
   ```

   Selenium 模塊`aws_synthetics`確保金絲雀可以發出指標和日誌，生成 HAR 文件，並與其他 CloudWatch Synthetics 功能一起工作。
2. 建立一個處理常式函數並呼叫您的 Selenium 方法。處理常式是指令碼的進入點函數。

   如果您使用的是 `syn-python-selenium-1.0`，則處理常式函數必須命名為 `handler`。如果您使用的是 `syn-python-selenium-1.1` 或更高版本，函數可以具有任何名稱，但其必須與指令碼中使用的名稱相同。另外，如果您使用的是 `syn-python-selenium-1.1` 或更高版本，您可以將指令碼存放在任何資料夾下，並將該資料夾指定為處理常式名稱的一部分。

   ```
   def handler(event, context):
       basic_selenium_script()
   ```

腳本現在更新為 CloudWatch Synthetics 金絲雀。這是更新後的指令碼：

```
from aws_synthetics.selenium import synthetics_webdriver as webdriver
```

```
def basic_selenium_script():
    browser = webdriver.Chrome()
    browser.get('https://example.com')
    browser.save_screenshot('loaded.png')

def handler(event, context):
    basic_selenium_script()
```

更改現有的木偶工 Synthetics 腳本以驗證非標準證書

Synthetics 金絲雀的一個重要用例是監視自己的端點。如果您想要監視尚未準備好接受外部流量的端點，此監視有時可能表示您沒有受信任的協力廠商憑證授權單位簽署的適當憑證。

這種情況的兩種可能的解決方案如下：

- 若要驗證用戶端憑證，請參閱如何使用 Amazon CloudWatch Synthetics 驗證身份驗證 — 第 2 部分。
- 若要驗證自我簽署的憑證，請參閱如何在 Amazon Synthetics 中使用自我簽署憑證驗證驗證 CloudWatch

當您使用 CloudWatch Synthetics 金絲雀時，您不僅限於這兩個選項。您可以擴展初期測試代碼來擴展這些功能並添加業務邏輯。

> **Note**
>
> 在 Python 運行時運行的 Synthetics 金絲雀本來就啟用了`--ignore-certificate-errors`標誌，所以那些金絲雀不應該有任何問題到達非標準證書配置的站點。

適用於 Canary 指令碼的程式庫函數

CloudWatch Synthetics 包括幾個內置的類和函數，您可以在編寫 Node.js 腳本時調用它們作為金絲雀。

某些函數同時適用於 UI 和 API Canary。某些函數僅適用於 UI Canary。UI Canary 是使用 `getPage()` 函數的 Canary，它會使用 Puppeteer 作為 Web 驅動程式來瀏覽網頁並進行互動。

> **Note**
>
> 每當您升級金絲雀以使用新版本的 Synthetics 執行期時，您的金絲雀使用的所有 Synthetics 程式庫函數也會自動升級到 Synthetics 執行期支援的相同版本的 NodeJS。

主題

- 適用於 Node.js Canary 指令碼的程式庫函數
- 可用於使用 Selenium 的 Python Canary 指令碼的程式庫函數

適用於 Node.js Canary 指令碼的程式庫函數

本區段列出了適用於 Node.js Canary 指令碼的程式庫函數

主題

- 適用於所有 Canary 的 Node.js 程式庫類別和函數
- 僅適用於 UI Canary 的 Node.js 程式庫類別和函數
- 僅適用於 API Canary 的 Node.js 程式庫類別和函數

適用於所有 Canary 的 Node.js 程式庫類別和函數

以下 Node.js 的 CloudWatch Synthetics 庫函數對於所有金絲雀都很有用。

主題

- Synthetics 類別
- SyntheticsConfiguration 類
- Synthetics Logger
- SyntheticsLogHelper 類

Synthetics 類別

以下適用於所有 Canary 的函數都在 Synthetics 類別中。

addExecutionError（errorMessage，前）；

`errorMessage` 描述錯誤，而 `ex` 是遇到的例外狀況

您可以使用 addExecutionError 為您的 Canary 設定執行錯誤。它會在不中斷指令碼執行的情況下使 Canary 失敗。它也不會影響您的 successPercent 指標。

只有在錯誤表示 Canary 指令碼成功或失敗並不重要時，才應將錯誤追蹤為執行錯誤。

addExecutionError 的使用範例如下所示。您正在監控端點的可用性，並在頁面載入後擷取螢幕擷取畫面。由於擷取螢幕擷取畫面的失敗並不會決定端點的可用性，因此您可以捕捉擷取螢幕擷取畫面時遇到的任何錯誤，並將它們新增為執行錯誤。您的可用性指標仍會顯示端點已啟動並執行，但您的 Canary 狀態會標示為失敗。下面的範本程式碼區塊可捕獲這樣的錯誤，並將其新增為執行錯誤。

```
try {
        await synthetics.takeScreenshot(stepName, "loaded");
    } catch(ex) {
        synthetics.addExecutionError('Unable to take screenshot ', ex);
    }
```

getCanaryName();

傳回 Canary 的名稱。

getCanaryArn();

傳回 Canary 的 ARN。

getCanaryUserAgentString();

傳回 Canary 的自訂使用者代理程式。

getRuntimeVersion();

此函數在執行時間版本 syn-nodejs-puppeteer-3.0 和更新版本中可用。其會傳回 Canary 的 Synthetics 執行時間版本。例如，傳回值可以是 syn-nodejs-puppeteer-3.0。

getLogLevel();

擷取 Synthetics 程式庫的目前日誌層級。可能的值如下：

- 0 – 偵錯
- 1 – 資訊
- 2 – 警告
- 3 – 錯誤

範例：

```
let logLevel = synthetics.getLogLevel();
```

setLogLevel();

設定 Synthetics 程式庫的日誌層級。可能的值如下：

- 0 – 偵錯
- 1 – 資訊
- 2 – 警告
- 3 – 錯誤

範例：

```
synthetics.setLogLevel(0);
```

SyntheticsConfiguration 類

此類別僅在 syn-nodejs-2.1 執行時間版本或更新版本中可用。

您可以使用 SyntheticsConfiguration 類來配置 Synthetics 庫函數的行為。例如，您可以使用此類別來將 executeStep() 函數設定為不擷取螢幕擷取畫面。

您可以在全局級別設置 CloudWatch Synthetics 配置，該配置應用於加那利群島的所有步驟。您也可以透過傳遞組態鍵/值對來在步驟層級覆寫這些組態。

您可以在步驟層級傳遞選項。如需範例，請參閱 異步 executeStep（步驟名， functionToExecute[步驟配置]）; 和 executeHttpStep（stepName，請求選項，[回調]，[步驟配置]）

函數定義：

setConfig(options)

options 是一個物件，適用於您的 Canary 的可設定選項集。以下各節會說明了 *options* 中的可能欄位。

適用於所有 Canary 的 setConfig(options)

對於使用 syn-nodejs-puppeteer-3.2 或更新版本的 Canary，setConfig 的 (選項) 可以包含以下參數：

- includeRequestHeaders (布林值)— 是否要在報告中包含請求標頭。預設值為 false。
- includeResponseHeaders (布林值)— 是否要在報告中包含回應標頭。預設值為 false。
- restrictedHeaders (陣列)— 如果包含標頭，則要忽略的標頭值清單。這適用於請求和回應標頭。例如，您可以通過將身份傳遞includeRequestHeaders為true和限制標題來隱藏您的憑據。['Authorization']
- includeRequestBody (布林值)— 是否要在報告中包含請求內文。預設值為 false。
- includeResponseBody (布林值)— 是否要在報告中包含回應內文。預設值為 false。

設定設定 (選項) 有關 CloudWatch 度量

對於使用 syn-nodejs-puppeteer-3.1 或更新版本的 Canary，setConfig 的 (選項) 可以包含以下布林值參數，可確定 Canary 發佈哪些指標。這些選項的預設值為 true。以 aggregated 開頭的選項可判斷是否會發出沒有 CanaryName 維度的指標。您可以使用這些指標來查看所有 Canary 的彙總結果。其他選項可判斷是否會發出具有 CanaryName 維度的指標。您可以使用這些指標來查看每個 Canary 的結果。

如需加那利群島發出的 CloudWatch 指標清單，請參閱。CloudWatch 加那利群島發布的指標

- failedCanaryMetric (布林值)— 是否為此 canary 發出 Failed 指標 (具有 CanaryName 維度)。預設值為 true。
- failedRequestsMetric (布林值)— 是否為此 canary 發出 Failed requests 指標 (具有 CanaryName 維度)。預設值為 true。
- _2xxMetric (布林值)— 是否為此 canary 發出 2xx 指標 (具有 CanaryName 維度)。預設值為 true。
- _4xxMetric (布林值)— 是否為此 canary 發出 4xx 指標 (具有 CanaryName 維度)。預設值為 true。
- _5xxMetric (布林值)— 是否為此 canary 發出 5xx 指標 (具有 CanaryName 維度)。預設值為 true。
- stepDurationMetric (布林值)— 是否為此 canary 發出 Step duration 指標 (具有 CanaryName StepName 維度)。預設值為 true。
- stepSuccessMetric (布林值)— 是否為此 canary 發出 Step success 指標 (具有 CanaryName StepName 維度)。預設值為 true。
- aggregatedFailedCanaryMetric (布林值)— 是否為此 canary 發出 Failed 指標 (沒有 CanaryName 維度)。預設值為 true。

- aggregatedFailedRequestsMetric (布林值)— 是否為此 canary 發出 Failed Requests 指標 (沒有 CanaryName 維度)。預設值為 true。
- aggregated2xxMetric (布林值)— 是否為此 canary 發出 2xx 指標 (沒有 CanaryName 維度)。預設值為 true。
- aggregated4xxMetric (布林值)— 是否為此 canary 發出 4xx 指標 (沒有 CanaryName 維度)。預設值為 true。
- aggregated5xxMetric (布林值)— 是否為此 canary 發出 5xx 指標 (沒有 CanaryName 維度)。預設值為 true。
- visualMonitoringSuccessPercentMetric (布林值)— 是否為此 canary 發出 visualMonitoringSuccessPercent 指標。預設值為 true。
- visualMonitoringTotalComparisonsMetric (布林值)— 是否為此 canary 發出 visualMonitoringTotalComparisons 指標。預設值為 false。
- stepsReport(布林值) — 是否要報告步驟執行摘要。預設值為 true。
- includeUrlPassword(布林值) — 是否要包含在 URL 中顯示的密碼。依預設，在 URL 中顯示的密碼會從日誌和報告進行修訂，以防止揭露敏感資料。預設值為 false。
- restrictedUrlParameters (陣列)— 要修訂的 URL 路徑或查詢參數清單。這適用於出現在日誌、報告和錯誤中的 URL。參數區分大小寫。您可以將星號 (*) 傳遞為數值，以修訂所有 URL 路徑和查詢參數值。預設為空陣列。
- logRequest (布林值)— 是否要在 Canary 日誌中記錄每個請求。對於 UI Canary，這會記錄瀏覽器傳送的每個請求。預設值為 true。
- logResponse (布林值)— 是否要在 Canary 日誌中記錄每個回應。對於 UI Canary，這會記錄瀏覽器收到的每個回應。預設值為 true。
- logRequestBody (布林值)— 是否要在 Canary 日誌中記錄請求內文及請求。此組態僅在 logRequest 為 true 時適用。預設值為 false。
- logResponseBody (布林值)— 是否要在 Canary 日誌中記錄回應內文及回應。此組態僅在 logResponse 為 true 時適用。預設值為 false。
- logRequestHeaders (布林值)— 是否要在 Canary 日誌中記錄請求標頭及請求。此組態僅在 logRequest 為 true 時適用。預設值為 false。

  請注意，includeRequestHeaders 會啟用成品中的標頭。
- logResponseHeaders (布林值)— 是否要在 Canary 日誌中記錄回應標頭及回應。此組態僅在 logResponse 為 true 時適用。預設值為 false。

  請注意，includeResponseHeaders 會啟用成品中的標頭。

> **Note**
>
> 總是會針對每個 Canary 發出 `Duration` 和 `SuccessPercent` 指標，而無論是否有 `CanaryName` 指標。

啟用或停用指標的方法

disableAggregatedRequest度量 ()

禁止 Canary 發出所有沒有 `CanaryName` 維度發出的請求指標。

disableRequestMetrics()

停用所有請求指標，包括每個 Canary 指標和彙總所有 Canary 的指標。

disableStepMetrics()

停用所有步驟指標，包括步驟成功指標和步驟持續時間指標。

enableAggregatedRequest度量 ()

啟用 Canary，使其可以發出所有沒有 `CanaryName` 維度發出的請求指標。

enableRequestMetrics()

啟用所有請求指標，包括每個 Canary 指標和彙總所有 Canary 的指標。

enableStepMetrics()

啟用所有步驟指標，包括步驟成功指標和步驟持續時間指標。

get2xxMetric()

傳回 Canary 是否發出具有 `CanaryName` 維度的 `2xx` 指標。

get4xxMetric()

傳回 Canary 是否發出具有 `CanaryName` 維度的 `4xx` 指標。

get5xxMetric()

傳回 Canary 是否發出具有 `CanaryName` 維度的 `5xx` 指標。

getAggregated2xxMetric()

傳回 Canary 是否發出沒有維度的 2xx 指標。

getAggregated4xxMetric()

傳回 Canary 是否發出沒有維度的 4xx 指標。

getAggregatedFailedCanaryMetric()

傳回 Canary 是否發出沒有維度的 Failed 指標。

getAggregatedFailedRequestsMetric()

傳回 Canary 是否發出沒有維度的 Failed requests 指標。

getAggregated5xxMetric()

傳回 Canary 是否發出沒有維度的 5xx 指標。

getFailedCanary公制 ()

傳回 Canary 是否發出具有 CanaryName 維度的 Failed 指標。

getFailedRequests公制 ()

傳回 Canary 是否發出具有 CanaryName 維度的 Failed requests 指標。

getStepDuration公制 ()

傳回 Canary 是否為此 Canary 發出具有 CanaryName 維度的 Duration 指標。

getStepSuccess公制 ()

傳回 Canary 是否為此 Canary 發出具有 CanaryName 維度的 StepSuccess 指標。

with2xxMetric(_2xxMetric)

接受布林值引數，它會指定是否為此 Canary 發出具有 CanaryName 維度的 2xx 指標。

with4xxMetric(_4xxMetric)

接受布林值引數，它會指定是否為此 Canary 發出具有 CanaryName 維度的 4xx 指標。

with5xxMetric(_5xxMetric)

接受布林值引數，它會指定是否為此 Canary 發出具有 CanaryName 維度的 5xx 指標。

withAggregated2xxMetric(aggregated2xxMetric)

接受布林值引數，它會指定是否為此 Canary 發出沒有維度的 2xx 指標。

withAggregated4xxMetric(aggregated4xxMetric)

接受布林值引數，它會指定是否為此 Canary 發出沒有維度的 4xx 指標。

withAggregated5xxMetric(aggregated5xxMetric)

接受布林值引數，它會指定是否為此 Canary 發出沒有維度的 5xx 指標。

withAggregatedFailedCanaryMetric(aggregatedFailedCanary公制)

接受布林值引數，它會指定是否為此 Canary 發出沒有維度的 Failed 指標。

withAggregatedFailedRequestsMetric(aggregatedFailedRequests公制)

接受布林值引數，它會指定是否為此 Canary 發出沒有維度的 Failed requests 指標。

withFailedCanary公制 (failedCanaryMetric)

接受布林值引數，它會指定是否為此 Canary 發出具有 CanaryName 維度的 Failed 指標。

withFailedRequests公制 (failedRequestsMetric)

接受布林值引數，它會指定是否為此 Canary 發出具有 CanaryName 維度的 Failed requests 指標。

withStepDuration公制 (stepDurationMetric)

接受布林值引數，它會指定是否為此 Canary 發出具有 CanaryName 維度的 Duration 指標。

withStepSuccess公制 (stepSuccessMetric)

接受布林值引數，它會指定是否為此 Canary 發出具有 CanaryName 維度的 StepSuccess 指標。

啟用或停用其他功能的方法

withHarFile()

接受布林值引數，它會指定是否為此 Canary 建立 HAR 檔案。

withStepsReport()

接受布林值引數，它會指定是否為此 Canary 報告步驟摘要。

withIncludeUrl密碼 ()

接受布林值引數，它會指定是否要在日誌和報告中包含在 URL 中顯示的密碼。

withRestrictedUrl參數 ()

接受要修訂的 URL 路徑或查詢參數陣列。這適用於出現在日誌、報告和錯誤中的 URL。您可以將星號 (*) 傳遞為數值，以修訂所有 URL 路徑和查詢參數值

withLogRequest()

接受布林值引數，它會指定是否要在 Canary 日誌中記錄每個請求。

withLogResponse()

接受布林值引數，它會指定是否要在 Canary 日誌中記錄每個回應。

withLogRequest身體 ()

接受布林值引數，它會指定是否要在 Canary 日誌中記錄每個請求內文。

withLogResponse身體 ()

接受布林值引數，它會指定是否要在 Canary 日誌中記錄每個回應內文。

withLogRequest標頭 ( )

接受布林值引數，它會指定是否要在 Canary 日誌中記錄每個請求標頭。

withLogResponse標頭 ( )

接受布林值引數，它會指定是否要在 Canary 日誌中記錄每個回應。

getHarFile()

傳回 Canary 是否建立 HAR 檔案。

getStepsReport()

傳回 Canary 是否報告步驟執行摘要。

getIncludeUrl密碼 ()

傳回 Canary 是否要在日誌和報告中包含在 URL 中顯示的密碼。

getRestrictedUrl參數 ()

傳回 Canary 是否修訂 URL 路徑或查詢參數。

getLogRequest()

傳回 Canary 是否記錄 Canary 日誌中的每個請求。

getLogResponse()

傳回 Canary 是否記錄 Canary 日誌中的每個回應。

getLogRequest身體 ()

傳回 Canary 是否記錄 Canary 日誌中的每個請求內文。

getLogResponse身體 ()

傳回 Canary 是否記錄 Canary 日誌中的每個回應內文。

getLogRequest標頭 ( )

傳回 Canary 是否記錄 Canary 日誌中的每個請求標頭。

getLogResponse標頭 ( )

傳回 Canary 是否記錄 Canary 日誌中的每個回應標頭。

適用於所有 Canary 的函數

- `withIncludeRequestHeaders`(includeRequestHeaders)
- `withIncludeResponseHeaders`(includeResponseHeaders)
- `withRestrictedHeaders`(restrictedHeaders)
- `withIncludeRequestBody`(includeRequestBody)
- `withIncludeResponseBody`(includeResponseBody)
- `enableReportingOptions`() — 啟用所有報告選項 —-includeRequestHeaders、includeResponseHeaders、 和includeRequestBodyincludeResponseBody、。

- disableReportingOptions() — 停用所有報告選項-includeRequestHeaders、includeResponseHeaders、和includeRequestBodyincludeResponseBody、。

適用於 UI Canary 的 setConfig(options)

對於 UI Canary，setConfig 可以包括下列布林值參數：

- continueOnStepFailure (布林值)— 是否在步驟失敗之後繼續執行 Canary 指令碼 (其代表 executeStep 函數)。如果任何步驟失敗，Canary 執行仍將被標記為失敗。預設值為 false。
- harFile (布林值)— 是否建立 HAR 檔案。預設值為 True。
- screenshotOnStepStart (布林值)— 是否在開始步驟之前擷取螢幕擷取畫面。
- screenshotOnStepSuccess (布林值)— 是否在成功完成步驟之後擷取螢幕擷取畫面。
- screenshotOnStepFailure (布林值)— 是否在步驟失敗之後擷取螢幕擷取畫面。

啟用或停用螢幕擷取畫面的方法

disableStepScreenshots()

停用所有螢幕快照選項 ([screenshotOnStep開始]、[screenshotOnStep成功] 和 [screenshotOnStep失敗])。

enableStepScreenshots()

啟用所有螢幕擷取畫面選項 (screenshotOnStep開始、 screenshotOnStep成功和 screenshotOnStep失敗)。預設不會啟用所有這些方法。

getScreenshotOnStepFailure()

傳回 Canary 是否在步驟失敗之後擷取螢幕擷取畫面。

getScreenshotOnStepStart()

傳回 Canary 是否在開始步驟之前擷取螢幕擷取畫面。

getScreenshotOnStepSuccess()

傳回 Canary 是否在成功完成步驟之後擷取螢幕擷取畫面。

withScreenshotOnStepStart（screenshotOnStep開始）

接受布林值引數，它會指示是否在開始步驟之前擷取螢幕擷取畫面。

withScreenshotOnStepSuccess（screenshotOnStep成功）

接受布林值引數，它會指示是否在成功完成步驟之後擷取螢幕擷取畫面。

withScreenshotOnStepFailure(screenshotOnStep失敗)

接受布林值引數，它會指示是否在步驟失敗之後擷取螢幕擷取畫面。

在 UI Canary 中的使用方式

首先，匯入綜合相依性並擷取組態。

```
// Import Synthetics dependency
const synthetics = require('Synthetics');

// Get Synthetics configuration
const synConfig = synthetics.getConfiguration();
```

然後，使用下列其中一個選項呼叫 setConfig 方法來設定每個選項的組態。

```
// Set configuration values
    synConfig.setConfig({
        screenshotOnStepStart: true,
        screenshotOnStepSuccess: false,
        screenshotOnStepFailure: false
    });
```

或

```
synConfig.withScreenshotOnStepStart(false).withScreenshotOnStepSuccess(true).withScreenshotOnSt
```

若要停用所有螢幕擷取畫面，請使用 disableStepScreenshots () 函數，如本範例所示。

```
synConfig.disableStepScreenshots();
```

您可以在程式碼中的任何時候啟用和停用螢幕擷取畫面。例如，若只要停用一個步驟的螢幕擷取畫面，請在執行該步驟之前停用這些螢幕擷取畫面，然後在步驟之後予以啟用。

適用於 API Canary 的 setConfig(options)

對於 API Canary，setConfig 可以包括下列布林值參數：

- `continueOnHttpStepFailure`(布林值) — HTTP 步驟失敗後是否繼續執行初期測試指令碼 (這是指executeHttpStep函數)。如果任何步驟失敗，Canary 執行仍將被標記為失敗。預設值為 `true`。

視覺化監控

視覺化監控可比較在 Canary 執行期間擷取的螢幕擷取畫面與基準 Canary 執行期間擷取的螢幕擷取畫面。如果兩個螢幕擷取畫面之間的差異超出臨界值百分比，則 Canary 會失敗，您可以在 Canary 執行報告中看到顏色差異的區域。運行 syn-puppeteer-node-3.2 及更高版本的金絲雀支持可視化監視。目前，執行 Python 和 Selenium 的 Canary 不支援它。

若要啟用視覺化監控，請將下列程式碼行新增至 Canary 指令碼。如需詳細資訊，請參閱SyntheticsConfiguration 類。

```
syntheticsConfiguration.withVisualCompareWithBaseRun(true);
```

將此行新增至指令碼之後，Canary 第一次成功執行時，它會使用該執行期間擷取的螢幕擷取畫面作為比較基準。在第一次初次測試執行之後，您可以使用 CloudWatch 主控台編輯初期測試，以執行下列任一項作業：

- 將 Canary 的下一個執行設定為新基準。
- 在目前的基準螢幕擷取畫面上繪製邊界，以指定要在視覺化比較期間忽略的螢幕擷取畫面區域。
- 移除用於視覺化監控的螢幕擷取畫面。

如需有關使用 CloudWatch 主控台編輯初期測試的詳細資訊，請參閱編輯或刪除 Canary。

其他視覺化監控選項

合成配置。 withVisualVarianceThresholdPercentage(所需百分比)

在視覺化效果比較中，設定螢幕擷取畫面差異的可接受百分比。

合成配置。 withVisualVarianceHighlightHexColor(」#fafa00「)

當您查看使用視覺化監控的 Canary 執行報告時，設定指定差異區域的反白顏色。

合成配置。 withFailCanaryRunOnVisualVariance（失敗加那利）

設定當有視覺化差異超過閾值時，Canary 是否失敗。預設為 Canary 失敗。

Synthetics Logger

SyntheticsLogger 將登出寫入主控台和相同記錄層級的本機記錄檔。只有在日誌層級等於或低於呼叫之日誌函數所需的日誌層級時，才會將此日誌檔案同時寫入至這兩個位置。

本機日誌檔案中的日誌陳述式會加上「偵錯」、「資訊」等前綴，以符合呼叫之函數的日誌層級。

您可以使用 SyntheticsLogger，假設您想要在與 Synthetics 金絲雀日誌記錄相同的日誌級別上運行 Synthetics 庫。

不需要使用即可建立上傳到 S3 結果位置的日誌檔。 SyntheticsLogger 您可改為在 `/tmp` 資料夾中建立不同的日誌檔案。`/tmp` 資料夾下所建立的任何檔案都會上傳至 S3 中的結果位置作為成品。

如何使用 Synthetics 程式庫記錄器：

```
const log = require('SyntheticsLogger');
```

實用的函數定義：

log.debug(***message***, ***ex***);

參數：*message* 是要記錄的訊息。*ex* 要記錄的任何例外狀況 (如果有的話)。

範例：

```
log.debug("Starting step - login.");
```

log.error(***message***, ***ex***);

參數：*message* 是要記錄的訊息。*ex* 要記錄的任何例外狀況 (如果有的話)。

範例：

```
try {
  await login();
catch (ex) {
  log.error("Error encountered in step - login.", ex);
}
```

log.info(***message***, ***ex***);

參數：*message* 是要記錄的訊息。*ex* 要記錄的任何例外狀況 (如果有的話)。

範例：

```
log.info("Successfully completed step - login.");
```

log.log(***message***, ***ex***);

這是 log.info 的別名。

參數：*message* 是要記錄的訊息。*ex* 要記錄的任何例外狀況 (如果有的話)。

範例：

```
 log.log("Successfully completed step - login.");
```

log.warn(***message***, ***ex***);

參數：*message* 是要記錄的訊息。*ex* 要記錄的任何例外狀況 (如果有的話)。

範例：

```
log.warn("Exception encountered trying to publish CloudWatch Metric.", ex);
```

SyntheticsLogHelper 類

SyntheticsLogHelper 類別在執行時間 syn-nodejs-puppeteer-3.2 和更新版本的執行時間中可用。它已經在 CloudWatch 合成材料庫中初始化，並配置了 Synthetics 配置。您可以在指令碼中將其新增為相依性。此類別可讓您淨化 URL、標頭和錯誤訊息，以修訂敏感資訊。

> **Note**
>
> Synthetics 會根據 Synthetics 器組態設定 restrictedUrlParameters，淨化其記錄的所有 URL 和錯誤訊息，然後在將它們納入日誌、報告、HAR 檔案和 Canary 執行錯誤。僅當您在指令碼中記錄 URL 或錯誤時，才必須使用 getSanitizedUrl 或 getSanitizedErrorMessage。Synthetics 不會存放任何 Canary 成品，除了指令碼擲出的 Canary 錯誤。Canary 執行成品會存放於您的客戶帳戶中。如需詳細資訊，請參閱 Synthetics Canary 的安全考量。

getSanitizedUrl（網址，步驟配置 = 空）

此功能在 syn-nodejs-puppeteer-3.2 和更新版本中可用。它會傳回基於配置淨化的 url 字串。您可以透過設定屬性 restrictedUrlParameters 來選擇修訂敏感的 URL 參數，例如密碼和 access_token。預設情況下，URL 中的密碼已經修訂。如果需要，您可以啟用 URL 密碼，方法是將 includeUrlPassword 設定為 true。

如果傳入的 URL 不是有效的 URL，則此函數會擲回錯誤。

參數

- *url* 是一個字串，且是要淨化的 URL。
- *stepConfig* (選用) 會覆寫此函數的全域 Synthetics 組態。如果沒有傳入 stepConfig，則全域組態可用於淨化 URL。

範例

這個範例會使用下列範本 URL：https://example.com/learn/home?access_token=12345&token_type=Bearer&expires_in=1200。在此範例中，access_token 包含您不應該記錄的敏感資訊。請注意，Synthetics 服務不會存放任何 Canary 執行成品。日誌、螢幕擷取畫面和報告等成品都儲存在客戶帳戶中的 Simple Storage Service (Amazon S3) 儲存貯體中。

第一步是設定 Synthetics 組態。

```
// Import Synthetics dependency
const synthetics = require('Synthetics');

// Import Synthetics logger for logging url
const log = require('SyntheticsLogger');

// Get Synthetics configuration
const synConfig = synthetics.getConfiguration();

// Set restricted parameters
synConfig.setConfig({
   restrictedUrlParameters: ['access_token'];
});
```

接下來，淨化並記錄 URL

```
// Import SyntheticsLogHelper dependency
const syntheticsLogHelper = require('SyntheticsLogHelper');

const sanitizedUrl = synthetics.getSanitizedUrl('https://example.com/learn/home?
access_token=12345&token_type=Bearer&expires_in=1200');
```

這樣一來，將在您的 Canary 日誌中記錄以下內容。

```
My example url is: https://example.com/learn/home?
access_token=REDACTED&token_type=Bearer&expires_in=1200
```

您可以傳入包含 Synthetics 組態選項的選用參數，以覆寫 URL 的 Synthetics 組態，如下列範例所示。

```
const urlConfig = {
   restrictedUrlParameters = ['*']
};
const sanitizedUrl = synthetics.getSanitizedUrl('https://example.com/learn/home?
access_token=12345&token_type=Bearer&expires_in=1200', urlConfig);
logger.info('My example url is: ' + sanitizedUrl);
```

上述範例會修訂所有查詢參數，並記錄如下：

```
My example url is: https://example.com/learn/home?
access_token=REDACTED&token_type=REDACTED&expires_in=REDACTED
```

getSanitizedError訊息

此功能在 `syn-nodejs-puppeteer-3.2` 和更新版本中可用。它可透過淨化基於 Synthetics 組態存在的任何 URL 傳回淨化的錯誤字串。當您呼叫此函數時，您可以選擇覆寫全域 Synthetics 組態，方法是傳遞選用 `stepConfig` 參數。

參數

- *##*是要淨化的錯誤。它可以是一個 Error 物件或一個字串。
- *`stepConfig`* (選用) 會覆寫此函數的全域 Synthetics 組態。如果沒有傳入 `stepConfig`，則全域組態可用於淨化 URL。

範例

此範例使用下列錯誤：`Failed to load url: https://example.com/learn/home?access_token=12345&token_type=Bearer&expires_in=1200`

第一步是設定 Synthetics 組態。

```
// Import Synthetics dependency
const synthetics = require('Synthetics');

// Import Synthetics logger for logging url
const log = require('SyntheticsLogger');

// Get Synthetics configuration
const synConfig = synthetics.getConfiguration();

// Set restricted parameters
synConfig.setConfig({
   restrictedUrlParameters: ['access_token'];
});
```

接下來，淨化並記錄錯誤消息

```
// Import SyntheticsLogHelper dependency
const syntheticsLogHelper = require('SyntheticsLogHelper');

try {
   // Your code which can throw an error containing url which your script logs
} catch (error) {
    const sanitizedErrorMessage = synthetics.getSanitizedErrorMessage(errorMessage);
    logger.info(sanitizedErrorMessage);
}
```

這樣一來，將在您的 Canary 日誌中記錄以下內容。

```
Failed to load url: https://example.com/learn/home?
access_token=REDACTED&token_type=Bearer&expires_in=1200
```

getSanitizedHeaders（標題，步驟配置 = 空）

此功能在 `syn-nodejs-puppeteer-3.2` 和更新版本中可用。它會傳回基於 `syntheticsConfiguration` 的 `restrictedHeaders` 屬性而淨化的標頭。`restrictedHeaders` 屬性中指定的標頭會從日誌、HAR 檔案和報告中進行修訂。

參數

- *##*是一個包含要淨化的標頭的物件。
- *stepConfig* (選用) 會覆寫此函數的全域 Synthetics 組態。如果 stepConfig 沒有傳入，則全域組態可用於淨化標頭。

僅適用於 UI Canary 的 Node.js 程式庫類別和函數

以下 Node.js 的 CloudWatch Synthetics 庫函數僅適用於 UI 金絲雀。

主題

- Synthetics 類別
- BrokenLinkCheckerReport 類
- SyntheticsLink 類

Synthetics 類別

以下函數都在 Synthetics 類別中。

異步 addUserAgent (頁， userAgentString) ;

此函數會附加*userAgentString*到指定頁面的使用者代理標頭。

範例：

```
await synthetics.addUserAgent(page, "MyApp-1.0");
```

頁面使用者代理程式標頭中的結果設為 *browsers-user-agent-header-value*MyApp-1.0

異步 executeStep (步驟名， functionToExecute[步驟配置]) ;

執行提供的步驟，使用開始/通過/失敗記錄、開始/通過/失敗螢幕擷取畫面及通過/失敗和持續時間指標進行包裝。

> **Note**
>
> 如果您是使用 syn-nodejs-2.1 或更新版本的執行時間，您可以設定是否擷取螢幕擷取畫面以及何時擷取。如需詳細資訊，請參閱 SyntheticsConfiguration 類。

executeStep 函數還會執行下列操作：

- 記錄步驟已開始。
- 建立名為 <stepName>-starting 的螢幕擷取畫面。
- 啟動計時器。
- 執行提供的函數。
- 如果函數正常傳回結果，則會計為通過。如果函數擲回錯誤，則會計為失敗。
- 結束計時器。
- 記錄步驟是否已通過或失敗
- 建立名為 <stepName>-succeeded 或 <stepName>-failed 的螢幕擷取畫面。
- 發出 stepName SuccessPercent 指標，100 代表通過，0 代表失敗。
- 發出 stepName Duration 指標，顯示根據步驟開始和結束時間的值。
- 最後，傳回 functionToExecute 已傳回的結果，或重新擲回 functionToExecute 已擲回的錯誤。

如果 canary 使用 syn-nodejs-2.0 執行時間或更新版本，此函數也會將步驟執行摘要新增至 Canary 報告。摘要包括每個步驟的詳細資訊，例如開始時間、結束時間、狀態 (通過/失敗)、失敗原因 (如果失敗)，以及每個步驟執行期間擷取的螢幕擷取畫面。

範例：

```
await synthetics.executeStep('navigateToUrl', async function (timeoutInMillis = 30000)
 {
           await page.goto(url, {waitUntil: ['load', 'networkidle0'], timeout:
 timeoutInMillis});});
```

回應：

傳回 functionToExecute 所傳回的結果。

syn-nodejs-2.2 更新

從開始syn-nodejs-2.2，您可以選擇性地傳遞步驟組態，以覆寫 CloudWatch 步驟層級的合成材料組態。如需可傳遞至 executeStep 的選項清單，請參閱 SyntheticsConfiguration 類。

下列範例會將 continueOnStepFailure 的預設 false 組態覆寫為 true，並指定何時擷取螢幕擷取畫面。

```
var stepConfig = {
    'continueOnStepFailure': true,
    'screenshotOnStepStart': false,
    'screenshotOnStepSuccess': true,
    'screenshotOnStepFailure': false
}

await executeStep('Navigate to amazon', async function (timeoutInMillis = 30000) {
      await page.goto(url, {waitUntil: ['load', 'networkidle0'], timeout:
 timeoutInMillis});
 }, stepConfig);
```

getDefaultLaunch選項（ ）；

該getDefaultLaunchOptions( )函數返回 S CloudWatch ynthetics 使用的瀏覽器啟動選項。如需詳細資訊，請參閱 LaunchOptions type

```
// This function returns default launch options used by Synthetics.
const defaultOptions = await synthetics.getDefaultLaunchOptions();
```

getPage();

以 Puppeteer 物件形式傳回目前已開啟的頁面。如需詳細資訊，請參閱 Puppeteer API v1.14.0。

範例：

```
let page = synthetics.getPage();
```

回應：

在目前瀏覽器工作階段中目前開啟的頁面 (Puppeteer 物件)。

getRequestResponseLogHelper();

> **Important**
>
> 在使用 `syn-nodejs-puppeteer-3.2` 執行時間或更新版本的 Canary 中，此函數會與 `RequestResponseLogHelper` 類別一起被取代。此函數的任何用法都會導致您的 Canary 日誌中出現警告。未來的執行時間版本中將會移除此函數。如果您正在使用此函數，請改用 RequestResponseLogHelper 類。

使用此函數作為建置器模式，調整請求和回應日誌旗標。

範例：

```
synthetics.setRequestResponseLogHelper(getRequestResponseLogHelper().withLogRequestHeaders(fals
```

回應：

```
{RequestResponseLogHelper}
```

launch(options)

此函數的選項僅適用於 `syn-nodejs-2.1` 執行時間版本或更新版本。

此函數僅用於 UI Canary。它會關閉現有的瀏覽器並啟動一個新的瀏覽器。

> **Note**
>
> CloudWatch 在開始運行腳本之前，Synthetics 始終會啟動瀏覽器。除非您想啟動具有自訂選項的新瀏覽器，否則不需要呼叫 launch()。

(選項) 是可設定選項集，可在瀏覽器上設定。如需詳細資訊，請參閱 LaunchOptions type。

如果您在沒有選項的情況下呼叫此函數，Synthetics 會啟動具有預設引數、`executablePath` 和 `defaultViewport` 的瀏覽器。 CloudWatch Synthetics 中的默認視口是 1080 之前的 1920。

您可以覆蓋 CloudWatch Synthetics 使用的啟動參數，並在啟動瀏覽器時傳遞其他參數。例如，下面的程式碼片段啟動瀏覽器，具有預設引數和預設的可執行文件路徑，但具有 800 x 600 的檢視口。

```
await synthetics.launch({
        defaultViewport: {
            "deviceScaleFactor": 1,
            "width": 800,
            "height": 600
    }});
```

下列範例程式碼會將新`ignoreHTTPSErrors`參數新增至 S CloudWatch ynthetics 啟動參數：

```
await synthetics.launch({
```

```
        ignoreHTTPSErrors: true
 });
```

您可以通過在 CloudWatch Synthetics 啟動參數中添加`--disable-web-security`標誌來禁用 Web 安全性：

```
// This function adds the --disable-web-security flag to the launch parameters
const defaultOptions = await synthetics.getDefaultLaunchOptions();
const launchArgs = [...defaultOptions.args, '--disable-web-security'];
await synthetics.launch({
     args: launchArgs
  });
```

RequestResponseLogHelper 類

> **Important**
>
> 在使用 `syn-nodejs-puppeteer-3.2` 執行時間或更新版本的 Canary 中，此類別會被取代。此類別的任何用法都會導致您的 Canary 日誌中出現警告。未來的執行時間版本中將會移除此函數。如果您正在使用此函數，請改用 RequestResponseLogHelper 類。

處理精細定義的組態以及請求和回應承載字串顯示方式的建立。

```
class RequestResponseLogHelper {

    constructor () {
        this.request = {url: true, resourceType: false, method: false, headers: false,
 postData: false};
        this.response = {status: true, statusText: true, url: true, remoteAddress:
 false, headers: false};
    }

    withLogRequestUrl(logRequestUrl);

    withLogRequestResourceType(logRequestResourceType);

    withLogRequestMethod(logRequestMethod);

    withLogRequestHeaders(logRequestHeaders);
```

```
    withLogRequestPostData(logRequestPostData);


    withLogResponseStatus(logResponseStatus);

    withLogResponseStatusText(logResponseStatusText);

    withLogResponseUrl(logResponseUrl);

    withLogResponseRemoteAddress(logResponseRemoteAddress);

    withLogResponseHeaders(logResponseHeaders);
```

範例：

```
synthetics.setRequestResponseLogHelper(getRequestResponseLogHelper()
.withLogRequestPostData(true)
.withLogRequestHeaders(true)
.withLogResponseHeaders(true));
```

回應：

```
{RequestResponseLogHelper}
```

setRequestResponseLogHelper();

> **Important**
>
> 在使用 `syn-nodejs-puppeteer-3.2` 執行時間或更新版本的 Canary 中，此函數會與 `RequestResponseLogHelper` 類別一起被取代。此函數的任何用法都會導致您的 Canary 日誌中出現警告。未來的執行時間版本中將會移除此函數。如果您正在使用此函數，請改用 RequestResponseLogHelper 類。

使用此函數作為建置器模式，設定請求和回應日誌旗標。

範例：

```
synthetics.setRequestResponseLogHelper().withLogRequestHeaders(true).withLogResponseHeaders(tru
```

回應：

```
{RequestResponseLogHelper}
```

async takeScreenshot(name, suffix);

使用名稱和尾碼 (選用)，擷取目前頁面的螢幕擷取畫面 (.PNG)。

範例：

```
await synthetics.takeScreenshot("navigateToUrl", "loaded")
```

此範例會捕獲並上傳名為 `01-navigateToUrl-loaded.png` 的螢幕擷取畫面至 Canary 的 S3 儲存貯體。

您可以透過將 `stepName` 傳遞為第一個參數，擷取特定 Canary 的螢幕擷取畫面。螢幕擷取畫面會連結至報告中的 Canary 步驟，以協助您在偵錯時追蹤每個步驟。

CloudWatch Synthetics 成金絲雀會在開始步驟（`executeStep`功能）之前和步驟完成後自動拍攝屏幕截圖（除非您配置金絲雀以禁用屏幕截圖）。您可以透過在 `takeScreenshot` 函數中傳入步驟名稱，來擷取更多的螢幕擷取畫面。

下列範例以 `signupForm` 為 `stepName` 的值擷取了螢幕擷取畫面。螢幕擷取畫面會命名為`02-signupForm-address`，並將連結到 Canary 報告中名為 `signupForm` 的步驟

```
await synthetics.takeScreenshot('signupForm', 'address')
```

BrokenLinkCheckerReport 類

此類別提供了新增綜合連結的方法。它只支援使用執行時間的 `syn-nodejs-2.0-beta` 版本或更新版本的 Canary。

若要使用 `BrokenLinkCheckerReport`，請在指令碼中包含下列幾行：

```
const BrokenLinkCheckerReport = require('BrokenLinkCheckerReport');

const brokenLinkCheckerReport = new BrokenLinkCheckerReport();
```

實用的函數定義：

addLink(*syntheticsLink*, isBroken)

syntheticsLink 是可代表連結的 SyntheticsLink 物件。該函數可根據狀態碼新增連結。根據預設，如果狀態碼不可用或狀態碼為 400 或更高，則會將連結視為中斷。您可以透過傳入值為 `true` 或 `false` 的選用參數 `isBrokenLink` 來覆寫此預設行為。

此函數沒有傳回值。

getLinks()

此函數會傳回包含在中斷連結檢查程式報告中的 SyntheticsLink 物件陣列。

getTotalBroken連結 ()

該函數會傳回表示中斷連結總數的數字。

getTotalLinks勾選（ ）

該函數會傳回一個數字，表示報告中包含的中斷連結總數。

如何使用 BrokenLinkCheckerReport

下面的 Canary 指令碼程式碼片段示範了導覽至連結，並將其新增至中斷連結檢查程式報告的範例。

1. 匯入 `SyntheticsLink`、`BrokenLinkCheckerReport` 及 `Synthetics`。

```
const BrokenLinkCheckerReport = require('BrokenLinkCheckerReport');
const SyntheticsLink = require('SyntheticsLink');

// Synthetics dependency
const synthetics = require('Synthetics');
```

2. 若要將連結新增至報告，請建立 `BrokenLinkCheckerReport` 的執行個體。

```
let brokenLinkCheckerReport = new BrokenLinkCheckerReport();
```

3. 導覽至 URL 並將其新增至中斷連結檢查程式報告。

```
let url = "https://amazon.com";

let syntheticsLink = new SyntheticsLink(url);

// Navigate to the url.
```

```
let page = await synthetics.getPage();

// Create a new instance of Synthetics Link
let link = new SyntheticsLink(url)

try {
    const response = await page.goto(url, {waitUntil: 'domcontentloaded', timeout:
 30000});
} catch (ex) {
    // Add failure reason if navigation fails.
    link.withFailureReason(ex);
}

if (response) {
    // Capture screenshot of destination page
    let screenshotResult = await synthetics.takeScreenshot('amazon-home', 'loaded');

    // Add screenshot result to synthetics link
    link.addScreenshotResult(screenshotResult);

    // Add status code and status description to the link
    link.withStatusCode(response.status()).withStatusText(response.statusText())
}

// Add link to broken link checker report.
brokenLinkCheckerReport.addLink(link);
```

4. 將報告新增至 Synthetics。如此一來，將針對每個 Canary 執行在您的 S3 儲存貯體中建立一個名為 `BrokenLinkCheckerReport.json` JSON 檔案。您可以在主控台中看到每個 Canary 執行的連結報告，以及螢幕擷取畫面、日誌和 HAR 檔案。

```
await synthetics.addReport(brokenLinkCheckerReport);
```

SyntheticsLink 類

此類別提供了可包裝資訊的方法。它只支援使用執行時間的 `syn-nodejs-2.0-beta` 版本或更新版本的 Canary 。

若要使用 `SyntheticsLink`，請在指令碼中包含下列幾行：

```
const SyntheticsLink = require('SyntheticsLink');
```

```
const syntheticsLink = new SyntheticsLink("https://www.amazon.com");
```

此函數會傳回 syntheticsLink*Object*

實用的函數定義：

withUrl(***url***)

url 是 URL 字串。此函數會傳回 syntheticsLink*Object*

withText(***text***)

text 是可表示錨點文字的字串。此函數會傳回 syntheticsLink*Object*。它增加了對應於連結的錨點文本。

withParentUrl(***##)***

parentUrl 是可表示父 (源頁面) URL 的字串。此函數會傳回 syntheticsLink*Object*

withStatusCode(***statusCode***)

statusCode 是表示狀態碼的字串。此函數會傳回 syntheticsLink*Object*

withFailureReason(***###***因)

failureReason 是表示失敗原因的字串。此函數會傳回 syntheticsLink*Object*

addScreenshotResult (***##***結果)

screenshotResult 是物件。它是由 Synthetics 函數 takeScreenshot 傳回的 ScreenshotResult 的執行個體。物件包含以下屬性：

- fileName— 表示 screenshotFileName 的字串
- pageUrl (選用)
- error (選用)

僅適用於 API Canary 的 Node.js 程式庫類別和函數

以下 Node.js 的 CloudWatch Synthetics 庫函數僅適用於 API 金絲雀。

主題

- executeHttpStep（stepName，請求選項，[回調]，[步驟配置]）

executeHttpStep（stepName，請求選項，[回調]，[步驟配置]）

執行提供的 HTTP 請求作為一個步驟，並發佈 `SuccessPercent` (通過/失敗) 和 `Duration` 指標。

executeHttpStep根據請求中指定的協議，在引擎蓋下使用 HTTP 或 HTTPS 本地函數。

此函數也會將步驟執行摘要新增至 Canary 報告。摘要包含每個 HTTP 請求的詳細資訊，如下所示：

- 開始時間
- 結束時間
- 狀態 (通過/失敗)
- 失敗的原因，如果失敗
- HTTP 呼叫詳細資訊，如請求/回應標頭、內文、狀態碼、狀態訊息和效能計時。

參數

stepName(***`String`***)

指定步驟的名稱。此名稱也可用於發佈此步驟的 CloudWatch 量度。

requestOptions(***`Object or String`***)

此參數的值可以是 URL、URL 字串或物件。如果它是一個物件，那麼它必須是可設定的選項組，以發出 HTTP 請求。它支援 Node.js 文件中的 http.request(options[, callback]) 的所有選項。

除了這些 Node.js 選項之外，requestOptions 支援其他參數 body。您可以使用 body 參數將資料作為請求內文進行傳遞。

callback(***`response`***)

(選用) 這是使用 HTTP 回應呼叫的使用者函數。回應的類型為類別：http。 IncomingMessage。

stepConfig(***`object`***)

(選用) 使用此參數，以此步驟的不同組態覆寫全域綜合組態。

使用示例 executeHttpStep

以下範例系列會建立會相互建置，以說明此選項的各種用途。

這第一個範例設定了請求參數。您可以將 URL 傳遞請求選項：

```
let requestOptions = 'https://www.amazon.com';
```

或者你可以傳遞選項集：

```
let requestOptions = {
        'hostname': 'myproductsEndpoint.com',
        'method': 'GET',
        'path': '/test/product/validProductName',
        'port': 443,
        'protocol': 'https:'
    };
```

下一個範例會建立接受回應的回呼函數。默認情況下，如果您沒有指定回調， CloudWatch Synthetics 驗證狀態是否介於 200 和 299 之間。

```
// Handle validation for positive scenario
    const callback = async function(res) {
        return new Promise((resolve, reject) => {
            if (res.statusCode < 200 || res.statusCode > 299) {
                throw res.statusCode + ' ' + res.statusMessage;
            }

            let responseBody = '';
            res.on('data', (d) => {
                responseBody += d;
            });

            res.on('end', () => {
                // Add validation on 'responseBody' here if required. For ex, your
 status code is 200 but data might be empty
                resolve();
            });
        });
    };
```

下一個範例會為此步驟建立一個模型組態，以覆寫全域 CloudWatch Synthetics 組態。此範例中的步驟組態允許在您的報告中包含請求標頭、回應標頭、請求內文 (發佈資料) 和回應內文，並限制 'X-Amz-Security-Token' 和 'Authorization' 標頭值。根據預設，基於安全性考量，這些值不會包含在報告中。如果選擇包含它們，則資料只會存放於 S3 儲存貯體中。

```
// By default headers, post data, and response body are not included in the report for
 security reasons.
// Change the configuration at global level or add as step configuration for individual
 steps
let stepConfig = {
    includeRequestHeaders: true,
    includeResponseHeaders: true,
    restrictedHeaders: ['X-Amz-Security-Token', 'Authorization'], // Restricted header
 values do not appear in report generated.
    includeRequestBody: true,
    includeResponseBody: true
};
```

此最後一個範例會將您的要求傳送至步驟executeHttpStep並命名該步驟。

```
await synthetics.executeHttpStep('Verify GET products API', requestOptions, callback,
 stepConfig);
```

透過這組範例， CloudWatch Synthetics 會新增報告中每個步驟的詳細資訊，並使用 step Name 為每個步驟產生量度。

您將看見適用於 `Verify GET products API` 步驟的 `successPercent` 和`duration`的 指標步驟。您可以監控 API 呼叫步驟的指標，以監控您的 API 效能。

如需使用這些函數的完整指令碼範例，請參閱 多步驟 API Canary。

可用於使用 Selenium 的 Python Canary 指令碼的程式庫函數

本區段列出了適用於 Python Canary 指令碼的 Selenium 程式庫函數

主題

- 適用於所有 Canary 的 Python 和 Selenium 程式庫類別和函數
- 僅適用於 UI Canary 的 Python 和 Selenium 程式庫類別和函數

適用於所有 Canary 的 Python 和 Selenium 程式庫類別和函數

下面的 CloudWatch Synthetics 硒庫函數為 Python 是所有金絲雀有用。

主題

- SyntheticsConfiguration 類
- SyntheticsLogger 類

SyntheticsConfiguration 類

您可以使用 SyntheticsConfiguration 類來配置 Synthetics 庫函數的行為。例如，您可以使用此類別來將 `executeStep()` 函數設定為不擷取螢幕擷取畫面。

您可以在全局級別設置 CloudWatch Synthetics 配置。

函數定義：

set_config(options)

```
from aws_synthetics.common import synthetics_configuration
```

`options` 是一個物件，適用於您的 Canary 的可設定選項集。以下各節會說明了 *`options`* 中的可能欄位。

- `screenshot_on_step_start` (布林值)— 是否在開始步驟之前擷取螢幕擷取畫面。
- `screenshot_on_step_success` (布林值)— 是否在成功完成步驟之後擷取螢幕擷取畫面。
- `screenshot_on_step_failure` (布林值)— 是否在步驟失敗之後擷取螢幕擷取畫面。

with_screenshot_on_step_start(screenshot_on_step_start)

接受布林值引數，它會指示是否在開始步驟之前擷取螢幕擷取畫面。

with_screenshot_on_step_success(screenshot_on_step_success)

接受布林值引數，它會指示是否在成功完成步驟之後擷取螢幕擷取畫面。

with_screenshot_on_step_failure(screenshot_on_step_failure)

接受布林值引數，它會指示是否在步驟失敗之後擷取螢幕擷取畫面。

get_screenshot_on_step_start()

傳回是否在開始步驟之前擷取螢幕擷取畫面。

get_screenshot_on_step_success()

傳回是否在成功完成步驟之後擷取螢幕擷取畫面。

get_screenshot_on_step_failure()

傳回是否在步驟失敗之後擷取螢幕擷取畫面。

disable_step_screenshots()

停用所有螢幕擷取畫面選項 (get_screenshot_on_step_start、get_screenshot_on_step_success 和 get_screenshot_on_step_failure)。

enable_step_screenshots()

啟用所有螢幕擷取畫面選項 (get_screenshot_on_step_start、get_screenshot_on_step_success 和 get_screenshot_on_step_failure)。預設不會啟用所有這些方法。

設定設定 (選項) 有關 CloudWatch 度量

對於使用 `syn-python-selenium-1.1` 或更新版本的 Canary，setConfig 的 (選項) 可以包含以下布林值參數，可確定 Canary 發佈哪些指標。這些選項的預設值為 `true`。以 `aggregated` 開頭的選項可判斷是否會發出沒有 `CanaryName` 維度的指標。您可以使用這些指標來查看所有 Canary 的彙總結果。其他選項可判斷是否會發出具有 `CanaryName` 維度的指標。您可以使用這些指標來查看每個 Canary 的結果。

如需加那利群島發出的 CloudWatch 指標清單，請參閱。CloudWatch 加那利群島發布的指標

- `failed_canary_metric` (布林值)— 是否為此 canary 發出 `Failed` 指標 (具有 `CanaryName` 維度)。預設值為 `true`。
- `failed_requests_metric` (布林值)— 是否為此 canary 發出 `Failed requests` 指標 (具有 `CanaryName` 維度)。預設值為 `true`。
- `2xx_metric` (布林值)— 是否為此 canary 發出 `2xx` 指標 (具有 `CanaryName` 維度)。預設值為 `true`。
- `4xx_metric` (布林值)— 是否為此 canary 發出 `4xx` 指標 (具有 `CanaryName` 維度)。預設值為 `true`。
- `5xx_metric` (布林值)— 是否為此 canary 發出 `5xx` 指標 (具有 `CanaryName` 維度)。預設值為 `true`。

- step_duration_metric (布林值)— 是否為此 canary 發出 Step duration 指標 (具有 CanaryName StepName 維度)。預設值為 true。
- step_success_metric (布林值)— 是否為此 canary 發出 Step success 指標 (具有 CanaryName StepName 維度)。預設值為 true。
- aggregated_failed_canary_metric (布林值)— 是否為此 canary 發出 Failed 指標 (沒有 CanaryName 維度)。預設值為 true。
- aggregated_failed_requests_metric (布林值)— 是否為此 canary 發出 Failed Requests 指標 (沒有 CanaryName 維度)。預設值為 true。
- aggregated_2xx_metric (布林值)— 是否為此 canary 發出 2xx 指標 (沒有 CanaryName 維度)。預設值為 true。
- aggregated_4xx_metric (布林值)— 是否為此 canary 發出 4xx 指標 (沒有 CanaryName 維度)。預設值為 true。
- aggregated_5xx_metric (布林值)— 是否為此 canary 發出 5xx 指標 (沒有 CanaryName 維度)。預設值為 true。

with_2xx_metric(2xx_metric)

接受布林值引數，它會指定是否為此 Canary 發出具有 CanaryName 維度的 2xx 指標。

with_4xx_metric(4xx_metric)

接受布林值引數，它會指定是否為此 Canary 發出具有 CanaryName 維度的 4xx 指標。

with_5xx_metric(5xx_metric)

接受布林值引數，它會指定是否為此 Canary 發出具有 CanaryName 維度的 5xx 指標。

withAggregated2xxMetric(aggregated2xxMetric)

接受布林值引數，它會指定是否為此 Canary 發出沒有維度的 2xx 指標。

withAggregated4xxMetric(aggregated4xxMetric)

接受布林值引數，它會指定是否為此 Canary 發出沒有維度的 4xx 指標。

with_aggregated_5xx_metric(aggregated_5xx_metric)

接受布林值引數，它會指定是否為此 Canary 發出沒有維度的 5xx 指標。

with_aggregated_failed_canary_metric(aggregated_failed_canary_metric)

接受布林值引數，它會指定是否為此 Canary 發出沒有維度的 Failed 指標。

with_aggregated_failed_requests_metric(aggregated_failed_requests_metric)

接受布林值引數，它會指定是否為此 Canary 發出沒有維度的 Failed requests 指標。

with_failed_canary_metric(failed_canary_metric)

接受布林值引數，它會指定是否為此 Canary 發出具有 CanaryName 維度的 Failed 指標。

with_failed_requests_metric(failed_requests_metric)

接受布林值引數，它會指定是否為此 Canary 發出具有 CanaryName 維度的 Failed requests 指標。

with_step_duration_metric(step_duration_metric)

接受布林值引數，它會指定是否為此 Canary 發出具有 CanaryName 維度的 Duration 指標。

with_step_success_metric(step_success_metric)

接受布林值引數，它會指定是否為此 Canary 發出具有 CanaryName 維度的 StepSuccess 指標。

啟用或停用指標的方法

disable_aggregated_request_metrics()

禁止 Canary 發出所有沒有 CanaryName 維度發出的請求指標。

disable_request_metrics()

停用所有請求指標，包括每個 Canary 指標和彙總所有 Canary 的指標。

disable_step_metrics()

停用所有步驟指標，包括步驟成功指標和步驟持續時間指標。

enable_aggregated_request_metrics()

啟用 Canary，使其可以發出所有沒有 CanaryName 維度發出的請求指標。

enable_request_metrics()

啟用所有請求指標，包括每個 Canary 指標和彙總所有 Canary 的指標。

enable_step_metrics()

啟用所有步驟指標，包括步驟成功指標和步驟持續時間指標。

在 UI Canary 中的使用方式

首先，匯入綜合相依性並擷取組態。然後，使用下列其中一個選項呼叫 setConfig 方法來設定每個選項的組態。

```
from aws_synthetics.common import synthetics_configuration

synthetics_configuration.set_config(
     {
        "screenshot_on_step_start": False,
        "screenshot_on_step_success": False,
        "screenshot_on_step_failure": True
     }
)

or
```

或

```
synthetics_configuration.with_screenshot_on_step_start(False).with_screenshot_on_step_success(F
```

若要停用所有螢幕擷取畫面，請使用 disableStepScreenshots () 函數，如本範例所示。

```
synthetics_configuration.disable_step_screenshots()
```

您可以在程式碼中的任何時候啟用和停用螢幕擷取畫面。例如，若只要停用一個步驟的螢幕擷取畫面，請在執行該步驟之前停用這些螢幕擷取畫面，然後在步驟之後予以啟用。

適用於 UI Canary 的 set_config(options)

對於 UI Canary，以 `syn-python-selenium-1.1` 為開頭的 `set_config` 可以包括下列布林值參數：

- continue_on_step_failure (布林值)— 是否在步驟失敗之後繼續執行 Canary 指令碼 (其代表 executeStep 函數)。如果任何步驟失敗，Canary 執行仍將被標記為失敗。預設值為 false。

SyntheticsLogger 類

synthetics_logger 會將日誌同時寫入至主控台和相同日誌層級的本機日誌檔案。只有在日誌層級等於或低於呼叫之日誌函數所需的日誌層級時，才會將此日誌檔案同時寫入至這兩個位置。

本機日誌檔案中的日誌陳述式會加上「偵錯」、「資訊」等前綴，以符合呼叫之函數的日誌層級。

使用 synthetics_logger 並不需要建立上傳至 Simple Storage Service (Amazon S3) 結果位置的日誌檔案。您可改為在 /tmp 資料夾中建立不同的日誌檔案。/tmp 資料夾下所建立的任何檔案都會上傳至 S3 儲存貯體中的結果位置作為成品。

若要使用 synthetics_logger:

```
from aws_synthetics.common import synthetics_logger
```

實用的函數定義：

取得日誌層級：

```
log_level = synthetics_logger.get_level()
```

設定日誌層級：

```
synthetics_logger.set_level()
```

記錄具有指定層級的訊息。該層級可以是 DEBUG、INFO、WARN 或 ERROR，如下列語法範例所示：

```
synthetics_logger.debug(message, *args, **kwargs)
```

```
synthetics_logger.info(message, *args, **kwargs)
```

```
synthetics_logger.log(message, *args, **kwargs)
```

```
synthetics_logger.warn(message, *args, **kwargs)
```

```
synthetics_logger.error(message, *args, **kwargs)
```

如需偵錯參數的相關資訊，請參閱 logging.debug 中的標準的 Python 文件，網址為記錄 .debug。

在這些記錄函數中，message 是訊息格式字串。args 是使用字串格式化運算符合併到 msg 的引數。

kwargs 中有三個關鍵字引數：

- exc_info— 如果未評估為 false，則會將例外狀況資訊新增至記錄訊息。
- stack_info– 預設為 false。如果為 true，則將堆疊資訊新增至日誌訊息，包括實際的日誌記錄呼叫。
- extra— 第三個選用關鍵字引數，您可以使用此引數傳入字典，而該字典可用來使用使用者定義的屬性填入為日誌事件建立的 LogRecord 的 __dict__。

範例：

記錄 DEBUG 層級的訊息：

```
synthetics_logger.debug('Starting step - login.')
```

記錄 INFO 層級的據悉 。logger.log 是 logger.info 的同義詞：

```
synthetics_logger.info('Successfully completed step - login.')
```

或

```
synthetics_logger.log('Successfully completed step - login.')
```

記錄 WARN 層級的訊息：

```
synthetics_logger.warn('Warning encountered trying to publish %s', 'CloudWatch Metric')
```

記錄 ERROR 層級的訊息：

```
synthetics_logger.error('Error encountered trying to publish %s', 'CloudWatch Metric')
```

記錄例外：

```
synthetics_logger.exception(message, *args, **kwargs)
```

記錄 ERROR 層級的訊息。例外狀況資訊會新增至記錄訊息。您應該只從例外狀況處理常式呼叫此函數。

如需例外狀況參數的相關資訊，請參閱 logging.exception 中的標準的 Python 文件。

message 是訊息格式字串。args 是使用字串格式化運算符合併到 msg 的引數。

kwargs 中有三個關鍵字引數：

- exc_info— 如果未評估為 false，則會將例外狀況資訊新增至記錄訊息。
- stack_info– 預設為 false。如果為 true，則將堆疊資訊新增至日誌訊息，包括實際的日誌記錄呼叫。
- extra— 第三個選用關鍵字引數，您可以使用此引數傳入字典，而該字典可用來使用使用者定義的屬性填入為日誌事件建立的 LogRecord 的 __dict__。

範例：

```
synthetics_logger.exception('Error encountered trying to publish %s', 'CloudWatch
 Metric')
```

僅適用於 UI Canary 的 Python 和 Selenium 程式庫類別和函數

Python 下面的 CloudWatch Synthetics 硒庫函數僅用於 UI 金絲雀有用。

主題

- SyntheticsBrowser 類
- SyntheticsWebDriver 類

SyntheticsBrowser 類

當您透過呼叫 synthetics_webdriver.Chrome() 建立瀏覽器執行個體時，傳回的瀏覽器執行個體是類型 SyntheticsBrowser。該SyntheticsBrowser類控制 ChromeDriver，並使金絲雀腳本驅動瀏覽器，允許硒與 Synthetics WebDriver 一起工作。

除了標準 Selenium 方法，它還提供了以下方法。

set_viewport_size(width, height)

設定瀏覽器的檢視區。範例：

```
browser.set_viewport_size(1920, 1080)
```

save_screenshot(filename, suffix)

將螢幕擷取畫面儲存至 /tmp 目錄。螢幕擷取畫面從那裡上傳到 S3 儲存貯體中的 Canary 成品資料夾。

filename 是螢幕擷取畫面的檔案名稱，suffix 是用來命名螢幕擷取畫面的選用字串。

範例：

```
browser.save_screenshot('loaded.png', 'page1')
```

SyntheticsWebDriver 類

若要使用此類別，請在指令碼中使用以下內容：

```
from aws_synthetics.selenium import synthetics_webdriver
```

add_execution_error(errorMessage, ex);

errorMessage 描述錯誤，而 ex 是遇到的例外狀況

您可以使用 add_execution_error 為您的 Canary 設定執行錯誤。它會在不中斷指令碼執行的情況下使 Canary 失敗。它也不會影響您的 successPercent 指標。

只有在錯誤表示 Canary 指令碼成功或失敗並不重要時，才應將錯誤追蹤為執行錯誤。

add_execution_error 的使用範例如下所示。您正在監控端點的可用性，並在頁面載入後擷取螢幕擷取畫面。由於擷取螢幕擷取畫面的失敗並不會決定端點的可用性，因此您可以捕捉擷取螢幕擷取畫面時遇到的任何錯誤，並將它們新增為執行錯誤。您的可用性指標仍會顯示端點已啟動並執行，但您的 Canary 狀態會標示為失敗。下面的範本程式碼區塊可捕獲這樣的錯誤，並將其新增為執行錯誤。

```
try:
    browser.save_screenshot("loaded.png")
except Exception as ex:
   self.add_execution_error("Unable to take screenshot", ex)
```

add_user_agent(user_agent_str)

將 user_agent_str 的數值附加到瀏覽器的使用者代理程式標頭。您必須先指派 user_agent_str，再建立瀏覽器執行個體。

範例：

```
synthetics_webdriver.add_user_agent('MyApp-1.0')
```

execute_step(step_name, function_to_execute)

處理一個函數。它也會執行以下操作：

- 記錄步驟已開始。
- 建立名為 <stepName>-starting 的螢幕擷取畫面。
- 啟動計時器。
- 執行提供的函數。
- 如果函數正常傳回結果，則會計為通過。如果函數擲回錯誤，則會計為失敗。
- 結束計時器。
- 記錄步驟是否已通過或失敗
- 建立名為 <stepName>-succeeded 或 <stepName>-failed 的螢幕擷取畫面。
- 發出 stepName SuccessPercent 指標，100 代表通過，0 代表失敗。
- 發出 stepName Duration 指標，顯示根據步驟開始和結束時間的值。
- 最後，傳回 functionToExecute 已傳回的結果，或重新擲回 functionToExecute 已擲回的錯誤。

範例：

```
from selenium.webdriver.common.by import By

def custom_actions():
        #verify contains
        browser.find_element(By.XPATH, "//*[@id=\"id_1\"][contains(text(),'login')]")
        #click a button
        browser.find_element(By.XPATH, '//*[@id="submit"]/a').click()

    await synthetics_webdriver.execute_step("verify_click", custom_actions)
```

Chrome()

啟動 Chromium 瀏覽器的執行個體，並傳回建立的瀏覽器執行個體。

範例：

```
browser = synthetics_webdriver.Chrome()
browser.get("https://example.com/)
```

若要在無痕模式下啟動瀏覽器，請使用以下命令：

```
add_argument('—-incognito')
```

若要新增代理設定，請使用以下命令：

```
add_argument('--proxy-server=%s' % PROXY)
```

範例：

```
from selenium.webdriver.chrome.options import Options
chrome_options = Options()
chrome_options.add_argument("—-incognito")
browser = syn_webdriver.Chrome(chrome_options=chrome_options)
```

使用 Cron 排程 Canary 執行

當您排程 Canary 時，使用 Cron 表達式為您提供靈活性。Cron 表達式包含五個或六個欄位，其排序如下表所示。欄位以空格隔開。根據您是使用 CloudWatch 控制台創建初期測試還是 AWS CLI 或 AWS SDK，語法會有所不同。當您使用主控台時，只會指定前五個欄位。使用 AWS CLI 或 AWS SDK 時，您必須指定全部六個欄位，並且必須*為Year欄位指定。

欄位	允許的值	允許的特殊字元
分鐘	0-59	, - * /
小時	0-23	, - * /
D ay-of-month	1-31	, - * ? / L W
月	1-12 或 JAN-DEC	, - * /

欄位	允許的值	允許的特殊字元
D ay-of-week	1-7 或 SUN-SAT	, - * ? L #
年	*	

特殊字元

- , (逗號) 在欄位的表達式中包含多個值。例如，在 Month (月) 欄位，JAN、FEB、MAR 包括 January (一月)、February (二月) 與 March (三月)。
- - (破折號) 特殊字元用於指定範圍。在 Day (日) 欄位，1-15 包含指定月份的 1 至 15 號。
- * (星號) 特殊字元包含欄位中所有的值。在 Hours (小時) 欄位，* 包含每個小時。您無法在相同運算式的 D ay-of-month 和 D ay-of-week 欄位中使用 *。若您在其中一個欄位使用它，您必須在另一個欄位使用 ?。
- / (斜線) 用於指定增量。在 Minutes (分鐘) 欄位，您可以輸入 1/10 指定每十分鐘的間隔，從小時的第一分鐘開始 (例如第 11、第 21、第 31 分鐘等)。
- ? (問號) 用於表示不限定任何一個。如果您在 D ay-of-month 字段中輸入 7，並且不在乎第七週的哪一天，則可以輸入？ 在 D 字ay-of-week段中。
- D ay-of-month 或 D ay-of-week 欄位中的 L 萬用字元會指定月份或週的最後一天。
- D ay-of-month 欄位中的**W**萬用字元指定工作日。在 D ay-of-month 欄位中，**3W**指定最接近月份第三天的星期幾。
- D ay-of-week 欄位中的 # 萬用字元會指定一個月內星期中指定日期的特定執行個體。例如，3#2 是該月的第二個星期二。3 是指星期二，因為它是每週的第三天，2 指的是一個月內該類型的第二天。

限制

- 您無法在相同的 cron 運算式中指定 D ay-of-month 和 D ay-of-week 欄位。如果您在其中一個欄位指定了數值 或 * (星號)，就必須在另一個欄位中使用 ? (問號)。
- 不支援頻率多於一分鐘的 Cron 表達式。
- 您不能設置 Canary 在執行之前等待一年以上，所以您只能在 `Year` 欄位中指定 *。

範例

當您建立 Canary 時，您可以參考下列 Cron 字串範例。下列範例是使用 AWS CLI 或 AWS SDK 建立或更新初期測試的正確語法。如果您正在使用 CloudWatch 控制台，請*在每個示例中省略 final。

表達式	意義
`0 10 * * ? *`	在每天上午 10:00 (UTC) 執行
`15 12 * * ? *`	在每天下午 12:15 (UTC) 執行
`0 18 ? * MON-FRI *`	在每週一至週五下午 6:00 (UTC) 執行
`0 8 1 * ? *`	在每個月第一天上午 8:00 (UTC) 執行
`0/10 * ? * MON-SAT *`	在每週週一至週六每 10 分鐘執行
`0/5 8-17 ? * MON-FRI *`	在週一至週五上午 8:00 至下午 5:55 (UTC) 之間每 5 分鐘執行

群組

您可以建立群組，以將 Canary (包括跨區域的 Canary) 相互關聯。使用群組可以幫助您管理和自動化 Canary，您還可以查看一個群組中所有 Canary 的彙總執行結果和統計資料。

群組為全域資源。當您建立群組時，會在所有支援群組的 AWS 區域中複製該群組，而且您可以將這些區域中任何一個區域的金絲雀新增至該群組，並在這些區域中檢視該群組。雖然群組 ARN 格式會反映建立群組的區域名稱，不過群組不會限制於任何區域。這意味著您可以將來自多個區域的 Canary 放入同一群組中，然後使用該群組在單個視圖中檢視和管理所有這些 Canary。

除了預設停用的區域以外，所有區域都支援群組。如需有關這些區域的詳細資訊，請參閱啟用區域。

每個群組最多可包含 10 個 Canary。您的帳戶最多可有 20 個群組。任何一個 Canary 最多可為 10 個群組的成員。

建立群組

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，Synthetics 金絲雀。
3. 選擇 Create Group (建立群組)。
4. 在 Group Name (群組名稱) 下，輸入群組的名稱。
5. 選取要與此群組建立關聯的 Canary。若要選取 Canary，請在 Exact canary name (確切的 Canary 名稱) 中輸入其完整名稱，然後選擇 Search (搜尋)。選取 Canary 名稱旁的核取方塊。如果在不同的區域中有多個具有相同名稱的 Canary，請確保選取所需的 Canary。

您可以重複此步驟，將最多 10 個 Canary 與該群組建立關聯。

6. (選用) 在 Tags (標籤) 下，新增一或多個鍵/值對，作為此群組的標籤。標籤可以幫助您識別和組織 AWS 資源並追蹤 AWS 成本。如需詳細資訊，請參閱 標記您的 Amazon CloudWatch 資源。
7. 選擇 Create Group (建立群組)。

在本地測試金絲雀

本節介紹如何直接在代碼編輯器或Microsoft Visual Studio代碼編輯器中修改，測試和調試 CloudWatch Synthetics 金絲雀。JetBrains IDE本機偵錯環境使用無伺服器應用程式模型 (SAM) 容器來模擬 Lambda 函數，以模擬 Synthetics 初期測試的行為。

> **Note**
>
> 執行依賴於視覺監視的本地調試金絲雀是不切實際的。視覺監控依賴於在初始運行期間捕獲基本屏幕截圖，然後將這些屏幕截圖與後續運行的屏幕截圖進行比較。在本機開發環境中，不會儲存或追蹤執行，而且每個版序都是獨立的獨立執行。由於沒有金絲雀運行歷史記錄，因此對依賴於可視監視的金絲雀進行調試是不切實際的。

先決條件

1. 選擇或建立 Amazon S3 儲存貯體，用於存放來自本機初期測試執行的成品，例如 HAR 檔案和螢幕擷取畫面。這需要您使用 IAM 進行佈建。如果您略過設定 Amazon S3 儲存貯體，您仍然可以在本機測試初期測試，但是您會看到有關遺失儲存貯體的錯誤訊息，而且您將無法存取初期測試成品。

   如果您使用 Amazon S3 儲存貯體，建議您將儲存貯體生命週期設定為在幾天後刪除物件，以節省成本。如需詳細資訊，請參閱管理儲存生命週期。
2. 為您的 AWS 帳戶設定預設設定 AWS 檔。如需詳細資訊，請參閱組態和認證檔案設定。
3. 將除錯環境的預設 AWS 區域設定為您偏好的區域，例如`us-west-2`。
4. 安裝 AWS SAM CLI。如需詳細資訊，請參閱安裝 AWS SAM CLI。
5. 安裝Visual Studio Code Editor或JetBrains IDE。如需詳細資訊，請參閱 Visual Studio Code 或 JetBrains IDE
6. 安裝Docker以使用 AWS SAM CLI。確保啟動 docker 守護進程。如需詳細資訊，請參Docker閱安裝以搭配 AWS SAM CLI 使用。

   或者，您也可以安裝其他容器管理軟體，例如Rancher，只要它使用Docker執行階段即可。

7. 為您的首選編輯器安裝 AWS 工具包擴展。如需詳細資訊，請參閱安裝 AWS Toolkit for Visual Studio Code或安裝 AWS Toolkit for JetBrains.

主題

- 設置測試和調試環境
- 使用 Visual Studio Code IDE
- 使用 JetBrains IDE
- 使用 SAM CLI 在本機執行初期測試
- 將您的本地測試環境集成到現有的 Canary 軟件包中
- 變更 CloudWatch Synthetics 執行階段
- 常見錯誤

設置測試和調試環境

首先，克隆通過輸入以下命令 AWS 提供的 Github 存儲庫。該存儲庫包含 Node.js 金絲雀和 Python 金絲雀的代碼示例。

```
git clone https://github.com/aws-samples/synthetics-canary-local-debugging-sample.git
```

然後執行以下操作之一，具體取決於您的加那利群島的語言。

對於 Node.js 加那利群島

1. 輸入下列命令，移至 Node.js 初期測試來源目錄。

   ```
   cd synthetics-canary-local-debugging-sample/nodejs-canary/src
   ```

2. 輸入以下命令以安裝初期測試相依性。

   ```
   npm install
   ```

對於 Python 金絲雀

1. 通過輸入以下命令轉到 Python 初期測試源目錄。

   ```
   cd synthetics-canary-local-debugging-sample/python-canary/src
   ```

2. 輸入以下命令以安裝初期測試相依性。

```
pip3 install -r requirements.txt -t .
```

使用 Visual Studio Code IDE

Visual Studio啟動組態檔案位於.vscode/launch.json。它包含的配置，以允許模板文件由 V isual Studio 代碼被發現。它使用所需的參數定義了 Lambda 有效負載，以成功調用初期測試。以下是 Node.js 金絲雀的啟動配置：

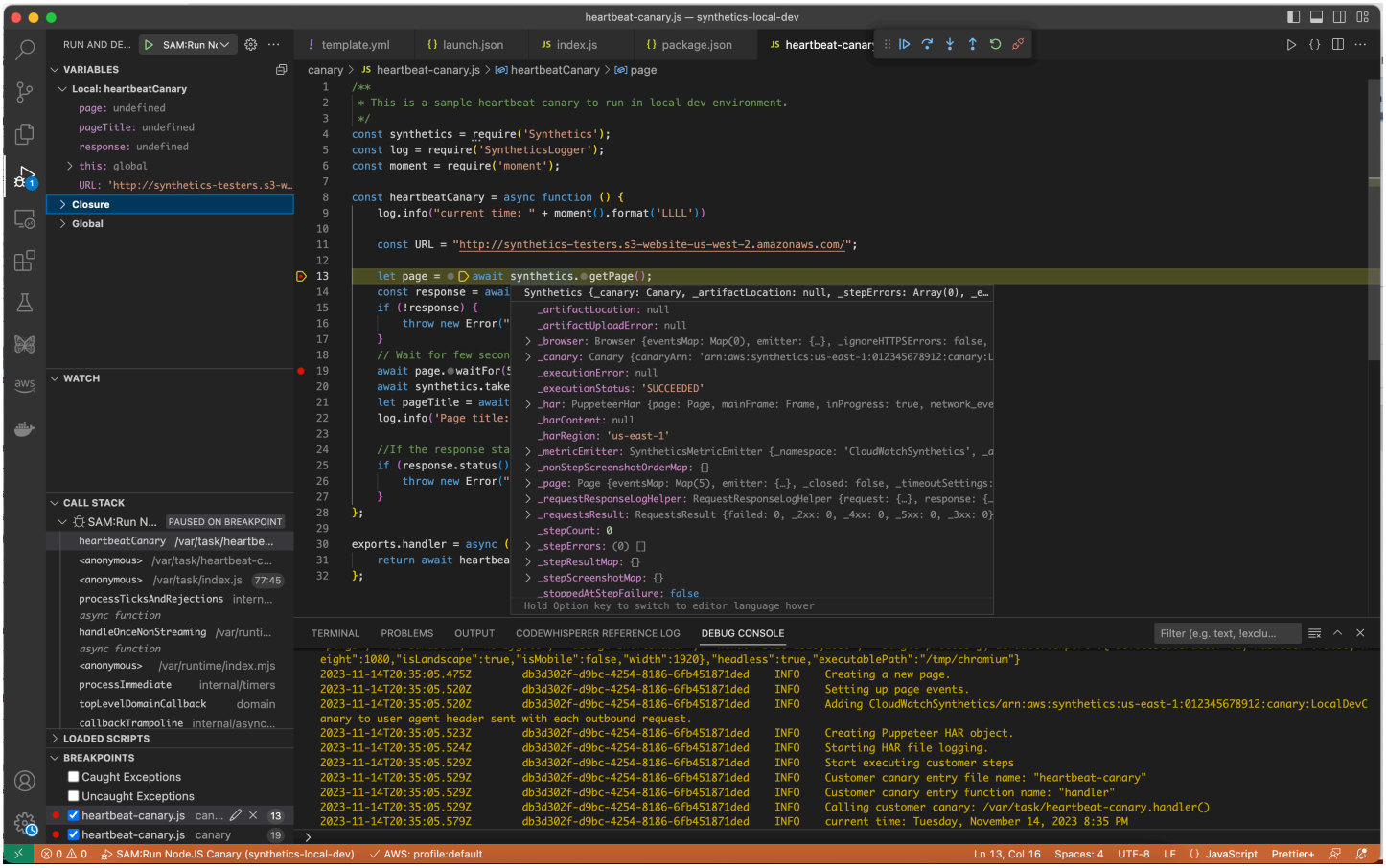
```
{
            ...
            ...
            "lambda": {
                "payload": {
                    "json": {
                        // Canary name. Provide any name you like.
                        "canaryName": "LocalSyntheticsCanary",
                        // Canary artifact location
                        "artifactS3Location": {
                            "s3Bucket": "cw-syn-results-123456789012-us-west-2",
                            "s3Key": "local-run-artifacts",
                        },
                        // Your canary handler name
                        "customerCanaryHandlerName": "heartbeat-canary.handler"
                    }
                },
                // Environment variables to pass to the canary code
                "environmentVariables": {}
            }
        }
    ]
}
```

您也可以選擇性地在承載 JSON 中提供下列欄位：

- s3EncryptionMode有效值：SSE_S3| SSE_KMS
- s3KmsKeyArn有效值：*KMS ## ARN*
- activeTracing有效值：true| false
- canaryRunId####################

若要在中偵錯初期測試Visual Studio，請在要暫停執行的初期測試程式碼中新增中斷點。要添加斷點，請選擇編輯器邊距，然後轉到編輯器中的運行和調試模式。通過單擊播放按鈕運行金絲雀。當 Canary 運行時，日誌將被跟踪在調試控制台中，為您提供有關 Canary 行為的實時見解。如果您新增中斷點，初期測試執行會在每個中斷點暫停，讓您逐步檢查程式碼並檢查變數值、執行個體方法、物件屬性和函數呼叫堆疊。

除了 Amazon S3 儲存貯體中存放的成品以及每次本機執行產生的 CloudWatch 指標外，在本機執行和偵錯 Canary 不會產生任何費用。



使用 JetBrains IDE

安裝 AWS Toolkit for JetBrains 擴充功能之後，如果您正在偵錯 Node.js 初期測試，請確定已啟用 Node.js 外掛程式和偵錯工 JavaScript 具執行。然後請遵循下列步驟。

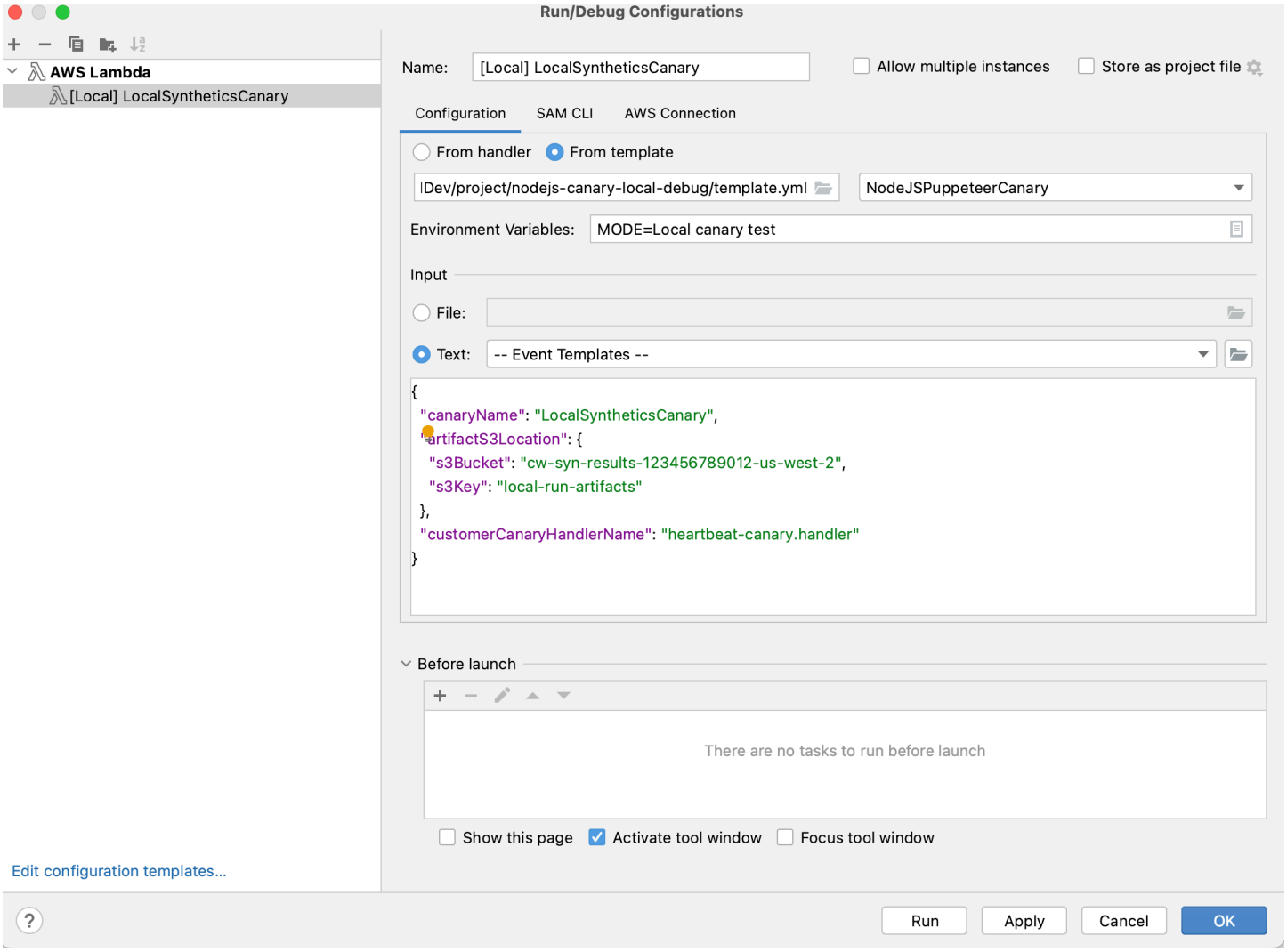
使用調試金絲雀 JetBrains IDE

1. 在的左側導覽窗格中JetBrains IDE，選擇 Lambda，然後選擇本機組態範本。
2. 輸入執行組態的名稱，例如 **`LocalSyntheticsCanary`**

3. 選擇 [從範本]，在 [範本] 欄位中選擇檔案瀏覽器，然後從專案中選取範本 .yml 檔案，無論是從 nodejs 目錄或是 python 目錄。
4. 在「輸入」區段中，輸入初期測試的承載，如以下畫面所示。

```
{
 "canaryName": "LocalSyntheticsCanary",
 "artifactS3Location": {
     "s3Bucket": "cw-syn-results-123456789012-us-west-2",
     "s3Key": "local-run-artifacts"
 },
 "customerCanaryHandlerName": "heartbeat-canary.handler"
}
```

您也可以在承載 JSON 中設定其他環境變數，如中所列使用 Visual Studio Code IDE。



使用 SAM CLI 在本機執行初期測試

使用下列其中一個程序，使用無伺服器應用程式模型 (SAM) CLI 在本機執行初期測試。請務必s3Bucket在中指定您自己的 Amazon S3 儲存貯體名稱 event.json

若要使用 SAM CLI 執行 Node.js 初始化測試

1. 輸入下列指令移至來源目錄。

```
cd synthetics-canary-local-debugging-sample/nodejs-canary
```

2. 輸入下列命令：

```
sam build
sam local invoke -e ../event.json
```

若要使用 SAM CLI 執行 Python 加那利測試

1. 輸入下列指令移至來源目錄。

```
cd synthetics-canary-local-debugging-sample/python-canary
```

2. 輸入下列命令：

```
sam build
sam local invoke -e ../event.json
```

將您的本地測試環境集成到現有的 Canary 軟件包中

您可以通過複製三個文件將本地 Canary 調試集成到現有的 Canary 包中：

- 將template.yml檔案複製到您的初期測試套件根目錄中。請務必修改指向初期CodeUri測試程式碼所在目錄的路徑。
- 如果您正在使用 Node.js 初期測試，請將cw-synthetics.js檔案複製到初期測試來源目錄。如果您正在使用 Python 初期測試，請將其複製cw-synthetics.py到您的初期測試源目錄。
- 複製啟動組態檔案。 vscode/launch.json進入軟件包根目錄。確保將其放在.vscode目錄中; 如果它不存在，請創建它。

變更 CloudWatch Synthetics 執行階段

作為調試的一部分，您可能希望嘗試使用不 CloudWatch Synthetics 運行時運行初期測試，而不是最新的運行時。若要這麼做，請從下列其中一個表格中尋找您要使用的執行階段。請務必為正確的「區域」選取執行階段。然後將該執行階段的 ARN 貼到`template.yml`檔案中的適當位置，然後執行初期測試。

Node.js 執行時間

-7.0 的 syn-nodejs-puppeteer ARN

下表列出每個可用 AWS 區域中 CloudWatch Synthetics 執行階段版本所使用`syn-nodejs-puppeteer-7.0`的 ARN。

區域	ARN
美國東部 (維吉尼亞北部)	`arn:aws:lambda:us-east-1:378653112637:layer:Synthetics:44`
美國東部 (俄亥俄)	`arn:aws:lambda:us-east-2:772927465453:layer:Synthetics:46`
美國西部 (加利佛尼亞北部)	`arn:aws:lambda:us-west-1:332033056316:layer:Synthetics:44`
美國西部 (奧勒岡)	`arn:aws:lambda:us-west-2:760325925879:layer:Synthetics:47`
非洲 (開普敦)	`arn:aws:lambda:af-south-1:461844272066:layer:Synthetics:44`
亞太區域 (香港)	`arn:aws:lambda:ap-east-1:129828061636:layer:Synthetics:45`
亞太區域 (海德拉巴)	`arn:aws:lambda:ap-south-2:280298676434:layer:Synthetics:20`
亞太區域 (雅加達)	`arn:aws:lambda:ap-southeast-3:246953257743:layer:Synthetics:26`

區域	ARN
亞太區域 (墨爾本)	arn:aws:lambda:ap-southeast-4:200724813040:layer:Synthetics:18
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:724929286329:layer:Synthetics:44
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:608016332111:layer:Synthetics:30
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:989515803484:layer:Synthetics:46
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:068035103298:layer:Synthetics:49
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:584677157514:layer:Synthetics:44
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:172291836251:layer:Synthetics:44
加拿大 (中部)	arn:aws:lambda:ca-central-1:236629016841:layer:Synthetics:44
加拿大西部 (卡加利)	arn:aws:lambda:ca-west-1:944448206667:layer:Synthetics:76
中國 (北京)	arn:aws-cn:lambda:cn-north-1:422629156088:layer:Synthetics:45
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:474974519687:layer:Synthetics:46
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:122305336817:layer:Synthetics:44
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:563204233543:layer:Synthetics:46

區域	ARN
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:565831452869:layer:Synthetics:44
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:525618516618:layer:Synthetics:45
Europe (Paris)	arn:aws:lambda:eu-west-3:469466506258:layer:Synthetics:44
歐洲 (西班牙)	arn:aws:lambda:eu-south-2:029793053121:layer:Synthetics:20
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:162938142733:layer:Synthetics:44
歐洲 (蘇黎世)	arn:aws:lambda:eu-central-2:224218992030:layer:Synthetics:19
以色列 (特拉維夫)	arn:aws:lambda:il-central-1:313249807427:layer:Synthetics:17
Middle East (Bahrain)	arn:aws:lambda:me-south-1:823195537320:layer:Synthetics:44
中東 (阿拉伯聯合大公國)	arn:aws:lambda:me-central-1:239544149032:layer:Synthetics:19
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:783765544751:layer:Synthetics:45
AWS GovCloud (美國東部)	arn:aws-us-gov:lambda:us-gov-east-1:946759330430:layer:Synthetics:41
AWS GovCloud (美國西部)	arn:aws-us-gov:lambda:us-gov-west-1:946807836238:layer:Synthetics:42

用 syn-nodejs-puppeteer於 -6.2 的 ARN

下表列出每個可用 AWS 區域中 CloudWatch Synthetics 執行階段版本所使用`syn-nodejs-puppeteer-6.2`的 ARN。

區域	ARN
美國東部 (維吉尼亞北部)	`arn:aws:lambda:us-east-1:378653112637:layer:Synthetics:41`
美國東部 (俄亥俄)	`arn:aws:lambda:us-east-2:772927465453:layer:Synthetics:43`
美國西部 (加利佛尼亞北部)	`arn:aws:lambda:us-west-1:332033056316:layer:Synthetics:41`
美國西部 (奧勒岡)	`arn:aws:lambda:us-west-2:760325925879:layer:Synthetics:44`
非洲 (開普敦)	`arn:aws:lambda:af-south-1:461844272066:layer:Synthetics:41`
亞太區域 (香港)	`arn:aws:lambda:ap-east-1:129828061636:layer:Synthetics:42`
亞太區域 (海德拉巴)	`arn:aws:lambda:ap-south-2:280298676434:layer:Synthetics:17`
亞太區域 (雅加達)	`arn:aws:lambda:ap-southeast-3:246953257743:layer:Synthetics:23`
亞太區域 (墨爾本)	`arn:aws:lambda:ap-southeast-4:200724813040:layer:Synthetics:15`
亞太區域 (孟買)	`arn:aws:lambda:ap-south-1:724929286329:layer:Synthetics:41`
亞太區域 (大阪)	`arn:aws:lambda:ap-northeast-3:608016332111:layer:Synthetics:27`

區域	ARN
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:989515803484:layer:Synthetics:42
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:068035103298:layer:Synthetics:46
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:584677157514:layer:Synthetics:41
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:172291836251:layer:Synthetics:41
加拿大 (中部)	arn:aws:lambda:ca-central-1:236629016841:layer:Synthetics:41
加拿大西部 (卡加利)	arn:aws:lambda:ca-west-1:944448206667:layer:Synthetics:73
中國 (北京)	arn:aws-cn:lambda:cn-north-1:422629156088:layer:Synthetics:42
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:474974519687:layer:Synthetics:43
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:122305336817:layer:Synthetics:41
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:563204233543:layer:Synthetics:43
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:565831452869:layer:Synthetics:41
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:525618516618:layer:Synthetics:42
Europe (Paris)	arn:aws:lambda:eu-west-3:469466506258:layer:Synthetics:41

區域	ARN
歐洲 (西班牙)	arn:aws:lambda:eu-south-2:029793053121:layer:Synthetics:17
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:162938142733:layer:Synthetics:41
歐洲 (蘇黎世)	arn:aws:lambda:eu-central-2:224218992030:layer:Synthetics:16
以色列 (特拉維夫)	arn:aws:lambda:il-central-1:313249807427:layer:Synthetics:14
Middle East (Bahrain)	arn:aws:lambda:me-south-1:823195537320:layer:Synthetics:41
中東 (阿拉伯聯合大公國)	arn:aws:lambda:me-central-1:239544149032:layer:Synthetics:16
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:783765544751:layer:Synthetics:42
AWS GovCloud (美國東部)	arn:aws-us-gov:lambda:us-gov-east-1:946759330430:layer:Synthetics:39
AWS GovCloud (美國西部)	arn:aws-us-gov:lambda:us-gov-west-1:946807836238:layer:Synthetics:39

-5.2 的 syn-nodejs-puppeteer ARN

下表列出每個可用 AWS 區域中 CloudWatch Synthetics 執行階段版本所使用syn-nodejs-puppeteer-5.2的 ARN。

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:378653112637:layer:Synthetics:42

區域	ARN
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:772927465453:layer:Synthetics:44
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:332033056316:layer:Synthetics:42
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:760325925879:layer:Synthetics:45
非洲 (開普敦)	arn:aws:lambda:af-south-1:461844272066:layer:Synthetics:42
亞太區域 (香港)	arn:aws:lambda:ap-east-1:129828061636:layer:Synthetics:43
亞太區域 (海德拉巴)	arn:aws:lambda:ap-south-2:280298676434:layer:Synthetics:18
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:246953257743:layer:Synthetics:24
亞太區域 (墨爾本)	arn:aws:lambda:ap-southeast-4:200724813040:layer:Synthetics:16
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:724929286329:layer:Synthetics:42
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:608016332111:layer:Synthetics:28
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:989515803484:layer:Synthetics:44
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:068035103298:layer:Synthetics:47
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:584677157514:layer:Synthetics:42

區域	ARN
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:172291836251:layer:Synthetics:42
加拿大 (中部)	arn:aws:lambda:ca-central-1:236629016841:layer:Synthetics:42
加拿大西部 (卡加利)	arn:aws:lambda:ca-west-1:944448206667:layer:Synthetics:74
中國 (北京)	arn:aws-cn:lambda:cn-north-1:422629156088:layer:Synthetics:43
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:474974519687:layer:Synthetics:44
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:122305336817:layer:Synthetics:42
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:563204233543:layer:Synthetics:44
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:565831452869:layer:Synthetics:42
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:525618516618:layer:Synthetics:43
Europe (Paris)	arn:aws:lambda:eu-west-3:469466506258:layer:Synthetics:42
歐洲 (西班牙)	arn:aws:lambda:eu-south-2:029793053121:layer:Synthetics:18
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:162938142733:layer:Synthetics:42
歐洲 (蘇黎世)	arn:aws:lambda:eu-central-2:224218992030:layer:Synthetics:17

區域	ARN
以色列 (特拉維夫)	arn:aws:lambda:il-central-1:313249807427:layer:Synthetics:15
Middle East (Bahrain)	arn:aws:lambda:me-south-1:823195537320:layer:Synthetics:42
中東 (阿拉伯聯合大公國)	arn:aws:lambda:me-central-1:239544149032:layer:Synthetics:17
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:783765544751:layer:Synthetics:43
AWS GovCloud (美國東部)	arn:aws-us-gov:lambda:us-gov-east-1:946759330430:layer:Synthetics:40
AWS GovCloud (美國西部)	arn:aws-us-gov:lambda:us-gov-west-1:946807836238:layer:Synthetics:40

Python 執行時間

用 syn-python-selenium於 -3.0 的 ARN

下表列出每個可用 AWS 區域中 CloudWatch Synthetics 執行階段版本所使用`syn-python-selenium-3.0`的 ARN。

區域	ARN
美國東部 (維吉尼亞北部)	aarn:aws:lambda:us-east-1:378653112637:layer:Synthetics_Selenium:32
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:772927465453:layer:Synthetics_Selenium:34
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:332033056316:layer:Synthetics_Selenium:32

區域	ARN
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:760325925879:layer:Synthetics_Selenium:34
非洲 (開普敦)	arn:aws:lambda:af-south-1:461844272066:layer:Synthetics_Selenium:32
亞太區域 (香港)	arn:aws:lambda:ap-east-1:129828061636:layer:Synthetics_Selenium:32
亞太區域 (海德拉巴)	arn:aws:lambda:ap-south-2:280298676434:layer:Synthetics_Selenium:20
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:246953257743:layer:Synthetics_Selenium:26
亞太區域 (墨爾本)	arn:aws:lambda:ap-southeast-4:200724813040:layer:Synthetics_Selenium:18
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:724929286329:layer:Synthetics_Selenium:32
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:608016332111:layer:Synthetics_Selenium:30
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:989515803484:layer:Synthetics_Selenium:34
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:068035103298:layer:Synthetics_Selenium:37
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:584677157514:layer:Synthetics_Selenium:32
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:172291836251:layer:Synthetics_Selenium:32
加拿大 (中部)	arn:aws:lambda:ca-central-1:236629016841:layer:Synthetics_Selenium:32

區域	ARN
加拿大西部 (卡加利)	arn:aws:lambda:ca-west-1:944448206667:layer:Synthetics_Selenium:76
中國 (北京)	arn:aws-cn:lambda:cn-north-1:422629156088:layer:Synthetics_Selenium:32
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:474974519687:layer:Synthetics_Selenium:32
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:122305336817:layer:Synthetics_Selenium:32
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:563204233543:layer:Synthetics_Selenium:34
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:565831452869:layer:Synthetics_Selenium:32
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:525618516618:layer:Synthetics_Selenium:33
Europe (Paris)	arn:aws:lambda:eu-west-3:469466506258:layer:Synthetics_Selenium:32
歐洲 (西班牙)	arn:aws:lambda:eu-south-2:029793053121:layer:Synthetics_Selenium:20
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:162938142733:layer:Synthetics_Selenium:32
歐洲 (蘇黎世)	arn:aws:lambda:eu-central-2:224218992030:layer:Synthetics_Selenium:19
以色列 (特拉維夫)	arn:aws:lambda:il-central-1:313249807427:layer:Synthetics_Selenium:17
Middle East (Bahrain)	arn:aws:lambda:me-south-1:823195537320:layer:Synthetics_Selenium:32

區域	ARN
中東 (阿拉伯聯合大公國)	arn:aws:lambda:me-central-1:239544149032:layer:Synthetics_Selenium:19
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:783765544751:layer:Synthetics_Selenium:33
AWS GovCloud (美國東部)	arn:aws-us-gov:lambda:us-gov-east-1:946759330430:layer:Synthetics_Selenium:30
AWS GovCloud (美國西部)	arn:aws-us-gov:lambda:us-gov-west-1:946807836238:layer:Synthetics_Selenium:31

用 syn-python-selenium於 -2.1 的 ARN

下表列出每個可用 AWS 區域中 CloudWatch Synthetics 執行階段版本所使用syn-python-selenium-2.1的 ARN。

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:378653112637:layer:Synthetics:29
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:772927465453:layer:Synthetics:31
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:332033056316:layer:Synthetics:29
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:760325925879:layer:Synthetics:31
非洲 (開普敦)	arn:aws:lambda:af-south-1:461844272066:layer:Synthetics:29
亞太區域 (香港)	arn:aws:lambda:ap-east-1:129828061636:layer:Synthetics:29

區域	ARN
亞太區域 (海德拉巴)	arn:aws:lambda:ap-south-2:280298676434:layer:Synthetics:17
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:246953257743:layer:Synthetics:23
亞太區域 (墨爾本)	arn:aws:lambda:ap-southeast-4:200724813040:layer:Synthetics:15
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:724929286329:layer:Synthetics:29
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:608016332111:layer:Synthetics:27
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:989515803484:layer:Synthetics:30
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:068035103298:layer:Synthetics:34
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:584677157514:layer:Synthetics:29
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:172291836251:layer:Synthetics:29
加拿大 (中部)	arn:aws:lambda:ca-central-1:236629016841:layer:Synthetics:29
加拿大西部 (卡加利)	arn:aws:lambda:ca-west-1:944448206667:layer:Synthetics:73
中國 (北京)	arn:aws-cn:lambda:cn-north-1:422629156088:layer:Synthetics:29
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:474974519687:layer:Synthetics:29

區域	ARN
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:122305336817:layer:Synthetics:29
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:563204233543:layer:Synthetics:31
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:565831452869:layer:Synthetics:29
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:525618516618:layer:Synthetics:30
Europe (Paris)	arn:aws:lambda:eu-west-3:469466506258:layer:Synthetics:29
歐洲 (西班牙)	arn:aws:lambda:eu-south-2:029793053121:layer:Synthetics:17
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:162938142733:layer:Synthetics:29
歐洲 (蘇黎世)	arn:aws:lambda:eu-central-2:224218992030:layer:Synthetics:16
以色列 (特拉維夫)	arn:aws:lambda:il-central-1:313249807427:layer:Synthetics:14
Middle East (Bahrain)	arn:aws:lambda:me-south-1:823195537320:layer:Synthetics:29
中東 (阿拉伯聯合大公國)	arn:aws:lambda:me-central-1:239544149032:layer:Synthetics:16
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:783765544751:layer:Synthetics:30
AWS GovCloud (美國東部)	arn:aws-us-gov:lambda:us-gov-east-1:946759330430:layer:Synthetics:29

區域	ARN
AWS GovCloud (美國西部)	`arn:aws-us-gov:lambda:us-gov-west-1:946807836238:layer:Synthetics:29`

常見錯誤

錯誤：在本地運行 AWS SAM 項目需要 Docker。你有它安裝並運行嗎？

確保在計算機Docker上啟動。

SAM 本地調用失敗：調用 GetLayerVersion 操作時發生錯誤（ExpiredTokenException）：請求中包含的安全令牌已過期

請確定已設定 AWS 預設設定檔。

更多常見錯誤

如需有關 SAM 常見錯誤的詳細資訊，請參閱 AWS SAM CLI 疑難排解。

對失敗的 Canary 進行故障診斷

如果 Canary 失敗，請檢查以下內容進行故障診斷。

一般性問題的故障診斷

- 使用 Canary 詳細資訊頁面來尋找更多資訊。在 CloudWatch 控制台中，在導航窗格中選擇 Canary，然後選擇初期測試的名稱以打開初期測試詳細信息頁面。在「使用狀態」索引標籤中，檢查SuccessPercent測量結果，以查看問題是常數還是間歇性問題。

  依然在 Availability (可用性) 標籤上，選擇失敗的資料點，以查看該失敗執行的螢幕擷取畫面、日誌和步驟報告 (如可用)。

  如果因為步驟屬於指令碼的一部分而提供步驟報告，請檢查哪個步驟失敗，並查看相關的螢幕擷取畫面，以查看客戶看到的問題。

  您也可以檢查 HAR 檔案，以查看是否有一或多個請求失敗。您可以使用日誌來深入探索失敗的請求和錯誤。最後，您可將這些成品與成功的金絲雀部署執行的成品進行比較，以確定問題。

  默認情況下， CloudWatch Synthetics 捕獲 UI 初期測試中每個步驟的屏幕截圖。不過，您的指令碼可能會設定為停用螢幕擷取畫面。在偵錯期間，您可能想要再次啟用螢幕擷取畫面。同樣，對於 API

Canary，您可能希望在偵錯過程中查看 HTTP 請求和回應標頭與內文。如需如何在報告內包含此資料的相關資訊，請參閱 executeHttpStep（stepName，請求選項，[回調]，[步驟配置]）。

- 如果您最近有部署到您的應用程式，請將其轉返，然後稍後再進行偵錯。
- 手動連接到您的端點，看看是否可以重現相同的問題。

主題

- Lambda 環境更新之後，金絲雀
- 我的金絲雀被阻止了 AWS WAF
- 等待元素出現
- 節點不可見或不是 page.click() 的 HTMLElement
- 無法上傳成品到 S3，例外狀況：無法擷取 S3 儲存貯體位置：拒絕存取
- 錯誤：通訊協定錯誤 (執行階段。 callFunctionOn ）：目標已關閉。
- Canary 失敗。錯誤：沒有資料點 - Canary 顯示逾時錯誤
- 嘗試存取內部端點
- Canary 執行時間版本升級和降級問題
- 跨來源請求共享 (CORS) 問題
- 金絲雀競爭狀況問題
- 對 VPC 上的 Canary 進行故障診斷

Lambda 環境更新之後，金絲雀

CloudWatch Synthetics 金絲雀在您的帳戶中作為 Lambda 函數實現。這些 Lambda 函數會定期進行 Lambda 執行階段更新，其中包含安全性更新、錯誤修正和其他改進。Lambda 致力於提供與現有函數向後相容的執行階段更新。但是與軟體修補一樣，在極少數情況下，執行階段更新可能會對現有函數產生負面影響。如果您認為您的初期測試受到 Lambda 執行階段更新的影響，則可以使用 Lambda 執行階段管理手動模式 (在支援的區域中) 暫時復原 Lambda 執行階段版本。這可以使您的 Canary 功能正常工作並最大限度地減少中斷，從而在返回到最新的運行時版本之前提供補救不兼容的時間

如果您的初期測試在 Lambda 執行階段更新後失敗，最好的解決方案是升級至最新的 Synthetics 執行階段之一。如需最新執行階段的詳細資訊，請參閱Synthetics 執行時間版本。

作為替代解決方案，在提供 Lambda 執行階段管理控制項的區域中，您可以使用手動模式進行執行階段管理控制，將初期測試還原回較舊的 Lambda 受管執行階段。您可以使用 AWS CLI 或使用 Lambda 主控台，使用以下各節中的步驟來設定手動模式。

> **Warning**
>
> 當您將執行階段設定變更為手動模式時，Lambda 函數在還原為自動模式之前，不會收到自動安全性更新。在此期間，您的 Lambda 函數可能容易受到安全性弱點影響。

先決條件

- 安裝 JQ
- 安裝最新版本的 AWS CLI. 如需詳細資訊，請參閱AWS CLI 安裝與更新指示。

步驟 1：取得 Lambda 函數 ARN

執行下列命令以從回應中擷取EngineArn欄位。這EngineArn是與金絲雀相關聯的 Lambda 函數的 ARN。您將在以下步驟中使用此 ARN。

```
aws synthetics get-canary --name my-canary | jq '.Canary.EngineArn'
```

輸出示例EngingArn：

```
"arn:aws:lambda:us-west-2:123456789012:function:cwsyn-my-canary-dc5015c2-db17-4cb5-
afb1-EXAMPLE991:8"
```

步驟 2：取得最後一個良好的 Lambda 執行階段版本 ARN

為了協助瞭解您的初期測試是否受到 Lambda 執行階段更新的影響，請檢查日誌中 Lambda 執行階段版本 ARN 變更的日期和時間是否顯示在您看到初期測試影響的日期和時間。如果它們不相符，則可能不是導致您問題的 Lambda 執行階段更新。

如果您的初期測試受到 Lambda 執行階段更新的影響，您必須識別先前使用的工作 Lambda 執行階段版本的 ARN。依照識別執行階段版本變更中的指示，尋找上一個執行階段的 ARN。記錄執行階段版本 ARN，並繼續執行步驟 3 以設定執行階段管理組態。

如果您的初期測試尚未受到 Lambda 環境更新的影響，您可以找到目前使用的 Lambda 執行階段版本的 ARN。執行下列命令，從回應中擷取 Lambda 函數。RuntimeVersionArn

```
aws lambda get-function-configuration \
--function-name "arn:aws:lambda:us-west-2:123456789012:function:cwsyn-my-canary-
dc5015c2-db17-4cb5-afb1-EXAMPLE991:8" | jq '.RuntimeVersionConfig.RuntimeVersionArn'
```

輸出示例RuntimeVersionArn：

```
"arn:aws:lambda:us-
west-2::runtime:EXAMPLE647b82f490a45d7ddd96b557b916a30128d9dcab5f4972911ec0f"
```

步驟 3：更新 Lambda 執行階段管理組態

您可以使用 AWS CLI 或 Lambda 主控台來更新執行階段管理組態。

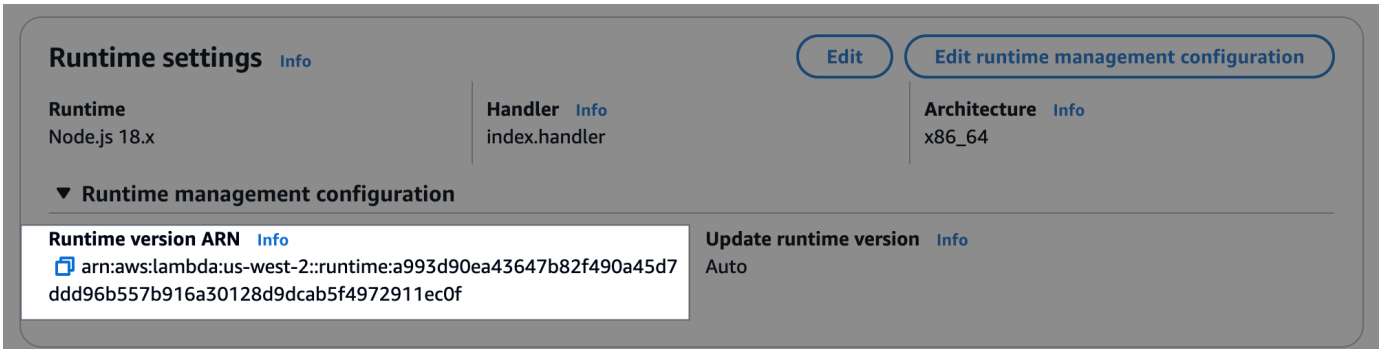
若要設定 Lambda 執行階段管理組態手動模式，請 AWS CLI

輸入下列命令，將 Lambda 函數的執行階段管理變更為手動模式。請務必使用您在步驟 1 中找到的值，將函數##和#定詞分別取代為 Lambda 函數 ARN 和 Lambda 函數版本號碼。同時以您在步驟 2 中找到的 ARN 版本取代。*runtime-version-arn*

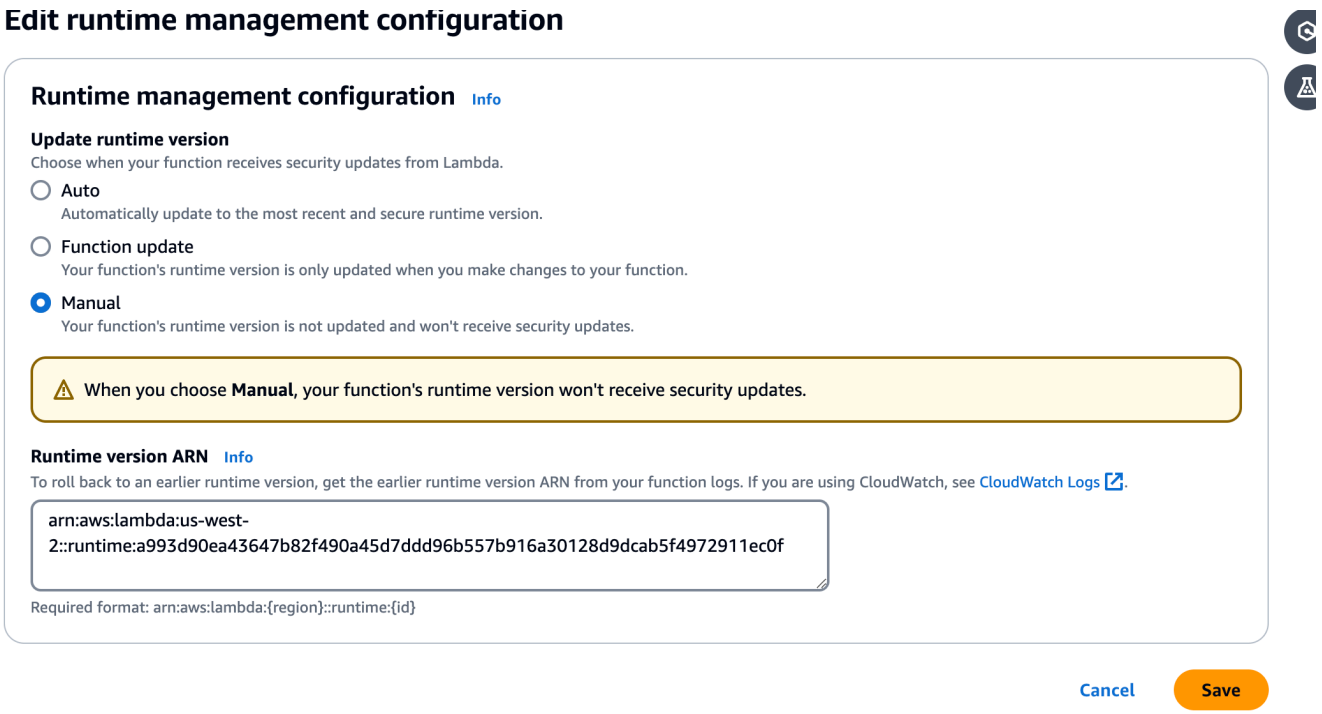
```
aws lambda put-runtime-management-config \
    --function-name "arn:aws:lambda:us-west-2:123456789012:function:cwsyn-my-canary-
dc5015c2-db17-4cb5-afb1-EXAMPLE991" \
    --qualifier 8 \
    --update-runtime-on "Manual" \
    --runtime-version-arn "arn:aws:lambda:us-
west-2::runtime:a993d90ea43647b82f490a45d7ddd96b557b916a30128d9dcab5f4972911ec0f"
```

使用 Lambda 主控台將初期測試變更為手動模式

1. 請在以下位置開啟 AWS Lambda 主控台。 https://console.aws.amazon.com/lambda/
2. 選擇 [版本] 索引標籤，選擇與 ARN 對應的版本編號連結，然後選擇 [程式碼] 索引標籤。
3. 向下捲動至 [執行階段設定]，展開 [執行階段管理組態]，然後複製 [執行階段] 版本 ARN



4. 選擇編輯程式實際執行管理組態，選擇手動，將您之前複製的程式實際執行版本 ARN 貼到「程式實際執行版本 ARN」欄位中。然後選擇 Save (儲存)。



我的金絲雀被阻止了 AWS WAF

若要防 AWS WAF 止封鎖初期測試，請設定允許 AWS WAF 字串的字串比對條件`CloudWatchSynthetics`。如需詳細資訊，請參閱文件中的使用字串比對條 AWS WAF 件。

等待元素出現

分析日誌和螢幕擷取畫面後，如果您看到指令碼正在等待元素出現在螢幕上並逾時，請檢查相關的螢幕擷取畫面，看看元素是否出現在頁面上。確認您的 `xpath`，以確保它正確無誤。

有關木偶工具的相關問題，請查看木偶師的頁面或互聯網論壇。 GitHub

節點不可見或不是 page.click() 的 HTMLElement

如果節點不可見或不是 `page.click()` 的 `HTMLElement`，首先驗證您正在使用的 `xpath`，以按一下元素。另外，如果您的元素位於屏幕底部，請調整視口。 CloudWatch 默認情況下，Synthetics 使用 1920* 1080 的視口。您可以在啟動瀏覽器或使用 Puppeteer 函數 `page.setViewport` 時，設定不同的檢視區。

無法上傳成品到 S3，例外狀況：無法擷取 S3 儲存貯體位置：拒絕存取

如果您的初期測試因為 Amazon S3 錯誤而失敗，則由於權限問題， CloudWatch Synthetics 無法上傳針對初期測試建立的螢幕擷取畫面、日誌或報告。請檢查以下內容：

- 檢查 Canary 的 IAM 角色是否具有 `s3:ListAllMyBuckets` 許可、正確 Simple Storage Service (Amazon S3) 儲存貯體的 `s3:GetBucketLocation` 許可，以及 Canary 存放其成品之儲存貯體的 `s3:PutObject` 許可。如果 Canary 執行視覺監控，則角色也需要儲存貯體的 `s3:GetObject` 許可。如果 Canary 部署在具有 VPC 端點的 VPC 中，則 Amazon VPC S3 閘道端點政策也需要這些相同的許可。
- 如果初期測試使用 AWS KMS 客戶受管金鑰進行加密，而不是標準 AWS 受管金鑰 (預設)，則 Canary 的 IAM 角色可能沒有使用該金鑰加密或解密的權限。如需詳細資訊，請參閱 加密 Canary 成品。
- 您的儲存貯體政策可能不允許 Canary 使用的加密機制。例如，如果您的儲存貯體政策要求使用特定的加密機制或 KMS 金鑰，則您必須為 Canary 選取相同的加密模式。

如果 Canary 執行視覺監控，請參閱 使用視覺監控時更新成品位置和加密 以了解詳細資訊。

錯誤：通訊協定錯誤 (執行階段。 callFunctionOn ）：目標已關閉。

如果頁面或瀏覽器關閉後存在某些網路請求，就會出現此錯誤。您可能忘記等待非同步操作。執行腳本後， CloudWatch Synthetics 關閉瀏覽器。瀏覽器關閉後執行任何非同步操作可能會導致 `target closed error`。

Canary 失敗。錯誤：沒有資料點 - Canary 顯示逾時錯誤

這意味著您的 Canary 執行超過逾時時間。在 CloudWatch Synthetics 可以發布成功百分比 CloudWatch 指標或更新成品（例如 HAR 文件，日誌和屏幕截圖）之前停止初期測試執行。如果您的逾時太低，您可以增加它。

預設情況下，Canary 逾時值等於其頻率。您可以手動將逾時值調整為小於或等於 Canary 頻率。如果您的 Canary 頻率很低，則必須增加頻率以增加逾時。您可以使用 CloudWatch Synthetics 主控台建立或更新初期測試時，在「排程」下調整頻率和逾時值。

請確定您的逾時值不短於 15 秒，以允許 Lambda 冷啟動和啟動 canary 儀器所需的時間。

發生此錯誤時，Canary 工件無 CloudWatch 法在合成材料控制台中查看。您可以使用 CloudWatch 日誌來查看初期測試的日誌。

若要使用 CloudWatch 記錄查看初期測試的記錄

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在左側導覽窗格中，選擇 Log groups (日誌群組)。

3. 在篩選條件方塊中輸入 Canary 名稱，以尋找日誌群組。Canary 的日誌組的名稱為 /aws/lambda/cwsyn-*canaryName*-randomId。

嘗試存取內部端點

如果您希望 Canary 存取內部網路上的端點，建議您將 CloudWatch Synthetics 設定為使用 VPC。如需詳細資訊，請參閱 在 VPC 上執行 Canary。

Canary 執行時間版本升級和降級問題

如果您最近將 Canary 從執行時間版本 `syn-1.0` 升級至最新版本，它可能是跨來源請求共享 (CORS) 問題。如需詳細資訊，請參閱 跨來源請求共享 (CORS) 問題。

如果您最近將初期測試降級為較舊的執行階段版本，請檢查以確定您正在使用的 CloudWatch Synthetics 函數可在您降級的較舊執行階段版本中使用。例如，`executeHttpStep` 函數可用於執行時間版本 `syn-nodejs-2.2` 和更新版本。若要檢查函數的可用性，請參閱 撰寫 Canary 指令碼。

> **Note**
>
> 當您計劃升級或降級初期測試的執行階段版本時，建議您先複製初期測試，並在複製的初期測試中更新執行階段版本。一旦您驗證了含新版執行時間的複製 Canary 可運作，您可以更新原始 Canary 的執行時間版本並刪除複製的 Canary。

跨來源請求共享 (CORS) 問題

在 UI Canary 中，如果某些網絡請求失敗並顯示 `403` 或 `net::ERR_FAILED`，請檢查 Canary 是否已啟用作用中追蹤，並使用 Puppeteer 函數 `page.setExtraHTTPHeaders` 以新增標頭。如果是這樣，失敗的網路請求可能是跨來源請求共享 (CORS) 限制所造成。您可以藉由停用作用中追蹤或移除額外的 HTTP 標頭來確認是否會發生這種情況。

為什麼會出現這種情況？

使用作用中追蹤時，會將額外的標頭新增至所有傳出要求，以追蹤呼叫。透過新增追蹤標頭或使用木偶新增額外標頭來修改要求標頭，`page.setExtraHTTPHeaders`會導致 CORS 檢查 XML HttpRequest (XHR) 要求。

如果您不想停用作用中追蹤或移除額外的標題，則可以更新 Web 應用程式以允許跨來源存取，或者您可以在您的指令碼中啟動 Chrome 瀏覽器時使用 `disable-web-security` 旗標。

您可以覆蓋 CloudWatch Synthetics 使用的啟動參數，並通過使用 S CloudWatch ynthetics 啟動功能傳遞其他`disable-web-security`標誌參數。如需詳細資訊，請參閱 適用於 Node.js Canary 指令碼的程式庫函數。

> **Note**
>
> 當您使用執行階段版本`syn-nodejs-2.1`或更新版本時，您可以覆寫 CloudWatch Synthetics 使用的啟動參數。

金絲雀競爭狀況問題

為了在使用 CloudWatch Synthetics 時獲得最佳體驗，請確保為 Canary 編寫的代碼是冪等的。否則，在極少數情況下，當金絲雀在不同的運行中與相同的資源進行交互時，Canary 運行可能會遇到競爭條件。

對 VPC 上的 Canary 進行故障診斷

如果您在建立或更新 VPC 上的 Canary 後發生問題，下列其中一個章節可能可以協助您對問題進行故障診斷。

處於錯誤狀態的新 Canary 或 Canary 無法更新

如果您建立要在 VPC 上執行的 Canary，而且它立即進入錯誤狀態，或您無法更新要在 VPC 上執行的 Canary，Canary 的角色可能不具備適當的許可。若要在 VPC 上執行，Canary 必須具有 `ec2:CreateNetworkInterface`、`ec2:DescribeNetworkInterfaces` 和 `ec2:DeleteNetworkInterface` 許可。這些許可都包含在 `AWSLambdaVPCAccessExecutionRole` 受管政策中。如需詳細資訊，請參閱執行角色和使用者許可。

如果此問題是在您建立 Canary 時發生，您必須刪除 Canary，然後建立新的 Canary。如果您使用 CloudWatch 主控台建立新的初期測試，請在 [存取權限] 下選取 [建立新角色]。系統會建立包含執行 Canary 所需之所有許可的新角色。

如果此問題是在您更新 Canary 時發生，您可以再次更新 Canary，並提供具備必要許可的新角色。

「No test result returned」(未傳回測試結果) 錯誤

如果 Canary 顯示「no test result returned」(未傳回測試結果) 錯誤，則原因可能是下列其中一個問題：

- 如果您的 VPC 無法存取網際網路，您必須使用 VPC 端點授予 CloudWatch 和 Amazon S3 的初期測試存取權。您必須在 VPC 中啟用 DNS resolution (DNS 解析) 和 DNS hostname (DNS 主機名稱) 選項，才能正確解析這些端點位址。如需詳細資訊，請參閱將 DNS 與 VPC 搭配使用 CloudWatch 和 CloudWatch Synthetics 與介面 V PC 端點。
- Canary 必須在 VPC 內的私有子網路中執行。若要檢查，請在 VPC 主控台中開啟 Subnets (子網路) 頁面。檢查您在設定 Canary 時所選取的子網路。如果子網路具有網際網路閘道 (igw-) 的路徑，則不是私有子網路。

為協助您對這些問題進行故障診斷，請參閱 Canary 的日誌。

如何查看來自 Canary 的日誌事件

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Log groups (日誌群組)。
3. 選擇 Canary 日誌群組的名稱。日誌群組名稱開頭為 `/aws/lambda/cwsyn-`*`canary-name`*。

Canary 指令碼的範本程式碼

本節包含的程式碼範例說明了 CloudWatch Synthetics 初期測試指令碼的一些可能函式。

Node.js 與 Puppeteer 的範例

設定 Cookie

網站依賴 Cookie 來提供自訂功能或追蹤使用者。通過在 CloudWatch Synthetics 腳本中設置 cookie，您可以模仿此自定義行為並對其進行驗證。

例如，網站可能會顯示 Login (登入) 連結，以重新造訪使用者而不是 Register (登錄) 連結。

```
var synthetics = require('Synthetics');
const log = require('SyntheticsLogger');

const pageLoadBlueprint = async function () {

    let url = "http://smile.amazon.com/";

    let page = await synthetics.getPage();

    // Set cookies.  I found that name, value, and either url or domain are required
 fields.
```

```
    const cookies = [{
      'name': 'cookie1',
      'value': 'val1',
      'url': url
    },{
      'name': 'cookie2',
      'value': 'val2',
      'url': url
    },{
      'name': 'cookie3',
      'value': 'val3',
      'url': url
    }];

    await page.setCookie(...cookies);

    // Navigate to the url
    await synthetics.executeStep('pageLoaded_home', async function (timeoutInMillis =
 30000) {

        var response = await page.goto(url, {waitUntil: ['load', 'networkidle0'],
 timeout: timeoutInMillis});

        // Log cookies for this page and this url
        const cookiesSet = await page.cookies(url);
        log.info("Cookies for url: " + url + " are set to: " +
 JSON.stringify(cookiesSet));
    });

};

exports.handler = async () => {
    return await pageLoadBlueprint();
};
```

裝置模擬

您可以編寫模擬各種裝置的指令碼，以便您可以大致了解頁面在這些裝置上的外觀和行為。

下列範例會模擬 iPhone 6 裝置。如需模擬的詳細資訊，請參閱 Puppeteer 文件中的 page.emulate(options)。

```
var synthetics = require('Synthetics');
const log = require('SyntheticsLogger');
```

```
const puppeteer = require('puppeteer-core');

const pageLoadBlueprint = async function () {

    const iPhone = puppeteer.devices['iPhone 6'];

    // INSERT URL here
    const URL = "https://amazon.com";

    let page = await synthetics.getPage();
    await page.emulate(iPhone);

    //You can customize the wait condition here. For instance,
    //using 'networkidle2' may be less restrictive.
    const response = await page.goto(URL, {waitUntil: 'domcontentloaded', timeout:
 30000});
    if (!response) {
        throw "Failed to load page!";
    }

    await page.waitFor(15000);

    await synthetics.takeScreenshot('loaded', 'loaded');

    //If the response status code is not a 2xx success code
    if (response.status() < 200 || response.status() > 299) {
        throw "Failed to load page!";
    }
};

exports.handler = async () => {
    return await pageLoadBlueprint();
};
```

多步驟 API Canary

此範本程式碼示範了具有兩個 HTTP 步驟的 API Canary：為正面和負面測試案例測試相同的 API。傳遞步驟組態，以啟用請求/回應標頭的報告。此外，它會隱藏授權標頭和 X-Amz-Security-Token，因為它們包含使用者憑證。

當此指令碼用作 Canary 時，您可以檢視每個步驟和相關 HTTP 請求的詳細資訊，例如步驟通過/失敗、持續時間和效能度量，例如 DNS 查詢時間和第一個位元組時間。您可以檢視 Canary 執行的 2xx、4xx 和 5xx 數目。

```
var synthetics = require('Synthetics');
const log = require('SyntheticsLogger');

const apiCanaryBlueprint = async function () {

    // Handle validation for positive scenario
    const validatePositiveCase = async function(res) {
        return new Promise((resolve, reject) => {
            if (res.statusCode < 200 || res.statusCode > 299) {
                throw res.statusCode + ' ' + res.statusMessage;
            }

            let responseBody = '';
            res.on('data', (d) => {
                responseBody += d;
            });

            res.on('end', () => {
                // Add validation on 'responseBody' here if required. For ex, your
 status code is 200 but data might be empty
                resolve();
            });
        });
    };

    // Handle validation for negative scenario
    const validateNegativeCase = async function(res) {
        return new Promise((resolve, reject) => {
            if (res.statusCode < 400) {
                throw res.statusCode + ' ' + res.statusMessage;
            }

            resolve();
        });
    };

    let requestOptionsStep1 = {
        'hostname': 'myproductsEndpoint.com',
        'method': 'GET',
        'path': '/test/product/validProductName',
        'port': 443,
        'protocol': 'https:'
```

```
    };

    let headers = {};
    headers['User-Agent'] = [synthetics.getCanaryUserAgentString(), headers['User-
Agent']].join(' ');

    requestOptionsStep1['headers'] = headers;

    // By default headers, post data and response body are not included in the report
 for security reasons.
    // Change the configuration at global level or add as step configuration for
 individual steps
    let stepConfig = {
        includeRequestHeaders: true,
        includeResponseHeaders: true,
        restrictedHeaders: ['X-Amz-Security-Token', 'Authorization'], // Restricted
 header values do not appear in report generated.
        includeRequestBody: true,
        includeResponseBody: true
    };


    await synthetics.executeHttpStep('Verify GET products API with valid name',
 requestOptionsStep1, validatePositiveCase, stepConfig);

    let requestOptionsStep2 = {
        'hostname': 'myproductsEndpoint.com',
        'method': 'GET',
        'path': '/test/canary/InvalidName(',
        'port': 443,
        'protocol': 'https:'
    };

    headers = {};
    headers['User-Agent'] = [synthetics.getCanaryUserAgentString(), headers['User-
Agent']].join(' ');

    requestOptionsStep2['headers'] = headers;

    // By default headers, post data and response body are not included in the report
 for security reasons.
    // Change the configuration at global level or add as step configuration for
 individual steps
    stepConfig = {
```

```
        includeRequestHeaders: true,
        includeResponseHeaders: true,
        restrictedHeaders: ['X-Amz-Security-Token', 'Authorization'], // Restricted
 header values do not appear in report generated.
        includeRequestBody: true,
        includeResponseBody: true
    };

    await synthetics.executeHttpStep('Verify GET products API with invalid name',
 requestOptionsStep2, validateNegativeCase, stepConfig);

};

exports.handler = async () => {
    return await apiCanaryBlueprint();
};
```

Python 和 Selenium 範例

以下範例 Selenium 程式碼是一個在未載入目標元素時失敗並顯示自訂錯誤訊息的 Canary。

```
from aws_synthetics.selenium import synthetics_webdriver as webdriver
from aws_synthetics.common import synthetics_logger as logger
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By

def custom_selenium_script():
    # create a browser instance
    browser = webdriver.Chrome()
    browser.get('https://www.example.com/')
    logger.info('navigated to home page')
    # set cookie
    browser.add_cookie({'name': 'foo', 'value': 'bar'})
    browser.get('https://www.example.com/')
    # save screenshot
    browser.save_screenshot('signed.png')
    # expected status of an element
    button_condition = EC.element_to_be_clickable((By.CSS_SELECTOR, '.submit-button'))
    # add custom error message on failure
    WebDriverWait(browser, 5).until(button_condition, message='Submit button failed to
 load').click()
    logger.info('Submit button loaded successfully')
```

```
    # browser will be quit automatically at the end of canary run,
    # quit action is not necessary in the canary script
    browser.quit()

# entry point for the canary
def handler(event, context):
    return custom_selenium_script()
```

Canary 和 X-Ray 追蹤

您可以選擇在使用syn-nodejs-2.0或更新執行階段的金絲雀上啟用作用中 AWS X-Ray 追蹤。啟用追蹤後，系統會針對使用瀏覽器、 AWS SDK 或 HTTP 或 HTTPS 模組的初期測試所發出的所有呼叫傳送追蹤。已啟用追蹤功能的 Canary 會在您為應用程式啟用它之後顯示在 X-Ray 追蹤地圖和 Application Signals 中。

> **Note**
>
> 在亞太地區（雅加達），啟用 canary 上的 X-Ray 追蹤尚未獲得支援。

當 Canary 出現在 X-Ray 追蹤地圖上時，它會顯示為新的用戶端節點類型。您可以將滑鼠暫留在 Canary 節點上，以查看有關延遲、請求和錯誤的資料。您也可以選擇 Canary 節點以在頁面底部查看更多資料。在頁面的這個區域中，您可以選擇「在 Synthetics 中檢視」，跳至「S CloudWatch ynthetics」主控台以取得有關初期測試的詳細資訊，或選擇「檢視追蹤」以查看此初期測試執行中追蹤的更多詳細資訊。

啟用追蹤的 Canary 也有一個 Tracing (追蹤) 標籤，其中包含有關 Canary 執行中的追蹤和區段的詳細資訊。

啟用追蹤會將 Canary 執行時間增加 2.5% 至 7%。

啟用追蹤的 Canary 必須使用具有下列許可的角色。如果您在建立 Canary 時使用主控台建立角色，則會授予這些許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Sid230934",
            "Effect": "Allow",
```

```
            "Action": [
                "xray:PutTraceSegments"
            ],
            "Resource": "*"
        }
    ]
}
```

Canary 產生的追蹤會產生費用。如需 X-Ray 儲存定價的詳細資訊，請參閱 AWS X-Ray 定價。

在 VPC 上執行 Canary

您可以在 VPC 的端點以及公有內部端點上執行 Canary。若要在 VPC 上執行 Canary，您必須同時在 VPC 上啟用 DNS Resolution (DNS 解析) 和 DNS hostnames (DNS 主機名稱) 選項。如需詳細資訊，請參閱使用 DNS 與您的 VPC 搭配。

當您在 VPC 端點上執行初期測試時，您必須提供一種將指標 CloudWatch 及其成品傳送到 Amazon S3 的方法。如果 VPC 已啟用網際網路存取功能，您不需要再執行任何其他操作。Canary 會在 VPC 中執行，但可存取網際網路以上傳其指標和成品。

若未針對網際網路存取啟用 VPC，則有下列兩種選擇：

- 將它啟用以進行網際網路存取。如需詳細資訊，請參閱下一節「為 VPC 上的 Canary 提供網際網路存取」。
- 如果您想要將 VPC 保持私有狀態，可以將初期測試設定為透過私有 VPC 端點將其資料傳送到 CloudWatch Amazon S3。如果您尚未這樣做，則必 VPC 為 CloudWatch (com.amazonaws. # #。監控）和 Amazon S3 的閘道端點。如需詳細資訊，請參閱 使用 CloudWatch 和 CloudWatch Synthetics 與介面 VPC 端點 和 Simple Storage Service (Amazon S3) 的 Amazon VPC 端點。

為 VPC 上的 Canary 提供網際網路存取

請按照以下步驟為您的 VPC Canary 提供互聯網訪問權限，或為您的初期測試分配一個靜態 IP 地址

為 VPC 上的 Canary 提供網際網路存取

1. 建立 VPC 公有子網路的 NAT 閘道。如需說明，請參閱建立 NAT 閘道。
2. 將新路由新增至啟動 Canary 之私有子網路中的路由表。指定下列內容：
    - 針對 Destination (目的地)，輸入 **`0.0.0.0/0`**

- 對於目標，選擇 NAT 閘道，然後選擇您所建立之 NAT 閘道的 ID。
- 選擇 Save routes (儲存路由)。

如需將路由新增至路由表的詳細資訊，請參閱從路由表新增和移除路由。

> **Note**
>
> 請確定 NAT 閘道的路由處於 active (作用中) 狀態。如果 NAT 閘道已刪除，而您尚未更新路由，則它們會處於黑洞狀態。如需詳細資訊，請參閱使用 NAT 閘道。

加密 Canary 成品

CloudWatch Synthetics 將金絲雀文物（例如屏幕截圖，HAR 文件和報告）存儲在您的 Amazon S3 存儲桶中。依預設，這些成品會使用 AWS 受管理的金鑰進行靜態加密。如需詳細資訊，請參閱客戶金鑰和 AWS 金鑰。

您可以選擇使用不同的加密選項。 CloudWatch Synthetics 支持以下內容：

- SSE-S3 – 使用 Simple Storage Service (Amazon S3) 受管金鑰的伺服器端加密。
- SSE-KMS – 使用 AWS KMS 客戶受管金鑰的伺服器端加密 (SSE)。

如果您想要將預設加密選項與 AWS 受管理金鑰搭配使用，則不需要任何其他權限。

若要使用 SSE-S3 加密，建立或更新 Canary 時，您可以指定 SSE_S3 作為加密模式。您不需要任何其他許可，即可使用此加密模式。如需詳細資訊，請參閱使用伺服器端加密與 Simple Storage Service (Amazon S3) 受管加密金鑰 (SSE-S3) 保護資料。

若要使用 AWS KMS 客戶受管金鑰，請在建立或更新初期測試時將 SSE-KMS 指定為加密模式，並提供金鑰的 Amazon 資源名稱 (ARN)。您也可以使用跨帳戶 KMS 金鑰。

若要使用客戶受管金鑰，您需要以下設定：

- 您的 Canary IAM 角色必須有許可才能使用您的金鑰加密成品。如果正在使用視覺監控，則您也必須為其授予許可以解密成品。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": {"Effect": "Allow",
        "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt"
        ],
        "Resource": "Your KMS key ARN"
    }
}
```

- 您可以將 IAM 角色新增至金鑰政策，而不是將許可新增至 IAM 角色。如果對多個 Canary 使用相同的角色，則您應該考慮這種方法。

```
{
    "Sid": "Enable IAM User Permissions",
    "Effect": "Allow",
    "Principal": {
        "AWS": "Your synthetics IAM role ARN"
    },
    "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
    ],
    "Resource": "*"
}
```

- 如果您使用的是跨帳戶 KMS 金鑰，請參閱允許其他帳戶中的使用者使用 KMS 金鑰。

使用客戶受管金鑰時檢視加密的 Canary 成品

若要檢視初期測試成品，請更新您的客戶管理金鑰，以將解密權限授與檢視成品的使用者。 AWS KMS 或者，將解密許可新增至正在檢視成品的使用者或 IAM 角色。

預設 AWS KMS 政策可讓帳戶中的 IAM 政策允許存取 KMS 金鑰。如果您使用跨帳戶 KMS 金鑰，請參閱為什麼跨帳戶使用者嘗試存取透過自訂 AWS KMS 金鑰加密的 Amazon S3 物件時，會收到拒絕存取錯誤？ 。

如需因 KMS 金鑰而存取遭拒之問題的疑難排解詳細資訊，請參閱對金鑰存取進行疑難排解。

使用視覺監控時更新成品位置和加密

為了執行視覺監控， CloudWatch Synthetics 將您的屏幕截圖與選定為基準的運行中獲取的基準屏幕截圖進行比較。如果更新成品位置或加密選項，則您必須執行以下其中一項：

- 確保您的 IAM 角色對成品的先前 Simple Storage Service (Amazon S3) 位置和新 Simple Storage Service (Amazon S3) 位置具有足夠的許可。同時請確保其具有先前和新加密方法以及 KMS 金鑰的許可。
- 透過選取下一個 Canary 執行作為新基準來建立新基準。如果使用此選項，則您只需確保您的 IAM 角色具有足夠的許可，可供新的成品位置和加密選項使用。

建議您選取下一次執行作為新基準的第二個選項。這避免了依賴於您不再為 Canary 使用的成品位置或加密選項。

例如，假設您的 Canary 使用成品位置 A 和 KMS 金鑰 K 來上傳成品。如果將 Canary 更新為成品位置 B 和 KMS 金鑰 L，則您可以確保 IAM 角色擁有兩個成品位置 (A 和 B) 和兩個 KMS 金鑰 (K 和 L) 的許可。或者，您可以選取下一次執行作為新基準，並確保您的 Canary IAM 角色擁有成品位置 B 和 KMS 金鑰 L 的許可。

檢視 Canary 統計資料和詳細資訊

您可以檢視 Canary 的詳細資訊，並查看其執行的統計資料。

您必須登入具有足夠許可的帳戶，才能查看 Canary 執行結果的所有詳細資訊。如需詳細資訊，請參閱 CloudWatch加那利群島所需的角色和權限。

檢視 Canary 統計資料和詳細資訊

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，Synthetics 金絲雀。

   在您建立的 Canary 的詳細資訊中：

   - Status (狀態) 會以視覺化方式顯示您的 Canary 有多少個已通過最近的執行。
   - Groups (群組) 會顯示您已建立的群組，並顯示其中有多少個已失敗或有警示的 Canary。
   - Slowest performers (效能最差) 會顯示 Canary 效能最差的群組和區域。此項的計算方法是將群組或區域內所有 Canary (所選取的時間範圍內) 的平均持續時間相加，然後除以群組或區域中的 Canary 數量。如果選擇 Slowest (最差) 群組指標，系統會對資料表進行篩選，僅顯示效能最差的群組及其 Canary。資料表是依平均持續時間排序。
   - 頁面底部附近有一個顯示了所有 Canary 的表格。其中一個資料行顯示的是為每個 Canary 建立的警示。只會顯示符合 Canary 警示命名標準的警示。此標準為 `Synthetics-Alarm-`*`canaryName`*`-`*`index`* 。您在 CloudWatch 控制台的「Synthetics」部分中創建的 Canary 警報會自動使用此命名約定。如果您在 CloudWatch 主控台的 [警報] 區段或使用建立初

期測試警示 AWS CloudFormation，而且未使用此命名慣例，警示會運作，但警示不會出現在此清單中。

3. 若要查看單一 Canary 的詳細資訊，請在 Canaries 資料表中選擇相應的 Canary 名稱。

   在關於 Canary 的詳細資訊中：

   - Availability (可用性) 標籤會顯示有關此 Canary 最近執行的資訊。

     在 Canary runs (Canary 執行) 下，您可以選擇其中一行，以查看該執行的詳細資訊。

     在圖形下方，您可以依次選擇 Steps (步驟)、Screenshot (螢幕擷取畫面)、Logs (日誌) 或 HAR file (HAR 檔案) 來查看這些類型的詳細資訊。如果 Canary 已啟用作用中的追蹤，您也可以選擇 Traces (追蹤) 查看來自 Canary 執行的追蹤資訊。

     初期測試執行的日誌存放在 S3 儲存貯體和 CloudWatch 日誌中。

     螢幕擷取畫面會顯示您的客戶如何檢視您的網頁。您可以使用 HAR 檔案 (HTTP 封存檔案) 來檢視有關網頁的詳細效能資料。您可以分析 Web 請求的清單，並掌握效能問題，例如項目載入的時間。日誌檔案顯示 Canary 執行和網頁之間的互動記錄，並可用於識別錯誤的詳細資訊。

     如果 Canary 使用 `syn-nodejs-2.0-beta` 執行時間或更新版本，您可以依狀態碼、請求大小或持續時間來排序 HAR 檔案。

     Steps (步驟) 索引標籤會顯示一份清單，其中包含 Canary 的步驟、每個步驟的狀態、失敗原因、步驟執行後的 URL、螢幕擷取畫面以及步驟執行持續時間。對於具有 HTTP 步驟的 API Canary，如果您使用的是執行時間 `syn-nodejs-2.2` 或更新版本，則可以查看步驟和相應的 HTTP 請求。

     選擇 HTTP Requests (HTTP 請求) 標籤來檢視由 Canary 發出的每個 HTTP 請求的日誌。您可以檢視請求/回應標頭、回應內文、狀態碼、錯誤和效能計時 (總持續時間、TCP 連線時間、TLS 交握時間、第一個位元組時間和內容傳輸時間)。此處可擷取所有實際使用 HTTP/HTTPS 模組的 HTTP 請求。

     根據預設，在 API Canary 中，基於安全原因，報告中不會包含請求標頭、回應標頭、請求內文和回應內文。如果選擇包含它們，則資料只會存放於 S3 儲存貯體中。如需如何在報告內包含此資料的相關資訊，請參閱 executeHttpStep（stepName，請求選項，[回調]，[步驟配置]）。

     支援文字、HTML 和 JSON 的回應內文內容類型。支持文本/HTML，文本/純文本，應用程序/JSON 和應用程序/-1.0 的內容類型。x-amz-json不支援壓縮回應。

- 監控索引標籤會顯示此初期測試所發佈的測 CloudWatch 量結果圖表。如需這些指標的詳細資訊，請參閱 CloudWatch 加那利群島發布的指標。

  在金絲雀發佈的圖 CloudWatch 形下方是與金絲雀 Lambda 程式碼相關的 Lambda 指標圖表。
- Configuration (組態) 標籤會顯示有關 Canary 的組態和排程資訊。
- Groups (群組) 索引標籤會顯示與此 Canary 關聯的群組 (如果有)。
- Tags (標籤) 標籤會顯示與 Canary 相關的標籤。

CloudWatch 加那利群島發布的指標

加那利群島將以下指標發佈到`CloudWatchSynthetics`命名空間 CloudWatch 中。如需檢視 CloudWatch 測量結果的詳細資訊，請參閱檢視可用的指標。

指標	描述
`SuccessPercent`	此 Canary 成功執行且找不到失敗的百分比。 有效尺寸: CanaryName 有效的統計數字：平均 單位：百分比
`Duration`	Canary 執行的持續時間 (以毫秒為單位)。 有效尺寸: CanaryName 有效的統計數字：平均 單位：毫秒
`Errors`	初期測試無法執行其完整指令碼的次數。 有效尺寸: CanaryName 有效統計資訊：總和
`2xx`	傳回 OK 回應的 Canary 執行的網路請求數目，回應代碼介於 200 到 299 之間。

指標	描述
	為使用執行時間版本 syn-nodejs-2.0 或更新版本的 UI Canary 報告了此指標，也為使用執行時間版本 syn-nodejs-2.2 或更新版本的 API Canary 報告了此指標。 有效尺寸: CanaryName 有效統計資訊：總和 單位：計數
4xx	傳回 Error 回應的 Canary 執行的網路請求數目，回應代碼介於 400 到 499 之間。 為使用執行時間版本 syn-nodejs-2.0 或更新版本的 UI Canary 報告了此指標，也為使用執行時間版本 syn-nodejs-2.2 或更新版本的 API Canary 報告了此指標。 有效尺寸: CanaryName 有效統計資訊：總和 單位：計數
5xx	傳回 Fault 回應的 Canary 執行的網路請求數目，回應代碼介於 500 到 599 之間。 為使用執行時間版本 syn-nodejs-2.0 或更新版本的 UI Canary 報告了此指標，也為使用執行時間版本 syn-nodejs-2.2 或更新版本的 API Canary 報告了此指標。 有效尺寸: CanaryName 有效統計資訊：總和 單位：計數

指標	描述
Failed	無法執行的 Canary 執行次數。這些失敗與 Canary 本身有關。 有效尺寸: CanaryName 有效統計資訊：總和 單位：計數
Failed requests	Canary 在目標網站上執行失敗且沒有回應的 HTTP 請求數目。 有效尺寸: CanaryName 有效統計資訊：總和 單位：計數
VisualMonitoringSuccessPercent	在 Canary 執行期間，成功比對基線螢幕擷取畫面的視覺化比較百分比。 有效尺寸: CanaryName 有效的統計數字：平均 單位：百分比
VisualMonitoringTotalComparisons	在 Canary 執行期間發生的視覺化比較總數。 有效尺寸: CanaryName 單位：計數

> **Note**
>
> 使用來自 Synthetics 程式庫的 executeStep() 或 executeHttpStep() 方法的 Canary 還會為每個步驟發佈具有維度 CanaryName 和 StepName 的 SuccessPercent 和 Duration 指標。

編輯或刪除 Canary

您可以編輯或刪除現有 Canary。

編輯 Canary

當您編輯 Canary 時，即使您沒有變更其排程，排程也會根據您編輯 Canary 的時間進行重設。例如，如果您的 Canary 每小時執行一次，並且您編輯該 Canary，則在編輯完成後會立即執行，然後每小時執行一次。

若要編輯或更新 Canary

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，Synthetics 金絲雀。
3. 選取 Canary 名稱旁邊的按鈕，然後選擇 Actions (動作)、Edit (編輯)。
4. (選用) 如果此 Canary 執行螢幕擷取畫面的視覺化監控，而您想要將下一次的 Canary 執行設定為基準，請選取 Set next run as new baseline (將下一次執行設定為新基準)。
5. (選用) 如果此 Canary 執行螢幕擷取畫面的視覺化監控，而您想要從視覺化監控中移除螢幕擷取畫面，或者您想要在視覺化比較期間指定要忽略螢幕擷取畫面的部分，請在 Visual Monitoring (視覺化監控) 下選擇 Edit Baseline (編輯基準)。

   出現螢幕擷取畫面時，您可以執行下列其中一項：

   - 若要移除用於視覺化監控的螢幕擷取畫面，請選取 Remove screenshot from visual test baseline (從視覺化測試基準中移除螢幕擷取畫面)。
   - 若要指定要在視覺化比較期間忽略的螢幕擷取畫面的部分，請按一下並拖曳以繪製要忽略的螢幕區域。一旦您對比較期間想要忽略的所有區域執行此操作，請選擇 Save (儲存)。
6. 對 Canary 進行任何其他您想要的變更，然後選擇 Save (儲存)。

刪除 Canary

刪除 Canary 時，您可以選擇是否同時刪除 Canary 使用和建立的其他資源。刪除 Canary 時，也應一併刪除下列內容：

- 此 Canary 使用的 Lambda 函數和圖層。它們的字首是 `cwsyn-`*`MyCanaryName`*。
- CloudWatch 為此金絲雀創建的警報。這些警示的名稱以 `Synthetics-Alarm-`*`MyCanaryName`* 為開頭。如需刪除警示的詳細資訊，請參閱編輯或刪除 CloudWatch 鬧鐘。
- Simple Storage Service (Amazon S3) 物件和儲存貯體，例如 Canary 的結果位置和成品位置。

- 為 Canary 建立的 IAM 角色。它們的名稱為 `role/service-role/CloudWatchSyntheticsRole-`*`MyCanaryName`*。
- 針對初期測試建立的 CloudWatch 記錄檔中的記錄群組。這些日誌群組具有下列名稱：`/aws/lambda/cwsyn-`*`MyCanaryName`*`-`*`randomId`*。

刪除 Canary 之前，您可能需要檢視 Canary 詳細資料，並記下此資訊。這樣，您才能在刪除 Canary 後刪除正確的資源。

若要刪除 Canary

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，Synthetics 金絲雀。
3. 如果 Canary 目前處於 RUNNING 狀態，必須將其停用。您只能刪除處於 STOPPED、READY(NOT_STARTED) 或 ERROR 狀態的 Canary。

   若要停用 Canary，請選取 Canary 名稱旁的按鈕，然後選擇 Actions (動作)、Stop (停用)。
4. 選取 Canary 名稱旁邊的按鈕，然後選擇 Actions (動作)、Delete (刪除)。
5. 選擇是否同時刪除 Canary 建立和使用的其他資源。這包含 Lambda 函數和層，以及 Canary 的 IAM 角色和 IAM 政策。

   若要刪除 Canary 的 IAM 角色和 IAM 政策，您必須擁有足夠的權限。如需詳細資訊，請參閱 AWS CloudWatch Synthetics 的管理 (預先定義) 政策。
6. 在方塊中輸入 **`Delete`**，然後選擇 Delete (刪除)。
7. 如本節稍早所列，刪除為 Canary 使用和建立的其他資源。

啟動、停止、刪除或更新多個 Canary 的執行階段

您可以使用一個動作來停止、啟動、刪除或更新多達五個 Canary 的執行階段。如果您更新 Canary 的執行階段，它會更新為適用於 Canary 使用的語言和架構的最新執行階段。

如果您選取多個 Canary，且只有部分處於對所選動作有效的狀態，則只會在該動作有效的 Canary 上執行動作。例如，如果您選取一些正在執行的 Canary 和一些沒有執行的 Canary，並且您選擇啟動 Canary，則尚未執行的 Canary 將會啟動，而已在執行的 Canary 不會受到影響。

如果您選取的 Canary 都不適用於某個動作，則該動作將無法在選單中使用。

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/

2. 在導航窗格中，選擇應用程序信號，Synthetics 金絲雀。
3. 選取要停止、啟動或刪除的 Canary 旁邊的核取方塊。
4. 選擇 Actions (動作)，然後選擇 Start (啟動)、Stop (停止)、Delete (刪除) 或 Update Runtime (更新執行階段)。

用 Amazon 監控金絲雀事件 EventBridge

Amazon EventBridge 事件規則可以在金絲雀變更狀態或完成執行時通知您。 EventBridge提供描述 AWS 資源變更的系統事件 near-real-time 串流。 CloudWatch Synthetics 將這些事件發送到盡最大努力的基礎 EventBridge 上。盡最大努力傳遞意味著 CloudWatch Synthetics 試圖將所有事件發送到 EventBridge，但在某些極少數情況下，事件可能無法傳遞。 EventBridge 處理所有接收到的事件至少一次。此外，您的事件接聽程式可能不會依事件發生順序接收事件。

> **Note**
>
> Amazon EventBridge 是一種事件匯流排服務，您可以使用它將應用程式與來自各種來源的資料連接起來。如需詳細資訊，請參閱什麼是 Amazon EventBridge？ 在 Amazon 用 EventBridge 戶指南。

CloudWatch 當金絲雀改變狀態或完成運行時，Synthetics 成材料會發出事件。您可以建立包含事件模式的 EventBridge 規則，以符合所有從合成傳送的事件類型，或僅符 CloudWatch Synthetics 特定事件類型的事件類型。當初期測試觸發規則時，會 EventBridge叫用規則中定義的目標動作。這可讓您傳送通知、擷取事件資訊、採取修正動作、回應 Canary 狀態變更或完成 Canary 執行。例如，您可以為下列使用案例建立規則：

- 調查 Canary 執行失敗的時間
- 調查 Canary 何時進入 `ERROR` 狀態
- 追蹤 Canary 的生命週期
- 作為工作流程的一部分，監控 Canary 執行成功或失敗

來自 CloudWatch Synthetics 的示例事件

本節列出了 CloudWatch Synthetics 的示例事件。如需有關事件格式的詳細資訊，請參閱 EventBridge.

Canary 狀態變更

在此事件類型中，current-state 和 previous-state 可以是下列項目：

CREATING | READY | STARTING | RUNNING | UPDATING | STOPPING | STOPPED | ERROR


```
{
                "version": "0",
                "id": "8a99ca10-1e97-2302-2d64-316c5dedfd61",
                "detail-type": "Synthetics Canary Status Change",
                "source": "aws.synthetics",
                "account": "123456789012",
                "time": "2021-02-09T22:19:43Z",
                "region": "us-east-1",
                "resources": [],
                "detail": {
                                "account-id": "123456789012",
                                "canary-id": "EXAMPLE-dc5a-4f5f-96d1-989b75a94226",
                                "canary-name": "events-bb-1",
                                "current-state": "STOPPED",
                                "previous-state": "UPDATING",
                                "source-location": "NULL",
                                "updated-on": 1612909161.767,
                                "changed-config": {
                                                "executionArn": {
                                                                "previous-value":
 "arn:aws:lambda:us-east-1:123456789012:function:cwsyn-events-bb-1-af3e3a05-
dc5a-4f5f-96d1-989EXAMPLE:1",
                                                                "current-value":
 "arn:aws:lambda:us-east-1:123456789012:function:cwsyn-events-bb-1-af3e3a05-
dc5a-4f5f-96d1-989EXAMPLE:2"
                                                },
                                                "vpcId": {
                                                                "current-value": "NULL"
                                                },
                                                "testCodeLayerVersionArn": {
                                                                "previous-
value": "arn:aws:lambda:us-east-1:123456789012:layer:cwsyn-events-bb-1-af3e3a05-
dc5a-4f5f-96d1-989EXAMPLE:1",
                                                                "current-value":
 "arn:aws:lambda:us-east-1:123456789012:layer:cwsyn-events-bb-1-af3e3a05-
dc5a-4f5f-96d1-989EXAMPLE:2"
                                                }
                                },
                                "message": "Canary status has changed"
                }
```

```
}
```

已完成成功的 Canary 執行

```
{
                "version": "0",
                "id": "989EXAMPLE-f4a5-57a7-1a8f-d9cc768a1375",
                "detail-type": "Synthetics Canary TestRun Successful",
                "source": "aws.synthetics",
                "account": "123456789012",
                "time": "2021-02-09T22:24:01Z",
                "region": "us-east-1",
                "resources": [],
                "detail": {
                                "account-id": "123456789012",
                                "canary-id": "989EXAMPLE-dc5a-4f5f-96d1-989b75a94226",
                                "canary-name": "events-bb-1",
                                "canary-run-id": "c6c39152-8f4a-471c-9810-989EXAMPLE",
                                "artifact-location": "cw-syn-results-123456789012-us-
east-1/canary/us-east-1/events-bb-1-ec3-28ddbe266797/2021/02/09/22/23-41-200",
                                "test-run-status": "PASSED",
                                "state-reason": "null",
                                "canary-run-timeline": {
                                                "started": 1612909421,
                                                "completed": 1612909441
                                },
                                "message": "Test run result is generated successfully"
                }
}
```

已完成失敗的 Canary 執行

```
{
                "version": "0",
                "id": "2644b18f-3e67-5ebf-cdfd-bf9f91392f41",
                "detail-type": "Synthetics Canary TestRun Failure",
                "source": "aws.synthetics",
                "account": "123456789012",
                "time": "2021-02-09T22:24:27Z",
                "region": "us-east-1",
                "resources": [],
                "detail": {
                                "account-id": "123456789012",
```

```
                                "canary-id": "af3e3a05-dc5a-4f5f-96d1-9989EXAMPLE",
                                "canary-name": "events-bb-1",
                                "canary-run-id": "0df3823e-7e33-4da1-8194-
b04e4d4a2bf6",
                                "artifact-location": "cw-syn-results-123456789012-us-
east-1/canary/us-east-1/events-bb-1-ec3-989EXAMPLE/2021/02/09/22/24-21-275",
                                "test-run-status": "FAILED",
                                "state-reason": "\"Error: net::ERR_NAME_NOT_RESOLVED
 \""
                                "canary-run-timeline": {
                                                "started": 1612909461,
                                                "completed": 1612909467
                                },
                                "message": "Test run result is generated successfully"
                }
}
```

事件可能會重複或不按順序。若要判斷事件的順序，請使用 `time` 屬性。

建立 EventBridge 規則的先決條件

在為 CloudWatch Synthetics 建立 EventBridge 規則之前，您應該執行下列動作：

- 熟悉中的事件、規則和目標。 EventBridge
- 建立並設定 EventBridge 規則呼叫的目標。規則可以叫用許多類型的目標，包括：
  - Amazon SNS 主題
  - AWS Lambda 函數
  - Kinesis 串流
  - Amazon SQS 佇列

如需詳細資訊，請參閱什麼是 Amazon EventBridge？ 並在 Amazon 用 EventBridge 戶指南EventBridge中開始使用 Amazon。

建立 EventBridge 規則 (CLI)

以下範例中的步驟會建立一個 EventBridge 規則，該規則會`my-canary-name`在中命名的初期測試`us-east-1`完成執行或變更狀態時發佈 Amazon SNS 主題。

1. 建立 規則。

```
aws events put-rule \
  --name TestRule \
  --region us-east-1 \
  --event-pattern "{\"source\": [\"aws.synthetics\"], \"detail\": {\"canary-name\":
 [\"my-canary-name\"]}}"
```

模式省略的任何屬性會遭到忽略。

2. 新增主題作為規則目標。

   - 使用 Amazon SNS 主題的 Amazon Resource Name (ARN) 取代為 *topic-arn*。

```
aws events put-targets \
  --rule TestRule \
  --targets "Id"="1","Arn"="topic-arn"
```

> **Note**
>
> 要允許 Amazon 調 EventBridge 用您的目標主題，您必須在主題中添加基於資源的政策。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南中的 Amazon SNS 許可。

如需詳細資訊，請參閱 Amazon EventBridge 使用者指南 EventBridge中的事件和事件模式。

CloudWatch 明顯地執行發射和 A/B 實驗

在推出此功能時，您可以使用 Amazon CloudWatch Edition 將新功能提供給指定百分比的使用者，以安全地驗證新功能。您可以監控新功能的效能，以協助您決定何時要提升使用者的流量。這可協助您在完全啟動功能之前降低風險並識別意外的後果。

您也可以執行 A/B 實驗，根據證據和資料作出功能設計決策。實驗可以一次測試多達五個變化。Evidently 會收集實驗資料並使用統計方法進行分析。它還提供明確的建議，說明哪些變化效能更好。您可以測試面向使用者的功能和後端功能。

Evidently 定價

Evidently 會根據 Evidently 事件和 Evidently 分析單位向您的帳戶收費。Evidently 事件包括資料事件 (例如點選和頁面檢視)，以及決定要提供給使用者之功能變化的指派事件。

Evidently 分析單位是根據您在 Evidently 中建立的規則，從 Evidently 事件中產生的。分析單位是事件上的規則相符數量。例如，使用者點選事件可能會產生單一 Evidently 分析單位，即點選計數。另一個範例是使用者結帳事件，可能會產生兩個 Evidently 分析單位，結帳值和購物車中的項目數量。如需有關定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

CloudWatch 顯然目前在以下地區可用：

- 美國東部 (俄亥俄)
- 美國東部 (維吉尼亞北部)
- 美國西部 (奧勒岡)
- 亞太區域 (新加坡)
- 亞太區域 (雪梨)
- 亞太區域 (東京)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (斯德哥爾摩)

主題

- 使用 Evidently 的 IAM 政策
- 建立專案、功能、啟動和實驗
- 管理功能、啟動和實驗
- 將程式碼新增至應用程式
- 專案資料儲存
- Evidently 如何計算結果
- 在儀表板中檢視啟動結果
- 在儀表板中檢視實驗結果
- 如何 CloudWatch 明顯地收集和存儲數據
- 使用 Evidently 的服務連結角色
- CloudWatch 顯然配額
- 教學：使用 Evidently 範例應用程式進行 A/B 測試

使用 Evidently 的 IAM 政策

若要完全 CloudWatch 管理，您必須以具有下列權限的 IAM 使用者或角色身分登入：

- AmazonCloudWatchEvidentlyFullAccess 政策
- ResourceGroupsandTagEditorReadOnlyAccess 政策

此外，若要建立將評估事件存放在 Amazon S3 或 CloudWatch 日誌中的專案，您需要下列許可：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketPolicy",
                "s3:PutBucketPolicy",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:DescribeResourcePolicies",
                "logs:PutResourcePolicy"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

CloudWatch RUM 整合的其他權限

此外，如果您打算管理明顯的啟動或與 Amazon CloudWatch RUM 整合的實驗，並使用 CloudWatch RUM 指標進行監控，則需要 AmazonCloudWatchRUM FullAccess 政策。若要建立 IAM 角色以授與 CloudWatch RUM 網路用戶端將資料傳送至 CloudWatch RUM 的權限，您需要下列權限：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": [
                "arn:aws:iam::*:role/service-role/CloudWatchRUMEvidentlyRole-*",
                "arn:aws:iam::*:policy/service-role/CloudWatchRUMEvidentlyPolicy-*"
            ]
        }
    ]
}
```

Evidently 唯讀存取的許可

對於其他需要查看「明顯」數據但不需要創建 Eviatic 資源的用戶，您可以授予該AmazonCloudWatchEvidentlyReadOnlyAccess策略。

建立專案、功能、啟動和實驗

要開始使用 CloudWatch 顯而易見，對於功能啟動或 A/B 實驗，您首先創建一個項目。專案是資源的邏輯分組。在專案中，您可以建立具有您想要測試或啟動之變化的功能。您可以在建立啟動或實驗前建立功能，也可以同時建立功能。

主題

- 建立新專案
- 使用 AWS AppConfig提供的用戶端評估
- 將功能新增至專案
- 使用客群功能聚焦您的受眾
- 建立啟動

- 建立實驗

建立新專案

使用這些步驟來設置一個新的 CloudWatch 顯而易見的項目。

要創建一個新的 CloudWatch 顯而易見的項目

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，顯而易見。
3. 選擇建立專案。
4. 在「專案名稱」中，輸入用於識別此專案的 CloudWatch 名稱。

   您可以選擇新增專案描述。
5. 對於 Evaluation event storage (評估事件儲存)，請選擇您是否要存放您使用 Evidently 收集的評估事件。即使您沒有存放這些事件，Evidently 也會彙總這些事件，以建立可在 Evidently 儀表板中檢視的指標和其他實驗資料。如需詳細資訊，請參閱 專案資料儲存。
6. 對於 Use client-side evaluation (使用用戶端評估)，選擇是否要啟用此專案的用戶端評估。透過用戶端評估，您的應用程式可以在本機指派變體給使用者工作階段，而不是呼叫EvaluateFeature作業。這樣可以減輕 API 呼叫所帶來的延遲和可用性風險。如需詳細資訊，請參閱 使用 AWS AppConfig提供的用戶端評估。

   若要建立具有用戶端評估的專案，您必須擁有 `evidently:ExportProjectAsConfiguration` 許可。

   如果您啟用用戶端評估，請同時執行下列動作：

   a. 選擇要使用現有的應用 AWS AppConfig 程式還是建立新的應用程式。

   b. 選擇是使用現有 AWS AppConfig 環境還是建立新環境。

      如需中應用程式和環境的詳細資訊 AWS AppConfig，請參閱如何 AWS AppConfig 運作。
7. (選用) 若要新增標籤至此專案，請選擇 Tags (標籤)、Add new tag (新增標籤)。

   之後，在 Key (索引鍵) 中，輸入標籤的名稱。您可以在 Value (值) 中為標籤新增選用值。

   若要新增另一個標籤，請再次選擇 Add new tag (新增標籤)。

   如需詳細資訊，請參閱標記 AWS 資源。

8. 選擇建立專案。

使用 AWS AppConfig提供的用戶端評估

您可以在專案中使用由 AWS AppConfig (用戶端評估) 提供支援的用戶端評估，這可讓您的應用程式將變體指派給本機的使用者工作階段，而不是透過呼叫EvaluateFeature作業來指派變體。這樣可以減輕 API 呼叫所帶來的延遲和可用性風險。

若要使用用戶端評估，請將 AWS AppConfig Lambda 延伸模組作為層附加至 Lambda 函數，並設定環境變數。用戶端評估會在本機主機上以副程序的形式執行。然後，您可以呼叫EvaluationFeature和PutProjectEvent作業`localhost`。用戶端評估程序會處理變化指派、快取和資料同步。如需相關資訊 AWS AppConfig，請參閱如何 AWS AppConfig 運作。

與整合時 AWS AppConfig，您可以將 AWS AppConfig 應用程式識別碼和 AWS AppConfig 環境 ID 指定為「顯而易見」。您可以在各 Evidently 專案中使用相同的應用程式 ID 和環境 ID。

當您創建啟用客戶端評估的項目時，顯然會為該項目創建 AWS AppConfig 配置文件。每個專案的組態設定檔都不同。

用戶端評估存取控制

Evidently 用戶端評估使用的存取控制機制與其餘 Evidently 功能不同。我們強烈建議您了解這一點，以便您可以實作適當的安全措施。

使用 Evidently，您可以建立 IAM 政策，這些政策會限制使用者可以對個別資源執行的動作。例如，您可以建立不允許使用者執EvaluateFeature行動作的使用者角色。如需有關可使用 IAM 政策控制的明顯動作的詳細資訊，請參閱 Amazon CloudWatch 明顯定義的動作。

用戶端評估模型允許對使用專案中繼資料的 Evidently 功能進行本機評估。已啟用用戶端評估的專案使用者可以針對本機主機端點呼叫 EvaluateFeatureAPI，而且此 API 呼叫無法明顯達到，且未經 Eviatic 服務的 IAM 政策進行驗證。即使使用者沒有使用該EvaluateFeature動作的 IAM 權限，此呼叫仍會成功。不過，使用者仍然需要代理程式緩衝評估事件或自訂事件的PutProjectEvents權限，以及將資料卸載至 Eviated 非同步。

此外，使用者必須擁有 `evidently:ExportProjectAsConfiguration` 許可才能夠建立使用用戶端評估的專案。這可協助您控制存EvaluateFeature取用戶端評估期間呼叫的動作。

如果您未留意，用戶端評估安全性模型可能會推翻您在其餘 Evidently 功能中設定的政策。具有`evidently:ExportProjectAsConfiguration`權限的使用者可以在啟用用戶端評估的情況下建立專案，然後使用該EvaluateFeature動作對該專案進行用戶端評估，即使他們明確拒絕 IAM 政策中的EvaluateFeature動作也是如此。

開始使用 Lambda

Evidently 目前支援透過使用 AWS Lambda 環境進行用戶端評估。若要開始使用，請先決定要使用的 AWS AppConfig 應用程式和環境。選擇現有的應用程式和環境，或建立新的應用程式和環境。

下列範例 AWS AppConfig AWS CLI 指令會建立應用程式和環境。

```
aws appconfig create-application --name YOUR_APP_NAME
```

```
aws appconfig create-environment --application-id YOUR_APP_ID --
name YOUR_ENVIRONMENT_NAME
```

接下來，使用這些 AWS AppConfig 資源創建一個明顯的項目。如需詳細資訊，請參閱 建立新專案。

在 Lambda 中支援使用 Lambda 層進行用戶端評估。這是一個公共層AWS-AppConfig-Extension，它是 AWS AppConfig 服務創建的公共 AWS AppConfig 擴展的一部分。如需有關 Lambda 層的詳細資訊，請參閱層。

若要使用用戶端評估，您必須將此層新增至 Lambda 函數，並設定許可和環境變數。

將 Evidently 用戶端評估 Lambda 層新增至 Lambda 函數並進行設定

1. 如果您尚未建立 Lambda 函數，請進行建立。
2. 將用戶端評估層新增至函數。您可以指定其 ARN，也可以從 AWS 圖層清單中選取它 (如果尚未指定)。如需詳細資訊，請參閱設定函數以使用層和 AWS AppConfig Lambda 擴充功能的可用版本。
3. 建立具有下列內容命名EvidentlyAppConfigCachingAgentPolicy的 IAM 政策，並將其附加到函數的執行角色。如需詳細資訊，請參閱 Lambda 執行角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "appconfig:GetLatestConfiguration",
                "appconfig:StartConfigurationSession",
                "evidently:PutProjectEvents"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

4. 將所需的環境變數 AWS_APPCONFIG_EXTENSION_EVIDENTLY_CONFIGURATIONS 新增至 Lambda 函數。此環境變數會指定「明顯」專案與 AWS AppConfig 資源之間的對應。

   如果您將此函數用於某個 Evidently 專案，請將環境變數的值設定為：applications/*APP_ID*/environments/*ENVIRONMENT_ID*/configurations/*PROJECT_NAME*

   如果您將此函數用於多個 Evidently 專案，請使用逗號來分隔值，如下列範例所示：applications/*APP_ID_1*/environments/*ENVIRONMENT_ID_1*/configurations/*PROJECT_NAME_1*, applications/*APP_ID_2*/environments/*ENVIRONMENT_ID_2*/configurations/*PROJECT_NAME_2*

5. (選用) 設定其他環境變數。如需詳細資訊，請參閱設定 AWS AppConfig Lambda 擴充功能。

6. 在應用程式中，透過將 EvaluateFeature 傳送至 localhost 以在本機取得 Evidently 評估。

   Python 範例：

```
import boto3
from botocore.config import Config

def lambda_handler(event, context):
    local_client = boto3.client(
        'evidently',
        endpoint_url="http://localhost:2772",
        config=Config(inject_host_prefix=False)
    )
    response = local_client.evaluate_feature(
        project=event['project'],
        feature=event['feature'],
        entityId=event['entityId']
    )
    print(response)
```

   Node.js 範例：

```
const AWS = require('aws-sdk');
const evidently = new AWS.Evidently({
    region: "us-west-2",
    endpoint: "http://localhost:2772",
    hostPrefixEnabled: false
```

```
});

exports.handler = async (event) => {

    const evaluation = await evidently.evaluateFeature({
        project: 'John_ETCProject_Aug2022',
        feature: 'Feature_IceCreamFlavors',
        entityId: 'John'
    }).promise()

    console.log(evaluation)
    const response = {
        statusCode: 200,
        body: evaluation,
    };
    return response;
};
```

Kotlin 範例：

```
String localhostEndpoint = "http://localhost:2772/"
public AmazonCloudWatchEvidentlyClient getEvidentlyLocalClient() {
    return AmazonCloudWatchEvidentlyClientBuilder.standard()

 .withEndpointConfiguration(AwsClientBuilder.EndpointConfiguration(localhostEndpoint,
 region))

 .withClientConfiguration(ClientConfiguration().withDisableHostPrefixInjection(true))
        .withCredentials(credentialsProvider)
        .build();
}

AmazonCloudWatchEvidentlyClient evidently = getEvidentlyLocalClient();

// EvaluateFeature via local client.
EvaluateFeatureRequest evaluateFeatureRequest = new
 EvaluateFeatureRequest().builder()
   .withProject(${YOUR_PROJECT}) //Required.
   .withFeature(${YOUR_FEATURE}) //Required.
   .withEntityId(${YOUR_ENTITY_ID}) //Required.
   .withEvaluationContext(${YOUR_EVAL_CONTEXT}) //Optional: a JSON object of
 attributes that you can optionally pass in as part of the evaluation event sent to
 Evidently.
```

```
    .build();

EvaluateFeatureResponse evaluateFeatureResponse =
 evidently.evaluateFeature(evaluateFeatureRequest);

// PutProjectEvents via local client.
PutProjectEventsRequest putProjectEventsRequest = new
 PutProjectEventsRequest().builder()
    .withData(${YOUR_DATA})
    .withTimeStamp(${YOUR_TIMESTAMP})
    .withType(${YOUR_TYPE})
    .build();

PutProjectEvents putProjectEventsResponse =
 evidently.putProjectEvents(putProjectEventsRequest);
```

設定用戶端將資料傳送至 Evidently 的頻率

若要指定用戶端評估將資料傳送至 Evidently 的頻率，您可以選擇性地設定兩個環境變數。

- AWS_APPCONFIG_EXTENSION_EVIDENTLY_EVENT_BATCH_SIZE 指定在將每個專案的事件傳送到 Evidently 之前批次處理的事件數量。有效值是介於 1 到 50 之間的整數，預設值為 40。
- AWS_APPCONFIG_EXTENSION_EVIDENTLY_BATCH_COLLECTION_DURATION 指定在將事件傳送到 Evidently 之前等待的持續時間 (以秒為單位)。預設值為 30。

故障診斷

請使用下列資訊，協助疑難排解使用 CloudWatch Evenly 搭配用戶端評估的 AWS AppConfig問題。

調用 EvaluateFeature 操作時發生錯誤（BadRequestException）：提供的路徑不支持 HTTP 方法

環境變數的設定可能不正確。例如，您可能已使用 EVIDENTLY_CONFIGURATIONS 作為環境變數名稱 (而不是 AWS_APPCONFIG_EXTENSION_EVIDENTLY_CONFIGURATIONS)。

ResourceNotFoundException：找不到部署

您對專案中繼資料的更新尚未部署到 AWS AppConfig。檢查您用於用戶端評估的 AWS AppConfig 環境中是否有作用中部署。

ValidationException：沒有明顯的項目配置

可能使用不正確的專案名稱設定了 AWS_APPCONFIG_EXTENSION_EVIDENTLY_CONFIGURATIONS 環境變數。

將功能新增至專案

CloudWatch 顯而易見的功能代表您要啟動或要測試變體的功能。

您必須先建立專案，才能新增功能。如需詳細資訊，請參閱 建立新專案。

將功能新增至專案

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，顯而易見。
3. 選擇專案的名稱。
4. 選擇 Add feature (新增功能)。
5. 對於 Feature name (功能名稱)，請輸入用於識別此專案中此功能的名稱。

   您可以選擇新增功能描述。
6. 對於 Feature variations (功能變化)，對於 Variation type (變化類型)，請選擇 Boolean (布林型)、Long (長整數)、Double (雙字)，或 String (字串)。如需詳細資訊，請參閱 變化類型。
7. 為您的功能新增最多五種變化。每個變化的 Value (值) 必須適用於您選取的 Variation type (變化類型)。

   指定其中一個變化作為預設值。這是比較其他變化的基準線，應該是目前正在為使用者提供服務的變化。這也是服務於未新增至此功能的啟動或實驗之使用者的變化。
8. 選擇 Sample code (範本程式碼)。程式碼範例會顯示您需要新增至應用程式以設定變化和為應用程式指派使用者工作階段的項目。您可以在 Java 和 Python 之間 JavaScript進行選擇作為代碼。

   您現在不需要將程式碼新增至您的應用程式，但必須在開始啟動或實驗之前這麼做。

   如需詳細資訊，請參閱 將程式碼新增至應用程式。
9. (選用) 若要指定某些使用者可以一直看到某些變化，請選擇 Override (覆寫)、Add override (新增覆寫)。然後，輸入使用者 ID、帳戶 ID 或 Identifier (識別符) 中的一些其他識別符，並指定使用者應該看到的變化。

   當您想確保他們看到特定的變化時，這對您自己的測試團隊成員或其他內部使用者都非常有幫助。指派了覆寫的使用者工作階段不會影響啟動或實驗指標。

您可以再次選擇 [新增覆寫]，對多達 20 個使用者重複此動作。

10. (選用) 若要新增標籤至此功能，請選擇 Tags (標籤)、Add new tag (新增標籤)。

    之後，在 Key (索引鍵) 中，輸入標籤的名稱。您可以在 Value (值) 中為標籤新增選用值。

    若要新增另一個標籤，請再次選擇 Add new tag (新增標籤)。

    如需詳細資訊，請參閱標記 AWS 資源。

11. 選擇 Add feature (新增功能)。

變化類型

當您建立功能並定義變化時，必須選取 variation type (變化類型)。可能類型如下：

- Boolean
- 長整數
- 雙精度浮點數
- 字串

變化類型會設定如何在程式碼中區分不同的變化。您可以使用變化類型來簡化「 CloudWatch顯而易見」的實作，也可以簡化在啟動和實驗中修改特徵的過程。

例如，如果您定義具有長整數變化類型的功能，則指定用來區分變化的整數可以是直接傳遞到程式碼中的數字。其中一個範例可能是測試按鈕的像素大小。變化類型的值可以是每個變化中使用的像素數。每個變化的代碼可以讀取變化類型值，並將其用作按鈕大小。若要測試新的按鈕大小，您可以變更用於變化值的數字，而不需要進行任何其他程式碼變更。

當您在功能中為變體類型設置值時，應避免為多個變體分配相同的值，除非您想進行 A/A 測試以初始嘗試 CloudWatch 顯而易見，或者有其他理由這樣做。

Evidently 沒有對 JSON 作為類型的原生支持，但您可以在字串變化類型中傳入 JSON，並在程式碼中解析該 JSON。

使用客群功能聚焦您的受眾

您可以定義受眾客群並在您的發布會和實驗中使用。客群是指具有一或多個共通特徵的部分受眾。例如，Chrome 瀏覽器使用者、位於歐洲的使用者，或是歐洲的 Firefox 瀏覽器使用者，並同時符合您應用程式收集的其他條件 (例如年齡)。

在實驗中使用客群能夠限制該實驗僅評估符合客群條件的使用者。在發布會中使用一或多個客群時，您可以針對不同的受眾客群定義不同的流量分割。

客群規則模式語法

若要建立客群，請定義客群規則模式。指定您要用哪些屬性來評估使用者工作階段是否在客群中。您建立的模式會與 Evidently 在使用者工作階段中找到 `evaluationContext` 的值進行比較。如需詳細資訊，請參閱 使用 EvaluateFeature。

若要建立客群規則模式，請指定要模式比對的欄位。您也可以在模式中運用邏輯，例如 `And`、`Or`、`Not` 和 `Exists`。

若要讓 `evaluationContext` 比對模式，`evaluationContext` 必須比對規則模式的所有部分。Evidently 會忽略 `evaluationContext` 中未包含在規則模式的欄位。

規則模式比對的值遵循 JSON 規則。您可以加入括在引號 (") 中的字串,、數字和關鍵字 `true`、`false`、`null`。

對於字符串，顯然使用精確 character-by-character 匹配而不折疊大小寫或任何其他字符串規範化。因此，規則比對會區分大小寫。例如，如果 `evaluationContext` 包含 `browser` 屬性，但規則模式檢查的是 `Browser`，就不會符合。

對於數字，Evidently 會使用字串表示法。例如，300、300.0 和 3.0e2 不會被視為相等。

當您撰寫規則模式來比對 `evaluationContext` 時，您可以使用 `TestSegmentPattern` API 或 `test-segment-pattern` CLI 指令，來測試您的模式是否能比對出正確的 JSON。如需詳細資訊，請參閱 TestSegmentPattern。

以下摘要顯示 Evidently 客群模式所提供的全部比較運算子。

Comparison (比較)	範例	Rule syntax (規則語法)
Null	UserID 為 Null 值	`{ "UserID": [ null ] }`
空白	LastName 是空的	`{ "LastName": [""] }`

Comparison (比較)	範例	Rule syntax (規則語法)
等於	Browser 為 "Chrome"	`{ "Browser": [ "Chrome" ] }`
及	Country 為 "France" 且 Device 為 "Mobile"	`{ "Country": [ "France" ], "Device": ["Mobile"] }`
Or (單一屬性的多個值)	Browser 為 "Chrome" 或 "Firefox"	`{ "Browser": ["Chrome", "Firefox"] }`
Or (不同的屬性)	瀏覽器為 "Safari" 或裝置為 "Tablet"	`{ "$or": [ {"Browser": ["Safari"]}, {"Device": ["Tablet" }] ] }`
Not	瀏覽器是 "Safari" 以外的所有值	`{ "Browser": [ { "anything-but": [ "Safari" ] } ] }`

Comparison (比較)	範例	Rule syntax (規則語法)
數字 (等於)	價格為 100	`{ "Price": [ { "numeric": [ "=", 100 ] } ] }`
數字 (範圍)	價格大於 10，且小於或等於 20	`{ "Price": [ { "numeric": [ ">", 10, "<=", 20 ] } ] }`
存在	時間欄位存在	`{ "Age": [ { "exists": true } ] }`
不存在	時間欄位不存在	`{ "Age": [ { "exists": false } ] }`
以前綴開頭	區域為美國境內	`{ "Region": [ {"prefix": "us-" } ] }`
以後綴結束	位置具有後綴 "West"	`{ "Region": [ {"suffix": "West" } ] }`

客群規則範例

以下所有的範例皆假設您傳遞值給 evaluationContext，且其欄位標籤和值與您在規則模式中使用的相同。

以下範例會比對 Browser 為 Chrome 還是 Firefox，且 Location 是否為 US-West (美國西部)。

```
{
  "Browser": ["Chrome", "Firefox"],
  "Location": ["US-West"]
}
```

以下範例會比對 Browser 是否為除 Chrome 以外的任何瀏覽器，Location 是否開頭為 US，且 Age 欄位是否存在。

```
{
  "Browser": [ {"anything-but": ["Chrome"]}],
  "Location": [{"prefix": "US"}],
  "Age": [{"exists": true}]
}
```

以下範例會比對 Location 是否為 Japan (日本)，另外 Browser 是否為 Safari，或者 Device 是否為 Tablet (平板電腦)。

```
{
 "Location": ["Japan"],
 "$or": [
   {"Browser": ["Safari"]},
   {"Device": ["Tablet"]}
 ]
}
```

建立客群

建立客群後，您可以在任何專案的任何發布會或實驗中使用該客群。

建立客群

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，顯而易見。

3. 選擇 Segments (客群) 索引標籤。
4. 選擇 Create segment (建立客群)。
5. 對於 Segment name (客群名稱)，請輸入用於識別此客群的名稱。

   您可以選擇新增描述。
6. 對於 Segment pattern (客群模式)，請輸入用來定義規則模式的 JSON 區塊。如需篩選規則模式語法的詳細資訊，請參閱 客群規則模式語法。

建立啟動

若要公開新功能或變更至指定百分比的使用者，請建立啟動。然後，您可以在向所有使用者推出此功能之前，監控關鍵指標 (例如頁面載入時間和轉換)。

您必須先建立專案，才能新增啟動。如需詳細資訊，請參閱 建立新專案。

新增啟動時，您可以使用已建立的功能，或在建立啟動時建立新功能。

將啟動新增至專案

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，顯而易見。
3. 選取專案名稱旁的按鈕，然後選擇 Project actions (專案動作)、Create launch (建立啟動)。
4. 對於 Launch name (啟動名稱)，請輸入用於識別此專案中此功能的名稱。

   您可以選擇新增描述。
5. 選擇 Select from existing features (從現有功能中選取) 或者 Add new feature (新增功能)。

   如果您正在使用現有功能，請在 Feature name (功能名稱) 下選取。

   如果選擇 Add new feature (新增功能)，則請執行下列動作：

   a. 對於 Feature name (功能名稱)，請輸入用於識別此專案中此功能的名稱。您可以選擇新增描述。
   b. 對於 Feature variations (功能變化)，對於 Variation type (變化類型)，請選擇 Boolean (布林型)、Long (長整數)、Double (雙字)，或 String (字串)。如需詳細資訊，請參閱 變化類型。
   c. 為您的功能新增最多五種變化。每個變化的 Value (值) 必須適用於您選取的 Variation type (變化類型)。

指定其中一個變化作為預設值。這是比較其他變化的基準線，應該是目前正在為使用者提供服務的變化。如果您停止實驗，則此預設變化會提供給所有使用者。

d. 選擇 Sample code (範本程式碼)。程式碼範例會顯示您需要新增至應用程式以設定變化和為應用程式指派使用者工作階段的項目。您可以在 Java 和 Python 之間 JavaScript進行選擇作為代碼。

您現在不需要將程式碼新增至您的應用程式，但必須在開始啟動之前這麼做。

如需詳細資訊，請參閱 將程式碼新增至應用程式。

6. 對於 Launch configuration (啟動組態) 中，選擇要立即啟動啟動，還是排定稍後啟動。
7. (選用) 若要為您已定義的對象客群指定不同的流量分割，而非您將用於一般對象的流量分割，請選擇 Add Segment Overrides (新增客群覆寫)。

在 Segment Overrides (客群覆寫) 中，選取客群並定義要用於該客群的流量分割。

您可以選擇是否要定義更多客群來定義流量分割，方法是選擇 Add Segment Overrides (新增客群覆寫)。一場發布會最多可以有六個客群覆寫。

如需詳細資訊，請參閱 使用客群功能聚焦您的受眾。

8. 對於 Traffic configuration (流量組態)，請選取流量百分比以指派給不符合客群覆寫的一般受眾的每個變體。您也可以選擇不為使用者提供變化服務。

Traffic summary (流量摘要) 會顯示您的整體流量可用於此次啟動。

9. 如果選擇將啟動排定為稍後開始，則您可以在啟動中新增多個步驟。每個步驟可以使用不同的百分比來提供變化服務。若要執行此操作，請選擇 Add another step (新增另一個步驟)，然後指定下一個步驟的排程和流量百分比。您最多可以在啟動中包含五個步驟。
10. 如果您想要在啟動期間利用指標來追蹤功能效能，請選擇 Metrics (指標)、Add metric (新增指標)。您可以使用 CloudWatch RUM 指標或自訂指標。

若要使用自訂指標，您可以在此處使用 Amazon EventBridge 規則建立指標。若要建立自訂指標，請執行下列動作：

- 選擇 Custom metrics (自訂指標)，然後輸入指標的名稱。
- 在 Metric rule (指標規則)，用於 Entity ID (實體 ID)，輸入識別實體的方式。這可以是執行導致指標值被記錄之動作的使用者或工作階段。例如，`userDetails.userID`。
- 對於 Value key (值索引鍵)，輸入要追蹤以產生指標的值。
- (可選) 輸入指標單位的名稱。此單位名稱僅供顯示用途，用於 Evidently 主控台中的圖形。

當您輸入這些欄位時，方塊會顯示如何編寫 EventBridge 規則程式碼以建立量度的範例。有關更多信息 EventBridge，請參閱什麼是 Amazon EventBridge？

若要使用 RUM 指標，您必須已為應用程式設定 RUM 應用程式監控。如需詳細資訊，請參閱 設定應用程式以使用 CloudWatch RUM。

> **Note**
>
> 如果您使用 RUM 指標，且應用程式監控未設定為 100% 的使用者工作階段取樣，則並非所有參與啟動的使用者工作階段都會將指標傳送至 Evidently。為確保啟動指標準確無誤，建議應用程式監控使用 100% 的使用者工作階段進行取樣。

11. (選擇性) 如果您為啟動建立至少一個量度，您可以將現有 CloudWatch 警示與此次啟動相關聯。若要這樣做，請選擇關聯 CloudWatch 鬧鐘。

    當您將警報與啟動相關聯時， CloudWatch 顯然必須在警報中添加標籤與項目名稱和啟動名稱。這是為了 CloudWatch 顯然可以在控制台的啟動信息中顯示正確的警報。

    若要確認 CloudWatch 顯然會新增這些標籤，請選擇「明顯允許」，以使用此啟動資源標記以下識別的警示資源。然後，選擇 Associate alarm (關聯警示)並輸入警示名稱。

    如需建立 CloudWatch 警示的資訊，請參閱 使用 Amazon CloudWatch 警報。

12. (選用) 若要新增標籤至此啟動，請選擇 Tags (標籤)、Add new tag (新增標籤)。

    之後，在 Key (索引鍵) 中，輸入標籤的名稱。您可以在 Value (值) 中為標籤新增選用值。

    若要新增另一個標籤，請再次選擇 Add new tag (新增標籤)。

    如需詳細資訊，請參閱標記 AWS 資源。

13. 選擇 Create launch (建立啟動)。

建立實驗

使用實驗來測試不同版本的功能或網站，並從真實的使用者工作階段收集資料。如此一來，您就可以根據證據和資料為您的應用程式作出選擇。

您必須先建立專案，才能新增實驗。如需詳細資訊，請參閱 建立新專案。

新增實驗時，您可以使用已建立的功能，或在建立實驗時建立新功能。

新增實驗至專案

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，顯而易見。
3. 選取專案名稱旁的按鈕，然後選擇 Project actions (專案動作)、Create experiment (建立實驗)。
4. 對於 Experiment name (實驗名稱)，請輸入用於識別此專案中此功能的名稱。

   您可以選擇新增描述。
5. 選擇 Select from existing features (從現有功能中選取) 或者 Add new feature (新增功能)。

   如果您正在使用現有功能，請在 Feature name (功能名稱) 下選取。

   如果選擇 Add new feature (新增功能)，則請執行下列動作：

   a. 對於 Feature name (功能名稱)，請輸入用於識別此專案中此功能的名稱。您也可以選擇輸入描述。

   b. 對於 Feature variations (功能變化)，對於 Variation type (變化類型)，請選擇 Boolean (布林型)、Long (長整數)、Double (雙字)，或 String (字串)。類型會定義哪種類型的值用於每個變化。如需詳細資訊，請參閱 變化類型。

   c. 為您的功能新增最多五種變化。每個變化的 Value (值) 必須適用於您選取的 Variation type (變化類型)。

      指定其中一個變化作為預設值。這是比較其他變化的基準線，應該是目前正在為使用者提供服務的變化。如果您停止使用此功能的實驗，則預設變化會依據先前在實驗中的使用者百分比提供。

   d. 選擇 Sample code (範本程式碼)。程式碼範例會顯示您需要新增至應用程式以設定變化和為應用程式指派使用者工作階段的項目。您可以在 Java 和 Python 之間 JavaScript進行選擇作為代碼。

      您現在不需要將程式碼新增至您的應用程式，但必須在開始實驗之前這麼做。如需詳細資訊，請參閱 將程式碼新增至應用程式。
6. 對於 Audience (受眾)，如果您希望此實驗僅套用至符合該客群的使用者，可決定是否選取您已建立的客群。如需客群的詳細資訊，請參閱 使用客群功能聚焦您的受眾。
7. 對於 Traffic split for the experiment (適用於實驗的流量分割)，指定將在實驗中使用其工作階段的已選取受眾的百分比。然後將流量配置為實驗所使用的不同變化。

如果啟動和實驗都在同一時間執行相同功能，受眾會先導向啟動。然後，針對啟動指定的流量百分比取自整體受眾。在此之後，您在此指定的百分比是用於實驗之剩餘受眾的百分比。之後的任何剩餘流量都會提供預設變化。

8. 對於 Metrics (指標)，選擇要用來評估實驗期間變化的指標。評估至少必須使用一項指標。

   a. 對於指標來源，請選擇要使用 CloudWatch RUM 量度還是自訂量度。

   b. 輸入指標的名稱。對於 Goal (目標)，如果您希望指標具有較高的值，以表示較佳的變化，則請選擇 Increase (增加)。如果您想要較低的指標值來表示較好的變化，則請選擇 Decrease (減少)。

   c. 如果您使用自訂指標，則可以在此處使用 Amazon EventBridge 規則建立指標。若要建立自訂指標，請執行下列動作：

      - 在 Metric rule (指標規則) 下，對於 Entity ID (實體 ID)，請輸入識別實體的方法，這可以是執行導致指標值被記錄之動作的使用者或工作階段。例如，`userDetails.userID`。
      - 對於 Value key (值索引鍵)，輸入要追蹤以產生指標的值。
      - (可選) 輸入指標單位的名稱。此單位名稱僅供顯示用途，用於 Evidently 主控台中的圖形。

      只有在您已設定 RUM 來監控此應用程式時，才能使用 RUM 指標。如需詳細資訊，請參閱 使用 CloudWatch 朗姆酒。

      > **Note**
      >
      > 如果您使用 RUM 指標，且應用程式監控未設定為 100% 的使用者工作階段取樣，則並非實驗中的所有使用者工作階段都會將指標傳送至 Evidently。為確保實驗指標準確無誤，建議應用程式監控使用 100% 的使用者工作階段進行取樣。

   d. (選用) 若要新增其他要評估的指標，請選擇 Add metric (新增指標)。您可以在實驗期間評估最多三個指標。

9. （可選）要創建用於此實驗的 CloudWatch 警報，請選擇CloudWatch警報。警示可以監控每個變化和預設變化之間的結果差異是否大於您指定的閾值。如果變化的效能比預設變化差，且差異大於閾值，則會進入警示狀態並通知您。

   在此處建立警示會為每個不是預設變化的變化建立一個警示。

   如果您建立警示，則請指定以下項目：

- 對於 Metric name (指標名稱)，選擇要用於警示的實驗指標。
- 對於 Alarm condition (警示條件) 選擇當變化指標值與預設變化指標值進行比較時，哪些條件會導致警示進入警示狀態。例如，選擇 Greater (大於) 或者 Greater/Equal (大於/等於)，如果表示變化的較大數字表示其執行不佳。例如，如果指標正在測量頁面載入時間，則這會適用。
- 輸入閾值的數字，這是造成警示進入 ALARM 狀態的效能百分比差異。
- 對於 Average over period (期間內的平均值)，選擇在比較之前會將每個變化的多少指標資料彙總在一起。

您可以再次選擇 Add new alarm (新增警示)，為實驗新增更多警示。

接下來，選擇 Set notifications for the alarm (設定警示的通知)，並選取或建立 Amazon Simple Notification Service 主題，以便將警示通知傳送至其中。如需詳細資訊，請參閱 設定 Amazon SNS 通知。

10. (選用) 若要新增標籤至此實驗，請選擇 Tags (標籤)、Add new tag (新增標籤)。

    之後，在 Key (索引鍵) 中，輸入標籤的名稱。您可以在 Value (值) 中為標籤新增選用值。

    若要新增另一個標籤，請再次選擇 Add new tag (新增標籤)。

    如需詳細資訊，請參閱標記 AWS 資源。
11. 選擇 Create experiment (建立實驗)。
12. 如果還沒有，則請將功能變體建置到您的應用程式中。
13. 選擇完成。在您啟動前，實驗不會開始。

完成下列程序中的步驟後，實驗會立即開始。

開始您已建立的實驗

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，顯而易見。
3. 選擇專案的名稱。
4. 選擇 Experiments (實驗) 索引標籤。
5. 選擇實驗名稱旁的按鈕，然後選擇 Actions (動作)、Start experiment (開始實驗)。
6. (選用) 若要檢視或修改建立實驗時所做的實驗設定，請選擇 Experiment setup (實驗設定)。
7. 選擇實驗結束的時間。

8. 選擇 Start experiment (開始實驗)。

   實驗會即刻開始。

管理功能、啟動和實驗

使用這些章節中的程序來管理您所建立的功能、啟動和實驗。

主題

- 查看功能的目前評估規則和受眾流量
- 修改啟動流量
- 修改啟動的後續步驟
- 修改實驗流量
- 停止啟動
- 停止實驗

查看功能的目前評估規則和受眾流量

您可以使用 CloudWatch Eviatic 控制台來查看該功能的評估規則如何在該功能的當前發布，實驗和變體之間分配受眾流量。

檢視功能的受眾流量

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，顯而易見。
3. 選擇包含功能的專案名稱。
4. 選擇 Features (功能) 索引標籤。
5. 選擇功能的名稱。

   在 Evaluation rules (評估規則) 索引標籤上，您可以查看功能之受眾流量的流程，如下所示：

   - 首先，評估複寫。這些指定一律會向某些使用者提供特定變化。指派了覆寫的使用者工作階段不會影響啟動或實驗指標。
   - 接下來，剩餘的流量可用於進行中的啟動 (如果有的話)。如果正在進行啟動，Launches (啟動) 區段的資料表會顯示啟動名稱和啟動流量在功能變化之間的分割。在 Launches (啟動) 區段的右

側，Traffic (流量) 指示器會顯示配置給此啟動的可用受眾數量 (覆寫之後)。未配置給啟動流程的其餘部分流量會流到實驗 (如果有的話)，然後是預設變化。

- 接下來，剩餘的流量可用於進行中的實驗 (如果有的話)。如果有實驗正在進行中，Experiments (實驗) 區段中的資料表會顯示實驗名稱和進度。在 Experiments (實驗) 區段的右側，Traffic (流量) 指示器會顯示配置給此啟動的可用受眾數量 (覆寫之後)。未配置給啟動或實驗的其餘部分流量會提供功能的預設變化。

修改啟動流量

您可以隨時修改啟動的流量配置，包括在啟動進行中修改。

如果您同時有進行中的啟動和進行中的相同功能實驗，則功能流量的任何變更都會導致實驗流量的變更。這是因為實驗可用的受眾是尚未配置給啟動之總受眾的一部分。增加啟動流量會減少實驗可用的受眾，降低啟動流量或結束啟動會增加實驗可用的受眾。

修改啟動的流量配置

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，顯而易見。
3. 選擇包含啟動的專案名稱。
4. 選擇 Launches (啟動) 索引標籤。
5. 選擇啟動的名稱。

   選擇 Modify launch traffic (修改啟動流量)。
6. 對於 Serve (提供服務)，請選取要指派給每個變化的新流量百分比。您也可以選擇不為使用者提供變化服務。當變更這些值時，您可以在 Traffic summary (流量摘要) 下查看整體功能流量的已更新效果。

   Traffic summary (流量摘要) 會顯示有多少可用於此次啟動的整體流量，以及有多少該可用流量分配給此次啟動。
7. 選擇 Modify (修改)。

修改啟動的後續步驟

您可以修改尚未發生的啟動步驟組態，也可以為啟動新增更多的步驟。

修改啟動的步驟

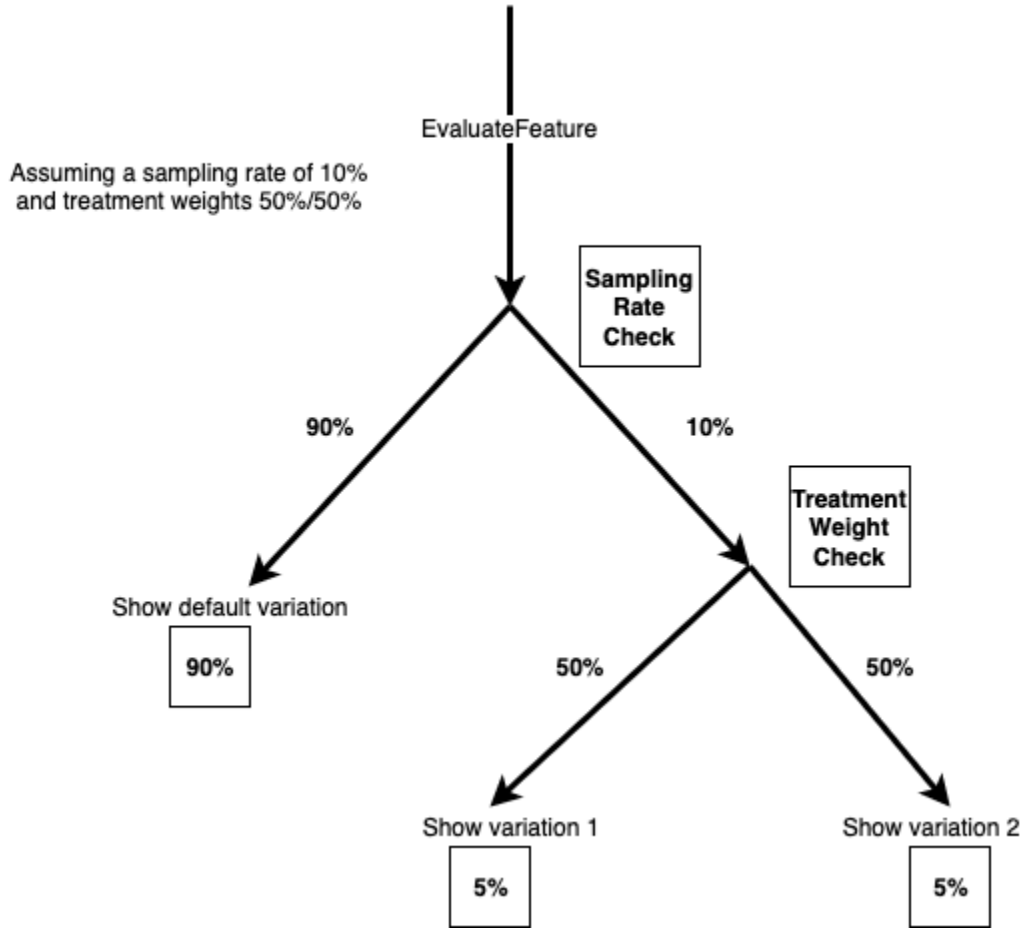
1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，顯而易見。
3. 選擇包含啟動的專案名稱。
4. 選擇 Launches (啟動) 索引標籤。
5. 選擇啟動的名稱。

   選擇 Modify launch traffic (修改啟動流量)。
6. 選擇 Schedule launch (排程啟動)。
7. 對於尚未開始的任何步驟，您可以修改實驗中要使用的可用受眾百分比。您也可以修改其流量在變化之間的配置方式。

   您可以在啟動中新增更多步驟，方法是選擇 Add another step (新增另一個步驟)。啟動最多可以有五個步驟。
8. 選擇 Modify (修改)。

修改實驗流量

您可以隨時修改實驗的取樣率，包括在實驗進行中修改。但是，在實驗執行後，即無法更新處理權重。因此，您可以在實驗執行後變更接觸到實驗的總流量，但無法變更每個處理方式的相對配置。如果修改進行中實驗的流量，則建議您只增加流量配置，以免造成偏差。



修改實驗的流量配置

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Application monitoring (應用程式監控)、Evidently。
3. 選擇包含啟動的專案名稱。
4. 選擇 Experiments (實驗) 索引標籤。
5. 選擇啟動的名稱。
6. 選擇 Modify experiment traffic (修改實驗流量)。
7. 輸入百分比或使用滑桿來指定要配置給此實驗的可用流量。可用的流量是總受眾減去配置給目前啟動的流量 (如果有的話)。未配置給啟動或實驗的流量會提供預設變化。
8. 選擇 Modify (修改)。

停止啟動

如果停止進行中的啟動，則您將無法繼續執行或重新啟動。此外，不會將其評估為流量配置的規則，而且配置給啟動的流量將改為可用於功能的實驗 (如果有的話)。否則，在啟動停止後，所有流量都會提供預設變化。

永久停止啟動

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，顯而易見。
3. 選擇包含啟動的專案名稱。
4. 選擇 Launch (啟動) 索引標籤。
5. 選擇啟動名稱左側的按鈕。
6. 選擇 Actions (動作)、Cancel launch (取消啟動) 或者 Actions (動作)、Mark as complete (標記為完成)。

停止實驗

如果停止進行中的實驗，則您將無法繼續執行或重新啟動。先前在實驗中使用的流量部分會提供預設變化。

實驗未手動停止並超過其結束日期時，流量不會變更。分配給實驗的流量部分仍然會進入實驗。若要停止這種情況，並讓實驗流量改為提供預設變化，請將實驗標記為完成。

停止實驗時，您可以選擇取消或將其標記為完成。如果取消，它會在實驗清單中顯示為 Cancelled (已取消)。如果您選擇將其標記為完成，則會顯示為 Completed (已完成)。

永久停止實驗

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，顯而易見。
3. 選擇其中包含實驗的專案名稱。
4. 選擇 Experiments (實驗) 索引標籤。
5. 選擇實驗名稱左側的按鈕。
6. 選擇 Actions (動作)、Cancel experiment (取消實驗) 或者 Actions (動作)、Mark as complete (標記為完成)。

將程式碼新增至應用程式

若要使用「 CloudWatch 顯而易見」，您可以在應用程式中新增程式碼，以便為每個使用者工作階段指派變體，並將指標傳送至 Edition。您可以使用「 CloudWatch 顯而易見」`EvaluateFeature` 操作將變體指派給使用者工作階段，並使用該`PutProjectEvents`作業將事件傳送至「顯而易見」，以用來計算啟動或實驗的指標。

當您建立變體或自訂指標時， CloudWatch Edition 主控台會提供您需要新增的程式碼範例。

如需範 end-to-end 例，請參閱教學：使用 Evidently 範例應用程式進行 A/B 測試。

使用 EvaluateFeature

在啟動或實驗中使用特徵變化時，應用程式會使用該 EvaluateFeature作業為每個使用者工作階段指派一個變化。將變化指派給使用者是評估事件。當您呼叫此操作時，您會傳入下列內容：

- 功能名稱 – 必要。Evidently 根據啟動或實驗的功能評估規則處理評估，並為實體選取變化。
- entityId – 必要。表示唯一使用者。
- evaluationContext – 選用。JSON 物件，代表有關使用者的其他資訊。如果您已建立客群，Evidently 會在功能評估期間使用這個值來將使用者與您受眾的一個客群進行比對。如需詳細資訊，請參閱 使用客群功能聚焦您的受眾。

  以下是您可以傳送給 Evidently 的一個範例 `evaluationContext` 值。

```
{
    "Browser": "Chrome",
    "Location": {
        "Country": "United States",
        "Zipcode": 98007
    }
}
```

黏性評估

CloudWatch 顯然使用「粘性」評估。`entityId` 的單一組態、功能、功能組態和 `evaluationContext` 始終會接收相同的變化指派。此變體指派變更僅發生在將實體新增至撥號的覆寫或實驗流量時。

功能組態包含以下內容：

- 功能變化
- 此功能目前執行中的實驗的變化組態 (指派給每個變化的百分比) (如果有)。
- 此功能目前執行中的啟動的變化組態 (如果有)。變化組態包括已定義區段覆寫 (如果有)。

若實驗的流量配置增加，先前指派給實驗組的任何 entityId 均將繼續接受相同的處理。根據為實驗指定的變化組態，先前指派給對照組的任何 entityId 可能會指派給實驗組。

若實驗的流量配置減少，則會有一個 entityId 可能會從實驗組轉為對照組，但不會進入不同的實驗組。

使用 PutProjectEvents

若要為「明顯」編寫自訂指標，請使用該 PutProjectEvents作業。以下是簡單的酬載範例。

```
{
    "events": [
        {
            "timestamp": {{$timestamp}},
            "type": "aws.evidently.custom",
            "data": "{\"details\": {\"pageLoadTime\": 800.0}, \"userDetails\":
 {\"userId\": \"test-user\"}}"
        }
    ]
}
```

entityIdKey 可以只是 entityId 或者您可以將其重命名為其他任何內容，例如 userId。在實際的事件中，entityId 可以是使用者名稱、工作階段 ID 等。

```
"metricDefinition":{
    "name": "noFilter",
    "entityIdKey": "userDetails.userId", //should be consistent with jsonValue in
 events "data" fields
    "valueKey": "details.pageLoadTime"
},
```

若要確保事件與正確的啟動或實驗關聯，在您呼叫 entityId 和 EvaluateFeature 時，必須傳遞相同的 PutProjectEvents。一定要在通話PutProjectEvents後EvaluateFeature打電話，否則數據被丟棄，不會被 CloudWatch 顯然使用。

PutProjectEvents 操作不需要將功能名稱作為輸入參數。這樣一來，您可以在多個實驗中使用單一事件。例如，假設您呼叫 EvaluateFeature，並將 entityId 設定為 userDetails.userId。如果您執行兩個或多個實驗，則可以讓該使用者工作階段中的單一事件針對每個實驗發出指標。若要執行此操作，使用相同的 entityId 針對每個實驗呼叫 PutProjectEvents 一次。

Timing (計時)

在應用程式呼叫 EvaluateFeature 之後，在一小時期間內，會根據該評估對 PutProjectEvents 的指標事件進行歸類。如果在一小時期間後再發生任何事件，則不會歸類這些事件。

但是，在初始呼叫的一小時期間內，如果相同的 entityId 用於新的 EvaluateFeature 呼叫，現在則會改用稍後的 EvaluateFeature 結果，並重新啟動一小時計時器。只有在某些情況下才能發生這種情況，例如，在兩個指派之間撥號的實驗流量，如前面黏性評估章節所述。

如需範 end-to-end 例，請參閱教學：使用 Evidently 範例應用程式進行 A/B 測試。

專案資料儲存

Evidently 會收集兩種事件：

- 評估事件與指派給使用者工作階段的功能變化有關。Evidently 使用這些事件來產生指標和其他實驗及啟動資料，您可以在 Evidently 主控台中檢視這些資料。

  您也可以選擇將這些評估事件存放在亞馬遜 CloudWatch 日誌或 Amazon S3 中。
- 自訂事件用於從使用者動作 (例如點選和結帳) 產生指標。Evidently 沒有為您提供存放自訂事件的方法。如果您想儲存自訂事件，則必須修改您的應用程式碼，將程式碼傳送至 Evidently 之外的儲存選項。

評估事件日誌的格式

如果您選擇將評估事件存放在 CloudWatch Logs 或 Amazon S3 中，每個評估事件都會以下列格式存放為日誌事件：

```
{
    "event_timestamp": 1642624900215,
    "event_type": "evaluation",
    "version": "1.0.0",
    "project_arn": "arn:aws:evidently:us-east-1:123456789012:project/petfood",
    "feature": "petfood-upsell-text",
    "variation": "Variation1",
    "entity_id": "7",
```

```
    "entity_attributes": {},
    "evaluation_type": "EXPERIMENT_RULE_MATCH",
    "treatment": "Variation1",
    "experiment": "petfood-experiment-2"
}
```

以下為上述評估事件格式的相關詳細資料：

- 時間戳記以 UNIX 時間為單位，毫秒為單位
- 變化是指派給此使用者工作階段功能變化的名稱。
- 實體 ID 是一個字串。
- 實體屬性是由用戶端傳送的任意值雜湊。例如，若 `entityId` 映射為藍色或綠色，則您可選擇從關聯和資料倉儲的角度傳送 userID、工作階段資料，或任何您想要的其他內容。

於 Simple Storage Service (Amazon S3) 中評估事件儲存的 IAM 政策和加密

若您選擇使用 Simple Storage Service (Amazon S3) 來存放評估事件，則必須新增如下所示的 IAM 政策，可讓 Evidently 將日誌發佈至 Simple Storage Service (Amazon S3) 儲存貯體。這是因為 Simple Storage Service (Amazon S3) 儲存貯體及其所包含的物件都是私有的，而且依預設，它們不允許存取其他服務。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AWSLogDeliveryWrite",
            "Effect": "Allow",
            "Principal": {"Service": "delivery.logs.amazonaws.com"},
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::bucket_name/optional_folder/AWSLogs/account_id/
*",
            "Condition": {"StringEquals": {"s3:x-amz-acl": "bucket-owner-full-
control"}}
        },
        {
            "Sid": "AWSLogDeliveryCheck",
            "Effect": "Allow",
            "Principal": {"Service": "delivery.logs.amazonaws.com"},
            "Action": ["s3:GetBucketAcl", "s3:ListBucket"],
            "Resource": "arn:aws:s3:::bucket_name"
```

```
        }
    ]
}
```

如果將 Evidently 資料存放在 Simple Storage Service (Amazon S3) 中，則您也可以選擇使用具有 AWS Key Management Service 金鑰的伺服器端加密 (SSE-KMS) 對其進行加密。如需詳細資訊，請參閱使用伺服器端加密保護資料。

如果您使用的是客戶受管金鑰 AWS KMS，則必須將下列項目新增至金鑰的 IAM 政策。這允許 Evidently 寫入儲存貯體。

```
{
    "Sid": "AllowEvidentlyToUseCustomerManagedKey",
    "Effect": "Allow",
    "Principal": {
        "Service": [
            "delivery.logs.amazonaws.com"
        ]
    },
   "Action": [
       "kms:Encrypt",
       "kms:Decrypt",
       "kms:ReEncrypt*",
       "kms:GenerateDataKey*",
       "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

Evidently 如何計算結果

您可以使用 Amazon CloudWatch 顯而易見的 A/B 測試作為資料驅動決策的工具。在 A/B 測試中，使用者被隨機指派給對照組 (也稱為預設變化) 或其中一個實驗組 (也稱為測試變化)。例如，對照組中的使用者可能會以實驗開始前的相同方式來體驗網站、服務或應用程式。同時，實驗組中的使用者可能會遇到變化。

CloudWatch 顯然，在實驗中支持多達五種不同的變化。Evidently 隨機將流量指派給這些變化。如此一來，您就可以追蹤每個群組的業務指標 (例如收入) 和效能指標 (例如延遲)。Evidently 會執行以下動作：

- 將實驗組與對照組進行比較。(例如，比較收入是隨著新的結帳程序增加還是減少。)

- 指出在實驗組和對照組之間觀察到的差異是否顯著。為此，Evidently 提供了兩種方法：頻率論顯著程度和貝葉斯概率。

為什麼要使用頻率論和貝葉斯方法？

考慮實驗組與對照組相比沒有效果的情況，或者實驗組與對照組相同的情況 (A/A 測試)。您仍然會在資料中觀察到實驗組和對照組之間的微小差異。這是因為測試參與者由有限的使用者樣本組成，佔了網站、服務或應用程式所有使用者的一小部分。頻率論顯著程度和貝葉斯概率能夠讓人們了解，觀察到的差異是顯著還是由於概率所致。

Evidently 會考慮以下幾點，以判斷觀察到的差異是否顯著：

- 差異有多大
- 有多少樣本參與了測試
- 資料的分佈方式

Evidently 中的頻率論分析

Evidently 使用循序測試，這避免了一般的偷看問題，也就是頻率論統計的常見陷阱。偷看是指檢查進行中的 A/B 測試結果，以便停止測試，並根據所觀察結果進行決策的做法。如需有關循序測試的詳細資訊，請參閱 Howard 等人編撰的 Time-uniform, nonparametric, nonasymptotic confidence sequences (時間均勻、無母數、非漸近的信賴序列) (Ann. Statist. 49 (2) 1055 - 1080, 2021)。

因為 Evidently 的結果隨時都有效 (隨時有效的結果)，您可以在實驗過程中偷看結果，並仍然得出合理的結論。這可以降低一些實驗成本，因為如果結果已相當顯著，您可以在排定的時間前停止實驗。

Evidently 會針對目標指標中測試變化和預設變化之間的差異，產生隨時有效的顯著程度以及隨時有效的 95% 信賴區間。實驗結果中的 Result (結果) 一欄指示測試變化效能，可為下列其中之一：

- Inconclusive (不確定)：顯著程度低於 95%
- Better (較佳)：顯著程度為 95% 或更高，且下列其中一項為真：
  - 95% 信賴區間的下限大於零，且指標應增加
  - 95% 信賴區間的上限小於零，且指標應減少
- Worse (更糟)：顯著程度為 95% 或更高，且下列其中一項為真：
  - 95% 信賴區間的上限大於零，且指標應增加
  - 95% 信賴區間的下限小於零，且指標應減少
- Best (最佳)：除了預設變化之外，該實驗還具有兩個以上測試變化，並且滿足以下條件：

- 該變化符合 Better (較佳) 指定的資格
- 下列其中一個條件為真：
  - 95% 信賴區間下限高於所有其他變化的 95% 信賴區間上限，且指標應增加
  - 95% 信賴區間的上限低於所有其他變化的 95% 信賴區間的下限，且指標應減少

Evidently 中的貝葉斯分析

您可以使用貝葉斯分析，計算測試變化中的平均值大於或小於預設變化中平均值的機率。Evidently 透過使用共軛先驗，對目標指標的平均值執行貝葉斯推論。使用共軛先驗，Evidently 可以更高效地推斷貝葉斯分析所需的後驗分佈。

Evidently 會等待直到實驗的結束日期來計算貝葉斯分析的結果。結果頁面會顯示下列資訊：

- 增加的機率：測試變化中指標平均值比預設變化中的平均值大至少 3% 的機率
- 減少的機率：測試變化中指標平均值比預設變化中的平均值小至少 3% 的概率
- 未變更的機率：測試變化中指標平均值在預設變化中平均值的 ±3% 以內的機率

Result (結果) 一欄指出變化的效能，而且可為下列其中之一：

- Better (較佳)：增加的機率至少為 90%，且指標應增加，或者減少的機率至少為 90%，且指標應減少
- Worse (更糟)：減少的機率至少為 90%，且指標應增加，或者增加的機率至少為 90%，且指標應減少

在儀表板中檢視啟動結果

您可以在實驗進行中及完成後，看到實驗的進度和指標結果。

查看啟動的進度和結果

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，顯而易見。
3. 選擇包含啟動的專案名稱。
4. 選擇 Launch (啟動) 索引標籤。
5. 選擇啟動的名稱。

6. 若要查看每個步驟的啟動步驟和流量配置，請選擇 Launch (啟動) 索引標籤。
7. 若要查看隨時間變化指派給每個變化的使用者工作階段數目，以及檢視啟動中每個變化的效能指標，請選擇 Monitoring (監控) 索引標籤。

   此檢視也會顯示啟動期間是否有任何啟動警示進入 ALARM 狀態。
8. 若要查看此次啟動的變化、指標、警示和標籤，請選擇 Configuration (組態) 索引標籤。

在儀表板中檢視實驗結果

您可以在實驗進行中及完成後，看到實驗的統計結果。實驗開始後 63 天內可獲得實驗結果。之後，由於 CloudWatch 資料保留原則，無法使用它們。

在每個變化至少有 100 個事件前，不會顯示任何統計結果。

Evidently 會在實驗結束時執行額外的離線 p 值分析。某些情況下，當實驗期間使用的任何時間 p 值未發現統計顯著性時，離線 p 值分析可以偵測到統計顯著性。

有關如何 CloudWatch 明顯地計算實驗結果的更多信息，請參閱Evidently 如何計算結果。

查看實驗結果

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，顯而易見。
3. 選擇其中包含實驗的專案名稱。
4. 選擇 Experiments (實驗) 索引標籤。
5. 選擇實驗的名稱，然後選擇 Results (結果) 索引標籤。
6. 依 Variation performance (變化效能)，有一個控制項，您可以在其中選取要顯示的實驗統計數字。如果選取一個以上的統計數字，則 Evidently 會顯示每個統計數字的圖形和資料表。

   每個圖形和資料表都會顯示到目前為止的實驗結果。

   每個圖形都可以顯示以下結果。您可以使用圖形右側的控制項來決定要顯示下列哪些項目：

   - 每個變化記錄的使用者工作階段事件數目。
   - 在圖形頂端針對每個變化選取的指標平均值。
   - 實驗的統計意義。這會比較圖形頂端所選指標之預設變化與每個其他變化之間的差異。
   - 所選指標差異的 95% 上下信賴繫結 (每個變化與預設變化之間)。

資料表會針對每個變化顯示一列。對於每個不是預設的變化，Evidently 會顯示其是否已經收到足夠的資料來聲明結果具有統計學意義。它還會顯示變化統計值的改進是否已達到 95% 的信賴等級。

最後，在 Result (結果) 資料欄中，Evidently 會根據此統計數字或結果是否不確定，來提供關於哪一種變化效能最佳的建議。

如何 CloudWatch 明顯地收集和存儲數據

Amazon CloudWatch 明顯地收集和存放與專案組態相關的資料，以便客戶可以執行實驗和啟動。資料包含以下內容：

- 關於專案、功能、啟動和實驗的中繼資料
- 指標事件
- 評估資料

資源中繼資料存放在 Amazon DynamoDB 中。默認情況下，靜態數據被加密，使用 AWS 擁有的金鑰. 這些金鑰是 AWS 服務 擁有和管理以供多個使用的 AWS KMS 金鑰的集合 AWS 帳戶。客戶無法檢視、管理或稽核這些金鑰的使用情況。客戶也不需要採取動作或變更程式，即可保護加密其資料的金鑰。

如需詳細資訊，請參閱AWS 擁有的金鑰開 AWS Key Management Service 發人員指南中的。

Evidently 指標事件和評估事件會直接傳送至客戶擁有的位置。

傳輸中的資料會自動使用 HTTPS 進行加密。此資料將傳送至客戶擁有的地點。

您也可以選擇將評估事件存放在 Amazon 簡單儲存服務或 Amazon CloudWatch 日誌中。如需有關如何在這些服務中保護資料的詳細資訊，請參閱啟用 Amazon S3 預設儲存貯體加密和使用 AWS KMS加密 CloudWatch 日誌中的日誌資料。

擷取資料

您可以使用 CloudWatch 顯而易見的 API 檢索數據。若要擷取專案資料，請使用GetProject或ListProjects。

若要擷取特徵資料，請使用GetFeature或ListFeatures。

若要擷取啟動資料，請使用GetLaunch或ListLaunches。

若要擷取實驗資料 GetExperiment，請使用ListExperiments、或GetExperimentResults。

修改和刪除資料

您可以使用 CloudWatch 顯而易見的 API 修改和刪除您的數據。對於專案資料，請使用UpdateProject或DeleteProject。

對於圖徵資料，請使用UpdateFeature或DeleteFeature。

對於啟動資料，請使用UpdateLaunch或DeleteLaunch。

對於實驗資料，請使用UpdateExperiment或DeleteExperiment。

使用 Evidently 的服務連結角色

CloudWatch 顯然使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 Evidently 的一種特殊 IAM 角色類型。服務連結的角色由 Elevised 預先定義，並包含服務代表您呼叫其他 AWS 服務所需的所有權限。

服務連結角色可讓 Evidently 的設定更為簡單，因為您不必手動新增必要的許可。Evidently 定義其服務連結角色的許可，除非另有定義，否則僅有 Evidently 可以擔任其角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

您必須先刪除角色的相關資源，才能刪除服務連結角色。如此可保護 Evidently 資源，因為您就不會不小心移除資源的存取許可。

如需有關支援服務連結角色之其他服務的資訊，請參閱可搭配 IAM 運作的AWS 服務，並尋找 Service-linked roles (服務連結角色) 資料行中顯示為 Yes (是) 的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

Evidently 的服務連結角色許可

顯然使用名為的服務鏈接角色 AWSServiceRoleForCloudWatchEvidently- CloudWatch 顯然允許代表客戶管理關聯的 AWS 資源。

服 AWSServiceRoleForCloudWatchEvidently 務連結角色會信任下列服務擔任該角色：

- `CloudWatch Evidently`

名為的角色權限策略 AmazonCloudWatchEvidentlyServiceRolePolicy 允許 Evistosed 對指定的資源完成以下操作：

- 動作：Evidently 複雜型用戶端上的 `appconfig:StartDeployment`、`appconfig:StopDeployment`、`appconfig:ListDeployments` 以及 `appconfig:TagResource`。

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的服務連結角色許可。

為 Evidently 建立服務連結角色

您不需要手動建立一個服務連結角色。當您開始在 AWS Management Console、或 AWS API 中使用明顯濃密的用戶端時 AWS CLI，顯然會為您建立服務連結的角色。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您開始使用 Evidently 複雜型用戶端時，Evidently 會再次為您建立服務連結角色。

編輯 Evidently 的服務連結角色

顯然不允許您編輯 AWSServiceRoleForCloudWatchEvidently 服務鏈接的角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱 IAM 使用者指南中的編輯服務連結角色。

刪除 Evidently 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，在手動刪除服務連結角色之前，您必須先清除資源。您必須刪除所有正在使用複雜型用戶端的 Evidently 專案。

> **Note**
>
> 如果 Evidently 服務在您嘗試刪除資源時正在使用該角色，刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

刪除明顯使用的資源 AWSServiceRoleForCloudWatchEvidently

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Application monitoring (應用程式監控)、Evidently。
3. 在專案清單中，選取使用複雜型用戶端的專案旁的核取方塊。
4. 選擇 Project actions (專案動作)、Delete project (刪除專案)。

使用 IAM 手動刪除服務連結角色

使用 IAM 主控台或 AWS API 刪除 AWSServiceRoleForCloudWatchEvidently服務連結角色。 AWS CLI如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

Evidently 服務連結角色的支援區域

Evidently 支援在所有提供服務的區域中使用服務連結角色。如需詳細資訊，請參閱 AWS 區域與端點。

CloudWatch 顯然配額

CloudWatch 顯然有以下配額。

資源	預設配額
專案	每個帳戶每個區域 50 個 您可以要求增加配額。
客群	每個帳戶每個區域 500 個 您可以要求增加配額。
每個專案的配額	• 共 100 個功能 • 共 500 個啟動 • 50 個執行中的啟動 • 共 500 個實驗 • 50 個執行中的實驗 對於所有這些配額，您可請求提高配額。
API 配額 (所有配額均按區域計算)	• PutProjectEvents：美國東部 (維吉尼亞北部)、美國西部 (奧勒岡) 和歐洲 (愛爾蘭) 的每秒 1000 筆交易 (TPS)。其他所有區域中為 200 TPS。 • EvaluateFeature：美國東部 (維吉尼亞北部)、美國西部 (奧勒岡) 和歐洲 (愛爾蘭) 為 1000 TPS。其他所有區域中為 200 TPS。

資源	預設配額
	• BatchEvaluateFeature：50 租息計劃 • 建立、讀取、更新、刪除 (CRUD) API：跨所有 CRUD API 組合的 10 TPS 對於所有這些配額，您可請求提高配額。

教學：使用 Evidently 範例應用程式進行 A/B 測試

本節提供了如何使用 Amazon 進行 A/B 測 CloudWatch 試的教程。本教學將介紹 Evidently 範例應用程式，這是一個簡單的回應應用程式。範例應用程式將設定為顯示或不顯示 `showDiscount` 功能。若是對使用者顯示功能，在購物網站上顯示的價格會以 20% 的折扣顯示。

除了對某些使用者顯示折扣，對其他使用者則不顯示，在本教學中，您可以設定 Evidently 從這兩個變體中收集頁面載入時間指標。

> **Warning**
>
> 此案例需要具有程式設計存取權限和長期登入資料的 IAM 使用者，這會帶來安全風險。為了減輕此風險，我們建議您僅向這些使用者提供執行工作所需的權限，並在不再需要這些使用者時移除這些使用者。如有必要，可更新存取金鑰。如需詳細資訊，請參閱 IAM 使用者指南中的更新存取金鑰。

步驟 1：下載範例應用程式

首先下載 Evidently 範例應用程式。

下載範例應用程式

1. 從以下 Simple Storage Service (Amazon S3) 儲存貯體下載範例應用程式：

   ```
   https://evidently-sample-application.s3.us-west-2.amazonaws.com/evidently-sample-shopping-app.zip
   ```

2. 解壓縮套件。

步驟 2：新增 Evidently 端點並設定憑證

接著，將 Evidently 的區域和端點新增至範例應用程式套件中 `src` 目錄的 `config.js` 檔案，如以下範例所示：

```
evidently: {
    REGION: "us-west-2",
    ENDPOINT: "https://evidently.us-west-2.amazonaws.com (https://evidently.us-
west-2.amazonaws.com/)",
},
```

您還必須確保應用程序有權調用「 CloudWatch 顯而易見」。

授予範例應用程式呼叫 Evidently 的許可

1. 聯合到您的 AWS 帳戶。
2. 建立 IAM 使用者並將AmazonCloudWatchEvidentlyFullAccess政策附加到此使用者。
3. 請記下 IAM 使用者的存取金鑰 ID 和私密存取金鑰，因為下一個步驟您會需要它們。
4. 在之前於本節中修改的同一個 `config.js` 檔案中，輸入存取金鑰 ID 和機密存取金鑰的值，如以下範例所示：

   ```
   credential: {
      accessKeyId: "Access key ID",
      secretAccessKey: "Secret key"
   }
   ```

   > **Important**
   >
   > 我們使用此步驟讓範例應用程式儘可能簡單，以便您嘗試。不建議您將 IAM 使用者憑證置於實際的生產應用程式。建議您改為使用 Amazon Cognito 進行身分驗證。如需詳細資訊，請參閱將 Amazon Cognito 與 Web 和行動應用程式整合。

步驟 3：設定功能評估的程式碼

當您使用「 CloudWatch 顯而易見」來評估特徵時，您必須使用此EvaluateFeature作業為每個使用者作業階段隨機選取特徵變化。此操作會根據您在實驗中指定的百分比，將使用者工作階段指派給功能的每個變化。

設定書店示範應用程式的功能評估程式碼

1. 將用戶端建置器新增至 src/App.jsx 檔案，以便範例應用程式呼叫 Evidently。

```
import Evidently from 'aws-sdk/clients/evidently';
import config from './config';

const defaultClientBuilder = (
    endpoint,
    region,
) => {
    const credentials = {
        accessKeyId: config.credential.accessKeyId,
        secretAccessKey: config.credential.secretAccessKey
    }
    return new Evidently({
        endpoint,
        region,
        credentials,
    });
};
```

2. 將以下項目新增至 const App 程式碼區段以啟動用戶端。

```
  if (client == null) {
      client = defaultClientBuilder(
        config.evidently.ENDPOINT,
        config.evidently.REGION,
      );
```

3. 透過新增以下程式碼建構 evaluateFeatureRequest。此程式碼會預先填寫我們在本教學後面推薦的專案名稱和功能名稱。只要您還在 Evidently 主控台中指定這些專案和功能名稱，則可以替換自己的專案名稱和功能名稱。

```
const evaluateFeatureRequest = {
              entityId: id,
              // Input Your feature name
              feature: 'showDiscount',
              // Input Your project name'
              project: 'EvidentlySampleApp',
            };
```

4. 新增可呼叫 Evidently 進行進行功能評估的程式碼。傳送請求時，Evidently 會隨機指派使用者工作階段是否查看 showDiscount 功能。

```
client.evaluateFeature(evaluateFeatureRequest).promise().then(res => {
    if(res.value?.boolValue !== undefined) {
        setShowDiscount(res.value.boolValue);
    }
    getPageLoadTime()
 })
```

步驟 4：設定實驗指標的程式碼

對於自訂指標，請使用 Evidently 的 PutProjectEvents API 將指標結果傳送至 Evidently。下方範例會介紹如何設定自訂指標，以及如何將實驗資料傳送到 Evidently。

新增以下函數來計算頁面載入時間並使用 PutProjectEvents 將指標值傳送到 Evidently。將下面的函數新增至 Home.tsx，並在 EvaluateFeature API 內呼叫此函數：

```
const getPageLoadTime = () => {
    const timeSpent = (new Date().getTime() - startTime.getTime()) * 1.000001;
    const pageLoadTimeData = `{
      "details": {
        "pageLoadTime": ${timeSpent}
      },
      "UserDetails": { "userId": "${id}", "sessionId": "${id}"}
    }`;
    const putProjectEventsRequest = {
      project: 'EvidentlySampleApp',
      events: [
        {
          timestamp: new Date(),
          type: 'aws.evidently.custom',
          data: JSON.parse(pageLoadTimeData)
        },
      ],
    };
    client.putProjectEvents(putProjectEventsRequest).promise();
  }
```

以下是您下載檔案時完成所有編輯後的 App.js 檔案狀態。

```
import React, { useEffect, useState } from "react";
import { BrowserRouter as Router, Switch } from "react-router-dom";
import AuthProvider from "contexts/auth";
import CommonProvider from "contexts/common";
import ProductsProvider from "contexts/products";
import CartProvider from "contexts/cart";
import CheckoutProvider from "contexts/checkout";
import RouteWrapper from "layouts/RouteWrapper";
import AuthLayout from "layouts/AuthLayout";
import CommonLayout from "layouts/CommonLayout";
import AuthPage from "pages/auth";
import HomePage from "pages/home";
import CheckoutPage from "pages/checkout";
import "assets/scss/style.scss";
import { Spinner } from 'react-bootstrap';

import Evidently from 'aws-sdk/clients/evidently';
import config from './config';

const defaultClientBuilder = (
  endpoint,
  region,
) => {
  const credentials = {
    accessKeyId: config.credential.accessKeyId,
    secretAccessKey: config.credential.secretAccessKey
  }
  return new Evidently({
    endpoint,
    region,
    credentials,
  });
};

const App = () => {
  const [isLoading, setIsLoading] = useState(true);
  const [startTime, setStartTime] = useState(new Date());
  const [showDiscount, setShowDiscount] = useState(false);
  let client = null;
  let id = null;

  useEffect(() => {
    id = new Date().getTime().toString();
```

```
    setStartTime(new Date());
    if (client == null) {
      client = defaultClientBuilder(
        config.evidently.ENDPOINT,
        config.evidently.REGION,
      );
    }
    const evaluateFeatureRequest = {
      entityId: id,
      // Input Your feature name
      feature: 'showDiscount',
      // Input Your project name'
      project: 'EvidentlySampleApp',
    };

    // Launch
    client.evaluateFeature(evaluateFeatureRequest).promise().then(res => {
      if(res.value?.boolValue !== undefined) {
        setShowDiscount(res.value.boolValue);
      }
    });

    // Experiment
    client.evaluateFeature(evaluateFeatureRequest).promise().then(res => {
      if(res.value?.boolValue !== undefined) {
        setShowDiscount(res.value.boolValue);
      }
      getPageLoadTime()
    })

    setIsLoading(false);
  },[]);

  const getPageLoadTime = () => {
    const timeSpent = (new Date().getTime() - startTime.getTime()) * 1.000001;
    const pageLoadTimeData = `{
      "details": {
        "pageLoadTime": ${timeSpent}
      },
      "UserDetails": { "userId": "${id}", "sessionId": "${id}"}
    }`;
    const putProjectEventsRequest = {
      project: 'EvidentlySampleApp',
      events: [
```

```
        {
          timestamp: new Date(),
          type: 'aws.evidently.custom',
          data: JSON.parse(pageLoadTimeData)
        },
      ],
    };
    client.putProjectEvents(putProjectEventsRequest).promise();
  }
  return (
    !isLoading? (
    <AuthProvider>
      <CommonProvider>
        <ProductsProvider>
          <CartProvider>
            <CheckoutProvider>
              <Router>
                <Switch>
                  <RouteWrapper
                    path="/"
                    exact
                    component={() => <HomePage showDiscount={showDiscount}/>}
                    layout={CommonLayout}
                  />
                  <RouteWrapper
                    path="/checkout"
                    component={CheckoutPage}
                    layout={CommonLayout}
                  />
                  <RouteWrapper
                    path="/auth"
                    component={AuthPage}
                    layout={AuthLayout}
                  />
                </Switch>
              </Router>
            </CheckoutProvider>
          </CartProvider>
        </ProductsProvider>
      </CommonProvider>
    </AuthProvider> ) : (
      <Spinner animation="border" />
    )
  );
```

```
};

export default App;
```

每當使用者存取範例應用程式時，自訂指標會傳送至 Evidently 進行分析。Evidently 會分析每個指標，並在 Evidently 儀表板上即時顯示結果。以下範例顯示了指標酬載：

```
[ {"timestamp": 1637368646.468, "type": "aws.evidently.custom", "data": "{\"details
\":{\"pageLoadTime\":2058.002058},\"userDetails\":{\"userId\":\"1637368644430\",
\"sessionId\":\"1637368644430\"}}" } ]
```

步驟 5：建立專案、功能和實驗

接下來，您可以在 CloudWatch Eviature 控制台中創建項目，功能和實驗。

為此教學課程建立專案、功能和實驗

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，顯而易見。
3. 選擇 Create project (建立專案) 並填寫欄位。您必須使用 **EvidentlySampleApp**，以使範例的專案名稱正常工作。針對評估事件儲存，選擇不存放評估事件。

   填寫完欄位後，選擇 Create Project (建立專案)。

   如需詳細資訊，請參閱建立新專案。
4. 專案建立之後，在該專案中建立功能。命名功能 **showDiscount**。在此功能中，建立 **Boolean** 類型的兩個變化。命名具有 **False** 值的第一個變化 **disable** 並命名第二個具有 **True** 值的第二個變化 **enable**。

   如需建立功能的詳細資訊，請參閱 將功能新增至專案。
5. 完成建立功能後，請在專案中建立實驗。命名實驗 **pageLoadTime**。

   此實驗將使用稱為 pageLoadTime 的自訂指標，測量正在測試之頁面的頁面載入時間。實驗的自訂指標是使用 Amazon 建立的 EventBridge。有關更多信息 EventBridge，請參閱什麼是 Amazon EventBridge？ 。

   若要建立該自訂指標，請在建立實驗時執行下列動作：

   - 在 Metrics (指標) 下，對於 Metric source (指標來源)，選擇 Custom metrics (自訂指標)。

- 對於 Metric name (指標名稱)，輸入 **pageLoadTime**。
- 對於 Goal (目標)，選擇 Decrease (降低)。這表示我們想要以此指標的較低值來表示功能的最佳變化。
- 對於 Metric rule (指標規則)，輸入下列內容：
  - 針對 Entity ID (實體 ID)，輸入 **UserDetails.userId**。
  - 對於 Value key (值索引鍵)，輸入 **details.pageLoadTime**。
  - 對於 Unit (單位)，輸入 **ms**。
- 選擇 Add metric (新增指標)。

對於 Audiences (受眾)，選取 100%，以便將所有使用者輸入實驗中。將變化之間的流量分割設定為每個 50%。

之後，請選擇 Create experiment (建立實驗) 來建立實驗。建立後，在您讓 Evidently 啟動前，它不會啟動。

步驟 6： CloudWatch 明顯地開始實驗和測試

最後的步驟是開始實驗並啟動範例應用程式。

開始教學課程實驗

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導航窗格中，選擇應用程序信號，顯而易見。
3. 選擇EvidentlySampleApp專案。
4. 選擇 Experiments (實驗) 索引標籤。
5. 選擇旁邊的按鈕，pageLoadTime然後選擇動作，開始實驗。
6. 選擇實驗結束的時間。
7. 選擇 Start experiment (開始實驗)。

   實驗會即刻開始。

接著，使用以下命令啟動 Evidently 範例應用程式：

```
npm install -f && npm start
```

應用程式啟動之後，系統會將您指派給正在測試的兩個功能變體之一。一個變體顯示「20% 的折扣」，另一個則不顯示。繼續重新整理頁面以查看不同的變體。

> **Note**
>
> Evidently 具有黏性評估。功能評估是確定性的，意味著針對相同的 `entityId` 和功能，使用者將始終收到相同的變體指派。變體指派變更僅發生在將實體新增至撥號的覆寫或實驗流量時。
> 但是，為了讓您輕鬆使用範例應用程式教學，Evidently 每次在您重新整理頁面時，都會重新指派範例應用程式功能評估，以便您可以體驗這兩種變體，而無需新增覆寫。

疑難排解

我們建議您使用 npm 6.14.14 版。如果您看到有關建置或啟動範例應用程式的任何錯誤，並且您使用不同版本的 npm，請執行以下操作。

安裝 **npm** 6.14.14 版

1. 使用瀏覽器連線至 https://nodejs.org/download/release/v14.17.5/。
2. 下載 node-v14.17.5.pkg 並執行此 pkg 來安裝 npm。

   如果您看到 `webpack not found` 錯誤，請導覽至 `evidently-sample-shopping-app` 資料夾並嘗試以下操作：

   a. 刪除 `package-lock.json`
   b. 刪除 `yarn-lock.json`
   c. 刪除 `node_modules`
   d. 從 `package.json` 刪除 Webpack 相依項
   e. 執行下列命令：

   ```
   npm install -f && npm
   ```

使用 CloudWatch 朗姆酒

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。您可以視覺化和分析的資料包括頁面載入時間、用戶端錯

誤和使用者行為。當您檢視此資料時，您可以看到所有資料彙總在一起，也可以查看依客戶使用之瀏覽器和裝置劃分的明細內容。

您可以使用收集到的資料來快速識別用戶端效能問題並進行偵錯。 CloudWatch RUM 可協助您將應用程式效能中的異常視覺化，並尋找相關的偵錯資料，例如錯誤訊息、堆疊追蹤和使用者工作階段。您也可以使用 RUM 來了解最終使用者影響的範圍，包括使用者數量、地理位置和使用的瀏覽器。

您針對 CloudWatch RUM 收集的使用者資料會保留 30 天，然後自動刪除。如果您想要將 RUM 事件保留較長時間，您可以選擇讓應用程式監視器將事件的副本傳送至您帳戶中的 CloudWatch 記錄檔。然後，您可以調整該日誌群組的保留期間。

若要使用 RUM，您可以建立應用程式監控並提供一些資訊。RUM 會產生一個程式 JavaScript 碼片段供您貼到應用程式中。程式碼片段會提取 RUM Web 用戶端程式碼。RUM Web 用戶端會從應用程式的使用者工作階段中擷取某個百分比的資料，而該工作階段會顯示在預先建置的儀表板中。您可以指定要收集資料的使用者工作階段百分比。

CloudWatch RUM 與應用程序信號集成在一起，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。若要查看 Application Signals 中的 RUM 用戶端頁面請求，請透過建立應用程式監視器或手動設定 RUM Web 用戶端，來開啟 X-Ray 主動追蹤。您的 RUM 用戶端會顯示在與您的服務相連的服務地圖及其呼叫之服務的服務詳細資訊頁面中。

RUM Web 用戶端是開放原始碼。如需詳細資訊，請參閱 CloudWatch RUM 網路用戶端。

效能考量

本節討論使用 CloudWatch RUM 的效能考量。

- 載入效能影響 — CloudWatch RUM Web 用戶端可以作為 JavaScript 模組安裝在 Web 應用程式中，或從內容傳遞網路 (CDN) 以非同步方式載入您的 Web 應用程式。它不會阻止應用程序的加載過程。CloudWatch RUM 旨在對應用程序的加載時間沒有明顯的影響。
- 執行階段影響 — RUM 網路用戶端會執行處理，以記錄 RUM 資料並將其傳送至 CloudWatch RUM 服務。由於事件很少發生，而且處理量很小，因此 CloudWatch RUM 的設計是為了不會對應用程式的效能造成任何可偵測的影響。
- 網路影響 — RUM 網頁用戶端會定期將資料傳送至 CloudWatch RUM 服務。資料會在應用程式執行時定期分派，也會在瀏覽器卸載應用程式之前立即分派。在瀏覽器卸載應用程式之前立即傳送的資料會以指標的形式傳送，而此設計對應用程式之卸載時間沒有任何可偵測到的影響。

RUM 定價

使用 CloudWatch RUM 時，您會對 RUM 收到的每個 RUM 事件 CloudWatch 產生費用。使用 RUM Web 用戶端收集的每個資料項目皆會視為 RUM 事件。RUM 事件的範例包括頁面檢視、 JavaScript 錯誤和 HTTP 錯誤。您可以選擇每個應用程式監控會收集哪些類型的事件。您可以啟用或停用選項，以收集效能遙測事件、 JavaScript 錯誤、HTTP 錯誤和 X-Ray 追蹤。如需這些選項的詳細資訊，請參閱步驟 2：建立應用程式監控 和 CloudWatch RUM 網頁用戶端收集的資訊。如需有關定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

區域可用性

CloudWatch RUM 目前在下列地區提供：

- 美國東部 (維吉尼亞北部)
- 美國東部 (俄亥俄)
- 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 (雅加達)
- 亞太區域 (孟買)
- 亞太區域 (海德拉巴)
- 亞太區域 (墨爾本)
- 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)
- Europe (Paris)
- 歐洲 (西班牙)

- 歐洲 (斯德哥爾摩)
- 歐洲 (蘇黎世)
- Middle East (Bahrain)
- 中東 (阿拉伯聯合大公國)
- 南美洲 (聖保羅)

主題

- 使用 RUM 的 IAM CloudWatch 政策
- 設定應用程式以使用 CloudWatch RUM
- 設定 R CloudWatch UM 網頁用戶端
- 區域化
- 使用頁面群組
- 指定自訂中繼資料
- 傳送自訂事件
- 檢視 R CloudWatch UM 儀表板
- CloudWatch 您可以使用 CloudWatch RUM 收集的指標
- CloudWatchRUM 的資料保護和資料隱私
- CloudWatch RUM 網頁用戶端收集的資訊
- 管理使用 CloudWatch RUM 的應用程式
- CloudWatch 朗姆酒配額
- 疑難排解 CloudWatch 朗

使用 RUM 的 IAM CloudWatch 政策

若要能夠完全管理 CloudWatch RUM，您必須以 IAM 使用者或具有 AmazonCloudWatchRUM FullAccess IAM 政策的角色身分登入。此外，您可能需要其他政策或許可：

- 若要建立可建立用於授權的新 Amazon Cognito 身分集區的應用程式監視器，您需要具有管理員 IAM 角色或 AdministratorAccessIAM 政策。
- 若要建立將資料傳送至 CloudWatch Logs 的應用程式監視器，您必須登入具有下列權限的 IAM 角色或政策：

```
{
```

```
    "Effect": "Allow",
    "Action": [
        "logs:PutResourcePolicy"
    ],
    "Resource": [
        "*"
    ]
}
```

其他需要檢視 CloudWatch RUM 資料但不需要建立 CloudWatch RUM 資源的使用者，可以授與 AmazonCloudWatchRUM ReadOnlyAccess 政策。

設定應用程式以使用 CloudWatch RUM

使用這些章節中的步驟來設定應用程式，以開始使用 CloudWatch RUM 從實際使用者工作階段收集效能資料。

主題

- 步驟 1：授權您的應用程序將數據發送到 AWS
- 步驟 2：建立應用程式監控
- (選擇性) 步驟 3：手動修改程式碼片段以設定 CloudWatch RUM Web 用戶端
- 步驟 4：將程式碼片段插入應用程式
- 步驟 5：透過產生使用者事件測試您的應用程式監控設定

步驟 1：授權您的應用程序將數據發送到 AWS

要使用 CloudWatch RUM，您的應用程序必須具有授權。

您有三個選項可以設定授權：

- 讓 CloudWatch RUM 為應用程式建立新的 Amazon Cognito 身分集區。這個方法需要最少的設定工作。此為預設選項。

  身分集區將包含未進行身分驗證的身分。這可讓 CloudWatch RUM 網頁用戶端在不驗證應用程式使用者的情況下將資料傳送至 CloudWatch RUM。

  Amazon Cognito 身分集區具有連接的 IAM 角色。Amazon Cognito 未經驗證的身分可讓網路用戶端採用獲授權將資料傳送至 RUM 的 IAM 角色。 CloudWatch

- 使用現有 Amazon Cognito 身分集區。在此情況下，您也必須修改連接至身分集區的 IAM 角色。針對支援未驗證使用者的身分識別集區使用此選項。您只能使用來自相同區域的身分集區。
- 使用您已經設定的現有身分提供者所提供的身分驗證。在此情況下，您必須從身分提供者取得憑證，且您的應用程式必須將這些憑證轉寄至 RUM Web 用戶端。

  針對僅支援已驗證使用者的身分識別集區使用此選項。

以下章節更會詳細探討這些選項。

CloudWatch RUM 創建一個新的 Amazon Cognito 身份池

這是最簡單的設定選項，如果您選擇此選項，則無需進一步設定步驟。您必須具備管理許可，才能使用此選項。如需詳細資訊，請參閱 使用 RUM 的 IAM CloudWatch 政策。

使用此選項， CloudWatch RUM 會建立下列資源：

- 新的 Amazon Cognito 身分集區
- 未進行身分驗證的 Amazon Cognito 身分。這可讓 RUM Web 用戶端擔任 IAM 角色，而無需對應用程式的使用者進行身分驗證。
- RUM Web 用戶端將擔任的 IAM 角色。連接至此角色的 IAM 政策允許其將 `PutRumEvents` API 與應用程式監控資源搭配使用。換句話說，其允許 RUM Web 用戶端將資料傳送至 RUM。

RUM 網路用戶端會使用 Amazon Cognito 身分來取得 AWS 登入資料。 AWS 登入資料與 IAM 角色相關聯。IAM 角色已獲授權可與 AppMonitor 資源`PutRumEvents`搭配使用。

Amazon Cognito 會傳送必要的安全性權杖，讓您的應用程式能夠將資料傳送至 CloudWatch RUM。CloudWatch RUM 生成的 JavaScript 代碼片段包括以下幾行以啟用身份驗證。

```
   {
    identityPoolId: [identity pool id],  // e.g., 'us-west-2:EXAMPLE4a-66f6-4114-902a-
EXAMPLEbad7'
   }
);
```

使用現有 Amazon Cognito 身分集區

如果您選擇使用現有的 Amazon Cognito 身分識別集區，請在將應用程式新增至 CloudWatch RUM 時指定身分集區。集區必須支援啟用對未經身分驗證之身分的存取權。您只能使用來自相同區域的身分集區。

您也必須將下列許可新增到連接至與此身分集區相關聯之 IAM 角色的 IAM 政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rum:PutRumEvents",
            "Resource": "arn:aws:rum:[region]:[accountid]:appmonitor/[app monitor
 name]"
        }
    ]
}
```

然後，Amazon Cognito 會傳送必要的安全性權杖，讓您的應用程式能夠存取 CloudWatch RUM。

第三方供應商

如果您選擇使用第三方供應商的私有身分驗證，則必須從身分提供者取得憑證，並將其轉寄至 AWS。最好的方法是使用安全字符廠商。您可以使用任何安全性權杖廠商，包括 Amazon Cognito 與 AWS Security Token Service. 如需詳細資訊 AWS STS，請參閱歡迎使用 AWS Security Token Service API 參考。

如果您想要在此案例中使用 Amazon Cognito 作為字符廠商，則可以將 Amazon Cognito 設定為與身分驗證供應商一起使用。如需詳細資訊，請參閱 Amazon Cognito 身分集區 (聯合身分) 入門。

將 Amazon Cognito 設定為與身分提供者一起使用後，您還需要執行下列動作：

- 建立具備下列許可的 IAM 角色。您的應用程式將使用此角色來存取 AWS。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rum:PutRumEvents",
```

```
            "Resource": "arn:aws:rum:[region]:[accountID]:appmonitor/[app monitor
 name]"
        }
    ]
}
```

- 將以下內容添加到您的應用程序中，以使其將憑據從您的提供商傳遞給 CloudWatch RUM。插入該行，以便在使用者登入您的應用程式，並且應用程式已收到用於存取 AWS的憑證後執行。

```
cwr('setAwsCredentials', {/* Credentials or CredentialProvider */});
```

有關 SDK 中認證提供者的詳細資訊，請參閱 v3 AWS JavaScript SDK 開發人員指南中的 Web 瀏覽器中的設定認證 JavaScript、以及 @aws-sd k/憑證提供者的 v2 開發人員指南中的在 Web 瀏覽器中設定認證。 JavaScript

您也可以使用 CloudWatch RUM 網頁用戶端的 SDK 來設定 Web 用戶端驗證方法。如需有關網路用戶端 SDK 的詳細資訊，請參閱 CloudWatch RUM 網路用戶端 SDK。

步驟 2：建立應用程式監控

若要開始搭配應用程式使用 CloudWatch RUM，請建立應用程式監視器。建立應用程式監視器時，RUM 會產生一個程式 JavaScript碼片段供您貼到應用程式中。程式碼片段會提取 RUM Web 用戶端程式碼。RUM Web 用戶端會擷取應用程式使用者工作階段中某個百分比的資料，並將其傳送至 RUM。

建立應用程式監控

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在瀏覽窗格中，選擇應用程式訊號，RUM。
3. 選擇 Add app monitor (新增應用程式監控)。
4. 輸入應用程式的資訊和設定：

   - 針對應用程式監視器名稱，請輸入要在 CloudWatch RUM 主控台中識別此應用程式監視器的名稱。
   - 對於 Application domain (應用程式網域)，輸入您應用程式擁有管理授權的頂層網域名稱。這必須是 URL 網域格式。

     選擇 Include sub domains (包含子網域)，讓應用程式監控也會從頂層網域下的所有子網域收集資料。

5. 對於 Configure RUM data collection (設定 RUM 資料收集)，指定是否要應用程式監控來收集下列各項：

   - Performance telemetry (效能遙測) – 收集頁面載入和資源載入時間的相關資訊
   - JavaScript 錯誤 — 收集應用程式所引發之未處理 JavaScript 錯誤的相關資訊
   - HTTP errors (HTTP 錯誤) – 收集應用程式所擲回之 HTTP 錯誤的相關資訊

   選取這些選項可提供有關應用程式的詳細資訊，但也會產生更多 CloudWatch RUM 事件，因此會產生更多費用。

   如果您沒有選取任何這些項目，則應用程式監控仍會收集工作階段啟動事件和頁面 ID，以便查看有多少使用者正在使用您的應用程式，包括依作業系統類型和版本、瀏覽器類型和版本、裝置類型和位置劃分的明細內容。
6. 如果您希望能夠從取樣的使用者工作階段收集使用者 ID 和工作階段 ID，請選取核取此選項，允許 CloudWatch RUM Web 用戶端設定 Cookie。使用者 ID 是由 RUM 隨機產生。如需詳細資訊，請參閱 CloudWatch RUM 網頁用戶端 Cookie（或類似技術）。
7. 對於 Session samples (工作階段範例)，輸入會用於收集 RUM 資料的使用者工作階段百分比。預設為 100%。減少此數量會讓您減少資料，但也會降低費用。如需有關 RUM 定價的詳細資訊，請參閱 RUM 定價。
8. 您針對 CloudWatch RUM 收集的使用者資料會保留 30 天，然後刪除。如果您想要在 CloudWatch 記錄中保留 RUM 事件的副本，並設定保留這些副本的時間長度，請選擇核取此選項，將應用程式遙測資料儲存在 [資料儲存體] 底下的 CloudWatch 記錄帳戶中。根據預設，記 CloudWatch 錄檔記錄群組會保留資料 30 天。您可以在「CloudWatch 記錄檔」主控台中調整保留期間。
9. 對於 Authorization (授權)，請指定要使用新的或現有的 Amazon Cognito 身分集區，還是使用不同的身分提供者。建立新的身分集區是不需要其他設定步驟的最簡單選項。如需詳細資訊，請參閱步驟 1：授權您的應用程序將數據發送到 AWS。

   建立新的 Amazon Cognito 身分集區需要管理許可。如需詳細資訊，請參閱 使用 RUM 的 IAM CloudWatch 政策。
10. (選擇性) 根據預設，當您將 RUM 程式碼片段新增至應用程式時，Web 用戶端會將標籤注入 JavaScript 標籤，以監控應用程式所有頁面的 HTML 程式碼中的使用情況。若要變更此選項，請選擇 Configure pages (設定頁面)，然後選擇 Include only these pages (僅包含這些頁面) 或 Exclude these pages (排除這些頁面)。然後，指定要包含或排除的頁面。若要指定要包含或排除的頁面，請輸入其完整 URL。若要指定其他頁面，請選擇 Add URL (新增網址)。
11. 若要啟用 AWS X-Ray 追蹤應用程式監視器取樣的使用者工作階段，請選擇 [作用中追蹤]，然後選取 [追蹤我的服務使用]。 AWS X-Ray

如果您選取此選項，則會追蹤由應用程式監控取樣的使用者工作階段期間所發出的 `XMLHttpRequest` 和 `fetch` 請求。然後，您可以在 RUM 儀表板、X-Ray 追蹤地圖以及追蹤詳細資訊頁面中查看這些使用者工作階段的追蹤和區段。在為應用程式啟用它之後，這些使用者工作階段也會在 Application Signals 中顯示為用戶端頁面。

透過對 R CloudWatch UM Web 用戶端進行其他組態變更，您可以將 X-Ray 追蹤標頭新增至 HTTP 要求，以便 end-to-end 追蹤下游 AWS 受管理服務的使用者工作階段。如需詳細資訊，請參閱 啟用 X-Ray end-to-end 追蹤。

12. (選用) 若要新增標籤至應用程式監控，請選擇 Tags (標籤)、Add new tag (新增標籤)。

    之後，在 Key (索引鍵) 中，輸入標籤的名稱。您可以在 Value (值) 中為標籤新增選用值。

    若要新增另一個標籤，請再次選擇 Add new tag (新增標籤)。

    如需詳細資訊，請參閱標記 AWS 資源。

13. 選擇 Add app monitor (新增應用程式監控)。

14. 在 Sample code (範本程式碼) 區段中，您可以複製程式碼片段，以便新增至應用程式中。我們建議您選擇JavaScript或TypeScript使用 NPM 將 CloudWatch RUM Web 用戶端安裝為 JavaScript 模組。

    或者，您可以選擇 HTML 來使用內容傳遞網路 (CDN) 來安裝 CloudWatch RUM 網路用戶端。使用 CDN 的缺點是 Web 用戶端通常會遭廣告封鎖程式封鎖。

15. 選擇 Copy (複製) 或 Download (下載)，然後選擇 Done (完成)。

(選擇性) 步驟 3：手動修改程式碼片段以設定 CloudWatch RUM Web 用戶端

您可以在將程式碼片段插入應用程式之前修改程式碼片段，以啟用或停用數個選項。如需詳細資訊，請參閱 CloudWatch RUM 網路用戶端文件。

有三個您應該務必注意到的組態選項，如這些章節所述。

防止收集可能包含個人資訊的資源 URL

根據預設， CloudWatch RUM Web 用戶端會設定為記錄應用程式下載的資源 URL。這些資源包括 HTML 文件，圖像，CSS JavaScript 文件，文件等。對於某些應用程式，URL 可能包含個人身分識別資訊 (PII)。

如果您的應用程式發生這種情況，強烈建議您透過在程式碼片段組態中設定 `recordResourceUrl: false` 來停用資源 URL 收集，之後再將其插入到您的應用程式中。

手動記錄頁面檢視

預設情況下，Web 用戶端會記錄頁面第一次載入時，以及呼叫瀏覽器歷史記錄 API 時的頁面檢視次數。預設頁面 ID 為 `window.location.pathname`。但在某些情況下，您可以覆寫此行為並檢測應用程式，進而以程式設計方式記錄頁面檢視。這樣做可讓您控制頁面 ID 以及其記錄時間。舉例來說，假設 Web 應用程式的 URI 具有變數識別符，例如 `/entity/123` 或 `/entity/456`。根據預設，CloudWatch RUM 會為每個 URI 產生一個頁面檢視事件，其中包含與路徑名稱相符的不同頁面 ID，但您可能想要將它們分組為相同的頁面 ID。若要完成此操作，請使用 `disableAutoPageView` 組態停用 Web 客戶端的頁面檢視自動化，然後使用 `recordPageView` 命令設定所需的頁面 ID。如需詳細資訊，請參閱上 GitHub的應用程式特定組態。

內嵌指令碼範例：

```
cwr('recordPageView', { pageId: 'entityPageId' });
```

JavaScript 模塊示例：

```
awsRum.recordPageView({ pageId: 'entityPageId' });
```

啟用 X-Ray end-to-end 追蹤

當您建立應用程式監控時，選取 Trace my service with AWS X-Ray (使用 追蹤我的服務) 會啟用對應用程式監控取樣之使用者工作階段期間所提出 `XMLHttpRequest` 和 `fetch` 請求的追蹤。然後，您可以在 R CloudWatch UM 儀表板以及 X-Ray 追蹤對應和追蹤詳細資料頁面中查看來自這些 HTTP 要求的追蹤。

根據預設，這些用戶端追蹤不會連線到下游伺服器端追蹤。若要將用戶端追蹤連線至伺服器端 end-to-end 追蹤並啟用追蹤，請在 Web 用戶端`true`中將`addXRayTraceIdHeader`選項設定為。這會導致 R CloudWatch UM 網頁用戶端將 X-Ray 追蹤標頭新增至 HTTP 要求。

下列程式碼區塊會顯示新增用戶端追蹤的範例。為了便於閱讀，此範例會省略某些組態選項。

```
<script>
    (function(n,i,v,r,s,c,u,x,z){...})(
        'cwr',
        '00000000-0000-0000-0000-000000000000',
```

```
        '1.0.0',
        'us-west-2',
        'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js',
        {
            enableXRay: true,
            telemetries: [
                'errors',
                'performance',
                [ 'http', { addXRayTraceIdHeader: true } ]
            ]
        }
    );
</script>
```

> **Warning**
>
> 設定 R CloudWatch UM 網頁用戶端以將 X-Ray 追蹤標頭新增至 HTTP 要求時，可能會導致跨來源資源共用 (CORS) 失敗或使要求的簽章無效 (如果使用 Sigv4 簽署)。如需詳細資訊，請參閱 CloudWatch RUM 網路用戶端文件。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

步驟 4：將程式碼片段插入應用程式

接下來，您會將上一節所建立的程式碼片段插入您的應用程式。

> **Warning**
>
> 由程式碼片段下載和設定的 Web 用戶端會使用 Cookie (或類似技術) 來協助您收集最終使用者資料。插入程式碼片段之前，請參閱 在主控台中依中繼資料屬性進行篩選。

如果您沒有先前產生的程式碼片段，則可以透過 如何找到我已經產生的程式碼片段？ 中的以下指示找到。

若要將 CloudWatch RUM 程式碼片段插入應用程式

1. 將您在上一節中複製或下載的程式碼片段插入應用程式的 <head> 元素。將程式碼片段插入 <body> 元素或任何其他 <script> 標籤。

以下是所產生程式碼片段的範例。

```
<script>
(function (n, i, v, r, s, c, x, z) {
    x = window.AwsRumClient = {q: [], n: n, i: i, v: v, r: r, c: c};
    window[n] = function (c, p) {
        x.q.push({c: c, p: p});
    };
    z = document.createElement('script');
    z.async = true;
    z.src = s;
    document.head.insertBefore(z, document.getElementsByTagName('script')[0]);
})('cwr',
    '194a1c89-87d8-41a3-9d1b-5c5cd3dafbd0',
    '1.0.0',
    'us-east-2',
    'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js',
    {
        sessionSampleRate: 1,
        identityPoolId: "us-east-2:c90ef0ac-e3b8-4d1a-b313-7e73cfd21443",
        endpoint: "https://dataplane.rum.us-east-2.amazonaws.com",
        telemetries: ["performance", "errors", "http"],
        allowCookies: true,
        enableXRay: false
    });
</script>
```

2. 如果您的應用程式是多頁面 Web 應用程式，則您必須針對要包含在資料收集中的每個 HTML 頁面重複步驟 1。

步驟 5：透過產生使用者事件測試您的應用程式監控設定

插入程式碼片段且已更新的應用程式正在執行之後，您可以手動產生使用者事件來對其進行測試。為了對此進行測試，建議您進行下列動作。此測試會產生標準 CloudWatch RUM 費用。

- 在 Web 應用程式中的頁面之間導航。
- 使用不同的瀏覽器和裝置建立多個使用者工作階段。
- 提出請求。
- 導致 JavaScript 錯誤。

產生一些事件之後，請在 CloudWatch RUM 儀表板中檢視這些事件。如需詳細資訊，請參閱 檢視 R CloudWatch UM 儀表板。

來自使用者工作階段的資料最多可能需要 15 分鐘才會顯示在儀表板中。

如果您在應用程式中產生事件 15 分鐘後看不到資料，則請參閱 疑難排解 CloudWatch 朗。

設定 R CloudWatch UM 網頁用戶端

您的應用程式可以使用 CloudWatch RUM 產生的其中一個程式碼片段來安裝 CloudWatch RUM 網頁用戶端。產生的程式碼片段支援兩種安裝方法：做為透過 NPM 的 JavaScript 模組，或來自內容傳遞網路 (CDN)。為獲得最佳效能，我們建議您使用 NPM 安裝方法。如需有關使用此方法的詳細資訊，請參閱安裝為 JavaScript 模組。

如果您使用 CDN 安裝選項，廣告攔截器可能會封鎖 CloudWatch RUM 提供的預設 CDN。這將停用對已安裝廣告封鎖程式的使用者的應用程式監控。因此，我們建議您僅將預設 CDN 用於 CloudWatch RUM 的初始上線。如需有關此問題緩解方法的詳細資訊，請參閱檢測應用程式。

程式碼片段位於 HTML 檔案的 <head> 標籤，並透過下載 Web 用戶端，然後為其監控的應用程式設定 Web 用戶端，來安裝 Web 用戶端。程式碼片段是自行執行的函數，看起來類似下列內容。在此範例中，為了易於閱讀，已省略程式碼片段函數的主體。

```
<script>
(function(n,i,v,r,s,c,u,x,z){...})(
'cwr',
'00000000-0000-0000-0000-000000000000',
'1.0.0',
'us-west-2',
'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js',
{ /* Configuration Options Here */ }
);
<script>
```

引數

程式碼片段接受六個引數：

- 用於在 Web 用戶端上執行命令的命名空間，例如 `'cwr'`
- 應用程式監控的 ID，例如 `'00000000-0000-0000-0000-000000000000'`
- 應用程式版本，例如 `'1.0.0'`

- 應用程式監視器的 AWS 區域，例如 'us-west-2'
- Web 用戶端的 URL，例如 'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js'
- 應用程式特定的組態選項。如需詳細資訊，請參閱下一節。

忽略錯誤

CloudWatch RUM 網頁用戶端會偵聽應用程式中發生的所有類型的錯誤。如果您的應用程式發出您不想在 CloudWatch RUM 儀表板中檢視的 JavaScript 錯誤，您可以設定 CloudWatch RUM Web 用戶端來篩選掉這些錯誤，讓您只能在 CloudWatch RUM 儀表板上看到相關的錯誤事件。例如，您可能會選擇不檢視儀表板中的某些 JavaScript錯誤，因為您已經識別錯誤的修正程式，而且這些錯誤的磁碟區正在遮罩其他錯誤。您可能還想忽略無法修正的錯誤，因為這些錯誤是由第三方擁有的程式庫所擁有。

有關如何檢測 Web 客戶端以過濾掉特定 JavaScript 錯誤的詳細信息，請參閱 Web 客戶端 Github 文檔中的錯誤中的示例。

組態選項

如需 CloudWatch RUM Web 用戶端可用組態選項的相關資訊，請參閱 CloudWatch RUM 網路用戶端文件

區域化

本節說明在不同區域中搭配應用程式使用 CloudWatch RUM 的策略。

我的 Web 應用程序部署在多個 AWS 區域

如果您的 Web 應用程式部署在乘數 AWS 區域中，您有三個選項：

- 在單一區域的單一帳戶中，部署一個應用程式監視器，並為所有區域提供服務。
- 在多個唯一的帳戶中，為每個區域部署獨立的應用程式監視器。
- 為每個區域部署獨立的應用程式監視器，且全部都在單一帳戶中。

使用一個應用程式監視器的優點是，所有資料都會集中在一個視覺效果中，而且所有記錄檔都會寫入記錄檔中的相同 CloudWatch 記錄群組。使用單一應用程式監視器時，請求會有少量的額外延遲，以及單一失敗點。

使用多個應用程式監視器可移除單一失敗點，但會導致所有資料無法合併為單一視覺效果。

CloudWatch RUM 尚未在部署我的應用程式的某些區域中啟動

CloudWatch RUM 被推出到許多地區，並具有廣泛的地理覆蓋範圍。透過在提供 CloudWatch RUM 的區域設定 RUM，您可以獲得優點。如果您在最終使用者連線的區域中設定應用程式監視器，則使用者將可以在任何地方，同時仍包含其工作階段。

然而， CloudWatch RUM 尚未在美 AWS GovCloud 國東部、 AWS GovCloud 美國西部或中國任何地區推出。您無法從這些區域傳送資料至 CloudWatch RUM。

使用頁面群組

使用頁面群組將應用程式中的不同頁面相互關聯，以便您可以查看網頁群組的彙總分析。例如，您可能想要查看所有登陸頁面的彙總頁面載入時間。

您可以在 CloudWatch RUM Web 用戶端中新增一或多個標記至頁面檢視事件，將頁面放入頁面群組。下列範例將 /home 頁面放入名為 en 的頁面群組和名為 `landing` 的頁面群組。

內嵌指令碼範例

```
cwr('recordPageView', { pageId: '/home', pageTags: ['en', 'landing']});
```

JavaScript 模塊示例

```
awsRum.recordPageView({ pageId: '/home', pageTags: ['en', 'landing']});
```

> **Note**
>
> 頁面群組旨在協助彙總不同頁面的分析。如需有關如何為應用程式定義和操作 `pageIds` 的資訊，請參閱「(選擇性) 步驟 3：手動修改程式碼片段以設定 CloudWatch RUM Web 用戶端」中的「手動記錄頁面檢視」一節。

指定自訂中繼資料

CloudWatch RUM 附加額外的數據作為元數據的每個事件。事件中繼資料是由鍵-值對形式的屬性組成。您可以使用這些屬性來搜尋或篩選 CloudWatch RUM 主控台中的事件。根據預設， CloudWatch RUM 會為您建立一些中繼資料。如需有關預設中繼資料的詳細資訊，請參閱 RUM 事件中繼資料。

您也可以使用 CloudWatch RUM 網頁用戶端，將自訂中繼資料新增至 CloudWatch RUM 事件。自訂中繼資料可以包含工作階段屬性和頁面屬性。

若要新增自訂中繼資料，您必須使用 CloudWatch RUM 網頁用戶端的 1.10.0 版或更新版本。

要求與語法

每個事件可以在中繼資料中包含多達 10 個自訂屬性。自訂屬性的語法要求如下：

- 鍵
  - 最多 128 個字元。
  - 可以包含英數字元、冒號 (:) 和底線 (_)。
  - 開頭不能是 `aws:`。
  - 不能完全由下一節中列出的任何保留關鍵字組成。可以將這些關鍵字用作鍵名稱的一部分。
- Values (數值)
  - 最多 256 個字元。
  - 必須是字串、數字或布林值

保留的關鍵字

您不能使用下列保留關鍵字作為完整鍵名稱。您可以使用下列關鍵字作為鍵名稱的一部分，例如 `applicationVersion`。

- `browserLanguage`
- `browserName`
- `browserVersion`
- `countryCode`
- `deviceType`
- `domain`
- `interaction`
- `osName`
- `osVersion`
- `pageId`
- `pageTags`
- `pageTitle`
- `pageUrl`

- parentPageId
- platformType
- referrerUrl
- subdivisionCode
- title
- url
- version

> **Note**
>
> CloudWatch 如果屬性包含無效的索引鍵或值，或者已達到每個事件 10 個自訂屬性的限制，則 RUM 會從 RUM 事件移除自訂屬性。

新增工作階段屬性

如果您設定自訂工作階段屬性，屬性會新增至工作階段中的所有事件。您可以在 CloudWatch RUM Web 用戶端初始化期間或執行階段使用addSessionAttributes命令來設定工作階段屬性。

例如，您可以將應用程式的版本新增為工作階段屬性。然後，在 CloudWatch RUM 主控台中，您可以依版本篩選錯誤，以找出提高的錯誤率是否與應用程式的特定版本相關聯。

在初始化期間新增工作階段屬性 (NPM 範例)

粗體的程式碼區段會新增工作階段屬性。

```
import { AwsRum, AwsRumConfig } from 'aws-rum-web';

try {
  const config: AwsRumConfig = {
    allowCookies: true,
    endpoint: "https://dataplane.rum.us-west-2.amazonaws.com",
    guestRoleArn: "arn:aws:iam::000000000000:role/RUM-Monitor-us-
west-2-000000000000-00xx-Unauth",
    identityPoolId: "us-west-2:00000000-0000-0000-0000-000000000000",
    sessionSampleRate: 1,
    telemetries: ['errors', 'performance'],
    sessionAttributes: {
        applicationVersion: "1.3.8"
```

```
    }
  };

  const APPLICATION_ID: string = '00000000-0000-0000-0000-000000000000';
  const APPLICATION_VERSION: string = '1.0.0';
  const APPLICATION_REGION: string = 'us-west-2';

  const awsRum: AwsRum = new AwsRum(
    APPLICATION_ID,
    APPLICATION_VERSION,
    APPLICATION_REGION,
    config
  );
} catch (error) {
  // Ignore errors thrown during CloudWatch RUM web client initialization
}
```

在執行階段期間新增工作階段屬性 (NPM 範例)

```
awsRum.addSessionAttributes({
    applicationVersion: "1.3.8"
})
```

在初始化期間新增工作階段屬性 (內嵌指令碼範例)

粗體的程式碼區段會新增工作階段屬性。

```
<script>
    (function(n,i,v,r,s,c,u,x,z){...})(
        'cwr',
        '00000000-0000-0000-0000-000000000000',
        '1.0.0',
        'us-west-2',
        'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js',
        {
            sessionSampleRate:1,
            guestRoleArn:'arn:aws:iam::000000000000:role/RUM-Monitor-us-
west-2-000000000000-00xx-Unauth',
            identityPoolId:'us-west-2:00000000-0000-0000-0000-000000000000',
            endpoint:'https://dataplane.rum.us-west-2.amazonaws.com',
            telemetries:['errors','http','performance'],
            allowCookies:true,
            sessionAttributes: {
```

```
                applicationVersion: "1.3.8"
            }
        }
    );
</script>
```

在執行階段期間新增工作階段屬性 (內嵌指令碼範例)

```
<script>
    function addSessionAttribute() {
        cwr('addSessionAttributes', {
            applicationVersion: "1.3.8"
        })
    }

</script>
```

新增頁面屬性

如果您設定自訂頁面屬性，這些屬性會新增至目前頁面上的所有事件。您可以在 CloudWatch RUM Web 用戶端初始化期間或在執行階段使用`recordPageView`命令來設定頁面屬性。

例如，您可以新增頁面範本作為頁面屬性。然後，在 CloudWatch RUM 主控台中，您可以依頁面範本篩選錯誤，以找出增加的錯誤率是否與應用程式的特定頁面範本相關聯。

在初始化期間新增頁面屬性 (NPM 範例)

粗體的程式碼區段會新增頁面屬性。

```
const awsRum: AwsRum = new AwsRum(
    APPLICATION_ID,
    APPLICATION_VERSION,
    APPLICATION_REGION,
    { disableAutoPageView:  true // optional }
);
awsRum.recordPageView({
    pageId:'/home',
    pageAttributes: {
      template: 'artStudio'
    }
});
const credentialProvider = new CustomCredentialProvider();
```

```
if(awsCreds) awsRum.setAwsCredentials(credentialProvider);
```

在執行階段期間新增頁面屬性 (NPM 範例)

```
awsRum.recordPageView({
    pageId: '/home',
    pageAttributes: {
        template: 'artStudio'
    }
});
```

在初始化期間新增頁面屬性 (內嵌指令碼範例)

粗體的程式碼區段會新增頁面屬性。

```
<script>
    (function(n,i,v,r,s,c,u,x,z){...})(
        'cwr',
        '00000000-0000-0000-0000-000000000000',
        '1.0.0',
        'us-west-2',
        'https://client.rum.us-east-1.amazonaws.com/1.0.2/cwr.js',
        {
            disableAutoPageView: true //optional
        }
    );
    cwr('recordPageView', {
       pageId: '/home',
       pageAttributes: {
           template: 'artStudio'
       }
    });
    const awsCreds = localStorage.getItem('customAwsCreds');
    if(awsCreds) cwr('setAwsCredentials', awsCreds)
</script>
```

在執行階段期間新增頁面屬性 (內嵌指令碼範例)

```
<script>
    function recordPageView() {
        cwr('recordPageView', {
            pageId: '/home',
```

```
            pageAttributes: {
                template: 'artStudio'
            }
        });
    }
</script>
```

在主控台中依中繼資料屬性進行篩選

若要使用任何內建或自訂中繼資料屬性篩選 CloudWatch RUM 主控台中的視覺效果，請使用搜尋列。在搜尋列中，您可以 key=value 的形式指定多達 20 個篩選條件，以套用至視覺化效果。例如，若僅要篩選 Chrome 瀏覽器的資料，您可以新增篩選條件用語 browserName=Chrome。

根據預設， CloudWatch RUM 主控台會擷取 100 個最常用的屬性索引鍵和值，以顯示在搜尋列的下拉式清單中。若要將更多中繼資料屬性新增為篩選條件用語，請在搜尋列中輸入完整的屬性鍵值。

一個篩選條件最多可包含 20 個篩選條件用語，每個應用程式監視器最多可儲存 20 個篩選條件。儲存篩選條件時，篩選條件會儲存在 Saved filters (已儲存的篩選條件) 下拉式清單中。您也可以刪除儲存的篩選條件。

傳送自訂事件

CloudWatch RUM 會記錄並擷取中 CloudWatch RUM 網頁用戶端收集的資訊列出的事件。如果您使用 CloudWatch RUM Web 用戶端的 1.12.0 版或更新版本，則可以定義、記錄和傳送其他自訂事件。您可以為您定義的每個事件類型定義事件類型名稱和要傳送的資料。每個自訂事件承載最多可達 6 KB。

只有在應用程式監視器啟用自訂事件時，才會擷取自訂事件。若要更新應用程式監視器的組態設定，請使用 CloudWatch RUM 主控台或 UpdateAppMonitorAPI。

啟用自訂事件後，接著定義並傳送自訂事件，之後您就可以搜尋這些事件。若要搜尋它們，請使用 CloudWatch RUM 主控台中的 [事件] 索引標籤。使用事件類型進行搜尋。

要求與語法

自訂事件包含事件類型和事件詳細資料。具體要求如下：

- 事件類型
  - 這可以是事件的 type (類型) 或 name (名稱)。例如，呼叫的 CloudWatch RUM 內建事件類型JsError具有的事件類型`com.amazon.rum.js_error_event`。
  - 長度必須介於 1 與 256 個字元之間。

  - 可以是英數字元、底線、連字號和句點的組合。
- 事件詳細資訊
  - 包含要在 CloudWatch RUM 中記錄的實際數據。
  - 必須是由欄位和值組成的物件。

記錄自訂事件的範例

有兩種方法可以在 CloudWatch RUM Web 客戶端中記錄自定義事件。

- 使用 R CloudWatch UM 網頁用戶端的 `recordEvent` API。
- 使用自訂外掛程式。

使用 **`recordEvent`** API傳送自訂事件 (NPM 範例)

```
awsRum.recordEvent('my_custom_event', {
        location: 'IAD',
        current_url: 'amazonaws.com',
        user_interaction: {
            interaction_1 : "click",
            interaction_2 : "scroll"
        },
        visit_count:10
    }
)
```

使用 **`recordEvent`** API 傳送自訂事件 (內嵌指令碼範例)

```
cwr('recordEvent', {
    type: 'my_custom_event',
    data: {
        location: 'IAD',
        current_url: 'amazonaws.com',
        user_interaction: {
            interaction_1 : "click",
            interaction_2 : "scroll"
        },
        visit_count:10
    }
})
```

使用自訂外掛程式傳送自訂事件的範例

```
// Example of a plugin that listens to a scroll event, and
// records a 'custom_scroll_event' that contains the timestamp of the event.
class MyCustomPlugin implements Plugin {
    // Initialize MyCustomPlugin.
    constructor() {
        this.enabled;
        this.context;
        this.id = 'custom_event_plugin';
    }
    // Load MyCustomPlugin.
    load(context) {
        this.context = context;
        this.enable();
    }
    // Turn on MyCustomPlugin.
    enable() {
        this.enabled = true;
        this.addEventHandler();
    }
    // Turn off MyCustomPlugin.
    disable() {
        this.enabled = false;
        this.removeEventHandler();
    }
    // Return MyCustomPlugin Id.
    getPluginId() {
        return this.id;
    }
    // Record custom event.
    record(data) {
        this.context.record('custom_scroll_event', data);
    }
    // EventHandler.
    private eventHandler = (scrollEvent: Event) => {
        this.record({timestamp: Date.now()})
    }
    // Attach an eventHandler to scroll event.
    private addEventHandler(): void {
        window.addEventListener('scroll', this.eventHandler);
    }
    // Detach eventHandler from scroll event.
    private removeEventHandler(): void {
```

```
        window.removeEventListender('scroll', this.eventHandler);
    }
}
```

檢視 R CloudWatch UM 儀表板

CloudWatch RUM 可協助您從使用者工作階段收集應用程式效能的相關資料，包括頁面載入時間、Apdex 分數、使用的瀏覽器和裝置、使用者工作階段的地理位置，以及有錯誤的工作階段。所有這些資訊都會顯示在儀表板中。

檢視 RUM 儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在瀏覽窗格中，選擇應用程式訊號，RUM。

   Overview (概觀) 索引標籤會顯示您建立的其中一個應用程式監控所收集的資訊。

   窗格的頂列會顯示此應用程式監視器的下列資訊：

   - 頁面載入數量
   - 平均頁面載入速度
   - Apdex 分數
   - 與應用程式監控相關聯的任何警示狀態

   應用程式效能指標 (Apdex) 分數代表最終使用者的滿意度。分數範圍從 0 (最不滿意) 到 1 (最滿意)。分數僅根據應用程式效能而定。不會要求使用者對應用程式進行評分。如需 Apdex 分數的詳細資訊，請參閱 CloudWatch 朗姆酒如何設置 Apdex 分數。

   其中數個窗格包含可用來進一步檢查資料的連結。選擇這些連結中的任何一個皆會顯示詳細資訊，並在畫面頂端顯示效能、錯誤、HTTP 請求、工作階段、事件瀏覽器和裝置，以及使用者旅程標籤。
3. 若要進一步聚焦，請選擇 List view (清單檢視) 索引標籤，然後選擇您要聚焦的應用程式監視器的名稱。這會顯示所選應用程式監視器的下列索引標籤。

   - Performance (效能) 索引標籤會顯示頁面效能資訊，包括載入時間、工作階段資訊、請求資訊、Web 關鍵數值，以及一段時間內的頁面載入。此檢視包含用來在聚焦於 Page loads (頁面載入)、Requests (請求) Location (位置) 之間切換檢視的控制項。

- 錯誤標籤會顯示 Javascript 錯誤資訊，包括使用者最常見的錯誤訊息，以及錯誤最多的裝置和瀏覽器。此檢視包含錯誤的長條圖和錯誤的清單檢視。您可以依使用者和事件詳細資訊，篩選錯誤清單。選擇錯誤訊息以查看更多詳細資訊。
- HTTP 請求標籤會顯示 HTTP 請求資訊，包括錯誤最多的請求 URL，以及錯誤最多的裝置和瀏覽器。此標籤包含請求的長條圖、請求的清單檢視，以及網路錯誤的清單檢視。您可以依使用者和事件詳細資訊篩選清單。選擇回應碼或錯誤訊息，分別查看請求或網路錯誤的詳細資訊。
- 工作階段標籤會顯示工作階段指標。此標籤包含工作階段開始事件的長條圖，以及工作階段的清單檢視。您可以依事件類型、使用者詳細資訊和事件詳細資訊，篩選工作階段清單。選擇 sessionId，查看工作階段的更多詳細資訊。
- 事件標籤會顯示 RUM 事件的長條圖，以及事件的清單檢視。您可以依事件類型、使用者詳細資訊和事件詳細資訊，篩選事件清單。選擇 RUM 事件以查看原始事件。
- Browsers & Devices (瀏覽器和裝置) 索引標籤會顯示資訊，例如用來存取應用程式的不同瀏覽器和裝置的效能和使用情況。此檢視包含用來在聚焦於瀏覽器和裝置之間切換檢視的控制項。

  如果您將範圍縮小為單一瀏覽器，則會看到依瀏覽器版本劃分的資料。
- User Journey (使用者旅程) 索引標籤會顯示客戶用來導覽應用程式的路徑。您可以看到客戶在何處進入您的應用程式，以及他們退出應用程式的頁面。您也可以查看他們選用的路徑，以及遵循這些路徑的客戶百分比。您可以在節點上暫停，以取得有關該頁面的更多詳細資訊。您可以選擇單一路徑來反白顯示連線，以便於檢視。

4. (選用) 在前六個標籤中的任何一個上，您可以選擇頁面按鈕，然後從清單中選取頁面或頁面群組。這會將顯示的資料縮小至應用程式的單一頁面或頁面群組。您也可以將清單中的頁面和頁面群組標記為我的最愛。

CloudWatch 朗姆酒如何設置 Apdex 分數

Apdex (應用程式效能指標) 是一項開放的標準，其定義了一種報告、基準化分析和評估應用程式回應時間的方法。Apdex 分數可協助您了解並識別隨時間推移對應用程式效能的影響。

Apdex 分數表示最終使用者的滿意程度，分數範圍從 0 (最不滿意) 到 1 (最滿意)。分數僅根據應用程式效能而定。不會要求使用者對應用程式進行評分。

每個個別 Apdex 分數分屬於三種閾值之一。根據 Apdex 閾值和實際應用程式回應時間，有三種效能類型，如下所示：

- 滿意 – 實際應用程式回應時間小於或等於 Apdex 閾值。對於 CloudWatch 朗姆酒，這個閾值是 2000 毫秒或更小。

- 可接受 – 實際應用程式回應時間大於 Apdex 閾值，但小於或等於 Apdex 閾值的四倍。對於 CloudWatch 朗姆酒來說，這個範圍是 2000 至 8000 毫秒。
- 令人沮喪 – 實際應用程式回應時間大於 Apdex 閾值的四倍。對於 CloudWatch 朗姆酒，這個範圍超過 8000 毫秒。

總計 0-1 Apdex 分數是使用以下公式來計算：

```
(positive scores + tolerable scores/2)/total scores * 100
```

CloudWatch 您可以使用 CloudWatch RUM 收集的指標

此段落中的表格列示您使用 CloudWatch RUM 自動收集的測量結果。您可以在 CloudWatch 控制台中看到這些指標。如需詳細資訊，請參閱 檢視可用的指標。

您也可以選擇性地將擴展指標發送到 CloudWatch 或 CloudWatch 明顯地發送。如需詳細資訊，請參閱 延伸指標。

這些指標會名稱為 AWS/RUM 的命名空間中發佈。所有以下指標在發佈時具有 application_name 維度：此維度的值是應用程式監控的名稱。某些指標還會在發佈時具有其他維度，如表中所列。

指標	單位	描述
HttpStatusCodeCount	計數	應用程式中依回應狀態程式碼顯示的 HTTP 回應計數。 其他維度： • event_details.response.status 是回應狀態程式碼，例如 200、400、404 等。 • event_type 事件的類型。目前，此

指標	單位	描述
		維度唯一可能的值是 http。
Http4xxCount	計數	應用程式中 HTTP 回應的計數，以及 4xx 回應狀態碼。 這些是根據導致 4xx 代碼的 R http_event UM 事件計算的。
Http5xxCount	計數	應用程式中 HTTP 回應的計數，以及 5xx 回應狀態碼。 這些都是根據導致 5xx 代碼的 R http_event UM 事件計算的。
JsErrorCount	計數	擷取的 JavaScript 錯誤事件計數。
NavigationFrustratedCount	計數	導覽事件計數，duration 高於令人困擾的閾值，即 8000ms。導覽事件的持續時間以 PerformanceNavigationDuration 指標進行追蹤。

指標	單位	描述
NavigationSatisfiedCount	計數	導覽事件計數，duration 低於 Apdex 目標，即 2000ms。導覽事件的持續時間以 PerformanceNavigationDuration 指標進行追蹤。
NavigationToleratedCount	計數	導覽事件計數，duration 介於 2000ms 和 8000ms 之間。導覽事件的持續時間以 PerformanceNavigationDuration 指標進行追蹤。
PageViewCount	計數	應用程式監視器擷取的頁面檢視事件計數。 這是通過計算 page_view_event RUM 事件計算的。

指標	單位	描述
PerformanceResourceDuration	毫秒	資源事件的 duration。 其他維度： • event_details.file.type 是資源事件的檔案類型，例如樣式表、文件、影像、指令碼或字型。 • event_type 事件的類型。目前，此維度唯一可能的值是 resource。
PerformanceNavigationDuration	毫秒	導覽事件的 duration。
RumEventPayloadSize	位元組	CloudWatch RUM 攝入的每個事件的大小。您還可以使用此指標的 SampleCount 統計數字，以監控應用程式監視器正在擷取的事件數目。
SessionCount	計數	應用程式監視器擷取的工作階段啟動事件計數。換言之，啟動的新工作階段數目。
WebVitalsCumulativeLayoutShift	無	追蹤累計版面配置移位事件的值。

指標	單位	描述
WebVitalsFirstInputDelay	毫秒	追蹤第一個輸入延遲事件的值。
WebVitalsLargestContentfulPaint	毫秒	追蹤最大的內容繪製事件的值。

CloudWatch 顯然可以發送的自定義指標 CloudWatch 和擴展指標

根據預設，RUM 應用程式會監控將指標傳送至 CloudWatch. 這些預設量度和維度會列在您可以使用 CloudWatch RUM 收集的CloudWatch 量度中。

您也可以設定應用程式監視器來匯出指標。應用程序監視器可以發送擴展指標，自定義指標或兩者。它可以將它們發送到 CloudWatch 或發送給 CloudWatch 顯然，或兩者。

- 自訂指標 – 自訂指標是您定義的指標。透過自訂指標，您可以使用任何指標名稱和命名空間。若要衍生指標，您可以使用任何自訂事件、內建事件、自訂屬性或預設屬性。

  您可以將自定義指標發送給 CloudWatch 和 CloudWatch 顯而易見。
- 擴展指標 — 允許您將默認的 CloudWatch RUM 指標發送到 CloudWatch 顯然可以用於顯而易見的實驗。您也可以將任何預設 CloudWatch RUM 量度傳送至 CloudWatch 其他維度。如此一來，這些指標就能提供您更精細的檢視。

主題

- 自訂指標
- 延伸指標

自訂指標

若要傳送自訂指標，您必須使用 AWS API 或 AWS CLI 取代主控台。如需使用 AWS API 的詳細資訊，請參閱PutRumMetricsDestination和BatchCreateRumMetricDefinitions。

一個目的地可包含的延伸指標和自訂指標定義數量上限為 2000 個。對於您傳送至每個目的地的每個自訂指標和延伸指標，每個維度名稱和維度值的組合都會計入此限制。這也算作定價的 CloudWatch 自訂指標。

下列範例顯示如何建立從自訂事件衍生的自訂指標。以下是使用的自訂事件範例：

```
cwr('recordEvent', {
    type: 'my_custom_event',
    data: {
        location: 'IAD',
        current_url: 'amazonaws.com',
        user_interaction: {
            interaction_1 : "click",
            interaction_2 : "scroll"
        },
        visit_count:10
    }
})
```

如果是此自訂事件，您可以建立自訂指標來計算透過 Chrome 瀏覽器造訪 amazonaws.com URL 的次數。在 RUM/CustomMetrics/PageVisits 命名空間中，下列定義會在您的帳戶中建立名為 AmazonVisitsCount 的指標。

```
{
    "AppMonitorName":"customer-appMonitor-name",
    "Destination":"CloudWatch",
    "MetricDefinitions":[
        {
            "Name":"AmazonVisitsCount",
            "Namespace":"PageVisit",
            "ValueKey":"event_details.visit_count",
            "UnitLabel":"Count",
            "DimensionKeys":{
                "event_details.current_url": "URL"
            },
            "EventPattern":"{\"metadata\":{\"browserName\":[\"Chrome\"]},\"event_type
\":[\"my_custom_event\"],\"event_details\": {\"current_url\": [\"amazonaws.com\"]}}"
        }
    ]
}
```

延伸指標

如果您設定延伸指標，就可以執行下列其中一個或全部動作：

- 將默認的 CloudWatch RUM 指標發送到 CloudWatch 顯而易見的實驗中使用。只有PerformanceNavigationDuration、、PerformanceResourceDurationWebVitalsCumulativeLayoutShiftWe和WebVitalsLargestContentfulPaint指標可以傳送至「明顯」。

- 將任何預設 CloudWatch RUM 量度傳送至 CloudWatch 其他維度，以便量度提供更精細的檢視。例如，您可以查看使用者所使用之特定瀏覽器的特定指標，或是特定地理位置中使用者的指標。

如需有關預設 CloudWatch RUM 度量的詳細資訊，請參閱CloudWatch 您可以使用 CloudWatch RUM 收集的指標。

一個目的地可包含的延伸指標和自訂指標定義數量上限為 2000 個。對於您傳送至每個目的地的每個延伸或自訂指標，每個維度名稱和維度值的組合都會計為此限制的延伸指標。這也算作定價的 CloudWatch 自訂指標。

當您將擴充指標傳送至時 CloudWatch，您可以使用 CloudWatch RUM 主控台在其上建立 CloudWatch 警示。

延伸指標會以 CloudWatch 自訂指標的形式收費。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

應用程式監視器可傳送的所有指標名稱的延伸指標支援下列維度。這些指標名稱列於CloudWatch 您可以使用 CloudWatch RUM 收集的指標中。

- `BrowserName`

  維度值範例：`Chrome`、`Firefox`、`Chrome Headless`
- `CountryCode` 此維度會使用 ISO-3166 格式 (兩個字母的代碼)。

  維度值範例：`US`、`JP`、`DE`
- `DeviceType`

  維度值範例：`desktop`、`mobile`、`tablet`、`embedded`
- `FileType`

  維度值範例：`Image`、`Stylesheet`
- `OSName`

  維度值範例：`Linux`、`Windows, iOS`、`Android`
- `PageId`

使用主控台設定延伸指標

若要使用主控台傳送延伸指標 CloudWatch，請使用下列步驟。

要將擴展指標發送到 CloudWatch 顯而易見，您必須使用 AWS API 或 AWS CLI 代替控制台。有關使用 AWS API 將擴展指標發送到其中一個 CloudWatch 或顯而易見的信息，請參閱PutRumMetricsDestination和BatchCreateRumMetricDefinitions。

使用主控台設定應用程式監視器，並將 RUM 延伸指標傳送至 CloudWatch

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在瀏覽窗格中，選擇應用程式訊號，RUM。
3. 選擇 List view (清單檢視)，然後選擇要傳送指標的應用程式監視器名稱。
4. 選擇 Configuration (組態) 索引標籤，然後選擇 RUM extended metrics (RUM 延伸指標)。
5. 選擇 Send metrics (傳送指標)。
6. 選取要與其他維度一起傳送的一或多個指標名稱。
7. 選取一或多個要作為這些指標之維度的因素。作出選擇後，您選擇建立的延伸指標數量會顯示在 Number of extended metrics (延伸指標的數量) 中。

   此數字的計算方式是將選擇的指標名稱數目乘以您建立的不同維度數量。此數字代表需要您支付費用的自訂指標數量。如需有關 CloudWatch 定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

   a. 若要傳送以頁面 ID 作為維度的指標，請選擇 Browse for page ID (瀏覽頁面 ID)，然後選取要使用的頁面 ID。
   b. 若要傳送以裝置類型作為維度的指標，請選擇 Desktop devices (桌面裝置) 或 Mobile and tablets (行動裝置和平板電腦)。
   c. 若要傳送以作業系統作為維度的指標，請在 Operating system (作業系統) 下選取一或多個作業系統。
   d. 若要傳送以瀏覽器類型作為維度的指標，請在 Browsers (瀏覽器) 下選取一或多個瀏覽器。
   e. 若要傳送以地理位置作為維度的指標，請在 Locations (位置) 下選取一或多個位置。

      只有此應用程式監視器已回報指標的位置才會顯示在清單中供您選擇。
8. 完成選擇後，選擇 Send metrics (傳送指標)。
9. (選用) 在 Extended metrics (延伸指標) 清單中，若要建立監看其中一個指標的警示，請在該指標列中選擇 Create alarm (建立警示)。

   如需 CloudWatch 警示的一般資訊，請參閱 使用 Amazon CloudWatch 警報。如需在 CloudWatch RUM 擴充量度上設定警示的教學課程，請參閱教學課程：建立延伸指標並設定其警示。

停止傳送延伸指標

使用主控台停止傳送延伸指標

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在瀏覽窗格中，選擇應用程式訊號，RUM。
3. 選擇 List view (清單檢視)，然後選擇要傳送指標的應用程式監視器名稱。
4. 選擇 Configuration (組態) 索引標籤，然後選擇 RUM extended metrics (RUM 延伸指標)。
5. 選取要停止傳送的一或多個指標名稱和維度組合。接著選擇 Actions (動作)，Delete (刪除)。

教學課程：建立延伸指標並設定其警示

本教學課程示範如何設定要傳送至的延伸量度 CloudWatch，以及如何設定該量度的警示。在本教學課程中，您會建立追蹤 Chrome 瀏覽器 JavaScript 錯誤的指標。

設定此延伸指標並設定其警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在瀏覽窗格中，選擇應用程式訊號，RUM。
3. 選擇 List view (清單檢視)，然後選擇要傳送指標的應用程式監視器名稱。
4. 選擇 Configuration (組態) 索引標籤，然後選擇 RUM extended metrics (RUM 延伸指標)。
5. 選擇 Send metrics (傳送指標)。
6. 選取 [JS] ErrorCount。
7. 在 Browsers (瀏覽器) 下，選取 Chrome。

   JS ErrorCount 和 Chrome 的這種組合將向其發送一個擴展指標 CloudWatch。該指標僅針對使用 Chrome 瀏覽器的使用者工作階段計算 JavaScript 錯誤。測量結果名稱將為 JsErrorCount，維度名稱將為「瀏覽器」。
8. 選擇 Send metrics (傳送指標)。
9. 在「延伸量度」清單中，在 JsErrorCount「名稱」下方顯示的列中選擇「建立警示」，並在下方顯示「Chrome」BrowserName。
10. 在「指定量度和條件」下，確認量度名稱和BrowserName欄位已預先填入正確的值。
11. 在 Statistic (統計資料) 中，選取您要用於警示的統計資料。對於這種類型的計數指標，Average (平均值) 是一個不錯的選擇。
12. 在 Period (期間) 中，選取 5 minutes (5 分鐘)。
13. 在 Conditions (條件) 下，執行下列動作：

- 選擇 Static (靜態)。
- 選擇 Greater (大)，指定當錯誤數量大於您要指定的閾值時，警示應進入 ALARM 狀態。
- 在 than... (於…) 下方，輸入警示閾值的數字。如果 5 分鐘的期間內錯誤數量超過此數量時，警示會進入 ALARM 狀態。

14. (選用) 依預設，一旦 5 分鐘期間內的錯誤數量超過您設定的閾值數量，警示就會進入 ALARM 狀態。您也可以選擇將此設定變更為只有在連續多個 5 分鐘期間內超過此數字時，警示才進入 ALARM 狀態。

    若要這麼做，請選擇 Additional configuration (其他組態)，然後在 Datapoints to alarm (要警示的資料點) 中，指定在連續多少個 5 分鐘期間內錯誤次數超過閾值才會觸發警示。例如，您可以選取 2 個 (共 2 個)，只有在連續兩個 5 分鐘期間內超過閾值時才觸發警示；如果選取 2個 (共 3 個)，只有連續三個 5 分鐘期間內任何兩個期間內超過閾值，才會觸發警示。

    如需有關此類型警示評估的詳細資訊，請參閱評估警示。

15. 選擇下一步。
16. 在 Configure actions (設定動作) 中，指定警示進入 ALARM 狀態時應採取的動作。若要使用 Amazon SNS 接收通知，請執行以下操作：

    - 選擇 Add notification (新增通知)。
    - 選擇警示中。
    - 選取現有的 SNS 主題，或建立新主題。如果您建立新主題，請為其指定名稱，並至少向其新增一個電子郵件地址。

17. 選擇下一步。
18. 輸入名稱和選用的警示描述，然後選擇 Next (下一步)。
19. 檢閱詳細資訊，並選擇 Create alarm (建立警示)。

CloudWatchRUM 的資料保護和資料隱私

AWS 共同的責任模型適用於 Amazon CloudWatch RUM 中的資料保護和資料隱私權。如此模型所述， AWS 負責保護執行所有 AWS 雲端的全域基礎結構。您負責維護在此基礎設施上託管內容的控制權。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的AWS 共同責任模型和 GDPR 部落格文章。如需有關遵守 GDPR 規定的詳細資源，請參閱一般資料保護規範 (GDPR) 中心。

Amazon CloudWatch RUM 會根據您要收集的最終使用者資料輸入，產生程式碼片段供您內嵌到網站或 Web 應用程式程式碼中。由程式碼片段下載和設定的 Web 用戶端會使用 Cookie (或類似技術) 來協助您收集最終使用者資料。Cookie (或類似技術) 的使用受某些司法管轄區的資料隱私權法規約束。在使用 Amazon CloudWatch RUM 之前，我們強烈建議您根據適用法律評估合規義務，包括任何適用的法律要求，以提供合法適當的隱私權通知，並取得使用 Cookie 和處理 (包括收集) 最終使用者資料的任何必要同意。有關 Web 客戶端如何使用 Cookie（或類似技術）以及 Web 客戶端收集哪些最終用戶數據的更多信息，請參閱 CloudWatch RUM 網頁用戶端收集的資訊和CloudWatch RUM 網頁用戶端 Cookie（或類似技術）。

強烈建議您，絕對不要將最終使用者帳戶號碼、電子郵件地址或其他個人資訊等敏感的識別資訊放入自由格式欄位中。您輸入 Amazon CloudWatch RUM 或其他服務的任何資料都可能包含在診斷日誌中。

CloudWatch RUM 網頁用戶端 Cookie（或類似技術）

CloudWatch RUM Web 客戶端默認收集有關用戶會話的某些數據。您可以選擇啟用 Cookie，讓網頁用戶端收集使用者 ID 和工作階段 ID，這些 ID 會在整個頁面載入過程中持續存在。使用者 ID 是由 RUM 隨機產生。

如果啟用這些 Cookie，當您檢視此應用程式監控的 RUM 儀表板時，RUM 可以顯示下列類型的資料。

- 根據使用者 ID 彙總的資料，例如唯一使用者的數量，以及發生錯誤的不同使用者的數量。
- 根據工作階段 ID 彙總的資料，例如工作階段數量和發生錯誤的工作階段數量。
- 使用者旅程，是每個取樣使用者工作階段會包含的頁面順序。

> **Important**
>
> 如果您未啟用這些 Cookie (或類似技術)，Web 用戶端仍會記錄有關使用者工作階段的某些資訊，例如瀏覽器類型/版本、作業系統類型/版本、裝置類型等。收集這些資訊是為了提供彙總頁面特定的洞察，例如 Web 關鍵數值、頁面檢視以及發生錯誤的頁面。如需已記錄資料的詳細資訊，請參閱 CloudWatch RUM 網頁用戶端收集的資訊。

CloudWatch RUM 網頁用戶端收集的資訊

本節說明結構PutRumEvents描述，它定義了您可以使用 CloudWatch RUM 從使用者工作階段收集的資料結構。

PutRumEvents請求將具有以下字段的數據結構發送到 CloudWatch RUM。

- 此批次 RUM 事件的 ID
- 應用程式監控詳細資訊，其中包含下列各項：
  - 應用程式監控 ID
  - 監控的應用程式版本
- 使用者詳細資訊，其中包含下列各項。只有在應用程式監控已啟用 Cookie 時，才會收集此資訊。
  - 由 Web 用戶端產生的使用者 ID
  - 工作階段 ID
- 此批次中的 RUM 事件陣列。

RUM 事件結構描述

每個 RUM 事件的結構都包含下列欄位。

- 事件的 ID
- 時間戳記
- 事件類型
- 使用者代理程式
- 中繼資料
- RUM 事件詳細資訊

RUM 事件中繼資料

中繼資料包括頁面中繼資料、使用者代理程式中繼資料、地理位置中繼資料和網域中繼資料。

頁面中繼資料

頁面中繼資料包括下列項目：

- 頁面 ID
- 頁面標題
- 父頁面 ID。– 只有在應用程式監視器已啟用 Cookie 時，才會收集此資訊。
- 互動深度 – 只有在應用程式監視器已啟用 Cookie 時，才會收集此資訊。

- 頁面標籤 – 您可以將標籤新增至頁面事件，以將頁面分類在一起。如需詳細資訊，請參閱 使用頁面群組。

使用者代理程式中繼資料

使用者代理程式中繼資料包括下列項目：

- 瀏覽器語言
- 瀏覽器名稱
- 瀏覽器版本
- 作業系統名稱
- 作業系統版本
- 裝置類型
- 平台類型

地理位置中繼資料

地理位置中繼資料包括下列項目：

- 國家代碼
- 細分代碼

網域中繼資料

網域中繼資料包含 URL 網域。

RUM 事件詳細資訊

事件的詳細資訊會遵循下列其中一種結構描述類型，視事件類型而定。

工作階段啟動事件

此事件不包含任何欄位。只有在應用程式監控已啟用 Cookie 時，才會收集此資訊。

頁面檢視結構描述

Page view (頁面檢視) 事件包含下列屬性。您可以透過設定 Web 用戶端來停用頁面檢視集合。如需詳細資訊，請參閱 CloudWatch RUM 網路用戶端文件。

名稱	Type	描述
頁面 ID	字串	在應用程式中唯一代表此頁面的 ID。根據預設，這是 URL 路徑。
父頁面 ID	字串	當使用者導覽至當前頁面時，使用者所在頁面的 ID。只有在應用程式監控已啟用 Cookie 時，才會收集此資訊。
互動深度	字串	只有在應用程式監控已啟用 Cookie 時，才會收集此資訊。

JavaScript 錯誤綱要

JavaScript 代理程式產生的錯誤事件包含下列屬性。只有在您選取收集錯誤遙測時，Web 用戶端才會收集這些事件。

名稱	Type	描述
錯誤類型	字串	錯誤名稱 (如果存在)。如需詳細資訊，請參閱 Error.prototype.name。 某些瀏覽器可能不支援錯誤類型。
錯誤訊息	字串	錯誤的訊息。如需詳細資訊，請參閱 Error.prototype.message。如果錯誤欄位不存在，則這就是錯誤事件的訊息。如需詳細資訊，請參閱ErrorEvent。 在不同的瀏覽器中，錯誤消息可能不一致。
堆疊追蹤	字串	錯誤的堆疊追蹤 (如果存在) 會截斷為 150 個字元。如需詳細資訊，請參閱 Error.prototype.stack。 某些瀏覽器可能不支援堆疊追蹤。

DOM 事件結構描述

代理程式產生的文件物件模型 (DOM) 事件包含下列屬性。這些事件預設為無法收集。只有在您啟用互動遙測時，才會收集這些事件。如需詳細資訊，請參閱 CloudWatch RUM 網路用戶端文件。

名稱	Type	描述
事件	字串	DOM 事件的類型，例如點選、滾動或懸停。如需詳細資訊，請參閱事件參考。
Element	字串	DOM 元素類型
元素 ID	字串	如果產生事件的元素具有 ID，則此屬性會存放該 ID。如需詳細資訊，請參閱 Element.id。
CSSLocator	字串	用於識別 DOM 元素的 CSS 定位器。
InteractionId	字串	使用者與 UI 間之互動的唯一 ID。

導覽事件結構描述

只有在應用程式監控啟用效能遙測時，才會收集導覽事件。

導覽事件使用導覽計時層級 1 和導覽計時層級 2 API。並非所有瀏覽器都支援層級 2 API，因此這些較新的欄位是選用的。

> **Note**
>
> 時間戳記量度是以 DOM 為基礎HighResTimestamp。使用層級 2 API 時，所有的計時都預設與 `startTime` 相關。但對於層級 1，系統會從時間戳記指標減去 `navigationStart` 指標，以取得相對值。所有時間戳記值都以毫秒為單位。

導覽事件包含下列屬性。

名稱	Type	描述	備註
initiatorType	字串	表示起始效能事件的資源類型。	值："navigation" 層級 1："navigation"

名稱	Type	描述	備註
			層級 2:entryData .initiatorType

名稱	Type	描述	備註
navigationType	字串	表示導覽的類型。 此屬性不是必要項目。	值：值必須是下列其中一個： • navigate 是透過選擇連結、在瀏覽器的地址欄中輸入 URL、提交表單，或使用指令碼操作 (除了 reload 或 back_forward) 起始而啟動的導覽。 • reload 是透過瀏覽器重新加載操作或 location.reload() 的導覽。 • back_forward 是透過瀏覽器歷史記錄周遊操作的導覽。 • prerender 是由預先轉譯器提示

名稱	Type	描述	備註
			起始的導覽。如需詳細資訊，請參閱預先轉譯器。
startTime	Number	指示事件觸發的時間。	值：0 層級 1：entryData.navigationStart - entryData.navigationStart 層級 2：entryData.startTime

名稱	Type	描述	備註
unloadEventStart	Number	指示視窗中的上一個文件在擲出 unload 事件後開始卸載的時間。	值：如果沒有上一個文件，或者如果上一個文件或其中一個所需的重新引導不是相同的原點，則傳回的值為 0。 層級 1： `entryData.unloadEventStart > 0 ? entryData.unloadEventStart - entryData.navigationStart : 0` 級別 2：項目數據。unloadEventStart

名稱	Type	描述	備註
promptForUnload	Number	卸載文件所花費的時間。換言之，unloadEventStart 到 unloadEventEnd 之間的時間。UnloadEventEnd 表示卸載事件處理程式完成時的時刻，以毫秒為單位。	值：如果沒有上一個文件，或者如果上一個文件或其中一個所需的重新引導不是相同的原點，則傳回的值為 0。 級別 1：項目數據。unloadEventEnd -項目數據。unloadEventStart 級別 2：項目數據。unloadEventEnd -項目數據。unloadEventStart

名稱	Type	描述	備註
redirectCount	Number	代表目前瀏覽環境下最後一次非重新引導導覽之後重新引導數量的數字。 此屬性不是必要項目。	值：如果沒有重新引導，或者有任何重新引導與目的地文件的原點不相同，則傳回的值為 0。 層級 1：不可用 層級 2：entryData.redirectCount

名稱	Type	描述	備註
redirectStart	Number	第一個 HTTP 重新引導啟動的時間。	值：如果沒有重新引導，或者有任何重新引導與目的地文件的原點不相同，則傳回的值為 0。 層級 1： `entryData.redirectStart > 0 ? entryData.redirectStart - entryData.navigationStart : 0` 層級 2：entryData.redirectStart

名稱	Type	描述	備註
redirectTime	Number	HTTP 重新引導所花費的時間。這是 `redirectStart` 與 `redirectEnd` 之間的差異。	層級 1：：entryData.redirectEnd - entryData.redirectStart 層級 2：：entryData.redirectEnd - entryData.redirectStart
workerStart	Number	這是 `PerformanceResourceTiming` 介面的屬性。它標誌著工作者執行緒操作的開始。 此屬性不是必要項目。	值：如果服務工作者執行緒已經在執行中，或立即在啟動服務工作者執行緒之前執行，則此屬性會在分派 `FetchEvent` 前立即傳回時間。如果服務工作者未攔截資源，則傳回 0。 層級 1：不可用 層級 2：entryData.workerStart

名稱	Type	描述	備註
workerTime	Number	如果服務工作者未攔截資源，則這樣會傳回工作者執行緒操作所需的時間。 此屬性不是必要項目。	層級 1：不可用 層級 2： `entryData.workerStart > 0 ? entryData.fetchStart - entryData.workerStart : 0`
fetchStart	Number	瀏覽器準備就緒可以使用 HTTP 請求擷取文件的時間。這是在檢查任何應用程式快取之前。	層級 1： `: entryData.fetchStart > 0 ? entryData.fetchStart - entryData.navigationStart : 0` 層級 2：entryData.fetchStart

名稱	Type	描述	備註
domainLookupStart	Number	開始網域查詢的時間。	值：如果使用永久連線，或者如果資訊存放在快取或本機資源中，該值會與 `fetchStart` 相同。 層級 1： `entryData.domainLookupStart > 0 ? entryData.domainLookupStart - entryData.navigationStart : 0` 級別 2：項目數據。domainLookupStart

名稱	Type	描述	備註
dns	Number	網域查詢所需的時間。	值：如果快取資源和 DNS 記錄，則預期的值為 0。 級別 1：項目數據。domainLookupEnd -項目數據。domainLookupStart 級別 2：項目數據。domainLookupEnd -項目數據。domainLookupStart
nextHopProtocol	字串	代表用於擷取資源之網路通訊協定的字串。 此屬性不是必要項目。	層級 1：不可用 級別 2：項目數據。nextHopProtocol

名稱	Type	描述	備註
connectStart	Number	使用者代理程式立即開始建立與伺服器的連線以擷取文件之前的時間。	值：如果使用 RFC2616 永久連線，或從相關應用程式快取或本機資源擷取目前的文件，則此屬性會傳回 domainLookupEnd 值。 層級 1： `entryData.connectStart > 0 ? entryData.connectStart - entryData.navigationStart : 0` 層級 2：entryData.connectStart

名稱	Type	描述	備註
connect	Number	測量建立傳輸連線或執行 SSL 身分驗證所需的時間。它還包括當瀏覽器發出太多並行請求時所花費的阻塞時間。	層級 1：entryData.connectEnd - entryData.connectStart 層級 2：entryData.connectEnd - entryData.connectStart
secureConnectionStart	Number	如果當前頁面的 URL 結構描述是「https」，此屬性會傳回使用者代理程式立即啟動交握程序以保護當前連線之前的時間。如果不使用 HTTPS，則傳回 0。如需 URL 結構描述的詳細資訊，請參閱 URL 表示法。	公式：項目數據。secureConnectionStart

名稱	Type	描述	備註
tlsTime	Number	完成 SSL 握手所花費的時間。	層級 1： `entryData.secureConnectionStart > 0 ? entryData.connectEnd - entryData.secureConnectionStart : 0` 層級 2： `entryData.secureConnectionStart > 0 ? entryData.connectEnd - entryData.secureConnectionStart : 0`

名稱	Type	描述	備註
requestStart	Number	使用者代理程式立即開始從伺服器或從相關應用程式快取或從本機資源請求資源之前的時間。	層級 1： `: entryData.requestStart > 0 ? entryData.requestStart - entryData.navigationStart : 0` 層級 2：entryData.requestStart
timeToFirst位元組	Number	提出請求後，接收資訊之第一個位元組所花費的時間。此時間與 `startTime` 相關。	層級 1：entryData.responseStart - entryData.requestStart 層級 2：entryData.responseStart - entryData.requestStart

名稱	Type	描述	備註
responseStart	Number	使用者代理程式的 HTTP 剖析器立即從相關應用程式快取、或從本機資源，或從伺服器接收回應之第一個位元組的時間。	層級 1： `entryData.responseStart > 0 ? entryData.responseStart - entryData.navigationStart : 0` 層級 2：entryData.responseStart

名稱	Type	描述	備註
responseTime	字串	從相關應用程式快取、從本機資源或從伺服器以位元組形式接收完整回應所花費的時間。	層級 1： `entryData.responseStart > 0 ? entryData.responseEnd - entryData.responseStart : 0` 層級 2： `entryData.responseStart > 0 ? entryData.responseEnd - entryData.responseStart : 0`

名稱	Type	描述	備註
domIntera ctive	Number	當剖析器在主要文件上完成其工作，並構建 HTML DOM 的時間。這時，其 `Document. readyState` 變更為「交互式」並擲出相應的 `readystatechange` 事件。	層級 1： `entryData .domInter active > 0 ? entryData .domInter active - entryData .navigati onStart : 0` 層級 2：entryData .domInter active

名稱	Type	描述	備註
domContentLoadedEventStart	Number	表示等於用戶代理在當前文檔觸發 DOM ContentLoaded 事件之前的時間值。當初始 HTML 文檔已被完全加載和解析時，DOM ContentLoaded 事件觸發。此時，主要 HTML 文件已經完成解析，瀏覽器開始構建轉譯樹狀結構，並且仍然需要載入子資源。這不會等待樣式表、映像和子框架完成載入。	層級 1： `entryData.domContentLoadedEventStart > 0 ? entryData.domContentLoadedEventStart - entryData.navigationStart : 0` 級別 2：項目數據。domContentLoadedEventStart

名稱	Type	描述	備註
domContentLoaded	Number	轉譯樹狀結構的開始時間和結束時間會由 domContentLoadedEventStart 和 domContentLoadedEventEnd 標記。它可以讓 CloudWatch RUM 跟踪執行。此屬性是 domContentLoadedStart 與 domContentLoadedEnd 之間的差異。 在此期間，DOM 和 CSSOM 已準備就緒。此屬性會等待指令碼執行，但非同步指令碼和動態建立的指令碼除外。如果指令碼依賴於樣式表，domContentLoaded 也會在樣式表上等待。它不會等待映像。 Note domContentLoadedStart 和 domContentLoadedEnd 的實際值近似於 Google Chrome 網路面板中的 domContentLoaded 。它表明 HTML DOM + CSSOM 轉譯樹狀結構構造時間始於頁面加載程序。對於導覽指標，domContentLoaded 值表示開始和結束值之間的差異，這是僅下載子資源和轉譯樹狀結構構造所需的時間。	級別 2：項目數據。domContentLoadedEventEnd -項目數據。domContentLoadedEventStart 級別 2：項目數據。domContentLoadedEventEnd -項目數據。domContentLoadedEventStart

名稱	Type	描述	備註
domComplete	Number	瀏覽器立即將當前文件的當前文件整備程度設定為完成之前的時間。此時，子資源 (例如映像) 的載入已完成。這包括下載阻止內容（例如 CSS 和同步）所花費的時間 JavaScript。這近似於 Google Chrome 網路面板中的 `loadTime`。	層級 1： `entryData.domComplete > 0 ? entryData.domComplete - entryData.navigationStart : 0` 層級 2：entryData.domComplete
domProcessingTime	Number	回應與載入事件開始之間的總時間。	級別 1：項目數據。loadEventStart -輸入資料. 回應結束 級別 2：項目數據。loadEventStart -輸入資料. 回應結束

名稱	Type	描述	備註
loadEvent Start	Number	立即觸發當前文件之 `load` 事件的時間。	層級 1： `entryData .loadEvent tStart > 0 ? entryData .loadEvent tStart - entryData .navigati onStart : 0` 級別 2：項目數據。loadEvent Start
loadEvent Time	Number	`loadEventStart` 與 `loadEventEnd` 之間的差異。系統會在此期間觸發等待此載入事件的其他函數或邏輯。	級別 1：項目數據。loadEvent End -項目數據。loadEvent Start 級別 2：項目數據。loadEvent End -項目數據。loadEvent Start

名稱	Type	描述	備註
duration	字串	持續時間是頁面總載入時間。它會記錄下載主要頁面及其所有同步子資源的時間，以及轉譯頁面的時間。非同步資源 (例如指令碼) 會在稍後繼續下載。這是 `loadEventEnd` 和 `startTime` 屬性之間的差異。	級別 1：項目數據。loadEventEnd -入口數據導航開始 層級 2：entryData.duration
headerSize	Number	傳回 `transferSize` 與 `encodedBodySize` 之間的差異。 此屬性不是必要項目。	層級 1：不可用 級別 2：項目數據。傳輸大小-條目數據。encodedBodySize 級別 2：項目數據。傳輸大小-條目數據。encodedBodySize

名稱	Type	描述	備註
compressionRatio	Number	`encodedBodySize` 和 `decodedBodySize` 的比率。`encodedBodySize` 值是排除 HTTP 標頭之資源的壓縮大小。`decodedBodySize` 值是排除 HTTP 標頭之資源的解壓縮大小。 此屬性不是必要項目。	層級 1：不可用。 層級 2： `entryData.encodedBodySize > 0 ? entryData.decodedBodySize / entryData.encodedBodySize : 0`
navigationTimingLevel	Number	導覽計時 API 版本。	值：1 或 2

資源事件結構描述

只有在應用程式監控啟用效能遙測時，才會收集資源事件。

時間戳記度量是以 DOM HighResTimeStamp 類型定義為基礎。使用層級 2 API 時，所有的計時都預設與 `startTime` 相關。但對於層級 1 API，系統會從時間戳記指標減去 `navigationStart` 指標，以取得相對值。所有時間戳記值都以毫秒為單位。

代理程式產生的資源事件包含下列屬性。

名稱	Type	描述	備註
targetUrl	字串	傳回資源的 URL。	公式：entryData.name

名稱	Type	描述	備註
initiatorType	字串	表示起始效能資源事件的資源類型。	值："resource" 公式：entryData.initiatorType
duration	字串	傳回 `responseEnd` 和 `startTime` 屬性之間的差異。 此屬性不是必要項目。	公式：entryData.duration
transferSize	Number	傳回擷取資源的大小 (以八位元為單位)，包括回應標頭欄位和回應酬載主體。 此屬性不是必要項目。	公式：entryData.transferSize
fileType	字串	衍生自目標 URL 模式的擴充。	

最大的有內容繪製事件結構描述

最大的有內容繪製事件包含下列屬性。

只有在應用程式監控啟用效能遙測時，才會收集這些事件。

名稱	描述		
Value	如需詳細資訊，請參閱 Web 關鍵數值。		

第一個輸入延遲事件

第一個輸入延遲事件包含下列屬性。

只有在應用程式監控啟用效能遙測時，才會收集這些事件。

名稱	描述		
Value	如需詳細資訊，請參閱 Web 關鍵數值。		

累計版面配置移位事件

累計版面配置移位事件包含下列屬性。

只有在應用程式監控啟用效能遙測時，才會收集這些事件。

名稱	描述		
Value	如需詳細資訊，請參閱 Web 關鍵數值。		

HTTP 事件

HTTP 事件可以包含下列屬性。它將包含 `Response` 欄位或 `Error` 欄位，但不能同時包含。

只有在應用程式監控啟用 HTTP 遙測時，才會收集這些事件。

名稱	描述
請求	請求欄位包含下列項目： • `Method` 欄位，可以具有 GET、POST 等值。 • URL
回應	回應欄位包含下列項目： • 狀態，例如 2xx、4xx 或 5xx • 狀態文字
錯誤	錯誤欄位包含下列項目：

名稱	描述
	• Type • 訊息 • 檔案名稱 • 行號 • 欄號 • 堆疊追蹤

X-Ray 追蹤事件結構描述

只有在應用程式監控啟用 X-Ray 追蹤時，才會收集這些事件。

如需 X-Ray 追蹤事件結構描述的詳細資訊，請參閱 AWS X-Ray 區段文件。

單頁應用程式的路由變更時序

在傳統的多頁面應用程式中，當使用者請求載入新內容時，使用者實際上是從伺服器請求新的 HTML 網頁。因此， CloudWatch RUM 網路用戶端會使用一般效能 API 指標擷取載入時間。

但是，單頁 Web 應用程序使用 JavaScript 和 Ajax 更新界面，而無需從服務器加載新頁面。瀏覽器時序 API 不會記錄單頁更新，而是使用路由變更時序。

CloudWatch RUM 支援監視來自伺服器的整頁載入和單頁更新，但差異如下：

- 對於路由變更時序，沒有瀏覽器提供的指標，例如 `tlsTime`、`timeToFirstByte` 等。
- 對於路由變更時序，`initiatorType` 欄位將為 `route_change`。

CloudWatch RUM Web 用戶端會偵聽可能導致路由變更的使用者互動，並且當記錄這類使用者互動時，Web 用戶端會記錄時間戳記。然後，如果下列項目皆為 true，路由變更時序會開始：

- 瀏覽器歷史記錄 API (瀏覽器下一頁和上一頁按鈕除外) 用於執行路由變更。
- 路由變更偵測的時間與最近的使用者互動時間戳記之間的差異小於 1000 ms。這可避免資料扭曲。

然後，一旦路由變更時序開始，如果沒有正在進行的 AJAX 請求和 DOM 變動，則該時序即完成。然後，將使用最近完成活動的時間戳記作為完成時間戳記。

如果正在進行的 AJAX 請求或 DOM 變動超過 10 秒 (預設情況下)，路由變更時序將逾時。在這種情況下， CloudWatch RUM Web 客戶端將不再記錄此路由更改的時間。

因此，路由變更事件的持續時間計算方式如下：

```
(time of latest completed activity) - (latest user interaction timestamp)
```

管理使用 CloudWatch RUM 的應用程式

使用這些章節中的步驟來管理應用程式對 CloudWatch RUM 的使用。

如何找到我已經產生的程式碼片段？

若要尋找已針對應用程式產生的 CloudWatch RUM 程式碼片段，請依照下列步驟執行。

尋找我已經產生的程式碼片段

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在瀏覽窗格中，選擇應用程式訊號，RUM。
3. 選擇 List view (清單檢視)。
4. 在應用程式監視器名稱旁，選擇 [檢視] JavaScript。
5. 在「JavaScript 片段」窗格中，選擇「複製到剪貼簿」。

編輯應用程式。

若要變更應用程式監控的設定，請按照下列步驟操作。您可以變更應用程式監控名稱以外的任何設定。

若要編輯應用程式使用 CloudWatch RUM 的方式

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在瀏覽窗格中，選擇應用程式訊號，RUM。
3. 選擇 List view (清單檢視)。
4. 選擇應用程式名稱旁邊的按鈕，然後選擇 Actions (動作)、Edit (編輯)。
5. 變更應用程式名稱以外的任何設定。如需設定的詳細資訊，請參閱 步驟 2：建立應用程式監控。
6. 完成時，選擇儲存。

   變更設定會變更程式碼片段。您現在必須將更新的程式碼片段貼到您的應用程式中。
7. 建立程式 JavaScript 碼片段之後，請選擇「複製到剪貼簿」或「下載」，然後選擇「完成」。

若要開始使用新設定監控，請將程式碼片段插入應用程式中。在 <body> 元素或任何其他 <script> 標籤之前，將程式碼片段插入應用程式的 <head> 元素中。

停止使用 CloudWatch RUM 或刪除應用程式監視器

若要停止將 CloudWatch RUM 與應用程式搭配使用，請移除 RUM 從應用程式程式碼產生的程式碼片段。

若要刪除 RUM 應用程式監控，請依照下列步驟。

刪除應用程式監控

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在瀏覽窗格中，選擇應用程式訊號，RUM。
3. 選擇 List view (清單檢視)。
4. 選擇應用程式名稱旁邊的按鈕，然後選擇 Actions (動作)、Delete (刪除)。
5. 在確認方塊中，輸入 **Delete**，然後選擇 Delete (刪除)。
6. 如果您尚未這麼做，請從應用程式的程式碼中刪除 CloudWatch RUM 程式碼片段。

CloudWatch 朗姆酒配額

CloudWatch RUM 具有以下配額。

資源	預設配額
應用程式監控	每個帳戶 20 個 您可以要求增加配額。
RUM 擷取速率	每秒 50 個PutRumEvents要求 (TPS)。 您可以要求增加配額。

疑難排解 CloudWatch 朗

本節包含協助您疑難排解 CloudWatch RUM 的秘訣。

我的應用程式中沒有資料

首先，確認程式碼片段已正確插入您的應用程式。如需詳細資訊，請參閱 步驟 4：將程式碼片段插入應用程式。

如果這不是問題，則可能還沒有流量到您的應用程式中。透過使用與使用者採用之相同的方式存取應用程式，產生一些流量。

我的應用程式已停止記錄資料

您的應用程式可能已更新，現在不再包含 CloudWatch RUM 程式碼片段。檢查您的應用程式程式碼。

另一種可能性是有人可能已經更新了程式碼片段，但後來沒有將更新的程式碼片段插入到應用程式中。按照 如何找到我已經產生的程式碼片段？ 中的指示尋找當前的正確程式碼片段，並將其與貼到應用程式中的程式碼片段進行比較。

網路監控

本節中的主題說明 Amazon 網際 CloudWatch 網路監控器和 Amazon 網路監控器所提供的網路和網 CloudWatch 際 CloudWatch 網路監控功能。這些服務可協助您取得網路的作業能見度，以及託管應用程式的網際網路效能和可用性 AWS。

- Internet Monitor 會使用從其全球網路佔用量 AWS 擷取的連線資料，計算面向網際網路流量的效能和可用性基準。您可以查看流量模式和健全狀況事件的全域檢視，並輕鬆深入瞭解事件的相關資訊。您也可以針對影響應用程式用戶端的網際網路健全狀況事件取得警示。此外，您可以使用 Internet Monitor 提供的見解，通過使用 Amazon CloudFront 或通過不同的路由來探索客戶體驗的潛在改進 AWS 區域。
- Network Monitor 使用完全受管的代理程式方法，讓您追蹤並視覺化混合式網路連線的延遲和封包遺失。若要收集測量值並啟用 Network Monitor 為您的應用程式建立健全狀況事件警示，您可以建立從託管資源傳送 AWS 至內部部署目的地 IP 位址的探查。您不需要安裝其他代理程式即可監視網路效能。與 Internet Monitor 一樣，您可以設定警示和閾值、取得資訊以協助您快速疑難排解問題，然後採取行動改善使用者體驗。

主題

- 使用 Amazon CloudWatch 網絡監控
- 使用 Amazon CloudWatch 網絡監控

使用 Amazon CloudWatch 網絡監控

Amazon CloudWatch Internet Monitor 可讓您瞭解網際網路問題如何影響託管於您的應用程式與最終使用者之間的效能 AWS 和可用性。此程式可以將診斷網際網路問題所需的時間從數天縮短到幾分鐘。Internet Monitor 會使用從其全球網路佔用量 AWS 擷取的連線資料，計算面向網際網路流量的效能和可用性基準。這與用於監視 Internet 正常運行時間和可 AWS 用性的數據相同。網路監視器會使用這些度量作為基準，在應用程式執行的不同地理位置的最終使用者 (用戶端) 發生重大問題時，提醒您發現問題。

在 Amazon CloudWatch 主控台中，您可以查看流量模式和健康狀態事件的全域檢視，並輕鬆深入瞭解不同地理粒度 (位置) 的事件相關資訊。您可清楚看出影響，並精確地找出受影響的用戶端位置和網路 (ASN，通常為網際網路服務供應商 (ISP))。如果「網際網路監視器」判斷網際網路可用性或效能問題是由特定 ASN 或 AWS 網路造成，就會提供該資訊。

網路監視器的主要特點

- 網路監視器會提供洞見和建議，協助您改善最終使用者的體驗。您可以近乎即時地探索如何切換到使用其他服務，或透過不同 AWS 區域服務將流量重新路由至工作負載，以改善應用程式的預計延遲。
- 您可以使用網路監視器快速識別影響應用程式效能和可用性的因素，以追蹤並解決問題。
- Internet Monitor 將網際網路測量發佈到 CloudWatch 記錄檔和 CloudWatch 指標，以支援使用 CloudWatch 具有適用於您應用程式特定位置和 ASN (網際網路服務提供者) 健康資訊的工具。您也可以選擇將網際網路度量發佈至 Amazon S3。
- 網際網路監控器會將健康事件傳送至 Amazon，以 EventBridge 便您可以設定通知。如果問題是由 AWS 網路造成的，您也會自動收到 AWS Health Dashboard 通知，其中包含緩解問題所採取的步驟。 AWS

如何使用網路監視器

若要使用網際網路監視器，您可以建立監視器，並將應用程式的資源與它 (VPC、網路負載平衡器、CloudFront 散佈或 WorkSpaces 目錄) 建立關聯，以便讓 Internet Monitor 知道應用程式的網際網路對向流量在何處。然後，Internet Monitor 會發佈特定於城市網路 AWS 的網際網路測量結果，也就是用戶端位置和 ASN (通常是網際網路服務供應商或 ISP)，用戶端存取您的應用程式。如需詳細資訊，請參閱 如何 Amazon CloudWatch 互聯網監視器。若要開始使用網路監視器，請參閱使用控制台開始使用 Amazon CloudWatch 互聯網監視器。

目錄

- 支持 AWS 區域 Amazon CloudWatch 互聯網監視器
- Amazon CloudWatch 互聯網監控定價
- 組件和條款 Amazon CloudWatch 互聯網監視器
- 在 Amazon 互聯網監控全球 CloudWatch 互聯網天氣圖
- 如何 Amazon CloudWatch 互聯網監視器
- Amazon CloudWatch 互聯網監控示例用例
- 互聯網監控跨帳戶觀察
- 使用控制台開始使用 Amazon CloudWatch 互聯網監視器
- 使用 CLI 與 Amazon CloudWatch 互聯網監視器的例子
- 使用網路監視器儀表板監控和最佳化
- 使用 CloudWatch 工具和網際網路監視器查詢介面探索您的資料
- 創建報警與 Amazon CloudWatch 互聯網監控
- 使用 Amazon CloudWatch 互聯網監控 Amazon EventBridge
- 疑難排解 CloudWatch 記錄和指標存取錯誤

- 資料保護和資料隱私與 Amazon CloudWatch 網際網路監控
- Amazon CloudWatch 互聯網監視器的 Identity and Access Management
- 在 Amazon CloudWatch 互聯網監控配額

支持 AWS 區域 Amazon CloudWatch 互聯網監視器

本節列出了支持 Amazon CloudWatch 互聯網監視器的 AWS 區域 位置。如需目前支援網際網路監控器的區域清單 (包括選擇加入的區域)，請參閱 Amazon CloudWatch 網路監控器端點和 Amazon Web Services 一般參考中的配額。

請注意，Internet Monitor 只會將監視器的資料儲存 AWS 區域 在您建立監視器的位置，但監視器可以包含多個區域的資源。

區域名稱 (選擇支援)	區域
非洲 (開普敦)	af-south-1
亞太區域 (香港)	ap-east-1
亞太區域 (海德拉巴)	ap-south-2
亞太區域 (雅加達)	ap-southeast-3
亞太區域 (墨爾本)	ap-southeast-4
歐洲 (米蘭)	eu-south-1
歐洲 (西班牙)	eu-south-2
歐洲 (蘇黎世)	eu-central-2
Middle East (Bahrain)	me-south-1
中東 (阿拉伯聯合大公國)	me-central-1

區域名稱 (預設支援)	區域
美國東部 (俄亥俄)	us-east-2

區域名稱 (預設支援)	區域
美國東部 (維吉尼亞北部)	us-east-1
美國西部 (加利佛尼亞北部)	us-west-1
美國西部 (奧勒岡)	us-west-2
亞太區域 (孟買)	ap-south-1
亞太區域 (大阪)	ap-northeast-3
亞太區域 (首爾)	ap-northeast-2
亞太區域 (新加坡)	ap-southeast-1
亞太區域 (雪梨)	ap-southeast-2
亞太區域 (東京)	ap-northeast-1
加拿大 (中部)	ca-central-1
歐洲 (法蘭克福)	eu-central-1
歐洲 (愛爾蘭)	eu-west-1
歐洲 (倫敦)	eu-west-2
歐洲 (巴黎)	eu-west-3
歐洲 (斯德哥爾摩)	eu-north-1
南美洲 (聖保羅)	sa-east-1

Amazon CloudWatch 互聯網監控定價

使用 Amazon CloudWatch 網際網路監控器，無需預付費用或長期承諾。網路監視器的定價有兩個元件：每個受監控的資源費用和每個城市網路費用。城市網路是指用戶端存取應用程式資源的位置，以及用戶端存取資源的網路 (ASN (例如網際網路服務供應商 (ISP)))。請注意，您還需要針對日誌以及您建立的任何其他指標、儀表板、警示或深入解析向您收取標準 CloudWatch 價格的費用。

您可以在建立監視器時選擇要監控的流量百分比。若要協助控制帳單，您還可以針對想要監控的城市網路上限數量設定限制。您可以透過編輯監視器隨時更新要監控的流量百分比或城市網路的上限數量。其中已包含前 100 個城市網路 (每個帳戶的所有監視器中)。接著，您僅須針對監控的實際額外城市網路數量 (最多至上限數量) 付費。

您僅需支付您所監控的城市網路實際額外數量 (可達最大數量) 的費用，前 100 個城市網路不收取費用 (每個帳戶的所有監視器之間)。從您的每月帳單中扣除相當於 100 個城市網路成本的固定金額。

舉例來說，一家大型全球公司可以選擇監控其 100% 的面向網際網路流量，並針對使用單一資源的一個監視器，設定最多 50,000 個城市網路。假設流量達到 50,000 個城市網路，則此部分的帳單費用可能約為 2,700 美元/月。對於另一家公司，在較少的地理區域，一台顯示器與一個資源和 200 城市網絡，該法案的這一部分將是 13 美元/月左右。如需詳細資訊，請參閱 選擇城市網路上限數量。

您可以使用定價計算器嘗試不同的選項。若要探索定價選項，請在 [定價計算器 CloudWatch] 頁面上，向下捲動至 [網際網路監控]。

如需有關網際網路監控和 CloudWatch 定價的詳細資訊，請參閱 Amazon CloudWatch 定價頁面。

組件和條款 Amazon CloudWatch 互聯網監視器

Amazon CloudWatch 互聯網監控器使用或引用以下內容。

監控

監視器會包含您想要檢視相關網際網路效能和可用性度量，以及取得相關運作狀態事件提醒的單一應用程式資源。當您建立應用程式的監視器時，可以新增應用程式的資源，以針對要監視的網路監視器定義城市 (位置)。網路監視器會使用您新增之應用程式資源的流量模式，以便其發佈與應用程式通訊之位置和 ASN [通常為網際網路服務供應商 (ISP)] 特定的網際網路效能和可用性度量。換句話說，新增的資源會建立您想要網路監視器進行監視與發佈相關度量的城市網路範圍。

已新增至監視的資源 (「受監控的資源」)

您新增至監視器的資源是網際網路監視器中的「受監視的資源」。也就是：

- 您在區域中新增的每個 VPC 皆為受監控的資源。當您新增 VPC 時，Internet 監控器會監控 VPC 中任何網際網路對向應用程式的流量，例如，在 Amazon EC2 執行個體上託管的應用程式、Network Load Balancer 後面或容器。 AWS Fargate
- 您在區域中新增的每個 Network Load Balancer 皆為受監控資源。
- 您在區域中新增的每個 WorkSpaces 目錄都是受監控的資源。
- 您新增的每個 CloudFront 散佈都是受監控的資源。

自治系統編號 (ASN)

在網路監視器中，ASN 通常是指網際網路服務供應商 (ISP)，例如 Verizon 或 Comcast。ASN 為用戶端用於存取網際網路應用程式的網路供應商。自治系統 (AS) 是一組可路由網際網路通訊協定 (IP) 字首，屬於由同一個組織管理、控制和監督的網路或網路集合。

城市網路 (位置和 ASN)

城市網路是用戶端存取應用程式資源的位置 (例如城市)，以及用戶端存取資源的 ASN (通常是網際網路服務供應商 (ISP)。若要協助控制您的帳單，您可以設定網際網路監視器監視每個監視器的城市網路數目上限。您僅須針對監控的實際城市網路數量 (最多至上限數量) 付費。如需詳細資訊，請參閱選擇城市網路上限數量。

網際網路測量結果

網際網路監視器會針對您帳戶中的前 500 個城市網路 (用戶端位置和 ASN，通常是網際網路服務提供者或 ISP)，每五分鐘將網際網路測量發佈到 CloudWatch 記錄檔中。這些度量會量化應用程式的效能分數、可用性分數、傳輸的位元組 (輸入的位元組和輸出的位元組)，以及應用程式之城市網路的往返時間。這些是針對 VPC、網路負載平衡器、 CloudFront 分發或目錄特定的城市網路測量。WorkSpaces 您也可以選擇將所有受監控之城市網路 (最多 500,000 個城市網路服務限制) 的網際網路度量和事件發佈至 Amazon S3 儲存貯體。

指標

Internet Monitor 會產生指標的彙總 CloudWatch 指標、應用程式的全域流量以及每個指標的全域流量 AWS 區域。如需詳細資訊，請參閱 使用 CloudWatch 指標與 Amazon CloudWatch 互聯網監控。

運作狀態事件

網路監視器會建立運作狀態事件，以提醒您注意影響應用程式的特定問題。網路監視器會偵測網際網路問題，例如世界各地發生的網路延遲增加情況。然後，它會使用來自全 AWS 球基礎架構足跡的歷史網際網路測量，計算目前問題對您應用程式的影響，並建立健康狀態事件。網路監視器預設會根據整體影響和局部影響閾值，建立運作狀態事件。如需設定閾值的詳細資訊，請參閱變更運作狀態事件閾值。

每個運作狀態事件皆包含受影響之城市網路的相關資訊。您可以在 CloudWatch 主控台中檢視健全狀況事件，或使用 AWS SDK 或 AWS CLI 網際網路監視器 API 動作來檢視健全狀況事件。網際網路監控器也會傳 EventBridge 送 Amazon 健康事件的通知。如需詳細資訊，請參閱網路監視器建立和解決運作狀態事件的時機。

互聯網事件

Internet Monitor 會在所有 AWS 客戶都可以使用的網際網路天氣圖上顯示有關最近全球健康事件 (稱為網際網路事件) 的資訊。您無需在互聯網監視器中創建監視器即可查看互聯網天氣圖。與健康事件不同，網際網路事件並不是針對個別客戶或其應用程式流量的特定。如需詳細資訊，請參閱 在 Amazon 互聯網監控全球 CloudWatch 互聯網天氣圖。

閾值

網路監視器會根據整體閾值和局部閾值，建立運作狀態事件。您可以變更預設閾值並設定其他選項，例如關閉局部閾值。如需設定閾值的詳細資訊，請參閱變更運作狀態事件閾值。

效能和可用性分數

透過分析 AWS 收集的資料，網際網路監視器可以偵測應用程式的效能和可用性何時下降，相較於 Internet Monitor 所計算的估計基準。為了方便您查看上述下降情況，網路監視器會以分數形式向您回報資訊。效能分數表示未發現效能下降的預估流量百分比。同樣地，可用性分數表示未發現可用性下降的預估流量百分比。如需詳細資訊，請參閱如何 AWS 計算效能和可用性分數。

傳輸的位元組和受監控的傳輸位元組

傳輸的位元組是指用戶端存取應用程式的應用程式與城市網路 (亦即位置 AWS 和 ASN，通常是網際網路服務提供者) 之間的入口和出口流量總位元組數。受監控的傳輸位元組為類似指標，但僅包含受監控之流量的位元組。

往返時間

往返時間 (RTT) 是指從用戶端使用者發出請求至回應傳回至該使用者所需的時間。彙總不同用戶端位置 (城市或其他地理位置) 的 RTT 時，值會根據每個用戶端位置驅動的應用程式流量多寡加權。

在 Amazon 互聯網監控全球 CloudWatch 互聯網天氣圖

Amazon CloudWatch 互聯網監視器顯示一個全球互聯網天氣圖，可供所有 AWS 客戶使用。若要檢視地圖，請在 Amazon 主 CloudWatch 控台中導覽至網際網路監控器。

該地圖強調了世界各地影響 AWS 客戶的網際網路事件 (「中斷」)，以及發生效能或可用性問題的特定城市和網路 (ASN，通常是網際網路服務提供者)。互聯網天氣圖包括過去 24 小時的互聯網事件。

您無需在互聯網監視器中創建監視器即可查看互聯網天氣圖。與網際網路監視器中的健康事件不同，網際網路事件並不是針對個別客戶或其應用程式流量的

在互聯網天氣地圖上，您可以選擇互聯網事件以了解有關該事件的詳細信息。對於網際網路事件，您可以查看開始時間、結束時間 (如果事件結束)、目前狀態 (「使用中」或「已解決」)，以及中斷問題類型

(「可用性」或「效能」)。若要深入瞭解如何建立網際網路天氣圖以及包含哪些內容，請參閱全球網際網路天氣地圖常見問題集。

若要檢視並使用特定於應用程式流量和用戶端位置的詳細資訊，您可以輕鬆地在 Internet Monitor 中為您的應用程式設定監視器。然後，您會看到效能和可用性模式和事件、目前和歷史，以及取得僅針對您的應用程式使用量和客戶量身打造的健全狀況事件警示。互聯網天氣地圖為您提供了整體視圖，而特定的監視器將信息過濾為與您的應用程序相關的測量和詳細信息。透過監視器，您還可以探索歷史指標，並取得改善應用程式用戶端體驗的建議。如需進一步了解，請參閱使用控制台開始使用 Amazon CloudWatch 互聯網監視器。

如何 Amazon CloudWatch 互聯網監視器

本節提供有關 Amazon CloudWatch 互聯網監視器如何工作的信息。這包括如何 AWS 收集資料的說明，以協助偵測網際網路上的連線問題，以及如何計算效能和可用性分數。

內容

- 互聯網監視器如何專注於您的應用程序流量足跡
- 如何測 AWS 量連接問題並計算測量
- 網路監視器的地理位置準確度
- 網路監視器建立和解決運作狀態事件的時機
- 運作狀態事件回報時機
- 網路監視器如何使用 IPv4 和 IPv6 流量
- 互聯網監控器如何選擇城市網絡的子集包括
- 如何創建全球互聯網天氣圖（常見問題）

互聯網監視器如何專注於您的應用程序流量足跡

Internet Monitor 專注於監控 AWS 資源使用者存取的網際網路子集，而不是像其他工具一樣，從全球每個區域廣泛監控您的網站。這也是一種具有成本效益的解決方案，大型公司和小型公司都負擔得起。

Internet Monitor 使用相同的功能強大的探查和問題偵測演算法，這些演算法 AWS 會利用內部優勢，並在 Internet Monitor 中建立健康狀態事件來警示您影響應用程式的連線問題。接著，網路監視器會依據應用程式資源，藉由覆蓋您作用中檢視者建立的流量設定檔，讓您存取產生的效能和可用性地圖。

網路監視器僅會使用這些資訊向您顯示相關事件 (即具有作用中檢視者之位置所發生的事件)，以及這些事件對整體檢視者數量的影響。因此，事件有多大的影響，百分比來說，是基於您的全球總流量。

網際網路監視器發佈到 CloudWatch 記錄網際網路測量每五分鐘的前 500 個城市網路 (用戶端位置和 ASN，通常是網際網路服務提供者或 ISP) 將流量傳送到每個監視器。您也可以選擇將所有受監控之城市網路 (最多 500,000 個城市網路服務限制) 的網際網路度量發佈至 Amazon S3 儲存貯體。如需詳細資訊，請參閱 在 Amazon 網際網路監視器中將 CloudWatch 網際網路測量發佈。

網路監視器的優點如下：

- 使用網路監視器不會給 AWS上託管的應用程式增加額外負載或成本。
- 您無需在用戶端資源或應用程式中納入效能測量代碼。
- 您可以了解應用程式連線之網際網路的效能和可用性，包括「最後一哩」相關資訊。

請注意，因為網際網路監視器會根據您的 AWS 資源建立度量，因此 Internet 監視器只會建立特定於您的應用程式流量的事件。一般來說，不會回報全域網際網路問題。此外，當服務位置是時 AWS 區域，發出的測量和事件是設計用來表示區域層級的連線能力，而不會精確地呈現使用者位置與可用區域之間的連線。

如何測 AWS 量連接問題並計算測量

Amazon CloudWatch 網際網路監控器透過自主系統編號 (ASN) (通常是網際網路服務供應商 (ISP)，在不同的用戶端位置使用不同 AWS 區域 和 Amazon 存放 CloudFront 點 (PoP) 之間的網際網路連線資料。這是 AWS 運營商每天在內部使用的連接數據，用於主動檢測全球互聯網上的連接問題。

對於每個人 AWS 區域，我們都知道互聯網的哪些部分與該地區進行通信，並執行以下操作：

- 我們會積極監控該部分的網際網路，30 天為一個時段，滾動監控。
- 我們會同時使用網路和高階通訊協定探查工具，包括輸入和輸出探查。

AWS 具有主動和被動探查，可以測量從每個服務到整個 Internet 的第 90 百分位數和可達性（ 可用性 ）的延遲（ AWS 區域 性能 ）。 CloudFront 監控服務與客戶位置之間連線中的異常模式，然後將其報告為警示給客戶。

計算可用性和 RTT

往返時間 (RTT) 是指從使用者發出請求至回應傳回至該使用者所需的時間。不同最終使用者位置的往返時間彙總值會根據每個最終使用者位置驅動的流量大小加權。

例如，若有兩個最終使用者位置，一個提供 90% 的流量 (RTT 為 5 毫秒)，另一個提供 10% 的流量 (RTT 為 10 毫秒)，則 RTT 彙總值結果會是 5.5 毫秒 (5 毫秒 * 0.9 + 10 毫秒 * 0.1)。

請注意，測量最後一哩延遲的資源有所不同。對於網際網路監視器延遲度量，VPC、網路負載平衡器和 WorkSpaces 目錄不包含最後一哩延遲。

計算效能與可用性分數

AWS 擁有關於 AWS 服務與不同城市網絡（地點和 ASN）之間的互聯網性能和可用性的大量歷史數據。網路監視器可藉由對這些資料進行統計分析，偵測應用程式效能和可用性下降的時機，並將效能和可用性與已計算的預估基準比較。為了方便您查看上述下降情況，會以運作狀態分數形式 (效能分數和可用性分數) 向您回報該資訊。

我們會以不同精細度計算運作狀態分數。我們會以最精細的程度運算地理區域 (例如城市或都會區) 和 ASN (「城市網路」) 的運作狀態分數。我們也會將監視器中應用程式的個別運作狀態分數，彙總為整體運作狀態分數。如果您在未篩選任何特定地理區域或服務提供者的情況下，檢視效能或可用性分數，網路監視器會提供整體運作狀態分數。

整體運作狀態分數會涵蓋指定時段內的整個應用程式情況。整個應用程式城市網路配對的效能或可用性分數達到或低於效能或可用性的相應運作狀態事件閾值時，網路監視器會觸發運作狀態事件。整體效能和可用性閾值預設都是 95%。網路監視器也會根據您設定的值，依局部閾值 (如果預設啟用該選項)建立運作狀態事件。若要進一步了解設定運作狀態事件閾值，請參閱變更運作狀態事件閾值。

瀏覽監視器和日誌檔案中的資訊以調查問題並進一步了解時，可以依特定城市 (位置)、網路 (ASN 或網際網路服務供應商) 或兩者來篩選。您便可使用篩選條件，依所選篩選條件，查看不同城市、ASN 或城市網路配對的運作狀態分數。

- 可用性分數表示未發現可用性下降的預估流量百分比。網路監視器會根據監控到的總流量和可用性指標測量結果，預估經歷可用性下降的流量百分比。例如，最終使用者/服務位置配對的可用性分數為 99%，這表示該對經歷可用性下降的流量為 1%。
- 效能分數表示未發現效能下降的流量百分比。例如，最終使用者/服務位置配對的效能分數為 99%，這表示該對經歷效能下降的流量為 1%。

計算 TTFB 和 RTT（延遲）

第一個字節的時間（TTFB）是指客戶端發出請求到從服務器接收信息的第一個字節之間的時間。 AWS TTFB 的計算會測量從 Amazon EC2 或 Amazon CloudFront 到網際網路監控器測量節點所經過的時間 (包括節點的最後一英里)。也就是說，網際網路監控器會測量從使用者到 Amazon EC2 區域 (適用於 EC2) 的時間，以及從使用者到 TTFB CloudFront 的時間。CloudFront

針對往返時間 (RTT)，網路監視器包含從城市網路 (即用戶端位置和 ASN，通常是網際網路服務供應商)，如公共 IP 地址所映射，到 AWS 區域的時間。這表示網路監視器無法在最後一哩掌握從閘道或 VPN 後存取網際網路的使用者。

請注意，測量最後一哩延遲的資源有所不同。對於網際網路監視器延遲度量，VPC、網路負載平衡器和 WorkSpaces 目錄不包含最後一哩延遲。

Internet Monitor 會在 CloudWatch 儀表板上 [流量見解] 索引標籤的 [流量最佳化建議] 區段中包含平均 TTFB 資訊，以協助您評估應用程式的不同設定選項，以改善效能。

區域和可用區域測量和彙總

雖然網際網路監視器彙總了區域層級的測量和共用影響，但它會在可用區域 (AZ) 層級計算影響。這意味著，如果某個事件只有一個 AZ 受到影響，並且大部分流量流經該 AZ，您確實會看到流量的影響。但是，對於相同的事件，如果您的應用程式流量未經過受影響的 AZ，您就不會看到影響。

請注意，這僅適用於不是 WorkSpaces 目錄的資源。 WorkSpaces 目錄僅在區域層級上進行測量。

網路監視器的地理位置準確度

對於位置信息，互聯網監視器使用 IP 地理位置數據提供。MaxMind網際網路監視器測量中位置資訊的準確性取決於資料 MaxMind的準確性。

請注意，Metro液位測量對於美國以外的地點可能不準確。

網路監視器建立和解決運作狀態事件的時機

網路監視器會根據目前設定的閾值，為您監控的應用程式流量建立和關閉運作狀態事件。網路監視器有預設閾值組態，您也可以設定自己的閾值組態。網路監視器會判斷連線問題對您應用程式造成的整體影響，以及對應用程式有用戶端之局部區域的影響，並在超過閾值時建立運作狀態事件。

Internet Monitor 會根據有關網際網路效能和網路流量可用性的歷史資料，計算連線問題對用戶端位置的影響 AWS。其根據用戶端使用您應用程式的 ASN 和服務地理位置：受影響的城市網路配對，套用與您應用程式相關的資訊。位置是根據您新增至監視器的資源決定。網路監視器就會使用統計分析，偵測效能和可用性下降的時機，這會影響應用程式用戶端體驗。

網路監視器計算的效能和可用性分數會以未發現下降的流量百分比來表示。影響則與此相反：表示問題對客戶的最終使用者造成的問題嚴重程度。例如，若全域可用性下降是 93%，則對應的影響將會是 7%。

應用程式城市網路配對的效能或可用性分數全域達到或低於效能或可用性的相應運作狀態事件閾值時，會使網路監視器產生運作狀態事件。效能和可用性閾值預設都是 95%。符合或低於閾值的值是累積計算，因此可能代表幾個較小事件合起來達到閾值百分比，或者單一事件達到或低於閾值層級。

只要觸發事件的效能或可用性分數符合或低於相應整體影響運作狀態事件閾值百分比，運作狀態事件就會保持未解決狀態。觸發事件的分數或合併分數上升到超過閾值時，網路監視器會解決運作狀態事件。

網路監視器也會根據局部閾值和問題影響的整體流量百分比，建立運作狀態事件。您可以設定局部閾值的選項，或一併關閉局部閾值。

若要進一步了解設定運作狀態事件閾值，請參閱變更運作狀態事件閾值。

運作狀態事件回報時機

網路監視器會使用彙總器來收集有關網際網路問題的所有訊號，以在幾分鐘內於監視器中建立運作狀態事件。

如果可能的話，網際網路監視器會分析健全狀況事件的來源，以判斷它是由 ASN 造成 AWS 還是由 ASN 所造成。在事件解決後，運作狀態事件分析會繼續進行。網路監視器最多可以使用新資訊更新事件一小時。

網路監視器如何使用 IPv4 和 IPv6 流量

如果透過任何 IP 系列 (IPv4 或 IPv6) 向該網路提供流量，則網路監視器僅透過 IPv4 測量網路的運作狀態，並向您顯示運作狀態事件，以及可用性和效能指標。如果您為來自雙堆疊資源 (例如雙堆疊 CloudFront 發佈) 的流量提供服務，Internet Monitor 會引發健全狀況事件，並且只有當 IPv4 流量與 IPv6 流量有相同的問題時，才會顯示效能分數或可用性分數下降。

請注意，網路監視器的總傳入位元組和傳出位元組指標可準確反映所有網際網路流量 (IPv4 和 IPv6)。

互聯網監控器如何選擇城市網絡的子集包括

當您設定監視器監視的城市網路數目上限，或選擇要監控的流量百分比時，Internet Monitor 會根據最近的最高流量來選擇要包含 (監控) 的城市網路。

例如，如果您將城市網路上限設定為 100，則 Internet Monitor 會根據您最近一小時的應用程式流量監控 (最多) 100 個城市網路。具體來說，互聯網監視器監控最近一小時窗口之前最近一小時窗口中流量最多的前 100 個城市網絡。

為了說明這一點，假設當前時間是 2:30 PM。在這個案例中，您在監視器中看到的流量是在下午 1:00 和 2:00 PM 之間擷取，而網際網路監視器用來判斷前 100 個城市網路的流量測量是在 12:00 PM 和 1:00 PM 之間擷取。

如何創建全球互聯網天氣圖（常見問題）

Amazon CloudWatch 網際網路監控器網際網路天氣圖可透過網際網路監控主控台提供給所有已驗證的 AWS 客戶。本節包括有關如何創建互聯網天氣圖以及如何使用它的詳細信息。

什麼是互聯網監控互聯網天氣圖？

互聯網天氣圖提供了世界各地的互聯網問題的可視化表示。它突出顯示受影響的客戶位置，即城市加 ASN（通常是互聯網服務提供商）。該地圖顯示了可用性和效能問題的組合，這些問題最近影響了客戶在全球頂級用戶端位置和 AWS 服務的互聯網體驗。

地圖的資料來自哪裡？

該數據是基於互聯網的主動和被動探測的組合。若要深入瞭解網際網路監視器如何測量資料，您可以閱讀「如何 AWS 衡量連線問題」一節。

地圖多久更新一次？

互聯網天氣地圖每 15 分鐘更新一次。

追蹤哪些網路是否有中斷？

AWS 跟踪世界各地的網絡，這些網絡代表客戶用於進行互聯網連接的重要 IP 前綴。 AWS我們會將中斷範圍限制到用戶端位置，這些用戶端位置是傳送至網路和從網路接收的流量的最佳發言者。 AWS

是什麼決定地圖上是否包含互聯網事件？

以下是我們用來判斷網際網路天氣圖中是否包含網際網路事件的一些高階標準：

- AWS 偵測是否存在可用性或效能事件。
- 例如，如果活動持續時間不到 5 分鐘，我們將忽略它。
- 然後，如果事件位於被歸類為最佳發言者的用戶端位置，則會視為中斷。

互聯網天氣圖使用哪些閾值？

確定中斷的閾值對於互聯網天氣圖來說並不是靜態的。互聯網監視器確定什麼構成基於檢測從預期值偏差的事件。您可以檢閱 Internet Monitor 如何判斷何時為您使用服務建立的監視器建立健全狀況事件，以深入瞭解其運作方式。當您建立監視器時，網際網路監視器會產生特定於您自己應用程式流量的網際網路流量健全狀況測量。網際網路監視器也會針對影響應用程式網際網路流量的問題提醒您健康狀況事件。

我可以用這些數據做什麼？

互聯網天氣圖提供了過去 24 小時內在世界各地發生的關鍵互聯網事件的快速摘要。它可以幫助您獲得互聯網監控體驗的味道，而無需板載自己的互聯網流量到互聯網監視器。若要充分利用網際網路監視功能的潛力， AWS 並針對託管的應用程式和服務進行個人化 AWS，您可以在網際網路監視器中建立監視器。

當您建立監視器時，您可以啟用 Internet Monitor 來識別會影響應用程式用戶端的特定網際網路路徑，而且您可以存取可協助您改善用戶端體驗的功能和功能。您也會主動收到特別影響應用程式流量和用戶端的新網際網路問題的通知。

如何取得更多有關活動的詳細資訊？

按一下地圖上的中斷以查看詳細資料，包括事件開始和結束的時間、受影響的城市和 ASN，以及問題的類型 (也就是效能問題或可用性問題)。

若要取得有關事件的詳細資訊，並取得應用程式流量的自訂度量，請在 Internet Monitor 中建立監視器。

Amazon CloudWatch 互聯網監控示例用例

在本節中，我們會說明幾個具體範例，並附上提供詳細資訊的部落格文章連結。這些範例說明如何使用 Amazon CloudWatch 網際網路監控器的功能來協助您監控應用程式並改善使用者體驗。

設定警示並決定要採取的動作

您可以使用網路監視器深入了解一段時間內的平均網際網路效能指標，以及按城市網路 (用戶端位置和 ASN (通常為網際網路服務供應商)) 分類的運作狀態事件。使用網際網路監控器，您可以針對 Amazon 虛擬私有雲端 (VPC)、網路負載平衡器、Amazon 或 Amazon 上託管的應用程式，識別影響最終使用者體驗的事件。 WorkSpaces CloudFront

建立監視器之後，您有幾個選項，讓您收到網路監視器運作狀態事件的警示。其中包括根據使用事件指標或 Amazon EventBridge 規則篩選健康事件的 CloudWatch 警示通知。您可以根據警示 (例如，通知或 CloudWatch 日誌群組的更新) 選擇不同的 AWS SMS 通知或動作選項。

要查看具有詳細指導的示例，請參閱以下博客文章：Amazon CloudWatch 互聯網監視器介紹。

找出延遲問題並改善 TTFB 以提升多人遊戲體驗

使用網路監視器，協助您快速找出跨國雲端遊戲應用程式遊戲玩家遭遇延遲問題的部分，並提供改善效能的洞察。透過找出多數玩家目前遭遇最慢的第一個位元組時間 (TTFB) 的部分，即可了解如何改善延遲，從而讓最多玩家群滿意。

現在，當您準備好為遊戲部署下一個 EC2 服務器時，請選擇 Internet Monitor 建議的將在具有高延遲和大量玩家的區域中降低 TTFB。 AWS 區域

如需針對此使用案例設定和使用網際網路監視器的詳細資訊，請參閱下列部落格文章：使用 Amazon CloudWatch 網際網路監視器獲得更好的遊戲體驗。

互聯網監控跨帳戶觀察

使用 Internet Monitor 跨帳戶可觀察性，您可以監視跨越多個 AWS 帳戶的應用程序在一個單一的。 AWS 區域

您可以使用 Amazon CloudWatch 觀察性存取管理員將一或多個 AWS 帳戶設定為監控帳戶。您可以在監控帳戶中建立接收器，讓監控帳戶能夠檢視來源帳戶中的資料。接收器是代表監控帳戶中連接點的資源。對於網際網路監視器，資源連接點是監視器。可以使用此接收器建立從來源帳戶到監控帳戶的連結。如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。

必要的資源

如需 CloudWatch 應用程式深入解析跨帳戶可觀察性的正確功能，請確定下列遙測類型是透過 CloudWatch 觀察性存取管理員共用。

- 監視器在互聯網監控
- Amazon 的指標 CloudWatch
- Amazon CloudWatch 日誌中的日誌群組

使用控制台開始使用 Amazon CloudWatch 互聯網監視器

若要開始使用 Amazon CloudWatch 網際網路監視器，您必須新增應用程式使用的 AWS 資源並設定數個組態選項，在網際網路監視器中建立監視器。本章提供在主控台中新增監視器的程序。還包括一節詳細說明網路監視器中的資源，以及其他小節，說明您可以或必須為監視器所設定不同選項的限制。

目錄

- 使用控制台在 Amazon CloudWatch 互聯網監控器中創建監視器
- 將資源新增至您的監視器
- 選擇要監控的應用程式流量百分比
- 選擇城市網路上限數量
- 在 Amazon 網際網路監視器中將 CloudWatch 網際網路測量發佈

- 使用網路監視器監控
- 編輯或刪除網路監視器的監視器
- 使用 Amazon VPC 添加或創建 Amazon CloudWatch 互聯網監視器監視器
- 添加或創建一個 Amazon CloudWatch 互聯網監視器監視器 CloudFront

使用控制台在 Amazon CloudWatch 互聯網監控器中創建監視器

您可以在 Amazon CloudWatch 網際網路監視器中為應用程式建立監視器，方法是新增監視器使用的 AWS 資源，然後設定數個組態選項。您新增的資源，Amazon 虛擬私有雲端 (VPC)、網路負載平衡器 (NLB)、 CloudFront 分佈或 WorkSpaces 目錄，會提供網際網路監控器的資訊，以對應您的應用程式的網際網路流量資訊。建立監視器之後，請等待 15-30 分鐘，以產生應用程式使用位置的特定流量設定檔。然後，您可以使用 Internet Monitor 監視器或其他工具，以視覺化方式呈現並探索有關用戶端使用情況的效能和可用性。這些工具會使用應用程式流量的測量結果 (由監視器收集並發佈至 CloudWatch 記錄)，為您提供深入分析資訊。

一般情況下，這是在網路監視器中針對一個應用程式建立一個監視器的最簡單方式。在相同的監視器中，您可以依據不同的位置和 ASN (通常是網際網路服務供應商) 或其他資訊，搜尋和排序網路監視器日誌檔案中的度量和指標。例如，您不需要為不同區域的應用程序建立獨立監視器。

此處的步驟將引導您透過主控台設定監視器。若要查看 AWS Command Line Interface 與網際網路監視器 API 動作搭配使用的範例，若要建立監視、檢視事件等，請參閱使用 CLI 與 Amazon CloudWatch 互聯網監視器的例子。

使用主控台建立監視器

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在左側導覽窗格的 [網路監控] 下，選擇 [網際網路監視器]。
3. 選擇 Create monitor (建立監視器)。
4. 在 Monitor name (監視器名稱) 中，輸入您要在網路監視器中用於此監視器的名稱。
5. 選擇 Add resources (新增資源)，然後選取資源，以設定要用於此監視器的網路監視器監控界限。

> **Note**
>
> 請注意以下事項：
>
> - 若要使用網際網路監視器產生有意義的輸出，您新增的 VPC 必須透過設定網際網路閘道來連線至網際網路。

- 您可以新增 VPC 和 CloudFront 分發的組合，也可以新增 WorkSpaces 目錄，也可以新增網路負載平衡器。您無法將網路負載平衡器或 WorkSpaces 目錄與其他類型的資源一起新增。

6. 選擇要監控的網際網路流量百分比。
7. 或者，在進階設定下指定其他選項。

   - 對於城市網路上限數量，可選取 Internet Monitor 將監控流量之城市網路 (位置和 ASN，或網際網路服務供應商) 數量的限制。您可以透過編輯監視器，隨時變更此限制。請參閱選擇城市網路上限數量。

     若要重設為預設值，請輸入 `500000`。

     無論選擇監控的流量百分比為何，如果您設定城市網路上限數量，則其會設定網路監視器為應用程式監控的城市網路數量上限。

   - 或者，可以指定 Amazon S3 儲存貯體名稱並自訂字詞，以針對所有監控的城市網路，將網際網路度量發佈至 Amazon S3。

     網際網路監視器會發佈您應用程式的前 500 名 (按流量計算) 網際網路測量，以 CloudWatch 記錄每五分鐘。如果您選擇將度量發佈到 S3，測量值仍會發佈到 CloudWatch 日誌。如需詳細資訊，請參閱 在 Amazon 網際網路監視器中將 CloudWatch 網際網路測量發佈。

   - 也可以選擇為監視器新增標籤。

8. 選擇 Create monitor (建立監視器)。

建立監視器後，您可以隨時編輯監視器，例如變更應用程式流量百分比、更新城市網路的上限數量，或者新增或移除資源。您也可以刪除監視器。若要在網路監視器主控台中執行這些任務，請選擇監視器，然後在動作選單中選擇選項。請注意，您無法變更監視器的名稱。

檢視網路監視器儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在功能窗格中，選擇 [網路監控]，然後選擇 [網際網路監控

   Monitor (監視器) 索引標籤會顯示您已建立的監視器清單。

若要查看特定監視器的詳細資訊，請選擇一個監視器。

將資源新增至您的監視器

建立監視器時，必須將應用程式的資源與其建立關聯：Amazon 虛擬私有雲 (VPC)、網路負載平衡器、Amazon CloudFront 分發、網路負載平衡器 (NLBS) 或 Amazon 目錄。 WorkSpaces 然後，Internet Monitor 會知道應用程式面向網際網路的流量和用戶端所在的位置，而且它可以建立和維護流量設定檔，以決定要針對監視器發佈的相關測量結果。

您可以將下列類型的資源新增至網際網路監視器中的「受監視的資源」。請注意，互聯網監視器不支持在一個監視器中添加不同類型的資源。

- VPC：您在區域中新增的每個 VPC 皆為受監控的資源。當您新增 VPC 時，Internet 監控器會監控 VPC 中任何網際網路對向應用程式的流量，例如 Amazon EC2 執行個體上託管的應用程式、Network Load Balancer 後方或容器中的應用程式。 AWS Fargate
- Network Load Balancer：您新增的每個 NLB 皆為受監控的資源。
- CloudFront 分 CloudFront 佈：您新增的每個分配都是受監視的資源。
- WorkSpaces 目錄：您在區域中新增的每個目 WorkSpaces 錄都是受監控的資源。

您監控 VPC 的流量時，會監控 VPC 後負載平衡器上託管之應用程式的流量。您可以選擇監控個別 Network Load Balancer 負載平衡器的流量，而不是監控具有多個負載平衡器的 VPC。例如，如果您需要了解和設定功能，以便在負載平衡器層級獲得更好的效能或效率，這很有幫助。或者，您可能需要 Network Load Balancer 層級的合規資訊。

將資源新增至網路監視器中的監視器時，請注意下列事項：

- 若要使用網際網路監視器產生有意義的輸出，您新增的 VPC 必須透過設定網際網路閘道來連線至網際網路。
- 互聯網監視器不支持在一個監視器中添加不同類型的資源。

將 VPC 或 NLB 新增為資源時，選擇加入的區域存在區域差異。如需詳細資訊，請參閱 支持 AWS 區域 Amazon CloudWatch 互聯網監視器。

此外，測量最後一哩延遲的資源也有所不同。對於網際網路監視器延遲度量，VPC、NLB 和 WorkSpaces 目錄不包含最後一哩延遲。

選擇要監控的應用程式流量百分比

您針對要監控的應用程式流量百分比選擇的涵蓋範圍，會決定受監控的應用程式城市網路 (用戶端位置和 ASN (通常是網際網路服務供應商)) 數量，最高可達您也可設定的選用城市網路上限。

如果您選擇監視低於 100% 的應用程式流量，則監視器可能有可觀測性差距。這是因為如果 Amazon CloudWatch 網際網路監控器在您未監控流量的地方建立健康事件，您將不會意識到這些問題。對於用戶端存取應用程式的效能和可用性分數資訊，您的涵蓋範圍也可能較小。

以下章節會描述探索流量百分比設定和涵蓋範圍的選項，並了解增加或減少涵蓋範圍的影響。

- 探索變更應用程式流量百分比
- 檢視以不同流量百分比設定監控的城市網路數量

探索變更應用程式流量百分比

您可以在變更應用程式流量百分比時，檢視監控的城市網路數量，探索建議您將百分比變更為的值。本節中的程序提供 step-by-step 相關資訊。

在網路監視器主控台中，您可以嘗試增加或減少監視器的應用程式流量百分比，並檢視因此涵蓋的城市網路預估數量。使用此選項，您可以快速查看變更流量百分比如何影響受監控的城市監視器數量。這可讓您了解可為應用程式選擇的適當應用程式流量百分比。

增加和減少應用程式流量百分比來探索監控涵蓋範圍

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在左側導覽窗格的 [網路監控] 下，選擇 [網際網路監視器]。
3. 在您的監視器清單中，選擇監視器。
4. 在概觀標籤「監控的流量」區段中，選擇百分比圖表，然後選擇更新監控涵蓋範圍。
5. 在探索並設定流量監控涵蓋範圍對話方塊中，按一下箭頭以增加或減少要監控的流量百分比。選擇 100% 流量，您可以看到全面監控多少城市網路，以監控您的應用程式。
6. 若要深入瞭解監控的城市網路數量 (此處估計) 可能會對您的成本造成影響，請選擇 [CloudWatch 定價計算器] 的連結，然後向下捲動至 [網際網路監控]。
7. 若要設定新的監控流量百分比，請選擇更新監控涵蓋範圍。或者，若要保持目前的涵蓋範圍層級，請選擇取消。

檢視以不同流量百分比設定監控的城市網路數量

您可以檢視以不同應用程式流量百分比，為您應用程式監控的城市網路數量。本節中的程序提供 step-by-step 相關資訊。

在網路監視器主控台中，您可以檢視圖表，其顯示在您指定的時間間隔內，以不同的應用程式流量百分比監控，城市網路的涵蓋範圍會如何變化。這會快速視覺化和比較特定流量百分比的應用程式監控涵蓋範圍，全都顯示在一張圖上。

檢視應用程式流量百分比及相應城市網路涵蓋範圍的圖表

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在左側導覽窗格的 [網路監控] 下，選擇 [網際網路監視器]。
3. 在您的監視器清單中，選擇監視器。
4. 選擇流量洞察標籤，然後向下捲動至網際網路流量圖。
5. 在比較流量涵蓋範圍的選項下，在下拉式清單中，選取一或多個百分比。您可以選擇一或多個應用程式流量百分比，系統會更新監控的城市網路總計圖，顯示網路監視器為該流量百分比提供的監控涵蓋範圍。選擇 100% 流量的城市網路，您可以看到全面監控多少城市網路，以監控您的應用程式。

請謹記以下幾點：

- 根據應用程式流量前一小時的城市網路數目來計算流量涵蓋範圍。這表示在您選擇要監控的特定流量百分比之後，與這裡的流量涵蓋範圍比較圖中所顯示的城市網路相較，監控應用程式的城市網路可能會比較少。
- 若要確保所有應用程式流量都受到監控，請將 `TrafficPercentageToMonitor` 設定為 100 且不要設定 `MaxCityNetworksToMonitor`。或者，您可將 `MaxCityNetworksToMonitor` 設定為 500,000，即網路監視器中的上限。
- 如果您設定了城市網路上限，則無論您選取應用程式流量百分比選項如何，受監控城市網路的總數都絕不會超過該上限。
- 您可以進一步了解監控的城市網路數量可能會對您成本造成的影響。在 CloudWatch 頁面的定價計算器上，向下捲動至網際網路監視器。

若要設定新的監控流量百分比，請在探索其他流量涵蓋範圍選項中，選擇更新監控涵蓋範圍。在對話方塊中，選擇流量百分比，然後選擇更新監視器涵蓋範圍。

選擇城市網路上限數量

Amazon CloudWatch Internet Monitor 可以監控用戶端存取應用程式資源的部分或所有位置，以及他們透過存取應用程式的所有 ASN (通常是網際網路服務供應商)，也就是應用程式網際網路流量的城市網路。您可以在建立監視器時，選擇要監控的應用程式流量百分比 (可以透過編輯監視器隨時更新)。

除設定流量百分比外，您也可以針對受監控的城市網路數量設定上限。本節說明城市網路上限如何協助您管理帳單費用，並提供資訊和範例，協助您決定要設定的上限。

您為城市網路數量設定的上限，可確保帳單費用可預期。如需詳細資訊，請參閱 Amazon CloudWatch 定價。您還可以使用 CloudWatch 價格計算器了解實際監控的城市網絡數量的不同值如何影響您的帳單。若要探索選項，請在 [定價計算器 CloudWatch] 頁面上，向下捲動至 [網際網路監控]。

若要更新監視器並變更城市網路上限數量，請參閱 編輯或刪除網路監視器的監視器。

帳單與城市網路上限的關係

針對受監控的城市網路數量設定上限，可協助避免帳單產生非預期的費用。舉例來說，如果您的流量模式差異很大，就會相當實用。超過包含的前 100 個城市網路後 (每個帳戶的所有監視器中)，帳單費用會按受監控的每個城市網路增加。無論選擇監控的流量百分比為何，如果您設定城市網路上限數量，則其會限制網路監視器為應用程式監控的城市網路數量。

您僅須為實際受監控的城市網路數量付費。您選擇的城市網路上限數量，可讓您設定網路監視器使用監視器監控流量時可包含的總數上限。您可以透過編輯監視器，隨時變更此上限數量。

若要探索選項，請在 [定價計算器 CloudWatch] 頁面上，向下捲動至 [網際網路監控]。如需網際網路監視器定價的詳細資訊，請參閱 Amazon 定 CloudWatch 價頁面上的網際網路監控器一節。

如何選擇城市網路上限數量

為協助您決定要選取的城市網路上限數量，請針對應用程式考慮想要監控的流量。下列網路監視器指標可協助您在建立監視器後分析流量使用情況和涵蓋範圍：`CityNetworksMonitored`、`TrafficMonitoredPercent` 及一或多個 `CityNetworksFor`*`NN`*`PercentTraffic` 指標，其中 *`NN`* 為下列其中一項的百分比值：25、50、90、95、99 或 100。若要查看這些指標的定義，以及所有其他網路監視器指標，請參閱使用 CloudWatch 指標與 Amazon CloudWatch 互聯網監控。

若要查看網際網路流量涵蓋範圍的概觀圖表，請前往 CloudWatch 儀表板上的 [流量見解] 索引標籤，然後在 [網際網路流量圖表] 區段中，選擇流量涵蓋範圍的比較選項選項。區段中顯示的圖表會顯示為您應用程式監控的實際城市網路數量，以及您在下拉式清單中所選取不同應用程式流量百分比的折線。若要進一步了解，請參閱設定應用程式流量百分比。

若要更詳細地探索您的選項，可以使用「網路監視器」指標 (如下列範例所述)。這些範例會顯示如何根據您想要的應用程式網際網路流量覆蓋範圍，選取適合您的城市網路上限數量。使用「指標」中的「網際網路監視器」 CloudWatch 度量查詢可協助您深入瞭解應用程式網際網路流量涵蓋範圍

決定城市網路上限數量的範例

舉例來說，假設您設定的監控上限數量為 100 個城市網路，且 2637 個城市網路的用戶端會存取您的應用程式。在 CloudWatch 指標中，您會看到下列網際網路監視器指標傳回：

```
CityNetworksMonitored 100
TrafficMonitoredPercent  12.5
CityNetworksFor90PercentTraffic  2143
CityNetworksFor100PercentTraffic  2637
```

在此範例中，您可以看到目前正在監控 12.5% 的網際網路流量 (上限數量設為 100 個城市網路)。如果您想要監控 90% 的流量，則下個指標會提供以下相關資訊：`CityNetworksFor90PercentTraffic` 會表示您可能需要監控 2,143 個城市網路，才可達到 90% 涵蓋範圍。為此，您可能要更新監視器並將城市網路上限數量設定為 2,143 個。

同樣地，假設您想要針對應用程式監控 100% 的網際網路流量。下個指標 `CityNetworksFor100PercentTraffic` 會表示若要達成此目標，您應該更新監視器，並將城市網路上限數量設定為 2,637 個。

如果您現在將上限設定為 5,000 個城市網路，由於該數量大於 2,637 個，因此您會看到下列傳回的指標：

```
CityNetworksMonitored 2637
TrafficMonitoredPercent  100
CityNetworksFor90PercentTraffic  2143
CityNetworksFor100PercentTraffic  2637
```

在這些指標中，您可以看到如果設定更高的限制，則可監控所有 2,637 個城市網路 (即為 100% 網際網路流量)。

在 Amazon 網際網路監視器中將 CloudWatch 網際網路測量發佈

您可以選擇讓 Amazon CloudWatch Internet Monitor 將網際網路測量發佈到 Amazon S3，將面向網際網路的流量發佈到監視器中受監控的城市網路 (用戶端位置和 ASN，通常是網際網路服務供應商)，最高可達 500,000 個城市網路服務限制。互聯網監視器自動發布互聯網測量到 CloudWatch 日誌每五分鐘的前 500 名（通過流量）城市網絡為每個監視器。它發佈到 S3 的測量包括發佈到 CloudWatch 日誌的前 500 名。

您可以選擇發布到 S3 的選項，並指定要在建立或更新監視器時將度量發布到的儲存貯體。您必須先在 S3 中建立儲存貯體，才可在網路監視器中進行指定。可發佈至 S3 的網際網路度量之服務限制為

500,000 個城市網路。網路監視器會將網際網路度量作為事件 (存放在儲存貯體的一系列壓縮日誌檔案物件) 發佈至 S3。

當您建立適用於網際網路監視器的 S3 儲存貯體以發佈測量時，請務必遵循 CloudWatch 日誌提供的許可指導。這樣做可確保 Internet Monitor 可以將日誌直接發佈到 S3，並且 AWS 可以在需要時建立和變更與接收日誌的日誌群組相關聯的資源政策。如需詳細資訊，請參閱 Amazon 日誌使用者指南中的傳送至 CloudWatch CloudWatch 日誌的日誌。

發佈的日誌檔案已壓縮。如果您使用 Amazon S3 主控台開啟日誌檔案，則其會解壓縮日誌檔案，並顯示網際網路度量事件。如果您下載這些檔案，則必須解壓縮才能查看事件。

您也可以使用 Amazon Athena 查詢日誌檔案中的網際網路度量。Amazon Athena 是一種互動式查詢服務，可讓您透過使用標準 SQL 輕鬆分析 Amazon S3 中的資料。如需詳細資訊，請參閱 使用 Amazon Athena 查詢 Amazon S3 日誌檔案中的網際網路度量。

使用網路監視器監控

建立 Amazon CloudWatch Internet Monitor 監視器後，有幾種方法可以使用它：例如，您可以在 CloudWatch 儀表板中檢視資訊、使用取得資訊 AWS Command Line Interface，以及設定健康狀態警示。

您的監視器會提供應用程式和組態偏好設定的相關資訊，讓網路監視器可以自訂度量和指標，為您在事件中發布。網路監視器會從 AWS的全域基礎架構足跡收集度量。這些度量是來自世界各地的大量網路效能和可用性資訊。透過使用您為應用程式新增之資源的資訊，網路監視器可將您範圍內的效能和可用性度量發佈至應用程式作用中的城市網路 (即用戶端位置和 ASN [通常為網際網路服務供應商 (ISP)])。因此，Internet Monitor 儀表板和 CloudWatch 記錄檔中的度量和指標 (關於可用性、效能、傳輸的監控位元組以及往返時間)，都是針對您的用戶端位置和 ASN 而定。

網路監視器也會確定效能和可用性何時出現異常。根據預設，Internet Monitor 會使用針對用戶端位置中每個來源-目標配 AWS 對收集的可用性和效能測量來覆蓋您的流量，以判斷效能或可用性何時明顯下降。當應用程式的位置和範圍出現顯著降級時，網路監視器會產生運作狀態事件，並將問題的相關資訊發佈至監視器。

在您建立監視器後，即可使用其存取網路監視器提供的資訊或收到該資訊相關的提醒，方法如下：

- 使用 CloudWatch 儀表板來檢視和探索效能、可用性和健全狀況事件；探索應用程式的歷史資料；以及深入瞭解設定應用程式以獲得更佳效能的新方法。要進一步了解，請參閱下列項目：
  - 追蹤 Amazon CloudWatch 網際網路監控器中的即時效能和可用性 (概觀標籤)
  - 過濾和查看 Amazon CloudWatch 互聯網監視器歷史數據 (歷史資源管理器選項卡

  - 在 Amazon CloudWatch 網際網路監控器中取得改善應用程式效能的見解 (流量見解索引標
- 設定運作狀態事件閾值，變更觸發網路監視器為您應用程式建立運作狀態事件的條件。您可以設定整體閾值和局部 (城市網路) 閾值。若要進一步了解，請參閱變更運作狀態事件閾值。
- 使用 AWS CLI 指令搭配 Internet Monitor API 動作來檢視流量設定檔資訊、檢視度量、列出健全狀況事件等。如需進一步了解，請參閱使用 CLI 與 Amazon CloudWatch 互聯網監視器的例子。
- 使用標準 CloudWatch 工具，例如 CloudWatch 參與者見解、 CloudWatch 指標總管和CloudWatch 日誌洞察，以視覺化方式呈現中 CloudWatch的資料。如需進一步了解，請參閱使用 CloudWatch 工具和網際網路監視器查詢介面探索您的資料。
- 如果您已開啟將測量結果發佈到 S3，請使用 Athena 搭配 S3 日誌存取和分析應用程式的網路監視器網際網路測量結果。
- 建立 Amazon EventBridge 通知，以在網際網路監控器判定有健康事件時提醒您。如需進一步了解，請參閱使用 Amazon CloudWatch 互聯網監控 Amazon EventBridge。
- 當互聯網監視器確定問題是由 AWS 網絡引起的時候，自動接收 AWS Health Dashboard 通知。該通知包括緩解問題所採取的步驟。 AWS

編輯或刪除網路監視器的監視器

使用 [動作] 功能表，您可以在建立監視器之後編輯或刪除 Amazon CloudWatch 網際網路監視器中的監視器。例如，您可以編輯監視器以執行下列動作：

- 變更要監控的應用程式流量百分比
- 設定或更新城市網路上限數量
- 針對可用性或效能分數變更運作狀態事件閾值
- 新增或移除資源
- 啟用或更新將事件發布至 Amazon S3

您也可以刪除監視器。請注意，建立監視器後，您無法變更監視器的名稱。

若要變更監視器或刪除監視器，請使用下列其中一個程序。

編輯監視器

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在左側導覽窗格的 [網路監控] 下，選擇 [網際網路監視器]。

3. 選擇您的監視器，然後選擇動作選單。
4. 選擇更新監視器。
5. 進行需要的更新。例如，若要變更要監控的流量百分比，請在要監控的應用程式流量下，選取或輸入百分比。
6. 選擇更新。

若要刪除監視器

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在左側導覽窗格的 [網路監控] 下，選擇 [網際網路監視器]。
3. 選擇您的監視器，然後選擇動作選單。
4. 選擇停用。
5. 再次選擇「動作」功能表，然後選擇「刪除」。

如需可更新之選項的詳細資訊，請參閱下列內容：

- 若要進一步了解您在網路監視器中新增的資源，請參閱將資源新增至您的監視器。
- 若要進一步了解應用程式流量百分比，請參閱選擇要監控的應用程式流量百分比。
- 若要進一步了解變更運作狀態事件的閾值，請參閱變更運作狀態事件閾值。
- 若要進一步解城市網路上限數量，請參閱選擇城市網路上限數量。
- 若要進一步了解選擇將事件發布到 S3，請參閱在 Amazon 網際網路監視器中將 CloudWatch 網際網路測量發佈。

使用 Amazon VPC 添加或創建 Amazon CloudWatch 互聯網監視器監視器

當您在中建立 Amazon Virtual Private Cloud VPC 時 AWS Management Console，您可以選擇性地選擇在 Amazon CloudWatch 網際網路監視器中為其設定監控。您可以將 VPC 新增至現有的監視器，或者您可以選擇在 Amazon VPC 主控台中為 VPC 建立新的監視器。

藉由將網路監視器與 VPC 搭配使用，您可以檢視並評估有關可用性、效能、傳輸的受監控位元組，以及應用程式用戶端位置和 ASN (通常是網際網路服務供應商) 往返時間的度量和指標。網路監視器還會確定效能和可用性何時出現異常狀況，並在監視器中建立運作狀態事件，您可以選擇是否收到該事件的通知。若要進一步了解如何使用監視器來管理並改善用戶端使用應用程式的體驗，請參閱 使用網路監視器監控。

> **Important**
>
> 若要建立監視器，或將 VPC 新增至現有監視器，您必須擁有正確的權限。如需詳細資訊，請參閱 Amazon CloudWatch 互聯網監視器的 Identity and Access Management。

將 VPC 新增至現有監視器

當您在中建立 VPC 時，您可以選擇讓 Amazon CloudWatch 網際網路監視器為您新增 VPC 至現有的監視器。 AWS Management Console新增 VPC 之後，請等待幾分鐘，然後 VPC 的指標就會開始顯示在網際網路監視器主控台上。

您可以隨時編輯監視器，以刪除 VPC 或添加其他 VPC 或其他資源。您也可以變更正在監控的流量百分比，或進行其他變更。如果您選擇從監視器中移除 VPC，則網路監視器不會再監控從用戶端到該 VPC 的流量。

若要進一步了解如何更新監視器，請參閱 編輯或刪除網路監視器的監視器。

為 VPC 建立監視器

如果您選擇為 VPC 建立監視器，則建立監視器精靈會引導您完成這些步驟。建立監視器時，您可以將 VPC 新增為受監視的資源。您也可以視需要選擇要監控應用程式的用戶端流量百分比 (預設值為 100%)。

您可檢閱 使用控制台在 Amazon CloudWatch 互聯網監控器中創建監視器 中的資訊以進一步了解。

定價

隨著 Amazon CloudWatch 互聯網監視器，你只需支付你使用什麼。網路監視器的定價有兩個元件：每個受監控的資源費用和每個城市網路費用。城市網路是用戶端存取應用程式資源的位置，以及用戶端存取資源的網路 (ASN，例如網際網路服務供應商或 ISP)。

如需詳細資訊，包括定價範例，請參閱 Amazon CloudWatch 互聯網監控定價

停止監控 VPC

如果您想要停止使用網際網路監視器監視您的 VPC 人雲端資源，請在網際網路監視器主控台中執行下列動作：

從監視器中移除標籤

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/

2. 在左側導覽窗格的 [網路監控] 下，選擇 [網際網路監視器]。
3. 選擇您的監視器，然後選擇動作選單。
4. 選擇更新監視器。
5. 在新增資源下，選擇移除資源。
6. 選擇要移除的 VPC，然後選擇移除。
7. 選擇更新。

添加或創建一個 Amazon CloudWatch 互聯網監視器監視器 CloudFront

在 Amazon CloudFront 主控台中分發的指標儀表板上，您可以在 Amazon CloudWatch 網際網路監控器中為分發設定其他監控。您可以將分發新增至現有的監視器，也可以為發佈建立新的監視器。

透過搭 CloudFront 配發佈使用 Internet Monitor，您可以檢視和評估有關可用性、效能、傳輸的監視位元組，以及應用程式用戶端位置和 ASN (通常是網際網路服務提供者) 特定的來回時間的度量和指標。網路監視器還會確定效能和可用性何時出現異常狀況，並在監視器中建立運作狀態事件，您可以選擇是否收到該事件的通知。若要進一步了解如何使用監視器來管理並改善用戶端使用應用程式的體驗，請參閱 使用網路監視器監控。

> **Important**
>
> 若要建立監視器，或將發佈新增至現有監視器，您必須擁有正確的權限。如需詳細資訊，請參閱 Amazon CloudWatch 互聯網監視器的 Identity and Access Management。

將分發新增至現有監視器

您可以選擇讓 Internet Monitor 直接從中的 CloudFront 指標儀表板將分發新增至現有監視器 AWS Management Console。新增分發之後，請等待幾分鐘，然後在 Internet Monitor 主控台上就會開始顯示該散發的指標。

您可以隨時編輯監視器，以移除散佈或新增其他發行版或其他資源。您也可以變更正在監控的流量百分比，或進行其他變更。如果您選擇從監視器移除散佈，則 Internet Monitor 將不再監控從用戶端傳送到該分發的流量。

若要進一步了解如何更新監視器，請參閱 編輯或刪除網路監視器的監視器。

建立發佈的監視器

如果您選擇建立發佈的監視器，[建立監視] 精靈會引導您完成這些步驟。您可以在建立監視器時將散佈新增為受監視的資源。您也可以視需要選擇要監控應用程式的用戶端流量百分比 (預設值為 100%)。

您可檢閱 使用控制台在 Amazon CloudWatch 互聯網監控器中創建監視器 中的資訊以進一步了解。

定價

隨著 Amazon CloudWatch 互聯網監視器，你只需支付你使用什麼。網路監視器的定價有兩個元件：每個受監控的資源費用和每個城市網路費用。城市網路是用戶端存取應用程式資源的位置，以及用戶端存取資源的網路 (ASN，例如網際網路服務供應商或 ISP)。

如需詳細資訊，包括定價範例，請參閱 Amazon CloudWatch 互聯網監控定價

停止監視發佈

如果您想要停止使用網際網路監視器監視您的散發資源，請在網際網路監視器主控台中執行下列動作：

從監視器中移除標籤

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在左側導覽窗格的 [網路監控] 下，選擇 [網際網路監視器]。
3. 選擇您的監視器，然後選擇動作選單。
4. 選擇更新監視器。
5. 在新增資源下，選擇移除資源。
6. 選擇要移除的分佈，然後選擇 [移除]。
7. 選擇更新。

使用 CLI 與 Amazon CloudWatch 互聯網監視器的例子

本節包括使用 AWS Command Line Interface 與 Amazon CloudWatch 網際網路監視器操作搭配使用的範例。

在開始之前，請確保您登錄以使用具 AWS CLI 有要監控的 Amazon 虛擬私有雲 (VPC)、網絡負載平衡器、Amazon CloudFront 分發或 Amazon WorkSpaces 目錄的相同 AWS 帳戶使用。網路監視器不支援跨帳戶存取資源。若要取得有關使用的更多資訊 AWS CLI，請參閱《AWS CLI 指令參考》。如需有關搭配 Amazon CloudWatch 網際網路監控器使用 API 動作的詳細資訊，請參閱 Amazon CloudWatch 網際網路監控器 API 參考指南。

主題

- 建立監視器
- 檢視螢幕詳細資料
- 列出運作狀態事件
- 檢視特定運作狀態事件
- 檢視監視器清單
- 編輯監視器
- 刪除監視器

建立監視器

在網路監視器中建立監視器時，您需要提供名稱並將資源與監視器建立關聯，以顯示應用程式網際網路流量的所在位置。您可以指定一個流量百分比來定義受監控的應用程式流量。這也會決定受監控的城市網路數量，城市網路即用戶端位置和 ASN [通常為網際網路服務供應商 (ISP)]。若要協助控制帳單，您也可以選擇針對應用程式資源，設定要監控的城市網路上限數量。如需詳細資訊，請參閱 選擇城市網路上限數量。

最後，您可以選擇是否要將應用程式的所有網際網路測量結果發佈到 Amazon S3。前 500 名城市網路的網際網路測量 (依流量) 會自動發佈至網際網路監控器的 CloudWatch 記錄檔，但您也可以選擇將所有測量結果發佈到 S3。

若要使用建立監視器 AWS CLI，請使用`create-monitor`指令。下列命令會建立一個監控 100% 流量但將城市網路上限設定為 10,000 個的監視器、新增 VPC 資源，以及選擇將網際網路測量結果發佈到 Amazon S3。

> **Note**
>
> 網際網路監視器發佈到 CloudWatch 記錄網際網路測量每五分鐘的前 500 個城市網路 (用戶端位置和 ASN，通常是網際網路服務提供者或 ISP) 將流量傳送到每個監視器。您也可以選擇將所有受監控之城市網路 (最多 500,000 個城市網路服務限制) 的網際網路度量發佈至 Amazon S3 儲存貯體。如需詳細資訊，請參閱 在 Amazon 網際網路監視器中將 CloudWatch 網際網路測量發佈。

```
aws internetmonitor --create-monitor monitor-name "TestMonitor" \
    --traffic-percentage-to-monitor 100 \
```

```
    --max-city-networks-to-monitor 10000 \
    --resources "arn:aws:ec2:us-east-1:111122223333:vpc/vpc-11223344556677889" \
    --internet-measurements-log-delivery
 S3Config="{BucketName=MyS3Bucket,LogDeliveryStatus=ENABLED}"
```

```
{
"Arn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/TestMonitor",
 "Status": "ACTIVE"
}
```

> **Note**
>
> 您無法變更監視器的名稱。

檢視螢幕詳細資料

若要使用檢視監視器的相關資訊 AWS CLI，請使用`get-monitor`指令。

```
aws internetmonitor get-monitor --monitor-name "TestMonitor"
```

```
{
    "ClientLocationType": "city",
    "CreatedAt": "2022-09-22T19:27:47Z",
    "ModifiedAt": "2022-09-22T19:28:30Z",
    "MonitorArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/TestMonitor",
    "MonitorName": "TestMonitor",
    "ProcessingStatus": "OK",
    "ProcessingStatusInfo": "The monitor is actively processing data",
    "Resources": [
        "arn:aws:ec2:us-east-1:111122223333:vpc/vpc-11223344556677889"
    ],
    "MaxCityNetworksToMonitor": 10000,
    "Status": "ACTIVE"
}
```

列出運作狀態事件

如果應用程式的網際網路流量效能降低，網路監視器會在監視器中建立運作狀態事件。若要使用查看目前健康事件的清單 AWS CLI，請使用指`list-health-events`令

```
aws internetmonitor list-health-events --monitor-name "TestMonitor"
```

```
{
    "HealthEvents": [
        {
            "EventId": "2022-06-20T01-05-05Z/latency",
            "Status": "RESOLVED",
            "EndedAt": "2022-06-20T01:15:14Z",
            "ServiceLocations": [
                {
                    "Name": "us-east-1"
                }
            ],
            "PercentOfTotalTrafficImpacted": 1.21,
            "ClientLocations": [
                {
                    "City": "Lockport",
                    "PercentOfClientLocationImpacted": 60.370000000000005,
                    "PercentOfTotalTraffic": 2.01,
                    "Country": "United States",
                    "Longitude": -78.6913,
                    "AutonomousSystemNumber": 26101,
                    "Latitude": 43.1721,
                    "Subdivision": "New York",
                    "NetworkName": "YAHOO-BF1"
                }
            ],
            "StartedAt": "2022-06-20T01:05:05Z",
            "ImpactType": "PERFORMANCE",
            "EventArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/
TestMonitor/health-event/2022-06-20T01-05-05Z/latency"
        },
        {
            "EventId": "2022-06-20T01-17-56Z/latency",
            "Status": "RESOLVED",
            "EndedAt": "2022-06-20T01:30:23Z",
            "ServiceLocations": [
                {
                    "Name": "us-east-1"
                }
            ],
            "PercentOfTotalTrafficImpacted": 1.29,
            "ClientLocations": [
```

```
            {
                "City": "Toronto",
                "PercentOfClientLocationImpacted": 75.32,
                "PercentOfTotalTraffic": 1.05,
                "Country": "Canada",
                "Longitude": -79.3623,
                "AutonomousSystemNumber": 14061,
                "Latitude": 43.6547,
                "Subdivision": "Ontario",
                "CausedBy": {
                    "Status": "ACTIVE",
                    "Networks": [
                        {
                            "AutonomousSystemNumber": 16509,
                            "NetworkName": "Amazon.com"
                        }
                    ],
                    "NetworkEventType": "AWS"
                },
                "NetworkName": "DIGITALOCEAN-ASN"
            },
            {
                "City": "Lockport",
                "PercentOfClientLocationImpacted": 22.91,
                "PercentOfTotalTraffic": 2.01,
                "Country": "United States",
                "Longitude": -78.6913,
                "AutonomousSystemNumber": 26101,
                "Latitude": 43.1721,
                "Subdivision": "New York",
                "NetworkName": "YAHOO-BF1"
            },
            {
                "City": "Hangzhou",
                "PercentOfClientLocationImpacted": 2.88,
                "PercentOfTotalTraffic": 0.7799999999999999,
                "Country": "China",
                "Longitude": 120.1612,
                "AutonomousSystemNumber": 37963,
                "Latitude": 30.2994,
                "Subdivision": "Zhejiang",
                "NetworkName": "Hangzhou Alibaba Advertising Co.,Ltd."
            }
        ],
```

```
            "StartedAt": "2022-06-20T01:17:56Z",
            "ImpactType": "PERFORMANCE",
            "EventArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/
TestMonitor/health-event/2022-06-20T01-17-56Z/latency"
        },
        {
            "EventId": "2022-06-20T01-34-20Z/latency",
            "Status": "RESOLVED",
            "EndedAt": "2022-06-20T01:35:04Z",
            "ServiceLocations": [
                {
                    "Name": "us-east-1"
                }
            ],
            "PercentOfTotalTrafficImpacted": 1.15,
            "ClientLocations": [
                {
                    "City": "Lockport",
                    "PercentOfClientLocationImpacted": 39.45,
                    "PercentOfTotalTraffic": 2.01,
                    "Country": "United States",
                    "Longitude": -78.6913,
                    "AutonomousSystemNumber": 26101,
                    "Latitude": 43.1721,
                    "Subdivision": "New York",
                    "NetworkName": "YAHOO-BF1"
                },
                {
                    "City": "Toronto",
                    "PercentOfClientLocationImpacted": 29.770000000000003,
                    "PercentOfTotalTraffic": 1.05,
                    "Country": "Canada",
                    "Longitude": -79.3623,
                    "AutonomousSystemNumber": 14061,
                    "Latitude": 43.6547,
                    "Subdivision": "Ontario",
                    "CausedBy": {
                        "Status": "ACTIVE",
                        "Networks": [
                            {
                                "AutonomousSystemNumber": 16509,
                                "NetworkName": "Amazon.com"
                            }
                        ],
```

```
                        "NetworkEventType": "AWS"
                    },
                    "NetworkName": "DIGITALOCEAN-ASN"
                },
                {
                    "City": "Hangzhou",
                    "PercentOfClientLocationImpacted": 2.88,
                    "PercentOfTotalTraffic": 0.7799999999999999,
                    "Country": "China",
                    "Longitude": 120.1612,
                    "AutonomousSystemNumber": 37963,
                    "Latitude": 30.2994,
                    "Subdivision": "Zhejiang",
                    "NetworkName": "Hangzhou Alibaba Advertising Co.,Ltd."
                }
            ],
            "StartedAt": "2022-06-20T01:34:20Z",
            "ImpactType": "PERFORMANCE",
            "EventArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/
TestMonitor/health-event/2022-06-20T01-34-20Z/latency"
        }
    ]
}
```

檢視特定運作狀態事件

若要使用 CLI 查看有關特定運作狀態事件的更詳細資訊，請使用您的監視器名稱和運作狀態事件 ID 執行 `get-health-event` 命令。

```
aws internetmonitor get-monitor --monitor-name "TestMonitor" --event-id "health-event/
TestMonitor/2021-06-03T01:02:03Z/latency"
```

```
{
    "EventId": "2022-06-20T01-34-20Z/latency",
    "Status": "RESOLVED",
    "EndedAt": "2022-06-20T01:35:04Z",
    "ServiceLocations": [
        {
            "Name": "us-east-1"
        }
    ],
    "EventArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/TestMonitor/
health-event/2022-06-20T01-34-20Z/latency",
```

```
    "LastUpdatedAt": "2022-06-20T01:35:04Z",
    "ClientLocations": [
        {
            "City": "Lockport",
            "PercentOfClientLocationImpacted": 39.45,
            "PercentOfTotalTraffic": 2.01,
            "Country": "United States",
            "Longitude": -78.6913,
            "AutonomousSystemNumber": 26101,
            "Latitude": 43.1721,
            "Subdivision": "New York",
            "NetworkName": "YAHOO-BF1"
        },
        {
            "City": "Toronto",
            "PercentOfClientLocationImpacted": 29.770000000000003,
            "PercentOfTotalTraffic": 1.05,
            "Country": "Canada",
            "Longitude": -79.3623,
            "AutonomousSystemNumber": 14061,
            "Latitude": 43.6547,
            "Subdivision": "Ontario",
            "CausedBy": {
                "Status": "ACTIVE",
                "Networks": [
                    {
                        "AutonomousSystemNumber": 16509,
                        "NetworkName": "Amazon.com"
                    }
                ],
                "NetworkEventType": "AWS"
            },
            "NetworkName": "DIGITALOCEAN-ASN"
        },
        {
            "City": "Shenzhen",
            "PercentOfClientLocationImpacted": 4.07,
            "PercentOfTotalTraffic": 0.61,
            "Country": "China",
            "Longitude": 114.0683,
            "AutonomousSystemNumber": 37963,
            "Latitude": 22.5455,
            "Subdivision": "Guangdong",
            "NetworkName": "Hangzhou Alibaba Advertising Co.,Ltd."
```

```
        },
        {
            "City": "Hangzhou",
            "PercentOfClientLocationImpacted": 2.88,
            "PercentOfTotalTraffic": 0.7799999999999999,
            "Country": "China",
            "Longitude": 120.1612,
            "AutonomousSystemNumber": 37963,
            "Latitude": 30.2994,
            "Subdivision": "Zhejiang",
            "NetworkName": "Hangzhou Alibaba Advertising Co.,Ltd."
        }
    ],
    "StartedAt": "2022-06-20T01:34:20Z",
    "ImpactType": "PERFORMANCE",
    "PercentOfTotalTrafficImpacted": 1.15
}
```

檢視監視器清單

若要使用 CLI 查看您帳戶中所有監視器的清單，請執行 `list-monitors` 命令。

```
aws internetmonitor list-monitors
```

```
{
    "Monitors": [
        {
            "MonitorName": "TestMonitor",
            "ProcessingStatus": "OK",
            "Status": "ACTIVE"
        }
    ],
    "NextToken": " zase12"
}
```

編輯監視器

若要使用 CLI 更新監視器的相關資訊，請使用 `update-monitor` 命令並指定要更新的監視器名稱。您可以更新要監控的流量百分比、城市網路的上限數量、新增或移除網路監視器用於監控流量的資源，以及將監視器狀態從 ACTIVE 變更為 INACTIVE，反之亦然。請注意，您無法變更監視器的名稱。

`update-monitor` 呼叫的回應僅會傳回 `MonitorArn` 和 `Status`。

下列範例會說明如何使用 update-monitor 命令將要監控的城市網路上限數量變更為 50000：

```
aws internetmonitor update-monitor --monitor-name "TestMonitor" --max-city-networks-to-monitor 50000
```

```
{
    "MonitorArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/TestMonitor",
    "Status": " ACTIVE "
}
```

以下範例會說明如何新增和移除資源：

```
aws internetmonitor update-monitor --monitor-name "TestMonitor" \
    --resources-to-add "arn:aws:ec2:us-east-1:111122223333:vpc/vpc-11223344556677889" \
    --resources-to-remove "arn:aws:ec2:us-east-1:111122223333:vpc/vpc-2222444455556666"
```

```
{
    "MonitorArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/TestMonitor",
    "Status": "ACTIVE"
}
```

以下範例會說明如何使用 update-monitor 命令將監視器狀態變更為 INACTIVE：

```
aws internetmonitor update-monitor --monitor-name "TestMonitor" --status "INACTIVE"
```

```
{
    "MonitorArn": "arn:aws:internetmonitor:us-east-1:111122223333:monitor/TestMonitor",
    "Status": "INACTIVE"
}
```

刪除監視器

您可以使用 delete-monitor 命令，利用 CLI 刪除監視器。首先，您必須將監視器設定為非作用中。若要執行此操作，請使用 update-monitor 命令將狀態變更為 INACTIVE。使用 get-monitor 命令並檢查狀態，確認監視器為非作用中。

監視器狀態為 INACTIVE 後，您便可以使用 CLI 執行 delete-monitor 命令來刪除監視器。成功的 delete-monitor 呼叫回應是空白的。

```
aws internetmonitor delete-monitor --monitor-name "TestMonitor"
```

```
{}
```

使用網路監視器儀表板監控和最佳化

本節中的資訊說明如何篩選和檢視 Amazon CloudWatch Internet Monitor 儀表板上的資訊，以視覺化方式呈現並取得有關 AWS 應用程式網際網路流量和設定的見解。

建立監視器以監控應用程式的網際網路效能和可用性後，Amazon CloudWatch Internet Monitor 會針對用戶端位置網路 (城市網路) 配對發佈包含網際網路測量的 CloudWatch 日誌，並針對應用程式以及每個 AWS 區域 節點和節點的流量發佈彙總 CloudWatch 指標。您可以使用數種不同方式篩選、探索，並從網路監視器的這些資訊獲得以行動為導向的建議。

若要開始使用，請在 CloudWatch 主控台的 [網路監控] 下，選擇 [網際網路監視器]。

本節主要說明如何使用篩選和檢視網際網路監視器測量結果 AWS Management Console。或者，您可以將網際網路監視器 API 作業與 AWS CLI 或 SDK 搭配使用，直接處理 CloudWatch 記錄檔中儲存的網際網路監視器事件。如需詳細資訊，請參閱使用監視器和度量資訊。如需使用 API 操作的詳細資訊，請參閱使用 CLI 與 Amazon CloudWatch 互聯網監視器的例子和 Amazon CloudWatch 網際網路監控器 API 參考。

網路監視器儀表板中具有三個索引標籤：

- 您可以在 Overview (概觀) 索引標籤上，查看有關應用程式目前和歷史效能和可用性的資訊，以及影響用戶端位置的運作狀態事件。
- 在下一個索引標籤歷史總管中，您可以依位置、ASN、日期等條件進行篩選，並使用圖表來視覺化一段時間內的網際網路流量指標。
- 在流量洞察索引標籤上，您不僅可以檢視以數種可自訂方式摘要之排序靠前的受監控流量相關資訊，也可以取得有關最佳化設定的建議，以針對不同的位置/ASN 配對改善效能。Internet Monitor 會根據流量模式和過去的效能，當您變更流量或使用 AWS 資源的方式時，預測應用程式的效能改善。您也可以查看圖表，根據您為監視器選擇的應用程式流量百分比，比較監控涵蓋範圍中包含的城市網路數量。

此外，由於 Internet Monitor 會產生並發佈記錄檔，其中包含有關流量的測量值，因此您可以使用主控台中的其他 CloudWatch 工具，進一步視覺化 Internet Monitor 發佈的資料，包括 CloudWatch 參與者

見解、 CloudWatch 指標和 CloudWatch 記錄深入解析。如需詳細資訊，請參閱 使用 CloudWatch 工具和網際網路監視器查詢介面探索您的資料。

請參閱下列各節，了解如何使用網路監視器來探索效能和可用性測量結果。

主題

- 追蹤 Amazon CloudWatch 網際網路監控器中的即時效能和可用性 (概觀標籤)
- 過濾和查看 Amazon CloudWatch 互聯網監視器歷史數據（歷史資源管理器選項卡
- 在 Amazon CloudWatch 網際網路監控器中取得改善應用程式效能的見解 (流量見解索引標

追蹤 Amazon CloudWatch 網際網路監控器中的即時效能和可用性 (概觀標籤)

使用 CloudWatch 主控台中的 [網際網路監視器] 底下的 [概觀] 索引標籤，以取得監視器追蹤之流量的效能和可用性的高階檢視。此索引標籤也會顯示網際網路流量概觀地圖，其中包含流量叢集，可協助您視覺化應用程式的全域流量，以及運作狀態事件的位置和影響。

運作狀態分數

Health 全狀況分數圖表會顯示全球流量的效能和可用性資訊。 AWS 具有關於不同 ASN 和 AWS 服務之地理位置之間網路流量的網際網路效能和可用性的大量歷史資料。Internet Monitor 會使用從其全球網路佔用量擷取的連線資料，計算網際網路流量的效能和可用性基準。 AWS 這與我們用 AWS 來監控我們自己的互聯網正常運行時間和可用性的數據相同。

網路監視器會使用這些測量結果作為基準，偵測應用程式效能和可用性的下降時機，並與基準進行比較。為方便您查看上述下降情況，我們會以效能分數和可用性分數形式向您回報該資訊。如需詳細資訊，請參閱 使用 CloudWatch 工具和網際網路監視器查詢介面探索您的資料。

運作狀態分數圖表包含您所選時段內發生的運作狀態事件。如果發生運作狀態事件，您會在圖表上看到效能或可用性線下降。如果您選取事件，便可看到更多詳細資訊和區間顯示在圖表上，其中包含日期和時間資訊，顯示事件持續的時間長度。

您也可以直接存取各資料點的日誌檔案，查看這些指標。在「動作」功能表中，選擇「檢視 CloudWatch 記錄」。

網際網路流量概觀

網際網路流量概觀地圖會顯示特定於您的最終使用者從中存取應用程式的位置和 ASN 的網際網路流量和運作狀態事件。地圖上的灰色國家/地區為包含應用程式流量的國家/地區。

地圖上的各圓圈表示在您所選時段內某個區域中的運作狀態事件。Internet Monitor 會在偵測到特定臨界值的問題時，建立健康事件， AWS 並在您的其中一個資源與使用者存取您的應用程式的城市

網路之間的連線。在地圖上選擇一個圓圈，便會顯示更多有關該位置運作狀態事件的詳細資料。此外，針對具有運作狀態事件的叢集，您可以在地圖下方的 Health events (運作狀態事件) 資料表中查看詳細資訊。

請注意，如果網路監視器判斷事件對應用程式有重大全域影響，會在監視器中建立運作狀態事件。如果在您所選時段內，未有任何運作狀態事件對用戶端位置的流量影響超過閾值，則地圖會顯示空白。如需詳細資訊，請參閱網路監視器建立和解決運作狀態事件的時機。

變更運作狀態事件閾值

您可以針對網路監視器為應用程式建立運作狀態事件的方式和時機，設定幾個選項。選擇更新閾值即可變更。

您可以變更觸發網路監視器建立運作狀態事件的整體閾值。效能分數和可用性分數的預設運作狀態事件閾值都是 95%。也就是說，應用程式的整體效能或可用性分數降至 95% 或以下時，網路監視器會建立運作狀態事件。對於整體閾值而言，運作狀態事件可能由單一大型問題或多個較小問題觸發。

您也可以變更本機 (即城市網路) 閾值及整體影響程度百分比，它們結合起來會觸發運作狀態事件。設定閾值，在分數低於一或多個城市網路 (位置和 ASN，通常是 ISP) 的閾值時建立運作狀態事件，您便可深入了解流量較低的地點何時發生問題等。

其他本機閾值選項可與可用性或效能評分的本機閾值搭配使用。第二個因素是網路監視器根據本機閾值建立運作狀態事件之前，必須受到影響的整體流量百分比。

透過設定總流量和本機流量的閾值選項，可微調建立運作狀態事件的頻率，以符合您的應用程式用途和需求。請注意，將局部閾值設得較低時，通常會建立更多運作狀態事件，這取決於您的應用程式和您設定的其他閾值組態值。

總而言之，您可以使用下列方式，為效能分數、可用性分數或兩者，設定運作狀態事件閾值：

- 選擇其他全域閾值來觸發運作狀態事件。
- 選擇其他局部閾值來觸發運作狀態事件。您也可以使用此選項，變更網路監視器建立事件之前，必須超過之對整體應用程式造成影響的百分比。
- 選擇關閉根據局部閾值觸發運作狀態事件，或啟用局部閾值選項。

您也可以設定效能分數、可用性分數或兩者的選項。您可以設定一組選項，或只設定其中一個選項。

若要更新效能分數、可用性分數或兩者的閾值和其他組態選項，請執行下列動作：

變更閾值組態選項

1. 在中 AWS Management Console，瀏覽至 CloudWatch，然後在左側導覽窗格中選擇 [網際網路監視器]。
2. 在概觀標籤的運作狀態事件時間表區段中，選擇更新閾值。
3. 在開啟的對話方塊頁面上，針對觸發網路監視器建立運作狀態事件的閾值和其他選項，選擇想要的新值和選項。您可以執行下列任何操作：
   - 為可用性分數閾值、效能分數閾值或兩者，選擇新值。

     每個設定的區段中的圖表，會顯示您應用程式可用性或效能的目前閾值設定和實際最近運作狀態事件分數。檢視一般值，您可以了解建議將閾值變更為的值。

     提示：若要檢視較大圖表並變更時間範圍，請選擇圖表右上角的展開按鈕。
   - 選擇開啟或關閉可用性、效能或兩者的局部閾值。啟用選項時，您可以設定希望網路監視器建立運作狀態事件時的閾值和影響等級。
4. 設定閾值選項之後，請選擇更新運作狀態事件閾值以儲存更新。

若要進一步了解運作狀態事件的運作方式，請參閱網路監視器建立和解決運作狀態事件的時機。

運作狀態事件表

運作狀態事件資料表會列出受運作狀態事件影響的用戶端位置，以及事件的相關資訊。下列資料欄包含在資料表中。

	描述
用戶端位置	受事件影響、遭遇延遲增加或可用性降低之最終使用者的位置。 若要進一步了解網路監視器中的用戶端位置準確度，請參閱網路監視器中的地理位置資訊和準確度。
流量影響	事件造成的影響程度 (延遲增加程度或可用性降低程度)。對於延遲，這是事件期間延遲增加的百分比，相較於從此用戶端位置到使用此用戶端網路的此 AWS 位置的一般流量效能。

	描述
用戶端網路	流量行經的網路。一般而言，此為網路流量的網際網路服務供應商 (ISP) 或自治系統編號 (ASN)。
AWS 位置	網路流量的 AWS 位置，可以是 AWS 區域 或網際網路邊緣位置。
影響類型	運作狀態事件的影響類型。運作狀態事件通常是由延遲增加 (效能問題) 或可達性 (可用性問題) 所造成。 您也可以按一下影響類型，以查看造成損害的原因。網際網路監視器可能會分析健全狀況事件的來源，以判斷其是由 AWS ASN (網際網路服務提供者) 所造成。 請注意，此分析會在事件解決後繼續進行。網路監視器最多可以使用新資訊更新事件一小時。

如果您在運作狀態事件資料表中選擇其中一個用戶端位置，則可查看該位置運作狀態事件的詳細資料。例如，您可以查看事件的開始和結束時間，以及本機流量影響。

網路路徑視覺化

完整的損害分析在網路路徑視覺化下具有完整的網路路徑。完整路徑會針對健全狀況事件，顯示應用程式網路路徑上的每個節點，位置與用戶端之間的用戶端- AWS 位置與用戶端之間，以及用戶端-位置配對。

如果網路監視器判定了損害的原因，將標示一個紅色虛線圓圈。損害可能是由 ASN [通常為網際網路服務供應商 (ISP)] 造成，也可能是由 AWS造成。若有數個造成損害的原因，則會圈出多個節點。

過濾和查看 Amazon CloudWatch 互聯網監視器歷史數據（歷史資源管理器選項卡

使用 CloudWatch 主控台中的 [網際網路監視器] 底下的 [歷程資源管理器] 索引標籤，篩選及檢視 CloudWatch 記錄中應用程式的資料。Internet Monitor 會將 CloudWatch 測量結果發佈到您應用程式特

定的可用性、效能、傳輸的監視位元組 (或僅限 WorkSpaces 目錄的用戶端連線計數)，以及監視城市網路中 AWS 區域的往返時間。

> **Note**
>
> 網際網路監視器會針對將流量傳送至每個監視器的前 500 名 (依流量計算) 城市網路 (即用戶端位置和 ASN，通常是網際網路服務提供者或 ISP)，每五分鐘發佈網際網路測量到 CloudWatch 記錄檔。您也可以選擇將所有受監控之城市網路 (最多 500,000 個城市網路服務限制) 的網際網路度量發佈至 Amazon S3 儲存貯體。如需詳細資訊，請參閱 在 Amazon 網際網路監視器中將 CloudWatch 網際網路測量發佈。

若要開始探索應用程式的資料，請選取對應的時段。接著，選擇特定的地理位置 (例如城市) 以及 (可選) 其他篩選條件。網路監視器會將篩選條件套用至網際網路測量結果日誌中的資料，這些資料是網路監視器針對您應用程式流量的城市網路發佈的。然後，您可以看到顯示效能分數、可用性分數、傳輸的監視位元組 (針對 VPC、網路負載平衡器和 CloudFront 分發) 或用戶端連線計數 (針對 WorkSpaces 目錄)，以及應用程式隨時間推移的往返時間 (RTT) 的資料圖表。

圖表下方的 All events (所有事件) 資料表，會顯示篩選條件傳回的應用程式流量運作狀態事件，以及各事件的相關資訊。其中包含下列欄位。

	描述
事件開始	運作狀態事件開始時間。
Status	事件是仍為作用中或已解決。
用戶端位置	受事件影響、遭遇延遲增加或效能降低之最終使用者的位置。 若要進一步了解網路監視器中的用戶端位置準確度，請參閱網路監視器中的地理位置資訊和準確度。
流量影響	事件對此運作狀態事件位置的加權影響。例如，對延遲的影響，相較於透過用戶端 ASN (通常是網際網路服務提供者 (ISP) 從用戶端 AWS 位置到位置的一般流量效能。同樣地，對於影響可用性的事件，您會看到對可用性的影響，與從屬端

	描述
	出貨預先通知上該位置的用戶端 AWS 位置的一般可用性相比。
事件期間	事件持續的時間長度。運作狀態事件對應用程式用戶端位置造成的影響不超過 (總計) 5% 時，網路監視器便會終止運作狀態事件。
用戶端 ISP	ASN (通常為網際網路服務供應商 (ISP)) 即網路流量的電信業者。
服務位置	網路流量來源的服務位置，可以是網際網路邊緣位置 AWS 區域 或網際網路邊緣位置。

您也可以直接存取各資料點的日誌，查看自己應用程式的測量結果。在「動作」功能表中，選擇「檢視 CloudWatch 記錄」。請注意，由於測量事件在建立時會發佈到您的帳戶，因此您也可以根據它們建立其他 CloudWatch 儀表板或警示。如需詳細資訊，請參閱 在 Amazon CloudWatch 網際網路監控器中取得改善應用程式效能的見解 (流量見解索引標 及 創建報警與 Amazon CloudWatch 互聯網監控。

您不僅可以探索和分析網路監視器測量結果和指標，根據其建立儀表板和警示，還可以使用網路監視器，協助您了解可能改善應用程式效能的方法。Traffic insights (流量洞察) 索引標籤會提供多種方式，協助您探索選項。如需詳細資訊，請參閱流量洞察索引標籤上的流量最佳化建議。此外，您可以查看網際網路監視器使用案例章節中的具體範例。

在 Amazon CloudWatch 網際網路監控器中取得改善應用程式效能的見解 (流量見解索引標

使用主控台中的 [網際網路監 CloudWatch 控器] 底下的 [流量見解] 索引標籤，查看應用程式的最高流量 (按數量) 的摘要資訊。您可以透過多種方式篩選應用程式流量，並加以排序。接著，請向下捲動並為應用程式選取不同的設定組合，查看網路監視器建議的最佳替代方案，以取得最快的第一個位元組時間 (TTFB) 效能。

網際網路監視器發佈到 CloudWatch 記錄網際網路測量每五分鐘的前 500 名 (依流量) 城市網路 (即，用戶端位置和 ASN，通常是網際網路服務提供者或 ISP) 將流量傳送到每個監視器。您也可以選擇將所有受監控之城市網路 (最多 500,000 個城市網路服務限制) 的網際網路度量發佈至 Amazon S3 儲存貯體。如需詳細資訊，請參閱 在 Amazon 網際網路監視器中將 CloudWatch 網際網路測量發佈。

排序靠前的流量摘要

您可以先依用戶端位置篩選，檢視應用程式在特定時段內整體流量和效能的高階摘要。您也可以依流量查看應用程式頂端 (或底端) 用戶端位置的效能，並以多種方式篩選和排序。例如，您可以依精細程度 (即城市、行政區域、國家/地區或都會區域)、總流量、第一位元組時間 (TTFB) 平均值，以及其他因素排序。

若要進一步了解網路監視器中的用戶端位置準確度，請參閱網路監視器中的地理位置資訊和準確度。

> **Note**
>
> 您使用的篩選條件會套用至整個頁面，因此這些條件會影響摘要圖表和總流量資訊中所包含的城市網路，以及接下來流量最佳化建議區段中所包含的城市網路。

流量最佳化建議

流量最佳化建議區段會針對流量顯示經篩選之一組受監控的城市網路 (位置和 ASN [網際網路服務供應商])，以及每個網路的總用戶端流量。表格中的項目是根據您針對頁面頂端之流量洞察的應用程式流量而選擇的篩選條件所建立。預設為流量排名前 10 名的城市。您通常會在表格中看到 10 個以上的資料列，這是因為每個唯一的城市網路配對都有項目。也就是說，用戶端透過達拉斯、德州、美國和康卡斯特等位置存取您應用程式的位置 (城市) 和 ASN (網路供應商) 的每個組合，都會有一個資料列。

> **Note**
>
> 若要查看所有受監控城市網路的流量最佳化建議，您可以直接在 CloudWatch Insights 中執行查詢。如需不包含限制本頁面上城市網路清單之地理資料精細程度篩選器的範例查詢，請參閱使用 CloudWatch 日誌洞察與 Amazon CloudWatch 互聯網監控。

在本節中，請選取不同的選項：Amazon EC2 CloudFront，或兩者。這可讓您查看與目前 TTFB 相比，當您將應用程式與不同 AWS 區域中的這些服務搭配使用時，用戶端的預測第一個位元組 (TTFB) 值的平均時間為何。如需有關 TTFB 計算的詳細資訊，請參閱 TTFB 和延遲的AWS 計算。

選取不同的選項，然後檢視資料表中的結果，您即可開始規劃設定和部署，以改善用戶端的效能。請注意，當資料無法顯示時，您可能會在資料欄中看到破折號 (-) 而非值。若要檢閱如何改善效能的特定範例，請參閱使用 Amazon CloudWatch 網際網路監視器獲得更好的遊戲體驗。

例如，若要開始使用，針對特定城市網路 (用戶端位置和 ASN 配對)，請嘗試選取 EC2 或 CloudFront 選項，或兩者。針對表格中列出的每個城市網路，Internet Monitor 會根據該選項的流量路由選擇 (透過特定的 AWS 區域)，與目前的設定相比，顯示 TTFB 的潛在效能改善。(請注意，為了完整性，該資料表亦包含已最佳化的路由。) 例如，您可能會看到透過 us-east-1 使用 EC2 路由的預測平均 TTFB 為 50 毫秒 (相較於您目前設定是透過 us-west-2 使用 EC2 路由，其 TTFB 為 100 毫秒)。因此您可能會考慮透過 us-west-2 進行路由。

另一個範例是，您可以選取 EC2，然後看到它不會對一個用戶端位置和 ASN 產生可衡量的效能差異，但接著請注意，當您選取 CloudFront 相同區域時，會稍微降低 TTFB。這表明，如果您在應用程式前面新增 CloudFront 發行版，可能會導致效能改善，而且可能值得嘗試，這個用戶端位置和 ASN。

使用 CloudWatch 工具和網際網路監視器查詢介面探索您的資料

除了透過 Amazon CloudWatch 網際網路監控儀表板將應用程式的效能和可用性視覺化外，您還可以使用多種方法深入了解網際網路監控器為您產生的資料。這些方法包括使用 CloudWatch 工具與存儲在 CloudWatch 日誌文件中的互聯網監視器數據，並使用互聯網監視器查詢接口。您可以使用的工具包括 CloudWatch 日誌洞察、 CloudWatch 指標、 CloudWatch 貢獻者洞察和 Amazon Athena。取決於您的需求，您可以使用這些工具的一部分或全部，以及儀表板來探索網路監視器資料。

Internet Monitor 會彙總應用程式和每個應用程式的流 CloudWatch 量指標 AWS 區域，並包含總流量影響、可用性和往返時間等資料。此資料會發佈至 CloudWatch 記錄檔，也可與網際網路監視器查詢介面搭配使用。地理精細程度相關詳細資訊和可供探索的其他方面資訊略有差異。

Amazon CloudWatch 網際網路監控器每隔 5 分鐘為您的監視器發佈資料，然後透過多種方式提供資料。下表列出了存取網路監視器資料的案例，並描述了每個案例所收集資料的特徵。

功能	CloudWatch 日誌	匯出至 S3	查詢介面	CloudWatch 儀表板
預設為啟用	是	否	是	是
收集資料的城市網路數目	前 500 次 (請參閱下方註釋)	全部	全部	全部
資料保留	使用者控制	使用者控制	30 天	30 天
收集資料的地理精細程度	全部 (城市網路、都會區 + 網路、	城市網路	全部 (城市網路、都會區 + 網路、	全部 (城市網路、都會區 + 網路、

功能	CloudWatch 日誌	匯出至 S3	查詢介面	CloudWatch 儀表板
	行政區 + 網路、國家/地區 + 網路)		行政區 + 網路、國家/地區 + 網路)	行政區 + 網路、國家/地區 + 網路)
如何查詢和篩選資料	使用 CloudWatch 日誌洞察與 Amazon CloudWatch 互聯網監控	使用 Amazon Athena 查詢 Amazon S3 日誌檔案中的網際網路度量	使用 Amazon CloudWatch 互聯網監視器查詢界面	使用網路監視器儀表板監控和最佳化

注意事項：城市網路擷取的前 500 次度量；都會區 + 網路擷取的前 250 次，行政區 + 網路擷取的前 100 次，國家/地區 + 網路擷取的前 50 次。

本章說明如何使用 CloudWatch 工具或網際網路監視器查詢介面來查詢和探索資料，以及各種方法的範例。

目錄

- 使用 CloudWatch 日誌洞察與 Amazon CloudWatch 互聯網監控
- 使用貢獻者見解與 Amazon CloudWatch 互聯網監視器
- 使用 CloudWatch 指標與 Amazon CloudWatch 互聯網監控
- 使用 Amazon Athena 查詢 Amazon S3 日誌檔案中的網際網路度量
- 使用 Amazon CloudWatch 互聯網監視器查詢界面

使用 CloudWatch 日誌洞察與 Amazon CloudWatch 互聯網監控

Amazon CloudWatch Internet Monitor 會將可用性和往返時間的精細度量資訊發佈至CloudWatch 日 CloudWatch 誌，而且您可以使用日誌洞察查詢篩選特定城市或地理區 (用戶端位置)、用戶端 ASN (ISP) 和 AWS 來源位置的記錄子集。

若要進一步了解網路監視器中的用戶端位置準確度，請參閱網路監視器中的地理位置資訊和準確度。

本節中的範例可協助您建立 CloudWatch Logs Insights 查詢，以進一步瞭解您自己的應用程式流量測量值和指標。如果您在 CloudWatch 記錄檔見解中使用這些範例，請以您自己的監視器#### *Mon* itor 名稱。

檢視流量最佳化建議

在網路監視器中的流量洞察索引標籤中，您可以檢視依位置篩選的流量最佳化建議。若要查看該索引標籤上「流量最佳化建議」區段中顯示的相同資訊，但沒有位置資料粒度篩選器，您可以使用下列「 CloudWatch 記錄見解」查詢。

1. 在中 AWS Management Console，導覽至 [CloudWatch 記錄檔見解]。
2. 在 Log Group (日誌群組) 中，選取 `/aws/internet-monitor/monitorName/byCity` 和 `/aws/internet-monitor/monitorName/byCountry`，然後指定時間範圍。
3. 新增以下查詢，然後執行該查詢。

```
fields @timestamp,
clientLocation.city as @city, clientLocation.subdivision as @subdivision,
 clientLocation.country as @country,
`trafficInsights.timeToFirstByte.currentExperience.serviceName` as @serviceNameField,
concat(@serviceNameField, ` (`, `serviceLocation`, `)`) as @currentExperienceField,
concat(`trafficInsights.timeToFirstByte.ec2.serviceName`, ` (`,
 `trafficInsights.timeToFirstByte.ec2.serviceLocation`, `)`) as @ec2Field,
`trafficInsights.timeToFirstByte.cloudfront.serviceName` as @cloudfrontField,
concat(`clientLocation.networkName`, ` (AS`, `clientLocation.asn`, `)`) as @networkName
| filter ispresent(`trafficInsights.timeToFirstByte.currentExperience.value`)
| stats avg(`trafficInsights.timeToFirstByte.currentExperience.value`) as @averageTTFB,
avg(`trafficInsights.timeToFirstByte.ec2.value`) as @ec2TTFB,
avg(`trafficInsights.timeToFirstByte.cloudfront.value`) as @cloudfrontTTFB,
sum(`bytesIn` + `bytesOut`) as @totalBytes,
latest(@ec2Field) as @ec2,
latest(@currentExperienceField) as @currentExperience,
latest(@cloudfrontField) as @cloudfront,
count(*) by @networkName, @city, @subdivision, @country
| display @city, @subdivision, @country, @networkName, @totalBytes, @currentExperience,
 @averageTTFB, @ec2, @ec2TTFB, @cloudfront, @cloudfrontTTFB
| sort @totalBytes desc
```

檢視網際網路可用性和 RTT (p50、p90 及 p95)

若要檢視流量的網際網路可用性和往返時間 (p50、p90 和 p95)，您可以使用下列 CloudWatch 日誌深入解析查詢。

最終使用者地理位置：美國伊利諾州芝加哥

最終使用者網路 (ASN)：AS7018

AWS 服務地點：美國東部 (維吉尼亞北部) 區域

若要檢視日誌，請依下列步驟執行：

1. 在中 AWS Management Console，導覽至 [CloudWatch 記錄檔見解]。
2. 在 Log Group (日誌群組) 中，選取 `/aws/internet-monitor/monitorName/byCity` 和 `/aws/internet-monitor/monitorName/byCountry`，然後指定時間範圍。
3. 新增以下查詢，然後執行該查詢。

該查詢會傳回在所選時段內，從伊利諾州芝加哥 AS7018 連線至美國東部 (維吉尼亞北部) 區域的所有使用者效能資料。

```
fields @timestamp,
internetHealth.availability.experienceScore as availabilityExperienceScore,
internetHealth.availability.percentageOfTotalTrafficImpacted as
 percentageOfTotalTrafficImpacted,
internetHealth.performance.experienceScore as performanceExperienceScore,
internetHealth.performance.roundTripTime.p50 as roundTripTimep50,
internetHealth.performance.roundTripTime.p90 as roundTripTimep90,
internetHealth.performance.roundTripTime.p95 as roundTripTimep95
 | filter clientLocation.country == `United States`
 and clientLocation.city == `Chicago`
 and serviceLocation == `us-east-1`
 and clientLocation.asn == 7018
```

如需詳細資訊，請參閱使用日誌深入解析分析CloudWatch 記錄資料。

使用貢獻者見解與 Amazon CloudWatch 互聯網監視器

CloudWatch 貢獻者見解可協助您識別應用程式的主要用戶端位置和網路 (ASN 或網際網路服務供應商)。使用下列範例參與者見解規則，開始使用 Amazon CloudWatch 網際網路監視器有用的規則。如需詳細資訊，請參閱 建立 Contributor Insights 規則。

若要進一步了解網路監視器中的用戶端位置準確度，請參閱網路監視器中的地理位置資訊和準確度。

> **Note**
>
> 網路監視器每五分鐘會發佈一次資料，因此在您設定 Contributor Insights 規則之後，必須將期間調整為五分鐘才能看到圖表。

依可用性影響檢視最受影響的位置和 ASN

若要依可用性下降程度檢視最受影響的用戶端位置和 ASN，可以在語法編輯器中使用下列 Contributor Insights 規則。以您自有的監視器名稱取代 *monitor-name*。

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "AggregateOn": "Sum",
    "Contribution": {
        "Filters": [
            {
                "Match": "$.clientLocation.city",
                "IsPresent": true
            }
        ],
        "Keys": [
            "$.clientLocation.city",
            "$.clientLocation.networkName"
        ],
        "ValueOf": "$.awsInternetHealth.availability.percentageOfTotalTrafficImpacted"
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "/aws/internet-monitor/monitor-name/byCity"
    ]
}
```

依延遲影響檢視最受影響的用戶端位置和 ASN

若要依往返時間增加程度檢視最受影響的用戶端位置和 ASN，可以在語法編輯器中使用下列 Contributor Insights 規則。以您自有的監視器名稱取代 *monitor-name*。

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "AggregateOn": "Sum",
    "Contribution": {
        "Filters": [            {
```

```
                "Match": "$.clientLocation.city",
                "IsPresent": true
            }
        ],
        "Keys": [
            "$.clientLocation.city",
            "$.clientLocation.networkName"
        ],
        "ValueOf": "$.awsInternetHealth.performance.percentageOfTotalTrafficImpacted"
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "/aws/internet-monitor/monitor-name/byCity"
    ]
}
```

依總流量百分比檢視最受影響的用戶端位置和 ASN

若要依總流量百分比檢視最受影響的用戶端位置和 ASN，可以在語法編輯器中使用下列 Contributor Insights 規則。以您自有的監視器名稱取代 *monitor-name*。

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "AggregateOn": "Sum",
    "Contribution": {
        "Filters": [
            {
                "Match": "$.clientLocation.city",
                "IsPresent": true
            }
        ],
        "Keys": [
            "$.clientLocation.city",
            "$.clientLocation.networkName"
        ],
        "ValueOf": "$.percentageOfTotalTraffic"
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "/aws/internet-monitor/monitor-name/byCity"
```

```
    ]
}
```

使用 CloudWatch 指標與 Amazon CloudWatch 互聯網監控

Amazon CloudWatch Internet Monitor 會將指標發佈到您的帳戶，包括效能、可用性、往返時間和輸送量 (每秒位元組數) 的指標，您可以在 CloudWatch 主控台的 CloudWatch 指標中檢視這些指標。若要尋找監視器的所有指標，請在「 CloudWatch 指標」儀表板中查看自訂命名空間`AWS/InternetMonitor`。

指標會彙總至 VPC、網路負載平衡器、 CloudFront 分發或監視器中 WorkSpaces 目錄的所有網際網路流量，以 AWS 區域 及每個受監控之網際網路邊緣位置的所有流量。區域由服務位置定義，可以是所有位置或特定區域，例如 `us-east-1`。

注意：城市網路是用戶端位置和 ASN (通常為網際網路服務供應商 (ISP))。

網路監視器會提供下列指標。

指標	描述
PerformanceScore	效能分數表示未發現效能下降的預估流量百分比。
AvailabilityScore	可用性分數表示未發現可用性下降的預估流量百分比。
BytesIn	在所有應用程式的城市網路中，針對應用程式網際網路流量傳入的位元組。
BytesOut	在所有應用程式的城市網路中，針對應用程式網際網路流量傳出的位元組。
BytesInMonitored	在受監控之城市網路中，針對應用程式網際網路流量傳入的位元組。
BytesOutMonitored	在受監控之城市網路中，針對應用程式網際網路流量傳出的位元組。
往返時間 (RTT)	ASN (通常是網際網路服務提供者或 ISP) 與 VPC、網路負載平衡器、 CloudFront 分發或

指標	描述
	目錄特定的位置 (例如城市) 之間的往返時間。AWS 區域 WorkSpaces
CityNetworksMonitored	網際網路監視器針對應用程式網際網路流量監控的城市網路數量。此值不會超過您針對監視器設定為城市網路最大數量的上限值。
TrafficMonitoredPercent	按照網路監視器正在監控的城市網路數量表示 (納入) 之監視器總應用程式網際網路流量的百分比。如果用戶端存取應用程式所經由的城市網路數量，超過您針對監視器所設定的城市網路上限數量，則此值將會低於 100 (即低於 100%)。
CityNetworksFor100 PercentTraffic	如果您想要在網路監視器中監控 100% 的應用程式網際網路流量，應該要設定的城市網路上限數值。
CityNetworksFor99 PercentTraffic	如果您想要在網路監視器中監控 99% 的應用程式網際網路流量，應該要設定的城市網路上限數值。
CityNetworksFor95 PercentTraffic	如果您想要在網路監視器中監控 95% 的應用程式網際網路流量，應該要設定的城市網路上限數值。
CityNetworksFor90 PercentTraffic	如果您想要在網路監視器中監控 90% 的應用程式網際網路流量，應該要設定的城市網路上限數值。
CityNetworksFor75 PercentTraffic	如果您想要在網路監視器中監控 75% 的應用程式網際網路流量，應該要設定的城市網路上限數值。
CityNetworksFor50 PercentTraffic	如果您想要在網路監視器中監控 50% 的應用程式網際網路流量，應該要設定的城市網路上限數值。

指標	描述
CityNetworksFor25 PercentTraffic	如果您想要在網路監視器中監控 25% 的應用程式網際網路流量，應該要設定的城市網路上限數值。

> **Note**
>
> 若要查看使用以上數個指標的範例，以協助判斷為監視器選擇的城市網路上限值，請參閱選擇城市網路上限值。

如需詳細資訊，請參閱 使用 Amazon CloudWatch 指標。

使用 Amazon Athena 查詢 Amazon S3 日誌檔案中的網際網路度量

您可以使用 Amazon Athena 查詢和檢視 Amazon 網際網路監控器發佈到 Amazon S3 儲存貯體的 CloudWatch 網際網路測量結果。網路監視器中有個選項，可讓其針對受監控之城市網路 (用戶端位置和 ASN，通常為網際網路服務供應商 (ISP)) 的面向網際網路流量，將應用程式的網際網路度量發布至 S3 儲存貯體。無論您是否選擇將度量發佈到 S3，Internet Monitor 都會每五分鐘自動將每個監視器的前 500 名（按流量計算）城市網絡的互聯網度量發佈到 CloudWatch 日誌中。

本章節包含針對位於 S3 日誌檔案中的網際網路度量在 Athena 中建立資料表的方法步驟，以及提供範例查詢以查看度量的不同檢視。例如，您可以透過延遲影響查詢前 10 個受影響的城市網路。

使用 Amazon Athena 在網路監視器中建立網際網路度量的資料表

若要搭配網路監視器 S3 日誌檔案開始使用 Athena，請先建立網際網路度量的資料表。

按照此程序中的步驟，以根據 S3 日誌檔案在 Athena 中建立資料表。接著，您可以在資料表上執行 Athena 查詢 (例如，這些範例網際網路測量查詢)，以取得度量的相關資訊。

建立 Athena 資料表

1. 在 https://console.aws.amazon.com/athena/ 中開啟 Athena 主控台。
2. 在 Athena 查詢編輯器中，輸入查詢陳述式，以產生具有網路監視器網際網路度量的資料表。將 LOCATION 參數值取代為 S3 儲存貯體的位置 (存放網路監視器網際網路度量的位置)。

```
CREATE EXTERNAL TABLE internet_measurements (
```

```
    version INT,
    timestamp INT,
    clientlocation STRING,
    servicelocation STRING,
    percentageoftotaltraffic DOUBLE,
    bytesin INT,
    bytesout INT,
    clientconnectioncount INT,
    internethealth STRING,
    trafficinsights STRING
)
PARTITIONED BY (year STRING, month STRING, day STRING)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION
's3://bucket_name/bucket_prefix/AWSLogs/account_id/internetmonitor/AWS_Region/'
TBLPROPERTIES ('skip.header.line.count' = '1');
```

3. 輸入陳述式以建立讀取資料的分割區。例如，下列查詢會針對指定的日期和位置建立單一分割區：

```
ALTER TABLE internet_measurements
ADD PARTITION (year = 'YYYY', month = 'MM', day = 'dd')
LOCATION
's3://bucket_name/bucket_prefix/AWSLogs/account_id/internetmonitor/AWS_Region/YYYY/
MM/DD';
```

4. 選擇執行。

網際網路度量的範例 Athena 陳述式

下列為產生資料表的陳述式範例：

```
CREATE EXTERNAL TABLE internet_measurements (
    version INT,
    timestamp INT,
    clientlocation STRING,
    servicelocation STRING,
    percentageoftotaltraffic DOUBLE,
    bytesin INT,
    bytesout INT,
    clientconnectioncount INT,
    internethealth STRING,
    trafficinsights STRING
)
```

```
PARTITIONED BY (year STRING, month STRING, day STRING)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
LOCATION 's3://internet-measurements/TestMonitor/AWSLogs/1111222233332/internetmonitor/
us-east-2/'
TBLPROPERTIES ('skip.header.line.count' = '1');
```

下列為建立讀取資料之分割區的陳述式範例：

```
ALTER TABLE internet_measurements
ADD PARTITION (year = '2023', month = '04', day = '07')
LOCATION 's3://internet-measurements/TestMonitor/AWSLogs/1111222233332/internetmonitor/
us-east-2/2023/04/07/'
```

可搭配使用網路監視器之網際網路度量的範例 Amazon Athena 查詢

本節包含可與 Amazon Athena 搭配使用的範例查詢，可取得有關發佈至 Amazon S3 之應用程式網際網路度量的資訊。

查詢前 10 個受影響的 (依總流量百分比) 用戶端位置和 ASN

執行此 Athena 查詢可傳回前 10 個受影響的 (依總流量百分比) 城市網路[即用戶端位置和 ASN (通常為網際網路服務供應商)]。

```
SELECT json_extract_scalar(clientLocation, '$.city') as city,
    json_extract_scalar(clientLocation, '$.networkname') as networkName,
    sum(percentageoftotaltraffic) as percentageoftotaltraffic
FROM internet_measurements
GROUP BY json_extract_scalar(clientLocation, '$.city'),
    json_extract_scalar(clientLocation, '$.networkname')
ORDER BY percentageoftotaltraffic desc
limit 10
```

查詢前 10 個受影響的 (依可用性) 用戶端位置和 ASN

執行此 Athena 查詢可傳回前 10 個受影響的 (依總流量百分比) 城市網路[即用戶端位置和 ASN (通常為網際網路服務供應商)]。

```
SELECT json_extract_scalar(clientLocation, '$.city') as city,
    json_extract_scalar(clientLocation, '$.networkname') as networkName,
    sum(
        cast(
            json_extract_scalar(
```

```
                internetHealth,
                '$.availability.percentageoftotaltrafficimpacted'
            )
        as double )
    ) as percentageOfTotalTrafficImpacted
FROM internet_measurements
GROUP BY json_extract_scalar(clientLocation, '$.city'),
    json_extract_scalar(clientLocation, '$.networkname')
ORDER BY percentageOfTotalTrafficImpacted desc
limit 10
```

查詢前 10 個受影響的 (依延遲情況) 用戶端位置和 ASN

執行此 Athena 查詢可傳回前 10 個受影響的 (依延遲影響) 城市網路[即用戶端位置和 ASN (通常為網際網路服務供應商)]。

```
SELECT json_extract_scalar(clientLocation, '$.city') as city,
    json_extract_scalar(clientLocation, '$.networkname') as networkName,
    sum(
        cast(
            json_extract_scalar(
                internetHealth,
                '$.performance.percentageoftotaltrafficimpacted'
            )
        as double )
    ) as percentageOfTotalTrafficImpacted
FROM internet_measurements
GROUP BY json_extract_scalar(clientLocation, '$.city'),
    json_extract_scalar(clientLocation, '$.networkname')
ORDER BY percentageOfTotalTrafficImpacted desc
limit 10
```

查詢用戶端位置和 ASN 的流量重點

執行此 Athena 查詢可傳回流量重點資訊，(包含可用性分數、效能分數，以及城市網路的第一個位元組時間)，即用戶端位置和 ASN (通常為網際網路服務供應商)。

```
SELECT json_extract_scalar(clientLocation, '$.city') as city,
    json_extract_scalar(clientLocation, '$.subdivision') as subdivision,
    json_extract_scalar(clientLocation, '$.country') as country,
    avg(cast(json_extract_scalar(internetHealth, '$.availability.experiencescore') as
 double)) as availabilityScore,
```

```
    avg(cast(json_extract_scalar(internetHealth, '$.performance.experiencescore') as
 double)) performanceScore,
    avg(cast(json_extract_scalar(trafficinsights,
 '$.timetofirstbyte.currentexperience.value') as double)) as averageTTFB,
    sum(bytesIn) as bytesIn,
    sum(bytesOut) as bytesOut,
    sum(bytesIn + bytesOut) as totalBytes
FROM internet_measurements
where json_extract_scalar(clientLocation, '$.city') != 'N/A'
GROUP BY
json_extract_scalar(clientLocation, '$.city'),
    json_extract_scalar(clientLocation, '$.subdivision'),
    json_extract_scalar(clientLocation, '$.country')
ORDER BY totalBytes desc
limit 100
```

如需有關使用 Athena 的詳細資訊，請參閱《Amazon Athena 使用者指南》https://docs.aws.amazon.com/athena/latest/ug/。

使用 Amazon CloudWatch 互聯網監視器查詢界面

若要進一步瞭解 AWS 應用程式的網際網路流量，其中一個選項是使用 Amazon CloudWatch 網際網路監視器查詢介面。若要使用查詢介面，請使用您選擇的資料篩選條件來建立查詢，然後執行查詢，以傳回網路監視器資料的子集。探索查詢傳回的資料，這可讓您深入了解應用程式在網際網路上的執行情況。

您可查詢並探索網路監視器使用監視器擷取的所有指標，包括可用性和效能分數、傳輸的位元組、往返時間和第一個位元組的時間 (TTFB)。

網路監視器使用查詢介面來提供您可在網路監視器主控台儀表板中探索的資料。使用儀表板中的搜尋選項 (位於歷史總管索引標籤或流量洞察索引標籤)，您可查詢和篩選應用程式的網際網路資料。

如果您希望探索和篩選資料的彈性比儀表板所提供的資料更有彈性，您可以透過搭配 AWS Command Line Interface 或 AWS SDK 使用 Internet Monitor API 作業，自行使用查詢介面。本節會介紹可與查詢介面搭配使用的查詢類型，以及您可指定用於建立資料子集的篩選條件，以取得應用程式網際網路流量的相關洞察。

主題

- 如何使用查詢介面
- 查詢範例

- 取得查詢結果
- 故障診斷

如何使用查詢介面

您可選擇查詢類型，然後指定篩選值，以使用查詢介面建立查詢，進而傳回所需的特定日誌檔案資料子集。然後，您可使用資料子集，進一步篩選和排序、建立報告等等。

查詢運作程序如下：

1. 當您執行查詢時，網路監視器會傳回對查詢而言不重複的 `query ID`。本節會描述可用的查詢類型，以及在查詢中篩選資料的選項。若要了解運作方式，您還可檢閱查詢範例章節。
2. 您可以使用 GetQueryResultsAPI 作業指定您的監視器名稱的查詢 ID，以傳回查詢的資料結果。每個查詢類型都會傳回不同的資料集欄位。若要進一步了解，請參閱取得查詢結果。

查詢介面提供下列三種查詢類型。每個查詢類型都會從日誌檔案傳回有關流量的不同資訊集，如下所示。

- 測量：以 5 分鐘的間隔提供可用性分數、效能分數、總流量和往返時間。
- 熱門位置：依流量量，提供您監視的最上層位置和 ASN 組合的可用性分數、效能分數、總流量和第一位元組時間 (TTFB) 資訊。
- 熱門位置詳細資訊：每隔 1 小時提供適用於 Amazon 的 TTFB CloudFront、您目前的組態以及效能最佳的 Amazon EC2 組態。

使用這些查詢類型，您可指定下列一個或多個條件來篩選更多資料：

- AWS location：對於 AWS 位置，您可以指定 CloudFront 或 AWS 區域`us-east-2`，例如`us-west-2`，，等等。
- ASN：指定 ASN，這通常是網際網路服務供應商 (ISP)。
- 用戶端位置：針對位置，請指定城市、都會區、行政區或國家/地區。
- 地理：針對某些查詢指定 `geo`。這對於使用 `Top locations` 查詢類型的查詢為必要，但對於其他查詢類型則不允許。若要了解何時指定 `geo` 來篩選參數，請參閱查詢範例章節。

您可用來篩選資料的運算子為 `EQUALS` 和 `NOT_EQUALS`。如需篩選參數的詳細資訊，請參閱 FilterParameterAPI 作業。

若要查看有關查詢界面操作的詳細資訊，請參閱 Amazon CloudWatch 網際網路監控器 API 參考指南中的下列 API 操作：

- 若要建立並執行查詢，請參閱 StartQueryAPI 作業。
- 若要停止查詢，請參閱 StopQueryAPI 作業。
- 若要傳回已建立之查詢的資料，請參閱 GetQueryResultsAPI 作業。
- 若要擷取查詢的狀態，請參閱 GetQueryStatusAPI 作業。

查詢範例

若要建立可用來從監視器的記錄檔擷取一組篩選資料的查詢，請使用 StartQueryAPI 作業。您可指定查詢的查詢類型和篩選參數。然後，當您使用網路監視器查詢介面 API 操作，藉由查詢來取得查詢結果時，它會擷取您想要使用的資料子集。

為了說明查詢類型和篩選參數的運作方式，我們來看一些範例。

範例 1

假設您想要擷取特定國家/地區 (一個城市除外) 的所有監視器日誌資料。下列範例顯示了您可針對此案例，使用 `StartQuery` 操作建立查詢的篩選參數。

```
{
   MonitorName: "TestMonitor"
   StartTime: "2023-07-12T20:00:00Z"
   EndTime: "2023-07-12T21:00:00Z"
   QueryType: "MEASUREMENTS"
   FilterParameters: [
      {
       Field: "country",
       Operator: "EQUALS",
       Values: ["Germany"]
      },
      {
       Field: "city",
       Operator: "NOT_EQUALS",
       Values: ["Berlin"]
      },
    ]
}
```

範例 2

另舉一例，假設您想要按都會區查看熱門位置。您可針對此案例使用下列範例查詢。

```
{
   MonitorName: "TestMonitor"
   StartTime: "2023-07-12T20:00:00Z"
   EndTime: "2023-07-12T21:00:00Z"
   QueryType: "TOP_LOCATIONS"
   FilterParameters: [
      {
       Field: "geo",
       Operator: "EQUALS",
       Values: ["metro"]
      },
    ]
}
```

範例 3

現在，假設您想要看到洛杉磯都會區的熱門城市網路組合。若要執行此操作，請指定 geo=city，然後將 metro 設定為洛杉磯。現在，該查詢會傳回洛杉磯都會區的熱門城市網路，而不是整個熱門的都會區 + 網路。

以下是您可使用的範例查詢：

```
{
   MonitorName: "TestMonitor"
   StartTime: "2023-07-12T20:00:00Z"
   EndTime: "2023-07-12T21:00:00Z"
   QueryType: "TOP_LOCATIONS"
   FilterParameters: [
      {
       Field: "geo",
       Operator: "EQUALS",
       Values: ["city"]
      },
      {
       Field: "metro",
       Operator: "EQUALS",
       Values: ["Los Angeles"]
      }
    ]
}
```

範例 4

最後，假設您想要擷取特定行政區 (例如，美國州) 的 TTFB 資料。

以下是此案例的範例查詢：

```
{
   MonitorName: "TestMonitor"
   StartTime: "2023-07-12T20:00:00Z"
   EndTime: "2023-07-12T21:00:00Z"
   QueryType: "TOP_LOCATION_DETAILS"
   FilterParameters: [
      {
       Field: "subdivision",
       Operator: "EQUALS",
       Values: ["California"]
      },
    ]
}
```

取得查詢結果

定義查詢之後，您可以透過執行另一個網際網路監視器 API 作業，傳回一組包含查詢的結果GetQueryResults。執行 GetQueryResults 時，您可指定已定義查詢的查詢 ID，以及監視器的名稱。GetQueryResults 會將指定查詢的資料擷取至結果集。

執行查詢時，請確定查詢已完成執行，然後再使用 GetQueryResults 來查看結果。您可以使用GetQueryStatusAPI 作業判斷查詢是否已完成。當查詢的 Status 為 SUCCEEDED 時，您可繼續檢閱結果。

查詢完成後，您可使用下列資訊來協助您檢閱結果。您用於建立查詢的每個查詢類型都包括日誌檔案中不重複的資料集欄位，如下列清單所述：

衡量值

measurements 查詢類型會傳回下列資料：

timestamp, availability, performance, bytes_in, bytes_out, rtt_p50, rtt_p90, rtt_p95

熱門位置

top locations 查詢類型會依位置對資料分組，並提供一段時間的平均資料。傳回的資料包含下列內容：

`aws_location, city, metro, subdivision, country, asn, availability, performance, bytes_in, bytes_out, current_fbl, best_ec2, best_ec2_region, best_cf_fbl`

請注意，只有在您針對 geo 欄位選擇該位置類型時，才會傳回 city、metro 和 subdivision。視乎您為 geo 指定的位置類型會傳回下列位置欄位：

```
city = city, metro, subdivision, country
metro = metro, subdivision, country
subdivision = subdivision, country
country = country
```

熱門位置詳細資訊

`top locations details` 查詢類型會傳回依小時分組的資料。查詢會傳回下列資料：

`timestamp, current_service, current_fbl, best_ec2_fbl, best_ec2_region, best_cf_fbl`

當您執行 `GetQueryResults` API 操作時，網路監視器會在回應中傳回下列項目：

- 包含查詢傳回結果的資料字串陣列。資訊會在與 `Fields` 欄位相符的陣列中傳回，也會透過 API 呼叫傳回。使用 `Fields` 欄位，您可剖析 `Data` 儲存器中的資訊，然後根據您的用途進一步篩選或排序。
- 欄位陣列會列出查詢傳回資料的欄位 (在 `Data` 欄位回應中)。陣列中的每個項目都是一個名稱-資料類型對，例如 `availability_score-float`。

故障診斷

如果在使用查詢界面 API 操作時傳回錯誤，請確認您具有使用 Amazon CloudWatch 網際網路監控器所需的許可。特別確認您具有下列許可：

```
internetmonitor:StartQuery
internetmonitor:GetQueryStatus
internetmonitor:GetQueryResults
internetmonitor:StopQuery
```

這些權限包含在建議的 AWS Identity and Access Management 原則中，以便在主控台中使用 [網際網路監視器] 儀表板。如需詳細資訊，請參閱 Amazon CloudWatch 互聯網監控器的 IAM 許可。

創建報警與 Amazon CloudWatch 互聯網監控

您可以根據 Amazon CloudWatch 網際網路監視器指標建立 Amazon CloudWatch 警示，就像對其他 Amazon CloudWatch 指標一樣。

例如，您可以根據網路監視器指標 `PerformanceScore` 建立警示，並將其設定為在指標低於您選擇的值時傳送通知。您可以依照與其他度量相同的準則來設定「網際網路監視器」 CloudWatch 度量的警示。

您可以選擇為之建立警示的網路監視器指標範例如下：

- PerformanceScore
- AvailabilityScore
- RoundtripTime

若要查看網路監視器可用的所有指標，請參閱 使用 CloudWatch 指標與 Amazon CloudWatch 互聯網監控。

下列程序提供透PerformanceScore過導覽至 CloudWatch 儀表板中的量度來設定警示的範例。然後，您可以按照標準 CloudWatch 步驟根據您選擇的閾值來創建警報，並設置通知或選擇其他選項。

若要PerformanceScore在 CloudWatch 量度中建立警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 選擇指標，然後選擇所有指標。
3. 透過選擇 `AWS/InternetMonitor`，針對網路監視器進行篩選。
4. 選擇 MeasurementSource， MonitorName。
5. 在清單中，選取PerformanceScore。
6. 在GraphedMetrics索引標籤的動作下，選擇鈴鐺圖示以根據靜態臨界值建立警示。

現在，請按照標準 CloudWatch 步驟選擇警報選項。例如 PerformanceScore，您可以選擇在低於特定閾值數目時透過 Amazon SNS 訊息收到通知。另外，您也可以將警示新增至儀表板。

請謹記以下幾點：

- 網路監視器指標通常會在 20 分鐘內計算並發布。

- 當您根據網路監視器指標建立警示時，請務必在設定警示回顧期間考慮發布之前的短暫延遲。建議您設定評估期的回顧期至少為 25 分鐘。

若要進一步了解如何搭配網際網路監視器使用 CloudWatch 警示，請參閱下列部落格文章：使用 Amazon CloudWatch 網際網路監控器增強網際網路可觀察

如需建立 CloudWatch 鬧鐘時選項的詳細資訊，請參閱根據靜態閾值建立 CloudWatch 警示。

使用 Amazon CloudWatch 互聯網監控 Amazon EventBridge

Amazon CloudWatch Internet Monitor 針對網路問題所建立的運作狀態事件會透過 Amazon 發佈 EventBridge，因此您可以傳送有關最終使用者對應用程式體驗下降的通知。

若 EventBridge 要使用網際網路監控健全狀況事件，請遵循此處的指引。

若要在中設定網際網路監視器的規則 EventBridge

1. 在中 AWS Management Console EventBridge，選擇「規則」，然後輸入名稱和說明。在 Default (預設) 事件匯流排上建立規則。
2. 在步驟 2 中，針對事件來源部分選取其他，然後在事件模式下，比對下列來源。

```
{
  "source": ["aws.internetmonitor"]
}
```

3. 在步驟 3 中，針對目標選取AWS 服務和CloudWatch 記錄群組，然後選取現有的記錄群組或建立新的記錄群組。
4. 新增任何所需標籤，然後建立規則。這應該會使用來自的事件填入您選取的 CloudWatch 記錄群組 EventBridge。

如需 EventBridge 規則如何與事件模式搭配運作的詳細資訊，請參閱 Amazon EventBridge 使用者指南中的 Amazon EventBridge 事件模式。

疑難排解 CloudWatch 記錄和指標存取錯誤

若要支援某些功能，Amazon CloudWatch 網際網路監控器必須與特定 Amazon CloudWatch 資源 (包括日誌和指標) 互動。如果網際網路監視器無法存取它需要存取的 CloudWatch 資源，網際網路監視器會設定監視器FAULT_ACCESS_CLOUDWATCH的狀態碼。

監視器可能會出現 FAULT_ACCESS_CLOUDWATCH 狀態有數個原因。以下各節列出這些錯誤的可能原因，以及建議的疑難排解步驟。

互聯網監視器無法訪問您的帳戶中的 CloudWatch 日誌

Internet Monitor 會發佈監視器所追蹤之應用程式流量的診斷日誌。它會將這些記錄檔發佈至下列位置 CloudWatch 記錄檔中的記錄群組：`/aws/internet-monitor/monitor_name/[byCity|byMetro|bySubdivision|byCountry]`。網際網路監視器無法存取這些記錄群組。

錯誤狀態和潛在的解決方案：

- PutLogEvents 節流錯誤：當網際網路監視器服務嘗試發佈監視器的記錄檔時，可能已經限制。CloudWatch檢閱帳戶的限流限制，並在必要時請求提高限制。
- 找不到記錄群組：停用，然後重新啟用監視器。啟用監視器會重新啟動日誌群組建立，這可能會修正問題。
- PutLogEvents 訪問被拒絕錯誤：聯繫 AWS 支持以獲取幫助。
- PutLogEvents 未知或一般錯誤：請聯絡 AWS 支援人員以尋求協助。

網際網路監視器無法存取您帳戶中的 CloudWatch 指標

網際網路監視器會提供監視器所追蹤之應用程式流 CloudWatch 量的特定指標。當網際網路監視器嘗試將這些度量傳遞給時，就會發生錯誤 CloudWatch。

錯誤狀態和潛在的解決方案：

- PutMetricData 節流錯誤：當網際網路監視器服務嘗試將監視器的指標發佈到時，可能已經限制。CloudWatch檢閱帳戶的限流限制，並在必要時請求提高限制。
- PutMetricData 訪問被拒絕錯誤：聯繫 AWS 支持以獲取幫助。
- PutMetricData 未知或一般錯誤：請聯絡 AWS 支援人員以尋求協助。

資料保護和資料隱私與 Amazon CloudWatch 網際網路監控

AWS 共同的責任模型適用於 Amazon CloudWatch 網際網路監控中的資料保護和資料隱私。如此模型所述， AWS 負責保護執行所有 AWS 雲端的全域基礎結構。您負責維護在此基礎設施上託管內容的控制權。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的AWS 共同責任模型和 GDPR 部落格文章。如需有關遵守 GDPR 規定的詳細資源，請參閱一般資料保護規範 (GDPR) 中心。

強烈建議您，絕對不要將最終使用者帳戶號碼、電子郵件地址或其他個人資訊等敏感的識別資訊放入自由格式欄位中。您輸入 Amazon CloudWatch 網際網路監視器或其他服務的任何資料都可能包含在診斷日誌中。

Amazon CloudWatch 互聯網監視器的 Identity and Access Management

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 管理員可以控制完成身分驗證 (已登入) 和獲得授權 (具有許可) 的對象，以使用網路監視器資源。IAM 是您可以使用的 AWS 服務 ，無需額外付費。

> **Important**
>
> 2023 年 2 月 24 日網路監視器資源變更
> 如果您在 2023 年 2 月 24 日之前建立包含網路監視器資源的 IAM 政策，請注意下列網路監視器資源和資源類型的變更。
>
> - HealthEvents資源已重新命名為HealthEvent。
> - HealthEvent資源的 ARN 和正則表達式格式已更新。
> - Monitor 資源的 ARN 和 Regex 格式已更新。
> - GetHealthEvent動作的資源層級權限現在僅支援HealthEvent資源類型。Monitor 資源不支援該類型。
> - TagResourceUntagResource、和的「監視」資源類型已更新ListTagsForResource為必要項目。
>
> 若要查看有關可在政策中指定的動作、資源和條件金鑰的詳細資訊，以管理網際網路監控中的 AWS 資源存取權，請參閱 Amazon CloudWatch 網際網路監控器的動作、資源和條件金鑰。

目錄

- Amazon CloudWatch 互聯網監控器如何與 IAM 工作
- AWS Amazon CloudWatch 網際網路監視器的管理
- Amazon CloudWatch 互聯網監控器的 IAM 許可
- 服務鏈接的角色 Amazon CloudWatch 互聯網監控

Amazon CloudWatch 互聯網監控器如何與 IAM 工作

在您使用 IAM 管理網路監視器的存取權之前，請了解可搭配使用網路監視器的 IAM 功能。

若要查看顯示 AWS 服務如何與大多數 IAM 功能搭配運作的類似高階檢視的表格，請參閱 IAM 使用者指南中的搭配 IAM 使用的AWS 服務。

您可以使用 IAM 功能與 Amazon CloudWatch 互聯網監視器

IAM 功能	網路監視器支援
身分型政策	是
資源型政策	否
政策動作	是
政策資源	是
政策條件索引鍵 (服務特定)	是
ACL	否
ABAC(政策中的標籤)	部分
臨時憑證	是
主體許可	是
服務角色	否
服務連結角色	是

以身分為基礎的網路監視器政策

支援身分型政策	是

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。

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。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

網路監視器內以資源為基礎的政策

支援以資源基礎的政策	否

資源型政策是附加到資源的 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。

網路監視器的政策動作

支援政策動作	是

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。也有一些作業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予執行相關聯動作的許可。

若要查看網際網路監視器動作清單，請參閱服務授權參考中的 Amazon CloudWatch 網際網路監視器定義的動作。

網路監視器中的政策動作會在動作之前使用下列字首：

```
internetmonitor
```

如需在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "internetmonitor:action1",
      "internetmonitor: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 Describe 文字的所有動作，請包含以下動作：

```
"Action": "internetmonitor:Describe*"
```

網路監視器的政策資源

支援政策資源	是

在《服務授權參考》中，您可以看到下列與網路監視器相關的資訊：

- 若要查看網際網路監視器資源類型及其 ARN 的清單，請參閱 Amazon CloudWatch 網際網路監控器定義的資源。
- 若要了解您可以使用每個資源的 ARN 指定的動作，請參閱 Amazon CloudWatch 網際網路監控器定義的動作。

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。

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 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

網路監視器的政策條件金鑰

支援服務特定政策條件金鑰	是

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。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。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。

若要查看網際網路監視器條件金鑰清單，請參閱服務授權參考中的 Amazon CloudWatch 網際網路監視器的條件金鑰。若要了解您可以使用條件金鑰的動作和資源，請參閱 Amazon CloudWatch 網際網路監控器定義的動作。

網路監視器中的 ACL

支援 ACL	否

存取控制清單 (ACL) 可控制哪些主體 (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。

ABAC 搭配網路監視器

支援 ABAC (政策中的標籤)	部分

網路監視器針對政策中的標籤提供部分支援。此程式支援標記一個資源、多個監視器。

若要搭配網際網路監視器使用標籤，請使用 AWS Command Line Interface 或 AWS SDK。網際網路監視器的標記不支援 AWS Management Console。

若要深入了解一般在政策中使用標籤的方式，請檢閱下列資訊。

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在中 AWS，這些屬性稱為標籤。您可以將標籤附加到 IAM 實體 (使用者或角色) 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

若要根據標籤控制存取，請使用 `aws: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。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的什麼是 ABAC?。如要查看含有設定 ABAC 步驟的教學課程，請參閱《IAM 使用者指南》中的使用屬性型存取控制 (ABAC)。

搭配網路監視器使用暫時憑證

支援臨時憑證	是

當您使用臨時憑據登錄時，某些 AWS 服務 不起作用。如需其他資訊，包括哪些 AWS 服務 與臨時登入資料搭配AWS 服務 使用，請參閱 IAM 使用者指南中的 IAM。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立暫時認證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱《IAM 使用者指南》中的切換至角色 (主控台)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

網路監視器的跨服務主體許可

支援轉寄存取工作階段 (FAS)	是

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》。

網路監視器的服務角色

支援服務角色	否

服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。

網路監視器的服務連結角色

支援服務連結角色	是

服務連結角色是一種連結至. 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。

如需網路監視器服務連結角色的詳細資訊，請參閱 服務鏈接的角色 Amazon CloudWatch 互聯網監控。

如需有關在中建立或管理服務連結角色的詳細資訊 AWS，請參閱使用 IAM 的AWS 服務。在表格中尋找服務，其中包含服務連結角色欄中的 Yes。選擇是連結，以檢視該服務的服務連結角色文件。

AWS Amazon CloudWatch 網際網路監視器的管理

受 AWS 管理的策略是由建立和管理的獨立策略 AWS。 AWS 受管理的策略旨在為許多常見使用案例提供權限，以便您可以開始將權限指派給使用者、群組和角色。

請記住， AWS 受管理的政策可能不會為您的特定使用案例授與最低權限權限，因為這些權限可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的客戶管理政策，以便進一步減少許可。

您無法變更受 AWS 管理策略中定義的權限。如果 AWS 更新 AWS 受管理原則中定義的權限，則此更新會影響附加原則的所有主體識別 (使用者、群組和角色)。 AWS 當新的啟動或新 AWS 服務 的 API 操作可用於現有服務時，最有可能更新 AWS 受管理策略。

如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

AWS 受管理的策略： CloudWatchInternetMonitorServiceRolePolicy

此原則會附加至名為的服務連結角色，AWSServiceRoleForInternetMonitor以允許 Internet Monitor 存取帳戶中的資源，例如 Amazon Virtual Private Cloud 資源或網路負載平衡器，以便您在建立監視器時可以選取這些資源。如需詳細資訊，請參閱 服務鏈接的角色 Amazon CloudWatch 互聯網監控。

Amazon CloudWatch 互聯網監控器的 IAM 許可

若要存取在 Amazon CloudWatch 網際網路監視器中使用監視器和資料的動作，使用者必須擁有正確的許可。

如需 Amazon 中安全性的詳細資訊 CloudWatch，請參閱Amazon 的身份和訪問管理 CloudWatch。

Amazon CloudWatch 網際網路監視器的唯讀存取權限

若要存取唯讀動作以使用 Amazon CloudWatch Internet Monitor 中的監視器和資料，使用者必須以具有下列權限的使用者或角色登入：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "internetmonitor:Get*",
                "internetmonitor:List*",
                "internetmonitor:StartQuery",
                "internetmonitor:StopQuery",
                "logs:DescribeLogGroups",
                "logs:GetQueryResults",
                "logs:StartQuery",
                "logs:StopQuery"
                ],
            "Resource": "*"
        }
    ]
}
```

在 Amazon CloudWatch 互聯網監視器完全訪問許可

若要在 Amazon CloudWatch 網際網路監視器中建立監視器，並且要完整存取使用網際網路監控器中的監視器和資料的動作，使用者必須使用具有下列權限的使用者或角色登入：

- 建立與 Internet Monitor 相關聯的服務連結角色的許可。如需詳細資訊，請參閱 服務鏈接的角色 Amazon CloudWatch 互聯網監控。
- 允許進行完整存取的動作許可，以便在 Internet Monitor 中使用監視器和資料。

> **Note**
>
> 如果建立更嚴格的身分型許可政策，則採取該政策的使用者可能沒有在 Internet Monitor 中建立和使用監視器和資料的完整存取權。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "internetmonitor:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
internetmonitor.amazonaws.com/AWSServiceRoleForInternetMonitor",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "internetmonitor.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
internetmonitor.amazonaws.com/AWSServiceRoleForInternetMonitor"
        },
        {
```

```
            "Action": [
                "ec2:DescribeVpcs",
                "elasticloadbalancing:DescribeLoadBalancers",
                "workspaces:DescribeWorkspaceDirectories",
                "cloudfront:GetDistribution"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

服務鏈接的角色 Amazon CloudWatch 互聯網監控

Amazon CloudWatch 網際網路監控器使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至網路監視器的一種特殊 IAM 角色類型。服務連結的角色是由 Internet Monitor 預先定義，包含服務代表您呼叫其他服 AWS 務所需的所有權限。

網路監視器會定義此服務連結角色的許可，除非另外定義，否則只有網路監視器才能擔任此角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

您必須先刪除與角色相關的資源，才能刪除角色。此限制可保護您的網路監視器資源，避免您不小心移除資源的存取許可。

如需關於支援服務連結角色的其他服務的資訊，請參閱可搭配 IAM 運作的AWS 服務，尋找 Service-Linked Role (服務連結角色) 欄中顯示為 Yes (是) 的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

網路監視器的服務連結角色許可

網際網路監視器使用名為AWSServiceRoleForInternetMonitor的服務連結角色。此角色可讓網際網路監控器存取帳戶中的資源，例如 Amazon Virtual Private Cloud 資源、Amazon CloudFront 分發、Amazon WorkSpaces 目錄和網路負載平衡器，以便在建立監視器時選取這些資源。

此服務連結角色使用受管理策略`CloudWatchInternetMonitorServiceRolePolicy`。

服AWSServiceRoleForInternetMonitor務連結角色會信任下列服務擔任該角色：

- `internetmonitor.amazonaws.com`

若要檢視此原則的權限，請參閱AWS 受管理CloudWatchInternetMonitorServiceRolePolicy的策略參考中的。

建立網路監視器的服務連結角色

您不需要為網路監視器手動建立服務連結角色。您第一次創建一個監視器，互聯網監視器AWSServiceRoleForInternetMonitor為您創建。

如需詳細資訊，請參閱《IAM 使用者指南》中的「建立服務連結角色」。

編輯網路監視器的服務連結角色

網路監視器在帳戶中建立服務連接角色後，您就無法再變更角色名稱，因為有各種實體可能會參考該服務連結角色。您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱 IAM 使用者指南中的編輯服務連結角色。

刪除網路監視器的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，在手動刪除服務連結角色之前，您必須先清除服務連結角色的資源。

從 Internet Monitor 中的監視器中移除資源，然後刪除監視器之後，您可以刪除服務連結的角色AWSServiceRoleForInternetMonitor。

> **Note**
>
> 若網路監視器服務正在使用您試圖刪除的角色，刪除可能會失敗。若發生此情況，請等待數分鐘後並再次嘗試。

使用 IAM 手動刪除服務連結角色

使用 IAM 主控台或 AWS API 刪除AWSServiceRoleForInternetMonitor服務連結角色。 AWS CLI如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

網路監視器服務連接角色更新

如需 Internet Monitor 服務連結角色的 AWS 受管理策略的更新，請參閱AWS 受管理策略的 CloudWatch 更新。AWSServiceRoleForInternetMonitor如需有關中受管理原則變更的自動警示 CloudWatch，請訂閱 CloudWatch 文件歷史記錄頁面上的 RSS 摘要。

在 Amazon CloudWatch 互聯網監控配額

Amazon CloudWatch 互聯網監視器具有以下配額。

資源	預設配額
每個區域的監視器數量	50
每個監視器的資源數量	50
已解決的網路監視器運作狀態事件的保留天數	400

使用 Amazon CloudWatch 網絡監控

Amazon CloudWatch Network Monitor 可讓您查看將 AWS 託管應用程式連接到現場部署目的地的網路效能，並可讓您在幾分鐘內識別任何網路效能降低的來源。Network Monitor 完全由 AWS管理。因此，不需要安裝其他代理程式來監控網路效能。您可以快速視覺化混合式網路連線的封包遺失和延遲，設定警示和閾值，然後採取行動改善最終使用者的網路體驗。

Network Monitor 適用於想要即時深入了解網路效能的網路營運商和應用程式開發人員。

主要功能

- 使用 Network Monitor 透過持續的即時封包遺失和延遲指標來測試不斷變化的混合網路環境。
- 當您使用連線時 AWS Direct Connect，網路監視器會將網路 Health 指示器寫入您的 CloudWatch 帳戶，快速診斷 AWS 網路效能降級。此指標提供機率評分，以判斷網路降低是否發生在 AWS範圍內。
- Network Monitor 透過完全受管的代理程式方法提供順暢的監控，這表示無需在 VPC 或內部部署中安裝代理程式。只需要指定 VPC 子網路和內部部署 IP 地址即可開始。
- 網路監視器將指標發佈至 CloudWatch 「指標」 可以建立儀表板來檢視指標，並針對特定於應用程式的指標建立可操作的閾值和警示。

如需詳細資訊，請參閱the section called “Network Monitor 如何運作”。

Network Monitor 術語和要素

- 監視器 – 監視器會顯示相關資源，以便您檢視網路效能和可用性測量結果，並獲取運作狀態事件提醒。當您建立應用程式的監視器時，您可以將 AWS 託管資源新增為網路來源。然後，「網路監視器」會建立 AWS 託管資源和目標 IP 位址之間所有可能探查的清單。

- 探查 — 探查是指從 AWS 託管資源傳送到內部部署目的地 IP 位址的流量。網路監視器指標會針對監視器中設定的每個探查，寫入您的 CloudWatch 帳戶中。
- AWS 網路來源 — 這是網路監視器探查的原 AWS 始來源，這將是任何 VPC 中的子網路。
- 目的地 – 這是 AWS 網路來源內部部署網路中的目標。目的地是內部部署 IP 地址、網路通訊協定、連接埠和網路封包大小的組合。支援 IPv4 和 IPv6 地址。

Network Monitor 限制和要求

- Network Monitor 最多支援四個目的地 IP 地址，每個監視器最多 24 個探查。
- 每個區域每個帳戶最多可擁有 100 個監視器。
- 監視子網路必須擁有與監視器相同的帳戶。
- 網路監視器不會在發生網路問題時提供自動 AWS 網路容錯移轉。
- 您建立的每個探查都會收取費用。如需定價詳情，請參閱 the section called “定價”。

Amazon 網 CloudWatch 絡監視器的工作

透過提供完全受管且無代理解決方案，Network Monitor 讓監控變得更容易。當您在 AWS 託管資源中建立監視器時， AWS 會在背景中建立並管理所有基礎結構，以執行往返時間和封包遺失測量。因此，您可以快速擴展監控，而無需在 AWS 基礎結構中安裝或解除安裝任何代理程式。

網路監視器專注於監控來自 AWS 託管資源的流程所採取的路由，而不是廣泛地監視來自 AWS 區域. 如果您的工作負載分散在多個可用區域 (AZ)，則 Network Monitor 可以監控來自每個私有子網路的路由。

Network Monitor 會根據您在建立監視器時設定的彙總間隔，將往返時間和封包遺失指標發佈到您的 Amazon CloudWatch 帳戶。您也可以使用來為每個監視器設定個別的延遲和封包遺失閾值 CloudWatch。例如，如果封包遺失平均值高於封包遺失敏感工作負載的靜態 0.1% 閾值，可以建立警示來通知您。您也可以使用 CloudWatch 異常偵測，針對封包遺失或延遲指標超出所需範圍的警示。

可用性和效能測量

網路監視器會將定期的作用中探查從您的 AWS 資源傳送到內部部署目的地。建立監視器時，將指定下列值：

- 彙總間隔。 CloudWatch 接收量測結果的時間 (以秒為單位)。這將是每 30 秒或 60 秒一次。您為監視器選擇的彙總間隔會套用至該監視器中的所有探查。

- 探查通訊協定。新增至監視器的每個探查都必須使用網際網路控制訊息通訊協定 (ICMP) 或傳輸控制通訊協定 (TCP)。如需詳細資訊，請參閱the section called “通訊協定”。
- 封包大小。在單一探查中， AWS 託管資源和目的地之間傳輸的每個封包的大小 (以位元組為單位)。監視器中的每個探查都可以擁有自己的封包大小。

對於指標，

- 往返時間指標 (以毫秒為單位) 會測量並記錄效能測量結果，並記錄探查傳輸至目的地 IP 地址及收到其相關回應所花費的時間。
- 封包遺失指標會測量已傳送封包總數的百分比，並記錄未收到相關回應的已傳輸探查數目，這意味著這些封包在網路路徑中實際上遺失了。

支援的通訊協定

以 ICMP 為基礎的探查會將 AWS 託管資源的 ICMP 回應要求傳送至目的地位址，並預期 ICMP 回應會從目的地位址回覆。Network Monitor 使用 ICMP 回應請求和回覆訊息中的資訊來計算往返時間和封包遺失指標。

TCP 型探查會將 TCP SYN 封包從您的 AWS 託管資源傳送到目的地位址和連接埠，並預期會從目的地位址和連接埠傳回 TCP SYN+ACK 或 RST 封包。Network Monitor 使用 TCP SYN 和 TCP SYN+ACK 或 RST 訊息中的資訊來計算往返時間和封包遺失指標。此外，Network Monitor 會定期切換來源 TCP 連接埠以增加網路涵蓋範圍，進而提高偵測封包遺失的機率。

AWS 網路 Health 指示器

Network Monitor 會發佈網路運作狀態指示器 (NHI) 指標，它可提供透過 AWS Direct Connect連線之目的地的網路效能和可用性資訊。此指標是從 AWS 託管資源 (部署監視器的位置) 到「直 Connect」位置之 AWS 受控網路路徑健全狀況的統計度量。

Network Monitor 採用異常偵測來計算網路路徑上的可用性下降或效能降低。

> **Note**
>
> 每當您建立新的監視器、新增探測或重新啟動探測時，該監視器的 NHI 都會延遲幾個小時，以便 AWS 收集資料以執行異常偵測。

為了提供 NHI 運作狀態指標，Network Monitor 會在 AWS 範例資料集之間套用統計關聯性，並套用於模擬網路路徑之流量的封包遺失和往返延遲指標。指標可以是兩個變數之一：1 或 0。值 1 表示網路監視器在 AWS 受控制的網路路徑中觀察到網路效能降低。值 0 表示 Network Monitor 沒有觀察到路徑中的任何網路降低。這可讓您更快速地對網路問題進行疑難排解。您可以在 NHI 指標中設定警示，以通知網路路徑中持續發生的問題。

支援 IPv4 和 IPv6 地址

Network Monitor 可透過 IPv4 或 IPv6 網路提供可用性和效能指標，並可從雙堆疊 VPC 中監控 IPv4 或 IPv6 地址。Network Monitor 不允許在同一個監視器中設定 IPv4 和 IPv6 目的地，但是您可以單獨建立僅限 IPv4 和僅限 IPv6 目的地。

區域可用性

網路監視器目前可在下列項目中使用 AWS 區域：

區域	
亞太區域 (香港)	ap-east-1
亞太區域 (孟買)	ap-south-1
亞太區域 (首爾)	ap-northeast-2
亞太區域 (新加坡)	ap-southeast-1
亞太區域 (雪梨)	ap-southeast-2
亞太區域 (東京)	ap-northeast-1
加拿大西部 (卡加利)	ca-west-1

區域	
歐洲 (法蘭克福)	eu-central-1
歐洲 (愛爾蘭)	eu-west-1
歐洲 (倫敦)	eu-west-2
歐洲 (巴黎)	eu-west-3
歐洲 (斯德哥爾摩)	eu-north-1
中東 (巴林)	me-south-1
南美洲 (聖保羅)	sa-east-1
美國東部 (維吉尼亞北部)	us-east-1
美國東部 (俄亥俄)	us-east-2
美國西部 (加州北部)	us-west-1
美國西部 (奧勒岡)	us-west-2

建立 Network Monitor

下列步驟說明如何建立監視器，然後新增必要的探查。對於探查，您可以選擇來源子網路，以及每個監視器最多 24 個探查的最多四個目的地 IP 地址。可以使用 Amazon CloudWatch 主控台或使用命令列或 API 來建立監視器。

主題

- 使用主控台建立 Network Monitor
- 使用命令列或 API 建立網路監視器

使用主控台建立 Network Monitor

下列步驟說明使用 Amazon CloudWatch 主控台建立監視器。選擇來源子網路，然後新增最多四個目的地，為每個監視器建立最多 24 個探查。可以使用 Amazon CloudWatch 主控台或使用命令列或 SDK 來建立監視器。

> **Important**
>
> 這些步驟旨在一次全部完成。您無法儲存任何處理中的工作，以便稍後繼續。

定義監視器詳細資訊

建立監視器的第一步是定義基本詳細資料。這包括為監視器命名並定義彙總時段。可以將可選標籤新增到監視器。

定義監視器詳細資訊

1. 請在 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台，然後在 [網路監控] 下，選擇 [網路監視器]。
2. 選擇 Create monitor (建立監視器)。
3. 在監視器名稱中，輸入要用於此監視器的的名稱。
4. 對於「彙總」期間，選擇您要傳送量度的頻率 CloudWatch。可用的彙總時段為：
   - 30 秒
   - 60 秒

> **Note**
>
> 較短的彙總時段可加快偵測網路問題；不過，您選擇的彙總時段可能會影響帳單結構。如需有關定價的詳細資訊，請參閱 Amazon CloudWatch 定價頁面。

5. (選用) 在標籤區段中，新增索引鍵和值對，以進一步協助識別此資源，讓您可搜尋或篩選特定資訊。

   1. 選擇 Add new tag (新增標籤)。
   2. 輸入索引鍵名稱和關聯的值。
   3. 選擇新增新標籤以新增新標籤。

      您可以選擇新增新標籤來新增多個標籤，也可以選擇移除來移除任何標籤。
   4. 如果想要將標籤與監視器產生關聯，請選中將標籤新增至監視器建立的探查。這會將標籤新增至監視器探查，如果使用標籤式驗證或計量，這可能會很有幫助。

6. 選擇下一步以 the section called “選擇來源與目的地”。

選擇來源與目的地

網路監視器會使用網路運作所在區域中 VPC 和相關子網路的 AWS 來源。監視器目的地是內部部署 IP 地址、網路通訊協定、連接埠和網路封包大小的組合。

來源與目的地的組合稱為探查。每個子網路最多可以有四個探查，每個監視器最多可有 24 個探查。

> **Important**
>
> 這些步驟旨在一次全部完成。您無法儲存任何處理中的工作，以便稍後繼續。

選擇來源與目的地

1. 在 AWS 網路來源下，選擇要包含在監視器中的一個或多個子網路。可以選擇單一 VPC，然後選擇該 VPC 內的所有子網路，也可以選擇特定子網路。您選擇的 VPC 和子網路將成為網路監視器的來源。
2. 針對目的地 1，輸入內部部署網路的目的地 IP 地址。支援 IPv4 和 IPv6 地址。
3. 選擇 Advanced settings (進階設定)。

4. 針對此客戶管理的目的地，選擇網路通訊協定。它可以是：
   - ICMP
   - TCP
5. 如果通訊協定為 TCP，請輸入下列資訊。否則，請跳至下一步：
   1. 輸入您的網路用來連線的連接埠。連接埠必須是介於 1 到 65535 之間的數字。
   2. 輸入封包大小。這是在來源與目的地之間的探查中所傳送之每個封包的大小 (位元組)。封包大小必須是從 56 到 8500 之間的數字。
6. 選擇新增目的地，將另一個內部部署目的地新增至此監視器。為您要新增的每個目的地重複這些步驟。
7. 完成後，選擇下一步以確認探查。

確認探查

確認探查可讓您檢閱監視器的網路探查組合。此頁面會顯示您選擇的來源和目的地的所有可能組合。例如，如果您有六個來源子網路和四個目的地 IP，則可能總共有 24 個探查組合。

> **Important**
>
> - 這些步驟旨在一次全部完成。您無法儲存任何處理中的工作，以便稍後繼續。
> - 確認探查頁面不會指示探查是否有效。因此，建議您仔細檢閱此頁面，並刪除任何無效的探查。如果沒有移除無效探查，可能會向您收取費用。

確認監視器探查

1. 必要條件：the section called “選擇來源與目的地”。
2. 在確認探查頁面中，檢閱來源和目的地組合清單。
3. 選擇要從監視器中移除的一個或多個探查，然後選擇移除。

> **Note**
>
> 會提示您確認刪除。刪除探針之後，必須重新設定。您可以在 Network Monitor 頁面的網路監視器區段中將探針新增回監視器。如需詳細資訊，請參閱 the section called “將探針新增至監視器”。

4. 選擇下一步，在建立監視器之前檢閱監視器詳細資料。

檢閱和建立

建立監視器和探查的最後一個步驟是檢閱監視器和探查的詳細資料。此時您可以變更任何資訊。當您完成檢閱並建立監視器，且指標開始追蹤之後，就會開始針對任何探查向您收取費用。

> **Important**
>
> - 此步驟的設計目的是在建立監視器和探查時一次完成所有。您無法儲存任何處理中的工作，以便稍後繼續。
> - 如果選擇編輯任何區段，則需要從編輯的位置逐步完成監視器建立步驟。不過，無需重新建立任何後續步驟。這些頁面會保留先前填入的資訊。

檢閱和建立監視器

1. 在檢閱並建立探查頁面中，針對您要進行變更的任何區段選擇編輯。
2. 在該區段中進行任何變更。
3. 選擇下一步。
4. 執行下列任何一項：
   - 在其他監視器頁面中進行任何您想要的變更，然後選擇下一步，直到返回檢閱並建立頁面為止。
   - 如果沒有其他頁面需要變更，請選擇下一步，直到返回檢閱並建立頁面為止。
5. 選擇 Create monitor (建立監視器)。

   Network Monitor 頁面會在網路監視器區段中顯示目前的監視器建立狀態。在監視器建立期間，狀態為擱置中。當 [狀態] 變更為 [作用中] 時，您可以存取監視儀表板以檢視 CloudWatch 指標。

   如需有關使用監控儀表板的詳細資訊，請參閱 the section called “Network Monitor 儀表板”。

> **Note**
>
> 最近新增的網路監視器可能需要幾分鐘的時間才能開始收集網路指標。

使用命令列或 API 建立網路監視器

使用命令列或 API 來檢視和建立網路監視器。

使用命令列或 API 建立網路監視器

1. 使用 create-monitor 建立網路監視器。
2. 使用 create-probe 建立網路監視器探查。

使用 Network Monitor 監視器和探查

可以使用 Amazon CloudWatch 主控台或使用命令列或 API，透過監視器和探查執行下列任何任務。

主題：

- 編輯監視器
- 刪除監視器
- 啟用或停用探針
- 將探針新增至監視器
- 編輯探查
- 刪除探查
- 使用命令列或 API 標記或取消標記資源

編輯監視器

可以編輯 Network Monitor 的任何資訊，包括重新命名它，設定新的彙總時段，或者新增或移除標籤。變更監視器的資訊並不會變更任何關聯的探查。可以使用 Amazon CloudWatch 主控台或使用命令列或 API 來編輯監視器。

使用主控台編輯監視器

使用 CloudWatch 控制台編輯監視器。

使用主控台編輯監視器

1. 請在 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台，然後在 [網路監控] 下，選擇 [網路監視器]。
2. 在網路監視器區段中，選擇您要編輯的監視器。

3. 在監控儀表板頁面中，選擇編輯。
4. 在監視器名稱中，請輸入監視器的新名稱。
5. 對於「彙總」期間，選擇您要傳送量度的頻率 CloudWatch。有效時段為：
   - 30 秒
   - 60 秒

   > **Note**
   > 較短的彙總時段可加快偵測網路問題；不過，您選擇的彙總時段可能會影響帳單結構。如需有關定價的詳細資訊，請參閱 Amazon CloudWatch 定價頁面。

6. (選用) 在標籤區段中，新增索引鍵和值對，以進一步協助識別此資源，讓您可搜尋或篩選特定資訊。也可以僅變更當前任何索引鍵的值。
   1. 選擇 Add new tag (新增標籤)。
   2. 輸入索引鍵名稱和關聯的值。
   3. 選擇新增新標籤以新增新標籤。

      您可以選擇新增新標籤來新增多個標籤，也可以選擇移除來移除任何標籤。
   4. 如果想要將標籤與監視器產生關聯，請選中將標籤新增至監視器建立的探查。這會將標籤新增至監視器探查，如果使用標籤式驗證或計量，這可能會很有幫助。
7. 選擇儲存變更。

使用 CLI 或 API 編輯監視器

使用命令列或 API 來檢視和編輯監視器。

使用命令列或 API 編輯監視器

1. 如果您不知道監視器的名稱，請使用 list-monitors 來獲取監視器清單。記下您要編輯的監視器名稱。
2. 使用上一個步驟中的監視器名稱來使用 edit-monitor。

刪除監視器

無論監視器狀態為何，您都必須停用或刪除與該監視器相關聯的所有探查，才能刪除監視器。停用或刪除監視器後，無須再支付這些監視器探針的費用。無法復原已刪除的監視器。您可以使用 Amazon CloudWatch 控制台或使用命令行/API 刪除監視器。

雖然探查可能會刪除或停用，但 CloudWatch 仍會保留量度 15 天。

使用主控台刪除監視器

使用 CloudWatch 控制台刪除監視器。

使用主控台刪除監視器

1. 請在 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台，然後在 [網路監控] 下，選擇 [網路監視器]。
2. 在網路監視器區段中，選擇您要刪除的監視器。
3. 選擇動作，然後選擇刪除。
4. 如果您有任何作用中的探查，會提示您停用它們。選擇停用探查。

> **Note**
>
> 選擇停用探查後，無法取消或復原此動作。不過，不會從監視器中移除已停用的探查。稍後可以重新啟用它們。請參閱the section called “啟用或停用探針”。

5. 在確認欄位中輸入 **`confirm`**，然後選擇刪除。

使用命令列或 API 刪除監視器

使用命令列或 API 刪除監視器。

使用命令列或 API 刪除網路監視器

1. 將需要您要刪除之監視器的名稱。如果您不知道名稱，請使用 list-monitors 來獲取監視器清單。記下您要刪除的監視器名稱。
2. 確認該監視器是否包含任何探查。使用上一個步驟中的監視器名稱來使用 edit-monitor。這會傳回與該監視器相關聯的任何探查清單。
3. 如果監視器包含探查，您首先需要將這些探查設定為非作用中或刪除它們。

- 若要將探查設定為非作用中，請使用 update-probe，並將狀態設定為 INACTIVE。
- 若要刪除探查，請使用 delete-probe。

4. 將探查設定為 INACTIVE 或刪除之後，請使用 delete-monitor 來刪除監視器。不會刪除非作用中的探查。

啟用或停用探針

可以視需要啟用或停用監視器探查。如果您目前沒有使用探查，可能會想要停用它，但未來可能會想要再次使用它。透過停用探查，將不需要花時間重新設定它。已停用的探查不會計費。

您可以使用 Amazon CloudWatch 主控台或使用命令列或 API 來變更監視器的狀態。

使用主控台，將探查設定為作用中或非作用中

使用主 CloudWatch 控台將探查設定為作用中或非作用中。

使用主控台，將探查設定為作用中或非作用中

1. 請在 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台，然後在 [網路監控] 下，選擇 [網路監視器]。
2. 選擇監視器詳細資料索引標籤。
3. 在探查區段中，選擇您要啟用或停用的探查。
4. 選擇動作，然後選擇啟用或停用。

> **Note**
>
> 如果要重新啟用已停用的探查，將開始對該探查產生帳單費用。

使用命令列或 API，將探查設定為作用中或非作用中

使用命令列或 API，將探查設定為作用中或非作用中，或者停用。只能將此命令用於單一探查。

使用命令列或 API，將探查設定為作用中或非作用中

1. 如果您不知道監視器的名稱，請使用 list-monitors 來獲取監視器清單。請記下您想要變更其探查狀態的監視器名稱。

2. 使用上一個步驟中的監視器名稱來使用 edit-monitor。這會傳回與該監視器相關聯的任何探查清單。請記下您想要變更其狀態的探查 ID。
3. 使用 update-probe 並將您想要變更其狀態的探查設定為 ACTIVE 或 INACTIVE。

將探針新增至監視器

可以將探查新增至現有監視器。請注意，如果將任何探查新增至監視器，帳單結構將會更新，以顯示已新增的新探查。

使用主控台將探查新增至監視器

使用主控台將探查新增至監視器

1. 請在 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台，然後在 [網路監控] 下，選擇 [網路監視器]。
2. 在網路監視器區段中，執行下列其中一個動作：
   - 選擇您要為其新增探查的監視器的名稱連結。選擇監視器詳細資料索引標籤，然後在探查區段中選擇新增探查。
   - 選擇監視器核取方塊，選擇動作，然後選擇新增探查。
3. 在新增探查頁面中，執行下列操作：
   1. 在 AWS 網路來源下，選擇要為其新增監視器的子網路。

      > **Note**
      >
      > 一次只能新增一個探查，每個監視器最多四個探查。

   2. 輸入內部部署網路的目的地 IP 地址。支援 IPv4 和 IPv6 地址。
   3. 選擇 Advanced settings (進階設定)。
   4. 選擇目的地的網路通訊協定。它可以是 ICMP 或 TCP。
   5. 如果通訊協定為 TCP，請輸入下列資訊。否則，請跳至下一步：
      - 輸入您的網路用來連線的連接埠。連接埠必須是介於 1 到 65535 之間的數字。
      - 輸入封包大小。這是在來源與目的地之間的探查中所傳送之每個封包的大小 (位元組)。封包大小必須是從 56 到 8500 之間的數字。
4. (選用) 在標籤區段中，新增索引鍵和值對，以進一步協助識別此資源，讓您可搜尋或篩選特定資訊。

1. 選擇 Add new tag (新增標籤)。
2. 輸入索引鍵名稱和關聯的值。
3. 選擇新增新標籤以新增新標籤。

   您可以選擇新增新標籤來新增多個標籤，也可以選擇移除來移除任何標籤。

5. 選擇新增探查。

   啟動探查時，狀態會顯示擱置中。探查可能需要幾分鐘的時間才能變為作用中。

使用命令列或 API 將探查新增至監視器

使用命令列或 API 將探查新增至監視器。一次新增單一探查時，只能使用此命令。

使用命令列或 API 將探查新增至監視器

1. 如果您不知道監視器的名稱，請使用 list-monitors 來獲取監視器清單。記下您要新增探查的監視器名稱。
2. 使用 create-probe 將探查新增至監視器。

編輯探查

可以變更目前探查的任何資訊，無論該探查已啟用還是停用。可以使用 Amazon CloudWatch 主控台或使用命令列或 API 來編輯探查。

使用主控台編輯探查

使用主控台編輯探查

1. 請在 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台，然後在 [網路監控] 下，選擇 [網路監視器]。

   選擇名稱連結以開啟監控儀表板。

2. 選擇監視器詳細資料索引標籤。
3. 在探查區段中，選擇您要編輯之探查的連結。
4. 在探查儀表板頁面中，選擇編輯，或選擇動作，然後選擇編輯。
5. 在編輯探查頁面中，輸入新的目的地探查 IP 地址。支援 IPv4 和 IPv6 地址。

6. 選擇 Advanced settings (進階設定)。
7. 選擇網路通訊協定。它可以是 ICMP 或 TCP。
8. 如果通訊協定為 TCP，請輸入下列資訊。否則，請跳至下一步：
   - 輸入您的網路用來連線的連接埠。連接埠必須是介於 1 到 65535 之間的數字。
   - 輸入封包大小。這是在來源與目的地之間的探查中所傳送之每個封包的大小 (位元組)。封包大小必須是從 56 到 8500 之間的數字。
9. (選用) 在標籤區段中，新增索引鍵和值對，以進一步協助識別此資源，讓您可搜尋或篩選特定資訊。
   1. 選擇 Add new tag (新增標籤)。
   2. 輸入索引鍵名稱和關聯的值。
   3. 選擇新增新標籤以新增新標籤。

      您可以選擇新增新標籤來新增多個標籤，也可以選擇移除來移除任何標籤。
10. 選擇儲存變更。

使用命令列或 API 編輯探查

使用命令列編輯監視器探查。只能將此命令用於單一探查。

使用命令列或 API 編輯探查

1. 如果您不知道監視器的名稱，請使用 list-monitors 來獲取監視器清單。請記下您想要變更其探查狀態的監視器名稱。
2. 使用上一個步驟中的監視器名稱來使用 edit-monitor。這會傳回與該監視器相關聯的任何探查清單。記下您要編輯之任何探查的探查 ID。
3. 使用 update-probe 來變更探查的資訊。

刪除探查

如果您知道未來不再需要探查，則可以刪除探查而不是停用它。您無法復原已刪除的探查，而需要重新進行建立。刪除探查時，該探查的計費會停止。可以使用 Amazon CloudWatch 主控台、命令列或 API 來刪除探查。

使用主控台刪除探查

使用主控台刪除探查

1. 請在 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台，然後在 [網路監控] 下，選擇 [網路監視器]。
2. 在網路監視器區段中，選擇名稱連結以開啟監控儀表板。
3. 選擇監視器詳細資料索引標籤。
4. 選擇監視器核取方塊，選擇動作，然後選擇刪除。
5. 在刪除探查對話方塊中，選擇刪除以確認要刪除探查。
6. 選擇刪除以確認您要刪除探查。

   探查區段中的探查狀態會顯示正在刪除。刪除之後，就會從探查區段中移除該探查。

使用命令列或 API 刪除探查

使用命令列或 API 刪除探查。只能將此命令用於單一探查。

使用命令列或 API，將探查設定為作用中或非作用中

1. 如果您不知道監視器的名稱，請使用 list-monitors 來獲取監視器清單。記下您要刪除的帶有探查的監視器名稱
2. 使用上一個步驟中的監視器名稱來使用 edit-monitor。這會傳回與該監視器相關聯的任何探查清單。記下您要刪除之任何探查的探查 ID。
3. 使用 delete-probe。

使用命令列或 API 標記或取消標記資源

可以使用命令列或 CLI 來新增或更新資源標籤。

使用命令列或 API 更新網路監視器標籤

- 若要列出資源標籤，請使用list-tags-for-resources。
- 若要標記資源，請使用 tag-resource。
- 若要取消標記資源，請使用 untag-resource。

Network Monitor 儀表板

您可以使用 Amazon CloudWatch 網路監控儀表板來檢視 AWS 網路健康狀態，以及探查往返時間和封包遺失。可以檢視監視器和個別探查的這些指標。

Network Monitor 儀表板

- 監控儀表板
- 探查儀表板

探查警示

您可以根據 Amazon CloudWatch 網路監視器指標建立 Amazon CloudWatch 警示，就像對其他 Amazon CloudWatch 指標一樣。觸發警示時，您建立的任何警示都會顯示在 [網路監視器] 儀表板的 [監視器詳細資料] 區段的偵測的 [狀態] 欄中。狀態將為 [正常] 或 [在警報中]。如果偵測未顯示任何狀態，則不會為該探查建立警示。

例如，可以根據 Network Monitor 指標 `PacketLoss` 建立警示，並將其設定為在指標低於您選擇的值時傳送通知。您可以依照與其他度量相同的準則來設定網路監視器 CloudWatch 度量的警示。

為網路監視器建立 CloudWatch 警示`AWS/NetworkMonitor`時，提供下列指標。

- HealthIndicator
- PacketLoss
- RTT (往返時間)

如需建立 Network Monitor 警示的步驟，請參閱 the section called “建立以靜態閾值為基礎的警示”。

設定指標時間範圍

兩個儀表板上的指標和事件使用的預設時間為兩小時 (從目前時間開始計算)。您可以將預設值變更為使用下列其中一個預設集：

- 1h – 一小時
- 2h – 兩小時
- 1d – 一天
- 1w – 一週

也可以設定自訂時間範圍。選擇自訂，選擇絕對或相對時間，然後將時間範圍設定為您自己選擇的時間。相對時間僅支援從今天的日期返回 15 天，每個 CloudWatch 預設值。

此外，還可以根據 UTC 時區或當地時區選擇圖表中顯示的時間。

監控儀表板

您可以使用 Amazon CloudWatch 網路監控儀表板來檢視 AWS 網路健康狀態，以及探查往返時間和封包遺失。Network Monitor 具有用於監視器和探查的儀表板。

若要存取監控儀表板

1. 請在 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台，然後在 [網路監控] 下，選擇 [網路監視器]。
2. 在網路監視器區段中，選擇名稱連結以開啟監控儀表板。

概觀

概觀頁面會顯示以下監控資訊：

- AWS 網路健全狀況 — AWS 網路健全狀況只會顯示 AWS 網路的整體健康狀況。狀態將為正常運作或已降級。[狀況良好] 狀態表示網路監視器未發現任何 AWS 網路問題。[降級] 狀態表示網路監視器發現網 AWS 路發生問題。此區段中的狀態列會顯示一小時預設時間內網路的狀態。將滑鼠移至狀態列的任一點上可檢視其他詳細資料。
- 探查流量摘要 — 顯示監視器中來源 AWS 子網路與目標 IP 位址之間的流量目前狀態。探查流量摘要會顯示下列內容：
  - 警示中的探查 – 此數字表示有多少探查處於降級狀態。觸發設定為警示的指標時，就會觸發警示。如需有關網路監視器度量警示的資訊，請參閱the section called “探查警示”。
  - 封包遺失 – 從來源子網路到目的地 IP 地址遺失的封包數目。此值表示為傳送的封包總數的百分比。
  - 往返時間 – 來源子網路中的封包到達目的地 IP 地址然後再回來所需的時間 (毫秒)。

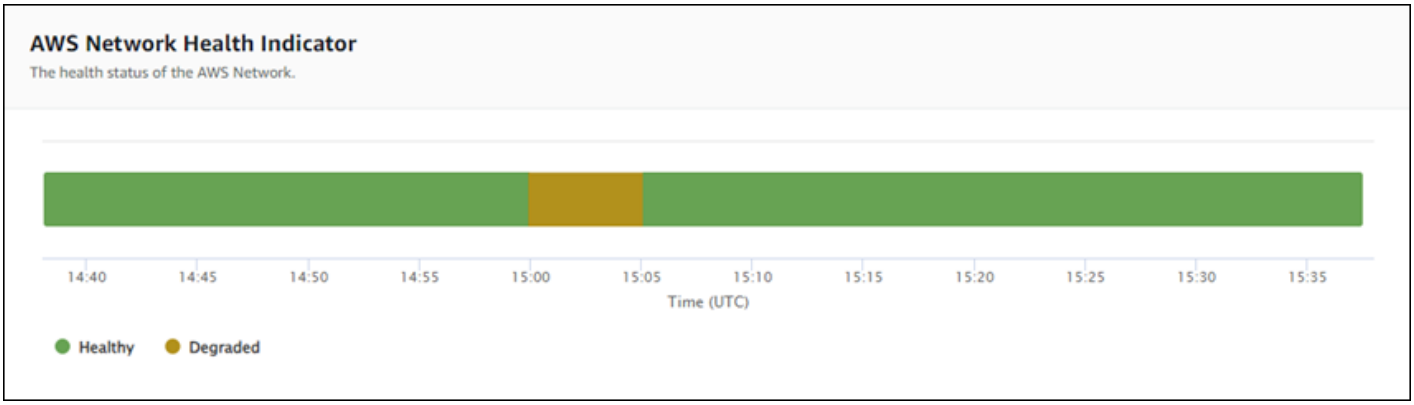
使用交互式圖表來表示資料，可讓您查看詳細資訊。

根據預設，資料可顯示兩小時，從目前日期和時間開始計算。但是，可以變更範圍以符合您的需求。如需詳細資訊，請參閱 the section called “設定指標時間範圍”。

追蹤指標

Network Monitor 儀表板會顯示監視器和探查的圖形表示。下列圖表可供使用：

- AWS 網路 Health 指示器 — 代表 AWS 網路在指定期間內的健全狀況。狀態將為正常運作或已降級。在下列範例中，您會看到從世界標準時間 15:00 到 15:05 UTC， AWS 網路處於降級狀態。15:05 之後，網路回到正常運作狀態。將滑鼠移至圖表的任何部分可檢視其他詳細資訊。



> **Note**
>
> 「網路 Health 指示器」不會指示探查的健全狀況，而只會指出 AWS 網路。

- 封包遺失 – 此圖表顯示唯一的線條，顯示監視器中每個探查的封包遺失百分比。頁面底部的圖例會顯示監視器中的每個探查，並用顏色編碼以確保唯一性。將滑鼠移至此圖表中的探查上會顯示來源子網路、目的地 IP 以及封包遺失百分比。在下列範例中，針對從子網路到 IP 地址 127.0.0.1 的探查設定封包遺失警示。當超過探查的封包遺失閾值時，就會觸發警示。將滑鼠移至圖表上可顯示探查來源和目的地，並顯示此探查在 11 月 21 日 02:41:30 的封包遺失達 30.97%。

- 往返時間 – 此圖表顯示每個探查的一條線，顯示每個探查的往返時間。頁面底部的圖例會顯示監視器中的每個探查，並用顏色編碼以確保唯一性。將滑鼠移至此圖表中的探查上會顯示來源子網路、目的地 IP 地址以及往返時間。下列範例顯示，在 11 月 21 日星期二 21:45:30，從子網路到 IP 地址 127.0.0.1 的探查往返時間為 0.075 秒。

監視器詳細資訊

監視器詳細資訊頁面顯示監視器的詳細資訊，包括探查。在此頁面中，可以管理標籤或新增探查。本頁分為下列三個部分：

- 監視器詳細資訊 – 此頁面提供監視器的詳細資訊。無法編輯此部分中的資訊。不過，您可以選擇角色名稱連結，檢視 Network Monitor 服務連結角色的詳細資訊。
- 探查 – 此部分顯示與監視器相關聯之所有探查的清單。選擇 VPC 或子網路 ID 連結，在 Amazon VPC 主控台中開啟 VPC 或子網路詳細資訊。也可以修改探查，包括啟用或停用它。如需詳細資訊，請參閱 the section called “使用監視器和探查”。

  偵測區段會顯示為該監視器設定之每個探查的相關資訊，包括探查 ID、VPC ID、子網路 ID、IP 位址、通訊協定，以及探查狀態為作用中還是非作用中。如果您已為探測設定警示，則會顯示該警示的目前狀態。確定表示沒有指標事件觸發任何警示；在警示中表示您在中設定的度量 CloudWatch 觸發了警示。如果偵測未顯示任何狀態，則表示未設定任何 CloudWatch 警示。如需有關您可以建立的網路監視器探測警示類型的資訊，請參閱the section called “探查警示”。
- 標籤 – 檢視監視器的目前標籤。透過選擇管理標籤來新增或移除標籤。這會開啟編輯探查頁面。如需有關編輯標籤的詳細資訊，請參閱 the section called “編輯監視器”。

探查儀表板

您可以使用 Amazon CloudWatch Network Monitor 儀表板來檢視 AWS 網路健康狀態，以及特定探測的特定往返時間和封包遺失的相關資訊。有兩個探查儀表板：概觀和探查詳細資訊。

您可以建立 CloudWatch 警示來設定封包遺失和往返時間測量結果臨界值。達到指標的臨界值時，CloudWatch 警示會通知您。如需有關建立探查警示的詳細資訊，請參閱 the section called “探查警示”。

存取探查儀表板

1. 請在 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台，然後在 [網路監控] 下，選擇 [網路監視器]。
2. 在網路監視器區段中，選擇名稱連結以開啟監控儀表板。
3. 選擇 ID 連結可檢視探查儀表板。

概觀

概觀頁面會顯示以下探查資訊：

- AWS 網路 Health 指示器詳細資料 — 這只提供 AWS 網路的整體健全狀況。狀態將為正常運作或已降級。[降級] 狀態表示 AWS 網路發生問題，並不表示您的探查是否有問題。
- 封包遺失 – 從來源子網路到此探查的目的地 IP 地址遺失的封包數目。

- 往返時間 – 來源子網路中的封包到達目的地 IP 地址然後再回來所需的時間 (毫秒)。

探查詳細資訊

探查詳細資訊頁面會顯示有關探查的詳細資訊。可在此頁面中編輯探查。如需詳細資訊，請參閱 the section called “使用監視器和探查”。

- 探查詳細資訊 – 此頁面提供有關探查的一般資訊。無法編輯此部分中的資訊。
- 探查來源和目的地 – 此區段顯示有關探查的詳細資訊。選擇 VPC 或子網路 ID 連結，在 Amazon VPC 主控台中開啟 VPC 或子網路詳細資訊。也可以修改探查，包括啟用或停用它。
- 標籤 – 檢視監視器的目前標籤。透過選擇管理標籤來新增或移除標籤。這會開啟編輯探查頁面。如需有關編輯標籤的詳細資訊，請參閱 the section called “編輯探查 ”。

Network Monitor 配額

以下是 Network Monitor 配額：

配額	預設	可調整
每個帳戶的最大監視器數量 AWS 區域	100	是
每個監視器的探查上限	24	是
每個監視器每個子網路的探查上限	4	是

Network Monitor 中的資料安全和資料保護

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，這些架構是為了滿足對安全性最敏感的組織的需求而建置的。

安全是 AWS 與您之間共同承擔的責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端安全性 — AWS 負責保護中執行 AWS 服務的基礎架構 AWS 雲端。 AWS 還為您提供可以安全使用的服務。若要了解適用於 Amazon CloudWatch Network Monitor 的合規計劃，請參閱合規計劃AWS 服務範圍內的合規計劃AWS的服務。

- 雲端中的安全性 — 您的責任取決於您使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件可協助您瞭解如何在使用「 CloudWatch 網路監視器」時套用共用的責任模型。下列主題說明如何設定 CloudWatch 網路監視器，以符合您的安全性和合規性目標。您也會學到如何使用其他 AWS 服務來協助您監視和保護您的 CloudWatch 網路監視器資源。

主題

- 在 Amazon CloudWatch 網絡監控器數據保護
- Amazon CloudWatch 網絡監視器的基礎設施

在 Amazon CloudWatch 網絡監控器數據保護

AWS 共同責任模型適用於 Amazon CloudWatch 網路監控器中的資料保護。如此模型中所述， AWS 負責保護執行所有 AWS 雲端. 您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

基於資料保護目的，我們建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 登入資料並設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源進行通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用設定 API 和使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案，以及其中的所有默認安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在透過命令列介面或 API 存取時需要經 AWS 過 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用主控台、API 或 AWS SDK AWS 服務 使用 CloudWatch 網路監視器或其他工作時。 AWS CLI您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

Amazon CloudWatch 網絡監視器的基礎設施

作為受管服務，Amazon CloudWatch 網路監視器受到 Amazon Web Services：安 AWS 全流程概觀白皮書中所述的全球網路安全程序的保護。

您可以使用 AWS 已發佈的 API 呼叫透過 CloudWatch 網路存取網路監視器。用戶端必須支援 Transport Layer Security (TLS) 1.0 或更新版本。建議使用 TLS 1.2 或更新版本。用戶端也必須支援具備完美轉送私密 (PFS) 的密碼套件，例如臨時 Diffie-Hellman (DHE) 或橢圓曲線臨時 Diffie-Hellman (ECDHE)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

Amazon CloudWatch 網絡監控器的身份和訪問管理

AWS Identity and Access Management (IAM) 是一項可協助管理員安全地控制 AWS 資源存取的 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 CloudWatch 網路監視器資源。IAM 是一項無需額外付費即可使用的 AWS 服務。您可以使用 IAM 的功能，來允許其他使用者、服務和應用程式完整地或有所限制地使用您的 AWS 資源，而不共享您的安全登入資料。

根據預設，IAM 使用者不具有建立、檢視或修改 AWS 資源的許可。若要允許 IAM 使用者存取資源 (例如全球網路) 和執行任務，您必須：

- 建立 IAM 政策以准許使用者使用他們所需的特定資源和 API 動作
- 將政策附接至 IAM 使用者或連線到使用者所屬的群組

將政策附加至使用者或使用者群組時，政策會允許或拒絕使用者對特定資源執行特定任務的許可。

條件索引鍵

`Condition` 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是可選用的。您可以建置使用條件運算子的條件表達式 (例如等於或小於)，來比對原則中的條件和請求中的值。如需詳細資訊，請參閱《AWS Identity and Access Management 使用者指南》中的 IAM JSON 政策元素：條件運算子。

若您在陳述式中指定多個 `Condition` 元素，或是在單一 `Condition` 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。

您可以將標籤附加至 CloudWatch 網路監視器資源，或將要求中的標籤傳遞至 Cloud WAN。若要根據標籤控制存取，請使用 `aws: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。如需詳細資訊，請參閱《AWS Identity and Access Management 使用者指南》中的 IAM JSON 政策元素：條件。

若要查看所有 AWS 全域條件金鑰，請參閱 AWS Identity and Access Management 使用者指南中的 AWS 全域條件內容金鑰。

標記核心網路資源

標籤是您或 AWS 指派給資源的中繼 AWS 資料標籤。每個標籤皆包含索引鍵與值。對於您指派的標籤，您可以定義索引鍵與值。例如，對於某個資源，您可能將索引鍵定義為 `purpose`，以及將值定義為 `test`。標籤可協助您執行以下操作：

- 識別和組織您的 AWS 資源。許多 AWS 服務都支援標記，因此您可以將相同標籤指派給來自不同服務的資源，以指出資源是相關的。
- 控制對 AWS 資源的存取。如需詳細資訊，請參閱《AWS 識別與存取管理使用指南》中的使用標籤控制 AWS 資源的存取。

Amazon CloudWatch 網絡監控器如何使用 IAM

在您使用 IAM 管理 CloudWatch 網路監視器的存取權限之前，請先了解哪些 IAM 功能可與 CloudWatch 網路監視器搭配使用。

您可以搭配 Amazon CloudWatch 網路監視器使用的 IAM 功能

IAM 功能	CloudWatch 網路監視器支援
身分型政策	是
資源型政策	否
政策動作	是
政策資源	是
政策條件索引鍵	是

IAM 功能	CloudWatch 網路監視器支援
ACL	否
ABAC(政策中的標籤)	部分
臨時憑證	是
主體許可	是
服務角色	否
服務連結角色	是

若要深入瞭解 CloudWatch 網路監視器和其他 AWS 服務如何搭配大多數 IAM 功能運作，請參閱 IAM 使用者指南中的搭配 IAM 使用的AWS 服務。

Amazon 網 CloudWatch 絡監控器的基於身份的政策

支援身分型政策	是

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。

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。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

網路監視器的身分識別原則範例 CloudWatch

若要檢視 CloudWatch 網路監視器以身分識別為基礎的原則範例，請參閱。Amazon 的基於身份的政策示例 CloudWatch

CloudWatch 網路監視器內的資源型政策

支援以資源基礎的政策	否

資源型政策是附加到資源的 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

若要啟用跨帳戶存取，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，作為資源型政策的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主體和資源位於不同時 AWS 帳戶，受信任帳戶中的 IAM 管理員也必須授與主體實體 (使用者或角色) 權限，才能存取資源。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 角色與資源型政策有何差異。

CloudWatch 網路監視器的原則動作

支援政策動作	是

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。也有一些作業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予執行相關聯動作的許可。

若要查看 CloudWatch 網路監視器動作清單，請參閱服務授權參考中的 Amazon CloudWatch 網路監視器定義的動作。

CloudWatch 網路監視器中的原則動作會在動作之前使用下列前置詞：

```
networkmonitor
```

若要在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "networkmonitor:action1",
      "networkmonitor:action2"
         ]
```

若要檢視 CloudWatch 網路監視器以身分識別為基礎的原則範例，請參閱。Amazon 的基於身份的政策示例 CloudWatch

CloudWatch 網路監視器的原則資源

支援政策資源	是

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 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

若要查看 CloudWatch 網路監視器資源類型及其 ARN 的清單，請參閱服務授權參考中的 Amazon CloudWatch 網路監控器定義的資源。若要了解可以使用哪些動作指定每個資源的 ARN，請參閱 Amazon CloudWatch 網路監視器定義的動作。

CloudWatch 網路監視器的原則條件金鑰

支援服務特定政策條件金鑰	是

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。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。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。

若要查看 CloudWatch 網路監視器條件金鑰清單，請參閱服務授權參考中的 Amazon CloudWatch 網路監視器的條件金鑰。若要了解可以使用條件金鑰的動作和資源，請參閱 Amazon CloudWatch 網路監控器定義的動作。

CloudWatch 網路監視器中的 ACL

支援 ACL	否

存取控制清單 (ACL) 可控制哪些主體 (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。

ABAC 與 CloudWatch 網絡監視器

支援 ABAC (政策中的標籤)	部分

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在中 AWS，這些屬性稱為標籤。您可以將標籤附加到 IAM 實體 (使用者或角色) 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

若要根據標籤控制存取，請使用 `aws: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。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的什麼是 ABAC?。如要查看含有設定 ABAC 步驟的教學課程，請參閱《IAM 使用者指南》中的使用屬性型存取控制 (ABAC)。

搭配 CloudWatch 網路監視器使用臨時認證

支援臨時憑證	是

當您使用臨時憑據登錄時，某些 AWS 服務 不起作用。如需其他資訊，包括哪些 AWS 服務 與臨時登入資料搭配AWS 服務 使用，請參閱 IAM 使用者指南中的 IAM。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立暫時認證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱《IAM 使用者指南》中的切換至角色 (主控台)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

網路監視器的跨服務主 CloudWatch 體權限

支援轉寄存取工作階段 (FAS)	是

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》。

CloudWatch 網路監視器的服務角色

支援服務角色	否

服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。

> **Warning**
>
> 變更服務角色的權限可能會中斷 CloudWatch 網路監視器功能。只有當 CloudWatch 網路監視器提供指引時，才編輯服務角色。

使用網路監視器的服務 CloudWatch 連結角色

支援服務連結角色	是

服務連結角色是一種連結至. 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。

如需建立或管理服務連結角色的詳細資訊，請參閱可搭配 IAM 運作的AWS 服務。在表格中尋找服務，其中包含服務連結角色欄中的 Yes。選擇是連結，以檢視該服務的服務連結角色文件。

網路監視器的身分識別原則範例 CloudWatch

根據預設，使用者和角色沒有建立或修改 CloudWatch 網路監視器資源的權限。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 來執行工作。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。

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》中的建立 IAM 政策。

有關 CloudWatch 網路監視器定義的動作和資源類型的詳細資訊，包括每種資源類型的 ARN 格式，請參閱服務授權參考中的 Amazon CloudWatch Network Monitor 的動作、資源和條件金鑰。

主題

- 政策最佳實務
- 使用 CloudWatch 網路監視器主控台
- 允許使用者檢視他們自己的許可
- 疑難排解 CloudWatch 網路監視器身分識別

政策最佳實務

以身分識別為基礎的原則會決定某人是否可以建立、存取或刪除您帳戶中的 CloudWatch 網路監視器資源。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。如需更多資訊，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。
- 套用最低許可許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的權限。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。如需使用 IAM 套用許可的更多相關資訊，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。如需更多資訊，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。如需更多資訊，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。若要在呼叫 API 作業時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，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 CloudWatch 網路監視器主控台

若要存取 Amazon CloudWatch 網路監控器主控台，您必須擁有最少一組許可。這些權限必須允許您列出並檢視您的 AWS 帳戶. CloudWatch 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。

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，只需允許存取符合他們嘗試執行之 API 操作的動作就可以了。

若要確保使用者和角色仍可使用 CloudWatch 網路監視器主控台，請同時將 CloudWatch 網路監視器*ConsoleAccess*或*ReadOnly* AWS 受管理的原則附加至實體。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。

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": "*"
        }
    ]
}
```

疑難排解 CloudWatch 網路監視器身分識別

使用下列資訊可協助您診斷並修正使用 CloudWatch 網路監視器和 IAM 時可能會遇到的常見問題。

主題

- 我沒有授權在 CloudWatch 網絡監視器中執行操作
- 我沒有授權執行 iam：PassRole
- 我想允許我以外的人訪問我 AWS 帳戶 的 CloudWatch 網絡監視器資源

我沒有授權在 CloudWatch 網絡監視器中執行操作

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

下列範例錯誤會在mateojackson IAM 使用者嘗試使用主控台檢視一個虛構 *my-example-widget* 資源的詳細資訊，但卻無虛構 networkmonitor:*GetWidget* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 networkmonitor:GetWidget on resource: my-example-widget
```

在此情況下，必須更新 mateojackson 使用者的政策，允許使用 networkmonitor:*GetWidget* 動作存取 *my-example-widget* 資源。

如果您需要協助，請聯絡您的 AWS 系統管理員。您的管理員提供您的簽署憑證。

我沒有授權執行 iam：PassRole

如果您收到未授權執行iam:PassRole動作的錯誤訊息，則必須更新您的原則，以允許您將角色傳遞給 CloudWatch 網路監視器。

有些 AWS 服務 允許您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為的 IAM 使用者marymajor嘗試使用主控台在 CloudWatch 網路監視器中執行動作時，就會發生下列範例錯誤。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。

如果您需要協助，請聯絡您的 AWS 系統管理員。您的管理員提供您的簽署憑證。

我想允許我以外的人訪問我 AWS 帳戶 的 CloudWatch 網絡監視器資源

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物件取得該角色。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的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要瞭解 CloudWatch 網路監視器是否支援這些功能，請參閱Amazon 如何與 IAM 合 CloudWatch 作。
- 若要了解如何提供對您所擁有資源 AWS 帳戶 的存取權，請參閱 IAM 使用者指南中您擁有的另一 AWS 帳戶 個 IAM 使用者提供存取權限。
- 若要了解如何將資源存取權提供給第三方 AWS 帳戶，請參閱 IAM 使用者指南中的提供第三方 AWS 帳戶 擁有的存取權。
- 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 CloudWatch 網路監視器的受管理原則

若要新增使用者、群組和角色的權限，使用 AWS 受管理的原則比自己撰寫原則更容易。建立 IAM 客戶受管政策需要時間和專業知識，而受管政策可為您的團隊提供其所需的許可。若要快速開始使用，您可以使用我們的 AWS 受管政策。這些政策涵蓋常見使用案例，並可在您的 AWS 帳戶中使用。如需 AWS 受管政策的詳細資訊，請參閱 IAM 使用者指南中的AWS 受管政策。

AWS 服務會維護和更新 AWS 受管理的策略。您無法變更 AWS 受管理原則中的權限。服務偶爾會在 AWS 受管政策中新增其他許可以支援新功能。此類型的更新會影響已連接政策的所有身分識別 (使用者、群組和角色)。當新功能啟動或新操作可用時，服務很可能會更新 AWS 受管政策。服務不會從 AWS 受管理的政策移除權限，因此政策更新不會破壞您現有的權限。

此外，還 AWS 支援跨多個服務之工作職能的受管理原則。例如，`ReadOnlyAccess` AWS 受管理的策略提供對所有 AWS 服務和資源的唯讀存取權。當服務啟動新功能時，會為新作業和資源新 AWS 增

唯讀權限。如需任務職能政策的清單和說明，請參閱 IAM 使用者指南中有關任務職能的AWS 受管政策。

AWS 受管理的策略： CloudWatchNetworkMonitorServiceRolePolicy

會CloudWatchNetworkMonitorServiceRolePolicy附加至服務連結角色，該角色可讓服務代表您執行動作，並存取與 CloudWatch 網路監視器相關聯的資源。無法將此政策附接到 IAM 身分。如需詳細資訊，請參閱 the section called “服務連結角色”。

CloudWatch AWS 受管理策略的網路監控更新

自此服務於 2023 年 11 月開始追蹤這些變更以來，檢視 CloudWatch 網路監控的 AWS 受管原則更新詳細資料。

變更	描述	日期
CloudWatchNetworkMonitorServiceRolePolicy：新政策。	新的策略添加到 CloudWatch 網絡監視器。	2023 年 11 月 27 日
the section called “AWSServiceRoleForNetworkMonitor ”。新角色。	新角色添加到 CloudWatch 網絡監視器。	2023 年 11 月 27 日

CloudWatch 網路監視器的 IAM 許可

若要使用 Amazon CloudWatch 網路監視器，使用者必須擁有正確的許可。

如需 Amazon 中安全性的詳細資訊 CloudWatch，請參閱Amazon 的身份和訪問管理 CloudWatch。

檢視監視器所需的許可

若要在中檢視 Amazon CloudWatch Network Monitor 的監視器 AWS Management Console，您必須以具有下列權限的使用者或角色登入：

```
{
"Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "networkmonitor:Get*",
                "networkmonitor:List*"
                ],
            "Resource": "*"
        }
    ]
}
```

建立監視器所需的許可

若要在 Amazon CloudWatch 網路監視器中建立監視器，使用者必須擁有建立與網路監視器相關聯的服務連結角色的權限。若要進一步了解服務連結角色，請參閱 使用網路監視器的服務 CloudWatch 連結角色。

若要在中建立 Amazon CloudWatch Network Monitor 的監視器 AWS Management Console，您必須以具有下列政策中包含許可的使用者或角色身分登入。

> **Note**
>
> 若您建立更嚴格的身分型許可政策，則採取該政策的使用者將無法建立監視器。

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "networkmonitor:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
networkmonitor.amazonaws.com/AWSServiceRoleForNetworkMonitor",
            "Condition": {
```

```
                "StringLike": {
                    "iam:AWSServiceName": "networkmonitor.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:GetRole",
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
networkmonitor.amazonaws.com/AWSServiceRoleForNetworkMonitor"
        },
        {
            "Action": [
                "ec2:CreateSecurityGroup",
                "ec2:CreateNetworkInterface",
                "ec2:CreateTags"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

使用網路監視器的服務 CloudWatch 連結角色

Amazon CloudWatch 網路監控器會針對代表您呼叫其他 AWS 服務所需的許可，使用下列服務連結角色：

- AWSServiceRoleForNetworkMonitor

AWSServiceRoleForNetworkMonitor

CloudWatch 網路監控會使用名為的服務連結角色AWSServiceRoleForNetworkMonitor來更新和管理 CloudWatch 網路監視器。

AWSServiceRoleForNetworkMonitor 服務連結角色信任下列服務來擔任此角色：

- networkmonitor.amazonaws.com

CloudWatchNetworkMonitorServiceRolePolicy 會附接到服務連結角色，並授予服務存取權，以存取您帳戶中的 VPC 和 EC2 資源，以及管理已建立的網路監視器。

許可分組

政策會分組為以下許可集：

- **cloudwatch**-這可讓服務主體將網路監視指標發佈至 CloudWatch 資源。
- **ec2** - 允許服務主體描述您帳戶中的 VPC 和子網路，以建立或更新監視器和探針。這也允許服務主體建立、修改和刪除安全群組、網路介面及其相關權限，以設定監視器或探查，以便將監控流量傳送至端點。

如需有關政策的詳細資訊，請參閱「the section called “AWS 受管理政策”」。

以下顯示 CloudWatchNetworkMonitorServiceRolePolicy：

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "PublishCw",
   "Effect": "Allow",
   "Action": "cloudwatch:PutMetricData",
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "cloudwatch:namespace": "AWS/NetworkMonitor"
    }
   }
  },
  {
   "Sid": "DescribeAny",
   "Effect": "Allow",
   "Action": [
    "ec2:DescribeNetworkInterfaces",
    "ec2:DescribeNetworkInterfaceAttribute",
    "ec2:DescribeVpcs",
    "ec2:DescribeNetworkInterfacePermissions",
    "ec2:DescribeSubnets",
    "ec2:DescribeSecurityGroups"
   ],
   "Resource": "*"
```

```
  },
  {
   "Sid": "DeleteModifyEc2Resources",
   "Effect": "Allow",
   "Action": [
    "ec2:AuthorizeSecurityGroupEgress",
    "ec2:CreateNetworkInterfacePermission",
    "ec2:DeleteNetworkInterfacePermission",
    "ec2:RevokeSecurityGroupEgress",
    "ec2:ModifyNetworkInterfaceAttribute",
    "ec2:DeleteNetworkInterface",
    "ec2:DeleteSecurityGroup"
   ],
   "Resource": [
    "arn:aws:ec2:*:*:network-interface/*",
    "arn:aws:ec2:*:*:security-group/*"
   ],
   "Condition": {
    "StringEquals": {
     "aws:ResourceTag/ManagedByCloudWatchNetworkMonitor": "true"
    }
   }
  }
 ]
}
```

建立服務連結角色

AWSServiceRoleForNetworkMonitor

您無須手動建立 AWSServiceRoleForNetworkMonitor 角色。

- CloudWatch 網路監視器會AWSServiceRoleForNetworkMonitor在您建立第一個網路監視器時建立角色。此角色將套用至您建立的任何後續監視器。

若要代表您建立服務連結角色，您必須具有必要的許可。如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

編輯服務連結角色

可使用 IAM 來編輯 AWSServiceRoleForNetworkMonitor 描述。如需更多資訊，請參閱 IAM 使用者指南中的編輯服務連線角色。

刪除服務連結角色

如果您不再需要使用 CloudWatch 網路監視器，建議您刪除AWSServiceRoleForNetworkMonitor角色。

只有在刪除網路監視器之後，才能刪除這些服務連結角色。如需有關刪除網路監視器的詳細資訊，請參閱刪除網路監視器。

您可以使用 IAM 主控台、IAM CLI 或 IAM API 刪除服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的刪除服務連結角色。

刪除AWSServiceRoleForNetworkMonitor CloudWatch 網絡監視器後，當你創建一個新的監視器將再次創建角色。

CloudWatch 網路監視器服務連結角色的支援區域

CloudWatch 網路監視器支援服務連結的 AWS 區域 所有服務可用的角色。如需詳細資訊，請參閱 AWS 一般參考 中的 AWS 端點。

刪除服務連結角色

如果您不再需要使用 CloudWatch 網路監視器，建議您刪除AWSServiceRoleForNetworkMonitor角色。

只有在刪除網路監視器之後，才能刪除這些服務連結角色。如需有關刪除網路監視器的詳細資訊，請參閱刪除網路監視器。

您可以使用 IAM 主控台、IAM CLI 或 IAM API 刪除服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的刪除服務連結角色。

刪除AWSServiceRoleForNetworkMonitor CloudWatch 網絡監視器後，當你創建一個新的監視器將再次創建角色。

定價

使用 Amazon CloudWatch 網路監控器，無需預付費用或長期承諾。Network Monitor 的定價包含下列兩個要素：

- 受監控資源的每小時費用，以及
- CloudWatch 指標費用。

當您建立網路監視器時，可以將資源與要監控的網路監視器建立關聯。對於網絡監視器，這些將是您 Amazon Virtual Private Cloud （VPC）中的子網絡。每個受監控資源可讓您建立從 VPC 中的每個子網路到四個目的地的最多四個探查。為了協助控制帳單，可以減少受監控資源的數量，以調整子網路涵蓋範圍和內部部署 IP 涵蓋範圍。

如需有關定價的詳細資訊，請參閱 Amazon CloudWatch 定價頁面。

基礎設施監控

本節中的主題說明可協助您取得 AWS 資源作業能見度的 CloudWatch 功能。

主題

- Container Insights
- Lambda Insights
- 使用貢獻者洞察分析高基數資料
- Amazon CloudWatch 應用洞察
- 使用主控台中的資源健全狀況檢 CloudWatch 視

Container Insights

使用 CloudWatch 容器深入解析，從容器化應用程式和微服務收集、彙總和摘要指標和記錄。Container Insights 適用於 Amazon Elastic Container Service (Amazon ECS)、Amazon Elastic Kubernetes Service (Amazon EKS) 及 Amazon EC2 上的 Kubernetes 平台。容器洞見支援從 AWS Fargate 針對 Amazon ECS 和 Amazon EKS 部署的叢集收集指標。

CloudWatch 自動收集許多資源的指標，例如 CPU、記憶體、磁碟和網路。Container Insights 還提供診斷資訊，例如容器重新啟動故障，協助您快速隔離和解決這些故障。您也可以針對容器深入解析收集的指標設定 CloudWatch 警示。

Container Insights 會使用內嵌指標格式將資料收集為效能日誌事件。這些效能日誌事件是使用結構化的 JSON 結構描述的項目，而該結構描述可擷取高基數資料並大量地進行存放。從此資料中，CloudWatch 在叢集、節點、網繭、工作和服務層級建立彙總 CloudWatch 度量做為指標。Container Insights 收集的指標可在 CloudWatch 自動儀表板中使用，也可在 CloudWatch 主控台的「指標」區段中檢視。指標要在容器任務執行一段時間後才會顯示。

當您部署 Container Insights 時，它會自動建立效能日誌事件的日誌群組。您不需要自行建立此日誌群組。

為了協助您管理容器洞察成本， CloudWatch 不會自動從日誌資料建立所有可能的指標。不過，您可以使用日誌深入分析來分析原始效能 CloudWatch 記錄事件，以檢視其他指標和其他精細度層級。

使用 Container Insights 的原始版本，收集的指標和擷取的指標會按自訂指標計費。使用 Container Insights 搭配 Amazon EKS 的增強可觀測性，Container Insights 指標和日誌會按觀測，而不是存放或擷取的指標計費。如需有關 CloudWatch 定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

在 Amazon EKS 和 Kubernetes 中，容器洞見使用容器化版本的 CloudWatch 代理程式來探索叢集中所有執行中的容器。然後，它會在效能堆疊的每一層上收集效能資料。

容器洞見支援使用其收集 AWS KMS key 的日誌和指標進行加密。若要啟用此加密，您必須手動為接收容器見解資料的記錄群組啟用 AWS KMS 加密。這會使 Container Insights 使用提供的 KMS 金鑰加密此資料。僅支援對稱金鑰。請勿使用非對稱 KMS 金鑰加密您的日誌群組。

如需詳細資訊，請參閱使用加密 CloudWatch 記錄檔中的記錄檔資料 AWS KMS。

Container Insights 搭配 Amazon EKS 的增強可觀測性

2023 年 11 月 6 日，推出了新版 Container Insights。此版本支援執行於 Amazon EC2 的 Amazon EKS 叢集的增強型可觀測性，並且可從這些叢集中收集更詳細的指標。安裝之後，它會自動收集 Amazon EKS 叢集的詳細基礎設施遙測和容器日誌。然後，您可使用經策管且立即可用的儀表板，來深入了解應用程式和基礎設施遙測。

Container Insights 搭配 Amazon EKS 的增強可觀測性，可收集容器層級的精細運作狀態、效能和狀態指標，以及控制平面指標。如需有關收集的額外指標和維度的詳細資訊，請參閱 Amazon EKS 和 Kubernetes Container Insights 指標。

如果您在 2023 年 11 月 6 日之後在 Amazon Amazon EC2 上的 Amazon EKS 叢集上使用 CloudWatch 代理程式安裝容器洞見，您就可以獲得具有增強 Amazon EKS 可觀察性的容器洞見。否則，您可遵循 正在升級至 Container Insights 搭配 Amazon EKS 的增強可觀測性 中的指示，將 Amazon EKS 叢集升級到此新版本。

容器洞見支援 CloudWatch 跨帳戶觀察能力。您可以使用單一監控帳戶來監控單一區域內跨多個 AWS 帳戶的應用程式並進行疑難排解。如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。

具有增強 Amazon EKS 可觀察性的容器洞見，也支援 Windows 工作者節點。

Fargate 不支援 Container Insights 搭配 Amazon EKS 的增強可觀測性。

> **Note**
>
> 您可導覽至 Container Insights 主控台，尋找是否有可升級為 Container Insights 搭配 Amazon EKS 的增強可觀測性的叢集。若要這麼做，請在 CloudWatch 主控台的導覽窗格中選擇 [深入解析]、[容器深入解析]。在 Container Insights 主控台中，橫幅會通知您是否有任何可升級的 Amazon EKS 叢集，以及升級頁面的連結。

支援平台

Container Insights 適用於 Amazon Elastic Container Service、Amazon Elastic Kubernetes Service 及 Amazon EC2 執行個體上的 Kubernetes 平台。

- 如為 Amazon ECS，Container Insights 會同時在 Linux 和 Windows Server 執行個體中收集叢集、任務和服務層級的指標。它只能於 Linux 執行個體中收集執行個體層級的指標。

  針對 Amazon ECS，只有 `bridge` 網路模式和 `awsvpc` 網路模式中的容器才能使用網路指標。`host` 網路模式中的容器無法使用這些指標。
- 如為 Amazon EC2 執行個體上的 Amazon Elastic Kubernetes Service 和 Kubernetes 平台，Container Insights 僅於 Linux 執行個體中支援。

CloudWatch 代理容器映像

Amazon 在 Amazon 彈性容器註冊表上提供了 CloudWatch 代理容器映像。如需詳細資訊，請參閱 Amazon ECR 上的 cloudwatch-agent。

支援地區

下列區域支援 Amazon ECS 的 Container Insights：

- 美國東部 (維吉尼亞北部)
- 美國東部 (俄亥俄)
- 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 (海德拉巴)
- 亞太區域 (雅加達)
- 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 (東京)

- 亞太區域 (悉尼)
- 加拿大西部 (卡加利)
- 加拿大 (中部)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- 歐洲 (米蘭)
- Europe (Paris)
- 歐洲 (西班牙)
- 歐洲 (斯德哥爾摩)
- 歐洲 (蘇黎世)
- Middle East (Bahrain)
- 中東 (阿拉伯聯合大公國)
- 南美洲 (聖保羅)
- AWS GovCloud (美國東部)
- AWS GovCloud (美國西部)
- 中國 (北京)
- 中國 (寧夏)

Amazon EKS 和 Kubernetes 支援的區域

下列區域支援 Amazon EKS 和 Kubernetes 的 Container Insights：

- 美國東部 (維吉尼亞北部)
- 美國東部 (俄亥俄)
- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 亞太區域 (香港)
- Asia Pacific (Mumbai)
- 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)
- 中國 (北京)
- 中國 (寧夏)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- 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)

設定 Container Insights

Amazon ECS、Amazon EKS 和 Kubernetes 的 Container Insights 設定程序是不同的。

主題

- 在 Amazon ECS 上設定 Container Insights
- 在 Amazon EKS 和 Kubernetes 上設定 Container Insights

在 Amazon ECS 上設定 Container Insights

您可以使用下列選項之一或兩個都用，啟用 Amazon ECS 叢集上的 Container Insights：

- 使用 AWS Management Console 或可開始收集叢集層次、作業層次和服務層次測量結果。 AWS CLI
- 將 CloudWatch 代理程式部署為精靈服務，以開始在 Amazon EC2 執行個體上託管的叢集上收集執行個體層級指標。

主題

- 在 Amazon ECS 上設定 Container Insights 以取得叢集和服務層級指標
- 使 AWS 用發行版在 Amazon ECS 上設置容器洞察 OpenTelemetry
- 部署 CloudWatch 代理程式以收集 Amazon ECS 上的 EC2 執行個體層級指標
- 部署發行 AWS 版 OpenTelemetry 以收集 Amazon ECS 叢集上的 EC2 執行個體層級指標
- 設定 FireLens 將記錄檔傳送至記 CloudWatch 錄

在 Amazon ECS 上設定 Container Insights 以取得叢集和服務層級指標

您可以在新的和現有的 Amazon ECS 叢集上啟用 Container Insights。Container Insights 會收集叢集、任務和服務層級的指標。您可以使用 Amazon ECS 主控台或. AWS CLI

如果您是在 Amazon EC2 執行個體上使用 Amazon ECS，且要從 Container Insights 收集網路和儲存體指標，請使用包含 Amazon ECS 代理 1.29 版的 AMI 啟動執行個體。如需更新代理程式版本的相關資訊，請參閱更新 Amazon ECS 容器代理程式

您可以使用設 AWS CLI 定帳戶層級權限，為帳戶中建立的任何新 Amazon ECS 叢集啟用容器洞見。若要執行此作業，請輸入以下命令。

```
aws ecs put-account-setting --name "containerInsights" --value "enabled"
```

> **Note**
>
> 如果您用於 Amazon ECS 容器洞見指標的客戶受管 AWS KMS 金鑰尚未設定為可使用 CloudWatch，則您必須更新金鑰政策以允許在日誌中加密日 CloudWatch 誌。您也必須將自己的 AWS KMS 金鑰與下的記錄群組產生關聯`/aws/ecs/containerinsights/`*`ClusterName`*`/performance`。如需詳細資訊，請參閱 CloudWatch 使用 AWS Key Management Service.

在現有的 Amazon ECS 叢集上設定 Container Insights

若要在現有的 Amazon ECS 叢集上啟用 Container Insights，請輸入以下命令。您必須執行的是 1.16.200 版或更新版本，下列命令才能運作。 AWS CLI

```
aws ecs update-cluster-settings --cluster myCICluster --settings
 name=containerInsights,value=enabled
```

在新的 Amazon ECS 叢集上設定 Container Insights

有兩種方式可在新的 Amazon ECS 叢集上啟用 Container Insights。您可以設定 Amazon ECS，讓 Container Insights 根據預設啟用所有的新叢集。否則，只能在建立新叢集時啟用該叢集。

使用 AWS Management Console

您可以依預設在所有新的叢集上開啟 Container Insights，或是在您建立個別叢集時將其開啟。

依預設在所有新的叢集上開啟 Container Insights

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽頁面中，選擇 Account Settings (帳戶設定)。
3. 選擇更新。
4. 若要針對叢集預設使用 CloudWatch 容器深入解析，請在CloudWatch容器深入解析下選取或清除CloudWatch 容器深入解析。
5. 選擇儲存變更。

如果您未使用上述程序，讓容器洞見在預設情況下於所有新叢集上啟用，請使用下列步驟來建立啟用容器洞見的叢集。

建立已開啟 Container Insights 的叢集

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇叢集。
3. 在 Clusters (叢集) 頁面上，選擇 Create cluster (建立叢集)。
4. 在 Cluster configuration (叢集組態) 下的 Cluster name (叢集名稱) 中，輸入唯一的名稱。

   名稱可以包含最多 255 個字母 (大小寫)、數字與連字號。
5. 若要開啟 Container Insights，請展開監控，然後開啟使用 Container Insights。

您現在可以在叢集中建立任務定義、執行任務，與啟動服務。如需詳細資訊，請參閱下列內容：

- 建立任務定義
- 執行任務
- 建立服務

使用在新的 Amazon ECS 叢集上設定容器洞見 AWS CLI

根據預設，若要在所有新叢集上啟用 Container Insights，請輸入以下命令。

```
aws ecs put-account-setting --name "containerInsights" --value "enabled"
```

如果您未使用上述命令，讓容器洞見在預設情況下在所有新叢集上啟用，輸入以下命令來建立已啟用容器洞見的新叢集。您必須執行 1.16.200 版或更新版本的 AWS CLI ，以下命令才能運作。

```
aws ecs create-cluster --cluster-name myCICluster --settings
 "name=containerInsights,value=enabled"
```

停用 Amazon ECS 叢集上的 Container Insights

若要停用現有 Amazon ECS 叢集上的 Container Insights，請輸入以下命令。

```
aws ecs update-cluster-settings --cluster myCICluster --settings
 name=containerInsights,value=disabled
```

使 AWS 用發行版在 Amazon ECS 上設置容器洞察 OpenTelemetry

如果您想要使用發行 AWS 版在 Amazon ECS 叢集上設 OpenTelemetry 定 CloudWatch容器洞見，請使用本節。有關開放遙測 AWS 發行版的更多信息，請參閱 AWS . OpenTelemetry

這些步驟假設您已經有一個執行 Amazon ECS 的叢集。有關將發行 AWS 版與 Amazon ECS 搭配使用開放遙測，以及為此目的設定 Amazon ECS 叢集的詳細資訊，請參閱在 Amazon 彈性容 OpenTelemetry 器服務中設定收集器的發行 AWS 版。

步驟 1：建立任務角色

第一個步驟是在收集 AWS OpenTelemetry器將使用的叢集中建立工作角色。

若要為 AWS 發行版建立工作角色 OpenTelemetry

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格中，選擇 Policies (政策)，然後選擇 Create policy (建立政策)。
3. 選擇 JSON 標籤，並複製下列政策：

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:DescribeLogGroups",
                "ssm:GetParameters"
            ],
            "Resource": "*"
        }
    ]
}
```

4. 選擇檢閱政策。
5. 對於名稱，輸入 **AWSDistroOpenTelemetryPolicy**，然後選擇 Create policy (建立政策)。
6. 在左側導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
7. 在服務清單中，選擇 Elastic Container Service。
8. 在頁面下方，選擇 Elastic Container Service Task (Elastic Container Service 任務)，然後選擇 Next: Permissions ((下一步：許可))。
9. 在策略清單中，搜尋AWSDistroOpenTelemetryPolicy。
10. 選取旁邊的核取方塊AWSDistroOpenTelemetryPolicy。
11. 選擇 Next: Tags (下一步：標籤)，然後選擇 Next: Review (下一步：檢閱)。
12. 針對 Role name (角色名稱)，輸入 **AWSOpenTelemetryTaskRole**，然後選擇 Create role (建立角色)。

步驟 2：建立任務執行角色

下一個步驟是為 AWS OpenTelemetry收集器建立任務執行角色。

為發行 AWS 版創建任務執行角色 OpenTelemetry

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 在服務清單中，選擇 Elastic Container Service。
4. 在頁面下方，選擇 Elastic Container Service Task (Elastic Container Service 任務)，然後選擇 Next: Permissions ((下一步：許可))。

5. 在政策清單中，搜尋 AmazonECs，然後選取 [AmazonECS TaskExecutionRolePolicy] 旁邊的核取方塊。TaskExecutionRolePolicy
6. 在策略清單中，搜尋CloudWatchLogsFullAccess並選取旁邊的核取方塊CloudWatchLogsFullAccess。
7. 在原則清單中，搜尋 AmazonSSM，然後選取 [亞馬遜 SSM ReadOnlyAccess] 旁邊的核取方塊。ReadOnlyAccess
8. 選擇 Next: Tags (下一步：標籤)，然後選擇 Next: Review (下一步：檢閱)。
9. 針對 Role name (角色名稱)，輸入 **`AWSOpenTelemetryTaskExecutionRole`**，然後選擇 Create role (建立角色)。

步驟 3：建立任務定義

下一步是建立任務定義。

要為 AWS 發行版創建任務定義 OpenTelemetry

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。
2. 在導覽窗格中，選擇 Task Definitions (任務定義)。
3. 選擇 Create new task definition (建立新任務定義)，以及 Create new task definitionN (建立新任務定義)。
4. 在任務定義系列中，請為任務定義指定唯一名稱。
5. 設定您的容器，然後選擇下一步。
6. 在指標和日誌記錄下，選取使用指標集合。
7. 選擇下一步。
8. 選擇建立。

如需將 AWS OpenTelemetry 收集器與 Amazon ECS 搭配使用的詳細資訊，請參閱在 Amazon 彈性容器服務中設定 OpenTelemetry 收集器發行 AWS 版。

步驟 4：執行任務

最後一個步驟是執行您建立的任務。

運行發行 AWS 版的任務 OpenTelemetry

1. 開啟主控台，網址為 https://console.aws.amazon.com/ecs/v2。

2. 在左側導覽窗格，選擇 Task Definitions (任務定義)，然後選取您剛建立的任務。
3. 選擇動作、部署、執行任務。
4. 選擇 Deploy (部署)、Run task (執行任務)。
5. 在運算選項區段中，從現有叢集中選擇叢集。
6. 選擇建立。
7. 接下來，您可以在 CloudWatch 控制台中檢查新的指標。
8. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
9. 在左側導覽窗格中，選擇 Metrics (指標)。

   您應該會看到 ECS/ ContainerInsights 命名空間。選擇該命名空間，然後您應該會看到八個指標。

部署 CloudWatch 代理程式以收集 Amazon ECS 上的 EC2 執行個體層級指標

若要部署 CloudWatch 代理程式以從 EC2 執行個體上託管的 Amazon ECS 叢集收集執行個體層級指標，請使用具有預設組態的快速入門設定，或手動安裝代理程式以便能夠進行自訂。

這兩種方法都要求您至少已經使用 EC2 啟動類型部署了一個 Amazon ECS 叢集，而且 CloudWatch 代理程式持續使用者必須能夠存取 Amazon EC2 執行個體中繼資料服務 (IMDS)。如需有關 IMDS 的詳細資訊，請參閱執行個體中繼資料與使用者資料。

這些方法也假設您已 AWS CLI 安裝。此外，若要在下列程序中執行命令，您必須登入具有 IAM FullAccess 和 Amazonec FullAccess s_ 政策的帳戶或角色。

主題

- 快速設置使用 AWS CloudFormation
- 手動和自訂設定

快速設置使用 AWS CloudFormation

若要使用快速設定，請輸入下列命令以用 AWS CloudFormation 來安裝代理程式。將 *cluster-name* 和 *cluster-region* 替換成您 Amazon ECS 叢集的名稱和區域。

這個命令會建立 IAM 角色。TaskRole ExecutionRole如果您的帳戶中已存在這些角色，請在輸入命令時改用 `ParameterKey=CreateIAMRoles,ParameterValue=False`，而不是使用 `ParameterKey=CreateIAMRoles,ParameterValue=True`。否則命令會失敗。

```
ClusterName=cluster-name
```

```
Region=cluster-region
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/daemon-service/cwagent-
ecs-instance-metric/cloudformation-quickstart/cwagent-ecs-instance-metric-cfn.json
aws cloudformation create-stack --stack-name CWAgentECS-${ClusterName}-${Region} \
    --template-body file://cwagent-ecs-instance-metric-cfn.json \
    --parameters ParameterKey=ClusterName,ParameterValue=${ClusterName} \
                 ParameterKey=CreateIAMRoles,ParameterValue=True \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${Region}
```

(替代) 使用您自己的 IAM 角色

如果您想要使用自己的自訂 ECS 任務角色和 ECS 任務執行角色，而不是使用 CWAGTECS TaskRole 和 CWAGTECS ExecutionRole 角色，請先確定已附加要用作 ECS 任務角色的角色。CloudWatchAgentServerPolicy此外，請確定要用作 ECS 任務執行角色的角色同時附加了CloudWatchAgentServerPolicy和 AmazonEC TaskExecutionRolePolicy S 原則。然後輸入下列命令。在指令中，以您自訂 ECS 任務角色的 ARN 取代 *task-role-arn*，並取代*execution-role-arn*為自訂 ECS 任務執行角色的 ARN。

```
ClusterName=cluster-name
Region=cluster-region
TaskRoleArn=task-role-arn
ExecutionRoleArn=execution-role-arn
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/daemon-service/cwagent-
ecs-instance-metric/cloudformation-quickstart/cwagent-ecs-instance-metric-cfn.json
aws cloudformation create-stack --stack-name CWAgentECS-${ClusterName}-${Region} \
    --template-body file://cwagent-ecs-instance-metric-cfn.json \
    --parameters ParameterKey=ClusterName,ParameterValue=${ClusterName} \
                 ParameterKey=TaskRoleArn,ParameterValue=${TaskRoleArn} \
                 ParameterKey=ExecutionRoleArn,ParameterValue=${ExecutionRoleArn} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${Region}
```

故障診斷快速設定

若要檢查 AWS CloudFormation 堆疊的狀態，請輸入以下指令。

```
ClusterName=cluster-name
Region=cluster-region
```

```
aws cloudformation describe-stacks --stack-name CWAgentECS-$ClusterName-$Region --
region $Region
```

如果您看到的 StackStatus 不是 CREATE_COMPLETE 或 CREATE_IN_PROGRESS，請檢查堆疊事件以找出錯誤。輸入以下命令。

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation describe-stack-events --stack-name CWAgentECS-$ClusterName-$Region
 --region $Region
```

若要檢查 cwagent 常駐程式服務的狀態，請輸入以下命令。在輸出中，您應該會看到 runningCount 等於 deployment 區段的 desiredCount。如果不相等，請檢查輸出的 failures 區段。

```
ClusterName=cluster-name
Region=cluster-region
aws ecs describe-services --services cwagent-daemon-service --cluster $ClusterName --
region $Region
```

您也可以使用 CloudWatch 記錄主控台來檢查代理程式記錄檔。尋找 /ecs/ ecs-cwagent-daemon-service 記錄群組。

刪除代 CloudWatch 理程式的 AWS CloudFormation 堆疊

如果您需要刪除 AWS CloudFormation 堆疊，請輸入以下指令。

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation delete-stack --stack-name CWAgentECS-${ClusterName}-${Region} --
region ${Region}
```

手動和自訂設定

請遵循本節中的步驟手動部署 CloudWatch 代理程式，從 EC2 執行個體上託管的 Amazon ECS 叢集收集執行個體層級指標。

必要的 IAM 角色和政策

需要兩個 IAM 角色。如果角色不存在，即必須建立角色。如需這些角色的詳細資訊，請參閱 IAM 任務角色和 Amazon ECS 任務執行角色。

- ECS 工作角色， CloudWatch 代理程式會使用此角色來發行量度。如果此角色已經存在，您必須確定它連接了 CloudWatchAgentServerPolicy 政策。
- ECS 任務執行角色，Amazon ECS 代理程式會使用此角色來啟動代理程式。 CloudWatch 如果此角色已經存在，您必須確定它連接了 AmazonECSTaskExecutionRolePolicy 和 CloudWatchAgentServerPolicy 政策。

如果您尚未擁有這些角色，您可以使用下列命令建立它們並連接必要的政策。第一個命令會建立 ECS 任務角色。

```
aws iam create-role --role-name CWAgentECSTaskRole \
    --assume-role-policy-document "{\"Version\": \"2012-10-17\",\"Statement\": [{\"Sid
\": \"\",\"Effect\": \"Allow\",\"Principal\": {\"Service\": \"ecs-tasks.amazonaws.com
\"},\"Action\": \"sts:AssumeRole\"}]}"
```

輸入上一個命令後，請將命令輸出Arn中的值記下為 "TaskRoleArn」。稍後建立任務定義時需要使用此值。然後輸入下列命令以連接必要的政策。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
CloudWatchAgentServerPolicy \
    --role-name CWAgentECSTaskRole
```

下一個命令會建立 ECS 任務執行角色。

```
aws iam create-role --role-name CWAgentECSExecutionRole \
    --assume-role-policy-document "{\"Version\": \"2012-10-17\",\"Statement\": [{\"Sid
\": \"\",\"Effect\": \"Allow\",\"Principal\": {\"Service\": \"ecs-tasks.amazonaws.com
\"},\"Action\": \"sts:AssumeRole\"}]}"
```

輸入上一個命令後，請將命令輸出Arn中的值記下為 "ExecutionRoleArn」。稍後建立任務定義時需要使用此值。然後輸入下列命令以連接必要的政策。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
CloudWatchAgentServerPolicy \
    --role-name CWAgentECSExecutionRole

aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy \
    --role-name CWAgentECSExecutionRole
```

建立任務定義並啟動常駐程式服務

建立作業定義，並使用它來啟動 CloudWatch 代理程式做為精靈服務。若要建立任務定義，請輸入下列命令。在第一行中，將預留位置取代為部署的實際值。 日*###*是 CloudWatch 日誌所在的區域，而*#########*所在的區域。*task-role-arn*是您正在使用之 ECS 任務角色的「Arn」，*execution-role-arn*也是 ECS 任務執行角色的「Arn」。

```
TaskRoleArn=task-role-arn
ExecutionRoleArn=execution-role-arn
AWSLogsRegion=logs-region
Region=cluster-region
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/daemon-service/cwagent-
ecs-instance-metric/cwagent-ecs-instance-metric.json \
    | sed "s|{{task-role-arn}}|${TaskRoleArn}|;s|{{execution-role-arn}}|
${ExecutionRoleArn}|;s|{{awslogs-region}}|${AWSLogsRegion}|" \
    | xargs -0 aws ecs register-task-definition --region ${Region} --cli-input-json
```

然後執行下列命令以啟動協助程式服務。將 *cluster-name* 和 *cluster-region* 替換成您 Amazon ECS 叢集的名稱和區域。

> **Important**
>
> 請先移除所有容量提供者策略，然後再執行此命令。否則，該命令將無法正常工作。

```
ClusterName=cluster-name
Region=cluster-region
aws ecs create-service \
    --cluster ${ClusterName} \
    --service-name cwagent-daemon-service \
    --task-definition ecs-cwagent-daemon-service \
    --scheduling-strategy DAEMON \
    --region ${Region}
```

如果看到此錯誤訊息 `An error occurred (InvalidParameterException) when calling the CreateService operation: Creation of service was not idempotent`，即表示您已經建立名為 `cwagent-daemon-service` 的協助程式服務。您必須使用下列命令作為範例，先刪除該服務。

```
ClusterName=cluster-name
Region=cluster-region
aws ecs delete-service \
    --cluster ${ClusterName} \
    --service cwagent-daemon-service \
    --region ${Region} \
    --force
```

(選用) 進階組態

或者，您可以使用 SSM 為託管在 EC2 執行個體上的 Amazon ECS 叢集中的 CloudWatch代理程式指定其他組態選項。選項如下：

- `metrics_collection_interval`— CloudWatch 代理程式收集指標的頻率 (以秒為單位)。預設為 60。範圍介於 1–172,000 之間。
- `endpoint_override` – (選用) 指定不同端點以傳送日誌。如果從 VPC 中的叢集發佈，且想要在 VPC 端點記錄資料，建議您這麼做。

  `endpoint_override` 的值必須是 URL 字串。
- `force_flush_interval` – 指定日誌在傳送到伺服器之前停留在記憶體緩衝區內的最長時間 (以秒為單位)。不論此欄位的設定如何，如果緩衝區中的日誌大小達到 1 MB，系統會立即將日誌傳送到伺服器。預設值為 5 秒。
- `region` – 根據預設，代理會將指標發佈至 Amazon ECS 容器執行個體所在的同一區域。若要覆寫此項目，您可在此指定不同的區域。例如 `"region" : "us-east-1"`

以下是自訂組態的範例：

```
{
    "agent": {
        "region": "us-east-1"
    },
    "logs": {
        "metrics_collected": {
            "ecs": {
                "metrics_collection_interval": 30
            }
        },
        "force_flush_interval": 5
    }
```

```
}
```

在 Amazon ECS 容器中自訂 CloudWatch 代理程式組態

1. 確保 Amazon SSM ReadOnlyAccess 政策已附加到您的亞馬遜 ECS 任務執行角色。您可以輸入以下命令來執行此操作。這個範例假設您的 Amazon ECS 任務執行角色為 C ExecutionRole wagEnteCS。如果使用不同的角色，請更換下列命令中的角色名稱。

   ```
   aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
   AmazonSSMReadOnlyAccess \
           --role-name CWAgentECSExecutionRole
   ```

2. 建立類似上述範例的自訂組態檔案。將此檔案命名為 `/tmp/ecs-cwagent-daemon-config.json`。

3. 執行下列命令，將此組態放入參數存放區。將 *`cluster-region`* 替換成您 Amazon ECS 叢集的區域。若要執行此命令，您必須登入具有 AmazonSS FullAccess M 政策的使用者或角色。

   ```
   Region=cluster-region
   aws ssm put-parameter \
       --name "ecs-cwagent-daemon-service" \
       --type "String" \
       --value "`cat /tmp/ecs-cwagent-daemon-config.json`" \
       --region $Region
   ```

4. 將任務定義檔案下載至本機檔案，例如 `/tmp/cwagent-ecs-instance-metric.json`

   ```
   curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
   insights/latest/ecs-task-definition-templates/deployment-mode/daemon-service/
   cwagent-ecs-instance-metric/cwagent-ecs-instance-metric.json -o /tmp/cwagent-ecs-
   instance-metric.json
   ```

5. 修改任務定義檔案。移除以下區段：

   ```
   "environment": [
                   {
                       "name": "USE_DEFAULT_CONFIG",
                       "value": "True"
                   }
               ],
   ```

   以下列內容取代該區段：

```
"secrets": [
                {
                    "name": "CW_CONFIG_CONTENT",
                    "valueFrom": "ecs-cwagent-daemon-service"
                }
            ],
```

6. 依照下列步驟重新啟動代理作為協助程式服務：

   a. 執行下列命令。

```
TaskRoleArn=task-role-arn
ExecutionRoleArn=execution-role-arn
AWSLogsRegion=logs-region
Region=cluster-region
cat /tmp/cwagent-ecs-instance-metric.json \
    | sed "s|{{task-role-arn}}|${TaskRoleArn}|;s|{{execution-role-arn}}|
${ExecutionRoleArn}|;s|{{awslogs-region}}|${AWSLogsRegion}|" \
    | xargs -0 aws ecs register-task-definition --region ${Region} --cli-input-
json
```

   b. 執行下列命令以啟動協助程式服務。將 *cluster-name* 和 *cluster-region* 替換成您 Amazon ECS 叢集的名稱和區域。

```
ClusterName=cluster-name
Region=cluster-region
aws ecs create-service \
    --cluster ${ClusterName} \
    --service-name cwagent-daemon-service \
    --task-definition ecs-cwagent-daemon-service \
    --scheduling-strategy DAEMON \
    --region ${Region}
```

   如果看到此錯誤訊息 `An error occurred (InvalidParameterException) when calling the CreateService operation: Creation of service was not idempotent`，即表示您已經建立名為 `cwagent-daemon-service` 的協助程式服務。您必須使用下列命令作為範例，先刪除該服務。

```
ClusterName=cluster-name
Region=Region
aws ecs delete-service \
```

```
    --cluster ${ClusterName} \
    --service cwagent-daemon-service \
    --region ${Region} \
    --force
```

部署發行 AWS 版 OpenTelemetry 以收集 Amazon ECS 叢集上的 EC2 執行個體層級指標

使用本節中的步驟使用發行 AWS 版收集 Amazon ECS 叢集上的 EC2 執行個體層級指標。OpenTelemetry 有關 AWS 發行版的更多信息 OpenTelemetry，請參閱 AWS . OpenTelemetry

這些步驟假設您已經有一個執行 Amazon ECS 的叢集。必須使用 EC2 啟動類型來部署此叢集。有關將發行 AWS 版與 Amazon ECS 搭配使用開放遙測，以及為此目的設定 Amazon ECS 叢集的詳細資訊，請參閱針對 ECS EC2 執行個 AWS 體層級指標在 Amazon 彈性容器服務中設定 OpenTelemetry 收集器發行版。

主題

- 快速設置使用 AWS CloudFormation
- 手動和自訂設定

快速設置使用 AWS CloudFormation

下 AWS CloudFormation 載用於在 EC2 上安裝 Amazon ECS OpenTelemetry 收集器發行 AWS 版的模板文件。執行下列 curl 命令。

```
curl -O https://raw.githubusercontent.com/aws-observability/aws-otel-collector/main/
deployment-template/ecs/aws-otel-ec2-instance-metrics-daemon-deployment-cfn.yaml
```

下載樣板檔案後，請開啟它並以儲存*CloudFormation######### PATH_TO_ _* 樣板。然後匯出下列參數並執行 AWS CloudFormation 命令，如下列命令所示。

- Cluster_Name– Amazon ECS 叢集名稱
- AWS_Region– 將傳送資料的區域
- 路徑至 CloudFormation _ 樣板 — 您儲存範本檔案的路徑。 AWS CloudFormation
- 命令 — 若要讓發行 AWS 版收集 OpenTelemetry 器能夠在 Amazon Amazon EC2 上收集 Amazon ECS 的執行個體層級指標，您必須為此參數指定`--config=/etc/ecs/otel-instance-metrics-config.yaml`。

```
ClusterName=Cluster_Name
Region=AWS_Region
command=--config=/etc/ecs/otel-instance-metrics-config.yaml
aws cloudformation create-stack --stack-name AOCECS-${ClusterName}-${Region} \
--template-body file://PATH_TO_CloudFormation_TEMPLATE \
--parameters ParameterKey=ClusterName,ParameterValue=${ClusterName} \
ParameterKey=CreateIAMRoles,ParameterValue=True \
ParameterKey=command,ParameterValue=${command} \
--capabilities CAPABILITY_NAMED_IAM \
--region ${Region}
```

執行此命令後，請使用 Amazon ECS 主控台查看任務是否正在執行。

故障診斷快速設定

若要檢查 AWS CloudFormation 堆疊的狀態，請輸入以下指令。

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation describe-stack --stack-name AOCECS-$ClusterName-$Region --region
 $Region
```

如果 StackStatus 的值不是 CREATE_COMPLETE 或 CREATE_IN_PROGRESS，請檢查堆疊事件以找出錯誤。輸入以下命令。

```
ClusterName=cluster-name
Region=cluster-region
aws cloudformation describe-stack-events --stack-name AOCECS-$ClusterName-$Region --
region $Region
```

若要檢查 AOCECS 常駐程式服務的狀態，請輸入以下命令。在輸出中，您應該會看到 runningCount 等於 deployment (部署) 區段的 desiredCount。如果不相等，請檢查輸出的 failures (失敗) 區段。

```
ClusterName=cluster-name
Region=cluster-region
aws ecs describe-services --services AOCECS-daemon-service --cluster $ClusterName --
region $Region
```

您也可以使用 CloudWatch 記錄主控台來檢查代理程式記錄檔。尋找 /aws/ecs/容器洞察/ {} /效能記錄群組。ClusterName

手動和自訂設定

請遵循本節中的步驟手動部署發行 AWS 版，以 OpenTelemetry 便從 Amazon Amazon EC2 執行個體上託管的 Amazon ECS 叢集收集執行個體層級指標。

步驟 1：必要的角色和政策

需要兩個 IAM 角色。如果角色不存在，即必須建立角色。如需這些角色的詳細資訊，請參閱建立 IAM 政策和建立 IAM 角色。

步驟 2：建立任務定義

創建一個任務定義並使用它來啟動 AWS Distro 作 OpenTelemetry為守護進程服務。

若要使用任務定義範本建立任務定義，請遵循使用 AWS oTel 收集器為 EC2 執行個體建立 ECS EC2 任務定義中的說明進行操作。

若要使用 Amazon ECS 主控台建立任務定義，請遵循安裝 AWS oTel 收集器中的指示，透過 AWS 主控台為 Amazon ECS EC2 執行個體指標建立任務定義。

步驟 3：啟動常駐程式服務

要啟動發行 AWS 版作 OpenTelemetry 為守護進程服務，請按照使用守護進程服務在亞馬遜彈性容器服務（Amazon ECS）上運行任務中的說明進行操作。

(選用) 進階組態

或者，您可以使用 SSM 為在 Amazon Amazon EC2 執行個體上託管的 Amazon ECS 叢集 OpenTelemetry 中為 AWS 發行版指定其他組態選項。如需有關建立規劃檔的詳細資訊，請參閱自訂 OpenTelemetry 組態。如需有關您在組態檔案中可使用之選項的詳細資訊，請參閱 AWS Container Insights Receiver。

設定 FireLens 將記錄檔傳送至記 CloudWatch 錄

FireLens Amazon ECS 可讓您使用任務定義參數將日誌路由到 Amazon 日誌以進行 CloudWatch 日誌儲存和分析。 FireLens 與流利的位和流暢的工作。我們提供了一個 AWS 流利的位圖像，或者您可以使用自己的流利位或流體圖像。使用 AWS 開發套件、 AWS CLI和支援使用 FireLens 組態建立 Amazon ECS 任務定義。 AWS Management Console如需有關 CloudWatch 記錄檔的詳細資訊，請參閱什麼是 CloudWatch 記錄檔？ 。

使用 Amazon ECS 時，有一些關鍵 FireLens 的考量因素。如需詳細資訊，請參閱考量。

若要尋找「 AWS 流利位元」影像，請參閱使 AWS 用流利位元影像。

若要建立使用 FireLens 組態的任務定義，請參閱建立使用 FireLens 組態的作業定義。

範例

下列工作定義範例示範如何指定將記錄檔轉寄至記錄檔記錄群組的 CloudWatch 記錄組態。如需詳細資訊，請參閱什麼是 Amazon CloudWatch 日誌？ 在 Amazon CloudWatch 日誌用戶指南中。

在日誌組態選項中，指定日誌群組名稱及其所在的區域。若要讓 Fluent 位元代表您建立日誌群組，請指定 "auto_create_group":"true"。您也可以將任務 ID 指定為日誌串流字首，以幫助篩選。如需詳細資訊，請參閱 CloudWatch 日誌的流利位元外掛程式。

```
{
 "family": "firelens-example-cloudwatch",
 "taskRoleArn": "arn:aws:iam::123456789012:role/ecs_task_iam_role",
 "containerDefinitions": [
  {
   "essential": true,
   "image": "906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-bit:latest",
   "name": "log_router",
   "firelensConfiguration": {
    "type": "fluentbit"
   },
   "logConfiguration": {
    "logDriver": "awslogs",
    "options": {
     "awslogs-group": "firelens-container",
     "awslogs-region": "us-west-2",
     "awslogs-create-group": "true",
     "awslogs-stream-prefix": "firelens"
    }
   },
   "memoryReservation": 50
   },
   {
    "essential": true,
    "image": "nginx",
    "name": "app",
    "logConfiguration": {
     "logDriver":"awsfirelens",
     "options": {
     "Name": "cloudwatch",
     "region": "us-west-2",
     "log_key": "log",
```

```
                                    "log_group_name": "/aws/ecs/containerinsights/
$(ecs_cluster)/application",
     "auto_create_group": "true",
     "log_stream_name": "$(ecs_task_id)"
    }
   },
   "memoryReservation": 100
  }
 ]
}
```

在 Amazon EKS 和 Kubernetes 上設定 Container Insights

Amazon EKS 1.23 及更新版本支援 Container Insights。只有 1.24 及更新版本支援快速入門安裝方法。

在 Amazon EKS 或 Kubernetes 上設定 Container Insights 的整體程序如下：

1. 確認您已完成必要的事前準備。
2. OpenTelemetry 在叢集上設定 Amazon 可 CloudWatch 觀測性 EKS 附加元件、 CloudWatch 代理程式或 AWS 發行版，以便將指標傳送至。 CloudWatch

> **Note**
>
> 若要使用具有增強 Amazon EKS 可觀察性的容器洞見，您必須使用 Amazon 可 CloudWatch 觀測性 EKS 附加元件或代理程式。 CloudWatch 如需有關此 Container Insights 版本的詳細資訊，請參閱 Container Insights 搭配 Amazon EKS 的增強可觀測性。要將容器見解與 Fargate 一起使用，您必須使用 AWS . OpenTelemetry Fargate 不支援 Container Insights 搭配 Amazon EKS 的增強可觀測性。

> **Note**
>
> 容器洞見現在支援 Amazon EKS 叢集中的 Windows 工作者節點。在 Windows 上也支援具有增強 Amazon EKS 可觀察性的容器洞見。如需在 Windows 上啟用容器深入解析的相關資訊，請參閱使用 CloudWatch 代理程式並啟用容器洞見增強的可觀察性。

設定「流利位元」或「Fluentd」，將記錄檔傳送至 CloudWatch 記錄檔。如果您安裝了 Amazon 可 CloudWatch 觀測性 EKS 附加元件，則預設會啟用此功能。)

如果您使用的是 CloudWatch 代理程式，您可以一次執行這些步驟，做為快速入門設定的一部分，或者分別執行這些步驟。

3. (選用) 設定 Amazon EKS 控制平面記錄。
4. (選擇性) 將 CloudWatch 代理程式設定為叢集上的 StatsD 端點，以將 StatsD 指標傳送至。CloudWatch
5. (選擇性) 啟用 App Mesh Envoy 存取日誌。

使用 Container Insights 的原始版本，收集的指標和擷取的指標會按自訂指標計費。使用 Container Insights 搭配 Amazon EKS 的增強可觀測性，Container Insights 指標和日誌會按觀測，而不是存放或擷取的指標計費。如需有關 CloudWatch 定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

主題

- 確認 先決條件
- 使用 CloudWatch 代理程式並啟用容器洞見增強的可觀察性
- 使用 AWS 發行版 OpenTelemetry
- 傳送記錄檔至 CloudWatch 記錄檔
- 在 Amazon EKS 和 Kubernetes 上更新或刪除 Container Insights

確認 先決條件

在 Amazon EKS 或 Kubernetes 上安裝 Container Insights 前，請檢查下列各項：無論您是使用 CloudWatch 代理程式還是 AWS Distro 在 Amazon EKS 叢集上設 OpenTelemetry 定容器洞見，都適用這些先決條件。

- 您擁有可運作的 Amazon EKS 或 Kubernetes 叢集，且其中包含在支援 Amazon EKS 和 Kubernetes Container Insights 的其中一個區域中連接的節點。如需支援區域的清單，請參閱 Container Insights。
- 您已安裝和執行 `kubectl`。如需詳細資訊，請參閱《Amazon EKS 使用者指南》中的安裝 `kubectl`。
- 如果您使用的是在上執行的 Kubernetes AWS 而不是使用 Amazon EKS，則下列先決條件也是必要的：

- 請確定 Kubernetes 叢集已啟用以角色為基礎的存取控制 (RBAC)。如需詳細資訊，請參閱 Kubernetes 參考中的使用 RBAC 授權。
- kubelet 已啟用 Webhook 授權模式。如需詳細資訊，請參閱 Kubernetes 參考中的 Kubelet authentication/authorization。

您還必須授予 IAM 許可，才能讓 Amazon EKS 工作者節點將指標和日誌傳送到 CloudWatch該節點。有兩種方式可以進行：

- 將政策連接至工作節點的 IAM 角色。這適用於 Amazon EKS 叢集和其他 Kubernetes 叢集。
- 針對叢集的服務帳戶使用 IAM 角色，並將政策連接至此角色。這僅適用於 Amazon EKS 叢集。

第一個選項會授與整個節點的權限，而針對服務帳戶使用 IAM 角色則只能 CloudWatch 存取適當的精靈集網繭。 CloudWatch

將政策連接至工作節點的 IAM 角色

請依照下列步驟將政策連接至工作節點的 IAM 角色。這適用於 Amazon EKS 外部的 Amazon EKS 叢集和 Kubernetes 叢集。

若要將必要的政策連接至工作節點的 IAM 角色

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 選取其中一個工作節點執行個體，然後在描述中選擇 IAM 角色。
3. 在 IAM role (IAM 角色) 頁面上，選擇 Attach policies (連接政策)。
4. 在策略清單中，選取旁邊的核取方塊CloudWatchAgentServerPolicy。如有需要，請使用搜尋方塊來尋找此政策。
5. 選擇連接政策。

如果您是在 Amazon EKS 之外執行 Kubernetes 叢集，您可能還沒有將 IAM 角色連接到工作節點。如果不是，您必須依照之前步驟中的說明，先將 IAM 角色連接至該執行個體，然後再新增政策。如需有關將角色連接至執行個體的詳細資訊，請參閱《Amazon EC2 Windows 執行個體使用者指南》中的將 IAM 角色連接至執行個體。

若您是在 Amazon EKS 外部執行 Kubernetes 叢集，且您希望在指標中收集 EBS 磁碟區 ID，您必須將另一個政策新增到連接至執行個體的 IAM 角色。新增以下內容作為內嵌政策。如需詳細資訊，請參閱《IAM 使用者指南》中的新增和移除 IAM 身分許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:DescribeVolumes"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

使用 IAM 服務帳戶角色

此方法僅適用於 Amazon EKS 叢集。

授與 CloudWatch 使用 IAM 服務帳戶角色的權限

1. 如果您尚未這麼做，請為叢集上的服務帳戶啟用 IAM 角色。如需詳細資訊，請參閱為叢集上的服務帳戶啟用 IAM 角色 。
2. 若尚未就緒，請將服務帳戶設定為使用 IAM 角色。如需詳細資訊，請參閱設定要擔任 IAM 角色的 Kubernetes 服務帳戶。

   建立角色時，除了為該角色建立的政策外，還將 CloudWatchAgentServerPolicyIAM 政策附加到該角色。此外，應該在amazon-cloudwatch命名空間中建立連結至此角色的相關聯 Kubernetes 服務帳戶，並在接下來的步驟中部署 CloudWatch 和 Fluent Bit 精靈集
3. 如果您尚未這麼做，請將 IAM 角色與叢集中的服務帳戶建立關聯。如需詳細資訊，請參閱設定要擔任 IAM 角色的 Kubernetes 服務帳戶。

使用 CloudWatch 代理程式並啟用容器洞見增強的可觀察性

使用下列其中一節中的指示，使用代理程式在 Amazon EKS 叢集或 Kubernetes 叢集上設定容器洞見。 CloudWatch 只有 Amazon EKS 1.24 及更新版本支援快速入門指示。

> **Note**
>
> 您可遵循下列任何一節中的指示來安裝 Container Insights。您無需遵循所有三組指示。

主題

- 安裝 Amazon CloudWatch 可觀測 EKS 附加元件
- Amazon EKS 和 Kubernetes 上 Container Insights 的 Quick Start 設定
- 設定 CloudWatch 代理程式以收集叢集度量

安裝 Amazon CloudWatch 可觀測 EKS 附加元件

您可使用 Amazon EKS 附加元件，來安裝 Container Insights 搭配 Amazon EKS 的增強可觀測性。附加元件會安裝 CloudWatch 代理程式以從叢集傳送基礎結構指標、安裝 Fluent Bit 以傳送容器記錄檔，CloudWatch Application Signals以及傳送應用程式效能遙測。

當您使用 Amazon EKS 附加元件 1.5.0 版或更新版本時，叢集中的 Linux 和 Windows 工作者節點都會啟用容器洞見。目前，Amazon EKS 中的視窗不支援應用程式訊號。

若叢集執行於 Kubernetes 而非 Amazon EKS，則不支援 Amazon EKS 附加元件。

如需 Amazon CloudWatch 可觀測性 EKS 附加元件的詳細資訊，請參閱。使用 Amazon CloudWatch 可觀測 EKS 附加元件安裝 CloudWatch 代理程式

若要安裝 Amazon CloudWatch 可觀測 EKS 附加元件

1. 首先，透過將 CloudWatchAgentServerPolicyIAM 政策附加到工作者節點來設定必要的許可。若要執行此作業，請輸入以下命令。取代*my-worker-node-role*為您的 Kubernetes 工作者節點所使用的 IAM 角色。

   ```
   aws iam attach-role-policy \
   --role-name my-worker-node-role \
   --policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy
   ```

2. 輸入下列命令來安裝附加元件：

   ```
   aws eks create-addon --cluster-name my-cluster-name --addon-name amazon-cloudwatch-observability
   ```

Amazon EKS 和 Kubernetes 上 Container Insights 的 Quick Start 設定

> **Important**
>
> 如果您要在 Amazon EKS 叢集上安裝容器洞見，建議您使用 Amazon 可 CloudWatch 觀測性 EKS 附加元件進行安裝，而不要使用本節中的指示。此外，若要擷取加速的運算網路，您必須使用 Amazon 可 CloudWatch 觀測性 EKS 附加元件。如需詳細資訊和指示，請參閱安裝 Amazon CloudWatch 可觀測 EKS 附加元件。

若要完成設定容器洞見，您可以遵循本節中的 quick start 說明。如果您要在 Amazon EKS 叢集中進行安裝，並且在 2023 年 11 月 6 日或之後使用本節中的指示，則您可在叢集中安裝 Container Insights 搭配 Amazon EKS 的增強可觀測性。

> **Important**
>
> 在完成本節中的步驟前，您必須先驗證包括 IAM 許可在內的先決條件。如需詳細資訊，請參閱確認 先決條件。

或者，您可以改為遵循下列兩節中的說明：設定 CloudWatch 代理程式以收集叢集度量 和 傳送記錄檔至 CloudWatch 記錄檔。這些部分提供有關 CloudWatch 代理程式如何與 Amazon EKS 和 Kubernetes 搭配使用的更多組態詳細資訊，但需要您執行更多安裝步驟。

使用 Container Insights 的原始版本，收集的指標和擷取的指標會按自訂指標計費。使用 Container Insights 搭配 Amazon EKS 的增強可觀測性，Container Insights 指標和日誌會按觀測，而不是存放或擷取的指標計費。如需有關 CloudWatch 定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

> **Note**
>
> Amazon 現已推出 Fluent Bit 作為 Container Insights 的預設日誌解決方案，且效能大幅提升。我們建議您使用 Fluent Bit 而不是 Fluentd。

使用 CloudWatch 代理程式操作員和流利位元快速入門

Fluent Bit 有兩種組態：最佳化版本和提供更類似於 Fluentd 體驗的版本。Quick Start 組態使用最佳化版本。如需 Fluentd 相容組態的詳細資訊，請參閱「將流利位元設定為 DaemonSet 將記錄檔傳送至記 CloudWatch 錄」。

CloudWatch 代理程式操作員是安裝到 Amazon EKS 叢集的額外容器。它是根據 Kubernetes 的「 OpenTelemetry 運算子」建模。操作員會管理叢集中 Kubernetes 資源的生命週期。它會 AWS 在 Amazon EKS 叢集上安裝 CloudWatch 代理程式、DCGM 匯出程式 (NVIDIA) 和神經元監視器，並對其進行管理。流利位元和 Windows 專用 CloudWatch 代理程式會直接安裝到 Amazon EKS 叢集，操作員無須管理這些叢集。

為了獲得更安全且功能豐富的憑證授權單位解決方案， CloudWatch 代理程式操作員需要 cert-manager，這是 Kubernetes 中廣泛採用的 TLS 憑證管理解決方案。使用 Cert-manager 可簡化取得、更新、管理及使用這些憑證的程序。它可確保憑證有效且是最新的，並嘗試在到期前的設定時間更新憑證。cert-manager 也有助於從各種支援來源 (包括 Certificate Manager 專用憑 AWS 證授權單位) 發行憑證。

使用快速入門部署容器見解

1. 如果尚未安裝在叢集中，請安裝憑證管理員。如需詳細資訊，請參閱憑證管理員安裝。
2. 輸入下列命令來安裝自訂資源定義 (CRD)。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/main/k8s-quickstart/cwagent-custom-resource-definitions.yaml | kubectl
 apply --server-side -f -
```

3. 輸入下列指令以安裝操作員。以*my-cluster-name*您的 Amazon EKS 或 Kubernetes 叢集的名稱取代 *my-cluster-region*，並以發佈日誌的區域名稱取代。我們建議您使用部署叢集的相同區域，以降低輸 AWS 出資料傳輸成本。

```
ClusterName=my-cluster-name
RegionName=my-cluster-region
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/main/k8s-quickstart/cwagent-operator-rendered.yaml | sed 's/
{{cluster_name}}/'${ClusterName}'/g;s/{{region_name}}/'${RegionName}'/g' | kubectl
 apply -f -
```

例如，若要在名為 MyCluster 的叢集上部署 Container Insights，並將日誌和指標發佈至美國西部 (奧勒岡)，請輸入以下命令。

```
ClusterName='MyCluster'
RegionName='us-west-2'
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/main/k8s-quickstart/cwagent-operator-rendered.yaml | sed 's/
```

```
{{cluster_name}}/'${ClusterName}'/g;s/{{region_name}}/'${RegionName}'/g' | kubectl
 apply -f -
```

從容器深入解析移轉

如果您已在 Amazon EKS 叢集中設定容器洞見，而且想要透過增強的 Amazon EKS 可觀察性遷移到容器洞見，請參閱 正在升級至 Container Insights 搭配 Amazon EKS 的增強可觀測性

刪除容器洞見

如果您想要在使用快速入門設定之後移除容器深入解析，請輸入下列命令。

```
ClusterName=my-cluster-name
RegionName=my-cluster-region
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/main/k8s-quickstart/cwagent-operator-rendered.yaml | sed 's/
{{cluster_name}}/'${ClusterName}'/g;s/{{region_name}}/'${RegionName}'/g' | kubectl
 delete -f -
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/main/k8s-quickstart/cwagent-custom-resource-definitions.yaml | kubectl delete
 -f -
```

設定 CloudWatch 代理程式以收集叢集度量

> **Important**
>
> 如果您要在 Amazon EKS 叢集上安裝容器洞見，建議您使用 Amazon 可 CloudWatch 觀測性 EKS 附加元件進行安裝，而不要使用本節中的指示。如需詳細資訊和指示，請參閱安裝 Amazon CloudWatch 可觀測 EKS 附加元件。

若要設定容器洞見收集指標，您可以遵循 Amazon EKS 和 Kubernetes 上 Container Insights 的 Quick Start 設定 中的步驟，或是遵循本節中的步驟。在下列步驟中，您將 CloudWatch 代理程式設定為能夠從叢集收集指標。

如果您要在 Amazon EKS 叢集中進行安裝，並且在 2023 年 11 月 6 日或之後使用本節中的指示，則您可在叢集中安裝 Container Insights 搭配 Amazon EKS 的增強可觀測性。

步驟 1：建立命名空間 CloudWatch

請使用下列步驟來建立呼叫的 Kubernetes 命名空間。amazon-cloudwatch CloudWatch如果您已建立此命名空間，則可以略過此步驟。

若要建立命名空間 CloudWatch

- 輸入以下命令。

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cloudwatch-namespace.yaml
```

步驟 2：在叢集中建立服務帳戶

如果您還沒有 CloudWatch 代理程式，請使用下列步驟建立服務帳戶。

建立 CloudWatch 代理程式的服務帳戶

- 輸入以下命令。

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-serviceaccount.yaml
```

如果您沒有遵循上述步驟，但您已經擁有要使用之 CloudWatch 代理程式的服務帳戶，則必須確定其具有下列規則。此外，在容器洞見安裝中的其他步驟，您必須使用該服務帳戶的名稱，而不是 cloudwatch-agent。

```
rules:
  - apiGroups: [""]
    resources: ["pods", "nodes", "endpoints"]
    verbs: ["list", "watch"]
  - apiGroups: [ "" ]
    resources: [ "services" ]
    verbs: [ "list", "watch" ]
  - apiGroups: ["apps"]
    resources: ["replicasets", "daemonsets", "deployments", "statefulsets"]
    verbs: ["list", "watch"]
  - apiGroups: ["batch"]
```

```
    resources: ["jobs"]
    verbs: ["list", "watch"]
  - apiGroups: [""]
    resources: ["nodes/proxy"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["nodes/stats", "configmaps", "events"]
    verbs: ["create", "get"]
  - apiGroups: [""]
    resources: ["configmaps"]
    resourceNames: ["cwagent-clusterleader"]
    verbs: ["get","update"]
  - nonResourceURLs: ["/metrics"]
    verbs: ["get", "list", "watch"]
```

步驟 3：為 CloudWatch代理 ConfigMap 程式建立

請使用下列步驟為 CloudWatch 代理 ConfigMap 程式建立。

建立 CloudWatch 代理程 ConfigMap 式的步驟

1. 執行下列命令，將 ConfigMap YAML 下載到kubectl用戶端主機：

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-configmap.yaml
   ```

2. 編輯已下載的 YAML 檔案，如下所示：

   - cluster_name – 在 kubernetes 區段，將 {{cluster_name}} 取代為叢集的名稱。移除 {{}} 字元。或者，如果您使用的是 Amazon EKS 叢集，您可以刪除 "cluster_name" 欄位和值。如果這樣做， CloudWatch 代理程式會從 Amazon EC2 標籤偵測叢集名稱。

3. (選擇性) ConfigMap 根據您的監視需求進一步變更，如下所示：

   - metrics_collection_interval – 在 kubernetes 區段，您可以指定代理程式收集指標的頻率。預設值為 60 秒。kubelet 中的預設 cadvisor 收集間隔是 15 秒，所以請不要將這個值設定為少於 15 秒。
   - endpoint_override — 如果您想要覆寫預設端點，可以在此logs區段中指定 CloudWatch 記錄端點。如果您透過 VPC 中的叢集進行發佈，且您想要將資料送往 VPC 端點，建議您這麼做。
   - force_flush_interval — 在此logs區段中，您可以指定將記錄事件發佈至記錄檔之前，批次處理記錄事件的間隔。 CloudWatch 預設為 5 秒。

- region – 在預設情況下，代理程式會將指標發佈至工作節點所在的區域。若要覆寫這個值，您可以在 agent 區段新增 region 欄位：例如，"region":"us-west-2"。
- StatsD 區段 — 如果您希望 CloudWatch Logs 代理程式也以 StatSD 接聽程式的身分在叢集的每個 Worker 節點中執行，您可以將statsd區段新增至metrics區段，如下列範例所示。如需此區段其他 StatsD 選項的資訊，請參閱 使用 StatsD 擷取自訂指標 。

```
"metrics": {
  "metrics_collected": {
    "statsd": {
      "service_address":":8125"
    }
  }
}
```

JSON 區段的完整範例如下。

```
{
    "agent": {
        "region": "us-east-1"
    },
    "logs": {
        "metrics_collected": {
            "kubernetes": {
                "cluster_name": "MyCluster",
                "metrics_collection_interval": 60
            }
        },
        "force_flush_interval": 5,
        "endpoint_override": "logs.us-east-1.amazonaws.com"
    },
    "metrics": {
        "metrics_collected": {
            "statsd": {
                "service_address": ":8125"
            }
        }
    }
}
```

4. 透過執行下列命令 ConfigMap 在叢集中建立。

```
kubectl apply -f cwagent-configmap.yaml
```

步驟 4：將 CloudWatch 代理程式部署為 DaemonSet

若要完成 CloudWatch 代理程式的安裝並開始收集容器指標，請使用下列步驟。

若要將 CloudWatch 代理程式部署為 DaemonSet

1. • 如果您不希望在叢集上使用 StatsD，請輸入以下命令。

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-
cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/
deployment-mode/daemonset/container-insights-monitoring/cwagent/cwagent-
daemonset.yaml
```

   • 若您希望使用 StatsD，請遵循下列步驟：

     a. 執行下列命令，將 DaemonSet YAML 下載到kubectl用戶端主機。

```
curl -O  https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-
mode/daemonset/container-insights-monitoring/cwagent/cwagent-daemonset.yaml
```

     b. 取消 cwagent-daemonset.yaml 檔案中 port 區段的註解，如下所示：

```
ports:
  - containerPort: 8125
    hostPort: 8125
    protocol: UDP
```

     c. 執行下列命令，在叢集中部署 CloudWatch 代理程式。

```
kubectl apply -f cwagent-daemonset.yaml
```

     d. 執行下列命令，在叢集中的 Windows 節點上部署 CloudWatch 代理程式。視窗上的代理程式不支援 StatsD 接聽 CloudWatch 程式。

```
kubectl apply -f cwagent-daemonset-windows.yaml
```

2. 透過執行下列命令確認已部署代理程式。

```
kubectl get pods -n amazon-cloudwatch
```

完成時， CloudWatch 代理程式會建立名為的記錄群組，/aws/containerinsights/*Cluster_Name*/performance並將效能記錄事件傳送至此記錄群組。如果您也將代理程式設定為 StatsD 接聽程式，此代理程式也會使用應用程式 pod 排定所在的節點 IP 地址，來接聽連接埠 8125 上的 StatsD 指標。

故障診斷

如果代理程式未正確部署，請嘗試下列項目：

- 執行以下命令來取得 Pod 清單。

```
kubectl get pods -n amazon-cloudwatch
```

- 執行以下命令和檢查輸出底部的事件。

```
kubectl describe pod pod-name -n amazon-cloudwatch
```

- 執行以下命令來檢查日誌。

```
kubectl logs pod-name  -n amazon-cloudwatch
```

使用 AWS 發行版 OpenTelemetry

您可以使用適 OpenTelemetry 用於收集器的發行 AWS 版，設定容器洞見，以從 Amazon EKS 叢集收集指標。有關 AWS 發行版的更多信息 OpenTelemetry，請參閱 AWS . OpenTelemetry

> **Important**
>
> 如果您使用 AWS Distro 進行安裝 OpenTelemetry，則安裝容器洞見，但不會獲得具有增強 Amazon EKS 可觀察性的容器見解。您不會在 Container Insights 搭配 Amazon EKS 的增強可觀測性中收集支援的詳細指標。

Container Insights 的設定方式取決於叢集是託管在 Amazon EC2 執行個體還是 AWS Fargate (Fargate)上。

託管於 Amazon EC2 的 Amazon EKS 叢集

如果您尚未這樣做，請確定您已滿足包含必要 IAM 角色在內的先決條件。如需詳細資訊，請參閱 確認先決條件。

Amazon 提供了 Helm Chart，您可以使用此功能來設定監控 Amazon EC2 上的 Amazon Elastic Kubernetes Service。此監視使用發行 AWS 版 OpenTelemetry（ADOT）收集器來獲取指標，並使用流利位用於日誌。因此，Helm 圖表對於在 Amazon Amazon EC2 上使用 Amazon EKS 並希望收集指標和日誌以傳送至 CloudWatch 容器見解的客戶非常有用。有關此頭盔圖表的詳細資訊，請參閱 EC2 指標上 EKS 的 ADOT 頭盔圖表和 Amazon CloudWatch 容器見解的日誌。

或者，您也可以使用本節其他部分中的說明。

首先， DaemonSet 通過輸入以下命令將 OpenTelemetry 收集器的 AWS 發行版部署為一個。

```
curl https://raw.githubusercontent.com/aws-observability/aws-otel-collector/main/
deployment-template/eks/otel-container-insights-infra.yaml |
kubectl apply -f -
```

若要確認收集器正在執行中，請輸入下列命令。

```
kubectl get pods -l name=aws-otel-eks-ci -n aws-otel-eks
```

如果此命令的輸出中包括多個處於 Running 狀態的 pod，則收集器正在執行並從叢集收集指標。收集器會建立名為 aws/containerinsights/*cluster-name*/performance 的日誌群組，並將效能日誌事件傳送給它。

如需如何在中查看容器見解量度的相關資訊 CloudWatch，請參閱檢視 Container Insights 指標。

AWS 也提供了有關此案例 GitHub 的說明文件。如果您想要自訂 Container Insights 發佈的指標和日誌，請參閱 https://aws-otel.github.io/docs/getting-started/container-insights/eks-infra。

託管於 Fargate 的 Amazon EKS 叢集

有關如何設定和部署 ADOT 收集 CloudWatch 器，以從部署到 Fargate 上 Amazon EKS 叢集的工作負載收集系統指標並將其傳送至容器洞見的說明，請參閱發行版中的容器洞察 EKS Fargate 以取得文件。 AWS OpenTelemetry

傳送記錄檔至 CloudWatch 記錄檔

若要將日誌從您的容器傳送到 Amazon CloudWatch 日誌，您可以使用流利位元或 Fluentd。如需詳細資訊，請參閱 Fluent Bit 和 Fluentd。

如果您尚未使用 Fluentd，我們建議您使用 Fluent Bit，原因如下：

- Fluent Bit 具有較小的資源佔用量，且在記憶體和 CPU 使用量方面比 Fluentd 更具資源效率。如需更詳細的比較資訊，請參閱 Fluent Bit 和 Fluentd 效能比較。
- 流利位圖像由開發和維護 AWS。這使 AWS 得能夠採用新的 Fluent Bit 圖像功能並更快地回應問題。

主題

- Fluent Bit 和 Fluentd 效能比較
- 將流利位元設定為 DaemonSet 將記錄檔傳送至記 CloudWatch 錄
- (選擇性) 將 Fluentd 設定為以將記錄檔傳送 DaemonSet至記錄檔 CloudWatch
- (選用) 設定 Amazon EKS 控制平面記錄。
- (選擇性) 啟用 App Mesh Envoy 存取日誌
- (選用) 針對大型叢集啟用 Use_Kubelet 功能

Fluent Bit 和 Fluentd 效能比較

下表顯示 Fluent Bit 在記憶體和 CPU 使用量方面比 Fluentd 更具效能優勢。以下數字僅供參考，可能會因環境而有所變化。

每秒日誌數	Fluentd CPU 用量	具有 Fluentd 相容組態的 Fluent Bit CPU 用量	具有最佳化組態的 Fluent Bit CPU 用量
100	0.35 vCPU	0.02 vCPU	0.02 vCP
1,000	0.32 vCPU	0.14 vCPU	0.11 vCPU
5,000	0.85 vCPU	0.48 vCPU	0.30 vCPU
10,000	0.94 vCPU	0.60 vCPU	0.39 vCPU

每秒日誌數	Fluentd 記憶體用量	具有 Fluentd 相容組態的 Fluent Bit 記憶體用量	具有最佳化組態的 Fluent Bit 記憶體用量
100	153 MB	46 MB	37 MB
1,000	270 MB	45 MB	40 MB
5,000	320 MB	55 MB	45 MB
10,000	375 MB	92 MB	75 MB

將流利位元設定為 DaemonSet 將記錄檔傳送至記 CloudWatch 錄

下列各節可協助您部署 Fluent Bit，以便將記錄從容器傳送至 CloudWatch 記錄檔。

主題

- 如果您已在使用 Fluentd，則其中存在差異
- 設定 Fluent Bit
- 多行日誌支援
- (選用) 減少 FluentD 的日誌量
- 故障診斷
- 儀表板

如果您已在使用 Fluentd，則其中存在差異

如果您已經使用 Fluentd 將日誌從容器發送到日 CloudWatch 誌，請閱讀本節以查看 Fluentd 和 Fluent Bit 之間的差異。如果您尚未將 Fluentd 搭配 Container Insights 使用，您可以跳至 設定 Fluent Bit。

我們為 Fluent Bit 提供兩種預設組態：

- Fluent Bit 最佳化組態 — 符合 Fluent Bit 最佳實務的組態。
- Fluentd 相容組態 — 盡可能符合 Fluentd 行為的組態。

下面的清單詳細說明了 Fluentd 和每個 Fluent Bit 組態的差異。

- 日誌串流名稱的差異 — 如果您使用 Fluent Bit 最佳化組態，則日誌串流名稱會有所不同。

在 /aws/containerinsights/Cluster_Name/application 下

- Fluent Bit 最佳化組態將日誌傳送至 *kubernetes-nodeName*-application.var.log.containers.*kubernetes-podName*_*kubernetes-namespace*_*kubernetes-container-name*-*kubernetes-containerID*
- Fluentd 將日誌傳送至 *kubernetes-podName*_*kubernetes-namespace*_*kubernetes-containerName*_*kubernetes-containerID*

在 /aws/containerinsights/Cluster_Name/host 下

- Fluent Bit 最佳化組態將日誌傳送至 *kubernetes-nodeName*.*host-log-file*
- Fluentd 將日誌傳送至 *host-log-file*-*Kubernetes-NodePrivateIp*

在 /aws/containerinsights/Cluster_Name/dataplane 下

- Fluent Bit 最佳化組態將日誌傳送至 *kubernetes-nodeName*.*dataplaneServiceLog*
- Fluentd 將日誌傳送至 *dataplaneServiceLog*-*Kubernetes-nodeName*

- Container Insights 寫入的 kube-proxy 和 aws-node 日誌檔案位於不同的位置。在 Fluentd 組態中，它們位於 /aws/containerinsights/Cluster_Name/application。在 Fluent Bit 最佳化組態中，它們位於 /aws/containerinsights/Cluster_Name/dataplane。
- 大多數中繼資料 (例如 pod_name 和 namespace_name) 在 Fluent Bit 和 Fluentd 中是相同的，但以下中繼資料則有所不同。
  - Fluent Bit 最佳化組態使用 docker_id，而 Fluentd 使用 Docker.container_id。
  - 這兩個 Fluent Bit 組態不會使用下列中繼資料。它們只存在於 Fluentd：container_image_id、master_url、namespace_id 以及 namespace_labels。

設定 Fluent Bit

若要設定 Fluent Bit 以從您的容器收集日誌，您可以遵循 Amazon EKS 和 Kubernetes 上 Container Insights 的 Quick Start 設定 中的步驟，或是遵循本節中的步驟。

無論使用何種方法，連接至叢集節點的 IAM 角色皆必須具有足夠的權限。如需有關執行 Amazon EKS 叢集所需許可的詳細資訊，請參閱《Amazon EKS 使用者指南》中的 Amazon EKS IAM 政策、角色和許可。

在下列步驟中，您將「流利位元」設定為 daemonSet，將記錄檔傳送至 CloudWatch 記錄檔。完成此步驟時，Fluent Bit 會建立下列日誌群組 (若尚未存在的話)。

> **Important**
>
> 如果您已在容器深入解析中設定 FluentD，且 FluentD DaemonSet 未如預期般執行 (如果您使用執行containerd階段，則可能會發生這種情況)，您必須先解除安裝 Fluent Bit，才能防止 FluentD 錯誤記錄檔訊息處理 FluentD 錯誤記錄檔訊息。否則，您必須在成功安裝 Fluent Bit 之後立即解除安裝 Fluentd。在安裝 Fluent Bit 後解除安裝 Fluentd，可確保在此遷移過程中記錄的唯一性。只需要一個「流暢位元」或「FluentD」，即可將記錄檔傳送至 CloudWatch 記錄檔。

日誌群組名稱	日誌來源
/aws/containerinsights/ *Cluster_Name* /application	/var/log/containers 中所有日誌檔
/aws/containerinsights/ *Cluster_Name* /host	來自 /var/log/dmesg 、/var/log/secure 以及 /var/log/messages 的日誌
/aws/containerinsights/ *Cluster_Name* /dataplane	/var/log/journal 中適用於 kubelet.service 、kubeproxy.service 和 docker.service 的日誌。

安裝 Fluent Bit 以將日誌從容器發送到 CloudWatch 日誌

1. 如果您還沒有名為 amazon-cloudwatch 的命名空間，請輸入下列命令建立一個：

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/
daemonset/container-insights-monitoring/cloudwatch-namespace.yaml
```

2. 執行下列命令來建立 ConfigMap 具cluster-info有叢集名稱的命名，以及要傳送記錄的地區。將 *cluster-name* 和 *cluster-region* 替換成您叢集的名稱和區域。

```
ClusterName=cluster-name
RegionName=cluster-region
FluentBitHttpPort='2020'
FluentBitReadFromHead='Off'
```

```
[[ ${FluentBitReadFromHead} = 'On' ]] && FluentBitReadFromTail='Off'||
 FluentBitReadFromTail='On'
[[ -z ${FluentBitHttpPort} ]] && FluentBitHttpServer='Off' ||
 FluentBitHttpServer='On'
kubectl create configmap fluent-bit-cluster-info \
--from-literal=cluster.name=${ClusterName} \
--from-literal=http.server=${FluentBitHttpServer} \
--from-literal=http.port=${FluentBitHttpPort} \
--from-literal=read.head=${FluentBitReadFromHead} \
--from-literal=read.tail=${FluentBitReadFromTail} \
--from-literal=logs.region=${RegionName} -n amazon-cloudwatch
```

在這個命令中，用於監控外掛程式指標的 FluentBitHttpServer 預設為開啟。若要將其關閉，請將命令中的第三行變更為 FluentBitHttpPort='' (空字串)。

此外，依預設，Fluent Bit 從尾部讀取日誌檔案，並將僅擷取部署後的新日誌。如果您想要相反結果，請設定 FluentBitReadFromHead='On'，它將收集檔案系統中的所有日誌。

3. 透過執行以下其中一個命令，下載 Fluent Bit daemonset 並將其部署到叢集。

   - 如果您想要 Linux 電腦的「流利位元」最佳化組態，請執行此指令。

     ```
     kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-
     cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/
     deployment-mode/daemonset/container-insights-monitoring/fluent-bit/fluent-
     bit.yaml
     ```

   - 如果您想要針對 Windows 電腦進行「流利位元」最佳化設定，請執行此命令。

     ```
     kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-
     cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/
     deployment-mode/daemonset/container-insights-monitoring/fluent-bit/fluent-bit-
     windows.yaml
     ```

   - 如果您使用的是 Linux 電腦，並且想要與 Fluentd 更類似的「流利位元」設定，請執行此命令。

     ```
     kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-
     cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/
     deployment-mode/daemonset/container-insights-monitoring/fluent-bit/fluent-bit-
     compatible.yaml
     ```

> **Important**
>
> Fluent Bit 守護程序組態預設會將記錄層級設定為 INFO，這可能會導致更高的 CloudWatch 記錄擷取成本。如果您想要減少日誌擷取量和成本，可以將日誌層級變更為 ERROR。
> 如需如何減少日誌量的詳細資訊，請參閱 (選用) 減少 FluentD 的日誌量。

4. 透過輸入以下命令來驗證部署。每個節點應有一個名為 fluent-bit-* 的 pod。

```
kubectl get pods -n amazon-cloudwatch
```

上述步驟在叢集中建立下列資源：

- 在 amazon-cloudwatch 命名空間中名為 Fluent-Bit 的服務帳戶。此服務帳戶會用來執行 Fluent Bit daemonSet。如需詳細資訊，請參閱 Kubernetes 參考中的管理服務帳戶。
- 在 amazon-cloudwatch 命名空間中名為的 Fluent-Bit-role 叢集角色。此叢集角色會在 pod 日誌上將 get、list 和 watch 許可授予給 Fluent-Bit 服務帳戶。如需詳細資訊，請參閱 Kubernetes 參考中的 API 概觀。
- ConfigMap 命名空間Fluent-Bit-config中的amazon-cloudwatch命名。這 ConfigMap 包含了流利位元要使用的組態。如需詳細資訊，請參閱 Kubernetes 工作說明文件 ConfigMap中的設定網繭以使用。

如果您想要驗證您的 Fluent Bit 設定，請遵循下列步驟。

驗證 Fluent Bit 設定

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Log groups (日誌群組)。
3. 請確定您位於部署 Fluent Bit 的區域。
4. 檢查區域中的日誌群組清單。請查看下列事項：
   - /aws/containerinsights/*Cluster_Name*/application
   - /aws/containerinsights/*Cluster_Name*/host
   - /aws/containerinsights/*Cluster_Name*/dataplane

5. 導覽至其中一個記錄群組，然後勾選日誌串流的 Last Event Time (上次事件時間)。如果它相對於您部署 Fluent Bit 的時間更近，則設定已受驗證。

   建立 /dataplane 日誌群組時可能會略微存在延遲。這是正常的，因為只有在 Fluent Bit 開始為該日誌組傳送日誌時，才會建立這些日誌群組。

多行日誌支援

如需如何搭配多行日誌使用 Fluent Bit 的詳細資訊，請參閱 Fluent Bit 文件的下列章節：

- Multiline Parsing
- Multiline and Containers (v1.8)
- Multiline Core (v1.8)
- Always use multiline in the tail input

(選用) 減少 FluentD 的日誌量

根據預設，我們會將流利位元應用程式記錄檔和 Kubernetes 中繼資料傳送至。 CloudWatch如果您想要減少傳送到的資料量 CloudWatch，可以停止傳送這些資料來源的其中一個或兩個資料來源 CloudWatch。

若要停止 Fluent Bit 應用程式日誌，請從 Fluent-Bit.yaml 檔案移除以下區段。

```
[INPUT]
        Name                tail
        Tag                 application.*
        Path                /var/log/containers/fluent-bit*
        Parser              docker
        DB                  /fluent-bit/state/flb_log.db
        Mem_Buf_Limit       5MB
        Skip_Long_Lines     On
        Refresh_Interval    10
```

若要移除附加至記錄事件的 Kubernetes 中繼資料 CloudWatch，請將下列篩選器新增至檔案中的application-log.conf區段。Fluent-Bit.yaml<Metadata_1>將類似欄位取代為實際的中繼資料識別碼。

```
application-log.conf: |
    [FILTER]
```

```
        Name                nest
        Match               application.*
        Operation           lift
        Nested_under        kubernetes
        Add_prefix          Kube.

    [FILTER]
        Name                modify
        Match               application.*
        Remove              Kube.<Metadata_1>
        Remove              Kube.<Metadata_2>
        Remove              Kube.<Metadata_3>

    [FILTER]
        Name                nest
        Match               application.*
        Operation           nest
        Wildcard            Kube.*
        Nested_under        kubernetes
        Remove_prefix       Kube.
```

故障診斷

如果您沒有看到這些日誌群組，且在正確的區域中尋找，請檢查日誌是否有 Fluent Bit daemonSet pod 以尋找錯誤。

執行以下命令並確保狀態為 Running。

```
kubectl get pods -n amazon-cloudwatch
```

如果日誌有與 IAM 許可相關的錯誤，請確認 IAM 角色是否已連接到叢集節點。如需有關執行 Amazon EKS 叢集所需許可的詳細資訊，請參閱《Amazon EKS 使用者指南》中的 Amazon EKS IAM 政策、角色和許可。

如果 pod 狀態為 CreateContainerConfigError，透過執行以下命令以取得確切的錯誤。

```
kubectl describe pod pod_name -n amazon-cloudwatch
```

儀表板

您可以建立一個儀表板來監控每個正在執行的外掛程式的指標。您可以查看輸入和輸出位元組、記錄處理速率以及輸出錯誤和重試/失敗率的資料。若要檢視這些指標，您需要為 Amazon EKS

和 Kubernetes 叢集安裝具有 Prometheus 指標集合的 CloudWatch 代理程式。如需如何設定儀表板的詳細資訊，請參閱 在 Amazon EKS 和 Kubernetes 叢集上安裝具有 Prometheus 指標集合的 CloudWatch 代理程式。

> **Note**
>
> 您必須先設定 Prometheus 指標的 Container Insights，才能設定此儀表板。如需詳細資訊，請參閱 Container Insights Prometheus 指標監控。

若要建立 Fluent Bit Prometheus 指標的儀表板

1. 建立環境變數，取代下列幾行中右側的數值，以符合您的部署。

```
DASHBOARD_NAME=your_cw_dashboard_name
REGION_NAME=your_metric_region_such_as_us-west-1
CLUSTER_NAME=your_kubernetes_cluster_name
```

2. 執行以下命令建立儀表板。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/sample_cloudwatch_dashboards/fluent-bit/cw_dashboard_fluent_bit.json \
| sed "s/{{YOUR_AWS_REGION}}/${REGION_NAME}/g" \
| sed "s/{{YOUR_CLUSTER_NAME}}/${CLUSTER_NAME}/g" \
| xargs -0 aws cloudwatch put-dashboard --dashboard-name ${DASHBOARD_NAME} --dashboard-body
```

(選擇性) 將 Fluentd 設定為以將記錄檔傳送 DaemonSet至記錄檔 CloudWatch

> **Warning**
>
> 對 Fluentd 的容器深入解析支援現在處於維護模式，這表示不 AWS 會為 Fluentd 提供任何進一步的更新，而且我們計劃在不久的將 future 將其淘汰。此外，Container Insights 目前的 Fluentd 組態是使用舊版的 Fluentd 映像 `fluent/fluentd-kubernetes-daemonset:v1.10.3-debian-cloudwatch-1.0`，其中沒有最新的改善項目和安全性修補程式。如需開放原始碼社群支援的最新 Fluentd 映像檔，請參閱。fluentd-kubernetes-daemonset

強烈建議您盡可能移轉以搭 FluentBit 配容器深入解析使用。 FluentBit 作為容器洞察的日誌轉發器可提供顯著的效能提升。
如需詳細資訊，請參閱 將流利位元設定為 DaemonSet 將記錄檔傳送至記 CloudWatch 錄 及 如果您已在使用 Fluentd，則其中存在差異。

若要設定 Fluentd 以從您的容器收集日誌，您可以按照「Amazon EKS 和 Kubernetes 上 Container Insights 的 Quick Start 設定」中或本節中的步驟操作。在下列步驟中，您將 Fluentd 設定為將記錄檔傳送 DaemonSet 至 CloudWatch 記錄檔。完成此步驟時，Fluentd 會建立下列日誌群組 (若尚未存在的話)。

日誌群組名稱	日誌來源
/aws/containerinsights/ *Cluster_Name* /application	/var/log/containers 中所有日誌檔
/aws/containerinsights/ *Cluster_Name* /host	來自 /var/log/dmesg 、/var/log/secure 以及 /var/log/messages 的日誌
/aws/containerinsights/ *Cluster_Name* /dataplane	/var/log/journal 中適用於 kubelet.service 、kubeproxy.service 和 docker.service 的日誌。

步驟 1：建立命名空間 CloudWatch

請使用下列步驟來建立呼叫的 Kubernetes 命名空間。amazon-cloudwatch CloudWatch如果您已建立此命名空間，則可以略過此步驟。

若要建立命名空間 CloudWatch

- 輸入以下命令。

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/container-insights-monitoring/cloudwatch-namespace.yaml
```

步驟 2：安裝 Fluentd

請下載 Fluentd 以開始此程序。完成以下步驟時，部署會在叢集建立下列資源：

- 在 amazon-cloudwatch 命名空間中名為 fluentd 的服務帳戶。此服務帳戶用於執行 Fluentd DaemonSet。如需詳細資訊，請參閱 Kubernetes 參考中的管理服務帳戶。
- 在 amazon-cloudwatch 命名空間中名為的 fluentd 叢集角色。此叢集角色會在 pod 日誌上將 get、list 和 watch 許可授予給 fluentd 服務帳戶。如需詳細資訊，請參閱 Kubernetes 參考中的 API 概觀。
- ConfigMap 命名空間fluentd-config中的amazon-cloudwatch命名。這 ConfigMap 包含了 Fluentd 要使用的配置。如需詳細資訊，請參閱 Kubernetes 工作說明文件 ConfigMap中的設定網繭以使用。

安裝 Fluentd

1. 建立 ConfigMap 具cluster-info有叢集名稱和記錄檔要傳送到的 AWS 地區的命名。執行以下命令，使用您的叢集和區域名稱更新預留位置。

```
kubectl create configmap cluster-info \
--from-literal=cluster.name=cluster_name \
--from-literal=logs.region=region_name -n amazon-cloudwatch
```

2. 執行下列命令，下載 Fluentd 並 DaemonSet 將其部署到叢集。確定您使用的是具有正確架構的容器映像。範例資訊清單僅適用於 x86 執行個體，並且如果您的叢集中有進階 RISC 機器 (ARM) 執行個體，則將輸入 CrashLoopBackOff。Fluentd daemonSet 沒有正式的多架構 Docker 映像檔可讓您將單一標籤用於多個基礎映像，並讓容器執行階段提取正確的標籤。Fluentd ARM 映像會使用不同的標籤，其中包含 arm64 尾碼。

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/
daemonset/container-insights-monitoring/fluentd/fluentd.yaml
```

> **Note**
>
> 由於最近的變更可最佳化 Fluentd 組態，並將 Fluentd API 請求對 Kubernetes API 端點的影響降至最低，Kubernetes 篩選條件的「監看」選項已預設為停用。如需詳細資訊，請參閱中fluent-plugin-kubernetes繼資料篩選。

3. 透過執行以下命令來驗證部署。每個節點應有一個名為 `fluentd-cloudwatch-*` 的 pod。

```
kubectl get pods -n amazon-cloudwatch
```

步驟 3：驗證 Fluentd 設定

若要驗證 Fluentd 設定，請使用下列步驟。

驗證 Container Insights 的 Fluentd 設定

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Log groups (日誌群組)。請確定您位於將 Fluentd 部署到容器的區域。

   在區域的日誌群組清單中，您應會看到下列項目：

   - `/aws/containerinsights/Cluster_Name/application`
   - `/aws/containerinsights/Cluster_Name/host`
   - `/aws/containerinsights/Cluster_Name/dataplane`

   如果您看到這些日誌群組，表示 Fluentd 設定已通過驗證。

多行日誌支援

2019 年 8 月 19 日，我們已為 Fluentd 所收集的日誌新增多行日誌支援。

根據預設，多行日誌項目啟動者是任何沒有空格的字元。這表示所有以未包含空格字元開始的日誌行都會視為新的多行日誌項目。

若您自己的應用程式日誌使用不同的多行啟動者，您可以在 `fluentd.yaml` 檔案中進行兩項變更來支援他們。

首先，將您日誌檔案的路徑名稱新增到 `fluentd.yaml` 中 `containers` 區段內的 `exclude_path` 欄位，從預設多行支援中排除他們。以下是範例。

```
<source>
  @type tail
  @id in_tail_container_logs
  @label @containers
```

```
  path /var/log/containers/*.log
  exclude_path ["full_pathname_of_log_file*", "full_pathname_of_log_file2*"]
```

接著，將您日誌檔案的區塊新增到 `fluentd.yaml` 檔案。以下範例用於 CloudWatch 代理程式的記錄檔，該記錄檔使用時間戳記規則運算式做為多行起動器。您可以複製此區塊並將它新增到 `fluentd.yaml`。變更指出的行，來反映應用程式日誌檔案名稱和您希望使用的多行啟動者。

```
<source>
  @type tail
  @id in_tail_cwagent_logs
  @label @cwagentlogs
  path /var/log/containers/cloudwatch-agent*
  pos_file /var/log/cloudwatch-agent.log.pos
  tag *
  read_from_head true
 <parse>
    @type json
    time_format %Y-%m-%dT%H:%M:%S.%NZ
 </parse>
</source>
```

```
<label @cwagentlogs>
  <filter **>
    @type kubernetes_metadata
    @id filter_kube_metadata_cwagent
  </filter>

  <filter **>
    @type record_transformer
    @id filter_cwagent_stream_transformer
    <record>
      stream_name ${tag_parts[3]}
    </record>
 </filter>

  <filter **>
    @type concat
    key log
    multiline_start_regexp /^\d{4}[-/]\d{1,2}[-/]\d{1,2}/
    separator ""
```

```
    flush_interval 5
    timeout_label @NORMAL
 </filter>

 <match **>
    @type relabel
    @label @NORMAL
 </match>
</label>
```

(選用) 減少 Fluentd 的日誌數量

根據預設，我們會將 Fluentd 應用程式記錄檔和 Kubernetes 中繼資料傳送至。 CloudWatch如果您想要減少傳送到的資料量 CloudWatch，可以停止傳送這些資料來源的其中一個或兩個資料來源 CloudWatch。

若要停止 Fluentd 應用程式日誌，請從 `fluentd.yaml` 檔案移除以下區段。

```
<source>
  @type tail
  @id in_tail_fluentd_logs
  @label @fluentdlogs
  path /var/log/containers/fluentd*
  pos_file /var/log/fluentd.log.pos
  tag *
  read_from_head true
   <parse>
   @type json
   time_format %Y-%m-%dT%H:%M:%S.%NZ
  </parse>
</source>

<label @fluentdlogs>
  <filter **>
    @type kubernetes_metadata
    @id filter_kube_metadata_fluentd
  </filter>

  <filter **>
    @type record_transformer
    @id filter_fluentd_stream_transformer
    <record>
```

```
      stream_name ${tag_parts[3]}
    </record>
   </filter>

   <match **>
     @type relabel
     @label @NORMAL
   </match>
</label>
```

若要移除附加至記錄事件的 Kubernetes 中繼資料 CloudWatch，請在檔案中的`record_transformer`區段中新增一行。`fluentd.yaml`在您希望移除此中繼資料的日誌來源中，新增下行。

```
remove_keys $.kubernetes.pod_id, $.kubernetes.master_url,
 $.kubernetes.container_image_id, $.kubernetes.namespace_id
```

例如：

```
<filter **>
  @type record_transformer
  @id filter_containers_stream_transformer
  <record>
    stream_name ${tag_parts[3]}
  </record>
  remove_keys $.kubernetes.pod_id, $.kubernetes.master_url,
 $.kubernetes.container_image_id, $.kubernetes.namespace_id
</filter>
```

故障診斷

如果您看不到這些記錄群組，而且在正確的區域中查看，請檢查 Fluentd DaemonSet Pod 的記錄檔以尋找錯誤。

執行以下命令並確保狀態為 `Running`。

```
kubectl get pods -n amazon-cloudwatch
```

在之前命令的結果中，請注意 pod 名稱開頭為 `fluentd-cloudwatch`。在下列命令使用此 pod 名稱。

```
kubectl logs pod_name -n amazon-cloudwatch
```

如果日誌有與 IAM 許可相關的錯誤，請確認 IAM 角色是否已連接到叢集節點。如需有關執行 Amazon EKS 叢集所需許可的詳細資訊，請參閱《Amazon EKS 使用者指南》中的 Amazon EKS IAM 政策、角色和許可。

如果 pod 狀態為 `CreateContainerConfigError`，透過執行以下命令以取得確切的錯誤。

```
kubectl describe pod pod_name -n amazon-cloudwatch
```

如果 pod 狀態為 `CrashLoopBackOff`，請確定當您安裝 Fluentd 時，Fluentd 容器映像的架構與節點相同。如果您的叢集同時具有 x86 和 ARM64 節點，則可以使用 kubernetes.io/arch 標籤將映像置於正確的節點上。如需詳細資訊，請參閱 kuberntes.io/arch。

(選用) 設定 Amazon EKS 控制平面記錄。

如果您使用的是 Amazon EKS，您可以選擇性地啟用 Amazon EKS 控制平面記錄，將稽核和診斷日誌直接從 Amazon EKS 控制平面提供給日誌。 CloudWatch 如需詳細資訊，請參閱 Amazon EKS 控制平面記錄。

(選擇性) 啟用 App Mesh Envoy 存取日誌

您可以設定容器見解 Fluentd，將 App Mesh 特使存取記錄傳送至記錄。 CloudWatch 如需詳細資訊，請參閱 Logging (記錄)。

將特使訪問日誌發送到 CloudWatch 日誌

1. 在叢集中設定 Fluentd。如需詳細資訊，請參閱 (選擇性) 將 Fluentd 設定為以將記錄檔傳送 DaemonSet至記錄檔 CloudWatch 。
2. 設定虛擬節點的 Envoy 存取日誌。如需說明，請參閱 Logging (記錄)。請務必將每個虛擬節點中的日誌路徑設定為 **`/dev/stdout`**。

完成後，Envoy 存取日誌會傳送至 `/aws/containerinsights/Cluster_Name/application` 日誌群組。

(選用) 針對大型叢集啟用 Use_Kubelet 功能

根據預設，在 Kubernetes 外掛程式中會停用 Use_Kubelet 功能。 FluentBit 啟用此功能可減少 API 伺服器的流量，並減輕 API 伺服器的瓶頸問題。建議您針對大型叢集啟用此功能。

若要啟用 Use_Kubelet，請先將節點和節點/代理許可新增至 clusterRole 組態。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: fluent-bit-role
rules:
  - nonResourceURLs:
      - /metrics
    verbs:
      - get
  - apiGroups: [""]
    resources:
      - namespaces
      - pods
      - pods/logs
      - nodes
      - nodes/proxy
    verbs: ["get", "list", "watch"]
```

在 DaemonSet 組態中，此功能需要存取主機網路。amazon/aws-for-fluent-bit 的映像版本應該是 2.12.0 或更高版本，或者流暢位元映像版本應該是 1.7.2 或更高版本。

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluent-bit
  namespace: amazon-cloudwatch
  labels:
    k8s-app: fluent-bit
    version: v1
    kubernetes.io/cluster-service: "true"
spec:
  selector:
    matchLabels:
      k8s-app: fluent-bit
  template:
    metadata:
      labels:
        k8s-app: fluent-bit
        version: v1
        kubernetes.io/cluster-service: "true"
    spec:
```

```
      containers:
      - name: fluent-bit
        image: amazon/aws-for-fluent-bit:2.19.0
        imagePullPolicy: Always
        env:
            - name: AWS_REGION
              valueFrom:
                configMapKeyRef:
                  name: fluent-bit-cluster-info
                  key: logs.region
            - name: CLUSTER_NAME
              valueFrom:
                configMapKeyRef:
                  name: fluent-bit-cluster-info
                  key: cluster.name
            - name: HTTP_SERVER
              valueFrom:
                configMapKeyRef:
                  name: fluent-bit-cluster-info
                  key: http.server
            - name: HTTP_PORT
              valueFrom:
                configMapKeyRef:
                  name: fluent-bit-cluster-info
                  key: http.port
            - name: READ_FROM_HEAD
              valueFrom:
                configMapKeyRef:
                  name: fluent-bit-cluster-info
                  key: read.head
            - name: READ_FROM_TAIL
              valueFrom:
                configMapKeyRef:
                  name: fluent-bit-cluster-info
                  key: read.tail
            - name: HOST_NAME
              valueFrom:
                fieldRef:
                  fieldPath: spec.nodeName
            - name: HOSTNAME
              valueFrom:
                fieldRef:
                  apiVersion: v1
                  fieldPath: metadata.name
```

```
            - name: CI_VERSION
              value: "k8s/1.3.8"
        resources:
            limits:
              memory: 200Mi
            requests:
              cpu: 500m
              memory: 100Mi
        volumeMounts:
        # Please don't change below read-only permissions
        - name: fluentbitstate
          mountPath: /var/fluent-bit/state
        - name: varlog
          mountPath: /var/log
          readOnly: true
        - name: varlibdockercontainers
          mountPath: /var/lib/docker/containers
          readOnly: true
        - name: fluent-bit-config
          mountPath: /fluent-bit/etc/
        - name: runlogjournal
          mountPath: /run/log/journal
          readOnly: true
        - name: dmesg
          mountPath: /var/log/dmesg
          readOnly: true
      terminationGracePeriodSeconds: 10
      hostNetwork: true
      dnsPolicy: ClusterFirstWithHostNet
      volumes:
      - name: fluentbitstate
        hostPath:
          path: /var/fluent-bit/state
      - name: varlog
        hostPath:
          path: /var/log
      - name: varlibdockercontainers
        hostPath:
          path: /var/lib/docker/containers
      - name: fluent-bit-config
        configMap:
          name: fluent-bit-config
      - name: runlogjournal
        hostPath:
```

```
          path: /run/log/journal
      - name: dmesg
        hostPath:
          path: /var/log/dmesg
      serviceAccountName: fluent-bit
      tolerations:
      - key: node-role.kubernetes.io/master
        operator: Exists
        effect: NoSchedule
      - operator: "Exists"
        effect: "NoExecute"
      - operator: "Exists"
        effect: "NoSchedule"
```

Kubernetes 外掛程式組態應該類似如下：

```
[FILTER]
        Name                kubernetes
        Match               application.*
        Kube_URL            https://kubernetes.default.svc:443
        Kube_Tag_Prefix     application.var.log.containers.
        Merge_Log           On
        Merge_Log_Key       log_processed
        K8S-Logging.Parser  On
        K8S-Logging.Exclude Off
        Labels              Off
        Annotations         Off
        Use_Kubelet         On
        Kubelet_Port        10250
        Buffer_Size         0
```

在 Amazon EKS 和 Kubernetes 上更新或刪除 Container Insights

使用這些章節中的步驟來更新您的 CloudWatch 代理程式容器映像，或從 Amazon EKS 或 Kubernetes 叢集移除容器洞見。

主題

- 正在升級至 Container Insights 搭配 Amazon EKS 的增強可觀測性
- 更新 CloudWatch 代理程式容器映像
- 刪除容器見解的 CloudWatch 代理程式和流利位元

正在升級至 Container Insights 搭配 Amazon EKS 的增強可觀測性

> **Important**
>
> 如果您要在 Amazon EKS 叢集上升級或安裝容器洞見，建議您使用 Amazon 可 CloudWatch 觀測性 EKS 附加元件進行安裝，而不要使用本節中的指示。此外，若要擷取加速的運算指標，您必須使用 Amazon 可 CloudWatch 觀測性 EKS 附加元件。如需詳細資訊和指示，請參閱安裝 Amazon CloudWatch 可觀測 EKS 附加元件。

Container Insights 搭配 Amazon EKS 的增強可觀測性是最新版 Container Insights。它會從執行於 Amazon EKS 的叢集收集詳細的指標，並提供經策管且立即可用的儀表板，以深入了解應用程式和基礎設施遙測。如需有關此 Container Insights 版本的詳細資訊，請參閱 Container Insights 搭配 Amazon EKS 的增強可觀測性。

如果您已在 Amazon EKS 叢集安裝原始版 Container Insights，並且您想要使用增強可觀測性將其升級至較新版本，請遵循本節中的指示進行操作。

> **Important**
>
> 在完成本節中的步驟之前，您必須先確認必要條件，包括憑證管理員。如需詳細資訊，請參閱使用 CloudWatch 代理程式操作員和流利位元快速入門。

將 Amazon EKS 叢集升級至 Container Insights 搭配 Amazon EKS 的增強可觀測性

1. 輸入下列命令以安裝 CloudWatch 代理程式操作員。以*my-cluster-name*您的 Amazon EKS 或 Kubernetes 叢集的名稱取代 *my-cluster-region*，並以發佈日誌的區域名稱取代。我們建議您使用部署叢集的相同區域，以降低輸 AWS 出資料傳輸成本。

   ```
   ClusterName=my-cluster-name
   RegionName=my-cluster-region
   curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
   container-insights/main/k8s-quickstart/cwagent-operator-rendered.yaml | sed 's/
   {{cluster_name}}/'${ClusterName}'/g;s/{{region_name}}/'${RegionName}'/g' | kubectl
    apply -f -
   ```

   如果您注意到資源衝突所造成的失敗，很可能是因為您已經有 CloudWatch 代理程式和 Fluent Bit 及其相關元件 ServiceAccount，例如叢集上 ClusterRoleBinding 安裝的、 ClusterRole 和。當

CloudWatch 代理程式操作員嘗試安裝 CloudWatch 代理程式及其相關元件時，如果偵測到內容有任何變更，依預設會失敗安裝或更新，以避免覆寫叢集上資源的狀態。建議您刪除先前已在叢集上安裝的任何具有 Container Insights 設定的現有 CloudWatch 代理程式，然後安裝 CloudWatch 代理程式操作員。

2. (選擇性) 若要套用現有的自訂 Fluent 位元組態，您必須更新與 Fluent 位元守護程式集關聯的組態映射。 CloudWatch 代理程式運算子提供 Fluent Bit 的預設組態，您可以視需要覆寫或修改預設組態。若要套用自訂組態，請依照下列步驟執行。

   a. 輸入以下指令來開啟現有的組態。

   ```
   kubectl edit cm fluent-bit-config -n amazon-cloudwatch
   ```

   b. 在檔案中進行變更，然後輸入`:wq`以儲存檔案並結束編輯模式。

   c. 通過輸入以下命令重新啟動流利位。

   ```
   kubectl rollout restart fluent-bit -n amazon-cloudwatch
   ```

更新 CloudWatch 代理程式容器映像

> **Important**
>
> 如果您要在 Amazon EKS 叢集上升級或安裝容器洞見，建議您使用 Amazon 可 CloudWatch 觀測性 EKS 附加元件進行安裝，而不要使用本節中的指示。此外，若要擷取加速運算指標，您必須使用 Amazon 可 CloudWatch 觀測性 EKS 附加元件或 CloudWatch 代理程式操作員。如需詳細資訊和指示，請參閱安裝 Amazon CloudWatch 可觀測 EKS 附加元件。

若您需要將您的容器映像更新至最新的版本，請使用本節中的步驟。

更新您的容器映像

1. 輸入下列指令，確認`amazoncloudwatchagent`客戶資源定義 (CRD) 是否已存在。

   ```
   kubectl get crds amazoncloudwatchagents.cloudwatch.aws.amazon.com -n amazon-
   cloudwatch
   ```

如果此命令傳回缺少 CRD 的錯誤，則叢集沒有針對使用代理程式操作員設定的 Amazon EKS 具有增強型可觀察性的容器見解。 CloudWatch 在這種情況下，請參閱 正在升級至 Container Insights 搭配 Amazon EKS 的增強可觀測性。

2. 輸入以下命令，套用最新的 `cwagent-version.yaml` 檔案。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/main/k8s-quickstart/cwagent-version.yaml | kubectl apply -f -
```

刪除容器見解的 CloudWatch 代理程式和流利位元

如果您使用安裝 Amazon EKS 的可 CloudWatch 觀測性附加元件來安裝容器洞見，您可以輸入下列命令來刪除容器洞見和 CloudWatch 代理程式：

> **Note**
>
> Amazon EKS 附加元件現在支援 Windows 工作者節點上的容器洞見。如果您刪除 Amazon EKS 附加元件，Windows 適用的容器深入解析也會一併刪除。

```
aws eks delete-addon –cluster-name my-cluster –addon-name amazon-cloudwatch-observability
```

否則，若要刪除與 CloudWatch 代理程式和 Fluent Bit 相關的所有資源，請輸入下列命令。在此命令中，*My_Cluster_Name* 是您的 Amazon EKS 或 Kubernetes 叢集的名稱，而############名稱。

```
ClusterName=My_Cluster_Name
RegionName=My-Region
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/main/k8s-quickstart/cwagent-operator-rendered.yaml | sed 's/{{cluster_name}}/'${ClusterName}'/g;s/{{region_name}}/'${RegionName}'/g' | kubectl delete -f -
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/main/k8s-quickstart/cwagent-custom-resource-definitions.yaml | kubectl delete -f -
```

檢視 Container Insights 指標

在您設定容器洞見並收集指標之後，您可以在 CloudWatch 主控台中檢視這些指標。

若要讓 Container Insights 指標顯示在儀表板上，您必須完成 Container Insights 設定。如需詳細資訊，請參閱 設定 Container Insights。

此程序會說明如何檢視 Container Insights 從收集的日誌資料自動產生的指標。本節的其餘部分將說明如何進一步深入了解您的資料，並使用 CloudWatch 日誌見解以更詳細的方式查看更多指標。

檢視容器洞見指標

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Insights，然後選擇 Container Insights。
3. 在 Container Insights 下方的下拉式方塊中，選擇效能監控。
4. 使用靠近頂端的下拉式方塊來選取要檢視的資源類型，以及特定資源。

您可以針對容器深入解析收集的任何量度設定 CloudWatch 警示。如需更多資訊，請參閱 使用 Amazon CloudWatch 警報

> **Note**
>
> 如果您已設定應用 CloudWatch 程式深入解析來監控您的容器化應用程式，「應用程式深入解析」儀表板會顯示在「容器見解」儀表板下方 如果您尚未啟用 Application Insights，您可以選擇 Container Insights 儀表板中效能檢視下方的 Auto-configure Application Insights (自動設定 Application Insights)。
> 如需 Application Insights 和容器化應用程式的詳細資訊，請參閱 啟用 Application Insights 進行 Amazon ECS 和 Amazon EKS 資源監控。

檢視頂端貢獻因子

對於容器深入解析效能監視中的某些檢視，您也可以查看記憶體或 CPU 或最近使用中的資源的前幾名參與者。如果您在頁面頂端附近的下拉式方塊中選取下列任何一個儀表板，則可使用此選項：

- ECS 服務
- ECS 任務
- EKS 命名空間
- EKS 服務
- EKS Pod

當您檢視其中一種資源類型時，頁面底部會顯示一個最初依 CPU 用量排序的表格。您可以將其變更為按記憶體用量或最近活動排序。若要查看表格中其中一列的詳細資訊，您可以選取該列旁邊的核取方塊，然後選擇 Actions (動作)，並選擇 Actions (動作) 選單中的其中一個選項。

使用 CloudWatch 日誌深入解析檢視容器見解資料

Container Insights 會使用內嵌指標格式的效能日誌事件收集指標。日誌存儲在 CloudWatch 日誌中。CloudWatch 從日誌中自動生成多個指標，您可以在 CloudWatch 控制台中查看。您也可以對使用 CloudWatch 日誌見解查詢收集的效能資料進行更深入的分析。

如需 CloudWatch 日誌深入解析的詳細資訊，請參閱使用日誌深入分析分析 CloudWatch 記錄資料 如需您可以在查詢中使用的日誌欄位詳細資訊，請參閱 Amazon EKS 和 Kubernetes 的 Container Insights 效能日誌事件。

使用 CloudWatch 日誌深入解析查詢您的容器指標資料

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Insights。

   靠近螢幕頂端是查詢編輯器。當您第一次開啟 CloudWatch 記錄深入解析時，此方塊包含一個預設查詢，會傳回 20 個最近的記錄事件。
3. 在查詢編輯器上方的方塊中，選取要查詢的其中一個容器洞見日誌群組。若要讓以下範例查詢運作，日誌群組名稱必須以 performance (效能) 為結尾。

   當您選取記錄群組時， CloudWatch Logs Insights 會自動偵測記錄群組中資料的欄位，並將這些欄位顯示在右窗格的 [探查] 欄位中。它也會顯示一段時間內此日誌群組中日誌事件的長條圖。此長條圖會顯示日誌群組中符合您查詢和時間範圍的事件分佈，而不只是表格中顯示的事件。
4. 在查詢編輯器中，將預設查詢取代為以下查詢，然後選擇 Run query (執行查詢)。

   ```
   STATS avg(node_cpu_utilization) as avg_node_cpu_utilization by NodeName
   | SORT avg_node_cpu_utilization DESC
   ```

   此查詢顯示的節點清單會依平均節點 CPU 使用率排序。
5. 若要嘗試另一個範例，請將查詢取代為另一個查詢，並選擇 Run query (執行查詢)。此頁面稍後會列出更多範例查詢。

   ```
   STATS avg(number_of_container_restarts) as avg_number_of_container_restarts by
    PodName
   ```

```
| SORT avg_number_of_container_restarts DESC
```

此查詢顯示的 pod 清單會依重新啟動的平均容器數量來排序。

6. 如果您想要嘗試另一個查詢，您可以在螢幕右側使用清單中的包含欄位。如需查詢語法的詳細資訊，請參閱CloudWatch 記錄深入解析查詢語法。

查看您的資源清單

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Resources (資源)。
3. 預設檢視是 Container Insights 所監視的資源清單，以及您已在這些資源上設定的警示。若要查看資源的視覺化地圖，請選擇 Map view (地圖檢視)。
4. 從地圖檢視中，您可以將指標暫停在地圖中的任何資源上，以查看有關該資源的基本指標。您可以選擇任何資源來查看關於該資源的更詳細圖形。

使用案例：查看 Amazon ECS 容器中的任務層級指標

下列範例說明如何使用 CloudWatch 日誌見解深入了解您的容器見解日誌。如需更多範例，請參閱針對 Amazon ECS 介紹 Amazon CloudWatch 容器見解的部落格。

Container Insights 不會在精細程度的任務層級自動產生指標。下列查詢會顯示 CPU 和記憶體用量的任務層級指標。

```
stats avg(CpuUtilized) as CPU, avg(MemoryUtilized) as Mem by TaskId, ContainerName
| sort Mem, CPU desc
```

Container Insights 的其他範例查詢

您的 Pod 清單，依照容器重新啟動的平均次數進行排序

```
STATS avg(number_of_container_restarts) as avg_number_of_container_restarts by PodName
| SORT avg_number_of_container_restarts DESC
```

請求的 Pod 與正在執行的 Pod

```
fields @timestamp, @message
| sort @timestamp desc
```

```
| filter Type="Pod"
| stats min(pod_number_of_containers) as requested,
 min(pod_number_of_running_containers) as running, ceil(avg(pod_number_of_containers-
pod_number_of_running_containers)) as pods_missing by kubernetes.pod_name
| sort pods_missing desc
```

叢集節點故障的計數

```
stats avg(cluster_failed_node_count) as CountOfNodeFailures
| filter Type="Cluster"
| sort @timestamp desc
```

應用程式日誌錯誤 (依容器名稱)

```
stats count() as countoferrors by kubernetes.container_name
| filter stream="stderr"
| sort countoferrors desc
```

Container Insights 收集的指標

容器洞見為 Amazon ECS 和 Amazon ECS 收集一組指標，以及 AWS Fargate Amazon EKS、Amazon EKS 和 Kubernetes AWS Fargate 上的一組不同指標。

指標要在容器任務執行一段時間後才會顯示。

主題

- Amazon ECS Container Insights 指標
- Amazon EKS 和 Kubernetes Container Insights 指標

Amazon ECS Container Insights 指標

下表列出 Container Insights 為 Amazon ECS 收集的指標和維度。這些指標會在 `ECS/ContainerInsights` 命名空間中。如需詳細資訊，請參閱 指標。

如果您沒有在主控台中看到任何容器洞見指標，請確定您已完成容器洞見的設定。在完整設定容器洞見前指標都不會出現。如需詳細資訊，請參閱 設定 Container Insights。

當您完成 在 Amazon ECS 上設定 Container Insights 以取得叢集和服務層級指標 中的步驟後即可使用下列指標

指標名稱	維度	描述
ContainerInstanceCount	ClusterName	執行 Amazon ECS 代理程式並與叢集註冊的 EC2 執行個體數量。 只會針對在叢集中執行 Amazon ECS 任務的容器執行個體來收集此指標。對於沒有任何 Amazon ECS 任務的空容器執行個體，不會收集此指標。 單位：計數
CpuUtilized	TaskDefinitionFamily，ClusterName ServiceName，ClusterName ClusterName	資源中任務所使用的 CPU 單位，由您正在使用的維度設定所指定。 此指標只會為在其任務定義中具備已定義 CPU 保留的任務進行收集。 單位：無
CpuReserved	TaskDefinitionFamily，ClusterName ServiceName，ClusterName ClusterName	資源中任務所預留的 CPU 單位，由您正在使用的維度設定所指定。 此指標只會為在其任務定義中具備已定義 CPU 保留的任務進行收集。 單位：無
DeploymentCount	ServiceName，ClusterName	在 Amazon ECS 服務中部署的數量。 單位：計數

指標名稱	維度	描述
DesiredTaskCount	ServiceName，ClusterName	Amazon ECS 服務所需的任務數。 單位：計數
EBSFilesystemSize	VolumeName，TaskDefinitionFamily，ClusterName TaskDefinitionFamily，ClusterName ServiceName，ClusterName	Amazon EBS 檔案系統儲存的總容量 (以 GB 為單位)，分配給您使用的維度所指定的資源。 此指標僅適用於使用平台版本1.4.0或更新版本的 Amazon EC2 執行個體在 Fargate 上執行的 Amazon ECS 基礎設施上執行的任務。1.79.0 單位：千兆位元組 (GB)

指標名稱	維度	描述
EBSFilesystemUtilized	VolumeName ，TaskDefinitionFamily ，ClusterName TaskDefinitionFamily ，ClusterName ServiceName ，ClusterName	您正在使用的維度所指定的資源正在使用的 Amazon EBS 檔案系統儲存體的總容量，以 GB 為單位。 此指標僅適用於使用平台版本1.4.0或更新版本的 Amazon EC2 執行個體在 Fargate 上執行的 Amazon ECS 基礎設施上執行的任務。1.79.0 對於在 Fargate 上執行的工作，Fargate 會在磁碟上保留僅供 Fargate 使用的空間。Fargate 使用的空間沒有相關的費用，但是您將使用類似df的工具看到此額外的存儲空間。 單位：千兆位元組 (GB)

指標名稱	維度	描述
EphemeralStorageReserved [1]	TaskDefinitionFamily，ClusterName ServiceName，ClusterName ClusterName	資源中從暫時性儲存裝置中所預留的位元組數，該資源由您正在使用的維度所指定。暫時性儲存裝置會用於容器根檔案系統，以及容器映像和任務定義中定義的任何綁定掛載主機磁碟區。在執行中的任務中，無法變更暫時性儲存裝置的數量。 此指標僅適用於在 Fargate Linux 平台版本 1.4.0 或更新版本上執行的任務。 單位：千兆位元組 (GB)
EphemeralStorageUtilized [1]	TaskDefinitionFamily，ClusterName ServiceName，ClusterName ClusterName	資源中從暫時性儲存裝置中使用的位元組數，該資源由您正在使用的維度所指定。暫時性儲存裝置會用於容器根檔案系統，以及容器映像和任務定義中定義的任何綁定掛載主機磁碟區。在執行中的任務中，無法變更暫時性儲存裝置的數量。 此指標僅適用於在 Fargate Linux 平台版本 1.4.0 或更新版本上執行的任務。 單位：千兆位元組 (GB)

指標名稱	維度	描述
MemoryUtilized	TaskDefinitionFamily，ClusterName ServiceName，ClusterName ClusterName	資源中任務正在使用的記憶體，由您正在使用的維度設定所指定。 此指標只會為在其任務定義中具備已定義記憶體保留的任務進行收集。 單位：MB
MemoryReserved	TaskDefinitionFamily，ClusterName ServiceName，ClusterName ClusterName	資源中任務所預留的記憶體，由您正在使用的維度設定所指定。 此指標只會為在其任務定義中具備已定義記憶體保留的任務進行收集。 單位：MB
NetworkRxBytes	TaskDefinitionFamily，ClusterName ServiceName，ClusterName ClusterName	資源收到的位元組數，由您正在使用的維度所指定。從 Docker 執行期獲取此指標。 此指標僅適用於使用 awsvpc 或 bridge 網路模式的任務中的容器。 單位：位元組/秒

指標名稱	維度	描述
NetworkTxBytes	TaskDefinitionFamily，ClusterName ServiceName，ClusterName ClusterName	資源傳輸的位元組數，由您正在使用的維度所指定。從 Docker 執行期獲取此指標。 此指標僅適用於使用 awsvpc 或 bridge 網路模式的任務中的容器。 單位：位元組/秒
PendingTaskCount	ServiceName，ClusterName	目前處於 PENDING 狀態的任務數。 單位：計數
RunningTaskCount	ServiceName，ClusterName	目前處於 RUNNING 狀態的任務數。 單位：計數
ServiceCount	ClusterName	叢集中的服務數量。 單位：計數
StorageReadBytes	TaskDefinitionFamily，ClusterName ServiceName，ClusterName ClusterName	資源中從執行個體儲存裝置讀取的位元組數，該資源由您正在使用的維度所指定。這不包括儲存裝置的讀取位元組。從 Docker 執行期獲取此指標。 單位：位元組

指標名稱	維度	描述
StorageWriteBytes	TaskDefinitionFamily，ClusterName ServiceName，ClusterName ClusterName	資源中寫入儲存裝置的位元組數，該資源由您正在使用的維度所指定。從 Docker 執行期獲取此指標。 單位：位元組
TaskCount	ClusterName	叢集中執行的任務數。 單位：計數
TaskSetCount	ServiceName，ClusterName	服務中的任務集數。 單位：計數

> **Note**
>
> 此 EphemeralStorageReserved 與 EphemeralStorageUtilized 指標僅適用於在 Fargate Linux 平台版本 1.4.0 或更新版本上執行的任務。
> Fargate 會在磁盤上保留空間。此空間僅由 Fargate 使用。我們不會向您收費。它不會顯示在這些指標中。但是，您可以在其他工具 (例如 df) 中看到此額外儲存空間。

當您完成 部署 CloudWatch 代理程式以收集 Amazon ECS 上的 EC2 執行個體層級指標 中的步驟後即可使用下列指標

指標名稱	維度	描述
instance_cpu_limit	ClusterName	可指派至此叢集中單一 EC2 執行個體的 CPU 單位數量上限。 單位：無

指標名稱	維度	描述
instance_cpu_reserved_capacity	ClusterName InstanceId ,ContainerInstanceId , ClusterName	目前保留在叢集中單一 EC2 執行個體上的 CPU 百分比。 單位：百分比
instance_cpu_usage_total	ClusterName	叢集中單一 EC2 執行個體正在使用的 CPU 單位數目。 單位：無
instance_cpu_utilization	ClusterName InstanceId ,ContainerInstanceId , ClusterName	叢集中單一 EC2 執行個體正在使用的 CPU 單位總百分比。 單位：百分比
instance_filesystem_utilization	ClusterName InstanceId ,ContainerInstanceId , ClusterName	叢集中單一 EC2 執行個體正在使用的檔案系統容量總百分比。 單位：百分比
instance_memory_limit	ClusterName	可指派至此叢集中單一 EC2 執行個體的記憶體數量上限 (以位元組為單位)。 單位：位元組
instance_memory_reserved_capacity	ClusterName InstanceId ,ContainerInstanceId , ClusterName	目前保留在叢集中單一 EC2 執行個體上的記憶體百分比。 單位：百分比

指標名稱	維度	描述
instance_memory_utilization	ClusterName InstanceId，ContainerInstanceId，ClusterName	叢集中單一 EC2 執行個體正在使用的記憶體總百分比。 單位：百分比
instance_memory_working_set	ClusterName	叢集中單一 EC2 執行個體正在使用的記憶體數量 (以位元組為單位)。 單位：位元組
instance_network_total_bytes	ClusterName	叢集中單一 EC2 執行個體上透過網路傳輸和接收的每秒位元組總數。 單位：位元組/秒
instance_number_of_running_tasks	ClusterName	叢集中單一 EC2 執行個體上執行的任務數目。 單位：計數

Amazon EKS 和 Kubernetes Container Insights 指標

下表列出容器洞見為 Amazon EKS 和 Kubernetes 收集的指標和維度。這些指標會在 ContainerInsights 命名空間中。如需詳細資訊，請參閱 指標。

如果您沒有在主控台中看到任何容器洞見指標，請確定您已完成容器洞見的設定。在完整設定容器洞見前指標都不會出現。如需詳細資訊，請參閱 設定 Container Insights。

如果您使用的是 1.5.0 版或更新版本的 Amazon EKS 附加元件或 CloudWatch 代理程式的 1.300035.0 版本，則會同時收集下表中列出的大多數指標。請參閱表格的「測量結果名稱」資料欄，瞭解未針對 Windows 收集的測量結果。

使用原始版本的 Container Insights，指標會以自訂指標計費。使用 Container Insights 搭配 Amazon EKS 的增強可觀測性，Container Insights 指標會按觀測，而不是存放或擷取的指標計費。如需有關 CloudWatch 定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

> **Note**
>
> 在 Windows 上，不會針對主機處理序容器收集網路度量 (例如pod_network_rx_bytes和pod_network_tx_bytes)。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
cluster_failed_node_count	ClusterName		叢集中失敗的工作者節點數量。如果節點受困於任何節點條件，則會將其判定為失敗。如需詳細資訊，請參閱 Kubernetes 文件中的條件。
cluster_node_count	ClusterName		叢集中的工作者節點總數。
namespace_number_of_running_pods	Namespace ClusterName ClusterName		資源中每個命名空間執行的 pod 數量，該資源由您正在使用的維度所指定。
node_cpu_limit	ClusterName	ClusterName，InstanceId，NodeName	可指派至此叢集中單一節點的 CPU 單位數量上限。
node_cpu_reserved_capacity	NodeName, ClusterName，InstanceId		為節點元件 (例如 kubelet、kube-proxy

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
	ClusterName		和 Docker) 預留的 CPU 單位百分比。 公式：node_cpu_request / node_cpu_limit Note node_cpu_request 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。
node_cpu_usage_total	ClusterName	ClusterName，InstanceId，NodeName	叢集中節點上正在使用的 CPU 單位數量。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
node_cpu_utilization	NodeName, ClusterName , InstanceId ClusterName		叢集中節點上正在使用的 CPU 單位百分比總數。 公式：node_cpu_usage_total / node_cpu_limit

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
node_filesystem_utilization	NodeName, ClusterName , InstanceId ClusterName		叢集中節點上正在使用的檔案系統容量百分比總數。 公式：node_filesystem_usage / node_filesystem_capacity **Note** node_filesystem_usage 和 node_filesystem_capacity 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
node_memory_limit	ClusterName	ClusterName，InstanceId，NodeName	可指派至此叢集中單一節點的記憶體數量上限 (以位元組為單位)。
node_filesystem_inodes 此指標僅適用於具有增強 Amazon EKS 可觀察性的容器洞見。它在視窗上不可用。		ClusterName ClusterName，InstanceId，NodeName	節點上的 inode 總數 (已使用和未使用)。
node_filesystem_inodes_free 此指標僅適用於具有增強 Amazon EKS 可觀察性的容器洞見。它在視窗上不可用。		ClusterName ClusterName，InstanceId，NodeName	節點上未使用的 inode 數目。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
node_memory_reserved_capacity	NodeName, ClusterName , InstanceId ClusterName		目前在叢集中節點上使用的記憶體百分比。 公式：node_memory_request / node_memory_limit **Note** node_memory_request 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
node_memory_utilization	NodeName, ClusterName , InstanceId ClusterName		節點目前使用的記憶體百分比。這是節點記憶體使用量除以節點記憶體限制的百分比。 公式：node_memory_working_set / node_memory_limit 。
node_memory_working_set	ClusterName	ClusterName , InstanceId , NodeName	叢集中運作中一組節點中正在使用的記憶體數量 (以位元組為單位)。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
node_network_total_bytes	NodeName, ClusterName , InstanceId ClusterName		叢集中每個節點每秒透過網路傳輸和接收的位元組總數。 公式：node_network_rx_bytes + node_network_tx_bytes **Note** node_network_rx_bytes 和 node_network_tx_bytes 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
node_number_of_running_containers	NodeName, ClusterName , InstanceId ClusterName		叢集中每個節點執行中的容器數。
node_number_of_running_pods	NodeName, ClusterName , InstanceId ClusterName		叢集中每個節點執行中的 pod 數。
node_status_allocatable_pods 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName , InstanceId , NodeName	可根據節點的可配置資源指派其 Pod 數目，其定義為在計算系統常駐程式保留項目和硬式移出閾值之後節點容量的剩餘部分。
node_status_capacity_pods 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName , InstanceId , NodeName	可根據節點容量指派給節點的 Pod 數目。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
node_status_condition_ready 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName，InstanceId，NodeName	表示節點狀態條件 Ready 是否為 True。
node_status_condition_memory_pressure 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName，InstanceId，NodeName	表示節點狀態條件 MemoryPressure 是否為 True。
node_status_condition_pid_pressure 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName，InstanceId，NodeName	表示節點狀態條件 PIDPressure 是否為 True。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
node_status_condition_disk_pressure 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName，InstanceId，NodeName	表示節點狀態條件 OutOfDisk 是否為 True。
node_status_condition_unknown 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName，InstanceId，NodeName	表示是否有任何節點狀態條件為「未知」。
node_interface_network_rx_dropped 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName，InstanceId，NodeName	節點上網路介面接收並隨後捨棄的封包數目。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
node_interface_network_tx_dropped 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName，InstanceId，NodeName	節點上的網路介面傳輸但捨棄的封包數目。
node_diskio_io_service_bytes_total 此指標僅適用於具有增強 Amazon EKS 可觀察性的容器洞見。它在視窗上不可用。		ClusterName ClusterName，InstanceId，NodeName	節點上所有 I/O 操作傳送的位元組總數。
node_diskio_io_serviced_total 此指標僅適用於具有增強 Amazon EKS 可觀察性的容器洞見。它在視窗上不可用。		ClusterName ClusterName，InstanceId，NodeName	節點上的 I/O 操作總數。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_cpu_reserved_capacity	PodName、命名空間、ClusterName ClusterName	ClusterName ，Namespace ，PodName, FullPodName ClusterName ，Namespace ，Service	叢集中每個 pod 預留的 CPU 容量。 公式：pod_cpu_request / node_cpu_limit **Note** pod_cpu_request 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_cpu_utilization	PodName、命名空間、ClusterName 命名空間、ClusterName 服務、命名空間、ClusterName ClusterName	ClusterName ，Namespace ，PodName, FullPodName	Pod 使用的 CPU 單位百分比。 公式：pod_cpu_usage_total / node_cpu_limit **Note** pod_cpu_usage_total 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_cpu_utilization_over_pod_limit	PodName、命名空間、ClusterName 命名空間、ClusterName 服務、命名空間、ClusterName ClusterName	ClusterName ，Namespace ，PodName, FullPodName	Pod 正在使用的 CPU 單位百分比，此百分比與 Pod 限制相對。 公式：pod_cpu_usage_total / pod_cpu_limit **Note** pod_cpu_usage_total 和 pod_cpu_limit 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_memory_reserved_capacity	PodName、命名空間、ClusterName ClusterName	ClusterName ，Namespace ，PodName, FullPodName ClusterName ，Namespace ，Service	為 Pod 保留的記憶體百分比。 公式：pod_memory_request / node_memory_limit **Note** pod_memory_request 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_memory_utilization	PodName、命名空間、ClusterName 命名空間、ClusterName 服務、命名空間、ClusterName ClusterName	ClusterName ，Namespace ，PodName, FullPodName	Pod 目前使用的記憶體百分比。 公式：pod_memory_working_set / node_memory_limit **Note** pod_memory_working_set 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_memory_utilization_over_pod_limit	PodName、命名空間、ClusterName 命名空間、ClusterName 服務、命名空間、ClusterName ClusterName	ClusterName ，Namespace ，PodName, FullPodName	Pod 正在使用的記憶體百分比，此百分比與 Pod 限制相對。如果 Pod 中有任何容器未定義記憶體限制，這個指標將不會顯示。 公式：pod_memory_working_set / pod_memory_limit **Note** pod_memory_working_set 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_network_rx_bytes	PodName、命名空間、ClusterName 命名空間、ClusterName 服務、命名空間、ClusterName ClusterName	ClusterName，Namespace，PodName, FullPodName	Pod 透過網路每秒接收的位元組數。 公式：sum(pod_interface_network_rx_bytes) **Note** pod_interface_network_rx_bytes 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_network_tx_bytes	PodName、命名空間、ClusterName 命名空間、ClusterName 服務、命名空間、ClusterName ClusterName	ClusterName，Namespace，PodName, FullPodName	Pod 透過網路每秒傳輸的位元組數。 公式：sum(pod_interface_network_tx_bytes) **Note** pod_interface_network_tx_bytes 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_cpu_request 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	Pod 的 CPU 請求。 公式：sum(container_cpu_request) **Note** pod_cpu_request 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_memory_request 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	Pod 的記憶體請求。 公式：sum(container_memory_request) **Note** pod_memory_request 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_cpu_limit 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	Pod 中的容器定義的 CPU 限制。如果 Pod 中有任何容器未定義 CPU 限制，這個指標將不會顯示。 公式：sum(container_cpu_limit) **Note** pod_cpu_limit 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_memory_limit 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	Pod 中的容器定義的記憶體限制。如果 Pod 中有任何容器未定義記憶體限制，這個指標將不會顯示。 公式：sum(container_memory_limit) **Note** pod_cpu_limit 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_status_failed 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	表示 Pod 中的所有容器都已終止，且至少有一個容器已經以非零狀態終止，或已由系統終止。
pod_status_ready 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	表示 Pod 中的所有容器都已就緒，且已達到條件 Container Ready 。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_status_running 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	表示 Pod 中的所有容器都在執行中。
pod_status_scheduled 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	表示 Pod 已排程至節點。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_status_unknown 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	表示無法取得 Pod 的狀態。
pod_status_pending 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	表示叢集已接受 Pod，但有一個或多個容器尚未準備就緒。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_status_succeeded 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	表示 Pod 中的所有容器都已成功終止，而且不會重新啟動。
pod_number_of_containers 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	報告 Pod 規格中定義的容器數目。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_number_of_running_containers 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	報告 Pod 中目前處於 Running 狀態的容器數目。
pod_container_status_terminated 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	報告 Pod 中處於 Terminated 狀態的容器數目。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_container_status_running 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	報告 Pod 中處於 Running 狀態的容器數目。
pod_container_status_waiting 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	報告 Pod 中處於 Waiting 狀態的容器數目。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
pod_interface_network_rx_dropped 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	Pod 上網路介面接收並隨後捨棄的封包數目。
pod_interface_network_tx_dropped 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName Namespace , ClusterName , Service ClusterName , Namespace , PodName, FullPodName	針對 Pod 傳輸但捨棄的封包數目。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
container_cpu_utilization 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName , ContainerName PodName, Namespace , ClusterName , ContainerName , FullPodName	容器使用的 CPU 單位百分比。 公式：container_cpu_usage_total / node_cpu_limit **Note** container_cpu_utilization 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
container_cpu_utilization_over_container_limit 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName , ContainerName PodName, Namespace , ClusterName , ContainerName , FullPodName	相對於容器限制，容器正在使用的 CPU 單位百分比。如果容器未定義 CPU 限制，這個指標將不會顯示。 公式：container_cpu_usage_total / container_cpu_limit **Note** container_cpu_utilization_over_container_limit 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
container_memory_utilization 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName , ContainerName PodName, Namespace , ClusterName , ContainerName , FullPodName	容器使用的記憶體單位百分比。 公式：container_memory_working_set / node_memory_limit **Note** container_memory_utilization 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
container_memory_utilization_over_container_limit 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName , ContainerName PodName, Namespace , ClusterName , ContainerName , FullPodName	相對於容器限制，容器正在使用的記憶體單位百分比。如果容器未定義記憶體限制，這個指標將不會顯示。 公式：container_memory_working_set / container_memory_limit **Note** container_memory_utilization_over_container_limit 不會直接回報為指標，而是效能日誌事件中的欄位。如需詳細資訊，請參閱 Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
container_memory_failures_total 此指標僅適用於具有增強 Amazon EKS 可觀察性的容器洞見。它在視窗上不可用。		ClusterName PodName, Namespace , ClusterName , ContainerName PodName, Namespace , ClusterName , ContainerName , FullPodName	容器經歷的記憶體配置失敗次數。
pod_number_of_container_restarts	PodName, Namespace , ClusterName		Pod 中重新啟動的容器總數。
service_number_of_running_pods	服務、Namespace 、ClusterName ClusterName		叢集中執行服務的 Pod 數量。
replicas_desired 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName	工作負載規格中定義的工作負載所需 Pod 數目。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
replicas_ready 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName	已達到就緒狀態的工作負載 Pod 數目。
status_replicas_available 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName	可供工作負載使用的 Pod 數目。當 Pod 準備好用於工作負載規格中定義的 minReadySeconds 時可供使用。
status_replicas_unavailable 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName PodName, Namespace , ClusterName	工作負載無法使用的 Pod 數目。當 Pod 準備好用於工作負載規格中定義的 minReadySeconds 時可供使用。如果 Pod 不符合此條件，則無法使用。
apiserver_storage_objects 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName , resource	上次檢查時存放在 etcd 中的物件數目。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
apiserver_request_total 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName , code, verb	向 Kubernetes API 伺服器發出的 API 請求總數。
apiserver_request_duration_seconds 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName , verb	向 Kubernetes API 伺服器發出的 API 請求的回應延遲。
apiserver_admission_controller_admission_duration_seconds 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName , operation	許可控制器延遲 (以秒為單位)。許可控制器是攔截向 Kubernetes API 伺服器發出請求的程式碼。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
rest_client_request_duration_seconds 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName，operation	用戶端呼叫 Kubernetes API 伺服器時遇到的回應延遲。此指標為實驗性質，並且可能會在 Kubernetes 的未來版本中變更。
rest_client_requests_total 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName，code, method	用戶端向 Kubernetes API 伺服器發出的 API 請求總數。此指標為實驗性質，並且可能會在 Kubernetes 的未來版本中變更。
etcd_request_duration_seconds 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName，operation	對 Etcd 的 API 呼叫的回應延遲。此指標為實驗性質，並且可能會在 Kubernetes 的未來版本中變更。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
apiserver_storage_size_bytes 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName，endpoint	實體配置的儲存資料庫檔案大小 (以位元組為單位)。此指標為實驗性質，並且可能會在 Kubernetes 的未來版本中變更。
apiserver_longrunning_requests 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName，resource	向 Kubernetes API 伺服器發出的長時間執行的作用中請求數目。
apiserver_current_inflight_requests 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName，request_kind	Kubernetes API 伺服器正在處理的請求數目。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
apiserver_admission_webhook_admission_duration_seconds 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName , name	許可 Webhook 延遲 (以秒為單位)。許可 Webhook 是接收許可請求並對其執行某些操作的 HTTP 回呼。
apiserver_admission_step_admission_duration_seconds 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName , operation	許可子步驟延遲 (以秒為單位)。
apiserver_requested_deprecated_apis 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName , group	在 Kubernetes API 伺服器上發出的取代 API 的請求數目。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述
apiserver_request_total_5XX 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName，code, verb	向 Kubernetes API 伺服器發出的請求數目，該請求使用 5XX HTTP 回應碼做出回應。
apiserver_storage_list_duration_seconds 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName，resource	Etcd 列示物件的回應延遲。此指標為實驗性質，並且可能會在 Kubernetes 的未來版本中變更。
apiserver_current_inqueue_requests 此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性		ClusterName ClusterName，request_kind	Kubernetes API 伺服器排入佇列的請求數目。此指標為實驗性質，並且可能會在 Kubernetes 的未來版本中變更。

指標名稱	任何版本 Container Insights 的維度	Container Insights 搭配 Amazon EKS 的增強可觀測性的額外維度	描述

`apiserver_flowcontrol_rejected_requests_total`
 	
此指標僅適用於 Container Insights 搭配 Amazon EKS 的增強可觀測性
		`ClusterName`
		
		`ClusterName`，`reason`
			API 優先順序與公平性子系統拒絕的請求數目。此指標為實驗性質，並且可能會在 Kubernetes 的未來版本中變更。

GPU 指標

從 CloudWatch 代理程式版`1.300034.0`本開始，具有增強 Amazon EKS 可觀察性的容器洞察預設會從 EKS 工作負載收集 NVIDIA GPU 指標。必須使用可 CloudWatch 觀察性 EKS 附加元件版本`v1.3.0-eksbuild.1`或更新版本來安裝 CloudWatch 代理程式。如需詳細資訊，請參閱 使用 Amazon CloudWatch 可觀測 EKS 附加元件安裝 CloudWatch 代理程式。這些收集到的 NVIDIA GPU 測量結果會列在本節的表格中。

若要讓容器深入解析收集 NVIDIA GPU 指標，您必須符合下列先決條件：

- 您必須搭配 Amazon EKS 附加元件版本或更新版本使用具有增強可 CloudWatch 觀察性的容器洞見。`v1.3.0-eksbuild.1`
- 您必須在叢集中安裝適用於庫伯尼特的 NVIDIA 裝置外掛程式。
- NVIDIA 容器工具組必須安裝在叢集的節點上。例如，Amazon EKS 最佳化的加速 AMI 是使用必要的元件建置的。

您可以將`false`開始 CloudWatch 代理程式設定檔中的`accelerated_compute_metrics`選項設定為，選擇不收集 NVIDIA GPU 指標。如需詳細資訊和退出設定範例，請參閱(選用) 額外組態。

指標名稱	維度	描述
container_gpu_memory_total	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , GpuDevice	配置給容器的 GPU 上的畫面緩衝區總大小 (以位元組為單位)。
container_gpu_memory_used	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , GpuDevice	分配給容器的 GPU 上使用的幀緩衝區的字節。
container_gpu_memory_utilization	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName	分配給容器的 GPU 使用的框架緩衝區百分比。

指標名稱	維度	描述
	ClusterName , Namespace , PodName, FullPodName , ContainerName , GpuDevice	
container_gpu_power_draw	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , GpuDevice	分配給容器的 GPU 的功率使用瓦特。
container_gpu_temperature	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , GpuDevice	分配給容器的 GPU（S）的攝氏度的溫度。

指標名稱	維度	描述
container_gpu_utilization	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , GpuDevice	配置給容器的 GPU 使用率百分比。
node_gpu_memory_total	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, GpuDevice	配置給節點的 GPU 上的畫面緩衝區總大小 (以位元組為單位)。
node_gpu_memory_used	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, GpuDevice	分配給節點的 GPU 上使用的幀緩衝區的字節。

指標名稱	維度	描述
node_gpu_memory_utilization	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, GpuDevice	分配給節點的 GPU 上使用的框架緩衝區百分比。
node_gpu_power_draw	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, GpuDevice	配置給節點之 GPU 的電源使用量 (瓦特)。
node_gpu_temperature	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, GpuDevice	分配給節點的 GPU (s) 的攝氏度的溫度。
node_gpu_utilization	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, GpuDevice	配置給節點的 GPU 使用率百分比。

指標名稱	維度	描述
pod_gpu_memory_total	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName . GpuDevice	配置給網繭的 GPU 上的框架緩衝區總大小 (以位元組為單位)。
pod_gpu_memory_used	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName . GpuDevice	配置給網繭之 GPU 上使用的框架緩衝區位元組。

指標名稱	維度	描述
pod_gpu_memory_utilization	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName . GpuDevice	配置給網繭之 GPU 使用的框架緩衝區百分比。
pod_gpu_power_draw	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName . GpuDevice	配置給網繭之 GPU 的電源使用量 (瓦特)。

指標名稱	維度	描述
pod_gpu_temperature	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName . GpuDevice	配置給網繭之 GPU 的攝氏度溫度。
pod_gpu_utilization	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , GpuDevice	配置給網繭的 GPU 使用率百分比。

AWS 針對 AWS 小靈和推論的神經元指標 AWS

從 CloudWatch 代理程式版本開始，具有增強 Amazon EKS 可觀察性1.300036.0的容器洞見預設會從 AWS Trainium 和 AWS 推論加速器收集加速運算指標。必須使用可 CloudWatch 觀察性 EKS 附加元件版本v1.5.0-eksbuild.1或更新版本來安裝 CloudWatch 代理程式。如需附加元件的詳細資訊，請參閱使用 Amazon CloudWatch 可觀測 EKS 附加元件安裝 CloudWatch 代理程式。有關翠鳥的更多信息，請參閱 AWSAWS 小草。如需有關推論的詳細資訊，請參閱 AWSAWS 推論。

若要讓容器深入解析收集 AWS 神經元指標，您必須符合下列先決條件：

- 您必須搭配 Amazon EKS 附加元件版本或更新版本使用具有增強可 CloudWatch 觀察性的容器洞見。`v1.5.0-eksbuild.1`
- 神經元驅動程式必須安裝在叢集的節點上。
- 神經元裝置外掛程式必須安裝在叢集上。例如，Amazon EKS 最佳化的加速 AMI 是使用必要的元件建置的。

收集的測量結果會列在此段落的表格中。收集的指標是針對指標 AWS，推論, 和 AWS 推論 2. AWS

CloudWatch 代理程式會從 N euron 監視器收集這些指標，並執行必要的 Kubernetes 資源關聯，以便在網繭和容器層級提供度量

指標名稱	維度	描述

`container_neuroncore_utilization`	`ClusterName`
	
	`ClusterName`，`Namespace`，`PodName`, `ContainerName`
	
	`ClusterName`，`Namespace`，`PodName`, `FullPodName`，`ContainerName`
	
	`ClusterName`，`Namespace`，`PodName`, `FullPodName`，`ContainerName`，`NeuronDevice`，`NeuronCore`
		NeuronCore 使用率，在 NeuronCore 配置給容器的擷取期間內。
		
		單位：百分比

`container_neuroncore_memory_usage_constants`	`ClusterName`
	
	`ClusterName`，`Namespace`，`PodName`, `ContainerName`
	
	`ClusterName`，`Namespace`，`PodName`, `FullPodName`，`ContainerName`
		訓練期間用於常數的裝置記憶體量 NeuronCore，分配給容器 (或推論期間的權重)。
		
		單位：位元組

指標名稱	維度	描述
	ClusterName , Namespace , PodName, FullPodName , ContainerName , NeuronDevice , NeuronCore	
container_neuroncore_memory_usage_model_code	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , NeuronDevice , NeuronCore	由配置給容器的模型的可執行程式碼所使用的 NeuronCore 裝置記憶體量。 單位：位元組
container_neuroncore_memory_usage_model_shared_scratchpad	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , NeuronDevice , NeuronCore	分配給容器的模型共用的暫存板所使用的裝置 NeuronCore 記憶體量。此記憶體區域會保留給模型使用。 單位：位元組

指標名稱	維度	描述
container_neuroncore_memory_usage_runtime_memory	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , NeuronDevice , NeuronCore	NeuronCore 配置給容器的 Neuron 執行階段所使用的裝置記憶體量。 單位：位元組
container_neuroncore_memory_usage_tensors	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , NeuronDevice , NeuronCore	NeuronCore 分配給容器的張量使用的裝置記憶體量。 單位：位元組

指標名稱	維度	描述
container_neuroncore_memory_usage_total	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , NeuronDevice , NeuronCore	NeuronCore 配置給容器所使用的記憶體總量。 單位：位元組
container_neurondevice_hw_ecc_events_total	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , NeuronDevice	節點上神經元裝置的晶片上 SRAM 和裝置記憶體的校正和未修正的 ECC 事件數目。 單位：計數

指標名稱	維度	描述
pod_neuroncore_utilization	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , NeuronDevice , NeuronCore	NeuronCore 配置給網繭的擷取期間內的 NeuronCore 使用率。 單位：百分比
pod_neuroncore_memory_usage_constants	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , NeuronDevice , NeuronCore	由配置給網繭 (或推論期間的權重) 在訓練期間用於常數的裝置記憶體量。NeuronCore 單位：位元組

指標名稱	維度	描述
pod_neuroncore_memory_usage_model_code	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , NeuronDevice , NeuronCore	配置給網繭之模型之可執行程式碼所使用的裝置記憶體量。 NeuronCore 單位：位元組
pod_neuroncore_memory_usage_model_shared_scratchpad	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , NeuronDevice , NeuronCore	分配給網繭之模型共用之暫存板所使用的裝置 NeuronCore 記憶體量。此記憶體區域會保留給模型使用。 單位：位元組

指標名稱	維度	描述
pod_neuroncore_memory_usage_runtime_memory	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , NeuronDevice , NeuronCore	NeuronCore 配置給網繭之 Neuron 執行階段所使用的裝置記憶體量。 單位：位元組
pod_neuroncore_memory_usage_tensors	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , NeuronDevice , NeuronCore	NeuronCore 配置給網繭之張量使用的裝置記憶體量。 單位：位元組

指標名稱	維度	描述
pod_neuroncore_memory_usage_total	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , NeuronDevice , NeuronCore	NeuronCore 配置給網繭之使用的記憶體總量。 單位：位元組
pod_neurondevice_hw_ecc_events_total	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , NeuronDevice	配置給網繭的神經元裝置的晶片上 SRAM 和裝置記憶體的已校正和未修正的 ECC 事件數目。 單位：位元組

指標名稱	維度	描述
node_neuroncore_utilization	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceType , InstanceId , NodeName, NeuronDevice , NeuronCore	NeuronCore 配置給節點之擷取期間內的 NeuronCore 使用率。 單位：百分比
node_neuroncore_memory_usage_constants	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceType , InstanceId , NodeName, NeuronDevice , NeuronCore	訓練期間配置給節點 (或推論期間的權重)，用於常數的裝置記憶體量。NeuronCore 單位：位元組
node_neuroncore_memory_usage_model_code	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceType , InstanceId , NodeName, NeuronDevice , NeuronCore	用於模型之可執行程式碼的裝置記憶體量 NeuronCore ，由配置給節點。 單位：位元組
node_neuroncore_memory_usage_model_shared_scratchpad	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceType , InstanceId , NodeName, NeuronDevice , NeuronCore	分配給節點的模型共用的暫存板所使用的裝置 NeuronCore 記憶體量。這是為模型保留的記憶體區域。 單位：位元組

指標名稱	維度	描述
node_neuroncore_memory_usage_runtime_memory	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceType , InstanceId , NodeName, NeuronDevice , NeuronCore	配置給節點的 Neuron 執行階段所使用的裝置記憶體量。 NeuronCore 單位：位元組
node_neuroncore_memory_usage_tensors	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceType , InstanceId , NodeName, NeuronDevice , NeuronCore	由配置給節點的張量使用的裝置記憶體量。 NeuronCore 單位：位元組
node_neuroncore_memory_usage_total	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceType , InstanceId , NodeName, NeuronDevice , NeuronCore	配置給節點的記憶體使用的總量。 NeuronCore 單位：位元組
node_neuron_execution_errors_total	ClusterName ClusterName , InstanceId , NodeName	節點上的執行錯誤總數。這是由 CloudWatch 代理程式彙總下列類型的錯誤來計算：genericnumerical 、、transient 、modelruntime、和 hardware 單位：計數

指標名稱	維度	描述
node_neuron device_runtime_memory_used_bytes	ClusterName ClusterName ，InstanceId ，NodeName	節點上的 Neuron 裝置記憶體使用量總計 (以位元組為單位)。 單位：位元組
node_neuron_execution_latency	ClusterName ClusterName ，InstanceId ，NodeName	以秒為單位，Neuron 執行階段測量的節點上執行的延遲。 單位：秒
node_neuron device_hw_ecc_events_total	ClusterName ClusterName ，InstanceId ，NodeName ClusterName ，InstanceId ，NodeName, NeuronDevice	節點上神經元裝置的晶片上 SRAM 和裝置記憶體的校正和未修正的 ECC 事件數目。 單位：計數

AWS Elastic Fabric Adapter (EFA) 指標

從 CloudWatch 代理程式版本1.300037.0開始，具有增強 Amazon EKS 可觀察性的容器洞察會從 Linux 執行個體上的 Amazon EKS 叢集收集 AWS 彈性結構介面卡 (EFA) 指標。必須使用可 CloudWatch 觀察性 EKS 附加元件版本v1.5.2-eksbuild.1或更新版本來安裝 CloudWatch 代理程式。如需附加元件的詳細資訊，請參閱使用 Amazon CloudWatch 可觀測 EKS 附加元件安裝 CloudWatch 代理程式。如需 AWS 彈性織物介面卡的詳細資訊，請參閱彈性織物配接器。

若要讓容器深入解析收集 AWS 彈性網狀架構介面卡度量，您必須符合下列先決條件：

- 您必須搭配 Amazon EKS 附加元件版本或更新版本使用具有增強可 CloudWatch 觀察性的容器洞見。v1.5.2-eksbuild.1
- 必須在叢集上安裝 EFA 裝置外掛程式。有關詳細資訊，請參閱中s-device-plugin的 aws-efa-k8 GitHub。

下表列出收集的測量結果。

指標名稱	維度	描述
container_efa_rx_bytes	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , EfaDevice	配置給容器的 EFA 裝置每秒接收的位元組數目。 單位：位元組/秒
container_efa_tx_bytes	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , EfaDevice	配置給容器的 EFA 裝置每秒傳輸的位元組數。 單位：位元組/秒
container_efa_rx_dropped	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName	配置給容器的 EFA 裝置接收並捨棄的封包數目。 單位：計數/秒

指標名稱	維度	描述
	ClusterName , Namespace , PodName, FullPodName , ContainerName , EfaDevice	
container_efa_rdma_read_bytes	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , EfaDevice	配置給容器的 EFA 裝置使用遠端直接記憶體存取讀取作業接收的每秒位元組數。 單位：位元組/秒
container_efa_rdma_write_bytes	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , EfaDevice	配置給容器的 EFA 裝置使用遠端直接記憶體存取讀取作業，每秒傳輸的位元組數目。 單位：位元組/秒

指標名稱	維度	描述
container_efa_rdma_write_recv_bytes	ClusterName ClusterName , Namespace , PodName, ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName ClusterName , Namespace , PodName, FullPodName , ContainerName , EfaDevice	配置給容器的 EFA 裝置在遠端直接記憶體存取寫入作業期間，每秒接收到的位元組數目。 單位：位元組/秒
pod_efa_rx_bytes	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , EfaDevice	配置給網繭的 EFA 裝置每秒接收的位元組數。 單位：位元組/秒

指標名稱	維度	描述
pod_efa_tx_bytes	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , EfaDevice	配置給網繭的 EFA 裝置每秒傳輸的位元組數目。 單位：位元組/秒
pod_efa_rx_dropped	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , EfaDevice	配置給網繭的 EFA 裝置接收並捨棄的封包數目。 單位：計數/秒

指標名稱	維度	描述
pod_efa_rdma_read_bytes	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , EfaDevice	配置給網繭的 EFA 裝置使用遠端直接記憶體存取讀取作業，每秒接收的位元組數目。 單位：位元組/秒
pod_efa_rdma_write_bytes	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , EfaDevice	配置給網繭的 EFA 裝置使用遠端直接記憶體存取讀取作業，每秒傳輸的位元組數目。 單位：位元組/秒

指標名稱	維度	描述
pod_efa_rdma_write_recv_bytes	ClusterName ClusterName , Namespace ClusterName , Namespace , Service ClusterName , Namespace , PodName ClusterName , Namespace , PodName, FullPodName ClusterName , Namespace , PodName, FullPodName , EfaDevice	配置給網繭的 EFA 裝置在遠端直接記憶體存取寫入作業期間，每秒接收到的位元組數目。 單位：位元組/秒
node_efa_rx_bytes	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, EfaDevice	配置給節點的 EFA 裝置每秒接收的位元組數。 單位：位元組/秒
node_efa_tx_bytes	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, EfaDevice	配置給節點的 EFA 裝置每秒傳輸的位元組數。 單位：位元組/秒

指標名稱	維度	描述
node_efa_rx_dropped	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, EfaDevice	配置給節點的 EFA 裝置接收並捨棄的封包數目。 單位：計數/秒
node_efa_rdma_read_bytes	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, EfaDevice	配置給節點的 EFA 裝置使用遠端直接記憶體存取讀取作業，每秒接收的位元組數目。 單位：位元組/秒
pod_efa_rdma_write_bytes	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, EfaDevice	配置給網繭的 EFA 裝置使用遠端直接記憶體存取讀取作業，每秒傳輸的位元組數目。 單位：位元組/秒
node_efa_rdma_write_recv_bytes	ClusterName ClusterName , InstanceId , NodeName ClusterName , InstanceId , InstanceType , NodeName, EfaDevice	配置給節點的 EFA 裝置在遠端直接記憶體存取寫入作業期間，每秒接收到的位元組數目。 單位：位元組/秒

Container Insights 效能日誌參考

本節包含 Container Insights 如何使用效能日誌事件收集指標的參考資訊。當您部署 Container Insights 時，它會自動建立效能日誌事件的日誌群組。您不需要自行建立此日誌群組。

主題

- Amazon ECS 的 Container Insights 效能日誌事件
- Amazon EKS 和 Kubernetes 的 Container Insights 效能日誌事件
- Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位

Amazon ECS 的 Container Insights 效能日誌事件

以下是 Container Insights 從 Amazon ECS 中收集的效能日誌事件範例。

這些記錄檔位於 CloudWatch 記錄檔中的記錄群組中，名為`/aws/ecs/containerinsights/`*`CLUSTER_NAME`*`/performance`. 在該日誌群組中，每個容器執行個體都會有日誌串流 `AgentTelemetry-`*`CONTAINER_INSTANCE_ID`*。

您可以使用 `{ $.Type = "Container" }` 等查詢來查詢這些日誌，以檢視所有容器日誌事件。

類型：容器

```
{
 "Version":"0",
 "Type":"Container",
 "ContainerName":"sleep",
 "TaskId":"7ac4dfba69214411b4783a3b8189c9ba",
 "TaskDefinitionFamily":"sleep360",
 "TaskDefinitionRevision":"1",
 "ContainerInstanceId":"0d7650e6dec34c1a9200f72098071e8f",
 "EC2InstanceId":"i-0c470579dbcdbd2f3",
 "ClusterName":"MyCluster",
 "Image":"busybox",
 "ContainerKnownStatus":"RUNNING",
 "Timestamp":1623963900000,
 "CpuUtilized":0.0,
 "CpuReserved":10.0,
 "MemoryUtilized":0,
 "MemoryReserved":10,
 "StorageReadBytes":0,
```

```
 "StorageWriteBytes":0,
 "NetworkRxBytes":0,
 "NetworkRxDropped":0,
 "NetworkRxErrors":0,
 "NetworkRxPackets":14,
 "NetworkTxBytes":0,
 "NetworkTxDropped":0,
 "NetworkTxErrors":0,
 "NetworkTxPackets":0
}
```

類型：任務

```
{
    "Version": "0",
    "Type": "Task",
    "TaskId": "7ac4dfba69214411b4783a3b8189c9ba",
    "TaskDefinitionFamily": "sleep360",
    "TaskDefinitionRevision": "1",
    "ContainerInstanceId": "0d7650e6dec34c1a9200f72098071e8f",
    "EC2InstanceId": "i-0c470579dbcdbd2f3",
    "ClusterName": "MyCluster",
    "AccountID": "637146863587",
    "Region": "us-west-2",
    "AvailabilityZone": "us-west-2b",
    "KnownStatus": "RUNNING",
    "LaunchType": "EC2",
    "PullStartedAt": 1623963608201,
    "PullStoppedAt": 1623963610065,
    "CreatedAt": 1623963607094,
    "StartedAt": 1623963610382,
    "Timestamp": 1623963900000,
    "CpuUtilized": 0.0,
    "CpuReserved": 10.0,
    "MemoryUtilized": 0,
    "MemoryReserved": 10,
    "StorageReadBytes": 0,
    "StorageWriteBytes": 0,
    "NetworkRxBytes": 0,
    "NetworkRxDropped": 0,
    "NetworkRxErrors": 0,
    "NetworkRxPackets": 14,
    "NetworkTxBytes": 0,
```

```
    "NetworkTxDropped": 0,
    "NetworkTxErrors": 0,
    "NetworkTxPackets": 0,
    "EBSFilesystemUtilized": 10,
    "EBSFilesystemSize": 20,
    "CloudWatchMetrics": [
        {
            "Namespace": "ECS/ContainerInsights",
            "Metrics": [
                {
                    "Name": "CpuUtilized",
                    "Unit": "None"
                },
                {
                    "Name": "CpuReserved",
                    "Unit": "None"
                },
                {
                    "Name": "MemoryUtilized",
                    "Unit": "Megabytes"
                },
                {
                    "Name": "MemoryReserved",
                    "Unit": "Megabytes"
                },
                {
                    "Name": "StorageReadBytes",
                    "Unit": "Bytes/Second"
                },
                {
                    "Name": "StorageWriteBytes",
                    "Unit": "Bytes/Second"
                },
                {
                    "Name": "NetworkRxBytes",
                    "Unit": "Bytes/Second"
                },
                {
                    "Name": "NetworkTxBytes",
                    "Unit": "Bytes/Second"
                },
                {
                    "Name": "EBSFilesystemSize",
                    "Unit": "Gigabytes"
```

```
                },
                {
                    "Name": "EBSFilesystemUtilzed",
                    "Unit": "Gigabytes"
                }
            ],
            "Dimensions": [
                ["ClusterName"],
                [
                    "ClusterName",
                    "TaskDefinitionFamily"
                ]
            ]
        }
    ]
}
```

類型：服務

```
{
    "Version": "0",
    "Type": "Service",
    "ServiceName": "myCIService",
    "ClusterName": "myCICluster",
    "Timestamp": 1561586460000,
    "DesiredTaskCount": 2,
    "RunningTaskCount": 2,
    "PendingTaskCount": 0,
    "DeploymentCount": 1,
    "TaskSetCount": 0,
    "CloudWatchMetrics": [
        {
            "Namespace": "ECS/ContainerInsights",
            "Metrics": [
                {
                    "Name": "DesiredTaskCount",
                    "Unit": "Count"
                },
                {
                    "Name": "RunningTaskCount",
                    "Unit": "Count"
                },
                {
```

```
                    "Name": "PendingTaskCount",
                    "Unit": "Count"
                },
                {
                    "Name": "DeploymentCount",
                    "Unit": "Count"
                },
                {
                    "Name": "TaskSetCount",
                    "Unit": "Count"
                }
            ],
            "Dimensions": [
                [
                    "ServiceName",
                    "ClusterName"
                ]
            ]
        }
    ]
}
```

類型：體積

```
{
    "Version": "0",
    "Type": "Volume",
    "TaskDefinitionFamily": "myCITaskDef",
    "TaskId": "7ac4dfba69214411b4783a3b8189c9ba",
    "ClusterName": "myCICluster",
    "ServiceName": "myCIService",
    "VolumeId": "vol-1233436545ff708cb",
    "InstanceId": "i-0c470579dbcdbd2f3",
    "LaunchType": "EC2",
    "VolumeName": "MyVolumeName",
    "EBSFilesystemUtilized": 10,
    "EBSFilesystemSize": 20,
    "CloudWatchMetrics": [
        {
            "Namespace": "ECS/ContainerInsights",
            "Metrics": [
                {
                    "Name": "EBSFilesystemSize",
```

```
                    "Unit": "Gigabytes"
                },
                {
                    "Name": "EBSFilesystemUtilzed",
                    "Unit": "Gigabytes"
                }
            ],
            "Dimensions": [
                ["ClusterName"],
                [
                    "VolumeName",
                    "TaskDefinitionFamily",
                    "ClusterName"
                ],
                [
                    "ServiceName",
                    "ClusterName"
                ]
            ]
        }
    ]
}
```

類型：叢集

```
{
    "Version": "0",
    "Type": "Cluster",
    "ClusterName": "myCICluster",
    "Timestamp": 1561587300000,
    "TaskCount": 5,
    "ContainerInstanceCount": 5,
    "ServiceCount": 2,
    "CloudWatchMetrics": [
        {
            "Namespace": "ECS/ContainerInsights",
            "Metrics": [
                {
                    "Name": "TaskCount",
                    "Unit": "Count"
                },
                {
                    "Name": "ContainerInstanceCount",
```

```
                    "Unit": "Count"
                },
                {
                    "Name": "ServiceCount",
                    "Unit": "Count"
                }
            ],
            "Dimensions": [
                [
                    "ClusterName"
                ]
            ]
        }
    ]
}
```

Amazon EKS 和 Kubernetes 的 Container Insights 效能日誌事件

以下是 Container Insights 從 Amazon EKS 和 Kubernetes 叢集所收集效能日誌事件的範例。

類型：節點

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "node_cpu_utilization"
        },
        {
          "Unit": "Percent",
          "Name": "node_memory_utilization"
        },
        {
          "Unit": "Bytes/Second",
          "Name": "node_network_total_bytes"
        },
        {
          "Unit": "Percent",
          "Name": "node_cpu_reserved_capacity"
        },
```

```
        {
          "Unit": "Percent",
          "Name": "node_memory_reserved_capacity"
        },
        {
          "Unit": "Count",
          "Name": "node_number_of_running_pods"
        },
        {
          "Unit": "Count",
          "Name": "node_number_of_running_containers"
        }
      ],
      "Dimensions": [
        [
          "NodeName",
          "InstanceId",
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    },
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "node_cpu_utilization"
        },
        {
          "Unit": "Percent",
          "Name": "node_memory_utilization"
        },
        {
          "Unit": "Bytes/Second",
          "Name": "node_network_total_bytes"
        },
        {
          "Unit": "Percent",
          "Name": "node_cpu_reserved_capacity"
        },
        {
          "Unit": "Percent",
          "Name": "node_memory_reserved_capacity"
        },
```

```
        {
          "Unit": "Count",
          "Name": "node_number_of_running_pods"
        },
        {
          "Unit": "Count",
          "Name": "node_number_of_running_containers"
        },
        {
          "Name": "node_cpu_usage_total"
        },
        {
          "Name": "node_cpu_limit"
        },
        {
          "Unit": "Bytes",
          "Name": "node_memory_working_set"
        },
        {
          "Unit": "Bytes",
          "Name": "node_memory_limit"
        }
      ],
      "Dimensions": [
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "Sources": [
    "cadvisor",
    "/proc",
    "pod",
    "calculated"
  ],
  "Timestamp": "1567096682364",
  "Type": "Node",
  "Version": "0",
```

```
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal"
  },
  "node_cpu_limit": 4000,
  "node_cpu_request": 1130,
  "node_cpu_reserved_capacity": 28.249999999999996,
  "node_cpu_usage_system": 33.794636630852764,
  "node_cpu_usage_total": 136.47852169244098,
  "node_cpu_usage_user": 71.67075111567326,
  "node_cpu_utilization": 3.4119630423110245,
  "node_memory_cache": 3103297536,
  "node_memory_failcnt": 0,
  "node_memory_hierarchical_pgfault": 0,
  "node_memory_hierarchical_pgmajfault": 0,
  "node_memory_limit": 16624865280,
  "node_memory_mapped_file": 406646784,
  "node_memory_max_usage": 4230746112,
  "node_memory_pgfault": 0,
  "node_memory_pgmajfault": 0,
  "node_memory_request": 1115684864,
  "node_memory_reserved_capacity": 6.7109407818311055,
  "node_memory_rss": 798146560,
  "node_memory_swap": 0,
  "node_memory_usage": 3901444096,
  "node_memory_utilization": 6.601302600149552,
  "node_memory_working_set": 1097457664,
  "node_network_rx_bytes": 35918.392817386324,
  "node_network_rx_dropped": 0,
  "node_network_rx_errors": 0,
  "node_network_rx_packets": 157.67565245448117,
  "node_network_total_bytes": 68264.20276554905,
  "node_network_tx_bytes": 32345.80994816272,
  "node_network_tx_dropped": 0,
  "node_network_tx_errors": 0,
  "node_network_tx_packets": 154.21455923431654,
  "node_number_of_running_containers": 16,
  "node_number_of_running_pods": 13
}
```

類型：NodeFS

```
{
```

```
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "node_filesystem_utilization"
        }
      ],
      "Dimensions": [
        [
          "NodeName",
          "InstanceId",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "EBSVolumeId": "aws://us-west-2b/vol-0a53108976d4a2fda",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "Sources": [
    "cadvisor",
    "calculated"
  ],
  "Timestamp": "1567097939726",
  "Type": "NodeFS",
  "Version": "0",
  "device": "/dev/nvme0n1p1",
  "fstype": "vfs",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal"
  },
  "node_filesystem_available": 17298395136,
  "node_filesystem_capacity": 21462233088,
  "node_filesystem_inodes": 10484720,
  "node_filesystem_inodes_free": 10367158,
```

```
  "node_filesystem_usage": 4163837952,
  "node_filesystem_utilization": 19.400767547940255
}
```

類型： NodeDiskIO

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "EBSVolumeId": "aws://us-west-2b/vol-0a53108976d4a2fda",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "Sources": [
    "cadvisor"
  ],
  "Timestamp": "1567096928131",
  "Type": "NodeDiskIO",
  "Version": "0",
  "device": "/dev/nvme0n1",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal"
  },
  "node_diskio_io_service_bytes_async": 9750.505814277016,
  "node_diskio_io_service_bytes_read": 0,
  "node_diskio_io_service_bytes_sync": 230.6174506688036,
  "node_diskio_io_service_bytes_total": 9981.123264945818,
  "node_diskio_io_service_bytes_write": 9981.123264945818,
  "node_diskio_io_serviced_async": 1.153087253344018,
  "node_diskio_io_serviced_read": 0,
  "node_diskio_io_serviced_sync": 0.03603397666700056,
  "node_diskio_io_serviced_total": 1.1891212300110185,
  "node_diskio_io_serviced_write": 1.1891212300110185
}
```

類型: NodeNet

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
```

```
  "InstanceType": "t3.xlarge",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "Sources": [
    "cadvisor",
    "calculated"
  ],
  "Timestamp": "1567096928131",
  "Type": "NodeNet",
  "Version": "0",
  "interface": "eni972f6bfa9a0",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal"
  },
  "node_interface_network_rx_bytes": 3163.008420864309,
  "node_interface_network_rx_dropped": 0,
  "node_interface_network_rx_errors": 0,
  "node_interface_network_rx_packets": 16.575629266820258,
  "node_interface_network_total_bytes": 3518.3935157426017,
  "node_interface_network_tx_bytes": 355.385094878293,
  "node_interface_network_tx_dropped": 0,
  "node_interface_network_tx_errors": 0,
  "node_interface_network_tx_packets": 3.9997714100370625
}
```

類型：Pod

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "pod_cpu_utilization"
        },
        {
          "Unit": "Percent",
          "Name": "pod_memory_utilization"
        },
        {
          "Unit": "Bytes/Second",
          "Name": "pod_network_rx_bytes"
```

```
        },
        {
          "Unit": "Bytes/Second",
          "Name": "pod_network_tx_bytes"
        },
        {
          "Unit": "Percent",
          "Name": "pod_cpu_utilization_over_pod_limit"
        },
        {
          "Unit": "Percent",
          "Name": "pod_memory_utilization_over_pod_limit"
        }
      ],
      "Dimensions": [
        [
          "PodName",
          "Namespace",
          "ClusterName"
        ],
        [
          "Service",
          "Namespace",
          "ClusterName"
        ],
        [
          "Namespace",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    },
    {
      "Metrics": [
        {
          "Unit": "Percent",
          "Name": "pod_cpu_reserved_capacity"
        },
        {
          "Unit": "Percent",
          "Name": "pod_memory_reserved_capacity"
```

```
        }
      ],
      "Dimensions": [
        [
          "PodName",
          "Namespace",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    },
    {
      "Metrics": [
        {
          "Unit": "Count",
          "Name": "pod_number_of_container_restarts"
        }
      ],
      "Dimensions": [
        [
          "PodName",
          "Namespace",
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "Namespace": "amazon-cloudwatch",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "PodName": "cloudwatch-agent-statsd",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "cadvisor",
    "pod",
    "calculated"
  ],
  "Timestamp": "1567097351092",
```

```
  "Type": "Pod",
  "Version": "0",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal",
    "labels": {
      "app": "cloudwatch-agent-statsd",
      "pod-template-hash": "df44f855f"
    },
    "namespace_name": "amazon-cloudwatch",
    "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
    "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
    "pod_owners": [
      {
        "owner_kind": "Deployment",
        "owner_name": "cloudwatch-agent-statsd"
      }
    ],
    "service_name": "cloudwatch-agent-statsd"
  },
  "pod_cpu_limit": 200,
  "pod_cpu_request": 200,
  "pod_cpu_reserved_capacity": 5,
  "pod_cpu_usage_system": 1.4504841104992765,
  "pod_cpu_usage_total": 5.817016867430125,
  "pod_cpu_usage_user": 1.1281543081661038,
  "pod_cpu_utilization": 0.14542542168575312,
  "pod_cpu_utilization_over_pod_limit": 2.9085084337150624,
  "pod_memory_cache": 8192,
  "pod_memory_failcnt": 0,
  "pod_memory_hierarchical_pgfault": 0,
  "pod_memory_hierarchical_pgmajfault": 0,
  "pod_memory_limit": 104857600,
  "pod_memory_mapped_file": 0,
  "pod_memory_max_usage": 25268224,
  "pod_memory_pgfault": 0,
  "pod_memory_pgmajfault": 0,
  "pod_memory_request": 104857600,
  "pod_memory_reserved_capacity": 0.6307275170893897,
  "pod_memory_rss": 22777856,
  "pod_memory_swap": 0,
  "pod_memory_usage": 25141248,
  "pod_memory_utilization": 0.10988455961791709,
  "pod_memory_utilization_over_pod_limit": 17.421875,
  "pod_memory_working_set": 18268160,
```

```
  "pod_network_rx_bytes": 9880.697124714186,
  "pod_network_rx_dropped": 0,
  "pod_network_rx_errors": 0,
  "pod_network_rx_packets": 107.80005532263283,
  "pod_network_total_bytes": 10158.829201483635,
  "pod_network_tx_bytes": 278.13207676944796,
  "pod_network_tx_dropped": 0,
  "pod_network_tx_errors": 0,
  "pod_network_tx_packets": 1.146027574644318,
  "pod_number_of_container_restarts": 0,
  "pod_number_of_containers": 1,
  "pod_number_of_running_containers": 1,
  "pod_status": "Running"
}
```

類型: PodNet

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "Namespace": "amazon-cloudwatch",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "PodName": "cloudwatch-agent-statsd",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "cadvisor",
    "calculated"
  ],
  "Timestamp": "1567097351092",
  "Type": "PodNet",
  "Version": "0",
  "interface": "eth0",
  "kubernetes": {
    "host": "ip-192-168-75-26.us-west-2.compute.internal",
    "labels": {
      "app": "cloudwatch-agent-statsd",
      "pod-template-hash": "df44f855f"
    },
    "namespace_name": "amazon-cloudwatch",
    "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
```

```
    "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
    "pod_owners": [
      {
        "owner_kind": "Deployment",
        "owner_name": "cloudwatch-agent-statsd"
      }
    ],
    "service_name": "cloudwatch-agent-statsd"
  },
  "pod_interface_network_rx_bytes": 9880.697124714186,
  "pod_interface_network_rx_dropped": 0,
  "pod_interface_network_rx_errors": 0,
  "pod_interface_network_rx_packets": 107.80005532263283,
  "pod_interface_network_total_bytes": 10158.829201483635,
  "pod_interface_network_tx_bytes": 278.13207676944796,
  "pod_interface_network_tx_dropped": 0,
  "pod_interface_network_tx_errors": 0,
  "pod_interface_network_tx_packets": 1.146027574644318
}
```

類型：容器

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-NodeGroup-
sample",
  "ClusterName": "myCICluster",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "Namespace": "amazon-cloudwatch",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "PodName": "cloudwatch-agent-statsd",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "cadvisor",
    "pod",
    "calculated"
  ],
  "Timestamp": "1567097399912",
  "Type": "Container",
  "Version": "0",
  "container_cpu_limit": 200,
  "container_cpu_request": 200,
  "container_cpu_usage_system": 1.87958283771964,
```

```
  "container_cpu_usage_total": 6.159993652997942,
  "container_cpu_usage_user": 1.6707403001952357,
  "container_cpu_utilization": 0.15399984132494854,
  "container_memory_cache": 8192,
  "container_memory_failcnt": 0,
  "container_memory_hierarchical_pgfault": 0,
  "container_memory_hierarchical_pgmajfault": 0,
  "container_memory_limit": 104857600,
  "container_memory_mapped_file": 0,
  "container_memory_max_usage": 24580096,
  "container_memory_pgfault": 0,
  "container_memory_pgmajfault": 0,
  "container_memory_request": 104857600,
  "container_memory_rss": 22736896,
  "container_memory_swap": 0,
  "container_memory_usage": 24453120,
  "container_memory_utilization": 0.10574541028701798,
  "container_memory_working_set": 17580032,
  "container_status": "Running",
  "kubernetes": {
    "container_name": "cloudwatch-agent",
    "docker": {
      "container_id":
 "8967b6b37da239dfad197c9fdea3e5dfd35a8a759ec86e2e4c3f7b401e232706"
    },
    "host": "ip-192-168-75-26.us-west-2.compute.internal",
    "labels": {
      "app": "cloudwatch-agent-statsd",
      "pod-template-hash": "df44f855f"
    },
    "namespace_name": "amazon-cloudwatch",
    "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
    "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
    "pod_owners": [
      {
        "owner_kind": "Deployment",
        "owner_name": "cloudwatch-agent-statsd"
      }
    ],
    "service_name": "cloudwatch-agent-statsd"
  },
  "number_of_container_restarts": 0
}
```

類型：ContainerFS

```
{
  "AutoScalingGroupName": "eksctl-myCICluster-nodegroup-standard-workers-
NodeGroup-1174PV2WHZAYU",
  "ClusterName": "myCICluster",
  "EBSVolumeId": "aws://us-west-2b/vol-0a53108976d4a2fda",
  "InstanceId": "i-1234567890123456",
  "InstanceType": "t3.xlarge",
  "Namespace": "amazon-cloudwatch",
  "NodeName": "ip-192-0-2-0.us-west-2.compute.internal",
  "PodName": "cloudwatch-agent-statsd",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "cadvisor",
    "calculated"
  ],
  "Timestamp": "1567097399912",
  "Type": "ContainerFS",
  "Version": "0",

  "device": "/dev/nvme0n1p1",
  "fstype": "vfs",
  "kubernetes": {
    "container_name": "cloudwatch-agent",
    "docker": {
      "container_id":
 "8967b6b37da239dfad197c9fdea3e5dfd35a8a759ec86e2e4c3f7b401e232706"
    },
    "host": "ip-192-168-75-26.us-west-2.compute.internal",
    "labels": {
      "app": "cloudwatch-agent-statsd",
      "pod-template-hash": "df44f855f"
    },
    "namespace_name": "amazon-cloudwatch",
    "pod_id": "2f4ff5ac-c813-11e9-a31d-06e9dde32928",
    "pod_name": "cloudwatch-agent-statsd-df44f855f-ts4q2",
    "pod_owners": [
      {
        "owner_kind": "Deployment",
        "owner_name": "cloudwatch-agent-statsd"
      }
    ],
    "service_name": "cloudwatch-agent-statsd"
```

```
  }
}
```

類型：叢集

```
{
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Count",
          "Name": "cluster_node_count"
        },
        {
          "Unit": "Count",
          "Name": "cluster_failed_node_count"
        }
      ],
      "Dimensions": [
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "Sources": [
    "apiserver"
  ],
  "Timestamp": "1567097534160",
  "Type": "Cluster",
  "Version": "0",
  "cluster_failed_node_count": 0,
  "cluster_node_count": 3
}
```

類型: ClusterService

```
{
  "CloudWatchMetrics": [
    {
      "Metrics": [
```

```
        {
          "Unit": "Count",
          "Name": "service_number_of_running_pods"
        }
      ],
      "Dimensions": [
        [
          "Service",
          "Namespace",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "Namespace": "amazon-cloudwatch",
  "Service": "cloudwatch-agent-statsd",
  "Sources": [
    "apiserver"
  ],
  "Timestamp": "1567097534160",
  "Type": "ClusterService",
  "Version": "0",
  "kubernetes": {
    "namespace_name": "amazon-cloudwatch",
    "service_name": "cloudwatch-agent-statsd"
  },
  "service_number_of_running_pods": 1
}
```

類型: ClusterNamespace

```
{
  "CloudWatchMetrics": [
    {
      "Metrics": [
        {
          "Unit": "Count",
          "Name": "namespace_number_of_running_pods"
```

```
        }
      ],
      "Dimensions": [
        [
          "Namespace",
          "ClusterName"
        ],
        [
          "ClusterName"
        ]
      ],
      "Namespace": "ContainerInsights"
    }
  ],
  "ClusterName": "myCICluster",
  "Namespace": "amazon-cloudwatch",
  "Sources": [
    "apiserver"
  ],
  "Timestamp": "1567097594160",
  "Type": "ClusterNamespace",
  "Version": "0",
  "kubernetes": {
    "namespace_name": "amazon-cloudwatch"
  },
  "namespace_number_of_running_pods": 7
}
```

Amazon EKS 和 Kubernetes 效能日誌事件中的相關欄位

對於 Amazon EKS 和 Kubernetes，容器化 CloudWatch 代理程式會將資料作為效能日誌事件發出。這可讓您擷 CloudWatch 取和儲存高基數資料。 CloudWatch 使用效能記錄事件中的資料在叢集、節點和網繭層級建立彙總 CloudWatch 指標，而不需要遺失精細的詳細資料。

下表列出在這些效能日誌事件中與容器洞見指標資料集合相關的欄位。您可以使用 CloudWatch 日誌深入解析查詢上述任何欄位，以收集資料或調查問題。如需詳細資訊，請參閱使用日誌深入分析分析 CloudWatch 記錄資料。

Type	日誌欄位	來源	公式或備註
Pod	pod_cpu_utilization	計算	公式：pod_cpu_u

Type	日誌欄位	來源	公式或備註
			sage_total / node_cpu_limit
Pod	pod_cpu_usage_total pod_cpu_usage_total 的報告單位為 millicore。	cadvisor	
Pod	pod_cpu_limit	計算	公式：sum(container_cpu_limit) sum(container_cpu_limit) 包含已完成的 Pod。 如果 pod 中的任何容器未定義 CPU 限制，這個欄位就不會在日誌事件中顯示。此項包含初始化容器。

Type	日誌欄位	來源	公式或備註
Pod	pod_cpu_request	計算	公式：sum(container_cpu_request) 不保證要設定 container_cpu_request 。只有已設定的項目會包含在總和中。
Pod	pod_cpu_utilization_over_pod_limit	計算	公式：pod_cpu_usage_total / pod_cpu_limit
Pod	pod_cpu_reserved_capacity	計算	公式：pod_cpu_request / node_cpu_limit

Type	日誌欄位	來源	公式或備註
Pod	pod_memory_utilization	計算	公式：pod_memory_working_set / node_memory_limit 它是 pod 記憶體使用量佔節點記憶體限制的百分比。
Pod	pod_memory_working_set	cadvisor	
Pod	pod_memory_limit	計算	公式：sum(container_memory_limit) 如果 pod 中的任何容器未定義記憶體限制，這個欄位就不會在日誌事件中顯示。此項包含初始化容器。

Type	日誌欄位	來源	公式或備註
Pod	pod_memory_request	計算	公式：sum(container_memory_request) 不保證要設定 container_memory_request 。只有已設定的項目會包含在總和中。
Pod	pod_memory_utilization_over_pod_limit	計算	公式：pod_memory_working_set / pod_memory_limit 如果 pod 中的任何容器未定義記憶體限制，這個欄位就不會在日誌事件中顯示。此項包含初始化容器。
Pod	pod_memory_reserved_capacity	計算	公式：pod_memory_request / node_memory_limit

Type	日誌欄位	來源	公式或備註
Pod	pod_network_tx_bytes	計算	公式：sum(pod_interface_network_tx_bytes) 此資料可用於每個 pod 的所有網路介面。CloudWatch 代理程式會計算總計，並新增測量結果擷取規則。
Pod	pod_network_rx_bytes	計算	公式：sum(pod_interface_network_rx_bytes)
Pod	pod_network_total_bytes	計算	公式：pod_network_rx_bytes + pod_network_tx_bytes
PodNet	pod_interface_network_rx_bytes	cadvisor	此資料是 pod 網路介面的網路 rx 每秒位元組。

Type	日誌欄位	來源	公式或備註
PodNet	pod_interface_network_tx_bytes	cadvisor	此資料是 pod 網路介面的網路 tx 每秒位元組。
容器	container_cpu_usage_total	cadvisor	
容器	container_cpu_limit	cadvisor	不保證要設定。如果未設定則不會發出。
容器	container_cpu_request	cadvisor	不保證要設定。如果未設定則不會發出。
容器	container_memory_working_set	cadvisor	
容器	container_memory_limit	Pod	不保證要設定。如果未設定則不會發出。
容器	container_memory_request	Pod	不保證要設定。如果未設定則不會發出。
節點	node_cpu_utilization	計算	公式：node_cpu_usage_total / node_cpu_limit

Type	日誌欄位	來源	公式或備註
節點	node_cpu_usage_total	cadvisor	
節點	node_cpu_limit	/proc	
節點	node_cpu_request	計算	公式：sum(pod_cpu_request) 對於 cronjobs，node_cpu_request 也包含來自已完成 Pod 的請求。這可能會導致較高的 node_cpu_reserved_capacity 值。
節點	node_cpu_reserved_capacity	計算	公式：node_cpu_request / node_cpu_limit
節點	node_memory_utilization	計算	公式：node_memory_working_set / node_memory_limit
節點	node_memory_working_set	cadvisor	

Type	日誌欄位	來源	公式或備註
節點	node_memory_limit	/proc	
節點	node_memory_request	計算	公式：sum(pod_memory_request)
節點	node_memory_reserved_capacity	計算	公式：node_memory_request / node_memory_limit
節點	node_network_rx_bytes	計算	公式：sum(node_interface_network_rx_bytes)
節點	node_network_tx_bytes	計算	公式：sum(node_interface_network_tx_bytes)
節點	node_network_total_bytes	計算	公式：node_network_rx_bytes + node_network_tx_bytes
節點	node_number_of_running_pods	Pod 清單	

Type	日誌欄位	來源	公式或備註
節點	node_number_of_running_containers	Pod 清單	
NodeNet	node_interface_network_rx_bytes	cadvisor	此資料是工作者節點網路介面的網路 rx 每秒位元組。
NodeNet	node_interface_network_tx_bytes	cadvisor	此資料是工作者節點網路介面的網路 tx 每秒位元組。
NodeFS	node_filesystem_capacity	cadvisor	
NodeFS	node_filesystem_usage	cadvisor	
NodeFS	node_filesystem_utilization	計算	公式：node_filesystem_usage / node_filesystem_capacity 可根據裝置名稱提供此資料。
叢集	cluster_failed_node_count	API 伺服器	
叢集	cluster_node_count	API 伺服器	
服務	service_number_of_running_pods	API 伺服器	

Type	日誌欄位	來源	公式或備註
Namespace	namespace_number_of_running_pods	API 伺服器	

指標計算範例

本區段包含的範例會說明先前資料表中部分值的計算方式。

假設您有一個處於以下狀態的叢集。

```
Node1
   node_cpu_limit = 4
   node_cpu_usage_total = 3

   Pod1
     pod_cpu_usage_total = 2

     Container1
        container_cpu_limit = 1
        container_cpu_request = 1
        container_cpu_usage_total = 0.8

     Container2
        container_cpu_limit = null
        container_cpu_request = null
        container_cpu_usage_total = 1.2

   Pod2
     pod_cpu_usage_total = 0.4

     Container3
        container_cpu_limit = 1
        container_cpu_request = 0.5
        container_cpu_usage_total = 0.4

Node2
   node_cpu_limit = 8
   node_cpu_usage_total = 1.5

   Pod3
     pod_cpu_usage_total = 1
```

```
    Container4
       container_cpu_limit = 2
       container_cpu_request = 2
       container_cpu_usage_total = 1
```

下表顯示如何使用此資料計算 pod CPU 指標。

指標	公式	Pod1	Pod2	Pod3
pod_cpu_utilization	pod_cpu_usage_total / node_cpu_limit	2 / 4 = 50%	0.4 / 4 = 10%	1 / 8 = 12.5%
pod_cpu_utilization_over_pod_limit	pod_cpu_usage_total / sum(container_cpu_limit)	不適用，因為未定義 Container 2 的 CPU 限制	0.4 / 1 = 40%	1 / 2 = 50%
pod_cpu_reserved_capacity	sum(container_cpu_request) / node_cpu_limit	(1 + 0) / 4 = 25%	0.5 / 4 = 12.5%	2 / 8 = 25%

下表顯示如何使用此資料計算節點 CPU 指標。

指標	公式	Node1	Node2
node_cpu_utilization	node_cpu_usage_total / node_cpu_limit	3 / 4 = 75%	1.5 / 8 = 18.75%
node_cpu_reserved_capacity	sum(pod_cpu_request) / node_cpu_limit	1.5 / 4 = 37.5%	2 / 8 = 25%

Container Insights Prometheus 指標監控

CloudWatch 適用於 Prometheus 的容器洞察監控，可自動從容器化系統和工作負載中探索 Prometheus 指標。Prometheus 是一種開放原始碼系統監控和警示工具組。如需詳細資訊，請參閱 Prometheus 文件中的什麼是 Prometheus？。

探索 Prometheus 指標可支援在 Amazon EC2 執行個體上執行的 Amazon Elastic Container Service、Amazon Elastic Kubernetes Service 和 Kubernetes 叢集。將會收集 Prometheus 計數器、量測計和摘要指標類型。計劃會在後續的版本支援長條圖指標。

對於 Amazon ECS 和 Amazon EKS 叢集，支援 EC2 和 Fargate 啟動類型。Container Insights 會自動從數個工作負載收集指標，而您可以將其設定為從任何工作負載收集指標。

您可以採用 Prometheus 作為開放原始碼和開放標準的方法來擷取自訂指標。 CloudWatch支援 Prometheus 的 CloudWatch 代理程式會探索並收集 Prometheus 指標，以便更快速地監控、疑難排解和警示應用程式效能降低和故障。這也會減少所需的監控工具數目以改善可觀察性。

容器洞察 Prometheus 支援涉及 pay-per-use 指標和日誌，包括收集、儲存和分析。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

適用於某些工作負載的預先建置儀表板

Container Insights Prometheus 解決方案包含適用於本節所列熱門工作負載的預先建置儀表板。如需這些工作負載的範例組態，請參閱 (選用) 設定範例容器化 Amazon ECS 工作負載，以進行 Prometheus 指標測試 和 (選用) 設定範例容器化 Amazon EKS 工作負載範例進行 Prometheus 指標測試。

您也可以編輯代理程式組態檔案，將 Container Insights 設定為從其他容器化服務和應用程式收集 Prometheus 指標。

具有預先建置儀表板的工作負載，適用於 Amazon EC2 執行個體上執行的 Amazon EKS 叢集和 Kubernetes 叢集：

- AWS App Mesh
- NGINX
- Memcached
- Java/JMX
- HAProxy

具有預先建置儀表板的工作負載，適用於 Amazon ECS 叢集：

- AWS App Mesh
- Java/JMX
- NGINX
- NGINX Plus

在 Amazon ECS 執行個體上安裝和設定 Prometheus 指標集合

若要從 Amazon ECS 叢集收集 Prometheus 指標，您可以使用 CloudWatch 代理程式做為收集器，或使用發行版收集器 AWS 。 OpenTelemetry 有關使用 AWS 發行版 OpenTelemetry 收集器的信息，請參閱 https://aws-otel.github.io/docs/getting-started/container-insights/ecs-prometheus。

下列各節說明如何使用 CloudWatch 代理程式做為收集器來擷取 Prometheus 測量結果。您可以在執行 Amazon ECS 的叢集上安裝具有 Prometheus 監控的 CloudWatch 代理程式，並且可以選擇性地設定代理程式來抓取其他目標。這些章節也提供選擇性的教學課程，用於設定範例工作負載，以便使用 Prometheus 監控進行測試。

Amazon ECS 上的 Container Insights 支援下列 Prometheus 指標的啟動類型和網路模式組合：

Amazon ECS 啟動類型	支援的網路模式
EC2 (Linux)	橋接器、主機和 awsvpc
Fargate	awsvpc

VPC 安全群組要求

Prometheus 工作負載的安全群組輸入規則必須向 CloudWatch 代理程式開啟 Prometheus 連接埠，以便透過私有 IP 擷取 Prometheus 度量。

代理程式的安全性群組輸出規則必須允許 CloudWatch 代理程式透過 CloudWatch 私有 IP 連線至 Prometheus 工作負載的連接埠。

主題

- 在 Amazon ECS 叢集上安裝具有 Prometheus 指標集合的 CloudWatch 代理程式
- 湊集其他 Prometheus 來源並匯入這些指標
- (選用) 設定範例容器化 Amazon ECS 工作負載，以進行 Prometheus 指標測試

在 Amazon ECS 叢集上安裝具有 Prometheus 指標集合的 CloudWatch 代理程式

本節說明如何在執行 Amazon ECS 的叢集中使用 Prometheus 監控來設定 CloudWatch 代理程式。執行這項操作之後，代理程式會自動湊集和匯入該叢集中執行的下列工作負載的指標。

- AWS App Mesh
- Java/JMX

您也可以設定代理程式，以從其他 Prometheus 工作負載和來源湊集和匯入指標。

設定 IAM 角色

CloudWatch 代理程式工作定義需要兩個 IAM 角色。如果您**`CreateIAMRoles=True`**在 AWS CloudFormation 堆疊中指定讓 Container Insights 為您建立這些角色，則會以正確的權限建立角色。如果您想要自行建立或使用現有角色，則需要下列角色和許可。

- CloudWatch 代理程式 ECS 工作角色 — CloudWatch 代理程式容器使用此角色。它必須包含CloudWatchAgentServerPolicy策略和客戶管理的策略，其中包含以下唯讀權限：
  - `ec2:DescribeInstances`
  - `ecs:ListTasks`
  - `ecs:ListServices`
  - `ecs:DescribeContainerInstances`
  - `ecs:DescribeServices`
  - `ecs:DescribeTasks`
  - `ecs:DescribeTaskDefinition`
- CloudWatch 代理程式 ECS 任務執行角色 — 這是 Amazon ECS 啟動和執行容器所需的角色。確保您的任務執行角色具有附加 AmazonECS SSM ReadOnlyAccess，亞馬遜TaskExecutionRolePolicy和策略。CloudWatchAgentServerPolicy如果您要存放更敏感的資料以供 Amazon ECS 使用，請參閱指定敏感資料。

使用以下方 CloudWatch 式安裝具有 Prometheus 監視的代理程式 AWS CloudFormation

您可 AWS CloudFormation 以使用針對 Amazon ECS 叢集安裝具有 Prometheus 監控的 CloudWatch 代理程式。下列清單顯示了您將在 AWS CloudFormation 範本中使用的參數。

- ECS ClusterName — 指定目標 Amazon ECS 叢集。

- CreateIAMRoles— 指定 **True**，以為 Amazon ECS 任務角色和 Amazon ECS 任務執行角色建立新角色。指定 **False** 以重複使用現有角色。
- TaskRoleName— 如果您**True**為建立角色指定，則會指定要用於新 Amazon ECS 任務角色的名稱。如果您將 CreateIAMRoles 指定為 **False**，則會指定要作為 Amazon ECS 任務角色使用的現有角色。
- ExecutionRoleName— 如果您**True**為建立角色指定，則會指定要用於新 Amazon ECS 任務執行角色的名稱。如果您將 CreateIAMRoles 指定為 **False**，則會指定要作為 Amazon ECS 任務執行角色使用的現有角色。
- ECS NetworkMode — 如果您使用 EC2 啟動類型，請在此處指定網路模式。其必須為 **bridge** 或 **host**。
- ECS LaunchType — 指定**fargate**或**EC2**。
- SecurityGroupID — 如果NetworkMode是 ECS**awsvpc**，請在此處指定安全性群組 ID。
- 子網 ID — 如果 ECS NetworkMode 是**awsvpc**，請在此處指定子網路識別碼。

命令範例

本節包括在各種情況下使用 Prometheus 監視來安裝容器見解的範例 AWS CloudFormation 命令。

在橋接網路模式下為 Amazon ECS 叢集建立 AWS CloudFormation 堆疊

```
export AWS_PROFILE=your_aws_config_profile_eg_default
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export ECS_CLUSTER_NAME=your_ec2_ecs_cluster_name
export ECS_NETWORK_MODE=bridge
export CREATE_IAM_ROLES=True
export ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
export ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-
prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-bridge-host.yaml

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
                 ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
                 ParameterKey=ECSNetworkMode,ParameterValue=${ECS_NETWORK_MODE} \
                 ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
```

```
                ParameterKey=ExecutionRoleName,ParameterValue=
${ECS_EXECUTION_ROLE_NAME} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
    --profile ${AWS_PROFILE}
```

在主機網路模式下為 Amazon ECS 叢集建立 AWS CloudFormation 堆疊

```
export AWS_PROFILE=your_aws_config_profile_eg_default
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export ECS_CLUSTER_NAME=your_ec2_ecs_cluster_name
export ECS_NETWORK_MODE=host
export CREATE_IAM_ROLES=True
export ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
export ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name


curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-
prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-bridge-host.yaml

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
                 ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
                 ParameterKey=ECSNetworkMode,ParameterValue=${ECS_NETWORK_MODE} \
                 ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
                 ParameterKey=ExecutionRoleName,ParameterValue=
${ECS_EXECUTION_ROLE_NAME} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
    --profile ${AWS_PROFILE}
```

在 awsvpc 網路模式下為 Amazon ECS 叢集建立 AWS CloudFormation 堆疊

```
export AWS_PROFILE=your_aws_config_profile_eg_default
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export ECS_CLUSTER_NAME=your_ec2_ecs_cluster_name
export ECS_LAUNCH_TYPE=EC2
export CREATE_IAM_ROLES=True
export ECS_CLUSTER_SECURITY_GROUP=your_security_group_eg_sg-xxxxxxxxxx
export ECS_CLUSTER_SUBNET=your_subnet_eg_subnet-xxxxxxxxxx
```

```
export ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
export ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-
prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-awsvpc.yaml

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-${ECS_LAUNCH_TYPE}-awsvpc \
    --template-body file://cwagent-ecs-prometheus-metric-for-awsvpc.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
                 ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
                 ParameterKey=ECSLaunchType,ParameterValue=${ECS_LAUNCH_TYPE} \
                 ParameterKey=SecurityGroupID,ParameterValue=
${ECS_CLUSTER_SECURITY_GROUP} \
                 ParameterKey=SubnetID,ParameterValue=${ECS_CLUSTER_SUBNET} \
                 ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
                 ParameterKey=ExecutionRoleName,ParameterValue=
${ECS_EXECUTION_ROLE_NAME} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
    --profile ${AWS_PROFILE}
```

在 awsvpc 網絡模式下為 Fargate 集群創建 AWS CloudFormation 堆棧

```
export AWS_PROFILE=your_aws_config_profile_eg_default
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export ECS_CLUSTER_NAME=your_ec2_ecs_cluster_name
export ECS_LAUNCH_TYPE=FARGATE
export CREATE_IAM_ROLES=True
export ECS_CLUSTER_SECURITY_GROUP=your_security_group_eg_sg-xxxxxxxxxx
export ECS_CLUSTER_SUBNET=your_subnet_eg_subnet-xxxxxxxxxx
export ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
export ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-
prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-awsvpc.yaml

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-${ECS_LAUNCH_TYPE}-awsvpc \
    --template-body file://cwagent-ecs-prometheus-metric-for-awsvpc.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
```

```
                ParameterKey=CreateIAMRoles,ParameterValue=${CREATE_IAM_ROLES} \
                ParameterKey=ECSLaunchType,ParameterValue=${ECS_LAUNCH_TYPE} \
                ParameterKey=SecurityGroupID,ParameterValue=
${ECS_CLUSTER_SECURITY_GROUP} \
                ParameterKey=SubnetID,ParameterValue=${ECS_CLUSTER_SUBNET} \
                ParameterKey=TaskRoleName,ParameterValue=${ECS_TASK_ROLE_NAME} \
                ParameterKey=ExecutionRoleName,ParameterValue=
${ECS_EXECUTION_ROLE_NAME} \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
    --profile ${AWS_PROFILE}
```

AWS 由 AWS CloudFormation 堆棧創建的資源

下表列出當您在 Amazon ECS 叢集上使用 AWS CloudFormation Prometheus 監控設定容器洞見時所建立的 AWS 資源。

資源類型	資源名稱	說明
AWS::SSM::Parameter	AmazonCloudWatch-*CW-$ EC_ ## _ ## AgentConfig-$ EC_ ## _ ## -$ EC_ ####*	這是具有預設應用程式網格和 Java/JMX 內嵌度量格式定義的 CloudWatch 代理程式。
AWS::SSM::Parameter	AmazonCloudWatch-*PrometheusConfigName-$ ## _ ##-$ EC_ ## _ ##-$ EC_ ####*	這是 Prometheus 湊集組態。
AWS::IAM::Role	$ECS_TASK_ROLE_NAME。	Amazon ECS 任務角色。這僅在您為 CREATE_IAM_ROLES 指定 **True** 時建立。
AWS::IAM::Role	${ECS_EXECUTION_ROLE_NAME}	Amazon ECS 任務執行角色。這僅在您為 CREATE_IAM_ROLES 指定 **True** 時建立。
AWS::ECS::TaskDefinition	cwagent-prometheus-$*ECS_CLUSTER_NAME* -$*ECS_LAUNCH_TYPE* -$*ECS_NETWORK_MODE*	

資源類型	資源名稱	說明
AWS::ECS::Service	cwagent-prometheus-replica-service-*$EC_ ## _ ##-$ EC_ ####*	

使用 Prometheus 監視刪除 CloudWatch 代理程式的 AWS CloudFormation 堆疊

若要從 Amazon ECS 叢集刪除 CloudWatch 代理程式，請輸入以下命令。

```
export AWS_PROFILE=your_aws_config_profile_eg_default
export AWS_DEFAULT_REGION=your_aws_region_eg_ap-southeast-1
export CLOUDFORMATION_STACK_NAME=your_cloudformation_stack_name

aws cloudformation delete-stack \
--stack-name ${CLOUDFORMATION_STACK_NAME} \
--region ${AWS_DEFAULT_REGION} \
--profile ${AWS_PROFILE}
```

湊集其他 Prometheus 來源並匯入這些指標

具有 Prometheus 監控的 CloudWatch 代理需要兩種配置來抓取 Prometheus 指標。其中一個是 Prometheus 文件的 <scrape_config> 中記錄的標準 Prometheus 湊集組態。另一個用於代 CloudWatch 理配置。

對於 Amazon ECS 叢集，組態會透過 Amazon ECS 任務定義中的秘密與 AWS Systems Manager 的參數存放區進行整合：

- 秘密 PROMETHEUS_CONFIG_CONTENT 適用於 Prometheus 湊集組態。
- 密碼CW_CONFIG_CONTENT是用於 CloudWatch 代理程式組態。

若要抓取其他 Prometheus 指標來源並將這些指標匯入 CloudWatch，請修改 Prometheus 抓取設定和代理程式組態，然後使用更新的組態重新部署 CloudWatch 代理程式。

VPC 安全群組要求

Prometheus 工作負載的安全群組輸入規則必須向 CloudWatch 代理程式開啟 Prometheus 連接埠，以便透過私有 IP 擷取 Prometheus 度量。

代理程式的安全性群組輸出規則必須允許 CloudWatch 代理程式透過 CloudWatch 私有 IP 連線至 Prometheus 工作負載的連接埠。

Prometheus 湊集組態

該 CloudWatch 代理支持標準的 Prometheus 抓取配置，如 Prometheus 文檔https://prometheus.io/docs/prometheus/latest/configuration/configuration/#scrape_config<scrape_config>中所述。您可以編輯此區段來更新已存在於此檔案中的組態，並新增其他 Prometheus 湊集目標。根據預設，範例組態檔案包含下列全域組態行：

```
global:
  scrape_interval: 1m
  scrape_timeout: 10s
```

- scrape_interval— 定義湊集目標的頻率。
- scrape_timeout— 定義湊集請求逾時之前要等待的時間。

您也可以在任務層級為這些設定定義不同的數值，以覆寫全域設定。

Prometheus 湊集任務

CloudWatch 代理程式 YAML 檔案已設定一些預設抓取工作。例如，在 Amazon ECS 的 YAML 檔案中，例如 `cwagent-ecs-prometheus-metric-for-bridge-host.yaml`，已在 `ecs_service_discovery` 區段中設定預設湊集任務。

```
"ecs_service_discovery": {
                  "sd_frequency": "1m",
                  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
                  "docker_label": {
                  },
                  "task_definition_list": [
                    {
                      "sd_job_name": "ecs-appmesh-colors",
                      "sd_metrics_ports": "9901",
                      "sd_task_definition_arn_pattern": ".*:task-definition\/.*-
ColorTeller-(white):[0-9]+",
                      "sd_metrics_path": "/stats/prometheus"
                    },
                    {
                      "sd_job_name": "ecs-appmesh-gateway",
                      "sd_metrics_ports": "9901",
                      "sd_task_definition_arn_pattern": ".*:task-definition/.*-
ColorGateway:[0-9]+",
                      "sd_metrics_path": "/stats/prometheus"
```

```
                }
              ]
            }
```

這些預設目標 CloudWatch 中的每一個都會被抓取，並使用內嵌指標格式將指標傳送至記錄事件中。如需詳細資訊，請參閱 在日誌中內嵌指標。

來自 Amazon ECS 叢集的日誌事件會存放在 /aws/ecs/containerinsights/***cluster_name***/prometheus 日誌群組。

每個湊集任務都包含在此日誌群組的不同日誌串流中。

若要新增湊集目標，您可在 YAML 檔案的 `ecs_service_discovery` 區段下的 `task_definition_list` 區段中新增條目，然後重新啟動代理程式。如需此程序的範例，請參閱新增 Prometheus 湊集目標的教學課程：Prometheus API 伺服器指標。

CloudWatch Prometheus 的代理程式組態

CloudWatch 代理配置文件在 Prometheus 抓`metrics_collected`取配置下有一個`prometheus`部分。其包含下列組態選項：

- cluster_name— 指定要在日誌事件中新增為標籤的叢集名稱。此欄位為選用欄位。如果您省略此值，代理程式可以偵測 Amazon ECS 叢集名稱。
- log_group_name— 為湊集的 Prometheus 指標指定日誌檔案群組名稱。此欄位為選用欄位。*##### ####### Amazon ECS ##### CloudWatch ##* ***/aws/ec/ #####/####/****Prometheus#*
- prometheus_config_path— 指定 Prometheus 湊集組態檔案路徑。如果此欄位的值以 `env:` 為開頭，則將從容器的環境變數中擷取 Prometheus 湊集組態檔案內容。請不要變更此欄位。
- ecs_service_discovery— 是指定 Amazon ECS Prometheus 目標自動探索功能組態的區段。支援兩種模式來探索 Prometheus 目標：根據容器的 Docker 標籤進行探索，或根據 Amazon ECS 任務定義 ARN 規則表達式進行探索。您可以同時使用這兩種模式， CloudWatch 代理程式會根據以下項目去除探索到的目標的重複資料：{private_ip}: {連接埠}/{metrics_path}。

  `ecs_service_discovery` 區段可以包含下列欄位：

  - `sd_frequency` 是發現 Prometheus Exporters 的頻率。指定數字和單位尾碼。例如：每分鐘一次 `1m` 或每 30 秒`30s`一次。有效的單位尾碼為`ns`、`us`、`ms`、`s`、`m` 以及 `h`。

    此欄位為選用欄位。預設值為 60 秒 (1 分鐘)。
  - `sd_target_cluster` 是用於自動探索的目標 Amazon ECS 叢集名稱。此欄位為選用欄位。預設值為安裝 CloudWatch 代理程式的 Amazon ECS 叢集名稱。

- sd_cluster_region 是目標 Amazon ECS 叢集的區域。此欄位為選用欄位。預設為安裝 CloudWatch代理程式的 Amazon ECS 叢集區域。
- sd_result_file 是 Prometheus 目標結果的 YAML 檔案路徑。Prometheus 湊集組態將參與此檔案。
- docker_label 是選用區段，您可以用它來指定 Docker 標籤型服務探索的組態。如果您省略此區段，則不會使用 Docker 標籤型探索。此區段可以包含下列欄位：
  - sd_port_label 是容器的 Docker 標籤名稱，用於指定 Prometheus 指標的容器連接埠。預設值為 ECS_PROMETHEUS_EXPORTER_PORT。如果容器沒有此 docker 標籤，則 CloudWatch 代理程式會略過它。
  - sd_metrics_path_label 是容器的 Docker 標籤名稱，用於指定 Prometheus 指標路徑。預設值為 ECS_PROMETHEUS_METRICS_PATH。如果容器沒有此 Docker 標籤，則代理程式會假設預設路徑 /metrics。
  - sd_job_name_label 是容器的 Docker 標籤名稱，用於指定 Prometheus 湊集任務名稱。預設值為 job。如果容器沒有此 docker 標籤，則 CloudWatch 代理程式會使用 Prometheus 抓取設定中的作業名稱。
- task_definition_list 是選用區段，您可以用它來指定任務定義型服務探索的組態。如果您省略此區段，則不會使用任務定義型探索。此區段可以包含下列欄位：
  - sd_task_definition_arn_pattern 是用來指定要探索的 Amazon ECS 任務定義的模式。這是規則表達式。
  - sd_metrics_ports 列出了 Prometheus 指標的 containerPort。使用分號分隔 containerPorts。
  - sd_container_name_pattern 指定了 Amazon ECS 任務容器名稱。這是規則表達式。
  - sd_metrics_path 指定了 Prometheus 指標路徑。如果您省略此項，代理程式會假設預設路徑 /metrics
  - sd_job_name 指定了 Prometheus 湊集任務名稱。如果您省略此欄位， CloudWatch 代理程式會使用 Prometheus 抓取設定中的工作名稱。
- service_name_list_for_tasks 是選用區段，您可以用它來指定服務名稱型探索的組態。如果您省略此區段，則不會使用服務名稱型探索。此區段可以包含下列欄位：
  - sd_service_name_pattern 是用來指定要探索任務的 Amazon ECS 服務的模式。這是規則表達式。
  - sd_metrics_ports 列出了 Prometheus 指標的 containerPort。使用分號分隔多個 containerPorts。
  - sd_container_name_pattern 指定了 Amazon ECS 任務容器名稱。這是規則表達式。

  - sd_metrics_path 指定了 Prometheus 指標路徑。如果您省略此項，代理程式會假設預設路徑 /metrics。
  - sd_job_name 指定了 Prometheus 湊集任務名稱。如果您省略此欄位， CloudWatch 代理程式會使用 Prometheus 抓取設定中的工作名稱。
- metric_declaration— 是以要產生之內嵌指標格式來指定日誌陣列的區段。根據預設， CloudWatch 代理程式從中匯入的每個 Prometheus 來源都有metric_declaration區段。這些區段各包括下列欄位：
  - label_matcher 是一個規則表達式，會檢查 source_labels 中列出的標籤值。符合的量度會啟用以包含在傳送至的內嵌量度格式中 CloudWatch。

    如果您在 source_labels 中指定了多個標籤，我們建議您不要在 label_matcher 的規則表達式中使用 ^ 或 $ 字元。
  - source_labels 指定由 label_matcher 行檢查的標籤值。
  - label_separator 指定要在 label_matcher 行中使用的分隔符號 (如果指定多個 source_labels)。預設值為 ;。您可以在下面的範例中看到 label_matcher 行中使用此預設值。
  - metric_selectors是一個規則運算式，用來指定要收集並傳送至的測量結果 CloudWatch。
  - dimensions是要作為每個選取量度之 CloudWatch 維度使用的標籤清單。

請參閱以下 metric_declaration 範例。

```
"metric_declaration": [
  {
     "source_labels":[ "Service", "Namespace"],
     "label_matcher":"(.*node-exporter.*|.*kube-dns.*);kube-system$",
     "dimensions":[
        ["Service", "Namespace"]
     ],
     "metric_selectors":[
        "^coredns_dns_request_type_count_total$"
     ]
  }
]
```

此範例會在符合下列條件時，設定內嵌指標格式區段，以作為日誌事件傳送：

- Service 的數值包含 node-exporter 或 kube-dns。

- Namespace 的值為 kube-system。
- Prometheus 指標 coredns_dns_request_type_count_total 包含 Service 和 Namespace 標籤。

傳送的日誌事件包含下列反白顯示的區段：

```
{
   "CloudWatchMetrics":[
      {
         "Metrics":[
            {
               "Name":"coredns_dns_request_type_count_total"
            }
         ],
         "Dimensions":[
            [
               "Namespace",
               "Service"
            ]
         ],
         "Namespace":"ContainerInsights/Prometheus"
      }
   ],
   "Namespace":"kube-system",
   "Service":"kube-dns",
   "coredns_dns_request_type_count_total":2562,
   "eks_amazonaws_com_component":"kube-dns",
   "instance":"192.168.61.254:9153",
   "job":"kubernetes-service-endpoints",
   ...
}
```

在 Amazon ECS 叢集上自動探索的詳細指南

Prometheus 提供數十種動態服務探索機制，如 <scrape_config> 中所述 不過，Amazon ECS 沒有內建的服務探索。 CloudWatch 代理程式會新增此機制。

啟用 Amazon ECS Prometheus 服務探索時， CloudWatch 代理程式會定期對 Amazon ECS 和 Amazon EC2 前端進行下列 API 呼叫，以擷取目標 ECS 叢集中執行中 ECS 任務的中繼資料。

```
EC2:DescribeInstances
ECS:ListTasks
```

```
ECS:ListServices
ECS:DescribeContainerInstances
ECS:DescribeServices
ECS:DescribeTasks
ECS:DescribeTaskDefinition
```

CloudWatch 代理程式會使用中繼資料來掃描 ECS 叢集內的 Prometheus 目標。 CloudWatch 代理程式支援三種服務探索模式：

- 容器 Docker 標籤型服務探索
- ECS 任務定義 ARN 規則表達式型服務探索
- ECS 服務名稱規則表達式型服務探索

所有模式都可以一起使用。 CloudWatch 代理程式會根據以下項目來取消探索到的目標的 `{private_ip}:{port}/{metrics_path}`

所有探索到的目標都會寫入 CloudWatch 代理程式容器內`sd_result_file`組態欄位所指定的結果檔案中。以下是範例結果檔案：

```
- targets:
  - 10.6.1.95:32785
  labels:
    __metrics_path__: /metrics
    ECS_PROMETHEUS_EXPORTER_PORT: "9406"
    ECS_PROMETHEUS_JOB_NAME: demo-jar-ec2-bridge-dynamic
    ECS_PROMETHEUS_METRICS_PATH: /metrics
    InstanceType: t3.medium
    LaunchType: EC2
    SubnetId: subnet-123456789012
    TaskDefinitionFamily: demo-jar-ec2-bridge-dynamic-port
    TaskGroup: family:demo-jar-ec2-bridge-dynamic-port
    TaskRevision: "7"
    VpcId: vpc-01234567890
    container_name: demo-jar-ec2-bridge-dynamic-port
    job: demo-jar-ec2-bridge-dynamic
- targets:
  - 10.6.3.193:9404
  labels:
    __metrics_path__: /metrics
    ECS_PROMETHEUS_EXPORTER_PORT_SUBSET_B: "9404"
    ECS_PROMETHEUS_JOB_NAME: demo-tomcat-ec2-bridge-mapped-port
```

```
    ECS_PROMETHEUS_METRICS_PATH: /metrics
    InstanceType: t3.medium
    LaunchType: EC2
    SubnetId: subnet-123456789012
    TaskDefinitionFamily: demo-tomcat-ec2-bridge-mapped-port
    TaskGroup: family:demo-jar-tomcat-bridge-mapped-port
    TaskRevision: "12"
    VpcId: vpc-01234567890
    container_name: demo-tomcat-ec2-bridge-mapped-port
    job: demo-tomcat-ec2-bridge-mapped-port
```

您可以將此結果檔案與 Prometheus 檔案型服務探索直接整合。如需 Prometheus 檔案型服務探索的詳細資訊，請參閱 <file_sd_config>。

假設結果檔案寫入 `/tmp/cwagent_ecs_auto_sd.yaml`。下列 Prometheus 湊集組態將會使用它。

```
global:
  scrape_interval: 1m
  scrape_timeout: 10s
scrape_configs:
  - job_name: cwagent-ecs-file-sd-config
    sample_limit: 10000
    file_sd_configs:
      - files: [ "/tmp/cwagent_ecs_auto_sd.yaml" ]
```

CloudWatch 代理程式也會為探索到的目標新增下列其他標籤。

- `container_name`
- `TaskDefinitionFamily`
- `TaskRevision`
- `TaskGroup`
- `StartedBy`
- `LaunchType`
- `job`
- `__metrics_path__`
- Docker 標籤

當叢集具有 EC2 啟動類型時，會新增以下三個標籤。

- InstanceType
- VpcId
- SubnetId

> **Note**
>
> 不符合規則表達式的 Docker 標籤 [a-zA-Z_][a-zA-Z0-9_]* 會被篩選掉。這與 Prometheus 文件中的組態檔案的 label_name 列出的 Prometheus 慣例相符。

ECS 服務探索組態範例

本節包含示範 ECS 服務探索的範例。

範例 1

```
"ecs_service_discovery": {
  "sd_frequency": "1m",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "docker_label": {
  }
}
```

此範例會啟用 Docker 標籤型服務探索。 CloudWatch 代理程式每分鐘會查詢 ECS 工作的中繼資料一次，並將探索到的目標寫入 CloudWatch 代理程式容器內的/tmp/cwagent_ecs_auto_sd.yaml檔案中。

docker_label 區段中的 sd_port_label 的預設值為 ECS_PROMETHEUS_EXPORTER_PORT。如果 ECS 工作中有任何執行中的容器具有 ECS_PROMETHEUS_EXPORTER_PORT docker 標籤，則 CloudWatch 代理程式會使用其值container port來掃描容器的所有公開連接埠。如果有相符項目，則會使用映射的主機連接埠加上容器的私有 IP，以下列格式建構 Prometheus 匯出工具目標：private_ip:host_port。

docker_label 區段中的 sd_metrics_path_label 的預設值為 ECS_PROMETHEUS_METRICS_PATH。如果容器具有此 Docker 標籤，則其值將被用作 __metrics_path__。如果容器沒有此標籤，則會使用預設值 /metrics。

docker_label 區段中的 sd_job_name_label 的預設值為 job。如果容器具有此 Docker 標籤，其值會附加為目標的其中一個標籤，以取代 Prometheus 組態中指定的預設任務名稱。此 docker 標籤的值會用作記錄檔記錄群組中的 CloudWatch 記錄資料流名稱。

範例 2

```
"ecs_service_discovery": {
  "sd_frequency": "15s",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "docker_label": {
    "sd_port_label": "ECS_PROMETHEUS_EXPORTER_PORT_SUBSET_A",
    "sd_job_name_label": "ECS_PROMETHEUS_JOB_NAME"
  }
}
```

此範例會啟用 Docker 標籤型服務探索。 CloudWatch 代理程式會每 15 秒查詢 ECS 工作的中繼資料，並將探索到的目標寫入 CloudWatch 代理程式容器內的/tmp/cwagent_ecs_auto_sd.yaml檔案中。將掃描帶有 Docker 標籤的容器 ECS_PROMETHEUS_EXPORTER_PORT_SUBSET_A。Docker 標籤的值 ECS_PROMETHEUS_JOB_NAME 可用作任務名稱。

範例 3

```
"ecs_service_discovery": {
  "sd_frequency": "5m",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "task_definition_list": [
    {
      "sd_job_name": "java-prometheus",
      "sd_metrics_path": "/metrics",
      "sd_metrics_ports": "9404; 9406",
      "sd_task_definition_arn_pattern": ".*:task-definition/.*javajmx.*:[0-9]+"
    },
    {
      "sd_job_name": "envoy-prometheus",
      "sd_metrics_path": "/stats/prometheus",
      "sd_container_name_pattern": "^envoy$",
      "sd_metrics_ports": "9901",
      "sd_task_definition_arn_pattern": ".*:task-definition/.*appmesh.*:23"
    }
  ]
}
```

此範例會啟用 ECS 任務定義 ARN 規則表達式型服務探索。 CloudWatch 代理程式會每五分鐘查詢 ECS 工作的中繼資料，並將探索到的目標寫入 CloudWatch 代理程式容器內的/tmp/cwagent_ecs_auto_sd.yaml檔案中。

定義了兩個任務定義 ARN 規則表達式部分：

- 對於第一個區段，會篩選出其 ECS 任務定義 ARN 中具有 javajmx 的 ECS 任務，以進行容器連接埠掃描。如果這些 ECS 任務中的容器在 9404 或 9406 上公開容器連接埠，則會使用映射的主機連接埠以及容器的私有 IP 來建立 Prometheus 匯出工具目標。sd_metrics_path 的值會將 __metrics_path__ 設定為 /metrics。因此， CloudWatch 代理將從中抓取 Prometheus 指標private_ip:host_port/metrics，抓取的指標將發送到日誌組中 CloudWatch 日誌中的日誌流。java-prometheus /aws/ecs/containerinsights/cluster_name/prometheus
- 對於第一個區段，會篩選出其 ECS 任務定義 ARN 中具有 appmesh 且 :23 的 version 的 ECS 任務，以進行容器連接埠掃描。對於名稱為 envoy 且在 9901 上公開容器連接埠的容器，則會使用映射的主機連接埠以及容器的私有 IP 來建立 Prometheus 匯出工具目標。這些 ECS 任務中的值在 9404 或 9406 上公開容器連接埠，則會使用映射的主機連接埠以及容器的私有 IP 來建立 Prometheus 匯出工具目標。sd_metrics_path 的值會將 __metrics_path__ 設定為 /stats/prometheus。因此， CloudWatch代理將從中抓取 Prometheus 指標private_ip:host_port/stats/prometheus，並將抓取的指標發送到日誌組中 CloudWatch 日誌中的日誌流。envoy-prometheus /aws/ecs/containerinsights/cluster_name/prometheus

範例 4

```
"ecs_service_discovery": {
  "sd_frequency": "5m",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "service_name_list_for_tasks": [
    {
      "sd_job_name": "nginx-prometheus",
      "sd_metrics_path": "/metrics",
      "sd_metrics_ports": "9113",
      "sd_service_name_pattern": "^nginx-.*"
    },
    {
      "sd_job_name": "haproxy-prometheus",
      "sd_metrics_path": "/stats/metrics",
      "sd_container_name_pattern": "^haproxy$",
      "sd_metrics_ports": "8404",
      "sd_service_name_pattern": ".*haproxy-service.*"
    }
```

```
  ]
}
```

此範例會啟用 ECS 服務名稱規則表達式型服務探索。 CloudWatch 代理程式會每五分鐘查詢 ECS 服務的中繼資料，並將探索到的目標寫入 CloudWatch 代理程式容器內的/tmp/cwagent_ecs_auto_sd.yaml檔案中。

定義了兩個服務名稱規則表達式區段：

- 對於第一個區段，會篩選出與 ECS 服務相關聯的 ECS 任務，且其名稱符合規則表達式 ^nginx-.*，以進行容器連接埠掃描。如果這些 ECS 任務中的容器在 9113 上公開容器連接埠，則會使用映射的主機連接埠以及容器的私有 IP 來建立 Prometheus 匯出工具目標。sd_metrics_path 的值會將 __metrics_path__ 設定為 /metrics。因此， CloudWatch 代理將從中抓取 Prometheus 指標private_ip:host_port/metrics，並將抓取的指標發送到日誌組中 CloudWatch 日誌中的日誌流。nginx-prometheus /aws/ecs/containerinsights/cluster_name/prometheus
- 對於第一個區段，會篩選出與 ECS 服務相關聯的 ECS 任務，且其名稱符合規則表達式 .*haproxy-service.*，以進行容器連接埠掃描。對於名稱為 haproxy 且在 8404 上公開容器連接埠的容器，則會使用映射的主機連接埠以及容器的私有 IP 來建立 Prometheus 匯出工具目標。sd_metrics_path 的值會將 __metrics_path__ 設定為 /stats/metrics。因此，CloudWatch 代理將從中抓取 Prometheus 指標private_ip:host_port/stats/metrics，並將抓取的指標發送到日誌組中 CloudWatch 日誌中的日誌流。haproxy-prometheus /aws/ecs/containerinsights/cluster_name/prometheus

範例 5

```
"ecs_service_discovery": {
  "sd_frequency": "1m30s",
  "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
  "docker_label": {
    "sd_port_label": "MY_PROMETHEUS_EXPORTER_PORT_LABEL",
    "sd_metrics_path_label": "MY_PROMETHEUS_METRICS_PATH_LABEL",
    "sd_job_name_label": "MY_PROMETHEUS_METRICS_NAME_LABEL"
  }
  "task_definition_list": [
    {
      "sd_metrics_ports": "9150",
      "sd_task_definition_arn_pattern": "*memcached.*"
    }
```

```
  ]
}
```

此範例會啟用兩種 ECS 服務探索模式。 CloudWatch 代理程式會每隔 90 秒查詢 ECS 工作的中繼資料，並將探索到的目標寫入 CloudWatch 代理程式容器內的/tmp/cwagent_ecs_auto_sd.yaml檔案中。

若為 Docker 型服務探索組態：

- 具有 Docker 標籤的 ECS 任務 MY_PROMETHEUS_EXPORTER_PORT_LABEL 將被篩選為 Prometheus 連接埠掃描。目標 Prometheus 容器連接埠由標籤 MY_PROMETHEUS_EXPORTER_PORT_LABEL 的值指定。
- Docker 標籤的值 MY_PROMETHEUS_EXPORTER_PORT_LABEL 可用於 __metrics_path__。如果容器沒有此 Docker 標籤，則會使用預設值 /metrics。
- Docker 標籤的值 MY_PROMETHEUS_EXPORTER_PORT_LABEL 可用作任務名稱。如果容器沒有此 Docker 標籤，則會使用 Prometheus 組態中定義的任務名稱。

若為 ECS 任務定義 ARN 規則表達式型服務探索組態：

- 會篩選出 ECS 任務定義 ARN 中具有 memcached 的 ECS 任務，以進行容器連接埠掃描。目標 Prometheus 容器連接埠由 sd_metrics_ports 定義為 9150。使用預設指標路徑 /metrics。使用 Prometheus 組態中定義的任務名稱。

(選用) 設定範例容器化 Amazon ECS 工作負載，以進行 Prometheus 指標測試

若要在 CloudWatch 容器深入解析中測試 Prometheus 指標支援，您可以設定下列一或多個容器化工作負載。支援 Prometheus 的 CloudWatch 代理程式會自動從這些工作負載收集指標。若要查看預設收集的指標，請參閱 代理程式收集的 Prometheus 測量結果 CloudWatch 。

主題

- 適用於 Amazon ECS 叢集的範例 App Mesh 工作負載
- 適用於 Amazon ECS 叢集的範例 Java/JMX 工作負載
- 適用於 Amazon ECS 叢集的範例 NGINX 工作負載
- 適用於 Amazon ECS 叢集的範例 NGINX Plus 工作負載
- 新增 Prometheus 湊集目標的教學：Amazon ECS 上的 Memcached
- 在 Amazon ECS Fargate 上抓取 Redis Prometheus 指標的教學課程

適用於 Amazon ECS 叢集的範例 App Mesh 工作負載

若要從 Amazon ECS 的範例 Prometheus 工作負載收集指標，您必須在叢集中執行 Container Insights。如需安裝 Container Insights 的相關資訊，請參閱 在 Amazon ECS 上設定 Container Insights。

首先，按照這個演練在 Amazon ECS 叢集上部署範例色彩應用程式。完成後，您將在連接埠 9901 上公開 App Mesh Prometheus 指標。

接下來，請依照下列步驟在安裝彩色應用 CloudWatch 程式的相同 Amazon ECS 叢集上安裝具有 Prometheus 監控的代理程式。本節中的步驟會在橋接網路模式下安裝 CloudWatch 代理程式。

下列步驟也將使用您在演練中設定的環境變數 `ENVIRONMENT_NAME`、`AWS_PROFILE` 以及 `AWS_DEFAULT_REGION`。

使用 Prometheus 監視安裝 CloudWatch 代理程式以進行測試

1. 輸入下列指令以下載 AWS CloudFormation 範本。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/
cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-
bridge-host.yaml
```

2. 輸入下列命令，設定網路模式。

```
export ECS_CLUSTER_NAME=${ENVIRONMENT_NAME}
export ECS_NETWORK_MODE=bridge
```

3. 輸入下列指令以建立 AWS CloudFormation 堆疊。

```
aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=${ECS_CLUSTER_NAME} \
                 ParameterKey=CreateIAMRoles,ParameterValue=True \
                 ParameterKey=ECSNetworkMode,ParameterValue=${ECS_NETWORK_MODE} \
                 ParameterKey=TaskRoleName,ParameterValue=CWAgent-Prometheus-
TaskRole-${ECS_CLUSTER_NAME} \
                 ParameterKey=ExecutionRoleName,ParameterValue=CWAgent-Prometheus-
ExecutionRole-${ECS_CLUSTER_NAME} \
    --capabilities CAPABILITY_NAMED_IAM \
```

```
    --region ${AWS_DEFAULT_REGION} \
    --profile ${AWS_PROFILE}
```

4. (選擇性) 建立 AWS CloudFormation 堆疊時，您會看到CREATE_COMPLETE訊息。如果您要在看到訊息之前檢查狀態，請輸入下列命令。

```
aws cloudformation describe-stacks \
--stack-name CWAgent-Prometheus-ECS-${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
--query 'Stacks[0].StackStatus' \
--region ${AWS_DEFAULT_REGION} \
--profile ${AWS_PROFILE}
```

疑難排解

演練中的步驟會使用 jq 來剖析 AWS CLI的輸出結果。如需安裝 jq 的詳細資訊，請參閱 jq。使用下列命令，將您 AWS CLI 的預設輸出格式設定為 JSON，以便 jq 可以正確進行剖析。

```
$ aws configure
```

當回應變為 `Default output format` 時，輸入 **`json`**。

使用 Prometheus 監控解除安裝 CloudWatch 代理程式

完成測試後，輸入下列命令，透過刪除 AWS CloudFormation 堆疊來解除安裝 CloudWatch代理程式。

```
aws cloudformation delete-stack \
--stack-name CWAgent-Prometheus-ECS-${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
--region ${AWS_DEFAULT_REGION} \
--profile ${AWS_PROFILE}
```

適用於 Amazon ECS 叢集的範例 Java/JMX 工作負載

JMX Exporter 是官方的 Prometheus 匯出工具，可以湊集 JMX mBeans 並將其公開為 Prometheus 指標。如需詳細資訊，請參閱 prometheus/jmx_exporter。

支援 Prometheus 的 CloudWatch 代理程式會根據 Amazon ECS 叢集中的服務探索組態抓取 Java/JMX Prometheus 指標。您可以設定 JMX Exporter 在不同的連接埠或 metrics_path 上公開指標。如果您確實變更了通訊埠或路徑，請更新 CloudWatch代理程式組態中的預設`ecs_service_discovery`區段。

若要從 Amazon ECS 的範例 Prometheus 工作負載收集指標，您必須在叢集中執行 Container Insights。如需安裝 Container Insights 的相關資訊，請參閱 在 Amazon ECS 上設定 Container Insights。

若要安裝適用於 Amazon ECS 叢集的 Java/JMX 範例工作負載

1. 請依照這些章節中的步驟建立 Docker 影像。
   - 範例：具有 Prometheus 指標的 Java Jar 應用程式 Docker 影像
   - 範例：具有 Prometheus 指標的 Apache Tomcat Docker 影像
2. 在 Amazon ECS 任務定義檔案中指定以下兩個 Docker 標籤。然後，您可以在叢集中以 Amazon ECS 服務或 Amazon ECS 任務的形式執行任務定義。
   - 將 ECS_PROMETHEUS_EXPORTER_PORT 設定為指向公開 Prometheus 指標的 containerPort。
   - 將 Java_EMF_Metrics 設定為 true。 CloudWatch 代理程式會使用此旗標，在記錄事件中產生內嵌的度量格式。

   以下是範例：

```
{
  "family": "workload-java-ec2-bridge",
  "taskRoleArn": "{{task-role-arn}}",
  "executionRoleArn": "{{execution-role-arn}}",
  "networkMode": "bridge",
  "containerDefinitions": [
    {
      "name": "tomcat-prometheus-workload-java-ec2-bridge-dynamic-port",
      "image": "your_docker_image_tag_for_tomcat_with_prometheus_metrics",
      "portMappings": [
        {
          "hostPort": 0,
          "protocol": "tcp",
          "containerPort": 9404
        }
      ],
      "dockerLabels": {
        "ECS_PROMETHEUS_EXPORTER_PORT": "9404",
        "Java_EMF_Metrics": "true"
      }
    }
  ],
```

```
  "requiresCompatibilities": [
    "EC2"  ],
  "cpu": "256",
  "memory": "512"
  }
```

AWS CloudFormation 範本中 CloudWatch 代理程式的預設設定可同時啟用 docker 標籤式服務探索和工作定義 ARN 型服務探索。若要檢視這些預設設定，請參閱 CloudWatch 代理程式 YAML 組態檔的第 65 行。將根據 Prometheus 湊集的指定容器連接埠自動探索具有 ECS_PROMETHEUS_EXPORTER_PORT 標籤的容器。

CloudWatch 代理程式的預設設定也會在相同檔案的第 112 行具有 Java/JMX 的metric_declaration設定。目標容器的所有 docker 標籤都會新增為 Prometheus 指標中的其他標籤，並傳送至記錄檔。 CloudWatch 對於帶有 Docker 標籤的 Java/JMX 容器 Java_EMF_Metrics="true"，會產生內嵌指標格式。

適用於 Amazon ECS 叢集的範例 NGINX 工作負載

NGINX Prometheus 匯出工具可以湊集和公開 NGINX 資料作為 Prometheus 指標。此範例會聯合使用匯出工具與 Amazon ECS 的 NGINX 反向代理服務。

有關 NGINX Prometheus 出口商的更多信息，請參閱 Github 上的。 nginx-prometheus-exporter有關 NGINX 反向代理的更多信息，請參閱 Github ecs-nginx-reverse-proxy上的。

支援 Prometheus 的 CloudWatch 代理程式會根據 Amazon ECS 叢集中的服務探索組態抓取 NGINX Prometheus 指標。您可以設定 NGINX Prometheus Exporter 在不同的連接埠或路徑上公開指標。如果您變更通訊埠或路徑，請更新 CloudWatch 代理程式組態檔中的ecs_service_discovery區段。

安裝適用於 Amazon ECS 叢集的 NGINX 反向代理範例工作負載

請遵循這些步驟，安裝 NGINX 反向代理範例工作負載。

建立 Docker 影像

若要為 NGINX 反向代理範例工作負載建立 Docker 影像

1. 從 NGINX 反向代理軟件庫下載以下文件夾：https://github.com/awslabs/ ecs-nginx-reverse-proxy /樹/主/反向代理/。
2. 尋找 app 目錄並從該目錄建置一個映像：

```
docker build -t web-server-app ./path-to-app-directory
```

3. 為 NGINX 建置一個自訂映像。首先，建立一個具有以下兩個檔案的目錄：

   - 一個範例 Dockerfile：

     ```
     FROM nginx
     COPY nginx.conf /etc/nginx/nginx.conf
     ```

   - 一個nginx.conf文件，從修改 https://github.com/awslabs/ ecs-nginx-reverse-proxy /樹/主/反向代理/：

     ```
     events {
       worker_connections 768;
     }

     http {
       # Nginx will handle gzip compression of responses from the app server
       gzip on;
       gzip_proxied any;
       gzip_types text/plain application/json;
       gzip_min_length 1000;

       server{
         listen 8080;
         location /stub_status {
             stub_status   on;
         }
       }

       server {
         listen 80;

         # Nginx will reject anything not matching /api
         location /api {
           # Reject requests with unsupported HTTP method
           if ($request_method !~ ^(GET|POST|HEAD|OPTIONS|PUT|DELETE)$) {
             return 405;
           }

           # Only requests matching the whitelist expectations will
           # get sent to the application server
     ```

```
        proxy_pass http://app:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_cache_bypass $http_upgrade;
    }
  }
}
```

> **Note**
>
> 必須在 nginx-prometheus-exporter 設定為從中湊集指標的相同連接埠啟用 stub_status。在我們的範例任務定義中，將 nginx-prometheus-exporter 設定為從連接埠 8080 湊集指標。

4. 從新目錄中的檔案建置映像：

```
docker build -t nginx-reverse-proxy ./path-to-your-directory
```

5. 將您的新映像上傳至映像儲存庫以供日後使用。

建立任務定義，以在 Amazon ECS 中執行 NGINX 和 Web 伺服器應用程式

接著，設定任務定義。

此任務定義可以啟用 NGINX Prometheus 指標的收集與匯出。NGINX 容器會追蹤應用程式的輸入，並將該資料公開到連接埠 8080，如 nginx.conf 中所設定。NGINX 普羅米修斯出口商集裝箱抓取這些指標，並將其發佈到端口 9113，以供使用。 CloudWatch

若要設定 NGINX 範例 Amazon ECS 工作負載的任務定義

1. 使用以下內容，建立名為任務定義 JSON 檔案。替換*your-customized-nginx-iamge*為自定義 NGINX 映像的圖像 URI，並將 *your-web-server-app-image* 替換為 Web 服務器應用程序映像的圖像 URI。

```
{
  "containerDefinitions": [
    {
      "name": "nginx",
```

```
      "image": "your-customized-nginx-image",
      "memory": 256,
      "cpu": 256,
      "essential": true,
      "portMappings": [
        {
          "containerPort": 80,
          "protocol": "tcp"
        }
      ],
      "links": [
        "app"
      ]
    },
    {
      "name": "app",
      "image": "your-web-server-app-image",
      "memory": 256,
      "cpu": 256,
      "essential": true
    },
    {
      "name": "nginx-prometheus-exporter",
      "image": "docker.io/nginx/nginx-prometheus-exporter:0.8.0",
      "memory": 256,
      "cpu": 256,
      "essential": true,
      "command": [
        "-nginx.scrape-uri",
        "http://nginx:8080/stub_status"
    ],
    "links":[
      "nginx"
    ],
      "portMappings":[
        {
          "containerPort": 9113,
          "protocol": "tcp"
        }
      ]
    }
  ],
  "networkMode": "bridge",
  "placementConstraints": [],
```

```
  "family": "nginx-sample-stack"
}
```

2. 透過輸入以下命令，註冊任務定義。

```
aws ecs register-task-definition --cli-input-json file://path-to-your-task-definition-json
```

3. 透過輸入以下命令，建立服務以執行任務：

   請務必不要變更服務名稱。我們將使用配置來運行 CloudWatch 代理服務，該配置使用啟動它們的服務的名稱模式搜索任務。例如，若要讓 CloudWatch 代理程式尋找由此命令啟動的工作，您可以指定`sd_service_name_pattern`要的值`^nginx-service$`。下一節將提供更多詳細資訊。

```
aws ecs create-service \
 --cluster your-cluster-name \
 --service-name nginx-service \
 --task-definition nginx-sample-stack:1 \
 --desired-count 1
```

配置 CloudWatch 代理程式以抓取 NGINX Prometheus 指標

最後一步是配置 CloudWatch 代理以抓取 NGINX 指標。在此範例中， CloudWatch 代理程式會透過服務名稱模式和連接埠 9113 來探索工作，匯出器會公開 NGINX 的 prometheus 度量。探索到工作和可用的指標後， CloudWatch 代理程式會開始將收集的指標張貼至記錄資料流`nginx-prometheus-exporter`。

設定 CloudWatch 代理程式以抓取 NGINX 指標

1. 輸入下列命令，以下載最新版本的必要 YAML 檔案。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-bridge-host.yaml
```

2. 使用文本編輯器打開文件，然後在部分的`value`密鑰中找到完整的 CloudWatch 代理程序合併。`resource:CWAgentConfigSSMParameter`然後，在 `ecs_service_discovery` 區段中，新增下列 `service_name_list_for_tasks` 區段。

```
"service_name_list_for_tasks": [
  {
    "sd_job_name": "nginx-prometheus-exporter",
    "sd_metrics_path": "/metrics",
    "sd_metrics_ports": "9113",
    "sd_service_name_pattern": "^nginx-service$"
   }
],
```

3. 在相同的檔案中，在 metric_declaration 區段中新增以下區段，以允許 NGINX 指標。請務必遵循現有的縮排模式。

```
{
  "source_labels": ["job"],
  "label_matcher": ".*nginx.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily", "ServiceName"]],
  "metric_selectors": [
    "^nginx_.*$"
  ]
},
```

4. 如果您尚未在此叢集中部署 CloudWatch 代理程式，請跳至步驟 8。

   如果您已使用將 CloudWatch 代理程式部署在 Amazon ECS 叢集中 AWS CloudFormation，則可以輸入下列命令來建立變更集：

```
ECS_CLUSTER_NAME=your_cluster_name
AWS_REGION=your_aws_region
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
```

```
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION \
    --change-set-name nginx-scraping-support
```

5. 請在以下位置開啟 AWS CloudFormation 主控台。 https://console.aws.amazon.com/cloudformation

6. 啟示新創建的變更集。nginx-scraping-support您應該會看到套用至 CW AgentConfig SSM 參數資源的一項變更。執行變更集，然後輸入下列命令來重新啟動 CloudWatch 代理程式工作：

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 0 \
--service cwagent-prometheus-replica-service-EC2-$ECS_NETWORK_MODE \
--region $AWS_REGION
```

7. 請等候約 10 秒鐘，然後輸入下列命令。

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service cwagent-prometheus-replica-service-EC2-$ECS_NETWORK_MODE \
--region $AWS_REGION
```

8. 如果您是第一次在叢集上使用 Prometheus 測量結果收集來安裝 CloudWatch 代理程式，請輸入下列命令。

```
ECS_CLUSTER_NAME=your_cluster_name
AWS_REGION=your_aws_region
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION
```

檢視您的 NGINX 指標和日誌

您現在可以檢視收集的 NGINX 指標。

若要檢視範例 NGINX 工作負載的指標

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在執行叢集的區域中，在左側的導覽窗格中選擇 Metrics (指標)。找到 ContainerInsights/Prometheus 命名空間以查看指標。
3. 若要查看記 CloudWatch 錄事件，請在導覽窗格中選擇 [記錄群組]。這些事件位於記錄檔資料流中的記錄群組 /aws/容器洞量/#的 _cluster_name/prometheus 中。*nginx-prometheus-exporter*

適用於 Amazon ECS 叢集的範例 NGINX Plus 工作負載

NGINX Plus 是 NGINX 的商用版本。您必須擁有授權才能使用。如需詳細資訊，請參閱 NGINX Plus

NGINX Prometheus 匯出工具可以湊集和公開 NGINX 資料作為 Prometheus 指標。此範例會聯合使用匯出工具與 Amazon ECS 的 NGINX Plus 反向代理服務。

有關 NGINX Prometheus 出口商的更多信息，請參閱 Github 上的。 nginx-prometheus-exporter有關 NGINX 反向代理的更多信息，請參閱 Github ecs-nginx-reverse-proxy上的。

支援 Prometheus 的 CloudWatch 代理程式會根據 Amazon ECS 叢集中的服務探索組態抓取 NGINX Plus Prometheus 指標。您可以設定 NGINX Prometheus Exporter 在不同的連接埠或路徑上公開指標。如果您變更通訊埠或路徑，請更新 CloudWatch 代理程式組態檔中的`ecs_service_discovery`區段。

安裝適用於 Amazon ECS 叢集的 NGINX Plus 反向代理範例工作負載

請遵循這些步驟，安裝 NGINX 反向代理範例工作負載。

建立 Docker 影像

若要為 NGINX Plus 反向代理範例工作負載建立 Docker 影像

1. 從 NGINX 反向代理軟件庫下載以下文件夾：https://github.com/awslabs/ ecs-nginx-reverse-proxy/樹/主/反向代理/。
2. 尋找 app 目錄並從該目錄建置一個映像：

```
docker build -t web-server-app ./path-to-app-directory
```

3. 為 NGINX Plus 建置一個自訂映像。在為 NGINX Plus 建置映像之前，您需要為您授權的 NGINX Plus 獲取名為 nginx-repo.key 的金鑰和 SSL 憑證 nginx-repo.crt。建立目錄，並將您的 nginx-repo.key 和 nginx-repo.crt 檔案存放於其中。

   在您剛才建立的目錄中，建立下列兩個檔案：

   - 含有下列內容的範例 Dockerfile。此 docker 檔案是從 https://docs.nginx.com/nginx/admin-guide/installing-nginx/installing-nginx-docker/#docker_plus_image 所提供的範例檔案中採用。我們做出的重要變更是我們會載入一個名為 nginx.conf 的單獨檔案，而該檔案將在下一個步驟中建立。

```
FROM debian:buster-slim

LABEL maintainer="NGINX Docker Maintainers <docker-maint@nginx.com>“

# Define NGINX versions for NGINX Plus and NGINX Plus modules
# Uncomment this block and the versioned nginxPackages block in the main RUN
# instruction to install a specific release
# ENV NGINX_VERSION 21
# ENV NJS_VERSION 0.3.9
# ENV PKG_RELEASE 1~buster

# Download certificate and key from the customer portal (https://cs.nginx.com
 (https://cs.nginx.com/))
# and copy to the build context
COPY nginx-repo.crt /etc/ssl/nginx/
COPY nginx-repo.key /etc/ssl/nginx/
# COPY nginx.conf /etc/ssl/nginx/nginx.conf

RUN set -x \
# Create nginx user/group first, to be consistent throughout Docker variants
&& addgroup --system --gid 101 nginx \
&& adduser --system --disabled-login --ingroup nginx --no-create-home --home /
nonexistent --gecos "nginx user" --shell /bin/false --uid 101 nginx \
&& apt-get update \
&& apt-get install --no-install-recommends --no-install-suggests -y ca-
certificates gnupg1 \
&& \
NGINX_GPGKEY=573BFD6B3D8FBC641079A6ABABF5BD827BD9BF62; \
```

```
found=''; \
for server in \
ha.pool.sks-keyservers.net (http://ha.pool.sks-keyservers.net/) \
hkp://keyserver.ubuntu.com:80 \
hkp://p80.pool.sks-keyservers.net:80 \
pgp.mit.edu (http://pgp.mit.edu/) \
; do \
echo "Fetching GPG key $NGINX_GPGKEY from $server"; \
apt-key adv --keyserver "$server" --keyserver-options timeout=10 --recv-keys
 "$NGINX_GPGKEY" && found=yes && break; \
done; \
test -z "$found" && echo >&2 "error: failed to fetch GPG key $NGINX_GPGKEY" &&
 exit 1; \
apt-get remove --purge --auto-remove -y gnupg1 && rm -rf /var/lib/apt/lists/* \
# Install the latest release of NGINX Plus and/or NGINX Plus modules
# Uncomment individual modules if necessary
# Use versioned packages over defaults to specify a release
&& nginxPackages=" \
nginx-plus \
# nginx-plus=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-xslt \
# nginx-plus-module-xslt=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-geoip \
# nginx-plus-module-geoip=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-image-filter \
# nginx-plus-module-image-filter=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-perl \
# nginx-plus-module-perl=${NGINX_VERSION}-${PKG_RELEASE} \
# nginx-plus-module-njs \
# nginx-plus-module-njs=${NGINX_VERSION}+${NJS_VERSION}-${PKG_RELEASE} \
" \
&& echo "Acquire::https::plus-pkgs.nginx.com::Verify-Peer \"true\";" >> /etc/apt/
apt.conf.d/90nginx \
&& echo "Acquire::https::plus-pkgs.nginx.com::Verify-Host \"true\";" >> /etc/apt/
apt.conf.d/90nginx \
&& echo "Acquire::https::plus-pkgs.nginx.com::SslCert \"/etc/ssl/nginx/nginx-
repo.crt\";" >> /etc/apt/apt.conf.d/90nginx \
&& echo "Acquire::https::plus-pkgs.nginx.com::SslKey \"/etc/ssl/nginx/nginx-
repo.key\";" >> /etc/apt/apt.conf.d/90nginx \
&& printf "deb https://plus-pkgs.nginx.com/debian buster nginx-plus\n" > /etc/
apt/sources.list.d/nginx-plus.list \
&& apt-get update \
&& apt-get install --no-install-recommends --no-install-suggests -y \
$nginxPackages \
```

```
gettext-base \
curl \
&& apt-get remove --purge --auto-remove -y && rm -rf /var/lib/apt/lists/* /etc/
apt/sources.list.d/nginx-plus.list \
&& rm -rf /etc/apt/apt.conf.d/90nginx /etc/ssl/nginx

# Forward request logs to Docker log collector
RUN ln -sf /dev/stdout /var/log/nginx/access.log \
&& ln -sf /dev/stderr /var/log/nginx/error.log

COPY nginx.conf /etc/nginx/nginx.conf

EXPOSE 80

STOPSIGNAL SIGTERM

CMD ["nginx", "-g", "daemon off;"]
```

- 一個nginx.conf文件，從修改 https://github.com/awslabs/ ecs-nginx-reverse-proxy /樹/主/反向代理/ nginx。

```
events {
  worker_connections 768;
}

http {
  # Nginx will handle gzip compression of responses from the app server
  gzip on;
  gzip_proxied any;
  gzip_types text/plain application/json;
  gzip_min_length 1000;

  upstream backend {
    zone name 10m;
    server app:3000    weight=2;
    server app2:3000    weight=1;
  }

  server{
    listen 8080;
    location /api {
      api write=on;
    }
```

```
  }

  match server_ok {
    status 100-599;
  }

  server {
    listen 80;
    status_zone zone;
    # Nginx will reject anything not matching /api
    location /api {
      # Reject requests with unsupported HTTP method
      if ($request_method !~ ^(GET|POST|HEAD|OPTIONS|PUT|DELETE)$) {
        return 405;
      }

      # Only requests matching the whitelist expectations will
      # get sent to the application server
      proxy_pass http://backend;
      health_check uri=/lorem-ipsum match=server_ok;
      proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection 'upgrade';
      proxy_set_header Host $host;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_cache_bypass $http_upgrade;
    }
  }
}
```

4. 從新目錄中的檔案建置映像：

```
docker build -t nginx-plus-reverse-proxy ./path-to-your-directory
```

5. 將您的新映像上傳至映像儲存庫以供日後使用。

建立任務定義，以在 Amazon ECS 中執行 NGINX Plus 和 Web 伺服器應用程式

接著，設定任務定義。

此任務定義可以啟用 NGINX Plus Prometheus 指標的收集與匯出。NGINX 容器會追蹤應用程式的輸入，並將該資料公開到連接埠 8080，如 `nginx.conf` 中所設定。NGINX 普羅米修斯出口商集裝箱抓取這些指標，並將其發佈到端口 9113，以供使用。 CloudWatch

若要設定 NGINX 範例 Amazon ECS 工作負載的任務定義

1. 使用以下內容，建立名為任務定義 JSON 檔案。將 *your-customized-nginx-plus-image* 替換為自定義 NGINX Plus 圖像的圖像 URI，並將 *your-web-server-app-image* 替換為 Web 服務器應用程序映像的圖像 URI。

```
{
  "containerDefinitions": [
    {
      "name": "nginx",
      "image": "your-customized-nginx-plus-image",
      "memory": 256,
      "cpu": 256,
      "essential": true,
      "portMappings": [
        {
          "containerPort": 80,
          "protocol": "tcp"
        }
      ],
      "links": [
        "app",
        "app2"
      ]
    },
    {
      "name": "app",
      "image": "your-web-server-app-image",
      "memory": 256,
      "cpu": 128,
      "essential": true
    },
    {
      "name": "app2",
      "image": "your-web-server-app-image",
      "memory": 256,
      "cpu": 128,
      "essential": true
    },
    {
      "name": "nginx-prometheus-exporter",
      "image": "docker.io/nginx/nginx-prometheus-exporter:0.8.0",
      "memory": 256,
```

```
      "cpu": 256,
      "essential": true,
      "command": [
        "-nginx.plus",
        "-nginx.scrape-uri",
         "http://nginx:8080/api"
    ],
    "links":[
      "nginx"
    ],
      "portMappings":[
        {
          "containerPort": 9113,
          "protocol": "tcp"
        }
      ]
    }
  ],
  "networkMode": "bridge",
  "placementConstraints": [],
  "family": "nginx-plus-sample-stack"
}
```

2. 註冊任務定義：

```
aws ecs register-task-definition --cli-input-json file://path-to-your-task-definition-json
```

3. 透過輸入以下命令，建立服務以執行任務：

```
aws ecs create-service \
 --cluster your-cluster-name \
 --service-name nginx-plus-service \
 --task-definition nginx-plus-sample-stack:1 \
 --desired-count 1
```

請務必不要變更服務名稱。我們將使用配置來運行 CloudWatch 代理服務，該配置使用啟動它們的服務的名稱模式搜索任務。例如，若要讓 CloudWatch 代理程式尋找由此命令啟動的工作，您可以指定`sd_service_name_pattern`要的值`^nginx-plus-service$`。下一節將提供更多詳細資訊。

配置 CloudWatch 代理程序以抓取 NGINX 加 Prometheus 指標

最後一步是配置 CloudWatch 代理以抓取 NGINX 指標。在此範例中， CloudWatch 代理程式會透過服務名稱模式和連接埠 9113 來探索工作，匯出器會公開 NGINX 的 prometheus 度量。探索到工作和可用的指標後， CloudWatch 代理程式會開始將收集的指標張貼至記錄資料流nginx-prometheus-exporter。

設定 CloudWatch 代理程式以抓取 NGINX 指標

1. 輸入下列命令，以下載最新版本的必要 YAML 檔案。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-bridge-host.yaml
```

2. 使用文本編輯器打開文件，然後在部分的`value`密鑰中找到完整的 CloudWatch 代理程序合併。`resource:CWAgentConfigSSMParameter`然後，在 `ecs_service_discovery` 區段中，新增下列 `service_name_list_for_tasks` 區段。

```
"service_name_list_for_tasks": [
  {
    "sd_job_name": "nginx-plus-prometheus-exporter",
    "sd_metrics_path": "/metrics",
    "sd_metrics_ports": "9113",
    "sd_service_name_pattern": "^nginx-plus.*"
   }
],
```

3. 在相同的檔案中，在 `metric_declaration` 區段中新增以下區段，以允許 NGINX Plus 指標。請務必遵循現有的縮排模式。

```
{
  "source_labels": ["job"],
  "label_matcher": "^nginx-plus.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily", "ServiceName"]],
  "metric_selectors": [
    "^nginxplus_connections_accepted$",
    "^nginxplus_connections_active$",
    "^nginxplus_connections_dropped$",
    "^nginxplus_connections_idle$",
    "^nginxplus_http_requests_total$",
```

```
    "^nginxplus_ssl_handshakes$",
    "^nginxplus_ssl_handshakes_failed$",
    "^nginxplus_up$",
    "^nginxplus_upstream_server_health_checks_fails$"
  ]
},
{
  "source_labels": ["job"],
  "label_matcher": "^nginx-plus.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily", "ServiceName",
 "upstream"]],
  "metric_selectors": [
    "^nginxplus_upstream_server_response_time$"
  ]
},
{
  "source_labels": ["job"],
  "label_matcher": "^nginx-plus.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily", "ServiceName", "code"]],
  "metric_selectors": [
    "^nginxplus_upstream_server_responses$",
    "^nginxplus_server_zone_responses$"
  ]
},
```

4. 如果您尚未在此叢集中部署 CloudWatch 代理程式，請跳至步驟 8。

   如果您已使用將 CloudWatch 代理程式部署在 Amazon ECS 叢集中 AWS CloudFormation，則可以輸入下列命令來建立變更集：

```
ECS_CLUSTER_NAME=your_cluster_name
AWS_REGION=your_aws_region
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
```

```
                ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION \
    --change-set-name nginx-plus-scraping-support
```

5. 請在以下位置開啟 AWS CloudFormation 主控台。 https://console.aws.amazon.com/cloudformation

6. 啟示新創建的變更集。nginx-plus-scraping-support您應該會看到套用至 CW AgentConfig SSM 參數資源的一項變更。輸入下列命令，執行變更集並重新執行 CloudWatch 代理程式工作：

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 0 \
--service cwagent-prometheus-replica-service-EC2-$ECS_NETWORK_MODE \
--region $AWS_REGION
```

7. 請等候約 10 秒鐘，然後輸入下列命令。

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service cwagent-prometheus-replica-service-EC2-$ECS_NETWORK_MODE \
--region $AWS_REGION
```

8. 如果您是第一次在叢集上使用 Prometheus 測量結果收集來安裝 CloudWatch 代理程式，請輸入下列命令。

```
ECS_CLUSTER_NAME=your_cluster_name
AWS_REGION=your_aws_region
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
```

```
                  ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION
```

檢視您的 NGINX Plus 指標和日誌

您現在可以檢視收集的 NGINX Plus 指標。

若要檢視範例 NGINX 工作負載的指標

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在執行叢集的區域中，在左側的導覽窗格中選擇 Metrics (指標)。找到 ContainerInsights/Prometheus 命名空間以查看指標。
3. 若要查看記 CloudWatch 錄事件，請在導覽窗格中選擇 [記錄群組]。這些事件位於記錄檔資料流中的記錄群組 /aws/容器洞量/#的 _cluster_name/prometheus 中。*nginx-plus-prometheus-exporter*

新增 Prometheus 湊集目標的教學：Amazon ECS 上的 Memcached

本教學課程提供實作介紹，讓您在具有 EC2 啟動類型的 Amazon ECS 叢集上湊集範例 Memcached 應用程式的 Prometheus 指標。代理程式會透過 ECS 工作定義型服務探索自動探索 Memcached Prometheus 匯出 CloudWatch 程式目標。

Memcached 是一個通用的分佈式記憶體快取系統。它通常用於透過在 RAM 中快取資料和物件，來加速動態資料庫驅動的網站，進而減少必須讀取外部資料來源 (例如資料庫或 API) 的次數。如需詳細資訊，請參閱什麼是 Memcached？

memchached_exporter (Apache License 2.0) 是 Prometheus 其中一個正式匯出工具。memcache_exporter 預設會在 `/metrics.` 的連接埠 0.0.0.0:9150 上提供服務

本教學課程會使用下列兩個 Docker Hub 儲存庫中的 Docker 影像：

- Memcached
- prom/memcached-exporter

必要條件

若要從 Amazon ECS 的範例 Prometheus 工作負載收集指標，您必須在叢集中執行 Container Insights。如需安裝 Container Insights 的相關資訊，請參閱 在 Amazon ECS 上設定 Container Insights。

主題

- 設定 Amazon ECS EC2 叢集環境變數
- 安裝範例 Memcached 工作負載
- 設定 CloudWatch 代理程式以抓取記憶體快取的 Prometheus 指標
- 檢視您的 Memcached 指標

設定 Amazon ECS EC2 叢集環境變數

如要設定 Amazon ECS EC2 叢集環境變數

1. 安裝 Amazon ECS CLI (如果您尚未安裝)。如需詳細資訊，請參閱安裝 Amazon ECS CLI。
2. 設定新的 Amazon ECS 叢集名稱和區域。例如：

```
ECS_CLUSTER_NAME=ecs-ec2-memcached-tutorial
AWS_DEFAULT_REGION=ca-central-1
```

3. (選擇性) 如果您還沒有要在其中安裝 Memcached 工作負載和 CloudWatch 代理程式範例的 EC2 啟動類型的 Amazon ECS 叢集，您可以輸入以下命令來建立一個叢集。

```
ecs-cli up --capability-iam --size 1 \
--instance-type t3.medium \
--cluster $ECS_CLUSTER_NAME \
--region $AWS_REGION
```

此命令的預期結果如下所示：

```
WARN[0000] You will not be able to SSH into your EC2 instances without a key pair.
INFO[0000] Using recommended Amazon Linux 2 AMI with ECS Agent 1.44.4 and Docker
 version 19.03.6-ce
INFO[0001] Created cluster                               cluster=ecs-ec2-memcached-
tutorial region=ca-central-1
INFO[0002] Waiting for your cluster resources to be created...
INFO[0002] Cloudformation stack status
 stackStatus=CREATE_IN_PROGRESS
```

```
INFO[0063] Cloudformation stack status
 stackStatus=CREATE_IN_PROGRESS
INFO[0124] Cloudformation stack status
 stackStatus=CREATE_IN_PROGRESS
VPC created: vpc-xxxxxxxxxxxxxxxxx
Security Group created: sg-xxxxxxxxxxxxxxxxx
Subnet created: subnet-xxxxxxxxxxxxxxxxx
Subnet created: subnet-xxxxxxxxxxxxxxxxx
Cluster creation succeeded.
```

安裝範例 Memcached 工作負載

若要安裝公開 Prometheus 指標的範例 Memcached 工作負載

1. 通過輸入以下命令下載 Memcached AWS CloudFormation 模板。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/
cwagent-prometheus/sample_traffic/memcached/memcached-traffic-sample.yaml
```

2. 輸入下列指令，設定要為 Memcached 建立的 IAM 角色名稱。

```
MEMCACHED_ECS_TASK_ROLE_NAME=memcached-prometheus-demo-ecs-task-role-name
MEMCACHED_ECS_EXECUTION_ROLE_NAME=memcached-prometheus-demo-ecs-execution-role-name
```

3. 輸入下列命令，以安裝範例 Memcached 工作負載。此範例會在 `host` 網路模式中安裝工作負載。

```
MEMCACHED_ECS_NETWORK_MODE=host

aws cloudformation create-stack --stack-name Memcached-Prometheus-Demo-ECS-
$ECS_CLUSTER_NAME-EC2-$MEMCACHED_ECS_NETWORK_MODE \
    --template-body file://memcached-traffic-sample.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=ECSNetworkMode,ParameterValue=
$MEMCACHED_ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=
$MEMCACHED_ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$MEMCACHED_ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION
```

該 AWS CloudFormation 堆棧創建四個資源：

- 一個 ECS 任務角色
- 一個 ECS 任務執行角色
- 一個 Memcached 任務定義
- 一個 Memcached 服務

在 Memcached 任務定義中，定義了兩個容器：

- 主要容器執行簡易的 Memcached 應用程式，並開啟連接埠 11211 以進行存取。
- 另一個容器會執行 Redis 匯出工具程序，以公開連接埠 9150 上的 Prometheus 指標。這是 CloudWatch 代理人要發現和抓取的容器。

設定 CloudWatch 代理程式以抓取記憶體快取的 Prometheus 指標

設定 CloudWatch 代理程式以抓取記憶體快取的 Prometheus 度量

1. 輸入下列命令，以下載最新版本的 `cwagent-ecs-prometheus-metric-for-awsvpc.yaml`。

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-awsvpc.yaml
   ```

2. 使用文字編輯器開啟檔案，並在區段中尋找`value`金鑰後面的完整 CloudWatch 代理程式設`resource:CWAgentConfigSSMParameter`定。

   然後，在 `ecs_service_discovery` 區段中，將下列組態新增至 `task_definition_list` 區段。

   ```
   {
       "sd_job_name": "ecs-memcached",
       "sd_metrics_ports": "9150",
       "sd_task_definition_arn_pattern": ".*:task-definition/memcached-prometheus-demo.*:[0-9]+"
   },
   ```

對於 metric_declaration 區段中，預設設定不允許任何 Memcached 指標。新增下列區段，以允許 Memcached 指標。請務必遵循現有的縮排模式。

```
{
  "source_labels": ["container_name"],
  "label_matcher": "memcached-exporter-.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily"]],
  "metric_selectors": [
    "^memcached_current_(bytes|items|connections)$",
    "^memcached_items_(reclaimed|evicted)_total$",
    "^memcached_(written|read)_bytes_total$",
    "^memcached_limit_bytes$",
    "^memcached_commands_total$"
  ]
},
{
  "source_labels": ["container_name"],
  "label_matcher": "memcached-exporter-.*",
  "dimensions": [["ClusterName", "TaskDefinitionFamily","status","command"],
 ["ClusterName", "TaskDefinitionFamily","command"]],
  "metric_selectors": [
    "^memcached_commands_total$"
  ]
},
```

3. 如果您已經在 Amazon ECS 叢集中部署了 CloudWatch 代理程式 AWS CloudFormation，則可以輸入下列命令來建立變更集。

```
ECS_NETWORK_MODE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
```

```
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION \
    --change-set-name memcached-scraping-support
```

4. 請在以下位置開啟 AWS CloudFormation 主控台。 https://console.aws.amazon.com/cloudformation

5. 檢閱新建立的變更集 `memcached-scraping-support`。您應該會看到一個可套用至 `CWAgentConfigSSMParameter` 資源的變更。執行變更集，然後輸入下列命令來重新啟動 CloudWatch 代理程式工作。

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 0 \
--service cwagent-prometheus-replica-service-EC2-$ECS_NETWORK_MODE \
--region $AWS_REGION
```

6. 請等候約 10 秒鐘，然後輸入下列命令。

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service cwagent-prometheus-replica-service-EC2-$ECS_NETWORK_MODE \
--region $AWS_REGION
```

7. 如果您是第一次使用叢集的 Prometheus 測量結果收集來安裝 CloudWatch 代理程式，請輸入下列命令：

```
ECS_NETWORK_MODEE=bridge
CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION
```

檢視您的 Memcached 指標

本教學課程會將下列度量傳送至中的 ECS/ ContainerInsights //Prome theus 命名空間。 CloudWatch 您可以使用 CloudWatch 控制台查看該命名空間中的指標。

指標名稱	維度	
memcached_current_items	ClusterName ，TaskDefinitionFamily	
memcached_current_connections	ClusterName ，TaskDefinitionFamily	
memcached_limit_bytes	ClusterName ，TaskDefinitionFamily	
memcached_current_bytes	ClusterName ，TaskDefinitionFamily	
memcached_written_bytes_total	ClusterName ，TaskDefinitionFamily	
memcached_read_bytes_total	ClusterName ，TaskDefinitionFamily	
memcached_items_evicted_total	ClusterName ，TaskDefinitionFamily	
memcached_items_reclaimed_total	ClusterName ，TaskDefinitionFamily	
memcached_commands_total	ClusterName ，TaskDefinitionFamily ClusterName 、 TaskDefinitionFamily、指令	

指標名稱	維度	
	ClusterName 、 TaskDefinitionFamily、狀態、指令	

> **Note**
>
> command (命令) 維度的數值可以是：delete、get、cas、set、decr、touch、incr 或 flush。
> status (狀態) 維度的數值可以是 hit、miss 或 badval。

您也可以為您的 Memcached Prometheus 指標創建 CloudWatch 儀表板。

若要建立 Memcached Prometheus 指標的儀表板

1. 建立環境變數，取代下面的數值，以符合您的部署。

```
DASHBOARD_NAME=your_memcached_cw_dashboard_name
ECS_TASK_DEF_FAMILY=memcached-prometheus-demo-$ECS_CLUSTER_NAME-EC2-
$MEMCACHED_ECS_NETWORK_MOD
```

2. 輸入下列命令建立儀表板。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/ecs-task-definition-templates/deployment-mode/
replica-service/cwagent-prometheus/sample_cloudwatch_dashboards/memcached/
cw_dashboard_memcached.json \
| sed "s/{{YOUR_AWS_REGION}}/$AWS_REGION/g" \
| sed "s/{{YOUR_CLUSTER_NAME}}/$ECS_CLUSTER_NAME/g" \
| sed "s/{{YOUR_TASK_DEF_FAMILY}}/$ECS_TASK_DEF_FAMILY/g" \
| xargs -0 aws cloudwatch put-dashboard --dashboard-name ${DASHBOARD_NAME} --region
 $AWS_REGION --dashboard-body
```

在 Amazon ECS Fargate 上抓取 Redis Prometheus 指標的教學課程

本教學課程提供實作介紹，讓您在 Amazon ECS Fargate 叢集中湊集範例 Redis 應用程式的 Prometheus 指標。具有 Prometheus 指標支援的 CloudWatch 代理程式會根據容器的 docker 標籤，自動探索 Redis Prometheus 匯出程式目標。

Redis (https://redis.io/) 是一個開放原始碼 (BSD 授權)、記憶體內的資料結構存放，可用作資料庫、存取和信息，緩存和訊息中介裝置。如需詳細資訊，請參閱 redis。

redis_exporter (授權的 MIT 授權) 可用於在指定的連接埠上公開 Redis Prometheus 指標 (預設：0.0.0.0:9121)。如需詳細資訊，請參閱 redis_exporter。

本教學課程會使用下列兩個 Docker Hub 儲存庫中的 Docker 影像：

- redis
- redis_exporter

必要條件

若要從 Amazon ECS 的範例 Prometheus 工作負載收集指標，您必須在叢集中執行 Container Insights。如需安裝 Container Insights 的相關資訊，請參閱 在 Amazon ECS 上設定 Container Insights。

主題

- 設定 Amazon ECS Fargate 叢集環境變數
- 設定 Amazon ECS Fargate 叢集的網路環境變數
- 安裝範例 Redis 工作負載
- 設定 CloudWatch 代理程式以抓取 Redis Prometheus 度量
- 檢視 Redis 指標

設定 Amazon ECS Fargate 叢集環境變數

若要設定 Amazon ECS Fargate 叢集環境變數

1. 安裝 Amazon ECS CLI (如果您尚未安裝)。如需詳細資訊，請參閱安裝 Amazon ECS CLI。
2. 設定新的 Amazon ECS 叢集名稱和區域。例如：

```
ECS_CLUSTER_NAME=ecs-fargate-redis-tutorial
AWS_DEFAULT_REGION=ca-central-1
```

3. (選擇性) 如果您還沒有要在其中安裝 Redis 範例工作負載和 CloudWatch 代理程式的 Amazon ECS Fargate 叢集，您可以輸入以下命令來建立一個叢集。

```
ecs-cli up --capability-iam \
```

```
--cluster $ECS_CLUSTER_NAME \
--launch-type FARGATE \
--region $AWS_DEFAULT_REGION
```

此命令的預期結果如下所示：

```
INFO[0000] Created cluster   cluster=ecs-fargate-redis-tutorial region=ca-central-1
INFO[0001] Waiting for your cluster resources to be created...
INFO[0001] Cloudformation stack status   stackStatus=CREATE_IN_PROGRESS
VPC created: vpc-xxxxxxxxxxxxxxxxx
Subnet created: subnet-xxxxxxxxxxxxxxxx
Subnet created: subnet-xxxxxxxxxxxxxxxx
Cluster creation succeeded.
```

設定 Amazon ECS Fargate 叢集的網路環境變數

若要設定 Amazon ECS Fargate 叢集的網路環境變數

1. 設定 Amazon ECS 叢集的 VPC 和子網路 ID。如果您在先前的程序中建立了新的叢集，您會在最終命令的結果中看到這些值。否則，請使用您要與 Redis 搭配使用的現有叢集的 ID。

```
ECS_CLUSTER_VPC=vpc-xxxxxxxxxxxxxxxxx
ECS_CLUSTER_SUBNET_1=subnet-xxxxxxxxxxxxxxxx
ECS_CLUSTER_SUBNET_2=subnet-xxxxxxxxxxxxxxxx
```

2. 在本教學中，我們將在 Amazon ECS 叢集 VPC 的預設安全群組中安裝 Redis 應用程式和 CloudWatch代理程式。預設安全性群組允許相同安全性群組內的所有網路連線，以便 CloudWatch 代理程式可擷取 Redis 容器上公開的 Prometheus 指標。在實際的生產環境中，您可能想要為 Redis 應用程 CloudWatch 式和代理程式建立專用的安全性群組，並為其設定自訂權限。

   輸入以下命令以獲取預設安全群組 ID。

```
aws ec2 describe-security-groups \
--filters Name=vpc-id,Values=$ECS_CLUSTER_VPC  \
--region $AWS_DEFAULT_REGION
```

   然後輸入以下命令來設定 Fargate 叢集 deafult 安全性群組變數，並以您從上一個命令中找到的值取代*my-default-security-group*。

```
ECS_CLUSTER_SECURITY_GROUP=my-default-security-group
```

安裝範例 Redis 工作負載

若要安裝公開 Prometheus 指標的範例 Redis 工作負載

1. 輸入下列命令以下載 Redis AWS CloudFormation 範本。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/
cwagent-prometheus/sample_traffic/redis/redis-traffic-sample.yaml
```

2. 輸入下列指令，設定要為 Redis 建立的 IAM 角色名稱。

```
REDIS_ECS_TASK_ROLE_NAME=redis-prometheus-demo-ecs-task-role-name
REDIS_ECS_EXECUTION_ROLE_NAME=redis-prometheus-demo-ecs-execution-role-name
```

3. 輸入下列命令以安裝範例 Redis 工作負載。

```
aws cloudformation create-stack --stack-name Redis-Prometheus-Demo-ECS-
$ECS_CLUSTER_NAME-fargate-awsvpc \
    --template-body file://redis-traffic-sample.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=SecurityGroupID,ParameterValue=
$ECS_CLUSTER_SECURITY_GROUP \
                 ParameterKey=SubnetID,ParameterValue=$ECS_CLUSTER_SUBNET_1 \
                 ParameterKey=TaskRoleName,ParameterValue=$REDIS_ECS_TASK_ROLE_NAME
 \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$REDIS_ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_DEFAULT_REGION
```

該 AWS CloudFormation 堆棧創建四個資源：

- 一個 ECS 任務角色
- 一個 ECS 任務執行角色
- 一個 Redis 任務定義

- 一個 Redis 服務

在 Redis 的任務定義中，定義了兩個容器：

- 主要容器執行簡易的 Redis 應用程式，並開啟連接埠 6379 以進行存取。
- 另一個容器會執行 Redis 匯出工具程序，以公開連接埠 9121 上的 Prometheus 指標。這是 CloudWatch 代理人要發現和抓取的容器。已定義下列 docker 標籤，以便 CloudWatch 代理程式可以根據該容器探索此容器。

```
ECS_PROMETHEUS_EXPORTER_PORT: 9121
```

設定 CloudWatch 代理程式以抓取 Redis Prometheus 度量

設定 CloudWatch 代理程式以抓取 Redis Prometheus 度量

1. 輸入下列命令，以下載最新版本的 `cwagent-ecs-prometheus-metric-for-awsvpc.yaml`。

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/ecs-task-definition-templates/deployment-mode/replica-service/cwagent-prometheus/cloudformation-quickstart/cwagent-ecs-prometheus-metric-for-awsvpc.yaml
   ```

2. 使用文字編輯器開啟檔案，並在區段中尋找`value`金鑰後面的完整 CloudWatch 代理程式設`resource:CWAgentConfigSSMParameter`定。

   然後，在此處顯示的 `ecs_service_discovery` 區段，使用基於 `ECS_PROMETHEUS_EXPORTER_PORT` 的預設設定啟用 `docker_label` 型服務探索，而其會與我們在 Redis ECS 任務定義中定義的 Docker 標籤相符。因此，我們不需要在本節中進行任何變更：

   ```
   ecs_service_discovery": {
     "sd_frequency": "1m",
     "sd_result_file": "/tmp/cwagent_ecs_auto_sd.yaml",
   *  "docker_label": {
     },*
     ...
   ```

對於 metric_declaration 區段中，預設設定不允許任何 Redis 指標。新增下列區段，以允許 Redis 指標。請務必遵循現有的縮排模式。

```
{
  "source_labels": ["container_name"],
  "label_matcher": "^redis-exporter-.*$",
  "dimensions": [["ClusterName","TaskDefinitionFamily"]],
  "metric_selectors": [
    "^redis_net_(in|out)put_bytes_total$",
    "^redis_(expired|evicted)_keys_total$",
    "^redis_keyspace_(hits|misses)_total$",
    "^redis_memory_used_bytes$",
    "^redis_connected_clients$"
  ]
},
{
  "source_labels": ["container_name"],
  "label_matcher": "^redis-exporter-.*$",
  "dimensions": [["ClusterName","TaskDefinitionFamily","cmd"]],
  "metric_selectors": [
    "^redis_commands_total$"
  ]
},
{
  "source_labels": ["container_name"],
  "label_matcher": "^redis-exporter-.*$",
  "dimensions": [["ClusterName","TaskDefinitionFamily","db"]],
  "metric_selectors": [
    "^redis_db_keys$"
  ]
},
```

3. 如果您已經在 Amazon ECS 叢集中部署了 CloudWatch 代理程式 AWS CloudFormation，則可以輸入下列命令來建立變更集。

```
ECS_LAUNCH_TYPE=FARGATE
CREATE_IAM_ROLES=True
ECS_CLUSTER_SUBNET=$ECS_CLUSTER_SUBNET_1
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name
```

```
aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
$ECS_CLUSTER_NAME-$ECS_LAUNCH_TYPE-awsvpc \
    --template-body file://cwagent-ecs-prometheus-metric-for-awsvpc.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSLaunchType,ParameterValue=$ECS_LAUNCH_TYPE \
                 ParameterKey=SecurityGroupID,ParameterValue=
$ECS_CLUSTER_SECURITY_GROUP \
                 ParameterKey=SubnetID,ParameterValue=$ECS_CLUSTER_SUBNET \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION} \
    --change-set-name redis-scraping-support
```

4. 請在以下位置開啟 AWS CloudFormation 主控台。 https://console.aws.amazon.com/cloudformation

5. 檢閱新建立的變更集 `redis-scraping-support`。您應該會看到一個可套用至 `CWAgentConfigSSMParameter` 資源的變更。執行變更集，然後輸入下列命令來重新啟動 CloudWatch 代理程式工作。

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 0 \
--service cwagent-prometheus-replica-service-$ECS_LAUNCH_TYPE-awsvpc \
--region ${AWS_DEFAULT_REGION}
```

6. 請等候約 10 秒鐘，然後輸入下列命令。

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service cwagent-prometheus-replica-service-$ECS_LAUNCH_TYPE-awsvpc \
--region ${AWS_DEFAULT_REGION}
```

7. 如果您是第一次使用叢集的 Prometheus 測量結果收集來安裝 CloudWatch 代理程式，請輸入下列命令：

```
ECS_LAUNCH_TYPE=FARGATE
CREATE_IAM_ROLES=True
ECS_CLUSTER_SUBNET=$ECS_CLUSTER_SUBNET_1
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name
```

```
aws cloudformation create-stack --stack-name CWAgent-Prometheus-ECS-
$ECS_CLUSTER_NAME-$ECS_LAUNCH_TYPE-awsvpc \
    --template-body file://cwagent-ecs-prometheus-metric-for-awsvpc.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSLaunchType,ParameterValue=$ECS_LAUNCH_TYPE \
                 ParameterKey=SecurityGroupID,ParameterValue=
$ECS_CLUSTER_SECURITY_GROUP \
                 ParameterKey=SubnetID,ParameterValue=$ECS_CLUSTER_SUBNET \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
                 ParameterKey=ExecutionRoleName,ParameterValue=
$ECS_EXECUTION_ROLE_NAME \
    --capabilities CAPABILITY_NAMED_IAM \
    --region ${AWS_DEFAULT_REGION}
```

檢視 Redis 指標

本教學課程會將下列度量傳送至中的 ECS/ ContainerInsights //Prome theus 命名空間。 CloudWatch 您可以使用 CloudWatch 控制台查看該命名空間中的指標。

指標名稱	維度	
redis_net_input_bytes_total	ClusterName, TaskDefinitionFamily	
redis_net_output_bytes_total	ClusterName, TaskDefinitionFamily	
redis_expired_keys_total	ClusterName, TaskDefinitionFamily	
redis_evicted_keys_total	ClusterName, TaskDefinitionFamily	

指標名稱	維度	
redis_keyspace_hits_total	ClusterName, TaskDefinitionFamily	
redis_keyspace_misses_total	ClusterName, TaskDefinitionFamily	
redis_memory_used_bytes	ClusterName, TaskDefinitionFamily	
redis_connected_clients	ClusterName, TaskDefinitionFamily	
redis_commands_total	ClusterName ，TaskDefinitionFamily ，cmd	
redis_db_keys	ClusterName ，TaskDefinitionFamily ，db	

> **Note**
>
> cmd 維度的數值可以是：append、client、command、config、dbsize、flushall、get、incr、info latency 或 slowlog。
> db 維度的數值可以是 db0 至 db15。

您也可以為 Redis Prometheus 度量建立 CloudWatch 儀表板。

若要建立 Redis Prometheus 指標的儀表板

1. 建立環境變數，取代下面的數值，以符合您的部署。

```
DASHBOARD_NAME=your_cw_dashboard_name
ECS_TASK_DEF_FAMILY=redis-prometheus-demo-$ECS_CLUSTER_NAME-fargate-awsvpc
```

2. 輸入下列命令建立儀表板。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/sample_cloudwatch_dashboards/redis/cw_dashboard_redis.json \
| sed "s/{{YOUR_AWS_REGION}}/${REGION_NAME}/g" \
| sed "s/{{YOUR_CLUSTER_NAME}}/${CLUSTER_NAME}/g" \
| sed "s/{{YOUR_NAMESPACE}}/${NAMESPACE}/g" \
```

在 Amazon EKS 和 Kubernetes 執行個體上安裝和設定 Prometheus 指標集合

若要從執行 Amazon EKS 或 Kubernetes 的叢集收集 Prometheus 指標，您可以使用 CloudWatch 代理程式做為收集器，或使用發行版收集器。 AWS OpenTelemetry 有關使用 AWS 發行版 OpenTelemetry 收集器的信息，請參閱 https://aws-otel.github.io/docs/getting-started/container-insights/eks-prometheus。

下列各節說明如何使用代理程式收集 Prometheus 度量。 CloudWatch 他們說明如何在執行 Amazon EKS 或 Kubernetes 的叢集上安裝具有 Prometheus 監控的 CloudWatch 代理程式，以及如何設定代理程式以抓取其他目標。它們也提供選擇性的教學課程，用於設定範例工作負載，以便使用 Prometheus 監控進行測試。

主題

- 在 Amazon EKS 和 Kubernetes 叢集上安裝具有 Prometheus 指標集合的 CloudWatch 代理程式

在 Amazon EKS 和 Kubernetes 叢集上安裝具有 Prometheus 指標集合的 CloudWatch 代理程式

本節說明如何在執行 Amazon EKS 或 Kubernetes 的叢集中使用 Prometheus 監控來設定 CloudWatch 代理程式。執行這項操作之後，代理程式會自動湊集和匯入該叢集中執行的下列工作負載的指標。

- AWS App Mesh
- NGINX
- Memcached
- Java/JMX
- HAProxy
- Fluent Bit

您也可以設定代理程式，以湊集和匯入其他 Prometheus 工作負載和來源。

在執行這些步驟安裝 Prometheus 指標收集的 CloudWatch 代理程式之前，您必須在 Amazon EKS 上執行一個叢集，或在 Amazon EC2 執行個體上執行 Kubernetes 叢集。

VPC 安全群組要求

Prometheus 工作負載的安全群組輸入規則必須向 CloudWatch 代理程式開啟 Prometheus 連接埠，以便透過私有 IP 擷取 Prometheus 度量。

代理程式的安全性群組輸出規則必須允許 CloudWatch 代理程式透過 CloudWatch 私有 IP 連線至 Prometheus 工作負載的連接埠。

主題

- 在 Amazon EKS 和 Kubernetes 叢集上安裝具有 Prometheus 指標集合的 CloudWatch 代理程式
- 湊集其他 Prometheus 來源並匯入這些指標
- (選用) 設定範例容器化 Amazon EKS 工作負載範例進行 Prometheus 指標測試

在 Amazon EKS 和 Kubernetes 叢集上安裝具有 Prometheus 指標集合的 CloudWatch 代理程式

本節說明如何在執行 Amazon EKS 或 Kubernetes 的叢集中使用 Prometheus 監控來設定 CloudWatch 代理程式。執行這項操作之後，代理程式會自動湊集和匯入該叢集中執行的下列工作負載的指標。

- AWS App Mesh
- NGINX
- Memcached
- Java/JMX
- HAProxy
- Fluent Bit

您也可以設定代理程式，以湊集和匯入其他 Prometheus 工作負載和來源。

在執行這些步驟安裝 Prometheus 指標收集的 CloudWatch 代理程式之前，您必須在 Amazon EKS 上執行一個叢集，或在 Amazon EC2 執行個體上執行 Kubernetes 叢集。

VPC 安全群組要求

Prometheus 工作負載的安全群組輸入規則必須向 CloudWatch 代理程式開啟 Prometheus 連接埠，以便透過私有 IP 擷取 Prometheus 度量。

代理程式的安全性群組輸出規則必須允許 CloudWatch 代理程式透過 CloudWatch 私有 IP 連線至 Prometheus 工作負載的連接埠。

主題

- 設定 IAM 角色
- 安裝 CloudWatch代理程式以收集 Prometheus 測量結果

設定 IAM 角色

第一步是在叢集中設定必要的 IAM 角色。有兩種方法：

- 設定服務帳戶的 IAM 角色， 亦稱為服務角色。此方法適用於 EC2 啟動類型和 Fargate 啟動類型。
- 將 IAM 政策新增至用於叢集的 IAM 角色。這只適用於 EC2 啟動類型。

設定服務角色 (EC2 啟動類型和 Fargate 啟動類型)

若要設定服務角色，請輸入下列命令。*MyCluster*以叢集的名稱取代。

```
eksctl create iamserviceaccount \
 --name cwagent-prometheus \
--namespace amazon-cloudwatch \
 --cluster MyCluster \
--attach-policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
--approve \
--override-existing-serviceaccounts
```

將政策新增至叢集的 IAM 角色 (僅適用於 EC2 啟動類型)

若要在叢集中設定 IAM 政策以獲得 Prometheus 支援

1. 在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 在導覽窗格中，選擇執行個體。
3. 您需要找出叢集的 IAM 角色名稱的字首。若要執行此操作，請選取叢集中執行個體名稱旁的核取方塊，然後選擇 Actions (動作)、Instance Settings (執行個體設定)、Attach/Replace IAM Role (連接/取代 IAM 角色)。然後複製 IAM 角色的字首，例如 `eksctl-dev303-workshop-nodegroup`。
4. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。

5. 在導覽窗格中，選擇角色。
6. 使用搜尋方塊尋找您先前在此程序中複製的前綴，然後選擇該角色。
7. 選擇連接政策。
8. 使用搜索框進行查找CloudWatchAgentServerPolicy。選取旁邊的核取方塊 CloudWatchAgentServerPolicy，然後選擇 [附加原則]。

安裝 CloudWatch代理程式以收集 Prometheus 測量結果

您必須在叢集中安裝 CloudWatch 代理程式，才能收集指標。如何安裝代理程式會因 Amazon EKS 叢集和 Kubernetes 叢集而有所不同。

刪除具有 Prometheus 支持的 CloudWatch 代理的以前版本

如果您已在叢集中安裝具有 Prometheus 支援的 CloudWatch 代理程式版本，則必須輸入下列命令來刪除該版本。只有具有 Prometheus 支援的舊版代理程式才需要這麼做。您不需要刪除在沒有 Prometheus 支援的情況下啟用容器洞見的 CloudWatch 代理程式。

```
kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch
```

使用 EC2 啟動類型在 Amazon EKS 叢集上安裝 CloudWatch 代理程式

若要在 Amazon EKS 叢集上安裝具有 Prometheus 支援的 CloudWatch 代理程式，請依照下列步驟執行。

在 Amazon EKS 叢集上安裝具有 Prometheus 支援的 CloudWatch 代理程式

1. 輸入下列命令以檢查是否已建立 `amazon-cloudwatch` 命名空間：

   ```
   kubectl get namespace
   ```

2. 如果結果中未顯示 `amazon-cloudwatch`，請輸入下列命令來建立它：

   ```
   kubectl create namespace amazon-cloudwatch
   ```

3. 若要使用預設組態部署代理程式，並讓它將資料傳送至其安裝的 AWS 區域，請輸入下列命令：

   ```
   kubectl apply -f https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
   container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/
   service/cwagent-prometheus/prometheus-eks.yaml
   ```

如果要讓代理程式將資料傳送到不同的區域，請依照下列步驟執行：

a. 輸入下列命令，以下載代理程式的 YAML 檔案：

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-
container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/
service/cwagent-prometheus/prometheus-eks.yaml
```

b. 使用文字編輯器開啟檔案，然後搜尋檔案的 `cwagentconfig.json` 區塊。

c. 新增反白的行，指定您想要的區域：

```
cwagentconfig.json: |
    {
      "agent": {
        "region": "us-east-2"
      },
      "logs": { ...
```

d. 儲存檔案並使用您更新的檔案來部署代理程式。

```
kubectl apply -f prometheus-eks.yaml
```

使用 Fargate 啟動類型在 Amazon EKS 叢集上安裝 CloudWatch 代理程式

若要在具有 Fargate 啟動類型的 Amazon EKS 叢集上安裝具有 Prometheus 支援的 CloudWatch 代理程式，請遵循下列步驟。

在具有 Fargate 啟動類型的 Amazon EKS 叢集上安裝具有 Prometheus 支援的 CloudWatch 代理程式

1. 輸入下列命令，為 CloudWatch 代理程式建立 Fargate 設定檔，以便它可以在叢集內執行。*MyCluster*以叢集的名稱取代。

```
eksctl create fargateprofile --cluster MyCluster \
--name amazon-cloudwatch \
--namespace amazon-cloudwatch
```

2. 若要安裝 CloudWatch 代理程式，請輸入下列命令。*MyCluster*以叢集的名稱取代。此名稱會用於存放代理程式所收集之日誌事件的日誌群組名稱中，也用作為代理程式所收集之指標的維度。

   將 *region* 取代為您想要傳送指標的區域名稱。例如 `us-west-1`。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/prometheus-eks-fargate.yaml |
sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" |
kubectl apply -f -
```

在 Kubernetes 叢集上安裝 CloudWatch 代理程式

若要在執行 Kubernetes 的叢集上安裝具有 Prometheus 支援的 CloudWatch 代理程式，請輸入下列命令：

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/prometheus-k8s.yaml |
sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" |
kubectl apply -f -
```

*MyCluster*以叢集的名稱取代。此名稱會用於存放代理程式所收集之日誌事件的日誌群組名稱中，也用作為代理程式所收集之指標的維度。

將*##*取代為您要傳送量度的 AWS 地區名稱。例如 **us-west-1**。

確認代理程式正在執行中

在 Amazon EKS 和 Kubernetes 叢集上，您可以輸入下列命令，以確認代理程式正在執行。

```
kubectl get pod -l "app=cwagent-prometheus" -n amazon-cloudwatch
```

如果結果包含Running狀態中的單一 CloudWatch 代理程式網繭，表示代理程式正在執行並收集 Prometheus 度量。默認情況下， CloudWatch代理程序每分鐘收集應用程序網格，NGINX，內存緩存，Java/JMX 和 HAProxy 的指標。如需這些指標的詳細資訊，請參閱 代理程式收集的 Prometheus 測量結果 CloudWatch 。有關如何在中查看您的 Prometheus 指標的說明，請參閱 CloudWatch 檢視 Prometheus 指標

您也可以將 CloudWatch 代理程式設定為從其他 Prometheus 匯出器收集指標。如需詳細資訊，請參閱湊集其他 Prometheus 來源並匯入這些指標。

湊集其他 Prometheus 來源並匯入這些指標

具有 Prometheus 監控的 CloudWatch 代理需要兩種配置來抓取 Prometheus 指標。其中一個是 Prometheus 文件的 <scrape_config> 中記錄的標準 Prometheus 湊集組態。另一個用於代 CloudWatch 理配置。

對於 Amazon EKS 叢集，組態在 `prometheus-eks.yaml` (EC2 啟動類型) 或 `prometheus-eks-fargate.yaml` (Fargate 啟動類型) 中定義為兩種組態映射：

- `name: prometheus-config` 區段包含 Prometheus 湊集的設定。
- 此`name: prometheus-cwagentconfig`區段包含代 CloudWatch 理程式的組態。您可以使用此段落來設定如何收集 Prometheus 測量結果。 CloudWatch例如，您可以指定要匯入哪些量度 CloudWatch，並定義其維度。

對於在 Amazon EC2 執行個體上執行的 Kubernetes 叢集，組態在 `prometheus-k8s.yaml` YAML 檔案中定義為兩種組態映射：

- `name: prometheus-config` 區段包含 Prometheus 湊集的設定。
- 此`name: prometheus-cwagentconfig`區段包含代 CloudWatch 理程式的組態。

若要抓取其他 Prometheus 指標來源並將這些指標匯入 CloudWatch，請修改 Prometheus 抓取設定和代理程式組態，然後使用更新的組態重新部署 CloudWatch 代理程式。

VPC 安全群組要求

Prometheus 工作負載的安全群組輸入規則必須向 CloudWatch 代理程式開啟 Prometheus 連接埠，以便透過私有 IP 擷取 Prometheus 度量。

代理程式的安全性群組輸出規則必須允許 CloudWatch 代理程式透過 CloudWatch 私有 IP 連線至 Prometheus 工作負載的連接埠。

Prometheus 湊集組態

該 CloudWatch 代理支持標準的 Prometheus 抓取配置，如 Prometheus 文檔https://prometheus.io/docs/prometheus/latest/configuration/configuration/#scrape_config<scrape_config>中所述。您可以編輯此區段來更新已存在於此檔案中的組態，並新增其他 Prometheus 湊集目標。根據預設，範例組態檔案包含下列全域組態行：

```
global:
```

```
  scrape_interval: 1m
  scrape_timeout: 10s
```

- scrape_interval— 定義湊集目標的頻率。
- scrape_timeout— 定義湊集請求逾時之前要等待的時間。

您也可以在任務層級為這些設定定義不同的數值，以覆寫全域設定。

Prometheus 湊集任務

CloudWatch 代理程式 YAML 檔案已設定一些預設抓取工作。例如，在 `prometheus-eks.yaml` 中，已在 `scrape_configs` 區段中的 `job_name` 行設定預設湊集任務。在此檔案中，下列預設 `kubernetes-pod-jmx` 區段會湊集 JMX Exporter 指標。

```
  - job_name: 'kubernetes-pod-jmx'
     sample_limit: 10000
     metrics_path: /metrics
     kubernetes_sd_configs:
     - role: pod
     relabel_configs:
     - source_labels: [__address__]
       action: keep
       regex: '.*:9404$'
     - action: labelmap
       regex: __meta_kubernetes_pod_label_(.+)
     - action: replace
       source_labels:
       - __meta_kubernetes_namespace
       target_label: Namespace
     - source_labels: [__meta_kubernetes_pod_name]
       action: replace
       target_label: pod_name
     - action: replace
       source_labels:
       - __meta_kubernetes_pod_container_name
       target_label: container_name
     - action: replace
       source_labels:
       - __meta_kubernetes_pod_controller_name
       target_label: pod_controller_name
     - action: replace
       source_labels:
```

```
        - __meta_kubernetes_pod_controller_kind
          target_label: pod_controller_kind
        - action: replace
          source_labels:
          - __meta_kubernetes_pod_phase
          target_label: pod_phase
```

這些預設目標 CloudWatch 中的每一個都會被抓取，並使用內嵌指標格式將指標傳送至記錄事件中。如需詳細資訊，請參閱 在日誌中內嵌指標。

*## Amazon EKS # Kubernetes ############### **/aws/####/####/** Prometheus ######* CloudWatch 來自 Amazon ECS 叢集的日誌事件會存放在 /aws/ecs/containerinsights/***cluster_name***/prometheus 日誌群組。

每個湊集任務都包含在此日誌群組的不同日誌串流中。例如，Prometheus 湊集任務 `kubernetes-pod-appmesh-envoy` 是針對 App Mesh 所定義。***## Amazon EKS # Kubernetes ##### App Mesh ############### /aws/####/##_## > Prometheus//#######*** kubernetes-pod-appmesh-envoy

若要新增湊集目標，請新增 `job_name` 區段到 YAML 檔案的 `scrape_configs` 區段，然後重新啟動代理程式。如需此程序的範例，請參閱新增 Prometheus 湊集目標的教學課程：Prometheus API 伺服器指標。

CloudWatch Prometheus 的代理程式組態

CloudWatch 代理配置文件在 Prometheus 抓`metrics_collected`取配置下有一個`prometheus`部分。其包含下列組態選項：

- cluster_name— 指定要在日誌事件中新增為標籤的叢集名稱。此欄位為選用欄位。如果您省略此值，代理程式可以偵測 Amazon EKS 或 Kubernetes 叢集名稱。
- log_group_name— 為湊集的 Prometheus 指標指定日誌檔案群組名稱。此欄位為選用欄位。如果您省略它，請針對來自 Amazon EKS 和 Kubernetes 叢集的日誌 CloudWatch 使用 /aws/容器探索/ ***#### ##/***prometheus。
- prometheus_config_path— 指定 Prometheus 湊集組態檔案路徑。如果此欄位的值以 `env:` 為開頭，則將從容器的環境變數中擷取 Prometheus 湊集組態檔案內容。請不要變更此欄位。
- ecs_service_discovery— 是指定 Amazon ECS Prometheus 服務探索組態的區段。如需詳細資訊，請參閱 在 Amazon ECS 叢集上自動探索的詳細指南。

  `ecs_service_discovery` 區段可以包含下列欄位：

- `sd_frequency` 是發現 Prometheus Exporters 的頻率。指定數字和單位尾碼。例如：每分鐘一次 `1m` 或每 30 秒`30s`一次。有效的單位尾碼為`ns`、`us`、`ms`、`s`、`m` 以及 `h`。

  此欄位為選用欄位。預設值為 60 秒 (1 分鐘)。

- `sd_target_cluster` 是用於自動探索的目標 Amazon ECS 叢集名稱。此欄位為選用欄位。預設值為安裝 CloudWatch 代理程式的 Amazon ECS 叢集名稱。

- `sd_cluster_region` 是目標 Amazon ECS 叢集的區域。此欄位為選用欄位。預設為安裝 CloudWatch 代理程式的 Amazon ECS 叢集區域。

- `sd_result_file` 是 Prometheus 目標結果的 YAML 檔案路徑。Prometheus 湊集組態將參與此檔案。

- `docker_label` 是選用區段，您可以用它來指定 Docker 標籤型服務探索的組態。如果您省略此區段，則不會使用 Docker 標籤型探索。此區段可以包含下列欄位：

  - `sd_port_label` 是容器的 Docker 標籤名稱，用於指定 Prometheus 指標的容器連接埠。預設值為 `ECS_PROMETHEUS_EXPORTER_PORT`。如果容器沒有此 docker 標籤，則 CloudWatch 代理程式會略過它。

  - `sd_metrics_path_label` 是容器的 Docker 標籤名稱，用於指定 Prometheus 指標路徑。預設值為 `ECS_PROMETHEUS_METRICS_PATH`。如果容器沒有此 Docker 標籤，則代理程式會假設預設路徑 `/metrics`。

  - `sd_job_name_label` 是容器的 Docker 標籤名稱，用於指定 Prometheus 湊集任務名稱。預設值為 `job`。如果容器沒有此 docker 標籤，則 CloudWatch 代理程式會使用 Prometheus 抓取設定中的作業名稱。

- `task_definition_list` 是選用區段，您可以用它來指定任務定義型服務探索的組態。如果您省略此區段，則不會使用任務定義型探索。此區段可以包含下列欄位：

  - `sd_task_definition_arn_pattern` 是用來指定要探索的 Amazon ECS 任務定義的模式。這是規則表達式。

  - `sd_metrics_ports` 列出了 Prometheus 指標的 containerPort。使用分號分隔 containerPorts。

  - `sd_container_name_pattern` 指定了 Amazon ECS 任務容器名稱。這是規則表達式。

  - `sd_metrics_path` 指定了 Prometheus 指標路徑。如果您省略此項，代理程式會假設預設路徑 `/metrics`

  - `sd_job_name` 指定了 Prometheus 湊集任務名稱。如果您省略此欄位， CloudWatch 代理程式會使用 Prometheus 抓取設定中的工作名稱。

- metric_declaration— 是以要產生之內嵌指標格式來指定日誌陣列的區段。根據預設， CloudWatch 代理程式從中匯入的每個 Prometheus 來源都有metric_declaration區段。這些區段各包括下列欄位：
  - label_matcher 是一個規則表達式，會檢查 source_labels 中列出的標籤值。符合的量度會啟用以包含在傳送至的內嵌量度格式中 CloudWatch。

    如果您在 source_labels 中指定了多個標籤，我們建議您不要在 label_matcher 的規則表達式中使用 ^ 或 $ 字元。
  - source_labels 指定由 label_matcher 行檢查的標籤值。
  - label_separator 指定要在 label_matcher 行中使用的分隔符號 (如果指定多個 source_labels)。預設值為 ;。您可以在下面的範例中看到 label_matcher 行中使用此預設值。
  - metric_selectors是一個規則運算式，用來指定要收集並傳送至的測量結果 CloudWatch。
  - dimensions是要作為每個選取量度之 CloudWatch維度使用的標籤清單。

請參閱以下 metric_declaration 範例。

```
"metric_declaration": [
  {
     "source_labels":[ "Service", "Namespace"],
     "label_matcher":"(.*node-exporter.*|.*kube-dns.*);kube-system",
     "dimensions":[
        ["Service", "Namespace"]
     ],
     "metric_selectors":[
        "^coredns_dns_request_type_count_total$"
     ]
  }
]
```

此範例會在符合下列條件時，設定內嵌指標格式區段，以作為日誌事件傳送：

- Service 的數值包含 node-exporter 或 kube-dns。
- Namespace 的值為 kube-system。
- Prometheus 指標 coredns_dns_request_type_count_total 包含 Service 和 Namespace 標籤。

傳送的日誌事件包含下列反白顯示的區段：

```
{
   "CloudWatchMetrics":[
      {
         "Metrics":[
            {
               "Name":"coredns_dns_request_type_count_total"
            }
         ],
         "Dimensions":[
            [
               "Namespace",
               "Service"
            ]
         ],
         "Namespace":"ContainerInsights/Prometheus"
      }
   ],
   "Namespace":"kube-system",
   "Service":"kube-dns",
   "coredns_dns_request_type_count_total":2562,
   "eks_amazonaws_com_component":"kube-dns",
   "instance":"192.168.61.254:9153",
   "job":"kubernetes-service-endpoints",
   ...
}
```

新增 Prometheus 湊集目標的教學課程：Prometheus API 伺服器指標

Kubernetes API 伺服器預設會在端點上公開 Prometheus 指標。Kubernetes API 伺服器湊集組態的官方範例可以在 Github 上取得。

下列教學課程說明如何執行下列步驟，以開始將 Kubernetes API 伺服器量度匯入至： CloudWatch

- 將 Kubernetes API 伺服器的 Prometheus 抓取設定新增至代理程式 YAML 檔案。 CloudWatch
- 在 CloudWatch 代理程式 YAML 檔案中設定內嵌度量格式度量定義。
- (選擇性) 為 Kubernetes API 伺服器量度建立 CloudWatch 儀表板。

> **Note**
>
> Kubernetes API 伺服器會公開量測、計數器、長條圖和摘要指標。在此版本的 Prometheus 量度支援中，只 CloudWatch 匯入具有量測計、計數器和摘要類型的度量。

若要開始收集 API 伺服器 Prometheus 指標 CloudWatch

1. 輸入下列其中一個命令，以下載最新版本的 `prometheus-eks.yaml`、`prometheus-eks-fargate.yaml` 或 `prometheus-k8s.yaml` 檔案。

   對於具有 EC2 啟動類型的 Amazon EKS 叢集，請輸入下列命令：

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks.yaml
   ```

   對於具有 Fargate 啟動類型的 Amazon EKS 叢集，請輸入下列命令：

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks-fargate.yaml
   ```

   對於在 Amazon EC2 執行個體上執行的 Kubernetes 叢集，請輸入下列命令：

   ```
   curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-k8s.yaml
   ```

2. 使用文字編輯器開啟檔案，找出 `prometheus-config` 區段，並在該區段內新增以下區段。儲存變更：

   ```
       # Scrape config for API servers
       - job_name: 'kubernetes-apiservers'
         kubernetes_sd_configs:
           - role: endpoints
             namespaces:
               names:
                 - default
         scheme: https
   ```

```
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        insecure_skip_verify: true
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
      relabel_configs:
      - source_labels: [__meta_kubernetes_service_name,
 __meta_kubernetes_endpoint_port_name]
        action: keep
        regex: kubernetes;https
      - action: replace
        source_labels:
        - __meta_kubernetes_namespace
        target_label: Namespace
      - action: replace
        source_labels:
        - __meta_kubernetes_service_name
        target_label: Service
```

3. 趁文字編輯器中開啟 YAML 檔案時，找出 cwagentconfig.json 區段。新增下列子區段並儲存變更。本節將 API 伺服器度量放入 CloudWatch 代理程式允許清單中。允許清單中會新增三種類型的 API 伺服器指標：

   - etcd 物件計數
   - API 伺服器註冊控制器指標
   - API 伺服器請求指標

```
{"source_labels": ["job", "resource"],
  "label_matcher": "^kubernetes-apiservers;(services|daemonsets.apps|
deployments.apps|configmaps|endpoints|secrets|serviceaccounts|replicasets.apps)",
  "dimensions": [["ClusterName","Service","resource"]],
  "metric_selectors": [
  "^etcd_object_counts$"
  ]
},
{"source_labels": ["job", "name"],
   "label_matcher": "^kubernetes-apiservers;APIServiceRegistrationController$",
   "dimensions": [["ClusterName","Service","name"]],
   "metric_selectors": [
   "^workqueue_depth$",
   "^workqueue_adds_total$",
   "^workqueue_retries_total$"
```

```
  ]
},
{"source_labels": ["job","code"],
  "label_matcher": "^kubernetes-apiservers;2[0-9]{2}$",
  "dimensions": [["ClusterName","Service","code"]],
  "metric_selectors": [
   "^apiserver_request_total$"
  ]
},
{"source_labels": ["job"],
  "label_matcher": "^kubernetes-apiservers",
  "dimensions": [["ClusterName","Service"]],
  "metric_selectors": [
  "^apiserver_request_total$"
  ]
},
```

4. 如果您已經在叢集中部署了具有 Prometheus 支援的 CloudWatch 代理程式，則必須輸入下列命令將其刪除：

```
kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch
```

5. 輸入下列其中一個命令，以您更新的組態部署 CloudWatch 代理程式。對於具有 EC2 啟動類型的 Amazon EKS 叢集，請輸入：

```
kubectl apply -f prometheus-eks.yaml
```

對於具有 Fargate 啟動類型的 Amazon EKS 叢集，請輸入下列命令。使用值取代*MyCluster*並*##*化，以符合您的部署。

```
cat prometheus-eks-fargate.yaml \
| sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" \
| kubectl apply -f -
```

對於 Kubernetes 叢集，請輸入下列命令。使用值取代*MyCluster*並*##*化，以符合您的部署。

```
cat prometheus-k8s.yaml \
| sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" \
| kubectl apply -f -
```

完成此操作後，您應該在 /aws/containerinsights/***cluster_name***/prometheus 日誌群組中查看名為 kubernetes-apiservers 的新日誌串流。此日誌串流應該包含具有內嵌指標格式定義的日誌事件，如下所示：

```
{
   "CloudWatchMetrics":[
      {
         "Metrics":[
            {
               "Name":"apiserver_request_total"
            }
         ],
         "Dimensions":[
            [
               "ClusterName",
               "Service"
            ]
         ],
         "Namespace":"ContainerInsights/Prometheus"
      }
   ],
   "ClusterName":"my-cluster-name",
   "Namespace":"default",
   "Service":"kubernetes",
   "Timestamp":"1592267020339",
   "Version":"0",
   "apiserver_request_count":0,
   "apiserver_request_total":0,
   "code":"0",
   "component":"apiserver",
   "contentType":"application/json",
   "instance":"192.0.2.0:443",
   "job":"kubernetes-apiservers",
   "prom_metric_type":"counter",
   "resource":"pods",
   "scope":"namespace",
   "verb":"WATCH",
   "version":"v1"
}
```

您可以在 ContainerInsights/Prome theus 命名空間的 CloudWatch 主控台中檢視您的指標。您也可以選擇為您的 Prometheus API 伺服器指標建立 CloudWatch 儀表板。

(選用) 為 Kubernetes API 伺服器指標建立儀表板。

若要在儀表板中查看 Kubernetes API 伺服器量度，您必須先完成上一節中的步驟，才能開始在中收集這些量度。 CloudWatch

若要建立 Kubernetes API 伺服器指標的儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 請確定您已選取正確的「 AWS 區域」。
3. 在導覽窗格中，選擇 Dashboards (儀表板)。
4. 選擇 Create Dashboard (建立儀表板)。輸入新儀表板的名稱，然後選擇 Create dashboard (建立儀表板)。
5. 在 Add to this dashboard (新增至此儀表板) 中，選擇 Cancel (取消)。
6. 選擇 Actions (動作)、View/edit sources (檢視/編輯來源)。
7. 下載下列 JSON 檔案：Kubernetes API 儀表板來源。
8. 使用文字編輯器開啟您下載的 JSON 檔案，然後進行下列變更：
   - 將所有 `{{YOUR_CLUSTER_NAME}}` 字串取代為您叢集的確切名稱。確定不要在文字前後加空格。
   - 將所有 `{{YOUR_AWS_REGION}}` 字串取代為收集指標的區域的名稱。例如 `us-west-2`。確定不要在文字前後加空格。
9. 複製整個 JSON blob 並將其粘貼到 CloudWatch控制台的文本框中，替換框中已經存在的內容。
10. 選擇 Update (更新)、Save dashboard (儲存儀表板)。

(選用) 設定範例容器化 Amazon EKS 工作負載範例進行 Prometheus 指標測試

若要在 CloudWatch 容器深入解析中測試 Prometheus 指標支援，您可以設定下列一或多個容器化工作負載。支援 Prometheus 的 CloudWatch 代理程式會自動從這些工作負載收集指標。若要查看預設收集的指標，請參閱 代理程式收集的 Prometheus 測量結果 CloudWatch 。

在安裝這些工作負載之前，您必須先輸入下列命令來安裝 Helm 3.x：

```
brew install helm
```

如需詳細資訊，請參閱 Helm。

主題

- 設定適用於 Amazon EKS 和 Kubernetes 的 AWS App Mesh 範例工作負載
- 在 Amazon EKS 和 Kubernetes 上使用範例流量設定 NGINX
- 在 Amazon EKS 和 Kubernetes 上使用指標匯出工具設定 memcached
- 設定適用於 Amazon EKS 和 Kubernetes 的 Java/JMX 範例工作負載
- 在 Amazon EKS 和 Kubernetes 上使用指標匯出工具設定 HAProxy
- 新增 Prometheus 湊集目標的教學課程：Amazon EKS 和 Kubernetes 叢集上的 Redis

設定適用於 Amazon EKS 和 Kubernetes 的 AWS App Mesh 範例工作負載

在 CloudWatch 容器洞察支持 Prometheus 支持。 AWS App Mesh以下各節說明如何設定 App Mesh。

CloudWatch 容器見解也可以收集 App Mesh 特使存取記錄。如需詳細資訊，請參閱 (選擇性) 啟用 App Mesh Envoy 存取日誌。

主題

- 設定具有 EC2 啟動類型或 Kubernetes 叢集的 Amazon EKS 叢集上的 AWS App Mesh 範例工作負載
- 使用 Fargate 啟動類型在 Amazon EKS 叢集上設定 AWS App Mesh 範例工作負載

設定具有 EC2 啟動類型或 Kubernetes 叢集的 Amazon EKS 叢集上的 AWS App Mesh 範例工作負載

如果您要在執行 Amazon EKS 且具有 EC2 啟動類型的叢集或 Kubernetes 叢集上設定 App Mesh，請使用這些指示。

設定 IAM 許可

您必須將AWSAppMeshFullAccess政策新增至 Amazon EKS 或 Kubernetes 節點群組的 IAM 角色。在 Amazon EKS 上，此節點群組名稱看起來類似於 `eksctl-integ-test-eks-prometheus-NodeInstanceRole-ABCDEFHIJKL`。在 Kubernetes 上，它可能看起來類似於 `nodes.integ-test-kops-prometheus.k8s.local`.

安裝 App Mesh

若要安裝 App Mesh Kubernetes 控制器，請遵循 App Mesh 控制器中的說明進行。

安裝範例應用程式

aws-app-mesh-examples包含數個 Kubernetes App Mesh 逐步解說。在本教學課程中，您將安裝一個範例色彩應用程式，該應用程式顯示 http 路由如何使用標頭來比對傳入的請求。

若要使用範例 App Mesh 應用程式來測試 Container Insights

1. 使用這些指示安裝應用程式：https://github.com/aws/aws-app-mesh-examples/tree/main/walkthroughs/howto-k8s-http-headers。
2. 啟動 curler pod 以產生流量：

```
kubectl -n default run -it curler --image=tutum/curl /bin/bash
```

3. 透過變更 HTTP 標頭來 Curl 不同的端點。多次執行 curl 命令，如此處所示：

```
curl -H "color_header: blue" front.howto-k8s-http-headers.svc.cluster.local:8080/;
 echo;

curl -H "color_header: red" front.howto-k8s-http-headers.svc.cluster.local:8080/;
 echo;

curl -H "color_header: yellow" front.howto-k8s-http-
headers.svc.cluster.local:8080/; echo;
```

4. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
5. 在叢集執行的 [AWS 區域] 中，選擇導覽窗格中的 [指標]。度量位於 ContainerInsights/Prometheus 命名空間中。
6. 若要查看記 CloudWatch 錄事件，請在導覽窗格中選擇 [記錄群組]。事件位於日誌串流 `kubernetes-pod-appmesh-envoy` 中的 日誌群組 `/aws/containerinsights/`*`your_cluster_name`*`/prometheus` 中。

刪除 App Mesh 測試環境

當您使用完 App Mesh 和範例應用程式時，請使用以下命令刪除不必要的資源。輸入下列命令以刪除範例應用程式：

```
cd aws-app-mesh-examples/walkthroughs/howto-k8s-http-headers/
kubectl delete -f _output/manifest.yaml
```

輸入以下命令以刪除 App Mesh 控制器：

```
helm delete appmesh-controller -n appmesh-system
```

使用 Fargate 啟動類型在 Amazon EKS 叢集上設定 AWS App Mesh 範例工作負載

如果您要在執行 Amazon EKS 且具有 Fargate 啟動類型的叢集上設定 App Mesh，請使用這些指示。

設定 IAM 許可

若要設定 IAM 許可，請輸入以下命令。*MyCluster*以叢集的名稱取代。

```
eksctl create iamserviceaccount --cluster MyCluster \
 --namespace howto-k8s-fargate \
 --name appmesh-pod \
 --attach-policy-arn arn:aws:iam::aws:policy/AWSAppMeshEnvoyAccess \
 --attach-policy-arn arn:aws:iam::aws:policy/AWSCloudMapDiscoverInstanceAccess \
 --attach-policy-arn arn:aws:iam::aws:policy/AWSXRayDaemonWriteAccess \
 --attach-policy-arn arn:aws:iam::aws:policy/CloudWatchLogsFullAccess \
 --attach-policy-arn arn:aws:iam::aws:policy/AWSAppMeshFullAccess \
 --attach-policy-arn arn:aws:iam::aws:policy/AWSCloudMapFullAccess \
 --override-existing-serviceaccounts \
 --approve
```

安裝 App Mesh

若要安裝 App Mesh Kubernetes 控制器，請遵循 App Mesh 控制器中的說明進行。對於具有 Fargate 啟動類型的 Amazon EKS，請務必遵循指示。

安裝範例應用程式

aws-app-mesh-examples包含數個 Kubernetes App Mesh 逐步解說。在本教學課程中，您將安裝適用於具有 Fargate 啟動類型的 Amazon EKS 叢集的範例色彩應用程式。

若要使用範例 App Mesh 應用程式來測試 Container Insights

1. 使用這些指示安裝應用程式：https://github.com/aws/aws-app-mesh-examples/tree/main/walkthroughs/howto-k8s-fargate。

   這些指示假設您正在建立一個具有正確 Fargate 設定檔的新叢集。對於此示範，如果您想要使用已經設定的 Amazon EKS 叢集，可以使用下列命令來設定該叢集。*MyCluster*以叢集的名稱取代。

   ```
   eksctl create iamserviceaccount --cluster MyCluster \
    --namespace howto-k8s-fargate \
   ```

```
--name appmesh-pod \
--attach-policy-arn arn:aws:iam::aws:policy/AWSAppMeshEnvoyAccess \
--attach-policy-arn arn:aws:iam::aws:policy/AWSCloudMapDiscoverInstanceAccess \
--attach-policy-arn arn:aws:iam::aws:policy/AWSXRayDaemonWriteAccess \
--attach-policy-arn arn:aws:iam::aws:policy/CloudWatchLogsFullAccess \
--attach-policy-arn arn:aws:iam::aws:policy/AWSAppMeshFullAccess \
--attach-policy-arn arn:aws:iam::aws:policy/AWSCloudMapFullAccess \
--override-existing-serviceaccounts \
--approve
```

```
eksctl create fargateprofile --cluster MyCluster \
--namespace howto-k8s-fargate --name howto-k8s-fargate
```

2. 前端應用程式部署的連接埠：

```
kubectl -n howto-k8s-fargate port-forward deployment/front 8080:8080
```

3. Curl 前端應用程式：

```
while true; do  curl -s http://localhost:8080/color; sleep 0.1; echo ; done
```

4. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
5. 在叢集執行的 [AWS 區域] 中，選擇導覽窗格中的 [指標]。度量位於 ContainerInsights/Prometheus 命名空間中。
6. 若要查看記 CloudWatch 錄事件，請在導覽窗格中選擇 [記錄群組]。事件位於日誌串流 kubernetes-pod-appmesh-envoy 中的 日誌群組 `/aws/containerinsights/`*`your_cluster_name`*`/prometheus` 中。

刪除 App Mesh 測試環境

當您使用完 App Mesh 和範例應用程式時，請使用以下命令刪除不必要的資源。輸入下列命令以刪除範例應用程式：

```
cd aws-app-mesh-examples/walkthroughs/howto-k8s-fargate/
kubectl delete -f _output/manifest.yaml
```

輸入以下命令以刪除 App Mesh 控制器：

```
helm delete appmesh-controller -n appmesh-system
```

在 Amazon EKS 和 Kubernetes 上使用範例流量設定 NGINX

NGINX 是一部 Web 伺服器，可同時用作為負載平衡器和反向代理。如需如何使用 NGINX 進行擷取的詳細資訊，請參閱 kubernetes/ingress-nginx。

使用範例流量服務安裝 Ingress-NGINX 以測試 Container Insights Prometheus 支援

1. 輸入以下命令來新增 Helm ingress-nginx 儲存庫：

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
```

2. 輸入下列命令：

```
kubectl create namespace nginx-ingress-sample

helm install my-nginx ingress-nginx/ingress-nginx \
--namespace nginx-ingress-sample \
--set controller.metrics.enabled=true \
--set-string controller.metrics.service.annotations."prometheus\.io/port"="10254" \
--set-string controller.metrics.service.annotations."prometheus\.io/scrape"="true"
```

3. 請輸入下列命令，檢查服務是否已正確啟動：

```
kubectl get service -n nginx-ingress-sample
```

   這個命令的輸出應該會顯示數欄，包括一個 EXTERNAL-IP 欄。

4. 將 EXTERNAL-IP 變數設定為 NGINX 傳入控制器一行中 EXTERNAL-IP 欄的數值。

```
EXTERNAL_IP=your-nginx-controller-external-ip
```

5. 輸入以下命令啟動一些範例 NGINX 流量。

```
SAMPLE_TRAFFIC_NAMESPACE=nginx-sample-traffic
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/sample_traffic/nginx-traffic/nginx-traffic-sample.yaml |
sed "s/{{external_ip}}/$EXTERNAL_IP/g" |
sed "s/{{namespace}}/$SAMPLE_TRAFFIC_NAMESPACE/g" |
kubectl apply -f -
```

6. 輸入下列命令以確認三個 pod 全都處於 Running 狀態。

```
kubectl get pod -n $SAMPLE_TRAFFIC_NAMESPACE
```

如果它們正在運行，您很快就會在 ContainerInsights/Prome theus 命名空間中看到指標。

解除安裝 NGINX 和範例流量應用程式

1. 輸入下列命令以刪除範例流量服務：

```
kubectl delete namespace $SAMPLE_TRAFFIC_NAMESPACE
```

2. 刪除 Helm 發行版本名稱的 NGINX egress。

```
helm uninstall my-nginx --namespace nginx-ingress-sample
kubectl delete namespace nginx-ingress-sample
```

在 Amazon EKS 和 Kubernetes 上使用指標匯出工具設定 memcached

memcached 是一個開放原始碼記憶體物件快取系統。如需詳細資訊，請參閱什麼是 Memcached？。

如果您在具有 Fargate 啟動類型的叢集上執行 memcached，則需要在執行此程序中的步驟之前設定 Fargate 描述檔。若要設定描述檔，請輸入下列命令。*MyCluster*以叢集的名稱取代。

```
eksctl create fargateprofile --cluster MyCluster \
--namespace memcached-sample --name memcached-sample
```

使用指標匯出工具安裝 memcached，以測試 Container Insights Prometheus 支援

1. 輸入以下命令來新增儲存庫：

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

2. 輸入以下命令來建立新的命名空間：

```
kubectl create namespace memcached-sample
```

3. 輸入下列命令以安裝 Memcached

```
helm install my-memcached bitnami/memcached --namespace memcached-sample \
--set metrics.enabled=true \
--set-string serviceAnnotations.prometheus\\.io/port="9150" \
--set-string serviceAnnotations.prometheus\\.io/scrape="true"
```

4. 輸入下列命令以確認執行中服務的註釋：

```
kubectl describe service my-memcached-metrics -n memcached-sample
```

您應該會看到下列兩個註釋：

```
Annotations:   prometheus.io/port: 9150
               prometheus.io/scrape: true
```

解除安裝 memcached

- 輸入下列命令：

```
helm uninstall my-memcached --namespace memcached-sample
kubectl delete namespace memcached-sample
```

設定適用於 Amazon EKS 和 Kubernetes 的 Java/JMX 範例工作負載

JMX Exporter 是官方的 Prometheus 匯出工具，可以湊集 JMX mBeans 並將其公開為 Prometheus 指標。如需詳細資訊，請參閱 prometheus/jmx_exporter。

Container Insights 可以使用 JMX Exporter，從 Java 虛擬機器 (JVM)、Java 和 Tomcat (Catalina) 收集預先定義的 Prometheus 指標。

預設 Prometheus 湊集組態

根據預設，具有 Prometheus 支援的 CloudWatch 代理程式會從 Amazon EKS 或 Kubernetes 叢集中的每個網繭`http://CLUSTER_IP:9404/metrics`上抓取 Java/JMX Prometheus 指標。這是透過會由 Prometheus `kubernetes_sd_config` 的 `role: pod` 探索的。9404 是 Prometheus 為 JMX Exporter 配置的預設連接埠。如需 `role: pod` 探索的詳細資訊，請參閱 pod。您可以設定 JMX Exporter 在不同的連接埠或 metrics_path 上公開指標。如果您確實變更了連接埠或路徑，請在代理程式設定對應中更新預設的 jmx scrape_config。 CloudWatch 執行下列命令以取得目前的 CloudWatch 代理程式 Prometheus 組態：

```
kubectl describe cm prometheus-config -n amazon-cloudwatch
```

要變更的欄位為 /metrics 和 regex: '.*:9404$' 欄位，如下列範例中反白所示。

```
job_name: 'kubernetes-jmx-pod'
sample_limit: 10000
metrics_path: /metrics
kubernetes_sd_configs:
- role: pod
relabel_configs:
- source_labels: [__address__]
  action: keep
  regex: '.*:9404$'
- action: replace
  regex: (.+)
  source_labels:
```

其他 Prometheus 湊集組態

如果您透過 Kubernetes Service 使用 Java/JMX Prometheus 匯出工具公開在一組 Pod 上執行的應用程式，則還可以切換到使用 role: service 探索或 Prometheus kubernetes_sd_config 的 role: endpoint 探索。如需這些探索方法的詳細資訊，請參閱服務、端點和 <kubernetes_sd_config>。

這兩種服務發現模式提供了更多中繼標籤，這對您構建指標維 CloudWatch 度可能很有用。例如，您可以重新標記 __meta_kubernetes_service_name 至 Service，並將其包含在您的指標維度中。如需有關自訂 CloudWatch量度及其維度的詳細資訊，請參閱 CloudWatch Prometheus 的代理程式組態。

Docker 影像與 JMX Exporter

下一步：建置 Docker 影像 下列各節提供兩個範例 Dockerfile。

當您建置映像後，請將它載入 Amazon EKS 或 Kubernetes，然後執行下列命令，以確認 JMX_EXPORTER 是在連接埠 9404 上公開 Prometheus 指標。將 *$JAR_SAMPLE_TRAFFIC_POD* 取代為執行中的 pod 名稱，並將 *$JAR_SAMPLE_TRAFFIC_NAMESPACE* 取代為您的應用程式命名空間。

如果您在具有 Fargate 啟動類型的叢集上執行 JMX Exporter，則還需要在執行此程序中的步驟之前設定 Fargate 描述檔。若要設定描述檔，請輸入下列命令。*MyCluster*以叢集的名稱取代。

```
eksctl create fargateprofile --cluster MyCluster \
```

```
--namespace $JAR_SAMPLE_TRAFFIC_NAMESPACE\
 --name $JAR_SAMPLE_TRAFFIC_NAMESPACE
```

```
kubectl exec $JAR_SAMPLE_TRAFFIC_POD -n $JARCAT_SAMPLE_TRAFFIC_NAMESPACE -- curl
 http://localhost:9404
```

範例：具有 Prometheus 指標的 Apache Tomcat Docker 影像

Apache Tomcat 伺服器預設會公開 JMX mBeans。您可以將 JMX Exporter 與 Tomcat 整合，以將 JMX mBeans 公開為 Prometheus 指標。下列範例 Dockerfile 顯示建置測試映像的步驟：

```
# From Tomcat 9.0 JDK8 OpenJDK
FROM tomcat:9.0-jdk8-openjdk

RUN mkdir -p /opt/jmx_exporter

COPY ./jmx_prometheus_javaagent-0.12.0.jar /opt/jmx_exporter
COPY ./config.yaml /opt/jmx_exporter
COPY ./setenv.sh /usr/local/tomcat/bin
COPY your web application.war /usr/local/tomcat/webapps/

RUN chmod  o+x /usr/local/tomcat/bin/setenv.sh

ENTRYPOINT ["catalina.sh", "run"]
```

下列清單解釋了這個 Dockerfile 的四個 COPY 行。

- 從 https://github.com/prometheus/jmx_exporter 下載最新的 JMX Exporter jar 檔案。
- config.yaml 是 JMX Exporter 組態檔。如需詳細資訊，請參閱 https://github.com/prometheus/jmx_exporter#Configuration。

  以下是 Java 和 Tomcat 的範例組態檔：

  ```
  lowercaseOutputName: true
  lowercaseOutputLabelNames: true

  rules:
  - pattern: 'java.lang<type=OperatingSystem><>(FreePhysicalMemorySize|
  TotalPhysicalMemorySize|FreeSwapSpaceSize|TotalSwapSpaceSize|SystemCpuLoad|
  ProcessCpuLoad|OpenFileDescriptorCount|AvailableProcessors)'
    name: java_lang_OperatingSystem_$1
  ```

```
  type: GAUGE

- pattern: 'java.lang<type=Threading><>(TotalStartedThreadCount|ThreadCount)'
  name: java_lang_threading_$1
  type: GAUGE

- pattern: 'Catalina<type=GlobalRequestProcessor, name=\"(\w+-\w+)-(\d+)\"><>(\w+)'
  name: catalina_globalrequestprocessor_$3_total
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina global $3
  type: COUNTER

- pattern: 'Catalina<j2eeType=Servlet, WebModule=//([-a-zA-Z0-9+&@#/%?=~_|!:.,;]*[-
a-zA-Z0-9+&@#/%=~_|]), name=([-a-zA-Z0-9+/$%~_-|!.]*), J2EEApplication=none,
 J2EEServer=none><>(requestCount|maxTime|processingTime|errorCount)'
  name: catalina_servlet_$3_total
  labels:
    module: "$1"
    servlet: "$2"
  help: Catalina servlet $3 total
  type: COUNTER

- pattern: 'Catalina<type=ThreadPool, name="(\w+-\w+)-(\d+)"><>(currentThreadCount|
currentThreadsBusy|keepAliveCount|pollerThreadCount|connectionCount)'
  name: catalina_threadpool_$3
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina threadpool $3
  type: GAUGE

- pattern: 'Catalina<type=Manager, host=([-a-zA-Z0-9+&@#/%?=~_|!:.,;]*[-a-zA-
Z0-9+&@#/%=~_|]), context=([-a-zA-Z0-9+/$%~_-|!.]*)><>(processingTime|sessionCounter|
rejectedSessions|expiredSessions)'
  name: catalina_session_$3_total
  labels:
    context: "$2"
    host: "$1"
  help: Catalina session $3 total
  type: COUNTER
```

```
- pattern: ".*"
```

- `setenv.sh` 是一個 Tomcat 啟動指令碼，用於啟動 JMX Exporter 和 Tomcat，並在 localhost 的連接埠 9404 上公開 Prometheus 指標。它還為 JMX Exporter 提供了 `config.yaml` 檔案路徑。

```
$ cat setenv.sh
export JAVA_OPTS="-javaagent:/opt/jmx_exporter/
jmx_prometheus_javaagent-0.12.0.jar=9404:/opt/jmx_exporter/config.yaml $JAVA_OPTS"
```

- 您的 Web 應用程式 .war 是您要由 Tomcat 載入的 Web 應用程式 `war` 檔案。

使用此組態建置一個 Docker 影像並將其上傳到映像儲存庫。

範例：具有 Prometheus 指標的 Java Jar 應用程式 Docker 影像

下列範例 Dockerfile 顯示建置測試映像的步驟：

```
# Alpine Linux with OpenJDK JRE
FROM openjdk:8-jre-alpine

RUN mkdir -p /opt/jmx_exporter

COPY ./jmx_prometheus_javaagent-0.12.0.jar /opt/jmx_exporter
COPY ./SampleJavaApplication-1.0-SNAPSHOT.jar /opt/jmx_exporter
COPY ./start_exporter_example.sh /opt/jmx_exporter
COPY ./config.yaml /opt/jmx_exporter

RUN chmod -R o+x /opt/jmx_exporter
RUN apk add curl

ENTRYPOINT exec /opt/jmx_exporter/start_exporter_example.sh
```

下列清單解釋了這個 Dockerfile 的四個 `COPY` 行。

- 從 https://github.com/prometheus/jmx_exporter 下載最新的 JMX Exporter jar 檔案。
- `config.yaml` 是 JMX Exporter 組態檔。如需詳細資訊，請參閱 https://github.com/prometheus/jmx_exporter#Configuration。

  以下是 Java 和 Tomcat 的範例組態檔：

```
lowercaseOutputName: true
lowercaseOutputLabelNames: true
```

```
rules:
- pattern: 'java.lang<type=OperatingSystem><>(FreePhysicalMemorySize|
TotalPhysicalMemorySize|FreeSwapSpaceSize|TotalSwapSpaceSize|SystemCpuLoad|
ProcessCpuLoad|OpenFileDescriptorCount|AvailableProcessors)'
  name: java_lang_OperatingSystem_$1
  type: GAUGE

- pattern: 'java.lang<type=Threading><>(TotalStartedThreadCount|ThreadCount)'
  name: java_lang_threading_$1
  type: GAUGE

- pattern: 'Catalina<type=GlobalRequestProcessor, name=\"(\w+-\w+)-(\d+)\"><>(\w+)'
  name: catalina_globalrequestprocessor_$3_total
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina global $3
  type: COUNTER

- pattern: 'Catalina<j2eeType=Servlet, WebModule=//([-a-zA-Z0-9+&@#/%?=~_|!:.,;]*[-
a-zA-Z0-9+&@#/%=~_|]), name=([-a-zA-Z0-9+/$%~_-|!.]*), J2EEApplication=none,
 J2EEServer=none><>(requestCount|maxTime|processingTime|errorCount)'
  name: catalina_servlet_$3_total
  labels:
    module: "$1"
    servlet: "$2"
  help: Catalina servlet $3 total
  type: COUNTER

- pattern: 'Catalina<type=ThreadPool, name="(\w+-\w+)-(\d+)"><>(currentThreadCount|
currentThreadsBusy|keepAliveCount|pollerThreadCount|connectionCount)'
  name: catalina_threadpool_$3
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina threadpool $3
  type: GAUGE

- pattern: 'Catalina<type=Manager, host=([-a-zA-Z0-9+&@#/%?=~_|!:.,;]*[-a-zA-
Z0-9+&@#/%=~_|]), context=([-a-zA-Z0-9+/$%~_-|!.]*)><>(processingTime|sessionCounter|
rejectedSessions|expiredSessions)'
  name: catalina_session_$3_total
  labels:
```

```
    context: "$2"
    host: "$1"
  help: Catalina session $3 total
  type: COUNTER

- pattern: ".*"
```

- `start_exporter_example.sh` 是使用匯出的 Prometheus 指標來啟動 JAR 應用程式的指令碼。它還為 JMX Exporter 提供了 `config.yaml` 檔案路徑。

```
$ cat start_exporter_example.sh
java -javaagent:/opt/jmx_exporter/jmx_prometheus_javaagent-0.12.0.jar=9404:/
opt/jmx_exporter/config.yaml -cp  /opt/jmx_exporter/SampleJavaApplication-1.0-
SNAPSHOT.jar com.gubupt.sample.app.App
```

- SampleJavaApplication-1.0 快照 .jar 是範例 Java 應用程式 jar 檔案。將其取代為您要監控的 Java 應用程式。

使用此組態建置一個 Docker 影像並將其上傳到映像儲存庫。

在 Amazon EKS 和 Kubernetes 上使用指標匯出工具設定 HAProxy

HAProxy 是一個開放程式碼代理應用程式。如需詳細資訊，請參閱 HAProxy。

如果您在具有 Fargate 啟動類型的叢集上執行 HAProxy，則需要在執行此程序中的步驟之前設定 Fargate 描述檔。若要設定描述檔，請輸入下列命令。*MyCluster*以叢集的名稱取代。

```
eksctl create fargateprofile --cluster MyCluster \
--namespace haproxy-ingress-sample --name haproxy-ingress-sample
```

使用指標匯出工具安裝 HAProxy，以測試 Container Insights Prometheus 支援

1. 輸入下列命令以新增 Helm incubator 儲存庫：

   ```
   helm repo add haproxy-ingress https://haproxy-ingress.github.io/charts
   ```

2. 輸入以下命令來建立新的命名空間：

   ```
   kubectl create namespace haproxy-ingress-sample
   ```

3. 輸入下列命令來安裝 HAProxy：

```
helm install haproxy haproxy-ingress/haproxy-ingress \
--namespace haproxy-ingress-sample \
--set defaultBackend.enabled=true \
--set controller.stats.enabled=true \
--set controller.metrics.enabled=true \
--set-string controller.metrics.service.annotations."prometheus\.io/port"="9101" \
--set-string controller.metrics.service.annotations."prometheus\.io/scrape"="true"
```

4. 輸入下列命令以確認服務的註釋：

```
kubectl describe service haproxy-haproxy-ingress-metrics -n haproxy-ingress-sample
```

您應該會看到下列註釋。

```
Annotations:   prometheus.io/port: 9101
               prometheus.io/scrape: true
```

解除安裝 HAProxy

- 輸入下列命令：

```
helm uninstall haproxy --namespace haproxy-ingress-sample
kubectl delete namespace haproxy-ingress-sample
```

新增 Prometheus 湊集目標的教學課程：Amazon EKS 和 Kubernetes 叢集上的 Redis

本教學課程提供實作介紹，讓您在 Amazon EKS 和 Kubernetes 上湊集範例 Redis 應用程式的 Prometheus 指標。Redis (https://redis.io/) 是一個開放原始碼 (BSD 授權)、記憶體內的資料結構存放，可用作資料庫、存取和信息，緩存和訊息中介裝置。如需詳細資訊，請參閱 redis。

redis_exporter (授權的 MIT 授權) 可用於在指定的連接埠上公開 Redis Prometheus 指標 (預設：0.0.0.0:9121)。如需詳細資訊，請參閱 redis_exporter。

本教學課程會使用下列兩個 Docker Hub 儲存庫中的 Docker 影像：

- redis
- redis_exporter

若要安裝公開 Prometheus 指標的範例 Redis 工作負載

1. 設定範例 Redis 工作負載的命名空間。

```
REDIS_NAMESPACE=redis-sample
```

2. 如果您在具有 Fargate 啟動類型的叢集上執行 Redis，則需要設定 Fargate 描述檔。若要設定描述檔，請輸入下列命令。*MyCluster*以叢集的名稱取代。

```
eksctl create fargateprofile --cluster MyCluster \
--namespace $REDIS_NAMESPACE --name $REDIS_NAMESPACE
```

3. 輸入下列命令以安裝範例 Redis 工作負載。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/sample_traffic/redis/redis-traffic-sample.yaml \
| sed "s/{{namespace}}/$REDIS_NAMESPACE/g" \
| kubectl apply -f -
```

4. 此安裝包含名為 `my-redis-metrics` 的服務，而該服務會在連接埠 9121 上公開 Redis Prometheus 指標。請輸入下列命令，以取得服務的詳細資訊：

```
kubectl describe service/my-redis-metrics  -n $REDIS_NAMESPACE
```

在結果`Annotations`部分中，您將看到兩個符合 CloudWatch 代理程式 Prometheus 抓取配置的註釋，以便它可以自動探索工作負載：

```
prometheus.io/port: 9121
prometheus.io/scrape: true
```

相關的 Prometheus 湊集組態可以在 `kubernetes-eks.yaml` 或 `kubernetes-k8s.yaml` 的 `- job_name: kubernetes-service-endpoints` 區段找到。

若要開始收集雷迪斯 Prometheus 度量 CloudWatch

1. 輸入下列其中一個命令，以下載最新版本的 `kubernetes-eks.yaml` 或 `kubernetes-k8s.yaml` 檔案。對於具有 EC2 啟動類型的 Amazon EKS 叢集，請輸入此命令。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks.yaml
```

對於具有 Fargate 啟動類型的 Amazon EKS 叢集，請輸入此命令。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-eks-fargate.yaml
```

對於在 Amazon EC2 執行個體上執行的 Kubernetes 叢集，請輸入下列命令。

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-prometheus/prometheus-k8s.yaml
```

2. 使用文字編輯器開啟檔案，然後找出 `cwagentconfig.json` 區段。新增下列子區段並儲存變更。請務必遵循現有的縮排模式。

```
{
  "source_labels": ["pod_name"],
  "label_matcher": "^redis-instance$",
  "dimensions": [["Namespace","ClusterName"]],
  "metric_selectors": [
    "^redis_net_(in|out)put_bytes_total$",
    "^redis_(expired|evicted)_keys_total$",
    "^redis_keyspace_(hits|misses)_total$",
    "^redis_memory_used_bytes$",
    "^redis_connected_clients$"
  ]
},
{
  "source_labels": ["pod_name"],
  "label_matcher": "^redis-instance$",
  "dimensions": [["Namespace","ClusterName","cmd"]],
  "metric_selectors": [
    "^redis_commands_total$"
  ]
},
{
  "source_labels": ["pod_name"],
```

```
  "label_matcher": "^redis-instance$",
  "dimensions": [["Namespace","ClusterName","db"]],
  "metric_selectors": [
    "^redis_db_keys$"
  ]
},
```

您新增的區段會將 Redis 量度放入 CloudWatch 代理程式允許清單中。如需了解這些指標的清單，請參閱下列章節。

3. 如果您已在此叢集中部署具有 Prometheus 支援的 CloudWatch 代理程式，則必須輸入下列命令將其刪除。

```
kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch
```

4. 輸入下列其中一個命令，以您更新的組態部署 CloudWatch 代理程式。替換*MyCluster*和*##*以匹配您的設置。

   對於具有 EC2 啟動類型的 Amazon EKS 叢集，請輸入此命令。

```
kubectl apply -f prometheus-eks.yaml
```

   對於具有 Fargate 啟動類型的 Amazon EKS 叢集，請輸入此命令。

```
cat prometheus-eks-fargate.yaml \
| sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" \
| kubectl apply -f -
```

   對於 Kubernetes 叢集，請輸入此命令。

```
cat prometheus-k8s.yaml \
| sed "s/{{cluster_name}}/MyCluster/;s/{{region_name}}/region/" \
| kubectl apply -f -
```

檢視 Redis Prometheus 指標

本教學課程會將下列度量傳送至中的 ContainerInsights/Promethe us 命名空間。 CloudWatch您可以使用 CloudWatch 控制台查看該命名空間中的指標。

指標名稱	維度	
redis_net_input_bytes_total	ClusterName, Namespace	
redis_net_output_bytes_total	ClusterName, Namespace	
redis_expired_keys_total	ClusterName, Namespace	
redis_evicted_keys_total	ClusterName, Namespace	
redis_keyspace_hits_total	ClusterName, Namespace	
redis_keyspace_misses_total	ClusterName, Namespace	
redis_memory_used_bytes	ClusterName, Namespace	
redis_connected_clients	ClusterName, Namespace	
redis_commands_total	ClusterName,Namespace ，CMD	
redis_db_keys	ClusterName,Namespace ，資料庫	

> **Note**
>
> cmd 維度的數值可以
> 是：append、client、command、config、dbsize、flushall、get、incr、info
> latency 或 slowlog。
> db 維度的數值可以是 db0 至 db15。

您也可以為 Redis Prometheus 度量建立 CloudWatch 儀表板。

若要建立 Redis Prometheus 指標的儀表板

1. 建立環境變數，取代下面的數值，以符合您的部署。

```
DASHBOARD_NAME=your_cw_dashboard_name
REGION_NAME=your_metric_region_such_as_us-east-1
CLUSTER_NAME=your_k8s_cluster_name_here
NAMESPACE=your_redis_service_namespace_here
```

2. 輸入下列命令建立儀表板。

```
curl https://raw.githubusercontent.com/aws-samples/amazon-cloudwatch-container-
insights/latest/k8s-deployment-manifest-templates/deployment-mode/service/cwagent-
prometheus/sample_cloudwatch_dashboards/redis/cw_dashboard_redis.json \
| sed "s/{{YOUR_AWS_REGION}}/${REGION_NAME}/g" \
| sed "s/{{YOUR_CLUSTER_NAME}}/${CLUSTER_NAME}/g" \
| sed "s/{{YOUR_NAMESPACE}}/${NAMESPACE}/g" \
```

由代理程式進行的 Prometheus 度量類型轉換 CloudWatch

Prometheus 用戶端程式庫提供四種核心指標類型：

- 計數器
- 量測計
- Summary
- 直方圖

CloudWatch 代理程式支援計數器、量測計數器和摘要量度類型。計劃會在後續的版本支援長條圖指標。

代理程式會捨棄具有不支援色階分佈圖度量類型的 Prometheus 度量。 CloudWatch 如需詳細資訊，請參閱 記錄日誌已捨棄 Prometheus 指標。

量測計指標

Prometheus 量測計指標是一種可代表能任意上下的單一數值的指標。 CloudWatch 代理抓取衡量指標並直接發送這些值。

計數器指標

Prometheus 計數器指標是累積度量，可代表單一單調增加計數器，且其值只能增加或重設為零。CloudWatch代理程式會計算先前抓取的增量值，並將差異值作為日誌事件中的指標值傳送。因此，CloudWatch 代理程式將從第二次抓取中開始產生一個記錄事件，並繼續執行後續抓取（如果有的話）。

摘要指標

Prometheus 摘要指標是一種複雜的指標類型，且由多個資料點表示。它提供了觀察值的總計數和所有觀察值的總和。它會計算在滑動時段的可設定分位數。

摘要指標的總和和計數是累計的，但分數不是。下列範例顯示分位數的差異。

```
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 7.123e-06
go_gc_duration_seconds{quantile="0.25"} 9.204e-06
go_gc_duration_seconds{quantile="0.5"} 1.1065e-05
go_gc_duration_seconds{quantile="0.75"} 2.8731e-05
go_gc_duration_seconds{quantile="1"} 0.003841496
go_gc_duration_seconds_sum 0.37630427
go_gc_duration_seconds_count 9774
```

CloudWatch 代理程式處理摘要量度的總和和計數與處理計數器量度的方式相同，如前一節所述。CloudWatch 代理程式會保留原始報告的分位數值。

代理程式收集的 Prometheus 測量結果 CloudWatch

具有 Prometheus 支援的 CloudWatch 代理程式會自動從多個服務和工作負載收集指標。預設收集的指標列於下列各節中。您也可以將代理程式設定為從這些服務收集更多指標，以及從其他應用程式和服務

收集 Prometheus 指標。如需更多有關收集其他指標詳細資訊，請參閱 CloudWatch Prometheus 的代理程式組態。

從 Amazon EKS 和庫伯尼特群集收集的 Prometheus 指標位於/Prometheus 命名空間中。ContainerInsights從 Amazon ECS 叢集收集的 Prometheus 指標位於 ECS//Prometheus 命名空間中。ContainerInsights

主題

- App Mesh 的 Prometheus 指標
- NGINX 的 Prometheus 指標
- memcached 的 Prometheus 指標
- Java/JMX 的 Prometheus 指標
- HAProxy 的 Prometheus 指標

App Mesh 的 Prometheus 指標

系統會從 App Mesh 自動收集下列指標。

CloudWatch 容器見解也可以收集 App Mesh 特使存取記錄。如需詳細資訊，請參閱 (選擇性) 啟用 App Mesh Envoy 存取日誌。

Amazon EKS 和 Kubernetes 叢集上 App Mesh 的 Prometheus 指標

指標名稱	維度	
envoy_http_downstream_rq_total	ClusterName, Namespace	
envoy_http_downstream_rq_xx	ClusterName, Namespace ClusterName,, 域名管理器前綴Namespace ，程式碼類別	
envoy_cluster_upstream_cx_rx_bytes_total	ClusterName, Namespace	

指標名稱	維度	
envoy_cluster_upstream_cx_tx_bytes_total	ClusterName, Namespace	
envoy_cluster_membership_healthy	ClusterName, Namespace	
envoy_cluster_membership_total	ClusterName, Namespace	
envoy_server_memory_heap_size	ClusterName, Namespace	
envoy_server_memory_allocated	ClusterName, Namespace	
envoy_cluster_upstream_cx_connect_timeout	ClusterName, Namespace	
envoy_cluster_upstream_rq_pending_failure_eject	ClusterName, Namespace	

指標名稱	維度	
envoy_cluster_upstream_rq_pending_overflow	ClusterName, Namespace	
envoy_cluster_upstream_rq_timeout	ClusterName, Namespace	
envoy_cluster_upstream_rq_try_per_timeout	ClusterName, Namespace	
envoy_cluster_upstream_rq_rx_reset	ClusterName, Namespace	
envoy_cluster_upstream_cx_destroy_local_with_active_rq	ClusterName, Namespace	
envoy_cluster_upstream_cx_destroy_remote_active_rq	ClusterName, Namespace	

指標名稱	維度	
envoy_cluster_upstream_rq_maintenance_mode	ClusterName, Namespace	
envoy_cluster_upstream_flow_control_paused_reading_total	ClusterName, Namespace	
envoy_cluster_upstream_flow_control_resumed_reading_total	ClusterName, Namespace	
envoy_cluster_upstream_flow_control_backed_up_total	ClusterName, Namespace	
envoy_cluster_upstream_flow_control_drained_total	ClusterName, Namespace	

指標名稱	維度	
envoy_cluster_upstream_rq_retry	ClusterName, Namespace	
envoy_cluster_upstream_rq_retry_success	ClusterName, Namespace	
envoy_cluster_upstream_rq_retry_overflow	ClusterName, Namespace	
envoy_server_live	ClusterName, Namespace	
envoy_server_uptime	ClusterName, Namespace	

Amazon ECS 叢集上 App Mesh 的 Prometheus 指標

指標名稱	維度	
envoy_http_downstream_rq_total	ClusterName, TaskDefinitionFamily	
envoy_http_downstream_rq_xx	ClusterName, TaskDefinitionFamily	
envoy_cluster_upst	ClusterName, TaskDefinitionFamily	

指標名稱	維度	
ream_cx_r x_bytes_total		
envoy_clu ster_upst ream_cx_t x_bytes_total	ClusterName, TaskDefinitionFamily	
envoy_clu ster_memb ership_healthy	ClusterName, TaskDefinitionFamily	
envoy_clu ster_memb ership_total	ClusterName, TaskDefinitionFamily	
envoy_ser ver_memor y_heap_size	ClusterName, TaskDefinitionFamily	
envoy_ser ver_memor y_allocated	ClusterName, TaskDefinitionFamily	
envoy_clu ster_upst ream_cx_c onnect_timeout	ClusterName, TaskDefinitionFamily	
envoy_clu ster_upst ream_rq_p ending_fa ilure_eject	ClusterName, TaskDefinitionFamily	

指標名稱	維度	
envoy_cluster_upstream_rq_pending_overflow	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_rq_timeout	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_rq_try_per_timeout	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_rq_rx_reset	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_cx_destroy_local_with_active_rq	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_cx_destroy_remote_active_rq	ClusterName, TaskDefinitionFamily	

指標名稱	維度	
envoy_cluster_upstream_rq_maintenance_mode	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_flow_control_paused_reading_total	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_flow_control_resumed_reading_total	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_flow_control_backed_up_total	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_flow_control_drained_total	ClusterName, TaskDefinitionFamily	

指標名稱	維度	
envoy_cluster_upstream_rq_retry	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_rq_retry_success	ClusterName, TaskDefinitionFamily	
envoy_cluster_upstream_rq_retry_overflow	ClusterName, TaskDefinitionFamily	
envoy_server_live	ClusterName, TaskDefinitionFamily	
envoy_server_uptime	ClusterName, TaskDefinitionFamily	
envoy_http_downstream_rq_xx	ClusterName,, 域名管理器前綴 TaskDefinitionFamily, 程式碼類別 ClusterName,, 程 TaskDefinitionFamily式碼類別	

> **Note**
>
> TaskDefinitionFamily 是 mesh 的 Kubernetes 命名空間。envoy_http_conn_manager_prefix 的值可以是 ingress、egress 或 admin。envoy_response_code_class 的值可以是 1 (代表 1xx)、2 代表 (2xx)、3 (代表 3xx)、4 (代表 4xx) 或 5 (代表5xx)。

NGINX 的 Prometheus 指標

系統會 Amazon EKS 和 Kubernetes 叢集上的 NGINX 自動收集下列指標。

指標名稱	維度	
nginx_ingress_controller_nginx_process_cpu_seconds_total	ClusterName,Namespace ，服務	
nginx_ingress_controller_success	ClusterName,Namespace ，服務	
nginx_ingress_controller_requests	ClusterName,Namespace ，服務	
nginx_ingress_controller_nginx_process_connections	ClusterName,Namespace ，服務	
nginx_ingress_controller_nginx_process_connections_total	ClusterName,Namespace ，服務	
nginx_ingress_controller_ng	ClusterName,Namespace ，服務	

指標名稱	維度	
inx_proce ss_reside nt_memory _bytes		
nginx_ing ress_cont roller_co nfig_last _reload_s uccessful	ClusterName,Namespace ，服務	
nginx_ing ress_cont roller_re quests	ClusterName、Namespace 、服務、狀態	

memcached 的 Prometheus 指標

系統會 Amazon EKS 和 Kubernetes 叢集上的 Memcached 自動收集下列指標。

指標名稱	維度	
memcached _current_items	ClusterName,Namespace ，服務	
memcached _current_ connections	ClusterName,Namespace ，服務	
memcached _limit_bytes	ClusterName,Namespace ，服務	
memcached _current_bytes	ClusterName,Namespace ，服務	

指標名稱	維度
memcached_written_bytes_total	ClusterName,Namespace ，服務
memcached_read_bytes_total	ClusterName,Namespace ，服務
memcached_items_evicted_total	ClusterName,Namespace ，服務
memcached_items_reclaimed_total	ClusterName,Namespace ，服務
memcached_commands_total	ClusterName,Namespace ，服務 ClusterName、Namespace 、服務、指令 ClusterName、Namespace 、服務、狀態、指令

Java/JMX 的 Prometheus 指標

在 Amazon EKS 和 Kubernetes 叢集上收集的指標

在 Amazon EKS 和 Kubernetes 叢集上，Container Insights 可以使用 JMX Exporter，從 Java 虛擬機器 (JVM)、Java 和 Tomcat (Catalina) 收集下列預先定義的 Prometheus 指標。如需詳細資訊，請參閱 Github 上的 prometheus/jmx_exporter。

Amazon EKS 和 Kubernetes 叢集上的 Java/JMX

指標名稱	維度
jvm_classes_loaded	ClusterName ，Namespace

指標名稱	維度	
jvm_threads_current	ClusterName ,Namespace	
jvm_threads_daemon	ClusterName ,Namespace	
java_lang_operatingsystem_totalswapspacesize	ClusterName ,Namespace	
java_lang_operatingsystem_systemcpuload	ClusterName ,Namespace	
java_lang_operatingsystem_processcpuload	ClusterName ,Namespace	
java_lang_operatingsystem_freeswapspacesize	ClusterName ,Namespace	
java_lang_operatingsystem_totalphysicalmemorysize	ClusterName ,Namespace	

指標名稱	維度	
java_lang_operatingsystem_freephysicalmemorysize	ClusterName ，Namespace	
java_lang_operatingsystem_openfiledescriptorcount	ClusterName ，Namespace	
java_lang_operatingsystem_availableprocessors	ClusterName ，Namespace	
jvm_memory_bytes_used	ClusterName 、Namespace 、區域	
jvm_memory_pool_bytes_used	ClusterName 、Namespace 、集區	

> **Note**
>
> area 維度的數值可以是 heap 或 nonheap。
> pool 維度的數值可以是 Tenured Gen、Compress Class Space、Survivor Space、Eden Space、Code Cache 或 Metaspace。

Amazon EKS 和 Kubernetes 叢集上的 Tomcat/JMX

除了上表中的 Java/JMX 指標之外，也會收集 Tomcat 工作負載的下列指標。

指標名稱	維度	
catalina_manager_activesessions	ClusterName ,Namespace	
catalina_manager_rejectedsessions	ClusterName ,Namespace	
catalina_globalrequestprocessor_bytesreceived	ClusterName ,Namespace	
catalina_globalrequestprocessor_bytessent	ClusterName ,Namespace	
catalina_globalrequestprocessor_requestcount	ClusterName ,Namespace	
catalina_globalrequestprocessor_errorcount	ClusterName ,Namespace	
catalina_globalrequestproce	ClusterName ,Namespace	

指標名稱	維度	
ssor_proc essingtime		

Amazon ECS 叢集上的 Java/JMX

指標名稱	維度	
jvm_class es_loaded	ClusterName ，TaskDefinitionFamily	
jvm_threa ds_current	ClusterName ，TaskDefinitionFamily	
jvm_threa ds_daemon	ClusterName ，TaskDefinitionFamily	
java_lang _operatin gsystem_t otalswaps pacesize	ClusterName ，TaskDefinitionFamily	
java_lang _operatin gsystem_s ystemcpuload	ClusterName ，TaskDefinitionFamily	
java_lang _operatin gsystem_p rocesscpuload	ClusterName ，TaskDefinitionFamily	
java_lang _operatin gsystem_f	ClusterName ，TaskDefinitionFamily	

指標名稱	維度	
reeswapspacesize		
java_lang_operatingsystem_totalphysicalmemorysize	ClusterName , TaskDefinitionFamily	
java_lang_operatingsystem_freephysicalmemorysize	ClusterName , TaskDefinitionFamily	
java_lang_operatingsystem_openfiledescriptorcount	ClusterName , TaskDefinitionFamily	
java_lang_operatingsystem_availableprocessors	ClusterName , TaskDefinitionFamily	
jvm_memory_bytes_used	ClusterName , TaskDefinitionFamily, 面積	
jvm_memory_pool_bytes_used	ClusterName 水池 TaskDefinitionFamily, 水池	

> **Note**
>
> area 維度的數值可以是 heap 或 nonheap。
> pool 維度的數值可以是 Tenured Gen、Compress Class Space、Survivor Space、Eden Space、Code Cache 或 Metaspace。

Amazon ECS 叢集上的 Tomcat/JMX

除了上表中的 Java/JMX 指標之外，也會收集 Amazon ECS 叢集上的 Tomcat 工作負載的下列指標。

指標名稱	維度	
catalina_manager_activesessions	ClusterName，TaskDefinitionFamily	
catalina_manager_rejectedsessions	ClusterName，TaskDefinitionFamily	
catalina_globalrequestprocessor_bytesreceived	ClusterName，TaskDefinitionFamily	
catalina_globalrequestprocessor_bytessent	ClusterName，TaskDefinitionFamily	
catalina_globalrequestprocessor_requestcount	ClusterName，TaskDefinitionFamily	

指標名稱	維度	
catalina_globalrequestprocessor_errorcount	ClusterName ，TaskDefinitionFamily	
catalina_globalrequestprocessor_processingtime	ClusterName ，TaskDefinitionFamily	

HAProxy 的 Prometheus 指標

系統會從 Amazon EKS 和 Kubernetes 叢集 上的 HAProxy 自動收集下列指標。

收集的指標取決於您所使用的 HAProxy Ingress 的版本。如需 HAProxy Ingress 及其版本的詳細資訊，請參閱 haproxy-ingress。

指標名稱	維度	可用性
haproxy_backend_bytes_in_total	ClusterName 、Namespace 、服務	HAProxy Ingress 的所有版本
haproxy_backend_bytes_out_total	ClusterName 、Namespace 、服務	HAProxy Ingress 的所有版本
haproxy_backend_connection_errors_total	ClusterName 、Namespace 、服務	HAProxy Ingress 的所有版本
haproxy_backend_co	ClusterName 、Namespace 、服務	HAProxy Ingress 的所有版本

指標名稱	維度	可用性
nnections _total		
haproxy_b ackend_cu rrent_sessions	ClusterName 、Namespace 、服務	HAProxy Ingress 的所有版本
haproxy_b ackend_ht tp_respon ses_total	ClusterName 、Namespace 、服務、程式碼、後端	HAProxy Ingress 的所有版本
haproxy_b ackend_status	ClusterName 、Namespace 、服務	只有 HAProxy Ingress 的 0.10 版或更新版本
haproxy_b ackend_up	ClusterName 、Namespace 、服務	只有 HAProxy Ingress 的 0.10 版
haproxy_f rontend_b ytes_in_total	ClusterName 、Namespace 、服務	HAProxy Ingress 的所有版本
haproxy_f rontend_b ytes_out_total	ClusterName 、Namespace 、服務	HAProxy Ingress 的所有版本
haproxy_f rontend_c onnection s_total	ClusterName 、Namespace 、服務	HAProxy Ingress 的所有版本
haproxy_f rontend_c urrent_se ssions	ClusterName 、Namespace 、服務	HAProxy Ingress 的所有版本

指標名稱	維度	可用性
haproxy_frontend_http_requests_total	ClusterName 、Namespace 、服務	HAProxy Ingress 的所有版本
haproxy_frontend_http_responses_total	ClusterName 、Namespace 、服務、程式碼、前端	HAProxy Ingress 的所有版本
haproxy_frontend_request_errors_total	ClusterName 、Namespace 、服務	HAProxy Ingress 的所有版本
haproxy_frontend_requests_denied_total	ClusterName 、Namespace 、服務	HAProxy Ingress 的所有版本

Note

code 維度的數值可以是 1xx、2xx、3xx、4xx、5xx 或 other。
backend 維度的數值可以是：

- 適用於 HAProxy Ingress 0.0.27 版或更早版本的 http-default-backend、http-shared-backend 或 httpsback-shared-backend。
- 適用於 HAProxy Ingress 0.0.27 以上版本的 _default_backend。

frontend 維度的數值可以是：

- 適用於 HAProxy Ingress 0.0.27 版或更早版本的 httpfront-default-backend、httpfront-shared-frontend 或 httpfronts。
- 適用於 HAProxy Ingress 0.0.27 以上版本的 _front_http 或 _front_https。

檢視 Prometheus 指標

您可以對所有 Prometheus 指標進行監控和警示，包括來自 App Mesh、NGINX、JavaA/JMX、Memcached 和 HAProxy，以及您可能已新增之任何其他手動設定的 Prometheus 匯出工具等經策管的預先彙總指標。如需從其他 Prometheus 匯出工具收集指標的詳細資訊，請參閱新增 Prometheus 湊集目標的教學課程：Prometheus API 伺服器指標。

在 CloudWatch 主控台中，容器深入解析提供下列預先建置的報告：

- 對於 Amazon EKS 和 Kubernetes 叢集，有針對 App Mesh、NGINX、HAPROXY、Memcached 和 Java/JMX 預先建置的報告。
- 對於 Amazon ECS 叢集，有適用於 App Mesh 和 Java/JMX 的預先建置報告。

Container Insights 也會針對 Container Insights 從中收集經策管指標的每個工作負載提供自訂儀表板。您可以從下載這些儀表板 GitHub

查看所有 Prometheus 指標

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 指標。
3. 在命名空間清單中，選擇 ContainerInsights/Prometheus 或 ECS//Prometheus。ContainerInsights
4. 在下列清單中選擇其中一組維度。然後選取您要查看的指標旁邊的核取方塊。

查看有關 Prometheus 指標的預先建置報告

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Performance Monitoring (效能監控)。
3. 在靠近頁面頂端的下拉式方塊中，選擇任意一個 Prometheus 選項。

   在另一個下拉式方塊中，選擇要檢視的叢集

我們還為 NGINX、App Mesh、Memcached、HAProxy 和 Java/JMX 提供了自訂儀表板。

使用 Amazon 提供的自訂儀表板

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/

2. 在導覽窗格中，選擇 Dashboards (儀表板)。
3. 選擇 Create Dashboard (建立儀表板)。輸入新儀表板的名稱，然後選擇 Create dashboard (建立儀表板)。
4. 在 Add to this dashboard (新增至此儀表板) 中，選擇 Cancel (取消)。
5. 選擇 Actions (動作)、View/edit sources (檢視/編輯來源)。
6. 下載下列其中一個 JSON 檔案：
   - Github 上的 NGINX 自訂儀表板來源。
   - Github 上的 App Mesh 自訂儀表板來源。
   - Github 上的 Memcached 自訂儀表板來源
   - Github 上的 HAProxy-Ingress 自訂儀表板來源
   - Github 上的 Java/JMX 自訂儀表板來源。
7. 使用文字編輯器開啟您下載的 JSON 檔案，然後進行下列變更：
   - 將所有 {{YOUR_CLUSTER_NAME}} 字串取代為您叢集的確切名稱。確定不要在文字前後加空格。
   - 將所有字{{YOUR_REGION}}串取代為執行叢集的 AWS 區域。例如，**us-west-1** 確定不要在文字前後加空格。
   - 將所有 {{YOUR_NAMESPACE}} 字串取代為您工作負載的確切命名空間。
   - 將所有 {{YOUR_SERVICE_NAME}} 字串取代為您工作負載的確切服務名稱。例如 **haproxy-haproxy-ingress-controller-metrics**
8. 複製整個 JSON blob 並將其粘貼到 CloudWatch 控制台的文本框中，替換框中已經存在的內容。
9. 選擇 Update (更新)、Save dashboard (儲存儀表板)。

Prometheus 指標故障診斷

本節提供對 Prometheus 指標設定進行故障診斷的說明。

主題

- Amazon ECS 上的 Prometheus 指標故障診斷
- Amazon EKS 和 Kubernetes 叢集上的 Prometheus 指標故障診斷

Amazon ECS 上的 Prometheus 指標故障診斷

本節提供對 Amazon ECS 叢集上的 Prometheus 指標設定進行故障診斷的說明。

我沒有看到 Prometheus 指標發送到日誌 CloudWatch

Prometheus 指標應在日誌群組 /aws/ecs/containerinsights/cluster-name/Prometheus 中擷取為日誌事件。如果未建立日誌群組或 Prometheus 測量結果未傳送至日誌群組，您必須先檢查代理程式是否已成功找到 Prometheus 目標。 CloudWatch 接下來檢查 CloudWatch代理程式的安全群組和權限設定。下列步驟會引導您執行偵錯。

步驟 1：啟用 CloudWatch 代理程式偵錯模式

首先，通過將以下粗體行添加到模 AWS CloudFormation 板文件中，將 CloudWatch 代理程序更改為調試模式，`cwagent-ecs-prometheus-metric-for-bridge-host.yaml`或`cwagent-ecs-prometheus-metric-for-awsvpc.yaml`. 接著儲存檔案。

```
cwagentconfig.json: |
    {
      "agent": {
        "debug": true
      },
      "logs": {
        "metrics_collected": {
```

針對現有堆疊建立新的 AWS CloudFormation 變更集。將變更集中的其他參數設定為與現有 AWS CloudFormation 堆疊中相同的值。下列範例適用於使用 EC2 啟動類型和橋接網路模式在 Amazon ECS 叢集中安裝的 CloudWatch 代理程式。

```
ECS_NETWORK_MODE=bridge
 CREATE_IAM_ROLES=True
ECS_TASK_ROLE_NAME=your_selected_ecs_task_role_name
ECS_EXECUTION_ROLE_NAME=your_selected_ecs_execution_role_name
NEW_CHANGESET_NAME=your_selected_ecs_execution_role_name

aws cloudformation create-change-set --stack-name CWAgent-Prometheus-ECS-
${ECS_CLUSTER_NAME}-EC2-${ECS_NETWORK_MODE} \
    --template-body file://cwagent-ecs-prometheus-metric-for-bridge-host.yaml \
    --parameters ParameterKey=ECSClusterName,ParameterValue=$ECS_CLUSTER_NAME \
                 ParameterKey=CreateIAMRoles,ParameterValue=$CREATE_IAM_ROLES \
                 ParameterKey=ECSNetworkMode,ParameterValue=$ECS_NETWORK_MODE \
                 ParameterKey=TaskRoleName,ParameterValue=$ECS_TASK_ROLE_NAME \
```

```
                ParameterKey=ExecutionRoleName,ParameterValue=$ECS_EXECUTION_ROLE_NAME
 \
    --capabilities CAPABILITY_NAMED_IAM \
    --region $AWS_REGION \
    --change-set-name $NEW_CHANGESET_NAME
```

移至主 AWS CloudFormation 控台以檢閱新的變更集。$NEW_CHANGESET_NAME應該有一個變更套用至 CW AgentConfig SSM 參數資源。執行變更集，然後輸入下列命令來重新啟動 CloudWatch 代理程式工作。

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 0 \
--service your_service_name_here \
--region $AWS_REGION
```

請等候約 10 秒鐘，然後輸入下列命令。

```
aws ecs update-service --cluster $ECS_CLUSTER_NAME \
--desired-count 1 \
--service your_service_name_here \
--region $AWS_REGION
```

步驟 2：檢查 ECS 服務探索日誌

根據預設， CloudWatch 代理程式的 ECS 任務定義會在下一節中啟用記錄。記錄檔會傳送至 CloudWatch 記錄群組 /ec ecs-cwagent-prometheus s/ 中的記錄檔。

```
LogConfiguration:
  LogDriver: awslogs
    Options:
      awslogs-create-group: 'True'
      awslogs-group: "/ecs/ecs-cwagent-prometheus"
      awslogs-region: !Ref AWS::Region
      awslogs-stream-prefix: !Sub 'ecs-${ECSLaunchType}-awsvpc'
```

依字串 ECS_SD_Stats 篩選日誌，以取得與 ECS 服務探索相關的指標，如以下範例所示。

```
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_DescribeContainerInstances: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_DescribeInstancesRequest: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_DescribeTaskDefinition: 2
```

```
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_DescribeTasks: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: AWSCLI_ListTasks: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: Exporter_DiscoveredTargetCount: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: LRUCache_Get_EC2MetaData: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: LRUCache_Get_TaskDefinition: 2
2020-09-1T01:53:14Z D! ECS_SD_Stats: LRUCache_Size_ContainerInstance: 1
2020-09-1T01:53:14Z D! ECS_SD_Stats: LRUCache_Size_TaskDefinition: 2
2020-09-1T01:53:14Z D! ECS_SD_Stats: Latency: 43.399783ms
```

特定 ECS 服務探索週期的每個指標的含義如下：

- AWSCLI_DescribeContainerInstances— 進行的 `ECS::DescribeContainerInstances` API 呼叫次數。
- AWSCLI_DescribeInstancesRequest— 進行的 `ECS::DescribeInstancesRequest` API 呼叫次數。
- AWSCLI_DescribeTaskDefinition— 進行的 `ECS::DescribeTaskDefinition` API 呼叫次數。
- AWSCLI_DescribeTasks— 進行的 `ECS::DescribeTasks` API 呼叫次數。
- AWSCLI_ListTasks— 進行的 `ECS::ListTasks` API 呼叫次數。
- ExporterDiscoveredTargetCount— 發現並成功匯出至容器內目標結果檔案的 Prometheus 目標數目。
- Lrucache_get_EC2 MetaData — 從快取擷取容器執行個體中繼資料的次數。
- Lrucache_get_ TaskDefinition — 從快取擷取 ECS 任務定義中繼資料的次數。
- Lrucache_size_ ContainerInstance — 快取記憶體中唯一容器執行個體中繼資料的數目。
- 「大小 _」TaskDefinition — 快取記憶體中的唯一 ECS 任務定義數目。
- Latency (延遲) – 服務探索週期所需的時間。

檢查 `ExporterDiscoveredTargetCount` 的數值，查看已搜索到 Prometheus 目標是否符合您的期望。如果沒有，可能的原因如下：

- ECS 服務探索的組態可能不符合應用程式的設定。對於以 docker 標籤為基礎的服務探索，您的目標容器可能沒有在 CloudWatch 代理程式中設定必要的 docker 標籤來 auto 探索它們。對於 ECS 任務定義 ARN 一般運算式式服務探索， CloudWatch 代理程式中的 regex 設定可能與應用程式的任務定義不符。
- CloudWatch 代理程式的 ECS 任務角色可能沒有擷取 ECS 任務中繼資料的權限。檢查 CloudWatch 代理程式是否已獲得下列唯讀權限：
  - `ec2:DescribeInstances`

- ecs:ListTasks
- ecs:DescribeContainerInstances
- ecs:DescribeTasks
- ecs:DescribeTaskDefinition

步驟 3：檢查網路連線和 ECS 任務角色政策

如果仍然沒有傳送至目標 CloudWatch 記錄檔日誌群組的記錄事件，即使的值Exporter_DiscoveredTargetCount表示探索到 Prometheus 目標，這可能是下列其中一個原因所造成：

- CloudWatch 代理程式可能無法連線到 Prometheus 目標連接埠。檢查 CloudWatch 代理程式後面的安全群組設定。私有 IP 應允許 CloudWatch 代理程式連線到 Prometheus 匯出器連接埠。
- CloudWatch 代理程式的 ECS 工作角色可能沒有CloudWatchAgentServerPolicy受管理的原則。CloudWatch 代理程式的 ECS 任務角色必須具有此原則，才能將 Prometheus 度量作為記錄事件傳送。如果您使用範例 AWS CloudFormation 範本自動建立 IAM 角色，ECS 任務角色和 ECS 執行角色都會被授與執行 Prometheus 監控的最低權限。

Amazon EKS 和 Kubernetes 叢集上的 Prometheus 指標故障診斷

本節提供對 Amazon EKS 和 Kubernetes 叢集上的 Prometheus 指標設定進行故障診斷的說明。

Amazon EKS 上的一般故障診斷步驟

若要確認 CloudWatch 代理程式正在執行，請輸入下列命令。

```
kubectl get pod -n amazon-cloudwatch
```

輸出應該在 NAME 欄中包含 cwagent-prometheus-*id* 一行和 STATUS column. 中包含 Running 一行

若要顯示有關執行中 pod 的詳細資訊，請輸入下列命令。將 *pod-name* 取代為 pod 的完整名稱，即以 cw-agent-prometheus 為開頭的名稱。

```
kubectl describe pod pod-name -n amazon-cloudwatch
```

如果您已安裝 CloudWatch 容器深入解析，則可以使用 CloudWatch 日誌深入解析來查詢 CloudWatch 代理程式收集 Prometheus 指標的記錄。

查詢應用程式日誌

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在功能窗格中，選擇 [CloudWatch 記錄檔見解]。
3. 選取應用程式日誌的日誌群組，/aws/containerinsights/***cluster-name***/application
4. 以下列查詢取代搜尋查詢表達式，然後選擇 Run query (執行查詢)

```
fields ispresent(kubernetes.pod_name) as haskubernetes_pod_name, stream,
 kubernetes.pod_name, log |
filter haskubernetes_pod_name and kubernetes.pod_name like /cwagent-prometheus
```

您也可以確認 Prometheus 指標和中繼資料已擷取為記錄事件。 CloudWatch

確認 Prometheus 資料正擷取中

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在功能窗格中，選擇 [CloudWatch 記錄檔見解]。
3. 選擇 /aws/containerinsights/***cluster-name***/prometheus
4. 以下列查詢取代搜尋查詢表達式，然後選擇 Run query (執行查詢)

```
fields @timestamp, @message | sort @timestamp desc | limit 20
```

記錄日誌已捨棄 Prometheus 指標

此版本不會收集長條圖類型的 Prometheus 指標。您可以使用 CloudWatch 代理程式來檢查是否有任何 Prometheus 測量結果因為它們是長條圖度量而被卸除。您也可以記錄前 500 個 Prometheus 度量的清單，這些量度會被捨棄且未傳送至， CloudWatch 因為它們是長條圖度量。

若要查看是否有任何指標遭到捨棄，請輸入下列命令：

```
kubectl logs -l "app=cwagent-prometheus" -n amazon-cloudwatch --tail=-1
```

如果有任何指標遭到捨棄，您會在 `/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log` 檔案中看到以下各行。

```
I! Drop Prometheus metrics with unsupported types. Only Gauge, Counter and Summary are
 supported.
```

```
I! Please enable CWAgent debug mode to view the first 500 dropped metrics
```

如果您看到這些行，而且想要知道哪些指標遭到捨棄，請使用下列步驟。

記錄捨棄的 Prometheus 指標量清單

1. 將下列粗體行新增至您的`prometheus-eks.yaml`或`prometheus-k8s.yaml`檔案，將 CloudWatch 代理程式變更為偵錯模式，然後儲存檔案。

   ```
   {
         "agent": {
           "debug": true
         },
   ```

   檔案的這個區段看起來應該像這樣：

   ```
   cwagentconfig.json: |
       {
         "agent": {
           "debug": true
         },
         "logs": {
           "metrics_collected": {
   ```

2. 輸入下列命令，重新安裝 CloudWatch 代理程式以啟動除錯模式：

   ```
   kubectl delete deployment cwagent-prometheus -n amazon-cloudwatch
   kubectl apply -f prometheus.yaml
   ```

   捨棄的量度會記錄在 CloudWatch 代理程式網繭中。

3. 若要從 CloudWatch 代理程式網繭擷取記錄，請輸入下列命令：

   ```
   kubectl logs -l "app=cwagent-prometheus" -n amazon-cloudwatch --tail=-1
   ```

   ```
   ############ Fluentd ############## CloudWatch ### /aws/##/##_name/##
   ####
   ```

   若要查詢這些日誌，您可以遵循Amazon EKS 上的一般故障診斷步驟中的步驟查詢應用程式日誌。

Prometheus 指標在哪裡擷取為 CloudWatch 日誌記錄事件？

該 CloudWatch 代理為每個 Prometheus 抓取作業配置創建一個日誌流。例如，在 `prometheus-eks.yaml` 和 `prometheus-k8s.yaml` 檔案中，`job_name: 'kubernetes-pod-appmesh-envoy'` 一行會湊集 App Mesh 指標。Prometheus 目標會定義為 `kubernetes-pod-appmesh-envoy`。因此，所有 App Mesh Prometheus 度量都會擷取為 CloudWatch 記錄檔資料流中的記錄檔事件，名稱為 /AWS/容器洞察/群集名稱/Prometheus 的記錄檔群組kubernetes-pod-appmesh-envoy下方。

我在指標中看不到 Amazon EKS 或 Prometheus 指標 CloudWatch

首先，請確定 Prometheus 指標在日誌群組 /aws/containerinsights/cluster-name/Prometheus 中擷取為日誌事件。使用 Prometheus 指標在哪裡擷取為 CloudWatch 日誌記錄事件？ 中的資訊來協助您檢查目標日誌串流。如果未建立日誌串流，或是日誌串流中沒有新的日誌事件，請檢查下列項目：

- 檢查 Prometheus 指標匯出工具端點是否已正確設定
- 檢查 CloudWatch 代理程式 YAML 檔案`config map: cwagent-prometheus`區段中的 Prometheus 抓取設定是否正確。該組態應該與 Prometheus 組態檔中的組態相同。如需詳細資訊，請參閱 Prometheus 文件中的 <scrape_config>。

如果 Prometheus 量度已正確擷取為記錄事件，請檢查內嵌度量格式設定是否已新增至記錄事件中，以產生指標。 CloudWatch

```
"CloudWatchMetrics":[
   {
      "Metrics":[
         {
            "Name":"envoy_http_downstream_cx_destroy_remote_active_rq"
         }
      ],
      "Dimensions":[
         [
            "ClusterName",
            "Namespace"
         ]
      ],
      "Namespace":"ContainerInsights/Prometheus"
   }
],
```

如需內嵌指標格式的詳細資訊，請參閱規格：內嵌指標格式 。

如果記錄事件中沒有內嵌的度量格式，請檢查 CloudWatch 代理程式安裝 YAML 檔案`config map: prometheus-cwagentconfig`區段中的區段是否正確設定區段。`metric_declaration`如需詳細資訊，請參閱 新增 Prometheus 湊集目標的教學課程：Prometheus API 伺服器指標。

與 Application Insights 整合

Amazon CloudWatch 應用程式深入解析可協助您監控應用程式，並在應用程式資源和技術堆疊中識別和設定關鍵指標、日誌和警示。如需詳細資訊，請參閱 Amazon CloudWatch 應用洞察。

您可以啟用 Application Insights，從您的容器化應用程式和微型服務收集額外資料。如果尚未這樣做，您可以透過選擇 Container Insights 儀表板中效能檢視下方的 Auto-configure Application Insights (自動設定 Application Insights) 來啟用。

如果您已設定應用 CloudWatch 程式深入解析來監控您的容器化應用程式，「應用程式深入解析」儀表板會顯示在「容器見解」儀表板下方

如需 Application Insights 和容器化應用程式的詳細資訊，請參閱 啟用 Application Insights 進行 Amazon ECS 和 Amazon EKS 資源監控。

在 Container Insights 中查看 Amazon ECS 生命週期事件

您可以在 Container Insights 主控台中檢視 Amazon ECS 生命週期事件。這有助於您在單一檢視中關聯容器指標、日誌和事件，以便更全面地了解相關操作。

事件包括容器執行個體狀態變更事件、任務狀態變更事件和服務動作事件。它們會由 Amazon ECS 自動傳送至 Amazon EventBridge ，也會以事件日誌格式收集。 CloudWatch 如需有關這些事件的詳細資訊，請參閱 Amazon ECS 事件。

標準容器洞見定價適用於 Amazon ECS 生命週期事件。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

若要設定生命週期事件資料表並為叢集建立規則，您必須具有 `events:PutRule`、`events:PutTargets` 以及 `logs:CreateLogGroup` 許可。您還必須確保有一個資源策略可 EventBridge 以創建日誌流並將日誌發送到 CloudWatch 日誌。如果此資源政策不存在，您可輸入下列命令來建立該政策：

```
aws --region region logs put-resource-policy --policy-name 'EventBridgeCloudWatchLogs'
 --policy-document '{
    "Statement": [
```

```
        {
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": ["events.amazonaws.com", "delivery.logs.amazonaws.com"]
            },
            "Resource": "arn:aws:logs:region:account-id:log-group:/aws/events/ecs/
containerinsights/*:*",
            "Sid": "TrustEventBridgeToStoreECSLifecycleLogEvents"
        }
    ],
    "Version": "2012-10-17"
}'
```

您可使用下列命令來檢查是否已建立此政策，並確認附接的政策是否正常運作。

```
aws logs describe-resource-policies --region region --output json
```

若要檢視生命週期事件資料表，您必須具有 `events:DescribeRule`、`events:ListTargetsByRule` 以及 `logs:DescribeLogGroups` 許可。

在 CloudWatch 容器洞見主控台中檢視 Amazon ECS 生命週期事件

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 依次選擇 Insights、Container Insights。
3. 選擇檢視效能儀表板。
4. 在下一個下拉式方塊中，選擇 ECS Clusters (ECS 叢集)、ECS Services (ECS 服務) 或 ECS Tasks (ECS 任務)。
5. 如果您在上一步中選擇 ECS Services (ECS 服務) 或 ECS Tasks (ECS 任務)，請選擇 Lifecycle events (生命週期事件) 索引標籤。
6. 在頁面底部，如果您看到「設定生命週期事件」，請選擇該事件以建立叢集的 EventBridge 規則。

   這些事件會顯示在 Container Insights 窗格下方以及 Application Insights 區段上方。若要對這些事件執行額外的分析並建立其他視覺化效果，請在 Lifecycle Events (生命週期事件) 資料表中選擇 View in Logs Insights (在 Logs Insights 中檢視)。

針對容器洞見進行故障診斷

若您在使用容器洞見時發生問題，下列各節可協助您。

在 Amazon EKS 或 Kubernetes 上部署失敗

若代理程式無法正確地在 Kubernetes 叢集上進行部署，請嘗試以下作業：

- 執行以下命令來取得 Pod 清單。

```
kubectl get pods -n amazon-cloudwatch
```

- 執行以下命令和檢查輸出底部的事件。

```
kubectl describe pod pod-name -n amazon-cloudwatch
```

- 執行以下命令來檢查日誌。

```
kubectl logs pod-name -n amazon-cloudwatch
```

未經授權的緊急事件：無法從 kubelet 擷取 cadvisor 資料

如果部署失敗，並出現錯誤 `Unauthorized panic: Cannot retrieve cadvisor data from kubelet`，您的 kubelet 可能還沒有啟用 Webhook 授權模式。容器洞見需要使用此模式。如需詳細資訊，請參閱 確認 先決條件。

在 Amazon ECS 上已刪除和重新建立的叢集上部署 Container Insights

若您刪除未啟用 Container Insights 的現有 Amazon ECS 叢集，且您使用相同的名稱將其重新建立，則您無法在重新建立此新叢集時在其上啟用 Container Insights。您可以透過重新建立它，然後輸入以下命令來進行啟用：

```
aws ecs update-cluster-settings --cluster myCICluster --settings
 name=containerInsights,value=enabled
```

無效端點錯誤

如果您看到類似以下的錯誤訊息，請檢查確認您已將您使用的命令中的所有預留位置 (例如 *cluster-name* 和 *region-name*) 取代為部署專用的正確資訊。

```
"log": "2020-04-02T08:36:16Z E! cloudwatchlogs: code: InvalidEndpointURL, message:
 invalid endpoint uri, original error: &url.Error{Op:\"parse\", URL:\"https://
logs.{{region_name}}.amazonaws.com/\", Err:\"{\"}, &awserr.baseError{code:
\"InvalidEndpointURL\", message:\"invalid endpoint uri\", errs:[]error{(*url.Error)
(0xc0008723c0)}}\n",
```

指標沒有出現在主控台

如果您在中沒有看到任何容器見解指標 AWS Management Console，請確定您已完成容器深入解析的設定。在完整設定容器洞見前指標都不會出現。如需詳細資訊，請參閱 設定 Container Insights。

升級叢集後，Amazon EKS 或 Kubernetes 上的 Pod 指標遺失

如果在新的或升級的叢集上將 CloudWatch 代理程式部署為精靈集後遺失所有或部分網繭度量，或者您看到包含訊息的錯誤記錄，則此區段可能會很有用。`W! No pod metric collected`

這些錯誤可能是由容器執行時間中的變更造成的，例如 containerd 或 Docker systemd cgroup 驅動程式。您通常可以透過更新部署資訊清單來解決這個問題，以便從主機的 containerd 套接字掛載到容器中。請參閱下列範例：

```
# For full example see https://github.com/aws-samples/amazon-cloudwatch-container-
insights/blob/latest/k8s-deployment-manifest-templates/deployment-mode/daemonset/
container-insights-monitoring/cwagent/cwagent-daemonset.yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: cloudwatch-agent
  namespace: amazon-cloudwatch
spec:
  template:
    spec:
      containers:
        - name: cloudwatch-agent
# ...
          # Don't change the mountPath
          volumeMounts:
# ...
            - name: dockersock
              mountPath: /var/run/docker.sock
              readOnly: true
            - name: varlibdocker
              mountPath: /var/lib/docker
```

```
              readOnly: true
            - name: containerdsock # NEW mount
              mountPath: /run/containerd/containerd.sock
              readOnly: true
# ...
      volumes:
# ...
        - name: dockersock
          hostPath:
            path: /var/run/docker.sock
        - name: varlibdocker
          hostPath:
            path: /var/lib/docker
        - name: containerdsock # NEW volume
          hostPath:
            path: /run/containerd/containerd.sock
```

使用 Bottlerocket for Amazon EKS 時，沒有 Pod 指標

Bottlerocket 是以 Linux 為基礎的開源作業系統，由 AWS 為特定用途而建置，用於執行容器。

Bottlerocket 使用主機上的不同 `containerd` 路徑，因此您需要將磁碟區變更為其位置。如果不提供，您會在日誌中看到錯誤，其中包括 `W! No pod metric collected`。請參閱以下範例。

```
volumes:
  # ...
    - name: containerdsock
      hostPath:
        # path: /run/containerd/containerd.sock
        # bottlerocket does not mount containerd sock at normal place
        # https://github.com/bottlerocket-os/bottlerocket/
commit/91810c85b83ff4c3660b496e243ef8b55df0973b
        path: /run/dockershim.sock
```

使用 Amazon EKS 或 Kubernetes 的 containerd 執行時間時，沒有容器檔案系統指標

這是已知問題，社群貢獻者正在處理。如需詳細資訊，請參閱 cadvisor 不支援在 containerd 上使用 containerd 的磁碟使用量度量和容器檔案系統度量。 GitHub

收集 Prometheus 指標時， CloudWatch 代理程式的記錄檔磁碟區意外增加

這是在代理程式的版本 1.247347.6b250880 中引入的迴歸。 CloudWatch 此迴歸功能已在代理程式的更新版本中予以修正。它的影響僅限於客戶收集 CloudWatch 代理程式本身的記錄檔，同時也使用

Prometheus 的案例。如需詳細資訊，請參閱 [prometheus] 代理程式正在登入時列印所有抓取的指標。 GitHub

在 Dockerhub 中找不到版本備註中提到的最新 Docker 影像

在我們內部開始實際發佈之前，我們在 Github 上更新發佈說明和標籤。在 Github 上增加版本編號後，通常需要 1-2 週的時間才能在登錄檔上查看最新的 Docker 影像。 CloudWatch 代理程式容器映像沒有每晚發行版本。您可以在以下位置直接從源代碼構建圖像：https://github.com/aws/ amazon-cloudwatch-agent /樹/主/ amazon-cloudwatch-container-insights cloudwatch-agent-dockerfile

CrashLoopBackoff CloudWatch 代理程式上的錯誤

如果您看到 CloudWatch 代理程式的`CrashLoopBackOff`錯誤，請確定您的 IAM 許可設定正確。如需詳細資訊，請參閱 確認 先決條件。

CloudWatch 代理程式或 Fluentd 網繭卡在擱置中

如果您的 CloudWatch 代理程式或 Fluentd Pod 卡住`Pending`或出現`FailedScheduling`錯誤，請根據代理程式所需的核心數量和 RAM 數量來判斷節點是否有足夠的運算資源。輸入以下命令來描述 Pod：

```
kubectl describe pod cloudwatch-agent-85ppg -n amazon-cloudwatch
```

建立您自己的 CloudWatch 代理碼頭形象

您可以通過參考碼頭文件位於 https://github.com/aws-samples/ amazon-cloudwatch-container-insights /blob /最新/ 碼頭文件構建自己的 CloudWatch 代理碼頭圖像。cloudwatch-agent-dockerfile

Dockerfile 支援直接使用 `docker buildx` 建置多架構映像。

在容器中部署其他 CloudWatch 代理程式功能

您可以使用 CloudWatch 代理程式在容器中部署其他監視功能。重要功能如下所示：

- 內嵌指標格式— 如需詳細資訊，請參閱 在日誌中內嵌指標。
- StatsD— 如需詳細資訊，請參閱 使用 StatsD 擷取自訂指標 。

說明和必要的檔案位 GitHub 於下列位置：

- 如需 Amazon ECS 容器，請參閱根據部署模式的範例 Amazon ECS 任務定義。

- 如需 Amazon EKS 和 Kubernetes 容器，請參閱根據部署模式的 Kubernetes YAML 檔案範例。

Lambda Insights

CloudWatch Lambda 洞察是一種監控和故障排除解決方案，適用於在上 AWS Lambda執行的無伺服此解決方案會收集、彙總和摘要系統層級的指標，包括 CPU 時間、記憶體、磁碟和網路。它也會收集、彙總和摘要診斷資訊，例如冷啟動和 Lambda 工作人員關閉，協助您隔離 Lambda 函數問題並快速加以解決。

Lambda 洞察使用新的 CloudWatch Lambda 擴充功能，該擴充功能是以 Lambda 層形式提供的 當您在 Lambda 函數上安裝此延伸模組時，它會收集系統層級指標，並針對該 Lambda 函數的每次叫用發出單一效能記錄事件。 CloudWatch 使用內嵌指標格式從記錄事件擷取指標。

如需 Lambda 擴充的詳細資訊，請參閱使用 AWS Lambda 擴充功能。如需內嵌指標格式的詳細資訊，請參閱在日誌中內嵌指標。

您可以使用 Lambda Insights 搭配任何 Lambda 函數，其中該函數使用了可支援 Lambda 延伸的 Lambda 執行時間。如需這些執行時間的清單，請參閱 Lambda Extensions API。

定價

對於為 Lambda Insights 啟用的每個 Lambda 函數，您只需為指標和日誌的使用量付費。如需定價範例，請參閱 Amazon CloudWatch 定價。

您需支付 Lambda 延伸所耗用的執行時間 (以 1 毫秒為單位遞增)。如需 Lambda 定價的詳細資訊，請參閱 AWS Lambda 定價。

開始使用 Lambda Insights

若要在 Lambda 函數上啟用 Lambda Insights，您可以在 Lambda 主控台中使用一鍵切換。或者，您可以 AWS CLI使用、 AWS Serverless Application Model CLI 或 AWS Cloud Development Kit (AWS CDK). AWS CloudFormation

以下各節提供完成這些步驟的詳細說明。

主題

- Lambda Insights 延伸的可用版本
- 使用主控台在現有的 Lambda 函數上啟用 Lambda Insights
- 使用在 AWS CLI 現有 Lambda 函數上啟用 Lambda 見解

- 使用 AWS SAM CLI 對現有 Lambda 函數啟用 Lambda 深入解析
- 用於啟 AWS CloudFormation 用對現有 Lambda 函數的 Lambda 深入解析
- 使用在 AWS CDK 現有 Lambda 函數上啟用 Lambda 見解
- 使用無伺服器架構在現有的 Lambda 函數上啟用 Lambda Insights
- 在 Lambda 容器映像部署上啟用 Lambda Insights

Lambda Insights 延伸的可用版本

本節列出 Lambda 見解擴充功能的版本，以及每個 AWS 區域中用於這些擴充功能的 ARN。

主題

- x86-64 平台
- ARM64 平台

x86-64 平台

本節列出 x86-64 平台的 Lambda 見解延伸模組版本，以及用於每個區域中這些擴充功能的 ARN。AWS

> **Important**
>
> Lambda 見解擴充功能 1.0.317.0 及更新版本不支援 Amazon Linux 1。

1.0.317.0

版本 1.0.317.0 包括刪除對 Amazon Linux 1 平台的支持和錯誤修復。它還包括對 AWS GovCloud (US) 區域的支持。

適用於版本 1.0.317.0 的 ARN

下表列出了在每個可用 AWS 區域中用於此版本擴充功能的 ARN。

區域	ARN
美國東部 (維吉尼亞北部)	`arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:52`

區域	ARN
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:52
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:52
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:52
非洲 (開普敦)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:43
亞太區域 (香港)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:43
亞太區域 (海德拉巴)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension:25
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension:29
亞太區域 (墨爾本)	arn:aws:lambda:ap-southeast-4:158895979263:layer:LambdaInsightsExtension:20
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:50
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:33
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:51
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:52
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:52

區域	ARN
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:79
加拿大 (中部)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:51
加拿大西部 (卡加利)	arn:aws:lambda:ca-west-1:946466191631:layer:LambdaInsightsExtension:12
中國 (北京)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:42
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:42
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:52
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:52
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:52
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:43
Europe (Paris)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:51
歐洲 (西班牙)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension:27
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:49
歐洲 (蘇黎世)	arn:aws:lambda:eu-central-2:033019950311:layer:LambdaInsightsExtension:26

區域	ARN
以色列 (特拉維夫)	arn:aws:lambda:il-central-1:459530977127:layer:LambdaInsightsExtension:20
Middle East (Bahrain)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:43
中東 (阿拉伯聯合大公國)	arn:aws:lambda:me-central-1:732604637566:layer:LambdaInsightsExtension:26
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:51
AWS GovCloud (美國東部)	arn:aws-us-gov:lambda:us-gov-east-1:122132214140:layer:LambdaInsightsExtension:19
AWS GovCloud (美國西部)	arn:aws-us-gov:lambda:us-gov-west-1:751350123760:layer:LambdaInsightsExtension:19

1.0.295.0

版本 1.0.295.0 包含所有相容執行階段的相依性更新。

適用於版本 1.0.295.0 的 ARN

下表列出了在每個可用 AWS 區域中用於此版本擴充功能的 ARN。

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:51
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:51
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:51

區域	ARN
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:51
非洲 (開普敦)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:42
亞太區域 (香港)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:42
亞太區域 (海德拉巴)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension:24
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension:28
亞太區域 (墨爾本)	arn:aws:lambda:ap-southeast-4:158895979263:layer:LambdaInsightsExtension:19
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:49
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:32
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:50
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:51
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:51
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:78
加拿大 (中部)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:50

區域	ARN
加拿大西部 (卡加利)	arn:aws:lambda:ca-west-1:946466191631:layer:LambdaInsightsExtension:11
中國 (北京)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:41
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:41
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:51
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:51
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:51
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:42
Europe (Paris)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:50
歐洲 (西班牙)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension:26
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:48
歐洲 (蘇黎世)	arn:aws:lambda:eu-central-2:033019950311:layer:LambdaInsightsExtension:25
以色列 (特拉維夫)	arn:aws:lambda:il-central-1:459530977127:layer:LambdaInsightsExtension:19
Middle East (Bahrain)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:42

區域	ARN
中東 (阿拉伯聯合大公國)	arn:aws:lambda:me-central-1:732604637566:layer:LambdaInsightsExtension:25
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:50

1.0.275.0

版本 1.0.275.0 包含所有相容執行階段的重要相依性更新。

適用於版本 1.0.275.0 的 ARN

下表列出了在每個可用 AWS 區域中用於此版本擴充功能的 ARN。

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:49
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:49
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:49
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:49
非洲 (開普敦)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:40
亞太區域 (香港)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:40
亞太區域 (海德拉巴)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension:22

區域	ARN
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension:26
亞太區域 (墨爾本)	arn:aws:lambda:ap-southeast-4:158895979263:layer:LambdaInsightsExtension:17
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:47
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:30
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:48
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:49
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:49
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:76
加拿大 (中部)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:48
加拿大西部 (卡加利)	arn:aws:lambda:ca-west-1:946466191631:layer:LambdaInsightsExtension:9
中國 (北京)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:39
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:39
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:49

區域	ARN
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:49
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:49
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:40
Europe (Paris)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:48
歐洲 (西班牙)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension:24
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:46
歐洲 (蘇黎世)	arn:aws:lambda:eu-central-2:033019950311:layer:LambdaInsightsExtension:23
以色列 (特拉維夫)	arn:aws:lambda:il-central-1:459530977127:layer:LambdaInsightsExtension:17
Middle East (Bahrain)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:40
中東 (阿拉伯聯合大公國)	arn:aws:lambda:me-central-1:732604637566:layer:LambdaInsightsExtension:23
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:48

1.0.273.0

版本 1.0.273.0 包括所有兼容運行時的重要錯誤修復，並增加了對加拿大西部（卡爾加里）的支持。

適用於 1.0.273.0 版本的 ARN

下表列出了在每個可用 AWS 區域中用於此版本擴充功能的 ARN。

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:45
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:45
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:45
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:45
非洲 (開普敦)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:35
亞太區域 (香港)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:35
亞太區域 (海德拉巴)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension:17
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension:21
亞太區域 (墨爾本)	arn:aws:lambda:ap-southeast-4:158895979263:layer:LambdaInsightsExtension:12
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:43
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:26
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:44

區域	ARN
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:45
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:45
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:72
加拿大 (中部)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:44
加拿大西部 (卡加利)	arn:aws:lambda:ca-west-1:946466191631:layer:LambdaInsightsExtension:4
中國 (北京)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:36
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:36
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:45
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:45
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:45
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:35
Europe (Paris)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:44
歐洲 (西班牙)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension:19

區域	ARN
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:42
歐洲 (蘇黎世)	arn:aws:lambda:eu-central-2:033019950311:layer:LambdaInsightsExtension:17
以色列 (特拉維夫)	arn:aws:lambda:il-central-1:459530977127:layer:LambdaInsightsExtension:12
Middle East (Bahrain)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:35
中東 (阿拉伯聯合大公國)	arn:aws:lambda:me-central-1:732604637566:layer:LambdaInsightsExtension:18
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:44

1.0.229.0

1.0.229.0 版包括所有相容執行期的重要錯誤修正，並新增對以色列 (特拉維夫) 區域的支援。

1.0.229.0 版的 ARN

下表列出了在每個可用 AWS 區域中用於此版本擴充功能的 ARN。

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:38
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:38
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:38

區域	ARN
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:38
非洲 (開普敦)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:28
亞太區域 (香港)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:28
亞太區域 (海德拉巴)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension:10
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension:14
亞太區域 (墨爾本)	arn:aws:lambda:ap-southeast-4:158895979263:layer:LambdaInsightsExtension:5
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:36
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:19
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:37
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:38
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:38
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:60
加拿大 (中部)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:37

區域	ARN
中國 (北京)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:29
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:29
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:38
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:38
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:38
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:28
Europe (Paris)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:37
歐洲 (西班牙)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension:12
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:35
歐洲 (蘇黎世)	arn:aws:lambda:eu-central-2:033019950311:layer:LambdaInsightsExtension:11
以色列 (特拉維夫)	arn:aws:lambda:il-central-1:459530977127:layer:LambdaInsightsExtension:5
Middle East (Bahrain)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:28
中東 (阿拉伯聯合大公國)	arn:aws:lambda:me-central-1:732604637566:layer:LambdaInsightsExtension:11

區域	ARN
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:37

1.0.178.0

版本 1.0.178.0 增加了對以下區域的支持。 AWS

- 亞太區域 (海德拉巴)
- 亞太區域 (雅加達)
- 歐洲 (西班牙)
- 歐洲 (蘇黎世)
- 中東 (阿拉伯聯合大公國)

1.0.178.0 版的 ARN

下表列出了在每個可用 AWS 區域中用於此版本擴充功能的 ARN。

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:35
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:33
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:33
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:33
非洲 (開普敦)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:25

區域	ARN
亞太區域 (香港)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:25
亞太區域 (海德拉巴)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension:8
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension:11
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:31
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:2
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:32
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:33
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:33
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:50
加拿大 (中部)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:32
中國 (北京)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:26
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:26
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:35

區域	ARN
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:33
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:33
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:25
Europe (Paris)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:32
歐洲 (西班牙)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension:10
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:30
歐洲 (蘇黎世)	arn:aws:lambda:eu-central-2:033019950311:layer:LambdaInsightsExtension:7
Middle East (Bahrain)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:25
中東 (阿拉伯聯合大公國)	arn:aws:lambda:me-central-1:732604637566:layer:LambdaInsightsExtension:9
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:32

1.0.143.0

1.0.143.0 版修正了與 Python 3.7 和 Go 1.x 的相容性錯誤。Python 3.6 Lambda 執行階段已被棄用。如需詳細資訊，請參閱 Lambda 執行階段。

1.0.143.0 版的 ARN

下表列出了在每個可用 AWS 區域中用於此版本擴充功能的 ARN。

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:21
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:21
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:20
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:21
非洲 (開普敦)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:13
亞太區域 (香港)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:13
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:21
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:2
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:20
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:21
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:21
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:32

區域	ARN
加拿大 (中部)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:20
中國 (北京)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:14
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:14
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:21
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:21
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:21
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:13
歐洲 (巴黎)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:20
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:20
中東 (巴林)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:13
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:20

1.0.135.0

1.0.135.0 版包含 Lambda Insights 如何收集和報告磁碟和檔案描述項使用情況的錯誤修復。在之前版本的延伸中，tmp_free 指標報告了運行函數時 /tmp 目錄的最大可用空間。此版本改為將指標變更

為報告最小值，使其在評估磁碟使用情況時更有用。如需有關 tmp 目錄儲存配額的詳細資訊，請參 Lambda 配額。

1.0.135.0 版現在還報告檔案描述項使用情況 (fd_use 和 fd_max) 作為跨程序的最大值，而不是報告作業系統層級。

適用於 1.0.135.0 版的 ARN

下表列出了在每個可用 AWS 區域中用於此版本擴充功能的 ARN。

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:18
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:18
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:18
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:18
非洲 (開普敦)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:11
亞太區域 (香港)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:11
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:18
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension:1
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:18

區域	ARN
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:18
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:18
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:25
加拿大 (中部)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:18
中國 (北京)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:11
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:11
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:18
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:18
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:18
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:11
歐洲 (巴黎)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:18
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:18
中東 (巴林)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:11

區域	ARN
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:18

1.0.119.0

適用於版本 1.0.119.0 的 ARN

下表列出了在每個可用 AWS 區域中用於此版本擴充功能的 ARN。

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:16
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:16
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:16
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:16
非洲 (開普敦)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:9
亞太區域 (香港)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:9
Asia Pacific (Mumbai)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:16
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:16
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:16

區域	ARN
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:16
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:23
加拿大 (中部)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:16
中國 (北京)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:9
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:9
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:16
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:16
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:16
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:9
歐洲 (巴黎)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:16
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:16
中東 (巴林)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:9
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:16

1.0.98.0

此版本會移除不必要的日誌記錄，同時還解決了 AWS Serverless Application Model CLI 本機呼叫的問題。如需有關此問題的詳細資訊，請參閱使用 「sam 本機叫用」新增逾時 LambdaInsightsExtension 結果。

1.0.98.0 版的 ARN

下表列出了在每個可用 AWS 區域中用於此版本擴充功能的 ARN。

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:14
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:14
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:14
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:14
非洲 (開普敦)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension:8
亞太區域 (香港)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension:8
Asia Pacific (Mumbai)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:14
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:14
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:14

區域	ARN
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:14
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:14
加拿大 (中部)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:14
中國 (北京)	arn:aws-cn:lambda:cn-north-1:488211338238:layer:LambdaInsightsExtension:8
中國 (寧夏)	arn:aws-cn:lambda:cn-northwest-1:488211338238:layer:LambdaInsightsExtension:8
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:14
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:14
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:14
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension:8
歐洲 (巴黎)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:14
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:14
中東 (巴林)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension:8
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:14

1.0.89.0

此版本修正了效能事件時間戳記，以始終表示函數叫用的開始。

1.0.89.0 版的 ARN

下表列出了在每個可用 AWS 區域中用於此版本擴充功能的 ARN。

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:12
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:12
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:12
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:12
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:12
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:12
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:12
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:12
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:12
加拿大 (中部)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:12

區域	ARN
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:12
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:12
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:12
歐洲 (巴黎)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:12
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:12
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:12

1.0.86.0

在 1.0.54.0 版中，記憶體指標有時報告不正確，有時候高於 100%。1.0.86.0 版使用與 Lambda 平台指標相同的事件資料，藉此修正記憶體測量問題。這表示您可能會看到記錄的記憶體指標值出現顯著變更。這可藉由使用新的 Lambda Logs API 來予以達成。這可以更準確地測量 Lambda 沙盒記憶體用量。不過，需要注意的是，如果函數沙盒逾時且隨後減速，Lambda Logs API 就無法傳遞平台報告事件。在此案例中。Lambda Insights 無法記錄叫用指標。如需 Lambda Logs API 的詳細資訊，請參閱 AWS Lambda Logs API。

1.0.86.0 版中的新功能

- 使用 Lambda Logs API 來糾正記憶體指標。這就解決了之前的記憶體統計資料大於 100% 的問題。
- `Init Duration`作為新量 CloudWatch 度引入。
- 使用叫用 ARN 為別名和叫用版本新增 version 維度。如果您使用 Lambda 別名或版本來達成增量改進部署 (例如藍綠色部署)，則可以根據叫用的別名檢視指標。如果函數未使用別名或版本，則不會套用 version 維度。如需詳細資訊，請參閱 Lambda 函數別名。
- 將 `billed_mb_ms field` 新增至效能事件，以顯示每次叫用的成本。這不會考慮與佈建並行相關聯的任何成本。

- 將 billed_duration 和 duration 欄位新增至效能事件。

1.0.86.0 版的 ARN

下表列出了在每個可用 AWS 區域中用於此版本擴充功能的 ARN。

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:11
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:11
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:11
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:11
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:11
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:11
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:11
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:11
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:11
加拿大 (中部)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:11

區域	ARN
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:11
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:11
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:11
歐洲 (巴黎)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:11
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:11
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:11

1.0.54.0

1.0.54.0 版是 Lambda Insights 延伸的初始發行版本。

1.0.54.0 版的 ARN

下表列出了在每個可用 AWS 區域中用於此版本擴充功能的 ARN。

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension:2
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension:2
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:2

區域	ARN
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension:2
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension:2
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension:2
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension:2
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension:2
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension:2
加拿大 (中部)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension:2
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension:2
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension:2
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension:2
歐洲 (巴黎)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension:2
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension:2
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension:2

ARM64 平台

本節列出適用於 ARM64 平台的 Lambda 見解延伸模組版本，以及用於每個 AWS 區域中這些擴充功能的 ARN。

> **Important**
>
> Lambda 見解擴充功能 1.0.317.0 及更新版本不支援 Amazon Linux 1。

1.0.317.0

版本 1.0.317.0 包括刪除對 Amazon Linux 1 平台的支持和錯誤修復。它還包括對 AWS GovCloud (US) 區域的支持。

區域	ARN
美國東部 (維吉尼亞北部)	`arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:19`
美國東部 (俄亥俄)	`arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:21`
美國西部 (加利佛尼亞北部)	`arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:17`
美國西部 (奧勒岡)	`arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:19`
非洲 (開普敦)	`arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension-Arm64:17`
亞太區域 (香港)	`arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension-Arm64:17`
亞太區域 (海德拉巴)	`arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension-Arm64:5`
亞太區域 (雅加達)	`arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension-Arm64:17`

區域	ARN
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:21
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension-Arm64:16
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:18
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:19
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:19
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:30
加拿大 (中部)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:17
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:19
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:19
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:19
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension-Arm64:17
Europe (Paris)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension-Arm64:17
歐洲 (西班牙)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension-Arm64:5

區域	ARN
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension-Arm64:17
中東 (巴林)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension-Arm64:17
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:17
AWS GovCloud (美國東部)	arn:aws-us-gov:lambda:us-gov-east-1:122132214140:layer:LambdaInsightsExtension-Arm64:1
AWS GovCloud (美國西部)	arn:aws-us-gov:lambda:us-gov-west-1:751350123760:layer:LambdaInsightsExtension-Arm64:1

1.0.295.0

版本 1.0.295.0 包含所有相容執行階段的相依性更新。

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:18
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:20
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:16
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:18
非洲 (開普敦)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension-Arm64:16

區域	ARN
亞太區域 (香港)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension-Arm64:16
亞太區域 (海德拉巴)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension-Arm64:4
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension-Arm64:16
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:20
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension-Arm64:15
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:17
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:18
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:18
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:29
加拿大 (中部)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:16
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:18
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:18
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:18

區域	ARN
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension-Arm64:16
Europe (Paris)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension-Arm64:16
歐洲 (西班牙)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension-Arm64:4
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension-Arm64:16
中東 (巴林)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension-Arm64:16
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:16

1.0.275.0

版本 1.0.275.0 包括所有兼容運行時的錯誤修復以及對歐洲（西班牙）和亞太地區（海德拉巴）區域的支持。

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:16
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:18
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:14
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:16

區域	ARN
非洲 (開普敦)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension-Arm64:14
亞太區域 (香港)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension-Arm64:14
亞太區域 (海德拉巴)	arn:aws:lambda:ap-south-2:891564319516:layer:LambdaInsightsExtension-Arm64:2
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension-Arm64:14
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:18
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension-Arm64:13
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:15
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:16
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:16
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:27
加拿大 (中部)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:14
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:16
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:16

區域	ARN
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:16
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension-Arm64:14
Europe (Paris)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension-Arm64:14
歐洲 (西班牙)	arn:aws:lambda:eu-south-2:352183217350:layer:LambdaInsightsExtension-Arm64:2
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension-Arm64:14
中東 (巴林)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension-Arm64:14
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:14

1.0.273.0

版本 1.0.273.0 包括所有兼容運行時的錯誤修復。

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:12
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:14
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:9

區域	ARN
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:12
非洲 (開普敦)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension-Arm64:9
Asia Pacific (Hong Kong)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension-Arm64:9
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension-Arm64:9
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:14
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension-Arm64:9
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:11
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:12
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:12
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:23
加拿大 (中部)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:10
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:12
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:12

區域	ARN
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:12
歐洲 (米蘭)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension-Arm64:9
歐洲 (巴黎)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension-Arm64:10
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension-Arm64:10
中東 (巴林)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension-Arm64:9
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:10

1.0.229.0

1.0.229.0 版包括所有相容執行期的錯誤修正。同時新增對下列區域的支援：

- 美國西部 (加利佛尼亞北部)
- 非洲 (開普敦)
- Asia Pacific (Hong Kong)
- 亞太區域 (雅加達)
- 亞太區域 (大阪)
- 亞太區域 (首爾)
- 加拿大 (中部)
- 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)

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:5
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:7
美國西部 (加利佛尼亞北部)	arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:3
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:5
非洲 (開普敦)	arn:aws:lambda:af-south-1:012438385374:layer:LambdaInsightsExtension-Arm64:2
Asia Pacific (Hong Kong)	arn:aws:lambda:ap-east-1:519774774795:layer:LambdaInsightsExtension-Arm64:2
亞太區域 (雅加達)	arn:aws:lambda:ap-southeast-3:439286490199:layer:LambdaInsightsExtension-Arm64:2
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:7
亞太區域 (大阪)	arn:aws:lambda:ap-northeast-3:194566237122:layer:LambdaInsightsExtension-Arm64:2
亞太區域 (首爾)	arn:aws:lambda:ap-northeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:4
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:5
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:5

區域	ARN
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:11
加拿大 (中部)	arn:aws:lambda:ca-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:3
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:5
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:5
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:5
歐洲 (西班牙)	arn:aws:lambda:eu-south-1:339249233099:layer:LambdaInsightsExtension-Arm64:2
Europe (Paris)	arn:aws:lambda:eu-west-3:580247275435:layer:LambdaInsightsExtension-Arm64:3
歐洲 (斯德哥爾摩)	arn:aws:lambda:eu-north-1:580247275435:layer:LambdaInsightsExtension-Arm64:3
中東 (巴林)	arn:aws:lambda:me-south-1:285320876703:layer:LambdaInsightsExtension-Arm64:2
南美洲 (聖保羅)	arn:aws:lambda:sa-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:3

1.0.135.0

1.0.135.0 版包含 Lambda Insights 如何收集和報告磁碟和檔案描述項使用情況的錯誤修復。在之前版本的延伸中，`tmp_free` 指標報告了運行函數時 `/tmp` 目錄的最大可用空間。此版本改為將指標變更為報告最小值，使其在評估磁碟使用情況時更有用。如需有關 `tmp` 目錄儲存配額的詳細資訊，請參 Lambda 配額。

1.0.135.0 版現在還報告檔案描述項使用情況 (fd_use 和 fd_max) 作為跨程序的最大值，而不是報告作業系統層級。

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:2
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:2
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:2
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:2
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:2
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:2
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:2
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:2
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:2
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:2

1.0.119.0

區域	ARN
美國東部 (維吉尼亞北部)	arn:aws:lambda:us-east-1:580247275435:layer:LambdaInsightsExtension-Arm64:1
美國東部 (俄亥俄)	arn:aws:lambda:us-east-2:580247275435:layer:LambdaInsightsExtension-Arm64:1
美國西部 (奧勒岡)	arn:aws:lambda:us-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:1
亞太區域 (孟買)	arn:aws:lambda:ap-south-1:580247275435:layer:LambdaInsightsExtension-Arm64:1
亞太區域 (新加坡)	arn:aws:lambda:ap-southeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:1
亞太區域 (雪梨)	arn:aws:lambda:ap-southeast-2:580247275435:layer:LambdaInsightsExtension-Arm64:1
亞太區域 (東京)	arn:aws:lambda:ap-northeast-1:580247275435:layer:LambdaInsightsExtension-Arm64:1
歐洲 (法蘭克福)	arn:aws:lambda:eu-central-1:580247275435:layer:LambdaInsightsExtension-Arm64:1
歐洲 (愛爾蘭)	arn:aws:lambda:eu-west-1:580247275435:layer:LambdaInsightsExtension-Arm64:1
歐洲 (倫敦)	arn:aws:lambda:eu-west-2:580247275435:layer:LambdaInsightsExtension-Arm64:1

使用主控台在現有的 Lambda 函數上啟用 Lambda Insights

在 Lambda 主控台中遵循這些步驟，以在現有的 Lambda 函數上啟用 Lambda Insights。

若要在 Lambda 函數上啟用 Lambda Insights

1. 開啟主 AWS Lambda 控台，網址為 https://console.aws.amazon.com/lambda/。
2. 選擇函數的名稱，然後選取接下來畫面上的 Configuration (組態) 索引標籤。
3. 在 [組態] 索引標籤下，選擇左側導覽功能表中的 [監視和作業工具]，然後選擇 [編輯]。

   您將被導向到可以編輯監控工具的畫面。
4. 透過 Lambda 洞察增強型監控，選擇編輯。
5. 在 CloudWatch Lambda 洞見下，啟用增強型監控，然後選擇儲存。

使用在 AWS CLI 現有 Lambda 函數上啟用 Lambda 見解

請依照下列步驟使用啟 AWS CLI 用有關現有 Lambda 函數的 Lambda 深入解析。

步驟 1：更新函數許可

若要更新函數的許可

- 輸入以下命令，將CloudWatchLambdaInsightsExecutionRolePolicy受管 IAM 政策附加到函數的執行角色。

```
aws iam attach-role-policy \
--role-name function-execution-role \
--policy-arn "arn:aws:iam::aws:policy/CloudWatchLambdaInsightsExecutionRolePolicy"
```

步驟 2：安裝 Lambda 延伸模組

輸入下列命令，安裝 Lambda 延伸。將 `layers` 參數的 ARN 值取代為與您的區域和您要使用的延伸相符的 ARN。如需詳細資訊，請參閱 Lambda Insights 延伸的可用版本。

```
aws lambda update-function-configuration \
   --function-name function-name \
   --layers "arn:aws:lambda:us-west-1:580247275435:layer:LambdaInsightsExtension:14"
```

步驟 3：啟用記 CloudWatch 錄 VPC 端點

只有在沒有網際網路存取權的私有子網路中執行的功能，以及尚未設定 CloudWatch 記錄虛擬私有雲端 (VPC) 端點時，才需要執行此步驟。

如果您需要執行此步驟，請輸入下列指令，以 VPC 的資訊取代預留位置。

如需詳細資訊，請參閱搭配介面 VPC 端點使用 CloudWatch 記錄檔。

```
aws ec2 create-vpc-endpoint \
--vpc-id vpcId \
--vpc-endpoint-type Interface \
--service-name com.amazonaws.region.logs \
--subnet-id subnetId
--security-group-id securitygroupId
```

使用 AWS SAM CLI 對現有 Lambda 函數啟用 Lambda 深入解析

請依照下列步驟使用啟 AWS SAM AWS CLI 用有關現有 Lambda 函數的 Lambda 深入解析。

如果您尚未安裝最新版本的 AWS SAM CLI，則必須先安裝或升級它。如需詳細資訊，請參閱安裝 AWS SAM CLI。

步驟 1：安裝層

若要讓 Lambda Insights 延伸可供您所有的 Lambda 函數使用，請使用 Lambda Insights 層的 ARN 將 `Layers` 屬性新增至 SAM 範本的 `Globals` 區段。下面的範例使用 Lambda Insights 的初始版本層。如需 Lambda Insights 延伸層的最新版本，請參閱 Lambda Insights 延伸的可用版本。

```
Globals:
  Function:
    Layers:
      - !Sub "arn:aws:lambda:
${AWS::Region}:580247275435:layer:LambdaInsightsExtension:14"
```

若要僅針對單一函數啟用此層，請將 `Layers` 屬性新增至函數，如此範例所示。

```
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      Layers:
        - !Sub "arn:aws:lambda:
${AWS::Region}:580247275435:layer:LambdaInsightsExtension:14"
```

步驟 2：新增受管政策

針對每個功能新增 CloudWatchLambdaInsightsExecutionRolePolicyIAM 政策。

AWS SAM 不支援全域原則，因此您必須個別在每個功能上啟用這些原則，如此範例所示。如需全域的詳細資訊，請參閱全域區段。

```
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      Policies:
        - CloudWatchLambdaInsightsExecutionRolePolicy
```

本機叫用

AWS SAM CLI 支援 Lambda 擴充功能。不過，每個本機執行的叫用都會重設執行階段環境。本地叫用將無法獲得 Lambda Insights 資料，因為執行時間會在沒有關機事件的情況下重新啟動。如需詳細資訊，請參閱版本 1.6.0-新增對 AWS Lambda 擴充功能本機測試的支援。

疑難排解

若要對您的 Lambda Insights 安裝進行疑難排解，請將下列環境變數新增至 Lambda 函數，以啟用偵錯日誌記錄。

```
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      Environment:
        Variables:
          LAMBDA_INSIGHTS_LOG_LEVEL: info
```

用於啟 AWS CloudFormation 用對現有 Lambda 函數的 Lambda 深入解析

請遵循下列步驟，以 AWS CloudFormation 針對現有 Lambda 函數啟用 Lambda 深入解析。

步驟 1：安裝層

將 Lambda Insights 層新增至 Lambda Insights 層 ARN 內的 `Layers` 屬性。下面的範例使用 Lambda Insights 的初始版本層。如需 Lambda Insights 延伸層的最新版本，請參閱 Lambda Insights 延伸的可用版本。

```
Resources:
  MyFunction:
    Type: AWS::Lambda::Function
    Properties:
      Layers:
        - !Sub "arn:aws:lambda:
${AWS::Region}:580247275435:layer:LambdaInsightsExtension:14"
```

步驟 2：新增受管政策

將 CloudWatchLambdaInsightsExecutionRolePolicyIAM 政策新增至您的函數執行角色。

```
Resources:
  MyFunctionExecutionRole:
    Type: 'AWS::IAM::Role'
    Properties:
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/CloudWatchLambdaInsightsExecutionRolePolicy'
```

步驟 3：(選用) 新增 VPC 端點

只有在沒有網際網路存取權的私有子網路中執行的功能，以及尚未設定 CloudWatch 記錄虛擬私有雲端 (VPC) 端點時，才需要執行此步驟。如需詳細資訊，請參閱搭配介面 VPC 端點使用 CloudWatch 記錄檔。

```
Resources:
  CloudWatchLogsVpcPrivateEndpoint:
    Type: AWS::EC2::VPCEndpoint
    Properties:
      PrivateDnsEnabled: 'true'
      VpcEndpointType: Interface
      VpcId: !Ref: VPC
      ServiceName: !Sub com.amazonaws.${AWS::Region}.logs
      SecurityGroupIds:
        - !Ref InterfaceVpcEndpointSecurityGroup
      SubnetIds:
        - !Ref PublicSubnet01
        - !Ref PublicSubnet02
        - !Ref PublicSubnet03
```

使用在 AWS CDK 現有 Lambda 函數上啟用 Lambda 見解

請依照下列步驟使用啟 AWS CDK 用有關現有 Lambda 函數的 Lambda 深入解析。若要使用這些步驟，您必須已使用 AWS CDK 來管理資源。

本節中的指令位於中 TypeScript。

首先，更新函數許可。

```
executionRole.addManagedPolicy(
 ManagedPolicy.fromAwsManagedPolicyName('CloudWatchLambdaInsightsExecutionRolePolicy')
);
```

接下來，在 Lambda 函數上安裝延伸。將 `layerArn` 參數的 ARN 值取代為與您的區域和您要使用的延伸相符的 ARN。如需詳細資訊，請參閱 Lambda Insights 延伸的可用版本。

```
import lambda = require('@aws-cdk/aws-lambda');
const layerArn = 'arn:aws:lambda:us-
west-1:580247275435:layer:LambdaInsightsExtension:14';
const layer = lambda.LayerVersion.fromLayerVersionArn(this, 'LayerFromArn', layerArn);
```

如有必要，請啟用 CloudWatch 記錄檔的虛擬私人雲端 (VPC) 端點。只有在沒有網際網路存取權的私有子網路中執行的功能，以及尚未設定 CloudWatch Logs VPC 端點時，才需要執行此步驟。

```
const cloudWatchLogsEndpoint = vpc.addInterfaceEndpoint('cwl-gateway', {
  service: InterfaceVpcEndpointAwsService.CLOUDWATCH_LOGS,
});

cloudWatchLogsEndpoint.connections.allowDefaultPortFromAnyIpv4();
```

使用無伺服器架構在現有的 Lambda 函數上啟用 Lambda Insights

使用無伺服器架構在現有的 Lambda 函數上啟用 Lambda Insights 如需無伺服器架構的詳細資訊，請參閱 serverless.com。

這是透過無伺服器的 Lambda Insights 外掛程式完成的。如需詳細資訊，請參閱serverless-plugin-lambda-insights。

如果您尚未安裝最新版本的無伺服器命令列介面，您必須先進行安裝或升級。如需詳細資訊，請參閱開始使用無伺服器架構開放原始碼 & AWS.

若要使用無伺服器架構在現有的 Lambda 函數上啟用 Lambda Insights 函數

1. 在無伺服器目錄中執行下列命令，以安裝 Lambda Insights 的無伺服器外掛程式：

```
npm install --save-dev serverless-plugin-lambda-insights
```

2. 在您的 `serverless.yml` 檔案中，將外掛程式新增至 `plugins` 區段，如下所示：

```
provider:
  name: aws
plugins:
  - serverless-plugin-lambda-insights
```

3. 啟用 Lambda Insights。
   - 您可以將下列屬性新增至無伺服器 .yml 檔案，個別啟用每個函數的 Lambda Insights

```
functions:
  myLambdaFunction:
    handler: src/app/index.handler
    lambdaInsights: true #enables Lambda Insights for this function
```

   - 您可以新增以下自訂區段，從而為 `serverless.yml` 檔案內的所有函數啟用 Lambda Insights：

```
custom:
  lambdaInsights:
    defaultLambdaInsights: true #enables Lambda Insights for all functions
```

4. 輸入下列命令，重新部署無伺服器服務：

```
serverless deploy
```

這會重新部署所有函數，並針對您指定的那些函數啟用 Lambda Insights。
其會新增 Lambda Insights 層並使用 `arn:aws:iam::aws:policy/CloudWatchLambdaInsightsExecutionRolePolicy` IAM 政策連接必要的許可，進而啟用 Lambda Insights。

在 Lambda 容器映像部署上啟用 Lambda Insights

若要在部署為容器映像的 Lambda 函數上啟用 Lambda Insights，請在您的 Dockerfile 中新增行。這些行會將 Lambda Insights 代理程式安裝為容器映像中的延伸。x86-64 容器和 ARM64 容器要新增的行不同。

> **Note**
>
> 僅在使用 Amazon Linux 2 的 Lambda 執行時間上支援 Lambda Insights 代理程式。

主題

- x86-64 容器映像部署
- ARM64 容器映像部署

x86-64 容器映像部署

若要在部署為在 x86-64 容器上執行之容器映像的 Lambda 函數上，啟用 Lambda Insights，請在您的 Dockerfile 中新增下列行。這些行會將 Lambda Insights 代理程式安裝為容器映像中的延伸。

```
RUN curl -O https://lambda-insights-extension.s3-ap-northeast-1.amazonaws.com/
amazon_linux/lambda-insights-extension.rpm && \
    rpm -U lambda-insights-extension.rpm && \
    rm -f lambda-insights-extension.rpm
```

建立 Lambda 函數後，將 CloudWatchLambdaInsightsExecutionRolePolicyIAM 政策指派給函數的執行角色，然後在容器映像型 Lambda 函數上啟用 Lambda 深入解析。

> **Note**
>
> 若要使用較舊版本的 Lambda Insights 延伸，請將先前命令中的 URL 取代為此 URL：`https://lambda-insights-extension.s3-ap-northeast-1.amazonaws.com/amazon_linux/lambda-insights-extension.1.0.111.0.rpm`。目前僅提供 Lambda Insights 1.0.111.0 版及更新版本。如需詳細資訊，請參閱 Lambda Insights 延伸的可用版本。

若要在 Linux 伺服器上驗證 Lambda Insights 代理程式套件的簽章

1. 輸入下列命令，以下載公有金鑰。

```
shell$ wget https://lambda-insights-extension.s3-ap-northeast-1.amazonaws.com/
lambda-insights-extension.gpg
```

2. 輸入下列命令，將公有金鑰匯入至您的 keyring。

```
shell$ gpg --import lambda-insights-extension.gpg
```

輸出會類似下列內容：請記下 key 值，在後續步驟您將會用到它。在此範例輸出中，索引鍵值為 848ABDC8。

```
gpg: key 848ABDC8: public key "Amazon Lambda Insights Extension" imported
gpg: Total number processed: 1
gpg: imported: 1  (RSA: 1)
```

3. 執行下列命令，驗證指紋。將 key-value 取代為上述步驟中的索引鍵值。

```
shell$  gpg --fingerprint key-value
```

在此命令的輸出中，指紋字串應為 E0AF FA11 FFF3 5BD7 349E E222 479C 97A1 848A BDC8。若字串不相符，請勿安裝代理程式並聯絡 AWS。

4. 在您驗證指紋之後，即可使用它來驗證 Lambda Insights 代理程式套件。輸入以下命令，下載套件簽章檔案。

```
shell$  wget https://lambda-insights-extension.s3-ap-northeast-1.amazonaws.com/
amazon_linux/lambda-insights-extension.rpm.sig
```

5. 執行下列命令，驗證簽章：

```
shell$ gpg --verify lambda-insights-extension.rpm.sig lambda-insights-extension.rpm
```

輸出應與以下類似：

```
gpg: Signature made Thu 08 Apr 2021 06:41:00 PM UTC using RSA key ID 848ABDC8
gpg: Good signature from "Amazon Lambda Insights Extension"
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: E0AF FA11 FFF3 5BD7 349E  E222 479C 97A1 848A BDC8
```

在預期的輸出中，可能會有關信任簽章的警告。只有您或您信任者所簽章的金鑰才能信任。這不表示該簽章是無效的，只是您尚未驗證該公有金鑰。

如果輸出包含 BAD `signature`，檢查您是否已正確執行步驟。如果您繼續收到BAD `signature`回應，請聯繫 AWS 並避免使用下載的文件。

x86-64 範例

本節包含在容器映像型 Python Lambda 函數上啟用 Lambda Insights 的範例。

在 Lambda 容器映像上啟用 Lambda Insights 的範例

1. 建立類似如下的 DDockerfile：

```
FROM public.ecr.aws/lambda/python:3.8

// extra lines to install the agent here
RUN curl -O https://lambda-insights-extension.s3-ap-northeast-1.amazonaws.com/
amazon_linux/lambda-insights-extension.rpm && \
    rpm -U lambda-insights-extension.rpm && \
    rm -f lambda-insights-extension.rpm

COPY index.py ${LAMBDA_TASK_ROOT}
CMD [ "index.handler" ]
```

2. 建立一個類似如下的名為 `index.py` 的 Python 檔案：

```
def handler(event, context):
  return {
    'message': 'Hello World!'
  }
```

3. 將 Dockerfile 和 `index.py` 放在同一個目錄中。然後，在該目錄中，執行以下步驟，從而建置 Docker 影像並將其上傳到 Amazon ECR。

```
// create an ECR repository
aws ecr create-repository --repository-name test-repository
// build the docker image
```

```
docker build -t test-image .
// sign in to AWS
aws ecr get-login-password | docker login --username AWS --password-stdin
 "${ACCOUNT_ID}".dkr.ecr."${REGION}".amazonaws.com
// tag the image
docker tag test-image:latest "${ACCOUNT_ID}".dkr.ecr."${REGION}".amazonaws.com/
test-repository:latest
// push the image to ECR
docker push "${ACCOUNT_ID}".dkr.ecr."${REGION}".amazonaws.com/test-
repository:latest
```

4. 使用您剛才建立的 Amazon ECR 映像來建立 Lambda 函數。
5. 將 CloudWatchLambdaInsightsExecutionRolePolicyIAM 政策指派給函數的執行角色。

ARM64 容器映像部署

若要在部署為在 AL2_aarch64 容器 (使用 ARM64 架構) 上執行之容器映像的 Lambda 函數上，啟用 Lambda Insights，請在您的 Dockerfile 中新增下列行。這些行會將 Lambda Insights 代理程式安裝為容器映像中的延伸。

```
RUN curl -O https://lambda-insights-extension-arm64.s3-ap-northeast-1.amazonaws.com/
amazon_linux/lambda-insights-extension-arm64.rpm && \
    rpm -U lambda-insights-extension-arm64.rpm && \
    rm -f lambda-insights-extension-arm64.rpm
```

建立 Lambda 函數後，將 CloudWatchLambdaInsightsExecutionRolePolicyIAM 政策指派給函數的執行角色，然後在容器映像型 Lambda 函數上啟用 Lambda 深入解析。

> **Note**
>
> 若要使用較舊版本的 Lambda Insights 延伸，請將先前命令中的 URL 取代為此 URL：`https://lambda-insights-extension-arm64.s3-ap-northeast-1.amazonaws.com/amazon_linux/lambda-insights-extension-arm64.1.0.229.0.rpm`。目前僅提供 Lambda Insights 1.0.229.0 版及更新版本。如需詳細資訊，請參閱 Lambda Insights 延伸的可用版本。

若要在 Linux 伺服器上驗證 Lambda Insights 代理程式套件的簽章

1. 輸入下列命令，以下載公有金鑰。

```
shell$ wget https://lambda-insights-extension-arm64.s3-ap-
northeast-1.amazonaws.com/lambda-insights-extension.gpg
```

2. 輸入下列命令，將公有金鑰匯入至您的 keyring。

```
shell$ gpg --import lambda-insights-extension.gpg
```

輸出會類似下列內容：請記下 key 值，在後續步驟您將會用到它。在此範例輸出中，索引鍵值為 848ABDC8。

```
gpg: key 848ABDC8: public key "Amazon Lambda Insights Extension" imported
gpg: Total number processed: 1
gpg: imported: 1  (RSA: 1)
```

3. 執行下列命令，驗證指紋。將 key-value 取代為上述步驟中的索引鍵值。

```
shell$  gpg --fingerprint key-value
```

在此命令的輸出中，指紋字串應為 E0AF FA11 FFF3 5BD7 349E E222 479C 97A1 848A BDC8。若字串不相符，請勿安裝代理程式並聯絡 AWS。

4. 在您驗證指紋之後，即可使用它來驗證 Lambda Insights 代理程式套件。輸入以下命令，下載套件簽章檔案。

```
shell$  wget https://lambda-insights-extension-arm64.s3-ap-
northeast-1.amazonaws.com/amazon_linux/lambda-insights-extension-arm64.rpm.sig
```

5. 執行下列命令，驗證簽章：

```
shell$ gpg --verify lambda-insights-extension-arm64.rpm.sig lambda-insights-
extension-arm64.rpm
```

輸出應與以下類似：

```
gpg: Signature made Thu 08 Apr 2021 06:41:00 PM UTC using RSA key ID 848ABDC8
gpg: Good signature from "Amazon Lambda Insights Extension"
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: E0AF FA11 FFF3 5BD7 349E  E222 479C 97A1 848A BDC8
```

在預期的輸出中，可能會有關信任簽章的警告。只有您或您信任者所簽章的金鑰才能信任。這不表示該簽章是無效的，只是您尚未驗證該公有金鑰。

如果輸出包含 BAD `signature`，檢查您是否已正確執行步驟。如果您繼續收到BAD `signature`回應，請聯繫 AWS 並避免使用下載的文件。

ARM64 範例

本節包含在容器映像型 Python Lambda 函數上啟用 Lambda Insights 的範例。

在 Lambda 容器映像上啟用 Lambda Insights 的範例

1. 建立類似如下的 DDockerfile：

```
FROM public.ecr.aws/lambda/python:3.8
// extra lines to install the agent here
RUN curl -O https://lambda-insights-extension-arm64.s3-ap-
northeast-1.amazonaws.com/amazon_linux/lambda-insights-extension-arm64.rpm && \
    rpm -U lambda-insights-extension-arm64.rpm && \
    rm -f lambda-insights-extension-arm64.rpm

COPY index.py ${LAMBDA_TASK_ROOT}
CMD [ "index.handler" ]
```

2. 建立一個類似如下的名為 `index.py` 的 Python 檔案：

```
def handler(event, context):
  return {
    'message': 'Hello World!'
  }
```

3. 將 Dockerfile 和 `index.py` 放在同一個目錄中。然後，在該目錄中，執行以下步驟，從而建置 Docker 影像並將其上傳到 Amazon ECR。

```
// create an ECR repository
aws ecr create-repository --repository-name test-repository
// build the docker image
docker build -t test-image .
// sign in to AWS
aws ecr get-login-password | docker login --username AWS --password-stdin
 "${ACCOUNT_ID}".dkr.ecr."${REGION}".amazonaws.com
```

```
// tag the image
docker tag test-image:latest "${ACCOUNT_ID}".dkr.ecr."${REGION}".amazonaws.com/
test-repository:latest
// push the image to ECR
docker push "${ACCOUNT_ID}".dkr.ecr."${REGION}".amazonaws.com/test-
repository:latest
```

4. 使用您剛才建立的 Amazon ECR 映像來建立 Lambda 函數。
5. 將 CloudWatchLambdaInsightsExecutionRolePolicyIAM 政策指派給函數的執行角色。

檢視您的 Lambda Insights 指標

在已叫用的 Lambda 函數上安裝 Lambda 見解延伸模組之後，您可以使用 CloudWatch 主控台查看您的指標。您可以查看多函數概觀，或專注於單一函數。

如需 Lambda Insights 指標的清單，請參閱 Lambda Insights 收集的指標。

若要檢視 Lambda Insights 指標的多函數概觀

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在左側導覽窗格的 Lambda Insights 下方，選擇 Multi-function (多函數)。

   頁面頂端會顯示具有啟用 Lambda Insights 之區域中所有 Lambda 函數的彙總指標的圖形。頁面下方是列出了函數的表格。
3. 若要依函數名稱進行篩選，以減少顯示的函數數目，請在頁面頂端附近的方塊中輸入部分函數名稱。
4. 若要將此檢視作為小工具新增至儀表板，請選擇 Add to dashboard (新增至儀表板)。

若要檢視函數的指標

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在左側導覽窗格的 Lambda Insights 下方，選擇 Single-function (單一函數)。

   頁面頂端會顯示含有所選函數之指標的圖形。
3. 如果您已啟用 X-Ray，您可以選擇單一追蹤 ID。這會為該調用開啟「X-Ray 追蹤地圖」頁面，您可以將其縮小，以查看分散式追蹤以及處理該特定交易所涉及的其他服務。如需有關「X-Ray 追蹤地圖」的詳細資訊，請參閱使用 X-Ray 追蹤地圖。
4. 若要開啟 CloudWatch 記錄深入分析並放大特定錯誤，請選擇頁面底部的表格旁邊的 [檢視記錄]。

5. 若要將此檢視作為小工具新增至儀表板，請選擇 Add to dashboard (新增至儀表板)。

與 Application Insights 整合

Amazon CloudWatch 應用程式深入解析可協助您監控應用程式，並在應用程式資源和技術堆疊中識別和設定關鍵指標、日誌和警示。如需詳細資訊，請參閱 Amazon CloudWatch 應用洞察。

您可以啟用 Aplication Insights，從 Lambda 函數收集額外資料。如果尚未這樣做，選擇 Lambda Insights 儀表板中效能檢視下 Application Insights 索引標籤中的 Auto-configure Application Insights (自動設定 Application Insights) 即可啟用此功能。

如果您已經設定 CloudWatch 應用程式深入解析來監控 Lambda 函數，則「應用程式深入解析」儀表板會顯示在 Lambda Insights 儀表板下方的「應用程式深入解析」

Lambda Insights 收集的指標

Lambda Insights 會從安裝它的 Lambda 函數收集數個指標。其中一些測量結果可作為「量度」中的時間序列彙總 CloudWatch 資料使用。其他指標不會彙總至時間序列資料，但可以使用日誌深入解析，在內嵌的指標格式 CloudWatch 記錄項目中找到。

下列測量結果可作為`LambdaInsights`命名空間「測 CloudWatch 量結果」中的時間序列彙總資料使用。

指標名稱	維度	描述
`cpu_total_time`	function_name function_name, version	`cpu_system_time` 和 `cpu_user_time` 的總和。 單位：毫秒
`init_duration`	function_name function_name, version	在 Lambda 執行環境生命週期的 `init` 階段花費的時間。 單位：毫秒
`memory_utilization`	function_name function_name, version	以配置給函數的記憶體百分比來測量的記憶體上限。

指標名稱	維度	描述
		單位：百分比
rx_bytes	function_name function_name, version	函數接收的位元組數目。 單位：位元組
tmp_used		/tmp 目錄中的所用空間量 單位：位元組
tx_bytes	function_name function_name, version	函數傳送的位元組數目。 單位：位元組
total_memory	function_name function_name, version	配置到您 Lambda 函數的記憶體數量。這與你的函數的記憶體大小相同。 單位：MB
total_network	function_name function_name, version	rx_bytes 和 tx_bytes 的總和。即使是不執行輸入/輸出任務的函數，由於 Lambda 執行時間會進行網路呼叫，這個值通常也會大於零。 單位：位元組
used_memory_max	function_name function_name, version	函數沙盒的測量記憶體。 單位：MB

您可以使用記錄深入解析，在內嵌指標格式 CloudWatch 記錄項目中找到下列指標。如需 CloudWatch 日誌深入解析的相關資訊，請參閱使用日誌深入分析分析 CloudWatch 記錄資料

如需內嵌指標格式的詳細資訊，請參閱在日誌中內嵌指標。

指標名稱	描述
cpu_system_time	CPU 執行核心程式碼所花費的時間。 單位：毫秒
cpu_total_time	cpu_system_time 和 cpu_user_time 的總和。 單位：毫秒
cpu_user_time	CPU 執行使用者程式碼所花費的時間。 單位：毫秒
fd_max	可用的檔案描述項上限。 單位：計數
fd_use	使用中的檔案描述項上限。 單位：計數
memory_utilization	以配置給函數的記憶體百分比來測量的記憶體上限。 單位：百分比
rx_bytes	函數接收的位元組數目。 單位：位元組
tx_bytes	函數傳送的位元組數目。 單位：位元組

指標名稱	描述	
threads_max	函數程序使用的執行緒數量。作為函數撰寫者，您不能控制執行時間建立的執行緒的初始數量。 單位：計數	
tmp_max	/tmp 目錄中的可用空間量 單位：位元組	
total_memory	配置到您 Lambda 函數的記憶體數量。這與你的函數的記憶體大小相同。 單位：MB	
total_network	rx_bytes 和 tx_bytes 的總和。即使是不執行輸入/輸出任務的函數，由於 Lambda 執行時間會進行網路呼叫，這個值通常也會大於零。 單位：位元組	
used_memory_max	函數沙盒的測量記憶體。 單位：位元組	

故障診斷和已知問題

故障診斷任何問題的第一個步驟是啟用 Lambda Insights 延伸上的偵錯記錄。為此，請在 Lambda 函數上設定下列環境變數：LAMBDA_INSIGHTS_LOG_LEVEL=info。如需詳細資訊，請參閱使用 AWS Lambda 環境變數。

該延伸將日誌發出至與您的函數相同的日誌群組 (/aws/lambda/*function-name*))。檢閱這些日誌，查看錯誤是否可能與安裝問題有關。

我看不到來自 Lambda Insights 的任何指標

如果看不到您預期會看到的 Lambda Insights 指標，請檢查下列可能性：

- 量度可能只是延遲 — 如果函數尚未叫用或資料尚未清除，您將不會在中看到量度。 CloudWatch如需詳細資訊，請參閱本節後續的已知問題部分。
- 確認 Lambda 函數具有正確的權限 — 請確定已將 CloudWatchLambdaInsightsExecutionRolePolicyIAM 政策指派給函數的執行角色。
- 檢查 Lambda 執行時間— Lambda Insights 僅支援特定的 Lambda 執行時間。如需支援的執行時間的清單，請參閱 Lambda Insights。

  例如，若要在 Java 8 上使用 Lambda Insights，您必須使用 `java8.al2` 執行時間，而不是 `java8` 執行時間。
- 檢查網路存取 — Lambda 函數可能位於沒有網際網路存取權的 VPC 私有子網路上，而且您沒有為 CloudWatch 記錄設定 VPC 端點。為了協助偵錯此問題，您可以將環境變數設為 `LAMBDA_INSIGHTS_LOG_LEVEL=info`。

已知問題

資料延遲最多可達 20 分鐘。當函數處理常式完成時，Lambda 會凍結沙盒，也會凍結 Lambda Insights 延伸。在函數執行時，我們使用基於 TPS 函數的自適應批次策略來輸出資料。不過，如果函數停止叫用一段時間，且緩衝區中仍有事件資料，則可能會延遲此資料，直到 Lambda 關閉閒置沙盒為止。當 Lambda 關閉沙盒時，我們會排清緩衝的資料。

範例遙測事件

每次叫用啟用了 Lambda Insights 的 Lambda 函數會將單一日誌事件寫入 `/aws/lambda-insights` 日誌群組。每個日誌事件都包含內嵌指標格式的指標。如需內嵌指標格式的詳細資訊，請參閱在日誌中內嵌指標。

若要分析這些日誌事件，您可以使用下列方法：

- CloudWatch 主控台的 Lambda 見解區段，如中所述檢視您的 Lambda Insights 指標。
- 使用日誌深入解析記 CloudWatch 錄事件查詢。如需詳細資訊，請參閱使用日誌深入分析分析 CloudWatch 記錄資料。
- 在`LambdaInsights`命名空間中收集的量度，您可以使用 CloudWatch量度繪製圖形。

以下是具有內嵌指標格式的 Lambda Insights 日誌事件範例。

```
{
    "_aws": {
```

```
        "Timestamp": 1605034324256,
        "CloudWatchMetrics": [
            {
                "Namespace": "LambdaInsights",
                "Dimensions": [
                    [ "function_name" ],
                    [ "function_name", "version" ]
                ],
                "Metrics": [
                    { "Name": "memory_utilization", "Unit": "Percent" },
                    { "Name": "total_memory", "Unit": "Megabytes" },
                    { "Name": "used_memory_max", "Unit": "Megabytes" },
                    { "Name": "cpu_total_time", "Unit": "Milliseconds" },
                    { "Name": "tx_bytes", "Unit": "Bytes" },
                    { "Name": "rx_bytes", "Unit": "Bytes" },
                    { "Name": "total_network", "Unit": "Bytes" },
                    { "Name": "init_duration", "Unit": "Milliseconds" }
                ]
            }
        ],
        "LambdaInsights": {
            "ShareTelemetry": true
        }
    },
    "event_type": "performance",
    "function_name": "cpu-intensive",
    "version": "Blue",
    "request_id": "12345678-8bcc-42f7-b1de-123456789012",
    "trace_id": "1-5faae118-12345678901234567890",
    "duration": 45191,
    "billed_duration": 45200,
    "billed_mb_ms": 11571200,
    "cold_start": true,
    "init_duration": 130,
    "tmp_free": 538329088,
    "tmp_max": 551346176,
    "threads_max": 11,
    "used_memory_max": 63,
    "total_memory": 256,
    "memory_utilization": 24,
    "cpu_user_time": 6640,
    "cpu_system_time": 50,
    "cpu_total_time": 6690,
    "fd_use": 416,
```

```
    "fd_max": 32642,
    "tx_bytes": 4434,
    "rx_bytes": 6911,
    "timeout": true,
    "shutdown_reason": "Timeout",
    "total_network": 11345,
    "agent_version": "1.0.72.0",
    "agent_memory_avg": 10,
    "agent_memory_max": 10
}
```

使用貢獻者洞察分析高基數資料

您可以使用 Contributor Insights 來分析日誌資料，以及建立顯示參與者資料的時間序列。您可以查看與前 N 個參與者有關的指標、唯一參與者的總數及其用量。這有助於您尋找最頂端的說話者，並了解正在影響系統效能的對象或項目。例如，您可以尋找錯誤的主機、識別負荷最重的網路使用者，或是尋找產生最多錯誤的 URL。

您可以從頭開始建立規則，使用時也 AWS Management Console 可以使用 AWS 已建立的範例規則。規則會定義您希望用來定義參與者的日誌欄位，例如 `IpAddress`。您也可以篩選日誌資料來尋找和分析個別參與者的行為。

CloudWatch 也提供內建規則，可用來分析來自其他 AWS 服務的指標。

所有規則都會即時分析傳入的資料。

如果您登入的帳戶設定為 CloudWatch 跨帳戶可觀察性的監視帳戶，您可以在該監視帳戶中建立 Contributor Insights 規則，以分析來源帳戶和監視帳戶中的記錄群組。您也可以建立單一規則，來分析多個帳戶中的日誌群組。如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。

> **Note**
>
> 如果您使用 Contributor Insights，每次出現符合規則的日誌事件都需要支付費用。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

主題

- 建立 Contributor Insights 規則
- Contributor Insights 規則語法

- Contributor Insights 規則範例
- 檢視 Contributor Insights 報告
- 繪製規則產生的指標
- 使用 Contributor Insights 內建規則

建立 Contributor Insights 規則

您可以建立規則來分析日誌資料。任何 JSON 或通用日誌格式 (CLF) 的日誌都可進行評估。這包括以下其中 AWS 一種格式的自訂日誌，以及來自 Amazon VPC 流程日誌、Amazon 路線 53 DNS 查詢日誌、Amazon ECS 容器日誌以及來自 Amazon AWS CloudTrail SageMaker、Amazon RDS AWS AppSync 和 API Gateway 的日誌。

在規格中，當您指定欄位名稱或值時，所有的比對都區分大小寫。

您可以在建立規則時使用內建範例規則，或者也可以從頭開始建立您自己的規則。Contributor Insights 包含下列日誌類型的範例規則：

- Amazon API Gateway 日誌
- Amazon Route 53 公有 DNS 查詢日誌
- Amazon Route 53 Resolver 查詢日誌
- CloudWatch 容器洞察日誌
- VPC 流量日誌

如果您已登入設定為 CloudWatch 跨帳戶觀察性監視帳戶的帳戶，則除了在監視帳戶中為記錄群組建立規則之外，還可以為連結至此監視帳戶的來源帳戶中的記錄群組建立 Contributor Insights 規則。您也可以設定單一規則，來監控不同帳戶中的日誌群組。如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。

> **Important**
>
> 當您授與使用者`cloudwatch:PutInsightRule`權限時，依預設，該使用者可以建立用來評估記錄檔中任何 CloudWatch 記錄群組的規則。您可以新增 IAM 政策條件，以限制這些許可，進而讓使用者包含和排除特定日誌群組。如需詳細資訊，請參閱 使用條件索引鍵限制 Contributor Insights 使用者存取日誌群組。

如何使用內建範例規則建立規則

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Insights (洞察)，然後選擇 Contributor Insights。
3. 選擇建立規則。
4. 在 Select log group(s) (選取日誌群組) 中，選取您要讓規則監控的日誌群組。您最多可以選取 20 個日誌群組。如果您已登入設定為 CloudWatch 跨帳戶觀察性的監視帳戶，則可以在來源帳戶中選取記錄群組，也可以設定單一規則來分析不同帳戶中的記錄群組。

   - (選用) 若要選取名稱開頭為特定字串的所有日誌群組，請選擇 Select by prefix match (依字首相符選取) 下拉式清單，然後輸入字首。如果這是監控帳戶，您可以選擇選取要搜尋的帳戶，否則會選取所有帳戶。

   > **Note**
   >
   > 每個符合規則的日誌事件都會產生費用。如果選擇 Select by prefix match (依字首相符選取) 下拉式清單，請注意字首可能匹配到的日誌群組數量。如果您搜尋到的日誌群組比所需的多，則可能會產生非預期的費用。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

5. 在 Rule type (規則類型) 中，選擇 Sample rule (範例規則)。然後選擇 Select sample rule (選取範例規則) 並選取規則。
6. 範例規則已填寫 Log format (日誌格式)、Contribution (貢獻)、Filters (篩選條件) 和 Aggregate on (彙整對象) 等欄位。您可以視需要調整這些值。
7. 選擇下一步。
8. 在 Rule name (規則名稱) 中，輸入名稱。有效字元為 A-Z、a-z、0-9、(連字號)、(底線) 和 (句號)。
9. 選擇是要以停用或啟用狀態建立規則。如果您選擇啟用，規則會立即開始分析您的資料。執行啟用的規則時會產生費用。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

   Contributor Insights 只會分析建立規則之後的新日誌事件。規則無法處理記錄檔先前由記 CloudWatch 錄檔處理的事件。
10. (選用) 在 Tags (標籤) 中，新增一或多個鍵/值對，作為此值的標籤。標籤可以幫助您識別和組織 AWS 資源並追蹤 AWS 成本。如需詳細資訊，請參閱 標記您的 Amazon CloudWatch 資源。
11. 選擇 Create (建立)。

如何從頭開始建立規則

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Contributor Insights。
3. 選擇建立規則。
4. 在 Select log group(s) (選取日誌群組) 中，選取您要讓規則監控的日誌群組。您最多可以選取 20 個日誌群組。如果您已登入設定為 CloudWatch 跨帳戶觀察性的監視帳戶，則可以在來源帳戶中選取記錄群組，也可以設定單一規則來分析不同帳戶中的記錄群組。

   - (選用) 若要選取名稱開頭為特定字串的所有日誌群組，請選擇 Select by prefix match (依字首相符選取) 下拉式清單，然後輸入字首。

   > **Note**
   >
   > 每個符合規則的日誌事件都會產生費用。如果選擇 Select by prefix match (依字首相符選取) 下拉式清單，請注意字首可能匹配到的日誌群組數量。如果您搜尋到的日誌群組比所需的多，則可能會產生非預期的費用。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

5. 在 Rule type (規則類型) 中，選擇 Custom rule (自訂規則)。
6. 針對 Log format (日誌格式)，選擇 JSON 或 CLF。
7. 您可以使用精靈完成建立規則，或是透過選擇 Syntax (語法) 標籤並手動指定您的規則語法。

   如要繼續使用精靈，請執行以下作業：

   a. 針對 Contribution (貢獻)、Key (索引鍵)，輸入您希望報告的參與者類型。報告會針對此參與者類型顯示前 N 個值。

      有效的項目是任何包含值的日誌欄位。範例包括 **`requestId`**、**`sourceIPaddress`** 和 **`containerID`**。

      如需有關為特定日誌群組中的日誌尋找日誌欄位名稱的資訊，請參閱「尋找日誌欄位」。

      大於 1 KB 的索引鍵會截斷至 1KB。

   b. (選用) 選擇 Add new key (新增鍵)，以新增更多鍵。您最多可以在規則中包含四個索引鍵。如果您輸入的索引鍵超過一個，報告中的參與者會由索引鍵的唯一值組合定義。例如，如果您指定了三個索引鍵，則三個索引鍵的每個唯一值組合都會計為唯一的參與者。

c. (選用) 如果您想要新增篩選條件，來縮小結果範圍，請選擇 Add filter (新增篩選條件)。針對 Match (比對)，輸入您希望據以篩選的日誌名稱。針對 Condition (條件)，選擇比較運算子並輸入您希望為其進行篩選的值。

您可以在規則中新增最多四個篩選條件。多個篩選條件會以 AND 邏輯聯結，因此只會評估符合所有篩選條件的日誌事件。

> **Note**
>
> 遵循比較運算子的陣列 (例如 `In`、`NotIn` 或 `StartsWith`) 最多可包含 10 個字串值。如需有關 Contributor Insights 規則語法的詳細資訊，請參閱 Contributor Insights 規則語法。

d. 針對 Aggregate on (彙整對象)，選擇 Count (計數) 或 Sum (總和)。選擇 Count (計數) 會使參與者排名以發生次數為基礎。選擇 Sum (總和) 會使排名以您為 Contribution (貢獻)、Value (值) 所指定的欄位值彙整總和為基礎。

8. 如要輸入您的規則作為 JSON 物件，而非使用精靈，請執行以下作業：

a. 選擇 Syntax (語法) 標籤。

b. 在 Rule body (規則主體) 中，輸入您規則的 JSON 物件。如需規則語法的資訊，請參閱 Contributor Insights 規則語法。

9. 選擇下一步。

10. 在 Rule name (規則名稱) 中，輸入名稱。有效字元是 A–Z、a–z、0–9、"-"、"_" 和 "."。

11. 選擇是要以停用或啟用狀態建立規則。如果您選擇啟用，規則會立即開始分析您的資料。執行啟用的規則時會產生費用。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

Contributor Insights 只會分析建立規則之後的新日誌事件。規則無法處理記錄檔先前由記 CloudWatch 錄檔處理的事件。

12. (選用) 在 Tags (標籤) 中，新增一或多個鍵/值對，作為此值的標籤。標籤可以幫助您識別和組織 AWS 資源並追蹤 AWS 成本。如需詳細資訊，請參閱 標記您的 Amazon CloudWatch 資源。

13. 選擇下一步。

14. 確認您輸入的設定值，然後選擇 Create rule (建立規則)。

您可以停用、啟用或刪除您已建立的規則。

啟用、停用或刪除 Contributor Insights 中的規則

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Contributor Insights。
3. 在規則清單中，選取單一規則旁邊的核取方塊。

   內建規則是由 AWS 服務建立，無法編輯、停用或刪除。
4. 選擇 Actions (動作)，然後選擇您希望的選項。

尋找日誌欄位

當您建立規則時，您需要了解日誌群組中日誌項目內的欄位名稱。

尋找日誌群組中的日誌欄位

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格的 Logs (日誌) 下，選擇 Insights (深入分析)。
3. 在查詢編輯器上方，選取一或多個要查詢的日誌群組。

   當您選取記錄群組時， CloudWatch Logs Insights 會自動偵測記錄群組中資料的欄位，並將這些欄位顯示在 [探查] 欄位的右窗格中。

Contributor Insights 規則語法

本節說明 Contributor Insights 規則的語法。請只在您透過輸入 JSON 區塊來建立規則時使用此語法。如果您使用精靈建立規則，您便不需要了解語法。如需使用精靈建立規則的詳細資訊，請參閱 建立 Contributor Insights 規則。

所有規則與日誌事件欄位名稱及值進行的比對都區分大小寫。

以下範例示範 JSON 日誌的語法。

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
```

```
    "LogGroupNames": [
        "API-Gateway-Access-Logs*",
        "Log-group-name2"
    ],
    "LogFormat": "JSON",
    "Contribution": {
        "Keys": [
            "$.ip"
        ],
        "ValueOf": "$.requestBytes",
        "Filters": [
            {
                "Match": "$.httpMethod",
                "In": [
                    "PUT"
                ]
            }
        ]
    },
    "AggregateOn": "Sum"
}
```

Contributor Insights 規則中的欄位

結構描述

分析 CloudWatch 記錄檔資料之規則的`Schema`值必須始終為 `{"Name": "CloudWatchLogRule", "Version": 1}`

LogGroupNames

字串陣列。針對陣列中的每個元素，您可以選擇在字串的結尾使用 * 來包含所有名稱開頭為該字首的日誌群組。

請小心搭配日誌群組名稱使用萬用字元。每個符合規則的日誌事件都會產生費用。如果您不小心搜尋了比您預期數量更多的日誌群組，您可能會產生未預期的費用。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

LogGroupARN

如果您要在 CloudWatch 跨帳戶觀察性監視帳戶中建立此規則，則可以使用在連結`LogGroupARNs`至監視帳戶的來源帳戶中指定記錄群組，並在監視帳戶本身中指定記錄群組。您必須在規則中指定 `LogGroupNames` 或 `LogGroupARNs`，但不能同時指定兩者。

LogGroupARNs 是字串陣列。對於陣列中的每個元素，您可以在某些情況下選擇使用 * 作為萬用字元。例如，您可以指定 arn:aws:logs:us-west-1:*:log-group/MyLogGroupName2，以指定美國西部 (加利佛尼亞北部) 區域中所有來源帳戶以及監控帳戶中命名為 MyLogGroupName2 的日誌群組。您也可以指定 arn:aws:logs:us-west-1:111122223333:log-group/GroupNamePrefix*，來指定 111122223333 帳戶在美國西部 (加利佛尼亞北部) 中名稱以 GroupNamePrefix 開頭的所有日誌群組。

您無法將部分 AWS 帳戶 ID 指定為具有萬用字元的前置字元。

將萬用字元用於日誌群組 ARN 用時應小心。每個符合規則的日誌事件都會產生費用。如果您不小心搜尋了比您預期數量更多的日誌群組，您可能會產生未預期的費用。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

LogFormat

有效值為 JSON 和 CLF。

Contribution

此物件包含 Keys 陣列 (其中最多可以有四個成員)、選用的單一 ValueOf，以及其中最多可包含四個 Filters 的選用陣列。

鍵

最多四個日誌欄位的陣列，會用來作為分類參與者的維度。如果您輸入的索引鍵超過一個，索引鍵值的每個唯一組合都會計為唯一的參與者。欄位必須使用 JSON 屬性格式表示法指定。

ValueOf

(選用) 請只在您指定 Sum 作為 AggregateOn 的值時才使用此項目。ValueOf 會使用數值指定日誌欄位。在這種規則類型中，參與者會根據此欄位值的總和進行排名，而非在日誌項目中發生的次數。例如，如果您想要根據其在特定期間內的總 BytesSent 排名參與者，您可將 ValueOf 設為 BytesSent，並為 AggregateOn 指定 Sum。

篩選條件

(選用) 指定最多四個篩選條件的陣列，縮小包含在報告中的日誌事件。如果您指定了多個篩選條件，Contributor Insights 會使用邏輯 AND 運算子進行評估。您可以使用此項目來在您的搜尋中篩選掉不相關的日誌事件，或是您可以用來選取單一參與者以分析其行為。

陣列中的每個成員都必須包含 Match 欄位，以及一個指出要使用比對運算子類型的欄位。

Match 欄位會指定要在篩選條件中評估的日誌欄位。日誌欄位是使用 JSON 屬性格式表示法指定。

比對運算子欄位必須是以下其中一個：In、NotIn、StartsWith、GreaterThan、LessThan、EqualTo、NotEqualTo 或 IsPresent。如果運算子欄位是 In、NotIn 或 StartsWith，其後會跟隨一個要檢查的字串值陣列。Contributor Insights 會使用 OR 運算子評估字串值陣列。陣列最多可包含 10 個字串值。

如果運算子欄位是 GreaterThan、LessThan、EqualTo 或 NotEqualTo，其後會跟隨要比較的單一數值。

如果運算子欄位是 IsPresent，其後會跟隨 true 或 false。此運算子會根據日誌事件中是否存在指定的日誌欄位來比對日誌事件。isPresent 只能搭配 JSON 屬性分葉節點中的值運作。例如，尋找與 c-count 相符的篩選條件不會評估值為 details.c-count.c1 的日誌事件。

請參閱以下四個篩選條件範例：

```
{"Match": "$.httpMethod", "In": [ "PUT", ] }
{"Match": "$.StatusCode", "EqualTo": 200 }
{"Match": "$.BytesReceived", "GreaterThan": 10000}
{"Match": "$.eventSource", "StartsWith": [ "ec2", "ecs" ] }
```

AggregateOn

有效值為 Count 和 Sum。指定是要根據發生計數彙整報告，還是在 ValueOf 欄位中指定的欄位值總和彙整報告。

JSON 屬性格式表示法

Keys、ValueOf 和 Match 欄位允許使用點表示法的 JSON 屬性格式，其中 $ 表示 JSON 物件的根。其後會跟隨一個句點，然後是包含子屬性名稱的英數字串。其支援多個屬性層級。

字串的第一個字元只能是 A-Z 或 a-z。字串的後續字元可以是 A-Z、a-z 或 0-9。

以下清單示範 JSON 屬性格式的有效範例：

```
$.userAgent
$.endpoints[0]
$.users[1].name
$.requestParameters.instanceId
```

CLF 日誌規則中的其他欄位

通用日誌格式 (CLF) 日誌事件不會跟 JSON 一樣包含欄位的名稱。如要提供欄位以用於 Contributor Insights 規則，CLF 日誌事件可以視為索引從 1 開始的陣列處理。您可以將第一個欄位指定為 **"1"**，將第二個欄位指定為 **"2"**，以此類推。

如要讓 CLF 日誌的規則更容易閱讀，您可以使用 `Fields`。這可以讓您為 CLF 欄位位置提供命名別名。例如，您可以指定位置 "4" 是一個 IP 位址。指定之後，`IpAddress` 便可以用來在規則的 `Keys`、`ValueOf` 和 `Filters` 中作為屬性。

以下是使用 `Fields` 欄位的 CLF 日誌規則範例。

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "API-Gateway-Access-Logs*"
    ],
    "LogFormat": "CLF",
    "Fields": {
        "4": "IpAddress",
        "7": "StatusCode"
    },
    "Contribution": {
        "Keys": [
            "IpAddress"
        ],
        "Filters": [
            {
                "Match": "StatusCode",
                "EqualTo": 200
            }
        ]
    },
    "AggregateOn": "Count"
}
```

Contributor Insights 規則範例

本節包含範例，示範 Contributor Insights 規則的使用案例。

VPC 流程日誌：來源及目的地 IP 地址傳輸的位元組

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "/aws/containerinsights/sample-cluster-name/flowlogs"
    ],
    "LogFormat": "CLF",
    "Fields": {
        "4": "srcaddr",
        "5": "dstaddr",
        "10": "bytes"
    },
    "Contribution": {
        "Keys": [
            "srcaddr",
            "dstaddr"
        ],
        "ValueOf": "bytes",
        "Filters": []
    },
    "AggregateOn": "Sum"
}
```

VPC 流程日誌：HTTPS 請求數量上限

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "/aws/containerinsights/sample-cluster-name/flowlogs"
    ],
    "LogFormat": "CLF",
    "Fields": {
        "5": "destination address",
        "7": "destination port",
        "9": "packet count"
    },
    "Contribution": {
        "Keys": [
```

```
            "destination address"
        ],
        "ValueOf": "packet count",
        "Filters": [
            {
                "Match": "destination port",
                "EqualTo": 443
            }
        ]
    },
    "AggregateOn": "Sum"
}
```

VPC 流程日誌：遭拒絕的 TCP 連線

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "/aws/containerinsights/sample-cluster-name/flowlogs"
    ],
    "LogFormat": "CLF",
    "Fields": {
        "3": "interfaceID",
        "4": "sourceAddress",
        "8": "protocol",
        "13": "action"
    },
    "Contribution": {
        "Keys": [
            "interfaceID",
            "sourceAddress"
        ],
        "Filters": [
            {
                "Match": "protocol",
                "EqualTo": 6
            },
            {
                "Match": "action",
                "In": [
```

```
                    "REJECT"
                ]
            }
        ]
    },
    "AggregateOn": "Sum"
}
```

依來源地址進行 Route 53 NXDomain 回應

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "AggregateOn": "Count",
    "Contribution": {
        "Filters": [
            {
                "Match": "$.rcode",
                "StartsWith": [
                    "NXDOMAIN"
                ]
            }
        ],
        "Keys": [
            "$.srcaddr"
        ]
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "<loggroupname>"
    ]
}
```

依網域名稱進行 Route 53 Resolver 查詢

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "AggregateOn": "Count",
```

```
    "Contribution": {
        "Filters": [],
        "Keys": [
            "$.query_name"
        ]
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "<loggroupname>"
    ]
}
```

依查詢類型和來源地址進行 Route 53 Resolver 查詢

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "AggregateOn": "Count",
    "Contribution": {
        "Filters": [],
        "Keys": [
            "$.query_type",
            "$.srcaddr"
        ]
    },
    "LogFormat": "JSON",
    "LogGroupNames": [
        "<loggroupname>"
    ]
}
```

檢視 Contributor Insights 報告

如要檢視報告資料的圖表和您規則找到的參與者排名清單，請遵循這些步驟。

檢視您的規則報告

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Contributor Insights。

3. 在規則清單中，選擇規則的名稱。

   圖表會顯示過去三個小時內的規則結果。圖表下方的表格會顯示前 10 個參與者。

4. 如要變更表格中顯示的參與者數，請選擇圖表頂端的 Top 10 contributors (前 10 個參與者)。

5. 如要篩選圖表，只顯示來自單一參與者的結果，請在表格圖例中選擇該參與者。如要再次顯示所有參與者，請在圖例中再次選擇相同的參與者。

6. 如要變更報告中顯示的時間範圍，請在圖表的頂端選擇 15m (15 分鐘)、30m (30 分鐘)、1h (1 小時)、2h (2 小時)、3h (3 小時) 或 custom (自訂)。

   報告的時間範圍上限是 24 小時，但您可以選擇長達過去 15 天內發生的 24 小時時段。如要選擇過去的時段，請選擇 custom (自訂)、absolute (絕對)，然後指定您的時段。

7. 如要變更用於彙整和排名參與者的時間期間長度，請選擇圖表頂端的 period (期間)。檢視更長的時間期間通常會顯示尖峰較少且較為平滑的報告。選擇較短的時間期間更有可能會顯示尖峰。

8. 若要將此圖形新增至 CloudWatch 控制面板，請選擇 [新增至儀表板]。

9. 若要開啟 [CloudWatch 記錄檔見解] 查詢視窗，且此報告中的記錄群組已載入查詢方塊中，請選擇 [檢視記錄]。

10. 如要將報告資料匯出至您的剪貼簿或 CSV 檔案，請選擇 Export (匯出)。

繪製規則產生的指標

Contributor Insights 提供指標數學函數 `INSIGHT_RULE_METRIC`。您可以使用此函數，將「參與者見解」報表中的資料新增至 CloudWatch 主控台「度量」索引標籤中的圖表。您也可以根據此數學函數設定警示。如需指標數學函數的詳細資訊，請參閱 使用指標數學。

若要使用此指標數學函數，您必須登入擁有 `cloudwatch:GetMetricData` 和 `cloudwatch:GetInsightRuleReport` 許可的帳戶。

語法是 `INSIGHT_RULE_METRIC(`*`ruleName`*`, `*`metricName`*`)`。*`ruleName`* 是 Contributor Insights 規則的名稱。*`metricName`* 是以下清單中的其中一個值。*`metricName`* 的值會判斷數學函數傳回的資料類型。

- `UniqueContributors` — 每個資料點的唯一貢獻因子數目。
- `MaxContributorValue` — 每個資料點的最佳貢獻因子值。圖表中每個資料點的參與者的身分可能會變更。

如果此規則是以 Count 彙整，則每個資料點的最高貢獻因子是在該期間中發生次數最多的貢獻因子。如果規則是以 Sum 彙整，則最高貢獻因子是該期間中，規則的 Value 所指定日誌欄位內總和數最大的貢獻因子。

- SampleCount — 規則符合的資料點數目。
- Sum — 該資料點所表示時間期間內來自所有貢獻因子的值的總和。
- Minimum — 該資料點所表示時間期間內單一觀察的最小值。
- Maximum — 該資料點所表示時間期間內單一觀察的最大值。
- Average — 該資料點所表示時間期間內來自所有貢獻因子的平均值。

在 Contributor Insights 指標資料上設定警示

您可以使用函數 INSIGHT_RULE_METRIC，對 Contributor Insights 產生的指標設定警示。例如，您可以根據遭拒絕傳輸控制通訊協定 (TCP) 連線百分比來建立警示。如果要開始使用此類警示，您可以建立類似於以下兩個範例中所示的規則：

範例規則："RejectedConnectionsRule"

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "/aws/containerinsights/sample-cluster-name/flowlogs"
    ],
    "LogFormat": "CLF",
    "Fields": {
        "3": "interfaceID",
        "4": "sourceAddress",
        "8": "protocol",
        "13": "action"
    },
    "Contribution": {
        "Keys": [
            "interfaceID",
            "sourceAddress"
        ],
        "Filters": [
            {
```

```
                "Match": "protocol",
                "EqualTo": 6
            },
            {
                "Match": "action",
                "In": [
                    "REJECT"
                ]
            }
        ]
    },
    "AggregateOn": "Sum"
}
```

範例規則："TotalConnectionsRule"

```
{
    "Schema": {
        "Name": "CloudWatchLogRule",
        "Version": 1
    },
    "LogGroupNames": [
        "/aws/containerinsights/sample-cluster-name/flowlogs"
    ],
    "LogFormat": "CLF",
    "Fields": {
        "3": "interfaceID",
        "4": "sourceAddress",
        "8": "protocol",
        "13": "action"
    },
    "Contribution": {
        "Keys": [
            "interfaceID",
            "sourceAddress"
        ],
        "Filters": [
            {
                "Match": "protocol",
                "EqualTo": 6
            }
    "AggregateOn": "Sum"
}
```

建立規則之後，您可以在 CloudWatch 主控台中選取「度量」索引標籤，您可以在其中使用下列範例量度數學運算式，繪製「參與者見解」報告的資料圖形：

範例：指標數學表達式

```
e1 INSIGHT_RULE_METRIC("RejectedConnectionsRule", "Sum")
e2 INSIGHT_RULE_METRIC("TotalConnectionsRule", "Sum")
e3 (e1/e2)*100
```

在該範例中，指標數學表達式 e3 會傳回所有遭拒絕的 TCP 連線。如果您想要在 20% 的 TCP 連線遭到拒絕時收到通知，您可以透過將閾值從 100 變更為 20 來修改表達式。

> **Note**
>
> 您可以在 Metrics (指標) 區段中，對您監控的指標設定警示。在 Graphed metrics (圖表化指標) 索引標籤上，您可以選取 Actions (動作) 資料欄下的 Create alarm (建立警示) 圖示。Create alarm (建立警示) 圖示看起來像個鈴鐺。

如需有關繪製指標圖表和使用指標數學函數的詳細資訊，請參閱以下區段：將數學運算式新增至 CloudWatch 圖表。

使用 Contributor Insights 內建規則

您可以使用貢獻者見解內建規則來分析來自其他 AWS 服務的指標。下列服務支援內建規則：

- 《Amazon DynamoDB 開發人員指南》中的 Contributor Insights for Amazon DynamoDB。
- 《AWS PrivateLink 指南》中的使用 Contributor Insights 內建規則。

Amazon CloudWatch 應用洞察

Amazon 應用 CloudWatch 程式洞見有助於觀察您的應用程式和基礎 AWS 資源。它可協助您設定應用程式資源的最佳監控，以持續分析資料的應用程式問題跡象。由SageMaker其他 AWS 技術提供支援的 Application Insights 提供自動化儀表板，顯示受監控應用程式的潛在問題，協助您快速找出應用程式和基礎架構持續發生的問題。Application Insights 為應用程式運作狀態提供的增強可見度，有助於降低平均修復時間 (MTTR)，進而故障診斷應用程式的問題。

當您將應用程式新增至 Amazon CloudWatch 應用程式洞察時，它會掃描應用程式中的資源，並建議和設定應用程式元件的指標和登入。CloudWatch範例應用程式元件可包含 SQL Server 後端資料庫和

Microsoft IIS/Web 層。Application Insights 會使用歷史資料分析指標模式以偵測異常，並持續偵測應用程式、作業系統和基礎設施日誌中的錯誤和異常。它會結合分類演算法和內建規則，建立這些觀察的關聯。然後，自動建立儀表板，顯示相關的觀察和問題嚴重性資訊，協助您決定動作的優先順序。針對 .NET 和 SQL 應用程式堆疊中的常見問題，例如應用程式延遲、SQL Server 故障備份、記憶體流失、大型 HTTP 請求，以及取消的輸入/輸出操作，它提供了額外的洞察，指出可能的根本原因和解決步驟。與 AWS SSM 的內建整合 OpsCenter可讓您執行相關的 Systems Manager 自動化文件來解決問題。

章節

- 什麼是 Amazon CloudWatch 應用程序洞察？
- Amazon 應 CloudWatch 用程式洞察如何運作
- 開始使 CloudWatch 用 Amazon 應用程式深入解析
- Application Insights 跨帳戶觀察
- 使用元件組態
- 使用 CloudFormation 範本建立和設定 CloudWatch 應用程式見解監控
- 教學課程：設定 SAP ASE 的監控
- 教學課程：設定 SAP HANA 的監控
- 教學課程：設定 SAP 的監督 NetWeaver
- 檢視 Amazon CloudWatch 應用程式深入解析所偵測到的問題並
- Amazon CloudWatch 應用程式深入解析支援的日誌和指標

什麼是 Amazon CloudWatch 應用程序洞察？

CloudWatch 應用程式洞見可協助您監控使用 Amazon EC2 執行個體的應用程式以及其他應用程式資源。它會識別和設定關鍵指標、日誌，並在您所有的應用程式資源和技術堆疊 (例如，Microsoft SQL Server 資料庫、Web (IIS) 和應用程式伺服器、作業系統、負載平衡器和佇列) 中發出警示。並持續監控指標和日誌，以偵測和建立異常及錯誤的關聯。偵測到錯誤和異常時，「應用程式深入解析」會產生CloudWatch 事件，讓您用來設定通知或採取動作。為協助進行故障診斷，它會建立已偵測到問題的自動化儀表板，包括關聯的指標異常和日誌錯誤以及其他洞見，指出可能的根本原因。自動化儀表板可協助您採用補救動作，讓應用程式保持良好的運作狀態，防止應用程式的最終使用者受到影響。它也會建立， OpsItems 以便您可以使用 AWS SSM OpsCenter 解決問題。

您可以設定重要的計數器，例如鏡像寫入交易/秒、復原佇列長度和交易延遲，以及 Windows 事件記錄檔開啟。 CloudWatch當 SQL HA 工作負載發生容錯移轉事件或問題 (例如查詢目標資料庫的限制存取) 時，「 CloudWatch 應用程式深入解析」會提供自動化的深入解析。

CloudWatch 應用程式深入解析與AWS Launch Wizard整合，提供在上部署 SQL Server HA 工作負載的一鍵式監控設定體驗 AWS。當您選取使用 Launch Wizard 主控台上的應用程式深入解析來設定監視和深入解析的選項時，Ap CloudWatch plication Insights 會自動設定相關指標、記錄和警示 CloudWatch，並開始監控新部署的工作負載。您可以在 CloudWatch 主控台上檢視自動化的見解和偵測到的問題，以及 SQL Server HA 工作負載的健全狀況。

目錄

- 功能
- 概念
- 定價
- 相關服務
- 支援的應用程式元件
- 支援的技術堆疊

功能

Application Insights 提供以下功能。

自動設定應用程式資源監控

CloudWatch 應用程式深入解析可縮短設定應用程式監控所需的時間。它透過掃描您的應用程式資源、提供可自訂的建議指標和日誌清單，然後進行設定，以提供對應用程式資源 (例如 Amazon EC2 和彈性負載平衡器 (ELB) 等應用程式資源的必要可見度 CloudWatch 來達成此目的。它還會對監控的指標設定動態警示。警示會根據兩週前偵測到的異常自動更新。

問題偵測和通知

CloudWatch 應用程式深入解析可偵測應用程式潛在問題的跡象，例如量度異常和記錄錯誤。它會建立這些觀察的關聯，以顯示應用程式的潛在問題。然後它會產生 CloudWatch 事件，這些事件可以設定為接收通知或採取動作。這可讓您不必建立指標或日誌錯誤的個別警示。

故障診斷

CloudWatch 應用程式洞察會針對偵測到的問題建立 CloudWatch 自動儀表板。此儀表板會顯示問題的詳細資訊，包括相關聯的指標異常和日誌錯誤，以利故障診斷。它們也會提供額外的洞見，指出異常和錯誤的可能根本原因。

概念

以下是了解 Application Insights 如何監控應用程式的重要概念。

元件

構成應用程式的自動分組、獨立或自訂分組的類似資源。我們建議將類似的資源分組為自訂的元件，以利監控。

觀察

偵測到的應用程式或應用程式資源個別事件 (指標異常、日誌錯誤或例外狀況)。

問題

透過建立關聯、分類和分組相關觀察偵測到問題。

如需 CloudWatch 應用程式深入解析其他關鍵概念的定義，請參閱 Amazon CloudWatch 概念。

定價

CloudWatch 「應用程式深入解析」會使用指標、記錄和事件，為所選應用程式資源設定建議的 CloudWatch 指標和記錄，以獲取偵測到的問題 這些功能會根據CloudWatch 定價向您的 AWS 帳戶收取費用。針對偵測到的問題，SSM 也 OpsItems會由應用程式見解建立，以通知您有關問題的相關資訊。此外，應用程式深入解析還會建立 SSM 參數存放區參數，以便在執行個體上設定 CloudWatch 代理 Amazon EC2 Systems Manager 功能是根據 SSM 定價收費。設定協助、監控、資料分析或問題偵測不收取費用。

CloudWatch 應用程式洞察的成本

Amazon EC2 的成本包含下列功能的使用情況：

- CloudWatch 代理
  - CloudWatch 用戶端記錄群組
  - CloudWatch 代理程式量
  - Prometheus 日誌群組 (適用於 JMX 工作負載)

所有資源的成本包括以下功能的使用：

- CloudWatch 警報（大部分成本）

- 超音波馬達 OpsItems (最低成本)

成本計算範例

在這個範例中的成本根據下列情況來計算。

您建立了包含下列項目的資源群組：

- 一個安裝了 SQL Server 的 Amazon EC2 執行個體。
- 連接了 Amazon EBS 磁碟區。

當您使 CloudWatch 用應用程式洞見將此資源群組上線時，會偵測到安裝在 Amazon EC2 執行個體上的 SQL Server 工作負載。 CloudWatch 應用程序洞察開始監視以下指標。

針對 SQL Server 執行個體，監控以下指標：

- CPUUtilization
- StatusCheckFailed
- 記憶體 % 使用中的認可位元組
- 記憶體可用的 MB 數
- 網路介面位元組總數/秒
- 分頁檔用量 %
- 實體磁碟 % 磁碟時間
- 處理器 % 處理器時間
- SQLServer:Buffer Manager 快取命中率
- SQLServer:Buffer Manager 預期壽命
- SQLServer:General Statistics Processes blocked
- SQLServer：一般統計資料使用者連線
- SQLServer：鎖定死鎖數目/秒
- SQLServer:SQL Statistics 每秒批次要求數
- 系統處理器佇列長度

針對連接到 SQL Server 執行個體的磁碟區，監控以下指標：

- VolumeReadBytes
- VolumeWriteBytes
- VolumeReadOps
- VolumeWriteOps
- VolumeTotalReadTime
- VolumeTotalWriteTime
- VolumeIdleTime
- VolumeQueueLength
- VolumeThroughputPercentage
- VolumeConsumedReadWriteOps
- BurstBalance

在這個案例中，成本是根據定CloudWatch 價頁面和 SSM 定價頁面來計算：

- 自訂指標

  針對此案例，上述 13 個度量會發出給 CloudWatch 使用 CloudWatch 代理程式。這些指標會被視為自訂指標。每個自訂指標的費用為每月 0.3 USD。這些自訂指標的總費用為 13 * 0.3 USD = 每月 3.90 USD。
- 警示

  在這個案例中， CloudWatch 應用程式深入解析總共會監控 26 個指標，這會建立 26 個警示。每個警示的費用為每月 0.1 USD。警示的總費用為 26 * 0.1 USD = 每月 2.60 USD。
- 資料擷取和錯誤日誌

  資料擷取的成本為每 GB 0.05 USD，而 SQL Server 錯誤日誌的儲存費用為每 GB 0.03 USD。資料擷取和錯誤日誌的總成本為每 GB 0.05 USD + 每 GB 0.03 USD = 每 GB 0.08 USD。
- Amazon EC2 Systems Manager OpsItems

  SSM OpsItem 會針對 CloudWatch 應用程式深入解析偵測到的每個問題建立。如果您的應用程式中有 n 個問題，總成本為每月 0.00267 USD * n。

相關服務

下列服務與應用 CloudWatch 程式見解一起使用：

相關 AWS 服務

- Amazon CloudWatch 提供全系統的資源使用率、應用程式效能和營運狀態的能見度。它會收集並追蹤指標、傳送警示通知、根據您定義的規則自動更新您正在監視的資源，並允許您監視自己的自訂指標。 CloudWatch 應用程式深入解析是 CloudWatch透過 CloudWatch 預設作業儀表板內的特別來啟動。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。
- CloudWatch Contain@@ er Insights 會從您的容器化應用程式和微服務收集、彙總和摘要指標和記錄。您可以使用 Container Insights 來監控 Amazon ECS、Amazon Elastic Kubernetes Service 和 Amazon EC2 上的 Kubernetes 平台。Application Insights 在 Container Insights 或 Application Insights 主控台上啟用時，Application Insights 會在 Container Insights 儀表板上顯示偵測到的問題。如需詳細資訊，請參閱Container Insights。
- Amazon DynamoDB 是全受管的 NoSQL 資料庫服務，可讓您卸下操作及擴展分散式資料庫的管理負擔，不再需要煩惱硬體佈建、設定和組態、複寫、軟體修補或叢集擴展。DynamoDB 還提供靜態加密，解決了保護敏感資料所涉及的操作負擔和複雜性。
- Amazon EC2 在 AWS 雲端提供可擴展的運算容量。您可使用 Amazon EC2 按需要啟動任意數量的虛擬伺服器，設定安全性和聯網功能以及管理儲存。您可以擴展與縮減規模，以處理需求或熱門峰值的變更，從而降低您預測流量的需求。如需詳細資訊，請參閱 Amazon EC2 Linux 執行個體使用者指南或 Amazon EC2 Windows 執行個體使用者指南。
- Amazon Elastic Block Store (Amazon EBS) 提供區塊層級儲存體磁碟區，可搭配使用 Amazon EC2 執行個體。Amazon EBS 磁碟區的行為與未格式化的原始區塊型儲存設備相似。您可以將這些磁碟區作為裝置，掛載在您的執行個體上。連接至執行個體的 Amazon EBS 磁碟區將顯示為儲存體磁碟區，其可永久保留，不受執行個體的壽命影響。您可以在這些磁碟區上建立檔案系統，或利用您使用區塊型儲存裝置 (例如硬碟) 的任何方式來使用它們。您可以動態變更連接到執行個體的磁碟區組態。如需詳細資訊，請參閱 Amazon EBS 使用者指南。
- Amazon EC2 Auto Scaling 能確保您有正確的 EC2 執行個體數量可處理應用程式的負載。如需詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南。
- Elastic Load Balancing 會將傳入的應用程式或網路流量分配到多個可用區域的多個目標，例如 EC2 執行個體、容器和 IP 地址。如需詳細資訊，請參閱《Elastic Load Balancing 使用者指南》。
- IAM 是一種 Web 服務，可協助您安全地控制使用者對 AWS 資源的存取。使用 IAM 控制誰可以使用您的 AWS 資源（身份驗證），並控制他們可以使用的資源以及他們如何使用它們（授權）。如需詳細資訊，請參閱 Amazon 的身分驗證和存取控制 CloudWatch。
- AWS Lambda可讓您建置由事件觸發的函數組成的無伺服器應用程式，並使用 CodePipeline 和 AWS CodeBuild自動部署它們。如需詳細資訊，請參閱 AWS Lambda 應用程式。

- AWS Launch Wizard 適用於 SQL Server 可減少將 SQL Server 高可用性解決方案部署到雲端所需的時間。您可以在服務主控台上輸入應用程式需求，包括效能、節點數目和連線能力，並 AWS Launch Wizard 識別要部署和執行 SQL Server 永遠開啟應用程式的正確 AWS 資源。
- AWS Resource Groups 可協助您組織構成應用程式的資源。使用 Resource Groups，您可以一次管理和自動化大量資源的任務。一個應用程式只能註冊一個資源群組。如需詳細資訊，請參閱《AWS Resource Groups 使用者指南》。
- Amazon SQS 提供安全、耐用且可用的託管佇列，可讓您整合與分離分散式軟體系統和元件。如需詳細資訊，請參閱 Amazon SQS 使用者指南。
- AWS Step Functions是無伺服器函數撰寫器，可讓您將各種 AWS 服務和資源 (包括 AWS Lambda 函數) 排序到結構化的視覺化工作流程中。如需詳細資訊，請參閱 AWS Step Functions 使用者指南。
- AWS SSM 可 OpsItems 跨服務 OpsCenter 彙總和標準化，同時提供有關每個 OpsItem、相關資源和相關 OpsItems資源的情境調查資料。 OpsCenter 也提供 Systems Manager 自動化文件 (Runbook)，您可以使用這些文件快速解決問題。您可以為每個資料指定可搜尋的自訂資料 OpsItem。您還可以 OpsItems 按狀態和來源查看自動生成的摘要報告。如需詳細資訊，請參閱 AWS Systems Manager 使用者指南。
- Amazon API Gateway 是一種 AWS 用於建立、發佈、維護、監控和保護任何規模的 REST、HTTP 和 WebSocket API 的服務。API 開發人員可以創建訪問 AWS 或其他 Web 服務的 API，以及存儲在 AWS 雲中的數據。如需詳細資訊，請參閱《Amazon API Gateway 使用者指南。

> **Note**
>
> Application Insights 僅支援 REST API 通訊協定 (API Gateway 服務的第 1 版)。

- Amazon Elastic Container Service (Amazon ECS) 是全受管容器協同運作服務。您可以使用 Amazon ECS 執行最敏感和任務關鍵型應用程式。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》。
- Amazon Elastic Kubernetes Service (Amazon EKS) 是一項受管服務，您可以使用它來執行 Kubernetes， AWS 而無需安裝、操作和維護自己的 Kubernetes 控制平面或節點。Kubernetes 是一套開放原始碼系統，用於容器化應用程式的自動化部署、擴展與管理。如需詳細資訊，請參閱《Amazon EKS 使用者指南》。
- Amazon EC2 上的 Kubernetes。Kubernetes 是開放原始碼軟體，可協助您大規模部署及管理容器化應用程式。Kubernetes 可管理 Amazon EC2 運算執行個體的叢集，並在這些執行個體上執行容器，其中包含部署、維護和擴展程序。藉助 Kubernetes，您可以在內部部署和雲端使用相同的工具集來執行各種類型的容器化應用程式。如需詳細資訊，請參閱 Kubernetes 文件：入門。

- Amazon FSx 可協助您啟動和執行由 AWS全受管的熱門檔案系統。使用 Amazon FSx，您可以利用常見的開放原始碼和商業授權檔案系統的功能集與效能，從而避免費時的管理任務。如需詳細資訊，請參閱 Amazon FSx 說明文件。
- Amazon Simple Notification Service (SNS) 是一種用於 application-to-application 和 application-to-person 通訊的全受管簡訊服務。您可以設定 Amazon SNS 以透過 Application Insights 進行監控。當 Amazon SNS 設定為用於監控的資源時，Application Insights 會追蹤 SNS 指標，以協助判斷 SNS 訊息可能會遇到問題或失敗的原因。
- Amazon Elastic File System (Amazon EFS) 是全受管的彈性 NFS 檔案系統，可搭配 AWS 雲端 服務和現場部署資源使用。該系統建置為可根據需求擴展到 PB 級，而不會中斷應用程式。儲存容量會在您新增和移除的檔案時自動擴展和縮減，讓您無需佈建和管理容量以配合擴展。如需詳細資訊，請參閱 Amazon Elastic File System 文件。

相關第三方服務

- 對於「應用程式深入解析」中監控的某些工作負載和應用程式，Prometheus JMX 匯出程式會使用「 AWS Systems Manager 散發者」安裝，以便「應用 CloudWatch 程式深入解析」可擷取 Java 當您選擇監控 Java 應用程式時，Application Insights 會自動為您安裝 Prometheus JMX Exporter。

支援的應用程式元件

CloudWatch 應用程式深入解析會掃描您的資源群組，以識別應用 元件可為獨立、自動分組 (例如 Auto Scaling 群組中或負載平衡器後端的執行個體) 或自訂 (將個別的 Amazon EC2 執行個體集合在一起)。

「 CloudWatch 應用程式深入解析」支援下列元件：

AWS 元件

- Amazon EC2
- Amazon EBS
- Amazon RDS
- Elastic Load Balancing：Application Load Balancer 和 Classic Load Balancer (這些負載平衡器的所有目標執行個體都會予以識別並設定)。
- Amazon EC2 Auto Scaling 群組：自動擴展 (AWS Auto Scaling 群組會針對所有目標執行個體動態設定；如果您的應用程式擴展， CloudWatch 應用程式深入解析會自動設定新執行個體)。CloudFormation 堆疊式資源群組不支援自動調整比例群組。
- AWS Lambda

- Amazon Simple Queue Service (Amazon SQS)
- Amazon DynamoDB 資料表
- Amazon S3 儲存貯體指標
- AWS Step Functions
- Amazon API Gateway REST API 階段
- Amazon Elastic Container Service (Amazon ECS)：叢集、服務和任務
- Amazon Elastic Kubernetes Service (Amazon EKS)：叢集
- Amazon EC2 上的 Kubernetes：在 EC2 上執行的 Kubernetes 叢集
- Amazon SNS 主題

任何其他元件類型資源目前未由「 CloudWatch 應用程式見解」追蹤。如果您的 Application Insights 應用程式不顯示支援的元件類型，此元件可能已註冊，並由 Application Insights 監控的您其他應用程式所管理。

支援的技術堆疊

您可以使用「應用 CloudWatch 程式深入解析」來監視在 Windows Server 和 Linux 作業系統上執行的應用程式，方法是針對下列其中一種技術選取應用程式層下拉式功能表選項：

- 前端：Microsoft Internet Information Services (IIS) Web 伺服器
- 工作者層：
  - .NET Framework
  - .NET Core
- 應用程式：
  - Java
  - SAP NetWeaver 標準、分散式和高可用性部署
- Active Directory
- SharePoint
- 資料庫：
  - 在 Amazon RDS 或 Amazon EC2 上執行的 Microsoft SQL Server (包括 SQL Server 高可用性組態。請參閱「元件組態範例」)。
  - 在 Amazon RDS、Amazon Aurora 或 Amazon EC2 上執行的 MySQL

- 在 Amazon RDS 或 Amazon EC2 上執行的 PostgreSQL
- Amazon DynamoDB 資料表
- 在 Amazon RDS 或 Amazon EC2 上執行的 Oracle
- 在單一 Amazon EC2 執行個體和多個 EC2 執行個體上的 SAP HANA 資料庫
- 跨可用區 SAP HANA 資料庫的高可用性設定
- 單一 Amazon EC2 執行個體上的 SAP 系統庫 ASE 資料庫
- 跨可用區域 SAP 系統庫 ASE 資料庫的高可用性設定

如果上面列出的任何技術堆疊都不適用於您的應用程式資源，您可以從 Manage monitoring (管理監控) 頁面上的應用程式層下拉式功能表中選擇 Custom (自訂)，進而監控您的應用程式堆疊。

Amazon 應 CloudWatch 用程式洞察如何運作

本節包含「 CloudWatch 應用程式見解」如何運作的相關資訊，包括：

- Application Insights 如何監控應用程式
- 資料保留
- 配額
- AWS CloudWatch 應用程式洞察所使用的 Systems Manager (SSM) 套件
- AWS CloudWatch 應用程式見解所使用的 Systems Manager (SSM) 文件

Application Insights 如何監控應用程式

Application Insights 會監控應用程式如下。

Application Discovery 和組態

第一次將應用程式新增至 Ap CloudWatch plication Insights 時，它會掃描應用程式元件，以建議要監視應用程式的關鍵指標、記錄和其他資料來源。然後，您就可以根據這些建議來設定您的應用程式。

資料預先處理

CloudWatch Application Insights 會持續分析整個應用程式資源所監控的資料來源，以發現量度異常並記錄錯誤 (觀察)。

智慧型問題偵測

CloudWatch 應用程式深入解析引擎會使用分類演算法和內建規則來關聯觀察，藉此偵測應用程式中的問題。為了協助進行疑難排解，它會建立自動化 CloudWatch 儀表板，其中包含有關問題的內容資訊。

提醒和動作

當 CloudWatch 應用程式深入解析偵測到應用程式的問題時，就會產生 CloudWatch 事件以通知您問題。如需如何設定這些事件的詳細資訊，請參閱 應用程式見解偵測到問題的 CloudWatch 事件和通知。

範例藍本

您有 SQL Server 資料庫支援的 ASP .NET 應用程式。突然間，您的資料庫因記憶體高壓開始故障。這可能會導致應用程式效能降級，而且 Web 伺服器和負載平衡器可能發生 HTTP 500 錯誤。

使用 Ap CloudWatch plication Insights 及其智慧型分析，您可以檢查動態建立的儀表板 (顯示相關指標和記錄檔片段)，以識別造成問題的應用程式層。在本例中，問題可能發生在 SQL 資料庫 layer。

資料保留

CloudWatch 應用程式洞察會保留問題 55 天，觀察 60 天。

配額

如需 CloudWatch 應用程式見解的預設配額，請參閱 Amazon CloudWatch 應用程式洞察端點和配額。除非另有說明，否則每個配額均為每個 AWS 區域。請聯絡 AWS Support 請求提高您的服務配額。許多服務包含的配額無法變更。如需有關特定服務的配額詳細資訊，請參閱該服務的文件。

AWS CloudWatch 應用程式洞察所使用的 Systems Manager (SSM) 套件

本節中列出的套件是由「應用程式深入解析」所使用，而且可以透過 AWS Systems Manager 發行者獨立管理和部署。如需 SSM Distributor 的詳細資訊，請參閱《AWS Systems Manager 使用者指南》中的 AWS Systems Manager Distributor。

套件：

- AWSObservabilityExporter-JMXExporterInstallAndConfigure
- AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure
- AWSObservabilityExporter-HAClusterExporterInstallAndConfigure
- AWSObservabilityExporter-SAP-SAPHostExporterInstallAndConfigure
- AWSObservabilityExporter-SQLExporterInstallAndConfigure

AWSObservabilityExporter-JMXExporterInstallAndConfigure

您可以從 Prometheus JMX Exporter 為 Application Insights 擷取工作負載特定的 Java 指標，進而設定和監控警示。在 Application Insights 主控台的 Manage monitoring (管理監控) 頁面，從 Application tier (應用程式層) 下拉式清單選取 JAVA 應用程式。然後，在 JAVA Prometheus exporter configuration (JAVA Prometheus 匯出程式組態) 下，選擇 Collection method (集合方法) 和 JMX port number (JMX 連接埠號碼)。

若要使用 AWS Systems Manager 散發程式來封裝、安裝和設定 AWS提供的 Prometheus JMX 匯出程式套件，而不受應用程式深入分析影響，請完成下列步驟。

使用 Prometheus JMX Exporter SSM 套件的先決條件

- 已安裝 SSM agent 2.3.1550.0 版或更新版本
- 已設定 JAVA_HOME 環境變數

安裝和設定 AWSObservabilityExporter-JMXExporterInstallAndConfigure 套件

AWSObservabilityExporter-JMXExporterInstallAndConfigure 封裝是一種 SSM Distributor 套件，您可將其用於安裝和設定 Prometheus JMX Exporter。當 Prometheus JMX 匯出程式傳送 Java 測量結果時，可將 CloudWatch 代理程式設定為擷取服務的測量結果。 CloudWatch

1. 根據您的喜好設定，準備位於 Prometheus 儲存庫中的 Prometheus JMX 匯出程式 YAML 設定檔。GitHub 使用範例組態和選項描述來引導您。
2. 將編碼為 Base64 的 Prometheus JMX Exporter YAML 組態檔案複製到 SSM 參數存放區中的新 SSM 參數。
3. 導覽至 SSM Distributor 主控台，並開啟 Owned by Amazon (由 Amazon 擁有) 索引標籤。選擇 AWSObservabilityExporter-JMX ExporterInstallAndConfigure，然後選擇安裝一次。
4. 以下列方式取代「其他引數」，更新您在第一個步驟中建立的 SSM 參數：

```
{
  "SSM_EXPORTER_CONFIGURATION": "{{ssm:<SSM_PARAMETER_STORE_NAME>}}",
  "SSM_EXPOSITION_PORT": "9404"
}
```

> **Note**
>
> 連接埠 9404 是用來傳送 Prometheus JMX 指標的預設連接埠。您可以更新此連接埠。

範例：設定 CloudWatch 代理程式以擷取 Java 測量結果

1. 安裝 Prometheus JMX Exporter，如上列程序所述。然後透過檢查連接埠狀態來驗證它是否正確安裝在您的執行個體上。

   在 Windows 執行個體上成功安裝範例

   ```
   PS C:\> curl http://localhost:9404 (http://localhost:9404/)
   StatusCode : 200
   StatusDescription : OK
   Content : # HELP jvm_info JVM version info
   ```

   在 Linux 執行個體上成功安裝範例

   ```
   $ curl localhost:9404
   # HELP jmx_config_reload_failure_total Number of times configuration have failed to
    be reloaded.
   # TYPE jmx_config_reload_failure_total counter
   jmx_config_reload_failure_total 0.0
   ```

2. 建立 Prometheus 服務探索 YAML 檔案。下列範例服務探索檔案會執行下列操作：

   - 將 Prometheus JMX Exporter 主機連接埠指定為 `localhost: 9404`。
   - 將標籤 (`ApplicationComponentName`、和`InstanceId`) 附加至量度，可設定為 CloudWatch 量度維度。

   ```
   $ cat prometheus_sd_jmx.yaml
   - targets:
     - 127.0.0.1:9404
     labels:
       Application: myApp
       ComponentName: arn:aws:elasticloadbalancing:us-
   east-1:123456789012:loadbalancer/app/sampl-Appli-MMZW8E3GH4H2/aac36d7fea2a6e5b
       InstanceId: i-12345678901234567
   ```

3. 建立 Prometheus JMX Exporter 組態 YAML 檔案。以下範例組態檔案會記錄以下內容：

   - 指標擷取任務間隔和逾時期間。
   - 指標擷取任務 (`jmx` 和 `sap`)，也稱為抓取，其中包括任務名稱、一次傳回的時間序列上限，以及服務探索檔案路徑。

```
$ cat prometheus.yaml
global:
  scrape_interval: 1m
  scrape_timeout: 10s
scrape_configs:
  - job_name: jmx
    sample_limit: 10000
    file_sd_configs:
      - files: ["/tmp/prometheus_sd_jmx.yaml"]
  - job_name: sap
    sample_limit: 10000
    file_sd_configs:
      - files: ["/tmp/prometheus_sd_sap.yaml"]
```

4. 確認 CloudWatch 代理程式已安裝在您的 Amazon EC2 執行個體上，且版本為 1.247346.1b249759 或更新版本。若要在 EC2 執行個體上安裝 CloudWatch代理程式，請參閱安裝 CloudWatch 代理程式。如果要驗證版本，請參閱尋找 CloudWatch 代理程式版本的相關資訊。
5. 設定代 CloudWatch 理程式。如需如何設定 CloudWatch 代理程式組態檔的相關資訊，請參閱手動建立或編輯 CloudWatch 代理程式組態檔。下列範例 CloudWatch 代理程式組態檔會執行下列動作：

   - 指定 Prometheus JMX Exporter 組態 YAML 檔案路徑。
   - 指定要發佈 EMF 指標日誌的目標日誌群組。
   - 為每個指標名稱指定兩組維度。
   - 傳送 8 個 (4 個度量名稱 * 每個度量名稱 2 組維 CloudWatch 度) 量度。

```
{
   "logs":{
      "logs_collected":{
         ....
      },
      "metrics_collected":{
         "prometheus":{
            "cluster_name":"prometheus-test-cluster",
            "log_group_name":"prometheus-test",
            "prometheus_config_path":"/tmp/prometheus.yaml",
            "emf_processor":{
               "metric_declaration_dedup":true,
```

```
                "metric_namespace":"CWAgent",
                "metric_unit":{
                   "jvm_threads_current":"Count",
                   "jvm_gc_collection_seconds_sum":"Second",
                   "jvm_memory_bytes_used":"Bytes"
                },
                "metric_declaration":[
                   {
                      "source_labels":[
                         "job"
                      ],
                      "label_matcher":"^jmx$",
                      "dimensions":[
                         [
                            "InstanceId",
                            "ComponentName"
                         ],
                         [
                            "ComponentName"
                         ]
                      ],
                      "metric_selectors":[
                         "^java_lang_threading_threadcount$",
                         "^java_lang_memory_heapmemoryusage_used$",
                         "^java_lang_memory_heapmemoryusage_committed$"
                      ]
                   }
                ]
             }
          }
       }
    },
    "metrics":{
       ....
    }
}
```

AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure

您可以從 Prometheus HANA 資料庫匯出工具為 Application Insights 擷取工作負載特定的 SAP HANA 指標，進而設定和監控警示。如需詳細資訊，請參閱本指南中的 設定您的 SAP HANA 資料庫以進行監控。

若要使用 AWS Systems Manager 散發程式來封裝、安裝及設定 AWS提供的 Prometheus HANA 資料庫匯出程式套件，而不受應用程式見解影響，請完成下列步驟。

使用 Prometheus HANA 資料庫匯出工具 SSM 套件的先決條件

- 已安裝 SSM agent 2.3.1550.0 版或更新版本
- SAP HANA 資料庫
- Linux 作業系統 (軟體 RedHat)
- 具有 SAP HANA 資料庫監控憑證的機密，使用 AWS Secrets Manager。使用索引鍵/值配對格式建立機密、指定金鑰使用者名稱，然後輸入值的資料庫使用者。新增第二個金鑰密碼，然後針對值輸入密碼。如需建立機密的詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的建立機密。機密的格式必須如下：

```
{
  "username": "<database_user>",
  "password": "<database_password>"
}
```

安裝和設定 **AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure** 套件

AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure 封裝是一種 SSM Distributor 套件，您可將其用於安裝和設定 Prometheus HANA 資料庫匯出工具。由 Prometheus HANA 資料庫匯出程式傳送 HANA 資料庫測量結果時，可以設定 CloudWatch 代理程式擷取服務的測量結果。 CloudWatch

1. 在 SSM 參數存放區中建立 SSM 參數，以存放匯出工具組態。以下是範例參數值。

```
{\"exposition_port\":9668,\"multi_tenant\":true,\"timeout\":600,\"hana\":{\"host\":
\"localhost\",\"port\":30013,\"aws_secret_name\":\"HANA_DB_CREDS\",\"scale_out_mode
\":true}}
```

> **Note**
>
> 在此範例中，匯出僅在具有作用中 SYSTEM 資料庫的 Amazon EC2 執行個體上執行，並且其將在其他 EC2 執行個體上保持閒置狀態，以避免重複指標。匯出工具可以從 SYSTEM 資料庫擷取所有資料庫租用戶資訊。

2. 在 SSM 參數存放區中建立 SSM 參數，以存放匯出工具指標查詢。套件可以接受多個指標參數。每個參數都必須具有有效的 JSON 物件格式。以下是範例參數值：

```
{\"SELECT MAX(TIMESTAMP) TIMESTAMP, HOST, MEASURED_ELEMENT_NAME CORE,
 SUM(MAP(CAPTION, 'User Time', TO_NUMBER(VALUE), 0)) USER_PCT, SUM(MAP(CAPTION,
 'System Time', TO_NUMBER(VALUE), 0)) SYSTEM_PCT, SUM(MAP(CAPTION, 'Wait
 Time', TO_NUMBER(VALUE), 0)) WAITIO_PCT, SUM(MAP(CAPTION, 'Idle Time', 0,
 TO_NUMBER(VALUE))) BUSY_PCT, SUM(MAP(CAPTION, 'Idle Time', TO_NUMBER(VALUE), 0))
 IDLE_PCT FROM sys.M_HOST_AGENT_METRICS WHERE MEASURED_ELEMENT_TYPE = 'Processor'
 GROUP BY HOST, MEASURED_ELEMENT_NAME;\":{\"enabled\":true,\"metrics\":[{\"name\":
\"hanadb_cpu_user\",\"description\":\"Percentage of CPU time spent by HANA DB in user
 space, over the last minute (in seconds)\",\"labels\":[\"HOST\",\"CORE\"],\"value\":
\"USER_PCT\",\"unit\":\"percent\",\"type\":\"gauge\"},{\"name\":\"hanadb_cpu_system
\",\"description\":\"Percentage of CPU time spent by HANA DB in Kernel space,
 over the last minute (in seconds)\",\"labels\":[\"HOST\",\"CORE\"],\"value\":
\"SYSTEM_PCT\",\"unit\":\"percent\",\"type\":\"gauge\"},{\"name\":\"hanadb_cpu_waitio
\",\"description\":\"Percentage of CPU time spent by HANA DB in IO mode, over the
 last minute (in seconds)\",\"labels\":[\"HOST\",\"CORE\"],\"value\":\"WAITIO_PCT\",
\"unit\":\"percent\",\"type\":\"gauge\"},{\"name\":\"hanadb_cpu_busy\",\"description
\":\"Percentage of CPU time spent by HANA DB, over the last minute (in seconds)\",
\"labels\":[\"HOST\",\"CORE\"],\"value\":\"BUSY_PCT\",\"unit\":\"percent\",\"type\":
\"gauge\"},{\"name\":\"hanadb_cpu_idle\",\"description\":\"Percentage of CPU time not
 spent by HANA DB, over the last minute (in seconds)\",\"labels\":[\"HOST\",\"CORE
\"],\"value\":\"IDLE_PCT\",\"unit\":\"percent\",\"type\":\"gauge\"}]}}
```

有關指標查詢的更多信息，請參SUSE / hanadb_exporter閱上的 GitHub.

3. 導覽至 SSM Distributor 主控台，並開啟 Owned by Amazon (由 Amazon 擁有) 索引標籤。選擇 AWSObservabilityExporter-SAP-HANADB ExporterInstallAndConfigure * 並選擇安裝一次。
4. 以下列方式取代「其他引數」，更新您在第一個步驟中建立的 SSM 參數：

```
{
  "SSM_EXPORTER_CONFIG": "{{ssm:<*SSM_CONFIGURATIONS_PARAMETER_STORE_NAME>*}}",
  "SSM_SID": "<SAP_DATABASE_SID>",
  "SSM_EXPORTER_METRICS_1": "{{ssm:<SSM_FIRST_METRICS_PARAMETER_STORE_NAME>}}",
  "SSM_EXPORTER_METRICS_2": "{{ssm:<SSM_SECOND_METRICS_PARAMETER_STORE_NAME>}}"
}
```

5. 選取具有 SAP HANA 資料庫的 Amazon EC2 執行個體，然後選擇 Run (執行)。

AWSObservabilityExporter-HAClusterExporterInstallAndConfigure

您可以從 Prometheus HANA 叢集匯出工具為 Application Insights 擷取工作負載特定的高可用性 (HA) 叢集指標，進而設定和監控警示，以進行 SAP HANA 資料庫高可用性設定。如需詳細資訊，請參閱本指南中的 設定您的 SAP HANA 資料庫以進行監控。

若要使用 AWS Systems Manager 散發程式來封裝、安裝及設定 AWS提供的 Prometheus HA 叢集匯出程式套件，而不受應用程式見解影響，請完成下列步驟。

使用 Prometheus HA 叢集匯出工具 SSM 套件的先決條件

- 已安裝 SSM agent 2.3.1550.0 版或更新版本
- 適用於 Pacemaker、Corosync、SBD 和 DRBD 的 HA 叢集
- Linux 作業系統 (軟體 RedHat)

安裝和設定 AWSObservabilityExporter-HAClusterExporterInstallAndConfigure 套件

AWSObservabilityExporter-HAClusterExporterInstallAndConfigure 封裝是一種 SSM Distributor 套件，您可將其用於安裝和設定 Prometheus HA Cluster Exporter。Prometheus HANA 資料庫匯出程式傳送叢集測量結果時，可設定 CloudWatch 代理程式以擷取服務的測量結果。CloudWatch

1. 在 SSM 參數存放區中建立 SSM 參數，以使用 JSON 格式存放匯出工具組態。以下是範例參數值。

   ```
   {\"port\":\"9664\",\"address\":\"0.0.0.0\",\"log-level\":\"info\",\"crm-mon-path
   \":\"/usr/sbin/crm_mon\",\"cibadmin-path\":\"/usr/sbin/cibadmin\",\"corosync-
   cfgtoolpath-path\":\"/usr/sbin/corosync-cfgtool\",\"corosync-quorumtool-path\":\"/
   usr/sbin/corosync-quorumtool\",\"sbd-path\":\"/usr/sbin/sbd\",\"sbd-config-path\":
   \"/etc/sysconfig/sbd\",\"drbdsetup-path\":\"/sbin/drbdsetup\",\"enable-timestamps
   \":false}
   ```

   有關導出器配置的更多信息，請參ClusterLabs / ha_cluster_exporter閱上的 GitHub.
2. 導覽至 SSM Distributor 主控台，並開啟 Owned by Amazon (由 Amazon 擁有) 索引標籤。選取 AWSObservabilityExporter-HA ClusterExporterInstallAndConfigure *，然後選擇 [安裝一次]。
3. 以下列方式取代「其他引數」，更新您在第一個步驟中建立的 SSM 參數：

   ```
   {
     "SSM_EXPORTER_CONFIG": "{{ssm:<*SSM_CONFIGURATIONS_PARAMETER_STORE_NAME>*}}"
   }
   ```

4. 選取具有 SAP HANA 資料庫的 Amazon EC2 執行個體，然後選擇 Run (執行)。

AWSObservabilityExporter-SAP-SAPHostExporterInstallAndConfigure

您可以從 Prometheus SAP 主機匯出程式擷取工作負載特定的 SAP NetWeaver 指標，以取得「應用程式見解」，以設定和監控 SAP NetWeaver 分散式和高可用性部署的警示。如需詳細資訊，請參閱開始使 CloudWatch 用 Amazon 應用程式深入解析。

若要使用 AWS Systems Manager Distributor 封裝、安裝和設定 SAP 主機匯出程式套件 (獨立於 Application Insights)，請完成下列步驟。

使用 Prometheus SAP 主機匯出程式 SSM 套件的先決條件

- 已安裝 SSM agent 2.3.1550.0 版或更新版本
- SAP NetWeaver 應用伺服器
- Linux 作業系統 (軟體 RedHat)

安裝和設定 **AWSObservabilityExporter-SAP-SAPHostExporterInstallAndConfigure** 套件

此AWSObservabilityExporter-SAP-SAPHostExporterInstallAndConfigure套件是 SSM 代理商套件，您可以用來安裝和設定 SAP NetWeaver Prometheus 指標匯出程式。當 Prometheus 匯出程式傳送 SAP NetWeaver 指標時，可將 CloudWatch 代理程式設定為擷取服務的指標。CloudWatch

1. 在 SSM 參數存放區中建立 SSM 參數，以使用 JSON 格式存放匯出工具組態。以下是範例參數值。

```
{\"address\":\"0.0.0.0\",\"port\":\"9680\",\"log-level\":\"info\",\"is-HA\":false}
```

- address

  將 Prometheus 指標傳送到的目標地址。預設值為 `localhost`。

- port

  將 Prometheus 指標傳送到的目標連接埠。預設值為 `9680`。

- is-HA

  `true`適用於 SAP NetWeaver 高可用性部署。對於所有其他部署，為 `false`。

2. 導覽至 SSM Distributor 主控台，並開啟 Owned by Amazon (由 Amazon 擁有) 索引標籤。選取 AWSObservabilityExporter-SAP-SAP HostExporterInstallAndConfigure，然後選擇一次安裝。
3. 以下列方式取代「其他引數」，更新您在第一個步驟中建立的 SSM 參數：

```
{
  "SSM_EXPORTER_CONFIG": "{{ssm:<SSM_CONFIGURATIONS_PARAMETER_STORE_NAME>}}",
  "SSM_SID": "<SAP_DATABASE_SID>",
  "SSM_INSTANCES_NUM": "<instances_number seperated by comma>"
}
```

範例

```
{
  "SSM_EXPORTER_CONFIG": "{{ssm:exporter_config_paramter}}",
  "SSM_INSTANCES_NUM": "11,12,10",
  "SSM_SID": "PR1"
}
```

4. 選取具有 SAP NetWeaver 應用程式的 Amazon EC2 執行個體，然後選擇執行。

> **Note**
>
> Prometheus 匯出程式會在本機端點上為 SAP NetWeaver 指標提供服務。本機端點只能由 Amazon EC2 執行個體上的作業系統使用者存取。因此，在安裝匯出程式套件之後，所有作業系統使用者都可以使用這些指標。預設本機端點為 `localhost:9680/metrics`。

AWSObservabilityExporter-SQLExporterInstallAndConfigure

您可以從 Prometheus SQL 匯出程式中為 Application Insights 擷取工作負載特定的 SQL Server 指標，進而監控關鍵指標。

若要使用 AWS Systems Manager Distributor 封裝、安裝和設定 SQL 匯出程式套件 (獨立於 Application Insights)，請完成下列步驟。

使用 Prometheus SQL Exporter SSM 套件的先決條件

- 已安裝 SSM agent 2.3.1550.0 版或更新版本
- 在啟用 SQL Server 使用者身分驗證的 Windows 中執行 SQL Server 的 Amazon EC2 執行個體。

- 具有下列許可的 SQL Server 使用者：

```
GRANT VIEW ANY DEFINITION TO
```

```
GRANT VIEW SERVER STATE TO
```

- 使用 AWS Secrets Manager包含資料庫連線字串的機密。如需建立機密的詳細資訊，請參閱《AWS Secrets Manager 使用者指南》中的建立機密。機密的格式必須如下：

```
{
  "data_source_name":"sqlserver://<username>:<password>@localhost:1433"
}
```

> **Note**
>
> 如果密碼或使用者名稱包含特殊字元，則必須對特殊字元進行百分比編碼，以確保資料庫連線成功。

安裝和設定 **AWSObservabilityExporter-SQLExporterInstallAndConfigure** 套件

AWSObservabilityExporter-SQLExporterInstallAndConfigure 套件是 SSM Distributor 套件，可用於安裝和設定 SQL Prometheus 指標匯出程式。當 Prometheus 匯出程式傳送指標時，可將 CloudWatch 代理程式設定為擷取服務的指標。 CloudWatch

1. 根據您的偏好設定，準備 SQL Exporter YAML 組態。下列範例組態已設定單一指標。使用範例組態以利用其他指標來更新組態或建立您自己的組態。

```
---
global:
  scrape_timeout_offset: 500ms
  min_interval: 0s
  max_connections: 3
  max_idle_connections: 3
target:
  aws_secret_name: <SECRET_NAME>
  collectors:
    - mssql_standard
collectors:
  - collector_name: mssql_standard
```

```
    metrics:
    - metric_name: mssql_batch_requests
      type: counter
      help: 'Number of command batches received.'
      values: [cntr_value]
      query: |
        SELECT cntr_value
        FROM sys.dm_os_performance_counters WITH (NOLOCK)
        WHERE counter_name = 'Batch Requests/sec'
```

2. 將編碼為 Base64 的 Prometheus SQL Exporter YAML 組態檔案複製到 SSM 參數存放區中的新 SSM 參數。
3. 導覽至 SSM Distributor 主控台，並開啟 Owned by Amazon (由 Amazon 擁有) 索引標籤。選取 [AWSObservabilityExporter-SQL]ExporterInstallAndConfigure，然後選擇 [安裝一次]。
4. 使用下列資訊取代「其他引數」。SSM_PARAMETER_NAME 是您在步驟 2 中建立的參數名稱。

```
{
  "SSM_EXPORTER_CONFIGURATION":
    "{{ssm:<SSM_PARAMETER_STORE_NAME>}}",
    "SSM_PROMETHEUS_PORT": "9399",
    "SSM_WORKLOAD_NAME": "SQL"
}
```

5. 選取帶有 SQL Server 資料庫的 Amazon EC2 執行個體，然後選擇執行。

AWS CloudWatch 應用程式見解所使用的 Systems Manager (SSM) 文件

Application Insights 使用本節中列出的 SSM 文件來定義 AWS Systems Manager 在受控執行個體上執行的動作。這些文件使用 Systems Manager 的 Run Command 功能，將執行 Application Insights 監控功能所需的任務自動化。這些文件的執行排程由 Application Insights 維護，且無法變更。

如需 SSM 文件的詳細資訊，請參閱《AWS Systems Manager 使用者指南》中的 AWS Systems Manager 文件。

由 CloudWatch 應用程式深入解析管理

下表列出 Application Insights 管理的 SSM 文件。

文件名稱	描述	執行排程
AWSEC2-DetectWorkload	自動偵測應用程式環境中執行的應用程式，這些應用程式可設定為由 Application Insights 監控。	本文件會在您的應用程式環境中每小時執行一次，以取得 up-to-date 應用
AWSEC2-CheckPerformanceCounterSets	檢查 Amazon EC2 Windows 執行個體是否已啟用效能計數器命名空間。	本文件會在您的應用程式環境中每小時執行一次，而且只有在啟用對應的命名空間時才會監控效能計數器指標。
AWSEC2-ApplicationInsightsCloudwatchAgentInstallAndConfigure	根據應用程式元件的監視組態安裝和設定 CloudWatch 代理程式。	本文件每 30 分鐘執行一次，以確保 CloudWatch 代理程式組態永遠正確無誤且 up-to-date。此文件也會在對應用程式監控設定 (例如新增或移除指標或更新日誌組態) 進行變更後立即執行。

管理的文件 AWS Systems Manager

下面的文件是由 CloudWatch 應用程序洞察使用，並通過系統管理器管理。

AWS-ConfigureAWSPackage

應用程式深入解析使用本文件來安裝和解除安裝 Prometheus 匯出程式代理商套件、收集工作負載特定指標，以及對客戶 Amazon EC2 執行個體上的工作負載進行全面監控。 CloudWatch 只有在您的執行個體上執行相關的目標工作負載時，應用程式深入解析才會安裝 Prometheus 匯出程式散發商套件。

下表列出 Prometheus 匯出程式經銷商套件及相關的目標工作負載。

Prometheus 匯出程式經銷商套件名稱	目標工作負載
AWSObservabilityExporter-HAClusterExporterInstallAndConfigure	SAP HANA HA

Prometheus 匯出程式經銷商套件名稱	目標工作負載
AWSObservabilityExporter-JMXExporterInstallAndConfigure	Java/JMX
AWSObservabilityExporter-SAP-HANADBExporterInstallAndConfigure	SAP HANA
AWSObservabilityExporter-SAP-SAPHostExporterInstallAndConfigure	NetWeaver
AWSObservabilityExporter-SQLExporterInstallAndConfigure	SQL Server (Windows) 和 SAP ASE (Linux)

AmazonCloudWatch-ManageAgent

應用程式深入解析使用本文件來管理執行個體上 CloudWatch 代理程式的狀態和組態，並從不同作業系統的 Amazon EC2 執行個體收集內部系統層級指標和日誌。

開始使 CloudWatch 用 Amazon 應用程式深入解析

若要開始使 CloudWatch 用應用程式深入解析，請確認您已符合下列先決條件，並已建立 IAM 政策。然後，您可以使用主控台連結開始啟用應用 CloudWatch 程式深入解析。若要設定您的應用程式資源，請依照安裝、設定及管理應用程式以進行監控下的步驟執行作業。

目錄

- 存取 CloudWatch 應用程式深
- 必要條件
- IAM 政策
- 帳戶型應用程式上架的 IAM 角色許可
- 安裝、設定及管理應用程式以進行監控

存取 CloudWatch 應用程式深

您可以透過下列其中一個介面存取和管理「 CloudWatch 應用程式見解」：

- CloudWatch 控制台。若要為應用程式新增監視器，請在CloudWatch 主控台左側導覽窗格中的 [見解] 下選擇 [應用程式深入解析] 設定應用程式之後，您可以使用主CloudWatch 控台來檢視和分析偵測到的問題。
- AWS 命令行界面（AWS CLI）。您可以使用 AWS CLI 來存取 AWS API 作業。若要取得更多資訊，請參閱《指 AWS 令行介面使用指南》中的〈安裝指AWS 令行介面〉。如需應用程式洞察 API 資訊，請參閱 Amazon CloudWatch 應用程式洞察 API 參考。

必要條件

您必須完成下列先決條件，才能使用應用程式深入解析來設定 CloudWatch 應用

- AWS Systems Manager 啟用 — 在 Amazon EC2 執行個體上安裝系統管理員代理程式 (SSM 代理程式)，並啟用 SSM 的執行個體。如需有關如何安裝 SSM Agent 的資訊，請參閱《AWS Systems Manager 使用者指南》中的設定 AWS Systems Manager。
- EC2 執行個體角色 — 您必須連接下列 Amazon EC2 執行個體角色才能啟用 Systems Manager
  - 您必須連接 `AmazonSSMManagedInstanceCore` 角色，以啟用 Systems Manager。如需詳細資訊，請參閱 AWS Systems Manager 以身分為基礎的政策範例。
  - 您必須附加`CloudWatchAgentServerPolicy`原則，才能啟用透過 CloudWatch發出執行個體指標和記錄檔。如需詳細資訊，請參閱建立 IAM 角色和使用者以搭配 CloudWatch 代理程式使用。
- AWS 資源群組 — 若要將您的應用程式上架至「應用 CloudWatch 程式見解」，請建立一個資源群組，其中包含應用程式堆疊所使用的所有相關 AWS 資源。這包括 Application Load Balancer、執行 IIS 和網路前端的 Amazon EC2 執行個體、.NET 工作者層和 SQL Server 資料庫。如需應用程式深入解析支援的應用程式元件和技術堆疊的詳細資訊，請參閱 支援的應用程式元件 CloudWatch 應用程式深入解析會自動包含使用與資源群組相同的標籤或 CloudFormation 堆疊的 Auto Scaling 群組，因為資 CloudFormation 源群組不支援 Auto Scaling 群組。如需詳細資訊，請參閱《 AWS Resource Groups 入門》。
- IAM 許可 — 對於沒有管理存取權的使用者，您必須建立 AWS Identity and Access Management (IAM) 政策，以允許應用程式洞察建立服務連結角色，並將其附加到使用者的身分識別。如需有關如何建立 IAM 政策的詳細資訊，請參閱 IAM 政策。
- 服務連結角色 — 應用程式深入解析使用 AWS Identity and Access Management (IAM) 服務連結角色。系統會在您於 Application Insights 主控台建立第一個 Application Insights 應用程式時，為您建立服務連結角色。如需詳細資訊，請參閱 將服務連結角色用於 CloudWatch 應用程式深入。
- 效能計數器指標支援 EC2 Windows 執行個體 — 若要監控 Amazon EC2 Windows 執行個體上的效能計數器指標，執行個體上必須安裝效能計數器。如需效能計數器指標和對應的效能計數器集名稱，請參閱效能計數器指標。如需效能計數器的詳細資訊，請參閱效能計數器。

- Amazon CloudWatch 代理程式 — 應用程式洞察會安裝和設定代理程式 CloudWatch 。如果您已安裝 CloudWatch 代理程式，應用程式洞察會保留您的組態。若要避免合併衝突，請從現有的代理程式組態檔案中移除您要在「應用程 CloudWatch 式深入解析」中使用的資源組態。如需詳細資訊，請參閱 手動建立或編輯 CloudWatch 代理程式組態檔。

IAM 政策

若要使用 CloudWatch 應用程式深入解析，您必須建立 AWS Identity and Access Management (IAM) 政策，並將其附加至您的使用者、群組或角色。如需有關使用者、群組及角色的詳細資訊，請參閱 IAM 身分 (使用者、群組及角色)。IAM 政策會定義使用者許可。

使用主控台建立 IAM 政策

若要使用 IAM 主控台建立 IAM 政策，請執行下列步驟。

1. 前往 IAM 主控台。在左導覽窗格中，選取 Policies (政策)。
2. 在頁面頂端，選取 Create policy (建立政策)。
3. 選取 JSON 標籤。
4. 複製下列 JSON 文件並貼在 JSON 標籤下。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "applicationinsights:*",
                "iam:CreateServiceLinkedRole",
                "iam:ListRoles",
                "resource-groups:ListGroups"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

5. 選取 Review Policy (檢閱政策)。
6. 輸入策略的名稱，例如「AppInsightsPolicy.」 或者輸入 Description (描述)。
7. 選取 Create Policy (建立政策)。

8. 在左側導覽窗格中，選取使用者群組、使用者或角色。
9. 選取您要連接政策之使用者群組、使用者或角色的名稱。
10. 選取 Add permissions (新增許可)。
11. 選取直接連接現有政策。
12. 搜尋剛剛建立的政策，然後選取該政策名稱左側的核取方塊。
13. 選取 Next: Review (下一步：檢閱)。
14. 確定列出正確的政策，然後選取 Add permissions (新增許可)。
15. 請確定您以使用應用程式深入解析時剛建立之原則相關聯的使 CloudWatch 用者登入。

若要建立 IAM 政策，請使用 AWS CLI

若要使用建立 IAM 政策 AWS CLI，請使用上述 JSON 文件做為目前資料夾中的檔案，從命令列執行建立政策作業。

若要使用建立 IAM 政策 AWS Tools for Windows PowerShell

若要使用建立 IAM 政策 AWS Tools for Windows PowerShell，請使用上述 JSON 文件做為目前資料夾中的檔案執行 New-IamPolicy 指令程式。

帳戶型應用程式上架的 IAM 角色許可

如果想要將帳戶中的所有資源上架，而且選擇不使用 Application Insights 受管政策完整存取 Application Insights 功能，則您必須將下列許可連接至 IAM 角色，以便 Application Insights 能夠探索您帳戶中的所有資源：

```
"ec2:DescribeInstances"
"ec2:DescribeNatGateways"
"ec2:DescribeVolumes"
"ec2:DescribeVPCs"
"rds:DescribeDBInstances"
"rds:DescribeDBClusters"
"sqs:ListQueues"
"elasticloadbalancing:DescribeLoadBalancers"
"autoscaling:DescribeAutoScalingGroups"
"lambda:ListFunctions"
"dynamodb:ListTables"
"s3:ListAllMyBuckets"
"sns:ListTopics"
```

```
"states:ListStateMachines"
"apigateway:GET"
"ecs:ListClusters"
"ecs:DescribeTaskDefinition"
"ecs:ListServices"
"ecs:ListTasks"
"eks:ListClusters"
"eks:ListNodegroups"
"fsx:DescribeFileSystems"
"route53:ListHealthChecks"
"route53:ListHostedZones"
"route53:ListQueryLoggingConfigs"
"route53resolver:ListFirewallRuleGroups"
"route53resolver:ListFirewallRuleGroupAssociations"
"route53resolver:ListResolverEndpoints"
"route53resolver:ListResolverQueryLogConfigs"
"route53resolver:ListResolverQueryLogConfigAssociations"
"logs:DescribeLogGroups"
"resource-explorer:ListResources"
```

安裝、設定及管理應用程式以進行監控

本節提供使用主控台和設定、設定和管理 CloudWatch 應用程式見解應用程式的 AWS CLI步驟 AWS Tools for Windows PowerShell。

主題

- 從主控台設定、設定和管理您的應用程式以進行監 CloudWatch 控
- 使用命令列安裝、設定及管理應用程式以進行監控
- 應用程式見解偵測到問題的 CloudWatch 事件和通知

從主控台設定、設定和管理您的應用程式以進行監 CloudWatch 控

本節提供設定、設定和管理要從 CloudWatch 主控台監視的應用程式的步驟。

主控台程序

- 新增並設定應用程式
- 啟用 Application Insights 進行 Amazon ECS 和 Amazon EKS 資源監控
- 停用應用程式元件的監控
- 刪除應用程式

新增並設定應用程式

從 CloudWatch 主控台新增和設定應用程式

若要從 CloudWatch 主控台開始使 CloudWatch 用應用程式深入解析，請執行下列步驟。

1. 啟動。開啟CloudWatch 主機登陸頁面。從左側導覽窗格中，選擇 Insights 下的 Application Insights。開啟的頁面會顯示使用「應用程式深入解析」監視的應用 CloudWatch 程式清單，以及其監控狀態。
2. 新增應用程式。若要設定應用程式的監控，請選擇 Add an application (新增應用程式)。當您選擇 Add an application (新增應用程式) 時，系統會提示您 Choose Application Type (選擇應用程式類型)。

   - 以資源群組為基礎的應用程式。選取此選項時，您可以選擇此帳戶中要監控的資源群組。若要在元件上使用多個應用程式，您必須使用資源群組型監控。
   - 以帳戶為基礎的應用程式。選取此選項時，您可以監控此帳戶中的所有資源。如果您想要監控帳戶中的所有資源，建議使用此選項，而不是以資源群組為基礎的選項，因為應用程式上架程序會更快。

   > **Note**
   >
   > 您無法使用 Application Insights，將以資源群組為基礎的監控與以帳戶為基礎的監控結合。若要變更應用程式類型，您必須刪除所有受監控的應用程式，並 Choose Application Type (選擇應用程式類型)。

   當您新增第一個應用程式進行監視時，Ap CloudWatch plication Insights 會在您的帳戶中建立服務連結角色，讓應用程式深入解析代表您呼叫其他 AWS 服務的權限。如需 Application Insights 在您帳戶中所建服務連結角色的詳細資訊，請參閱 將服務連結角色用於 CloudWatch 應用程式深入。
3. Resource-based application monitoring

   1. 選取資源群組。在 [指定應用程式詳細 AWS 資訊] 頁面上，從下拉式清單中選取包含應用程式資源的資源群組。這些資源包括前端伺服器、負載平衡器、自動調整規模群組和資料庫伺服器。

      如果尚未建立應用程式的資源群組，您可以透過選擇 Create new resource group (建立新的資源群組) 建立一個。如需建立資源群組的詳細資訊，請參閱《AWS Resource Groups 使用者指南》。

2. 監視 CloudWatch事件。選取此核取方塊，將應用程式洞察監控與 CloudWatch 事件整合，以取得來自 Amazon EBS、Amazon EC2 AWS CodeDeploy、Amazon ECS、 AWS Health API 和通知、亞馬遜 RDS、Amazon S3 和的見解。 AWS Step Functions
3. 與 AWS Systems Manager 集成 OpsCenter。若要檢視並在偵測到所選應用程式的問題時收到通知，請選取產生 Systems Manager 以 OpsCenter OpsItems 進行修復動作核取方塊。若要追蹤解決與 AWS 資源相關之作業工作項目 (OpsItems) 所採取的作業，請提供 SNS 主題 ARN。
4. 標籤-可選。 CloudWatch 應用程式深入解析同時支援以標籤 CloudFormation為基礎的資源群組和基礎資源群組 (Auto Scaling 群組除外) 如需詳細資訊，請參閱使用標籤編輯器。
5. 選擇下一步。

   會以下列格式為應用程式產生 ARN。

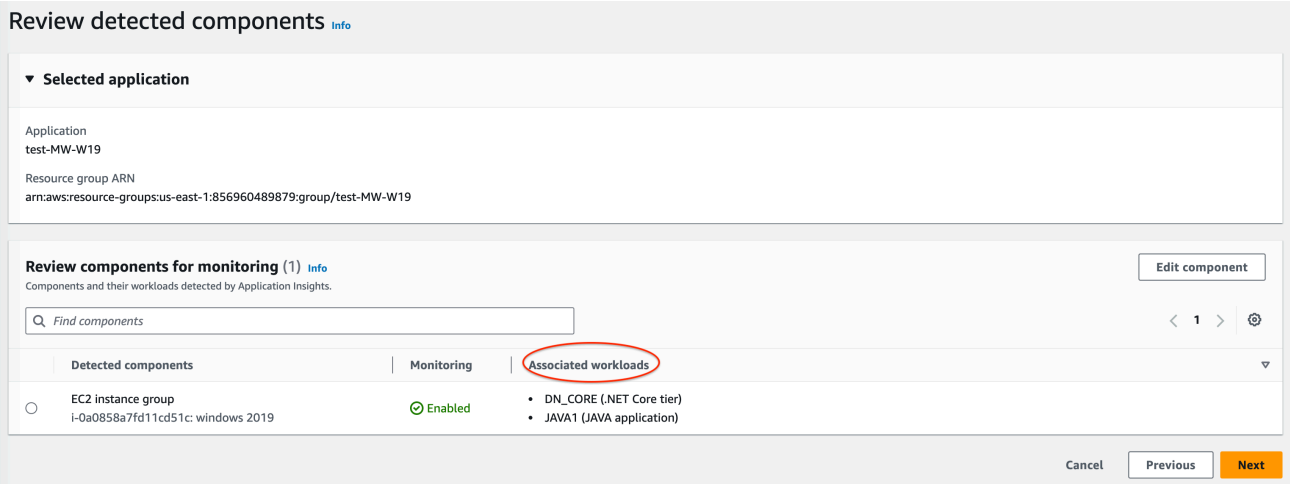
   ```
   arn:partition:applicationinsights:region:account-id:application/resource-group/resource-group-name
   ```

   範例

   ```
   arn:aws:applicationinsights:us-east-1:123456789012:application/resource-group/my-resource-group
   ```

6. 在檢閱偵測到的元件頁面的檢閱元件以供監控之用下，表格會列出偵測到的元件及其關聯的偵測到的工作負載。

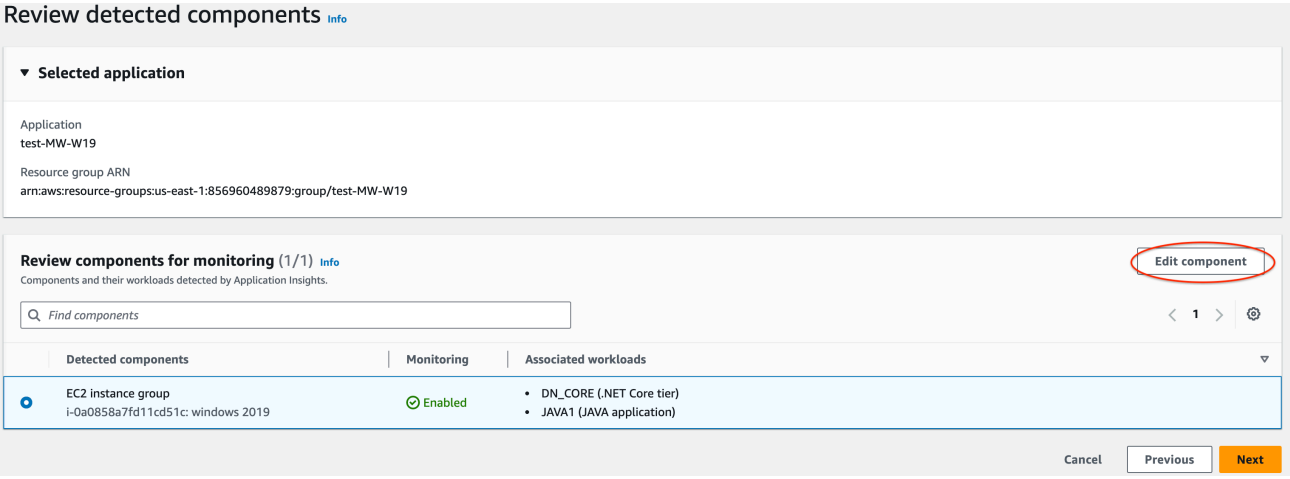
   > **Note**
   >
   > 對於支援多個自訂工作負載的元件，您最多可為每個元件監控五個工作負載。這些工作負載將與元件分開監控。



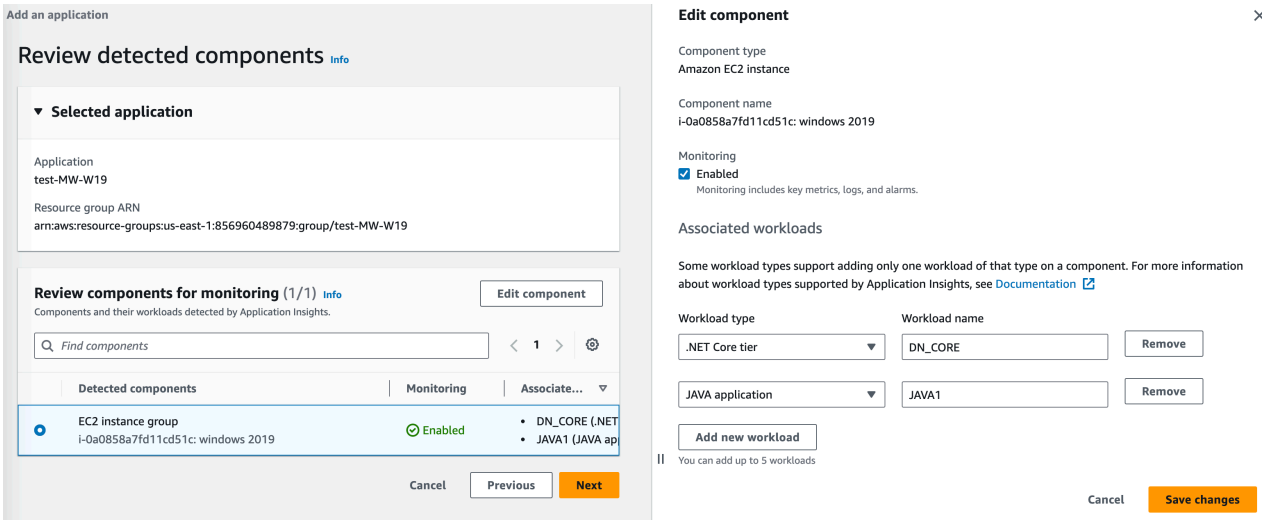
在關聯的工作負載下，如果未列出工作負載，畫面上會出現幾個可能的訊息。

- 無法偵測工作負載 – 嘗試偵測工作負載時發生問題。確定您已完成 必要條件。如果您需要新增工作負載，請選擇編輯元件。
- 未偵測到工作負載 – 我們未偵測到任何工作負載。您可能需要新增工作負載。若要這麼做，請選擇編輯元件。
- 不適用 – 元件不支援自訂工作負載，且會使用預設指標、警示和日誌進行監控。您無法將工作負載新增至這些元件。

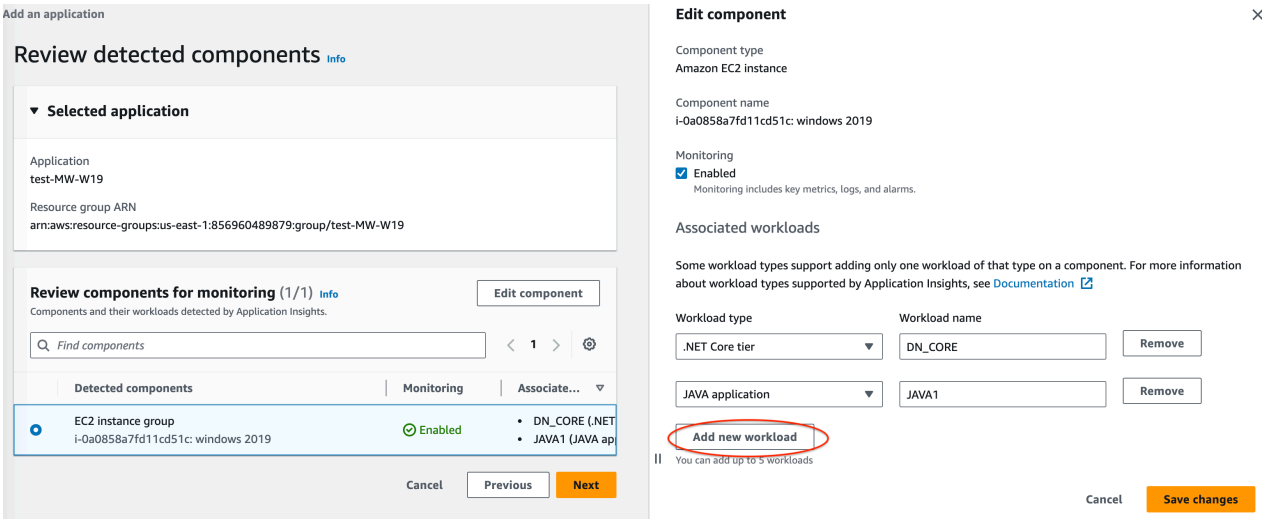
7. 若要編輯元件，請選取元件，然後選擇編輯元件。側邊面板隨即開啟，且在元件上偵測到工作負載。在此面板中，您可以編輯元件詳細資料並新增工作負載。



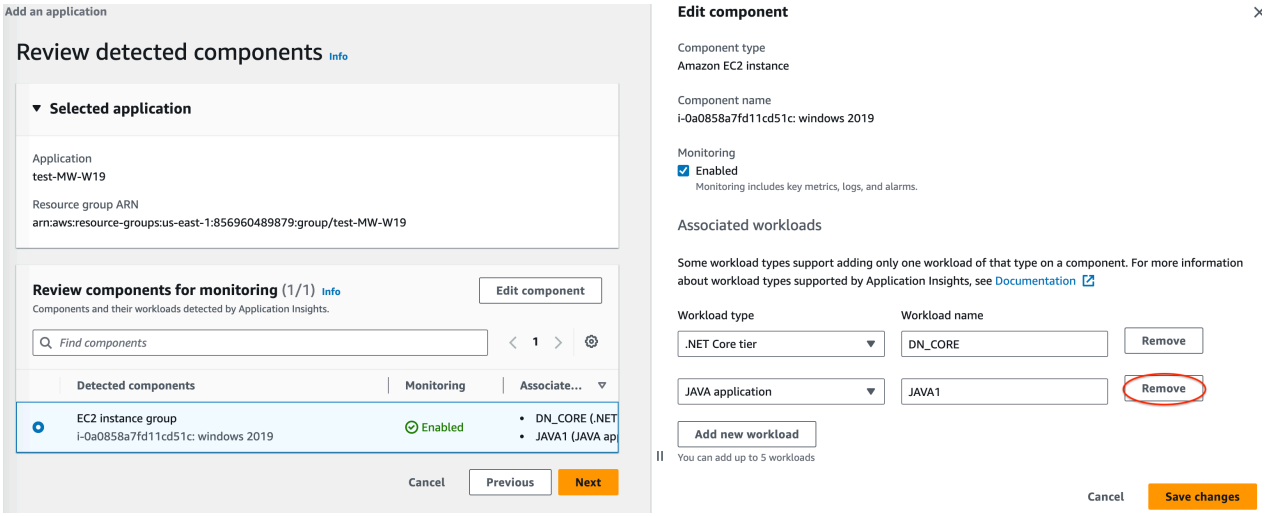
- 若要編輯工作負載類型或名稱，請使用下拉式清單。



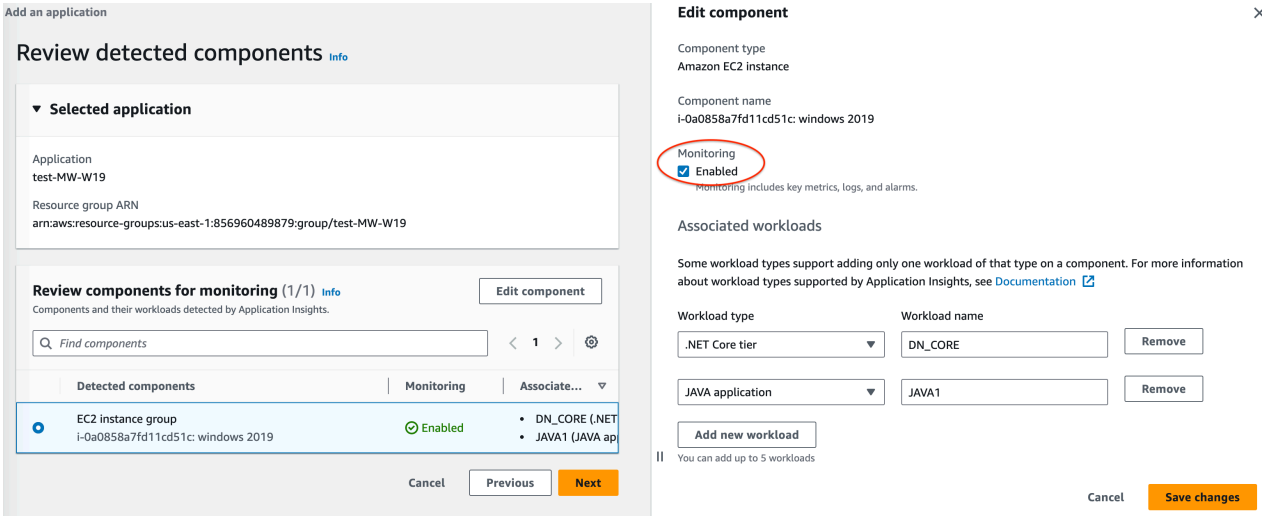
- 若要將工作負載新增至元件，請選擇新增工作負載。



  - 如果未顯示新增工作負載，則此元件不支援多個工作負載。
  - 如果未顯示關聯的工作負載標題，則此元件不支援自訂工作負載。
- 若要移除工作負載，請選擇您要從監控中移除之工作負載旁的移除。



- 若要停用整個元件的監控，請清除監控核取方塊。



- 編輯完元件後，請選擇右下角的儲存變更。對元件的工作負載所做的任何變更，都會顯示在關聯工作負載下的檢閱元件以供監控之用表格中。

8. 在檢閱偵測到的元件頁面上，選擇下一步。
9. 指定元件詳細資料頁面包含具有上一個步驟可自訂關聯工作負載的所有元件。

> **Note**
>
> 如果元件標頭有選擇性標籤，則該元件中工作負載的其他詳細資料為選擇性。

如果元件未出現在此頁面上，表示元件沒有可在此步驟中指定的任何其他詳細資料。

10選擇下一步。

11.在檢閱並提交頁面上，檢閱所有監督的元件和工作負載詳細資訊。

12選擇提交。

Account-based application monitoring

1. 應用程式名稱。輸入帳戶型應用程式的名稱。
2. 自動監控新資源。根據預設，在上架應用程式之後，Application Insights 會使用建議的設定為新增至帳戶的資源元件設定監控。您可以清除核取方塊，以排除上架應用程式之後新增的資源監控。
3. 監視 CloudWatch事件。選取此核取方塊，將應用程式洞察監控與 CloudWatch 事件整合，以取得來自 Amazon EBS、Amazon EC2 AWS CodeDeploy、Amazon ECS、 AWS Health API 和通知、亞馬遜 RDS、Amazon S3 和的見解。 AWS Step Functions
4. 與 AWS Systems Manager 集成 OpsCenter。若要檢視並在偵測到所選應用程式的問題時收到通知，請選取產生 Systems Manager 以 OpsCenter OpsItems 進行修復動作核取方塊。若要追蹤解決與 AWS 資源相關之作業工作項目 (OpsItems) 所採取的作業，請提供 SNS 主題 ARN。
5. 標籤-可選。 CloudWatch 應用程式深入解析同時支援以標籤 CloudFormation為基礎的資源群組和基礎資源群組 (Auto Scaling 群組除外) 如需詳細資訊，請參閱使用標籤編輯器。
6. 探索的資源。您帳戶中找到的所有資源都會新增至此清單。如果 Application Insights 無法找到您帳戶中的所有資源，頁面頂端會出現錯誤訊息。此訊息包含如何新增所需許可的文件連結。
7. 選擇下一步。

   會以下列格式為應用程式產生 ARN。

   ```
   arn:partition:applicationinsights:region:account-id:application/
   TBD/application-name
   ```

   範例

   ```
   arn:aws:applicationinsights:us-east-1:123456789012:application/TBD/my-
   application
   ```

4. 在您提交應用程式監控組態之後，您會來到應用程式的詳細資訊頁面，您可以在此檢視 Application summary (應用程式摘要)、Monitored components (受監控元件) 清單和 Unmonitored

components (未受監控元件)，以及透過選取 Components (元件)、Configuration history (組態歷史記錄)、Log patterns (日誌模式) 和任何已套用 Tags (標籤) 旁的索引標籤檢視。

若要檢視應用程式的洞察，請選擇 View Insights (檢視洞察)。

您可以選擇編輯來更新 CloudWatch 事件監視和與 AWS Systems Manager 整合 OpsCenter 的選擇。

在 Components (元件) 下，您可以選取 Actions (動作) 選單建立、修改或解除執行個體群組。

您可以管理元件的監控，包括應用程式層、日誌群組、事件日誌、指標和自訂警示，方法是選取元件旁的項目符號並選擇 Manage monitoring (管理監控)。

啟用 Application Insights 進行 Amazon ECS 和 Amazon EKS 資源監控

您可以從 Container Insights 主控台啟用 Application Insights 以監控容器化應用程式和微型服務。Application Insights 支援監控下列資源：

- Amazon ECS 叢集
- Amazon ECS 服務
- Amazon ECS 任務
- Amazon EKS 叢集

啟用 Application Insights 後，它會提供建議的指標和記錄、偵測潛在問題、產生 CloudWatch 事件，以及為您的容器化應用程式和微服務建立自動儀表板。

您可以從 Container Insights 或 Application Insights 主控台為容器化資源啟用 Application Insights。

從 Container Insights 主控台啟用 Application Insights

從 Container Insights 主控台的 Container Insights Performance monitoring (效能監控) 儀表板，選擇 Auto-configure Application Insights (自動設定 Application Insights)。啟用 Application Insights 時，會顯示偵測到問題的詳細資訊。

從 Application Insights 主控台啟用 Application Insights

當 ECS 叢集出現在元件清單中時，Application Insights 會自動啟用具有 Container Insights 的其他容器監控。

對於 EKS 叢集，您可以啟用具有 Container Insights 的其他監控，以提供診斷資訊，例如容器重新啟動故障，協助您隔離和解決問題。設定 EKS 的 Container Insights 需要其他步驟。如需相關資訊，請參閱設定 EKS 上 Container Insights 的 在 Amazon EKS 和 Kubernetes 上設定 Container Insights 步驟。

使用 EKS 的 Linux 執行個體支援對具有 Container Insights 的 EKS 進行額外監控。

如需 ECS 和 EKS 叢集 Container Insights 支援的詳細資訊，請參閱 Container Insights。

停用應用程式元件的監控

若要停用應用程式元件的監控，請從應用程式詳細資訊頁面選取您要停用監控的元件。選擇 Actions (動作)，然後 Remove from monitoring (從監控移除)。

刪除應用程式

若要刪除應用程式，請從 CloudWatch 儀表板的左側導覽窗格中，選擇 [深入解析] 下的 [應用程式見解]。選取您要刪除的應用程式。在 Actions (動作) 下，選擇 Delete application (刪除應用程式)。這會刪除監控並刪除所有已儲存的應用程式元件監控。不會刪除應用程式資源。

使用命令列安裝、設定及管理應用程式以進行監控

本節提供設定、設定和管理應用程式以使用 AWS CLI 和監視的步驟 AWS Tools for Windows PowerShell。

命令列程序

- 新增及管理應用程式
- 管理和更新監控
- 設定 SQL Always On 可用性群組的監控
- 設定監控 MySQL RDS
- 設定監控 MySQL EC2
- 設定 PostgreSQL RDS 的監控
- 設定 PostgreSQL EC2 的監控
- 設定 Oracle RDS 的監控
- 設定 Oracle EC2 的監控

新增及管理應用程式

您可以使用命令列新增、取得相關資訊、管理和設定您的 Application Insights 應用程式。

主題

- 新增應用程式
- 描述應用程式
- 列出應用程式中的元件
- 描述元件
- 將類似資源分組到自訂元件
- 取消群組自訂元件
- 更新應用程式
- 更新自訂元件

新增應用程式

使用新增應用程式 AWS CLI

若要使用為您的 AWS CLI 資源群組新增應用程式`my-resource-group`，並 OpsCenter 啟用可將建立的 opsItem 傳遞至 SNS 主題 ARN `arn:aws:sns:us-east-1:123456789012:MyTopic`，請使用下列命令。

```
aws application-insights create-application --resource-group-name my-resource-group --
ops-center-enabled --ops-item-sns-topic-arn arn:aws:sns:us-east-1:123456789012:MyTopic
```

使用新增應用程式 AWS Tools for Windows PowerShell

若要用 AWS Tools for Windows PowerShell 來為 OpsCenter 已啟用的資源群組新增應`my-resource-group`用程式，以將建立的 opsItem 傳遞至 SNS 主題 ARN `arn:aws:sns:us-east-1:123456789012:MyTopic`，請使用下列命令。

```
New-CWAIApplication -ResourceGroupName my-resource-group -OpsCenterEnabled true -
OpsItemSNSTopicArn arn:aws:sns:us-east-1:123456789012:MyTopic
```

描述應用程式

使用描述應用程式 AWS CLI

若要使用 AWS CLI 描述在名為的資源群組上建立的應用程式`my-resource-group`，請使用下列命令。

```
aws application-insights describe-application --resource-group-name my-resource-group
```

描述一個應用程序 AWS Tools for Windows PowerShell

若要使用 AWS Tools for Windows PowerShell 描述在名為的資源群組上建立的應用程式my-resource-group，請使用下列命令。

```
Get-CWAIApplication -ResourceGroupName my-resource-group
```

列出應用程式中的元件

列出應用程式中使用的元件 AWS CLI

若要使用列 AWS CLI 出在名為的資源群組上建立的元件my-resource-group，請使用下列命令。

```
aws application-insights list-components --resource-group-name my-resource-group
```

使用列出應用程式中的元件 AWS Tools for Windows PowerShell

若要使用列 AWS Tools for Windows PowerShell 出在名為的資源群組上建立的元件my-resource-group，請使用下列命令。

```
Get-CWAIComponentList -ResourceGroupName my-resource-group
```

描述元件

使用描述元件 AWS CLI

您可以使用下列 AWS CLI 命令來描述名為的元件，my-component該元件屬於在名為的資源群組上建立的應用程式my-resource-group。

```
aws application-insights describe-component --resource-group-name my-resource-group --
component-name my-component
```

使用描述組件 AWS Tools for Windows PowerShell

您可以使用下列 AWS Tools for Windows PowerShell 命令來描述名為的元件，my-component該元件屬於在名為的資源群組上建立的應用程式my-resource-group。

```
Get-CWAIComponent -ComponentName my-component -ResourceGroupName my-resource-group
```

將類似資源分組到自訂元件

建議您將類似的資源 (例如 .NET Web 伺服器執行個體) 分組到自訂元件，以更輕鬆採用及利於監控和洞見。目前， CloudWatch 應用程式洞見支援 EC2 執行個體的自訂群組。

若要使用 AWS CLI將資源分組到自訂元件

若要使用將三個執行個體 (arn:aws:ec2:us-east-1:123456789012:instance/i-11111arn:aws:ec2:us-east-1:123456789012:instance/i-22222、和arn:aws:ec2:us-east-1:123456789012:instance/i-33333) 分組到一個自訂元件中， AWS CLI 要my-component求為呼叫的資源群組建立的應用程式my-resource-group，請使用下列命令。

```
aws application-insights create-component --resource-group-name my-
resource-group --component-name my-component --resource-list arn:aws:ec2:us-
east-1:123456789012:instance/i-11111 arn:aws:ec2:us-east-1:123456789012:instance/
i-22222 arn:aws:ec2:us-east-1:123456789012:instance/i-33333
```

若要使用 AWS Tools for Windows PowerShell將資源分組到自訂元件

若 AWS Tools for Windows PowerShell 要使用將三個執行個體 (arn:aws:ec2:us-east-1:123456789012:instance/i-11111arn:aws:ec2:us-east-1:123456789012:instance/i-22222、和arn:aws:ec2:us-east-1:123456789012:instance/i-33333) 群組成名為的自訂元件my-component，針對為名為資源群組建立的應用程式my-resource-group，請使用下列命令。

```
New-CWAIComponent -ResourceGroupName my-resource-group -ComponentName my-component
 -ResourceList arn:aws:ec2:us-east-1:123456789012:instance/i-11111,arn:aws:ec2:us-
east-1:123456789012:instance/i-22222,arn:aws:ec2:us-east-1:123456789012:instance/
i-33333
```

取消群組自訂元件

使用取消群組自訂元件 AWS CLI

若要使用取 AWS CLI 消在資源群組上建立之應用程式my-component中名為的自訂元件的群組my-resource-group，請使用下列命令。

```
aws application-insights delete-component --resource-group-name my-resource-group --
component-name my-new-component
```

使用取消群組自訂元件 AWS Tools for Windows PowerShell

若要使用取 AWS Tools for Windows PowerShell 消在資源群組上建立之應用程式my-component中名為的自訂元件的群組my-resource-group，請使用下列命令。

```
Remove-CWAIComponent -ComponentName my-component -ResourceGroupName my-resource-group
```

更新應用程式

使用更新應用程式 AWS CLI

您可以使用 AWS CLI 來更新應用程式，以 OpsCenter OpsItems 針對與應用程式偵測到的問題產生 AWS Systems Manager，並使用下列命令 OpsItems 將建立的應用程式與 SNS 主題arn:aws:sns:us-east-1:123456789012:MyTopic相關聯。

```
aws application-insights update-application --resource-group-name my-resource-group --
ops-center-enabled --ops-item-sns-topic-arn arn:aws:sns:us-east-1:123456789012:MyTopic
```

使用 Windows 的 AWS 工具更新應用程式 PowerShell

您可以使用更新應 AWS Tools for Windows PowerShell 用程式，以 OpsCenter OpsItems 針對與應用程式偵測到的問題產生 AWS SSM，並使用下列命令 OpsItems 將建立的與 SNS 主題arn:aws:sns:us-east-1:123456789012:MyTopic相關聯。

```
Update-CWAIApplication -ResourceGroupName my-resource-group -OpsCenterEnabled true -
OpsItemSNSTopicArn arn:aws:sns:us-east-1:123456789012:MyTopic
```

更新自訂元件

使用更新自訂元件 AWS CLI

您可以使用下列指令 AWS CLI 來更新名為新元件名稱的自訂元件，以及更新的例證群組。my-component my-new-component

```
aws application-insights update-component --resource-group-name my-resource-
group --component-name my-component --new-component-name my-new-component --
resource-list arn:aws:ec2:us-east-1:123456789012:instance/i-44444 arn:aws:ec2:us-
east-1:123456789012:instance/i-55555
```

使用 Windows 的 AWS 工具更新自訂元件 PowerShell

您可以使用下列指令 AWS Tools for Windows PowerShell 來更新名為新元件名稱的自訂元件，以及更新的例證群組。`my-component` `my-new-component`

```
Update-CWAIComponent -ComponentName my-component -NewComponentName my-new-
component -ResourceGroupName my-resource-group -ResourceList arn:aws:ec2:us-
east-1:123456789012:instance/i-44444,arn:aws:ec2:us-east-1:123456789012:instance/
i-55555
```

管理和更新監控

您可以使用命令列來管理和更新您 Application Insights 應用程式的監控。

主題

- 列出您應用程式的問題
- 描述應用程式問題
- 描述與問題相關聯的異常或錯誤
- 描述應用程式的異常或錯誤
- 描述元件的監控組態
- 描述元件的建議監控組態
- 更新元件的監控組態
- 從 Application Insights 監控移除指定的資源群組

列出您應用程式的問題

使用列出應用程式的問題 AWS CLI

若要使用列 AWS CLI 出在名為的資源群組上建立的應用程式自 Unix Epoch 起 1,000 到 10,000 毫秒之間偵測到的應用程式的問題`my-resource-group`，請使用下列命令。

```
aws application-insights list-problems --resource-group-name my-resource-group --start-
time 1000 --end-time 10000
```

使用 Windows AWS 工具列出應用程式的問題 PowerShell

若要使用列 AWS Tools for Windows PowerShell 出在名為的資源群組上建立的應用程式自 Unix Epoch 起 1,000 到 10,000 毫秒之間偵測到的應用程式的問題`my-resource-group`，請使用下列命令。

```
$startDate = "8/6/2019 3:33:00"
$endDate = "8/6/2019 3:34:00"
Get-CWAIProblemList -ResourceGroupName my-resource-group -StartTime $startDate -
EndTime $endDate
```

描述應用程式問題

使用描述應用程式問題 AWS CLI

若要使用 AWS CLI 來描述問題 ID 的問題p-1234567890，請使用下列命令。

```
aws application-insights describe-problem –problem-id p-1234567890
```

使用 Windows 的 AWS 工具描述應用程式問題 PowerShell

若要使用 AWS Tools for Windows PowerShell 來描述問題 ID 的問題p-1234567890，請使用下列命令。

```
Get-CWAIProblem -ProblemId p-1234567890
```

描述與問題相關聯的異常或錯誤

描述與問題相關的異常或錯誤 AWS CLI

若要使用 AWS CLI 描述與問題 ID 的問題相關聯的異常或錯誤p-1234567890，請使用下列命令。

```
aws application-insights describe-problem-observations --problem-id -1234567890
```

使用 AWS Tools for Windows PowerShell描述與問題相關聯的異常或錯誤

若要使用 AWS Tools for Windows PowerShell 描述與問題 ID 的問題相關聯的異常或錯誤p-1234567890，請使用下列命令。

```
Get-CWAIProblemObservation -ProblemId p-1234567890
```

描述應用程式的異常或錯誤

使用 AWS CLI 描述應用程式的異常或錯誤

若要使用觀察 ID AWS CLI 來描述應用程式的異常或錯誤o-1234567890，請使用下列命令。

```
aws application-insights describe-observation –observation-id o-1234567890
```

使用 Windows AWS 工具描述應用程式的異常或錯誤 PowerShell

若要使用觀察 ID AWS Tools for Windows PowerShell 來描述應用程式的異常或錯誤o-1234567890，請使用下列命令。

```
Get-CWAIObservation -ObservationId o-1234567890
```

描述元件的監控組態

使用 AWS CLI描述元件的監控組態

若要使用 AWS CLI 描述在資源群組上建立之應用程式my-component中呼叫之元件的監視組態my-resource-group，請使用下列命令。

```
aws application-insights describe-component-configuration –resource-group-name my-resource-group –component-name my-component
```

使用 Windows 的 AWS 工具描述元件的監視組態 PowerShell

若要在資源群組上建立的應用程式中使用 AWS Tools for Windows PowerShell 來描述名為my-component之元件的監視組態my-resource-group，請使用下列命令。

```
Get-CWAIComponentConfiguration -ComponentName my-component -ResourceGroupName my-resource-group
```

如需元件組態的詳細資訊和範例 JSON 檔案，請參閱 使用元件組態。

描述元件的建議監控組態

描述元件的建議監視組態，使用 AWS CLI

當元件是 .NET Worker 應用程式的一部分時，您可以使用下列命令 AWS CLI 來描述my-component在資源群組上建立之應用程式中呼叫之元件的建議監視組my-resource-group態。

```
aws application-insights describe-component-configuration-recommendation --resource-group-name my-resource-group --component-name my-component --tier DOT_NET_WORKER
```

使用說明元件的建議監視組態 AWS Tools for Windows PowerShell

當元件是 .NET Worker 應用程式的一部分時，您可以使用下列命令 AWS Tools for Windows PowerShell 來描述my-component在資源群組上建立之應用程式中呼叫之元件的建議監視組my-resource-group態。

```
Get-CWAIComponentConfigurationRecommendation -ComponentName my-component -
ResourceGroupName my-resource-group -Tier DOT_NET_WORKER
```

如需元件組態的詳細資訊和範例 JSON 檔案，請參閱 使用元件組態。

更新元件的監控組態

使用 AWS CLI更新元件的監控組態

若要使用 AWS CLI 更新my-component在名為的資源群組上建立的應用程式中呼叫的元件my-resource-group，請使用下列命令。命令包含這些動作：

1. 啟用元件的監控。
2. 將元件的層設為 .NET Worker。
3. 將元件的 JSON 組態更新為從本機檔案 configuration.txt 讀取。

```
aws application-insights update-component-configuration --resource-group-name my-
resource-group --component-name my-component --tier DOT_NET_WORKER --monitor --
component-configuration "file://configuration.txt"
```

使用 AWS Tools for Windows PowerShell更新元件的監控組態

若要使用 AWS Tools for Windows PowerShell 更新my-component在名為的資源群組上建立的應用程式中呼叫的元件my-resource-group，請使用下列命令。命令包含這些動作：

1. 啟用元件的監控。
2. 將元件的層設為 .NET Worker。
3. 將元件的 JSON 組態更新為從本機檔案 configuration.txt 讀取。

```
[string]$config = Get-Content -Path configuration.txt
Update-CWAIComponentConfiguration -ComponentName my-component -ResourceGroupName my-
resource-group -Tier DOT_NET_WORKER -Monitor 1 -ComponentConfiguration $config
```

如需元件組態的詳細資訊和範例 JSON 檔案，請參閱 使用元件組態。

從 Application Insights 監控移除指定的資源群組

使用從「應用程式見解」監視中移除指定的資源群組 AWS CLI

若要使用my-resource-group從監視中移除在呼叫的資源群組上建立的應用程式，請使用下列命令。 AWS CLI

```
aws application-insights delete-application --resource-group-name my-resource-group
```

使用從「應用程式見解」監視中移除指定的資源群組 AWS Tools for Windows PowerShell

若要使用my-resource-group從監視中移除在呼叫的資源群組上建立的應用程式，請使用下列命令。 AWS Tools for Windows PowerShell

```
Remove-CWAIApplication -ResourceGroupName my-resource-group
```

設定 SQL Always On 可用性群組的監控

1. 為資源群組建立具有 SQL HA EC2 執行個體的應用程式。

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. 透過建立新的應用程式元件，來定義代表 SQL HA 叢集的 EC2 執行個體。

```
aws application-insights create-component --resource-group-name
 "<RESOURCE_GROUP_NAME>" --component-name SQL_HA_CLUSTER --resource-list
  "arn:aws:ec2:<REGION>:<ACCOUNT_ID>:instance/<CLUSTER_INSTANCE_1_ID>"
 "arn:aws:ec2:<REGION>:<ACCOUNT_ID>:instance/<CLUSTER_INSTANCE_2_ID>
```

3. 設定 SQL HA 元件。

```
aws application-insights  update-component-configuration --resource-group-name
 "<RESOURCE_GROUP_NAME>" --region <REGION> --component-name "SQL_HA_CLUSTER" --
monitor --tier SQL_SERVER_ALWAYSON_AVAILABILITY_GROUP --monitor  --component-
configuration '{
  "subComponents" : [ {
    "subComponentType" : "AWS::EC2::Instance",
    "alarmMetrics" : [ {
      "alarmMetricName" : "CPUUtilization",
      "monitor" : true
    }, {
```

```
      "alarmMetricName" : "StatusCheckFailed",
      "monitor" : true
    }, {
      "alarmMetricName" : "Processor % Processor Time",
      "monitor" : true
    }, {
      "alarmMetricName" : "Memory % Committed Bytes In Use",
      "monitor" : true
    }, {
      "alarmMetricName" : "Memory Available Mbytes",
      "monitor" : true
    }, {
      "alarmMetricName" : "Paging File % Usage",
      "monitor" : true
    }, {
      "alarmMetricName" : "System Processor Queue Length",
      "monitor" : true
    }, {
      "alarmMetricName" : "Network Interface Bytes Total/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "PhysicalDisk % Disk Time",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Buffer Manager Buffer cache hit ratio",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Buffer Manager Page life expectancy",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:General Statistics Processes blocked",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:General Statistics User Connections",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Locks Number of Deadlocks/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:SQL Statistics Batch Requests/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica File Bytes Received/sec",
      "monitor" : true
```

```
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Log Bytes Received/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Log remaining for undo",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Log Send Queue",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Mirrored Write Transaction/
sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Recovery Queue",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Redo Bytes Remaining",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Redone Bytes/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Total Log requiring undo",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Transaction Delay",
      "monitor" : true
    } ],
    "windowsEvents" : [ {
      "logGroupName" : "WINDOWS_EVENTS-Application-<RESOURCE_GROUP_NAME>",
      "eventName" : "Application",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL", "INFORMATION" ],
      "monitor" : true
    }, {
      "logGroupName" : "WINDOWS_EVENTS-System-<RESOURCE_GROUP_NAME>",
      "eventName" : "System",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL" ],
      "monitor" : true
    }, {
      "logGroupName" : "WINDOWS_EVENTS-Security-<RESOURCE_GROUP_NAME>",
      "eventName" : "Security",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL" ],
      "monitor" : true
```

```
    } ],
    "logs" : [ {
      "logGroupName" : "SQL_SERVER_ALWAYSON_AVAILABILITY_GROUP-
<RESOURCE_GROUP_NAME>",
      "logPath" : "C:\\Program Files\\Microsoft SQL Server\\MSSQL**.MSSQLSERVER\
\MSSQL\\Log\\ERRORLOG",
      "logType" : "SQL_SERVER",
      "monitor" : true,
      "encoding" : "utf-8"
    } ]
  }, {
    "subComponentType" : "AWS::EC2::Volume",
    "alarmMetrics" : [ {
      "alarmMetricName" : "VolumeReadBytes",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeWriteBytes",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeReadOps",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeWriteOps",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeQueueLength",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeThroughputPercentage",
      "monitor" : true
    }, {
    "alarmMetricName" : "BurstBalance",
      "monitor" : true
    } ]
  } ]
}'
```

> **Note**
>
> Application Insights 必須導入應用程式事件日誌 (資訊層級)，才能偵測叢集活動，例如容錯移轉。

設定監控 MySQL RDS

1. 為資源群組建立具有 RDS MySQL 資料庫執行個體的應用程式。

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. 錯誤日誌預設為啟用。使用資料參數群組可啟用慢查詢日誌。如需詳細資訊，請參閱存取 MySQL 慢查詢日誌及一般日誌。
   - `set slow_query_log = 1`
   - `set log_output = FILE`
3. 將要監視的記錄檔匯出至 CloudWatch 記錄檔。如需詳細資訊，請參閱將 MySQL 記錄發佈至 CloudWatch 記錄檔。
4. 設定 MySQL RDS 元件。

```
aws application-insights  update-component-configuration --resource-group-name
 "<RESOURCE_GROUP_NAME>" --region <REGION> --component-name "<DB_COMPONENT_NAME>"
 --monitor --tier DEFAULT --monitor  --component-configuration "{\"alarmMetrics\":
[{\"alarmMetricName\":\"CPUUtilization\",\"monitor\":true}],\"logs\":[{\"logType\":
\"MYSQL\",\"monitor\":true},{\"logType\": \"MYSQL_SLOW_QUERY\",\"monitor\":false}]}"
```

設定監控 MySQL EC2

1. 為資源群組建立具有 SQL HA EC2 執行個體的應用程式。

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. 錯誤日誌預設為啟用。使用資料參數群組可啟用慢查詢日誌。如需詳細資訊，請參閱存取 MySQL 慢查詢日誌及一般日誌。
   - `set slow_query_log = 1`
   - `set log_output = FILE`
3. 設定 MySQL EC2 元件。

```
aws application-insights  update-component-configuration --resource-group-name
 "<RESOURCE_GROUP_NAME>" --region <REGION> --component-name "<DB_COMPONENT_NAME>"
 --monitor --tier MYSQL --monitor  --component-configuration "{\"alarmMetrics\":
[{\"alarmMetricName\":\"CPUUtilization\",\"monitor\":true}],\"logs\":[{\"logGroupName
```

```
\":\"<UNIQUE_LOG_GROUP_NAME>\",\"logPath\":\"C:\\\\ProgramData\\\\MySQL\\\\MySQL
 Server **\\\\Data\\\\<FILE_NAME>.err\",\"logType\":\"MYSQL\",\"monitor\":true,
\"encoding\":\"utf-8\"}]}"
```

設定 PostgreSQL RDS 的監控

1. 為資源群組建立具有 PostgreSQL RDS 資料庫執行個體的應用程式。

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. 依預設，不會啟用將 CloudWatch PostgreSQL 記錄檔發佈到。若要啟用監控，請開啟 RDS 主控台，然後選取要監控的資料庫。選擇右上角的 Modify (修改)，然後選取標示為 PostgreSQL 日誌的核取方塊。選擇 Continue (繼續) 以儲存此設定。
3. 您的 PostgreSQL 記錄檔會匯出到. CloudWatch
4. 設定 PostgreSQL RDS 元件。

```
aws application-insights update-component-configuration --region <REGION> --resource-
group-name <RESOURCE_GROUP_NAME> --component-name <DB_COMPONENT_NAME> --monitor --
tier DEFAULT --component-configuration
"{
   \"alarmMetrics\":[
      {
         \"alarmMetricName\": \"CPUUtilization\",
         \"monitor\": true
      }
   ],
   \"logs\":[
      {
         \"logType\": \"POSTGRESQL\",
         \"monitor\": true
      }
   ]
}"
```

設定 PostgreSQL EC2 的監控

1. 為資源群組建立具有 PostgreSQL EC2 執行個體的應用程式。

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. 設定 PostgreSQL EC2 元件。

```
aws application-insights update-component-configuration --region <REGION> --resource-
group-name <RESOURCE_GROUP_NAME> --component-name <DB_COMPONENT_NAME> --monitor --
tier POSTGRESQL --component-configuration
"{
   \"alarmMetrics\":[
      {
         \"alarmMetricName\":\"CPUUtilization\",
         \"monitor\":true
      }
   ],
   \"logs\":[
      {
         \"logGroupName\":\"<UNIQUE_LOG_GROUP_NAME>\",
         \"logPath\":\"/var/lib/pgsql/data/log/\",
         \"logType\":\"POSTGRESQL\",
         \"monitor\":true,
         \"encoding\":\"utf-8\"
      }
   ]
}"
```

設定 Oracle RDS 的監控

1. 為資源群組建立具有 Oracle RDS 資料庫執行個體的應用程式。

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. 依預設，不會啟用將 CloudWatch Oracle 記錄檔發佈至。若要啟用監控，請開啟 RDS 主控台，然後選取要監控的資料庫。選擇右上角的 Modify (修改)，然後選取標示為 Alert (提醒) 日誌和 Listener (接聽程式) 日誌的核取方塊。選擇 Continue (繼續) 以儲存此設定。
3. 您的 Oracle 記錄檔會匯出至 CloudWatch。
4. 設定 Oracle RDS 元件。

```
aws application-insights update-component-configuration --region <REGION> --resource-
group-name <RESOURCE_GROUP_NAME> --component-name <DB_COMPONENT_NAME> --monitor --
tier DEFAULT --component-configuration
"{
   \"alarmMetrics\":[
      {
         \"alarmMetricName\": \"CPUUtilization\",
         \"monitor\": true
      }
   ],
   \"logs\":[
      {
         \"logType\": \"ORACLE_ALERT\",
         \"monitor\": true
      },
      {
         \"logType\": \"ORACLE_LISTENER\",
         \"monitor\": true
      }
   ]
}"
```

設定 Oracle EC2 的監控

1. 為資源群組建立具有 Oracle EC2 執行個體的應用程式。

```
aws application-insights create-application --region <REGION> --resource-group-name
  <RESOURCE_GROUP_NAME>
```

2. 設定 Oracle EC2 元件。

```
aws application-insights update-component-configuration --region <REGION> --resource-
group-name <RESOURCE_GROUP_NAME> --component-name <DB_COMPONENT_NAME> --monitor --
tier ORACLE --component-configuration
"{
   \"alarmMetrics\":[
      {
         \"alarmMetricName\":\"CPUUtilization\",
         \"monitor\":true
      }
   ],
```

```
   \"logs\":[
      {
         \"logGroupName\":\"<UNIQUE_LOG_GROUP_NAME>\",
         \"logPath\":\"/opt/oracle/diag/rdbms/*/*/trace\",
         \"logType\":\"ORACLE_ALERT\",
         \"monitor\":true,
      },
      {
         \"logGroupName\":\"<UNIQUE_LOG_GROUP_NAME>\",
         \"logPath\":\"/opt/oracle/diag/tnslsnr/$HOSTNAME/listener/trace/\",
         \"logType\":\"ORACLE_ALERT\",
         \"monitor\":true,
      }
   ]
}"
```

應用程式見解偵測到問題的 CloudWatch 事件和通知

針對新增至「應用程 CloudWatch 式見解」的每個應用程式，都會針對下列 CloudWatch 事件發佈事件，以最佳方式執行：

- 問題建立。當 CloudWatch 應用程式深入解析偵測到新問題時發出。
  - 詳細資訊類型：「偵測到 Application Insights 問題」
  - 詳細資訊：
    - `problemId`：偵測到的問題 ID。
    - `region`：建立問題的 AWS 區域。
    - `resourceGroupName`：偵測到問題的已註冊應用程式資源群組。
    - `status`：問題狀態。可能的狀態和定義如下：
      - `In progress`：已發現新問題。問題仍在接受觀察。
      - `Recovering`：問題在穩定下來。當問題處於此狀態時，您可以手動解決問題。
      - `Resolved`：問題已解決。沒有關於這個問題的新觀察。
      - `Recurring`：問題已在過去 24 小時內解決。由於有其他觀察結果，問題已重新開啟。
    - `severity`：問題嚴重性。
    - `problemUrl`：問題的主控台 URL。

- 問題更新。使用新觀察更新問題後，或更新現有觀察並隨後更新問題時發出；更新包括解決方案或關閉問題。
  - 詳細資訊類型：「已更新 Application Insights 問題」
  - 詳細資訊：
    - `problemId`：建立的問題 ID。
    - `region`：建立問題的 AWS 區域。
    - `resourceGroupName`：偵測到問題的已註冊應用程式資源群組。
    - `status`：問題狀態。
    - `severity`：問題嚴重性。
    - `problemUrl`：問題的主控台 URL。

如何收到應用程式產生的問題事件通知

在 CloudWatch 主控台中，選取左側導覽窗格中「事件」下的「規則」。從 Rules (規則) 頁面中選取 Create rule (建立規則)。從「服務名稱」下拉 CloudWatch 式清單中選擇 Amazon 應用程式洞察，然後選擇事件類型。接著，選擇 Add target (新增目標)，並選取目標和參數，例如 SNS topic (SNS 主題) 或 Lambda function (Lambda 函數)。

通過操作 AWS Systems Manager。 CloudWatch 應用程序洞察提供與 Systems Manager 內置集成 OpsCenter。如果您選擇將此整合用於您的應用程式，則會針對應 OpsItem 用程式偵測到的每個問題，在 OpsCenter 主控台上建立一個。從 OpsCenter 主控台，您可以檢視 CloudWatch 應用程式深入解析所偵測到問題的摘要資訊，並挑選 Systems Manager 自動化執行手冊以採取補救動作，或進一步識別造成應用程式資源問題的 Windows 處理序。

Application Insights 跨帳戶觀察

透過「應用 CloudWatch 程式深入解析」跨帳戶可觀察性，您可以監控單一區域內跨多個 AWS 帳戶的應用程式並進行疑難排解。

您可以使用 Amazon CloudWatch 觀察性存取管理員將一或多個 AWS 帳戶設定為監控帳戶。您可以在監控帳戶中建立接收器，讓監控帳戶能夠檢視來源帳戶中的資料。可以使用此接收器建立從來源帳戶到監控帳戶的連結。如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。

必要的資源

如需 CloudWatch 應用程式深入解析跨帳戶可觀察性的正確功能，請確定下列遙測類型是透過 CloudWatch 觀察性存取管理員共用。

- 應用見解中的 CloudWatch 應用
- Amazon 的指標 CloudWatch
- Amazon CloudWatch 日誌中的日誌群組
- AWS X-Ray 中的追蹤

使用元件組態

元件組態是 JSON 格式的文字檔案，描述元件的組態設定。本節提供了範例範本片段、元件組態區段的描述和範例元件組態。

主題

- 元件組態範本片段
- 元件組態區段
- 元件組態範例

元件組態範本片段

下列範例顯示 JSON 格式的範本片段。

```
{
  "alarmMetrics" : [
    list of alarm metrics
  ],
  "logs" : [
    list of logs
  ],
  "processes" : [
   list of processes
  ],
  "windowsEvents" : [
    list of windows events channels configurations
  ],
  "alarms" : [
    list of CloudWatch alarms
  ],
  "jmxPrometheusExporter": {
    JMX Prometheus Exporter configuration
  },
  "hanaPrometheusExporter": {
```

```
      SAP HANA Prometheus Exporter configuration
  },
  "haClusterPrometheusExporter": {
      HA Cluster Prometheus Exporter configuration
  },
  "netWeaverPrometheusExporter": {
      SAP NetWeaver Prometheus Exporter configuration
  },
  "subComponents" : [
    {
      "subComponentType" : "AWS::EC2::Instance" ...
      component nested instances configuration
    },
    {
      "subComponentType" : "AWS::EC2::Volume" ...
      component nested volumes configuration
    }
  ]
}
```

元件組態區段

元件組態包含數個主要區段。元件組態中的區段可以任何順序列出。

- alarmMetrics (選用)

  要針對元件監控的指標清單 。所有元件類型都可以有 alarmMetrics 區段。
- 日誌 (選用)

  要針對元件監控的日誌清單 。只有 EC2 執行個體可以有日誌區段。
- processes (選用)

  要針對元件監控的程式清單。只有 EC2 執行個體可以有程式區段。
- subComponents (選用)

  元件的巢狀執行個體和磁碟區子元件組態。下列類型的元件可以有巢狀執行個體和子元件區段：ELB、ASG、自訂分組的 EC2 執行個體和 EC2 執行個體。
- alarms (optional) (警示 (選用))

  監控元件的警示清單。所有元件類型都可以有 alarm (警示) 區段。
- windowsEvents (選用)

要監控元件的 Windows 事件清單。只有 EC2 執行個體上的 Windows 才有 windowsEvents 區段。

- JMXPrometheusExporter （ 可選 ）

  JMXPrometheus Exporter 組態。

- hanaPrometheusExporter (選擇性)

  SAP HANA Prometheus Exporter 組態。

- haClusterPrometheus出口商（ 可選 ）

  HA Cluster Prometheus Exporter 組態。

- netWeaverPrometheus出口商（ 可選 ）

  SAP NetWeaver Prometheus 出口商配置。

- sapAsePrometheus出口商（ 可選 ）

  SAP ASE Prometheus Exporter 組態。

以下範例以 JSON 格式顯示子元件區段片段的語法。

```
[
  {
    "subComponentType" : "AWS::EC2::Instance",
    "alarmMetrics" : [
      list of alarm metrics
    ],
    "logs" : [
      list of logs
    ],
    "processes": [
      list of processes
    ],
    "windowsEvents" : [
      list of windows events channels configurations
    ]
  },
  {
    "subComponentType" : "AWS::EC2::Volume",
    "alarmMetrics" : [
      list of alarm metrics
```

```
    ]
  }
]
```

元件組態區段屬性

本節提供了個元件組態區段的屬性。

章節

- 指標
- Log
- 處理
- JMX Prometheus Exporter
- HANA Prometheus Exporter
- HA Cluster Prometheus Exporter
- NetWeaver Prometheus 出口商
- SAP ASE Prometheus Exporter
- Windows 事件
- 警示

指標

定義要針對元件監控的指標。

JSON

```
{
  "alarmMetricName" : "monitoredMetricName",
  "monitor" : true/false
}
```

屬性

- alarmMetricName (必填)

  要針對元件監控的指標名稱。如需 Application Insights 支援的指標，請參閱 Amazon CloudWatch 應用程式深入解析支援的日誌和指標。
- monitor (選用)

布林值，指出是否要監控該指標。預設值為 `true`。

Log

定義要針對元件監控的日誌。

JSON

```
{
  "logGroupName" : "logGroupName",
  "logPath" : "logPath",
  "logType" : "logType",
  "encoding" : "encodingType",
  "monitor" : true/false
}
```

屬性

- logGroupName (必填)

  要與受監控 CloudWatch 記錄檔相關聯的記錄群組名稱。如需記錄群組名稱限制的資訊，請參閱CreateLogGroup。

- logPath (EC2 執行個體元件需要；不使用 CloudWatch代理程式的元件則不需要，例如 AWS Lambda)

  要監控的日誌路徑。日誌路徑必須是 Windows 系統的檔案絕對路徑。如需詳細資訊，請參閱CloudWatch 代理程式組態檔：記錄檔段落。

- logType (必要)

  日誌類型會決定 Application Insights 用來分析日誌的日誌模式。日誌類型可從下列選項中選取：

  - `SQL_SERVER`
  - `MYSQL`
  - `MYSQL_SLOW_QUERY`
  - `POSTGRESQL`
  - `ORACLE_ALERT`
  - `ORACLE_LISTENER`
  - `IIS`

- APPLICATION
- WINDOWS_EVENTS
- WINDOWS_EVENTS_ACTIVE_DIRECTORY
- WINDOWS_EVENTS_DNS
- WINDOWS_EVENTS_IIS
- WINDOWS_EVENTS_SHAREPOINT
- SQL_SERVER_ALWAYSON_AVAILABILITY_GROUP
- SQL_SERVER_FAILOVER_CLUSTER_INSTANCE
- DEFAULT
- CUSTOM
- STEP_FUNCTION
- API_GATEWAY_ACCESS
- API_GATEWAY_EXECUTION
- SAP_HANA_LOGS
- SAP_HANA_TRACE
- SAP_HANA_HIGH_AVAILABILITY
- SAP_NETWEAVER_DEV_TRACE_LOGS
- PACEMAKER_HIGH_AVAILABILITY

- encoding (選用)

  要監控的日誌編碼類型。指定的編碼應包含在CloudWatch 代理程式支援的編碼清單中。如果未提供，「 CloudWatch 應用程式深入解析」會使用 utf-8 類型的預設編碼，但下列情況除外：

  - SQL_SERVER：utf-16 編碼
  - IIS：ascii 編碼

- monitor (選用)

  布林值，指出是否監控日誌。預設值為 true。

處理

定義要針對元件監控的程式。

```
{
  "processName" : "monitoredProcessName",
  "alarmMetrics" : [
      list of alarm metrics
  ]
}
```

屬性

- processName (必填)

  要針對元件監控的程式名稱。程式名稱不得包含程式主體，例如 `sqlservr` 或者 `sqlservr.exe`。
- alarmMetrics (必填)

  針對此程式要監控的指標清單。若要檢視應用程 CloudWatch 式深入解析支援的流程指標，請參閱 Amazon Elastic Compute Cloud (EC2)

JMX Prometheus Exporter

定義 JMX Prometheus Exporter 設定。

JSON

```
"JMXPrometheusExporter": {
  "jmxURL" : "JMX URL",
  "hostPort" : "The host and port",
  "prometheusPort" : "Target port to emit Prometheus metrics"
}
```

屬性

- jmxURL (選用)

  要連線的完整 JMX URL。
- hostPort (選用)

  透過遠端 JMX 連線的主機和連接埠。只能指定其中一個 `hostPort` 和 `jmxURL`。
- prometheusPort (選用)

  要傳送 Prometheus 指標的目標連接埠。如果未指定，則會使用預設連接埠 9404。

HANA Prometheus Exporter

定義 HANA Prometheus Exporter 設定。

JSON

```
"hanaPrometheusExporter": {
    "hanaSid": "SAP HANA  SID",
    "hanaPort": "HANA database port",
    "hanaSecretName": "HANA secret name",
    "prometheusPort": "Target port to emit Prometheus metrics"
}
```

屬性

- hanaSid

  SAP HANA 系統的三個字元 SAP 系統 ID (SID)。
- hanaPort

  匯出工具會用來查詢 HANA 指標的 HANA 資料庫連接埠。
- hanaSecretName

  存儲 HANA 監視用戶憑據的 AWS Secrets Manager 秘密。HANA Prometheus 匯出工具會使用這些憑證來連線至資料庫並查詢 HANA 指標。
- prometheusPort (選用)

  要 Prometheus 要傳送指標的目標連接埠。如果未指定，則會使用預設連接埠 9668。

HA Cluster Prometheus Exporter

定義 HA Cluster Prometheus Exporter 設定。

JSON

```
"haClusterPrometheusExporter": {
    "prometheusPort": "Target port to emit Prometheus metrics"
}
```

屬性

- prometheusPort (選用)

  要 Prometheus 要傳送指標的目標連接埠。如果未指定，則會使用預設連接埠 9664。

NetWeaver Prometheus 出口商

定義 NetWeaver Prometheus 匯出器設定。

JSON

```
"netWeaverPrometheusExporter": {
    "sapSid": "SAP NetWeaver  SID",
    "instanceNumbers": [ "Array of instance Numbers of SAP NetWeaver system "],
"prometheusPort": "Target port to emit Prometheus metrics"
}
```

屬性

- sapSid

  SAP 系統的三個字元 SAP 系統識別碼 (SID)。 NetWeaver
- instanceNumbers

  SAP NetWeaver 系統的執行個體編號陣列。

  範例：`"instanceNumbers": [ "00", "01"]`
- prometheusPort (選用)

  將 Prometheus 指標傳送到的目標連接埠。如果未指定，則會使用預設連接埠 9680。

SAP ASE Prometheus Exporter

定義 SAP ASE Prometheus Exporter 設定。

JSON

```
"sapASEPrometheusExporter": {
    "sapAseSid": "SAP ASE SID",
    "sapAsePort": "SAP ASE database port",
    "sapAseSecretName": "SAP ASE secret name",
    "prometheusPort": "Target port to emit Prometheus metrics",
```

```
    "agreeToEnableASEMonitoring": true
}
```

屬性

- sapAseSid

  SAP ASE 系統的三個字元 SAP 系統 ID (SID)。
- sapAsePort

  匯出工具會用來查詢 ASE 指標的 SAP ASE 資料庫連接埠。
- sapAseSecret姓名

  儲存 ASE 監視使用者認證的 AWS Secrets Manager 密碼。SAP ASE Prometheus 匯出工具會使用這些憑證來連線至資料庫並查詢 ASE 指標。
- prometheusPort (選用)

  要 Prometheus 要傳送指標的目標連接埠。如果未指定，則會使用預設連接埠 9399。如果有另一個使用預設連接埠的 ASE 資料庫，則我們將使用 9499。

Windows 事件

定義欲記錄的 Windows 事件。

JSON

```
{
  "logGroupName" : "logGroupName",
  "eventName" : "eventName",
  "eventLevels" : ["ERROR","WARNING","CRITICAL","INFORMATION","VERBOSE"],
  "monitor" : true/false
}
```

屬性

- logGroupName (必填)

  要與受監控 CloudWatch 記錄檔相關聯的記錄群組名稱。如需記錄群組名稱限制的資訊，請參閱CreateLogGroup。
- eventName (必要)

欲記錄的 Windows 事件類型。這和 Windows 事件日誌頻道名稱相同，例如，系統，安全 CustomEventName性等。要記錄的每個 Windows 事件類型都需要此欄位。

- eventLevels (必要)

  欲記錄的事件層級。您必須指定要記錄的每個層級。可能的值包括 `INFORMATION`、`WARNING`、`ERROR`、`CRITICAL` 及 `VERBOSE`。要記錄的每個 Windows 事件類型都需要此欄位。

- monitor (選用)

  布林值，指出是否監控日誌。預設值為 `true`。

警示

定義元件要監視的 CloudWatch 警示。

JSON

```
{
  "alarmName" : "monitoredAlarmName",
  "severity" : HIGH/MEDIUM/LOW
}
```

屬性

- 警示名稱 (必填)

  要監視元件的 CloudWatch 警示名稱。

- 嚴重性 (選用)

  指示警示影響的範圍程度。

元件組態範例

以下範例顯示相關服務的 JSON 格式元件組態。

元件組態範例

- Amazon DynamoDB 資料表
- Amazon EC2 Auto Scaling (ASG)

- Amazon EKS 叢集
- Amazon Elastic Compute Cloud (EC2) 執行個體
- Amazon Elastic Container Service (Amazon ECS)
- Amazon ECS 服務
- Amazon ECS 任務
- Amazon Elastic File System (Amazon EFS)
- Amazon FSx
- Amazon Relational Database Service (RDS) Aurora MySQL
- Amazon Relational Database Service (RDS) 執行個體
- Amazon Route 53 運作狀態檢查
- Amazon Route 53 託管區域
- Amazon Route 53 Resolver endpoint
- Amazon Route 53 Resolver 查詢記錄組態
- Amazon S3 儲存貯體
- Amazon Simple Queue Service (SQS)
- Amazon SNS 主題
- Amazon Virtual Private Cloud (Amazon VPC)
- Amazon VPC 網路位址轉譯 (NAT) 閘道
- API Gateway REST API 階段
- Application Elastic Load Balancing
- AWS Lambda 函數
- AWS Network Firewall 規則群組
- AWS Network Firewall 規則群組關聯
- AWS Step Functions
- 客戶分組的 Amazon EC2 執行個體
- Elastic Load Balancing
- Java
- Amazon EC2 上的 Kubernetes
- RDS MariaDB 和 RDS MySQL
- RDS Oracle

- RDS PostgreSQL
- Amazon EC2 上的 SAP ASE
- 在 Amazon EC2 上的 SAP ASE 高可用性
- Amazon EC2 上的 SAP HANA
- 在 Amazon EC2 上的 SAP HANA 高可用性
- Amazon EC2 NetWeaver 上的 SAP
- Amazon EC2 上的 SAP NetWeaver 高可用性
- SQL Always On 可用性群組
- SQL 容錯移轉叢集執行個體

Amazon DynamoDB 資料表

下列範例顯示 Amazon DynamoDB 資料表 JSON 格式的元件組態。

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "SystemErrors",
      "monitor": false
    },
    {
      "alarmMetricName": "UserErrors",
      "monitor": false
    },
    {
      "alarmMetricName": "ConsumedReadCapacityUnits",
      "monitor": false
    },
    {
      "alarmMetricName": "ConsumedWriteCapacityUnits",
      "monitor": false
    },
    {
      "alarmMetricName": "ReadThrottleEvents",
      "monitor": false
    },
    {
      "alarmMetricName": "WriteThrottleEvents",
      "monitor": false
```

```
    },
    {
      "alarmMetricName": "ConditionalCheckFailedRequests",
      "monitor": false
    },
    {
      "alarmMetricName": "TransactionConflict",
      "monitor": false
    }
  ],
  "logs": []
}
```

Amazon EC2 Auto Scaling (ASG)

下列範例顯示 Amazon EC2 Auto Scaling (ASG) JSON 格式的元件組態。

```
{
    "alarmMetrics" : [
      {
        "alarmMetricName" : "CPUCreditBalance"
      }, {
        "alarmMetricName" : "EBSIOBalance%"
      }
    ],
    "subComponents" : [
      {
        "subComponentType" : "AWS::EC2::Instance",
        "alarmMetrics" : [
          {
            "alarmMetricName" : "CPUUtilization"
          }, {
            "alarmMetricName" : "StatusCheckFailed"
          }
        ],
        "logs" : [
          {
            "logGroupName" : "my_log_group",
            "logPath" : "C:\\LogFolder\\*",
            "logType" : "APPLICATION"
          }
        ],
        "processes" : [
          {
```

```
            "processName" : "my_process",
            "alarmMetrics" : [
              {
                  "alarmMetricName" : "procstat cpu_usage",
                  "monitor" : true
              }, {
                  "alarmMetricName" : "procstat memory_rss",
                  "monitor" : true
              }
          ]
      }
  ],
        "windowsEvents" : [
          {
            "logGroupName" : "my_log_group_2",
            "eventName" : "Application",
            "eventLevels" : [ "ERROR", "WARNING", "CRITICAL" ]
          }
        ]
      }, {
        "subComponentType" : "AWS::EC2::Volume",
        "alarmMetrics" : [
          {
            "alarmMetricName" : "VolumeQueueLength"
          }, {
            "alarmMetricName" : "BurstBalance"
          }
        ]
      }
    ],
    "alarms" : [
      {
        "alarmName" : "my_asg_alarm",
        "severity" : "LOW"
      }
    ]
  }
```

Amazon EKS 叢集

下列範例顯示 Amazon EKS 叢集 JSON 格式的元件組態。

```
{
    "alarmMetrics":[
```

```
        {
           "alarmMetricName": "cluster_failed_node_count",
           "monitor":true
        },
        {
           "alarmMetricName": "node_cpu_reserved_capacity",
           "monitor":true
        },
        {
           "alarmMetricName": "node_cpu_utilization",
           "monitor":true
        },
        {
           "alarmMetricName": "node_filesystem_utilization",
           "monitor":true
        },
        {
           "alarmMetricName": "node_memory_reserved_capacity",
           "monitor":true
        },
        {
           "alarmMetricName": "node_memory_utilization",
           "monitor":true
        },
        {
           "alarmMetricName": "node_network_total_bytes",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_cpu_reserved_capacity",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_cpu_utilization",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_cpu_utilization_over_pod_limit",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_memory_reserved_capacity",
           "monitor":true
        },
```

```
        {
           "alarmMetricName": "pod_memory_utilization",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_memory_utilization_over_pod_limit",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_network_rx_bytes",
           "monitor":true
        },
        {
           "alarmMetricName": "pod_network_tx_bytes",
           "monitor":true
        }
     ],
     "logs":[
        {
           "logGroupName": "/aws/containerinsights/kubernetes/application",
           "logType":"APPLICATION",
           "monitor":true,
           "encoding":"utf-8"
        }
     ],
     "subComponents":[
        {
           "subComponentType":"AWS::EC2::Instance",
           "alarmMetrics":[
              {
                 "alarmMetricName":"CPUUtilization",
                 "monitor":true
              },
              {
                 "alarmMetricName":"StatusCheckFailed",
                 "monitor":true
              },
              {
                 "alarmMetricName":"disk_used_percent",
                 "monitor":true
              },
              {
                 "alarmMetricName":"mem_used_percent",
                 "monitor":true
```

```
            }
         ],
         "logs":[
            {
               "logGroupName":"APPLICATION-KubernetesClusterOnEC2-IAD",
               "logPath":"",
               "logType":"APPLICATION",
               "monitor":true,
               "encoding":"utf-8"
            }
         ],
         "processes" : [
           {
               "processName" : "my_process",
               "alarmMetrics" : [
                   {
                       "alarmMetricName" : "procstat cpu_usage",
                       "monitor" : true
                   }, {
                       "alarmMetricName" : "procstat memory_rss",
                       "monitor" : true
                   }
               ]
           }
       ],
         "windowsEvents":[
            {
               "logGroupName":"my_log_group_2",
               "eventName":"Application",
               "eventLevels":[
                  "ERROR",
                  "WARNING",
                  "CRITICAL"
               ],
               "monitor":true
            }
         ]
      },
      {
         "subComponentType":"AWS::AutoScaling::AutoScalingGroup",
         "alarmMetrics":[
            {
               "alarmMetricName":"CPUCreditBalance",
               "monitor":true
```

```
            },
            {
               "alarmMetricName":"EBSIOBalance%",
               "monitor":true
            }
         ]
      },
      {
         "subComponentType":"AWS::EC2::Volume",
         "alarmMetrics":[
            {
               "alarmMetricName":"VolumeReadBytes",
               "monitor":true
            },
            {
               "alarmMetricName":"VolumeWriteBytes",
               "monitor":true
            },
            {
               "alarmMetricName":"VolumeReadOps",
               "monitor":true
            },
            {
               "alarmMetricName":"VolumeWriteOps",
               "monitor":true
            },
            {
               "alarmMetricName":"VolumeQueueLength",
               "monitor":true
            },
            {
               "alarmMetricName":"BurstBalance",
               "monitor":true
            }
         ]
      }
   ]
}
```

> **Note**
>
> - `AWS::EC2::Instance`、`AWS::EC2::Volume` 和 `AWS::AutoScaling::AutoScalingGroup` 的 `subComponents` 區段僅適用於在 EC2 啟動類型上執行的 Amazon EKS 叢集。
> - `subComponents` 中的 `AWS::EC2::Instance` 的 `windowsEvents` 區段僅適用於在 Amazon EC2 執行個體上執行的 Windows。

Amazon Elastic Compute Cloud (EC2) 執行個體

下列範例顯示 Amazon EC2 執行個體 JSON 格式的元件組態。

> **Important**
>
> 當 Amazon EC2 執行個體進入 `stopped` 狀態時，系統便會將其從監控中移除。當它返回到`running`狀態時，會將其新增至「應用程式深入解析」主控台之「應用程式詳細資訊」頁面上的未受監控元件清單。 CloudWatch 如果為應用程序啟用對新資源的自動監控，則會將執行個體新增到 Monitored components (受監控的元件)。但是，系統會將工作負載的日誌和指標設為預設值。不會儲存先前的日誌和指標組態。

```
{
    "alarmMetrics" : [
      {
        "alarmMetricName" : "CPUUtilization",
        "monitor" : true
      }, {
        "alarmMetricName" : "StatusCheckFailed"
      }
    ],
    "logs" : [
      {
        "logGroupName" : "my_log_group",
        "logPath" : "C:\\LogFolder\\*",
        "logType" : "APPLICATION",
        "monitor" : true
      },
      {
        "logGroupName" : "my_log_group_2",
```

```
        "logPath" : "C:\\LogFolder2\\*",
        "logType" : "IIS",
        "encoding" : "utf-8"
      }
    ],
    "processes" : [
        {
            "processName" : "my_process",
            "alarmMetrics" : [
                {
                    "alarmMetricName" : "procstat cpu_usage",
                    "monitor" : true
                }, {
                    "alarmMetricName" : "procstat memory_rss",
                    "monitor" : true
                }
            ]
        }
    ],
    "windowsEvents" : [
      {
        "logGroupName" : "my_log_group_3",
        "eventName" : "Application",
        "eventLevels" : [ "ERROR", "WARNING", "CRITICAL" ],
        "monitor" : true
      }, {
        "logGroupName" : "my_log_group_4",
        "eventName" : "System",
        "eventLevels" : [ "ERROR", "WARNING", "CRITICAL" ],
        "monitor" : true
    }],
     "alarms" : [
      {
        "alarmName" : "my_instance_alarm_1",
        "severity" : "HIGH"
      },
      {
        "alarmName" : "my_instance_alarm_2",
        "severity" : "LOW"
      }
    ],
     "subComponents" : [
     {
       "subComponentType" : "AWS::EC2::Volume",
```

```
      "alarmMetrics" : [
      {
        "alarmMetricName" : "VolumeQueueLength",
        "monitor" : "true"
      },
      {
        "alarmMetricName" : "VolumeThroughputPercentage",
        "monitor" : "true"
      },
      {
        "alarmMetricName" : "BurstBalance",
        "monitor" : "true"
      }
    }]
  }
```

Amazon Elastic Container Service (Amazon ECS)

下列範例顯示 Amazon Elastic Container Service (Amazon ECS) JSON 格式的元件組態。

```
{
    "alarmMetrics":[
       {
          "alarmMetricName":"CpuUtilized",
          "monitor":true
       },
       {
          "alarmMetricName":"MemoryUtilized",
          "monitor":true
       },
       {
          "alarmMetricName":"NetworkRxBytes",
          "monitor":true
       },
       {
          "alarmMetricName":"NetworkTxBytes",
          "monitor":true
       },
       {
          "alarmMetricName":"RunningTaskCount",
          "monitor":true
       },
       {
          "alarmMetricName":"PendingTaskCount",
```

```
            "monitor":true
         },
         {
            "alarmMetricName":"StorageReadBytes",
            "monitor":true
         },
         {
            "alarmMetricName":"StorageWriteBytes",
            "monitor":true
         }
      ],
      "logs":[
         {
            "logGroupName":"/ecs/my-task-definition",
            "logType":"APPLICATION",
            "monitor":true
         }
      ],
      "subComponents":[
         {
            "subComponentType":"AWS::ElasticLoadBalancing::LoadBalancer",
            "alarmMetrics":[
               {
                  "alarmMetricName":"HTTPCode_Backend_4XX",
                  "monitor":true
               },
               {
                  "alarmMetricName":"HTTPCode_Backend_5XX",
                  "monitor":true
               },
               {
                  "alarmMetricName":"Latency",
                  "monitor":true
               },
               {
                  "alarmMetricName":"SurgeQueueLength",
                  "monitor":true
               },
               {
                  "alarmMetricName":"UnHealthyHostCount",
                  "monitor":true
               }
            ]
         },
```

```
        {
           "subComponentType":"AWS::ElasticLoadBalancingV2::LoadBalancer",
           "alarmMetrics":[
              {
                 "alarmMetricName":"HTTPCode_Target_4XX_Count",
                 "monitor":true
              },
              {
                 "alarmMetricName":"HTTPCode_Target_5XX_Count",
                 "monitor":true
              },
              {
                 "alarmMetricName":"TargetResponseTime",
                 "monitor":true
              },
              {
                 "alarmMetricName":"UnHealthyHostCount",
                 "monitor":true
              }
           ]
        },
        {
           "subComponentType":"AWS::EC2::Instance",
           "alarmMetrics":[
              {
                 "alarmMetricName":"CPUUtilization",
                 "monitor":true
              },
              {
                 "alarmMetricName":"StatusCheckFailed",
                 "monitor":true
              },
              {
                 "alarmMetricName":"disk_used_percent",
                 "monitor":true
              },
              {
                 "alarmMetricName":"mem_used_percent",
                 "monitor":true
              }
           ],
           "logs":[
              {
                 "logGroupName":"my_log_group",
```

```
                "logPath":"/mylog/path",
                "logType":"APPLICATION",
                "monitor":true
            }
        ],
        "processes" : [
            {
                "processName" : "my_process",
                "alarmMetrics" : [
                    {
                        "alarmMetricName" : "procstat cpu_usage",
                        "monitor" : true
                    }, {
                        "alarmMetricName" : "procstat memory_rss",
                        "monitor" : true
                    }
                ]
            }
        ],
        "windowsEvents":[
            {
                "logGroupName":"my_log_group_2",
                "eventName":"Application",
                "eventLevels":[
                    "ERROR",
                    "WARNING",
                    "CRITICAL"
                ],
                "monitor":true
            }
        ]
    },
    {
        "subComponentType":"AWS::EC2::Volume",
        "alarmMetrics":[
            {
                "alarmMetricName":"VolumeQueueLength",
                "monitor":"true"
            },
            {
                "alarmMetricName":"VolumeThroughputPercentage",
                "monitor":"true"
            },
            {
```

```
                "alarmMetricName":"BurstBalance",
                "monitor":"true"
             }
          ]
       }
    ]
 }
```

> **Note**
>
> - `AWS::EC2::Instance` 和 `AWS::EC2::Volume` 的 `subComponents` 區段僅適用於具有在 EC2 啟動類型上執行的 ECS 服務或 ECS 任務的 Amazon ECS 叢集。
> - `subComponents` 中的 `AWS::EC2::Instance` 的 `windowsEvents` 區段僅適用於在 Amazon EC2 執行個體上執行的 Windows。

Amazon ECS 服務

下列範例顯示 Amazon ECS 服務 JSON 格式的元件組態。

```
{
    "alarmMetrics":[
       {
          "alarmMetricName":"CPUUtilization",
          "monitor":true
       },
       {
          "alarmMetricName":"MemoryUtilization",
          "monitor":true
       },
       {
          "alarmMetricName":"CpuUtilized",
          "monitor":true
       },
       {
          "alarmMetricName":"MemoryUtilized",
          "monitor":true
       },
       {
          "alarmMetricName":"NetworkRxBytes",
          "monitor":true
```

```
        },
        {
           "alarmMetricName":"NetworkTxBytes",
           "monitor":true
        },
        {
           "alarmMetricName":"RunningTaskCount",
           "monitor":true
        },
        {
           "alarmMetricName":"PendingTaskCount",
           "monitor":true
        },
        {
           "alarmMetricName":"StorageReadBytes",
           "monitor":true
        },
        {
           "alarmMetricName":"StorageWriteBytes",
           "monitor":true
        }
     ],
     "logs":[
        {
           "logGroupName":"/ecs/my-task-definition",
           "logType":"APPLICATION",
           "monitor":true
        }
     ],
     "subComponents":[
        {
           "subComponentType":"AWS::ElasticLoadBalancing::LoadBalancer",
           "alarmMetrics":[
              {
                 "alarmMetricName":"HTTPCode_Backend_4XX",
                 "monitor":true
              },
              {
                 "alarmMetricName":"HTTPCode_Backend_5XX",
                 "monitor":true
              },
              {
                 "alarmMetricName":"Latency",
                 "monitor":true
```

```
            },
            {
               "alarmMetricName":"SurgeQueueLength",
               "monitor":true
            },
            {
               "alarmMetricName":"UnHealthyHostCount",
               "monitor":true
            }
         ]
      },
      {
         "subComponentType":"AWS::ElasticLoadBalancingV2::LoadBalancer",
         "alarmMetrics":[
            {
               "alarmMetricName":"HTTPCode_Target_4XX_Count",
               "monitor":true
            },
            {
               "alarmMetricName":"HTTPCode_Target_5XX_Count",
               "monitor":true
            },
            {
               "alarmMetricName":"TargetResponseTime",
               "monitor":true
            },
            {
               "alarmMetricName":"UnHealthyHostCount",
               "monitor":true
            }
         ]
      },
      {
         "subComponentType":"AWS::EC2::Instance",
         "alarmMetrics":[
            {
               "alarmMetricName":"CPUUtilization",
               "monitor":true
            },
            {
               "alarmMetricName":"StatusCheckFailed",
               "monitor":true
            },
            {
```

```
                "alarmMetricName":"disk_used_percent",
                "monitor":true
             },
             {
                "alarmMetricName":"mem_used_percent",
                "monitor":true
             }
          ],
          "logs":[
             {
                "logGroupName":"my_log_group",
                "logPath":"/mylog/path",
                "logType":"APPLICATION",
                "monitor":true
             }
          ],
          "processes" : [
             {
                "processName" : "my_process",
                "alarmMetrics" : [
                {
                    "alarmMetricName" : "procstat cpu_usage",
                    "monitor" : true
                }, {
                    "alarmMetricName" : "procstat memory_rss",
                    "monitor" : true
                }
            ]
        }
    ],
          "windowsEvents":[
             {
                "logGroupName":"my_log_group_2",
                "eventName":"Application",
                "eventLevels":[
                   "ERROR",
                   "WARNING",
                   "CRITICAL"
                ],
                "monitor":true
             }
          ]
       },
       {
```

```
            "subComponentType":"AWS::EC2::Volume",
            "alarmMetrics":[
               {
                  "alarmMetricName":"VolumeQueueLength",
                  "monitor":"true"
               },
               {
                  "alarmMetricName":"VolumeThroughputPercentage",
                  "monitor":"true"
               },
               {
                  "alarmMetricName":"BurstBalance",
                  "monitor":"true"
               }
            ]
         }
      ]
   }
```

> **Note**
>
> - `AWS::EC2::Instance` 和 `AWS::EC2::Volume` 的 `subComponents` 區段僅適用於在 EC2 啟動類型上執行的 Amazon ECS。
> - `subComponents` 中的 `AWS::EC2::Instance` 的 `windowsEvents` 區段僅適用於在 Amazon EC2 執行個體上執行的 Windows。

Amazon ECS 任務

下列範例顯示 Amazon ECS 任務 JSON 格式的元件組態。

```
{
    "logs":[
       {
          "logGroupName":"/ecs/my-task-definition",
          "logType":"APPLICATION",
          "monitor":true
       }
    ],
    "processes" : [
        {
```

```
            "processName" : "my_process",
            "alarmMetrics" : [
                {
                    "alarmMetricName" : "procstat cpu_usage",
                    "monitor" : true
                }, {
                    "alarmMetricName" : "procstat memory_rss",
                    "monitor" : true
                }
            ]
        }
    ]
}
```

Amazon Elastic File System (Amazon EFS)

下列範例顯示 Amazon EFS JSON 格式的元件組態。

```
{
   "alarmMetrics": [
     {
       "alarmMetricName": "BurstCreditBalance",
       "monitor": true
     },
     {
       "alarmMetricName": "PercentIOLimit",
       "monitor": true
     },
     {
       "alarmMetricName": "PermittedThroughput",
       "monitor": true
     },
     {
       "alarmMetricName": "MeteredIOBytes",
       "monitor": true
     },
     {
       "alarmMetricName": "TotalIOBytes",
       "monitor": true
     },
     {
       "alarmMetricName": "DataWriteIOBytes",
       "monitor": true
     },
```

```
    {
      "alarmMetricName": "DataReadIOBytes",
      "monitor": true
    },
    {
      "alarmMetricName": "MetadataIOBytes",
      "monitor": true
    },
    {
      "alarmMetricName": "ClientConnections",
      "monitor": true
    },
    {
      "alarmMetricName": "TimeSinceLastSync",
      "monitor": true
    },
    {
      "alarmMetricName": "Throughput",
      "monitor": true
    },
    {
      "alarmMetricName": "PercentageOfPermittedThroughputUtilization",
      "monitor": true
    },
    {
      "alarmMetricName": "ThroughputIOPS",
      "monitor": true
    },
    {
      "alarmMetricName": "PercentThroughputDataReadIOBytes",
      "monitor": true
    },
    {
      "alarmMetricName": "PercentThroughputDataWriteIOBytes",
      "monitor": true
    },
    {
      "alarmMetricName": "PercentageOfIOPSDataReadIOBytes",
      "monitor": true
    },
    {
      "alarmMetricName": "PercentageOfIOPSDataWriteIOBytes",
      "monitor": true
    },
```

```
    {
      "alarmMetricName": "AverageDataReadIOBytesSize",
      "monitor": true
    },
    {
      "alarmMetricName": "AverageDataWriteIOBytesSize",
      "monitor": true
    }
  ],
  "logs": [
   {
   "logGroupName": "/aws/efs/utils",
   "logType": "EFS_MOUNT_STATUS",
   "monitor": true,
   }
  ]
}
```

Amazon FSx

下列範例顯示 Amazon FSx JSON 格式的元件組態。

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "DataReadBytes",
      "monitor": true
    },
    {
      "alarmMetricName": "DataWriteBytes",
      "monitor": true
    },
    {
      "alarmMetricName": "DataReadOperations",
      "monitor": true
    },
    {
      "alarmMetricName": "DataWriteOperations",
      "monitor": true
    },
    {
      "alarmMetricName": "MetadataOperations",
      "monitor": true
    },
```

```
    {
      "alarmMetricName": "FreeStorageCapacity",
      "monitor": true
    }
  ]
}
```

Amazon Relational Database Service (RDS) Aurora MySQL

下列範例顯示 Amazon RDS Aurora MySQL 執行個體 JSON 格式的元件組態。

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "CPUUtilization",
      "monitor": true
    },
    {
      "alarmMetricName": "CommitLatency",
      "monitor": true
    }
  ],
  "logs": [
    {
      "logType": "MYSQL",
      "monitor": true,
    },
    {
      "logType": "MYSQL_SLOW_QUERY",
      "monitor": false
    }
  ]
}
```

Amazon Relational Database Service (RDS) 執行個體

下列範例顯示 Amazon RDS 執行個體 JSON 格式的元件組態。

```
{
  "alarmMetrics" : [
    {
      "alarmMetricName" : "BurstBalance",
```

```
      "monitor" : true
    }, {
      "alarmMetricName" : "WriteThroughput",
      "monitor" : false
    }
  ],

  "alarms" : [
    {
      "alarmName" : "my_rds_instance_alarm",
      "severity" : "MEDIUM"
    }
  ]
}
```

Amazon Route 53 運作狀態檢查

下列範例顯示 Amazon Route 53 運作狀態檢查的 JSON 格式元件組態。

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "ChildHealthCheckHealthyCount",
      "monitor": true
    },
    {
      "alarmMetricName": "ConnectionTime",
      "monitor": true
    },
    {
      "alarmMetricName": "HealthCheckPercentageHealthy",
      "monitor": true
    },
    {
      "alarmMetricName": "HealthCheckStatus",
      "monitor": true
    },
    {
      "alarmMetricName": "SSLHandshakeTime",
      "monitor": true
    },
    {
      "alarmMetricName": "TimeToFirstByte",
      "monitor": true
```

```
    }
  ]
}
```

Amazon Route 53 託管區域

下列範例顯示 Amazon Route 53 託管區域 的 JSON 格式元件組態。

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "DNSQueries",
      "monitor": true
    },
    {
      "alarmMetricName": "DNSSECInternalFailure",
      "monitor": true
    },
    {
      "alarmMetricName": "DNSSECKeySigningKeysNeedingAction",
      "monitor": true
    },
    {
      "alarmMetricName": "DNSSECKeySigningKeyMaxNeedingActionAge",
      "monitor": true
    },
    {
      "alarmMetricName": "DNSSECKeySigningKeyAge",
      "monitor": true
    }
  ],
  "logs": [
    {
      "logGroupName":"/hosted-zone/logs",
      "logType": "ROUTE53_DNS_PUBLIC_QUERY_LOGS",
      "monitor": true
    }
  ]
}
```

Amazon Route 53 Resolver endpoint

下列範例顯示 Amazon Route 53 Resolver 端點採用 JSON 格式的元件組態。

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "EndpointHealthyENICount",
      "monitor": true
    },
    {
      "alarmMetricName": "EndpointUnHealthyENICount",
      "monitor": true
    },
    {
      "alarmMetricName": "InboundQueryVolume",
      "monitor": true
    },
    {
      "alarmMetricName": "OutboundQueryVolume",
      "monitor": true
    },
    {
      "alarmMetricName": "OutboundQueryAggregateVolume",
      "monitor": true
    }
  ]
}
```

Amazon Route 53 Resolver 查詢記錄組態

下列範例顯示 Amazon Route 53 Resolver 查詢日誌記錄組態的 JSON 格式元件組態。

```
{
  "logs": [
    {
      "logGroupName": "/resolver-query-log-config/logs",
      "logType": "ROUTE53_RESOLVER_QUERY_LOGS",
      "monitor": true
    }
  ]
}
```

Amazon S3 儲存貯體

下列範例顯示 Amazon S3 儲存貯體 JSON 格式的元件組態。

```
{
    "alarmMetrics" : [
        {
            "alarmMetricName" : "ReplicationLatency",
            "monitor" : true
        }, {
            "alarmMetricName" : "5xxErrors",
            "monitor" : true
        }, {
            "alarmMetricName" : "BytesDownloaded"
            "monitor" : true
        }
    ]
}
```

Amazon Simple Queue Service (SQS)

下列範例顯示 Amazon Simple Queue Service JSON 格式的元件組態。

```
{
  "alarmMetrics" : [
    {
      "alarmMetricName" : "ApproximateAgeOfOldestMessage"
    }, {
      "alarmMetricName" : "NumberOfEmptyReceives"
    }
  ],
  "alarms" : [
    {
      "alarmName" : "my_sqs_alarm",
      "severity" : "MEDIUM"
    }
  ]
}
```

Amazon SNS 主題

下列範例顯示 Amazon SNS 主題 JSON 格式的元件組態。

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "NumberOfNotificationsFailed",
```

```
      "monitor": true
    },
    {
      "alarmMetricName": "NumberOfNotificationsFilteredOut-InvalidAttributes",
      "monitor": true
    },
    {
      "alarmMetricName": "NumberOfNotificationsFilteredOut-NoMessageAttributes",
      "monitor": true
    },
    {
      "alarmMetricName": "NumberOfNotificationsFailedToRedriveToDlq",
      "monitor": true
    }
  ]
}
```

Amazon Virtual Private Cloud (Amazon VPC)

下列範例顯示 Amazon VPC 的 JSON 格式元件組態。

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "NetworkAddressUsage",
      "monitor": true
    },
    {
      "alarmMetricName": "NetworkAddressUsagePeered",
      "monitor": true
    },
    {
      "alarmMetricName": "VPCFirewallQueryVolume",
      "monitor": true
    }
  ]
}
```

Amazon VPC 網路位址轉譯 (NAT) 閘道

下列範例顯示 NAT 閘道的 JSON 格式元件組態。

```
{
```

```
  "alarmMetrics": [
    {
      "alarmMetricName": "ErrorPortAllocation",
      "monitor": true
    },
    {
      "alarmMetricName": "IdleTimeoutCount",
      "monitor": true
    }
  ]
}
```

API Gateway REST API 階段

下列範例顯示 API Gateway REST API 階段 JSON 格式的元件組態。

```
{
     "alarmMetrics" : [
         {
             "alarmMetricName" : "4XXError",
             "monitor" : true
         },
         {
             "alarmMetricName" : "5XXError",
             "monitor" : true
         }
     ],
    "logs" : [
        {
            "logType" : "API_GATEWAY_EXECUTION",
            "monitor" : true
        },
        {
            "logType" : "API_GATEWAY_ACCESS",
            "monitor" : true
        }
    ]
}
```

Application Elastic Load Balancing

下列範例顯示 Application Elastic Load Balancing JSON 格式的元件組態。

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "ActiveConnectionCount",
    }, {
      "alarmMetricName": "TargetResponseTime"
    }
  ],
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "CPUUtilization",
        }, {
          "alarmMetricName": "StatusCheckFailed"
        }
      ],
      "logs": [
        {
          "logGroupName": "my_log_group",
          "logPath": "C:\\LogFolder\\*",
          "logType": "APPLICATION",
        }
      ],
      "windowsEvents": [
        {
          "logGroupName": "my_log_group_2",
          "eventName": "Application",
          "eventLevels": [ "ERROR", "WARNING", "CRITICAL" ]
        }
      ]
    }, {
      "subComponentType": "AWS::EC2::Volume",
      "alarmMetrics": [
        {
          "alarmMetricName": "VolumeQueueLength",
        }, {
          "alarmMetricName": "BurstBalance"
        }
      ]
    }
  ],
```

```
  "alarms": [
    {
      "alarmName": "my_alb_alarm",
      "severity": "LOW"
    }
  ]
}
```

AWS Lambda 函數

下列範例顯示 AWS Lambda Function JSON 格式的元件組態。

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "Errors",
      "monitor": true
    },
    {
      "alarmMetricName": "Throttles",
      "monitor": true
    },
    {
      "alarmMetricName": "IteratorAge",
      "monitor": true
    },
    {
      "alarmMetricName": "Duration",
      "monitor": true
    }
  ],
  "logs": [
    {
      "logType": "DEFAULT",
      "monitor": true
    }
  ]
}
```

AWS Network Firewall 規則群組

下列範例顯示 AWS Network Firewall 規則群組的 JSON 格式元件組態。

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "FirewallRuleGroupQueryVolume",
      "monitor": true
    }
  ]
}
```

AWS Network Firewall 規則群組關聯

下列範例顯示 AWS Network Firewall 規則群組關聯的 JSON 格式元件組態。

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "FirewallRuleGroupQueryVolume",
      "monitor": true
    }
  ]
}
```

AWS Step Functions

下列範例顯示 AWS Step Functions JSON 格式的元件組態。

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "ExecutionsFailed",
      "monitor": true
    },
    {
      "alarmMetricName": "LambdaFunctionsFailed",
      "monitor": true
    },
    {
      "alarmMetricName": "ProvisionedRefillRate",
      "monitor": true
    }
  ],
  "logs": [
    {
```

```
     "logGroupName": "/aws/states/HelloWorld-Logs",
      "logType": "STEP_FUNCTION",
      "monitor": true,
    }
  ]
}
```

客戶分組的 Amazon EC2 執行個體

下列範例顯示客戶分組之 Amazon EC2 執行個體 JSON 格式的元件組態。

```
{
    "subComponents": [
      {
        "subComponentType": "AWS::EC2::Instance",
        "alarmMetrics": [
          {
            "alarmMetricName": "CPUUtilization",
          },
          {
            "alarmMetricName": "StatusCheckFailed"
          }
        ],
        "logs": [
          {
            "logGroupName": "my_log_group",
            "logPath": "C:\\LogFolder\\*",
            "logType": "APPLICATION",
          }
        ],
        "processes": [
            {
                "processName": "my_process",
                "alarmMetrics": [
                    {
                        "alarmMetricName": "procstat cpu_usage",
                        "monitor": true
                    }, {
                        "alarmMetricName": "procstat memory_rss",
                        "monitor": true
                    }
                ]
            }
        ],
```

```
        "windowsEvents": [
          {
            "logGroupName": "my_log_group_2",
            "eventName": "Application",
            "eventLevels": [ "ERROR", "WARNING", "CRITICAL" ]
          }
        ]
      }, {
        "subComponentType": "AWS::EC2::Volume",
        "alarmMetrics": [
          {
            "alarmMetricName": "VolumeQueueLength",
          }, {
            "alarmMetricName": "BurstBalance"
          }
        ]
      }
    ],
    "alarms": [
      {
        "alarmName": "my_alarm",
        "severity": "MEDIUM"
      }
    ]
  }
```

Elastic Load Balancing

下列範例顯示 Elastic Load Balancing JSON 格式的元件組態。

```
{
    "alarmMetrics": [
      {
        "alarmMetricName": "EstimatedALBActiveConnectionCount"
      }, {
        "alarmMetricName": "HTTPCode_Backend_5XX"
      }
    ],
    "subComponents": [
      {
        "subComponentType": "AWS::EC2::Instance",
        "alarmMetrics": [
          {
            "alarmMetricName": "CPUUtilization"
```

```
          }, {
            "alarmMetricName": "StatusCheckFailed"
          }
        ],
        "logs": [
          {
            "logGroupName": "my_log_group",
            "logPath": "C:\\LogFolder\\*",
            "logType": "APPLICATION"
          }
        ],
        "processes": [
          {
            "processName": "my_process",
            "alarmMetrics": [
                {
                  "alarmMetricName": "procstat cpu_usage",
                  "monitor": true
                }, {
                  "alarmMetricName": "procstat memory_rss",
                  "monitor": true
                }
              ]
          }
        ],
        "windowsEvents": [
          {
            "logGroupName": "my_log_group_2",
            "eventName": "Application",
            "eventLevels": [ "ERROR", "WARNING", "CRITICAL" ],
            "monitor": true
          }
        ]
      }, {
        "subComponentType": "AWS::EC2::Volume",
        "alarmMetrics": [
          {
            "alarmMetricName": "VolumeQueueLength"
          }, {
            "alarmMetricName": "BurstBalance"
          }
        ]
      }
    ],
```

```
    "alarms": [
      {
        "alarmName": "my_elb_alarm",
        "severity": "HIGH"
      }
    ]
  }
```

Java

下列範例顯示 Java JSON 格式的元件組態。

```
{
  "alarmMetrics": [ {
    "alarmMetricName": "java_lang_threading_threadcount",
    "monitor": true
  },
  {
    "alarmMetricName": "java_lang_memory_heapmemoryusage_used",
    "monitor": true
  },
  {
    "alarmMetricName": "java_lang_memory_heapmemoryusage_committed",
    "monitor": true
  }],
  "logs": [ ],
  "JMXPrometheusExporter": {
      "hostPort": "8686",
      "prometheusPort": "9404"
  }
}
```

> **Note**
>
> Application Insights 不支援為 Prometheus JMX Exporter 設定身分驗證。如需如何設定身分驗證的資訊，請參閱 Prometheus JMX Exporter 範例組態。

Amazon EC2 上的 Kubernetes

下列範例顯示 Amazon EC2 上 Kubernetes JSON 格式的元件組態。

```
{
    "alarmMetrics":[
       {
          "alarmMetricName":"cluster_failed_node_count",
          "monitor":true
       },
       {
          "alarmMetricName":"node_cpu_reserved_capacity",
          "monitor":true
       },
       {
          "alarmMetricName":"node_cpu_utilization",
          "monitor":true
       },
       {
          "alarmMetricName":"node_filesystem_utilization",
          "monitor":true
       },
       {
          "alarmMetricName":"node_memory_reserved_capacity",
          "monitor":true
       },
       {
          "alarmMetricName":"node_memory_utilization",
          "monitor":true
       },
       {
          "alarmMetricName":"node_network_total_bytes",
          "monitor":true
       },
       {
          "alarmMetricName":"pod_cpu_reserved_capacity",
          "monitor":true
       },
       {
          "alarmMetricName":"pod_cpu_utilization",
          "monitor":true
       },
       {
          "alarmMetricName":"pod_cpu_utilization_over_pod_limit",
          "monitor":true
       },
       {
```

```
            "alarmMetricName":"pod_memory_reserved_capacity",
            "monitor":true
         },
         {
            "alarmMetricName":"pod_memory_utilization",
            "monitor":true
         },
         {
            "alarmMetricName":"pod_memory_utilization_over_pod_limit",
            "monitor":true
         },
         {
            "alarmMetricName":"pod_network_rx_bytes",
            "monitor":true
         },
         {
            "alarmMetricName":"pod_network_tx_bytes",
            "monitor":true
         }
      ],
      "logs":[
         {
            "logGroupName":"/aws/containerinsights/kubernetes/application",
            "logType":"APPLICATION",
            "monitor":true,
            "encoding":"utf-8"
         }
      ],
      "subComponents":[
         {
            "subComponentType":"AWS::EC2::Instance",
            "alarmMetrics":[
               {
                  "alarmMetricName":"CPUUtilization",
                  "monitor":true
               },
               {
                  "alarmMetricName":"StatusCheckFailed",
                  "monitor":true
               },
               {
                  "alarmMetricName":"disk_used_percent",
                  "monitor":true
               },
```

```
            {
               "alarmMetricName":"mem_used_percent",
               "monitor":true
            }
         ],
         "logs":[
            {
               "logGroupName":"APPLICATION-KubernetesClusterOnEC2-IAD",
               "logPath":"",
               "logType":"APPLICATION",
               "monitor":true,
               "encoding":"utf-8"
            }
         ],
         "processes" : [
           {
               "processName" : "my_process",
               "alarmMetrics" : [
                   {
                       "alarmMetricName" : "procstat cpu_usage",
                       "monitor" : true
                   }, {
                       "alarmMetricName" : "procstat memory_rss",
                       "monitor" : true
                   }
               ]
           }
       ]
      },
      {
         "subComponentType":"AWS::EC2::Volume",
         "alarmMetrics":[
            {
               "alarmMetricName":"VolumeReadBytes",
               "monitor":true
            },
            {
               "alarmMetricName":"VolumeWriteBytes",
               "monitor":true
            },
            {
               "alarmMetricName":"VolumeReadOps",
               "monitor":true
            },
```

```
            {
               "alarmMetricName":"VolumeWriteOps",
               "monitor":true
            },
            {
               "alarmMetricName":"VolumeQueueLength",
               "monitor":true
            },
            {
               "alarmMetricName":"BurstBalance",
               "monitor":true
            }
         ]
      }
   ]
}
```

RDS MariaDB 和 RDS MySQL

下列範例顯示 RDS MariaDB 和 RDS MySQL 之 JSON 格式的元件組態。

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "CPUUtilization",
      "monitor": true
    }
  ],
  "logs": [
    {
      "logType": "MYSQL",
      "monitor": true,
    },
    {
      "logType": "MYSQL_SLOW_QUERY",
      "monitor": false
    }
  ]
}
```

RDS Oracle

下列範例顯示 RDS Oracle 執行個體 JSON 格式的元件組態。

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "CPUUtilization",
      "monitor": true
    }
  ],
  "logs": [
    {
      "logType": "ORACLE_ALERT",
      "monitor": true,
    },
    {
      "logType": "ORACLE_LISTENER",
      "monitor": false
    }
  ]
}
```

RDS PostgreSQL

下列範例顯示 RDS PostgreSQL JSON 格式的元件組態。

```
{
  "alarmMetrics": [
    {
      "alarmMetricName": "CPUUtilization",
      "monitor": true
    }
  ],
  "logs": [
    {
      "logType": "POSTGRESQL",
      "monitor": true
    }
  ]
}
```

Amazon EC2 上的 SAP ASE

下列範例顯示 Amazon EC2 上 SAP ASE JSON 格式的元件組態。

```
{
```

```
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "asedb_database_availability",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_trunc_log_on_chkpt_enabled",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_last_db_backup_age_in_days",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_last_transaction_log_backup_age_in_hours",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_suspected_database",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_db_space_usage_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_db_log_space_usage_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_locked_login",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_data_cache_hit_ratio",
          "monitor": true
        }
      ],
      "logs": [
        {
          "logGroupName": "SAP_ASE_SERVER_LOGS-my-resource-group",
```

```
          "logPath": "/sybase/SY2/ASE-*/install/SY2.log",
          "logType": "SAP_ASE_SERVER_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_ASE_BACKUP_SERVER_LOGS-my-resource-group",
          "logPath": "/sybase/SY2/ASE-*/install/SY2_BS.log",
          "logType": "SAP_ASE_BACKUP_SERVER_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        }
  ],
  "sapAsePrometheusExporter": {
    "sapAseSid": "ASE",
    "sapAsePort": "4901",
    "sapAseSecretName": "ASE_DB_CREDS",
    "prometheusPort": "9399",
    "agreeToEnableASEMonitoring": true
}
```

在 Amazon EC2 上的 SAP ASE 高可用性

下列範例顯示在 Amazon EC2 上的 SAP ASE 高可用性 JSON 格式的元件組態。

```
{
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "asedb_database_availability",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_trunc_log_on_chkpt_enabled",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_last_db_backup_age_in_days",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_last_transaction_log_backup_age_in_hours",
```

```
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_suspected_database",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_db_space_usage_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_ha_replication_state",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_ha_replication_mode",
          "monitor": true
        },
        {
          "alarmMetricName": "asedb_ha_replication_latency_in_minutes",
          "monitor": true
        }
      ],
      "logs": [
        {
          "logGroupName": "SAP_ASE_SERVER_LOGS-my-resource-group",
          "logPath": "/sybase/SY2/ASE-*/install/SY2.log",
          "logType": "SAP_ASE_SERVER_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_ASE_BACKUP_SERVER_LOGS-my-resource-group",
          "logPath": "/sybase/SY2/ASE-*/install/SY2_BS.log",
          "logType": "SAP_ASE_BACKUP_SERVER_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_ASE_REP_SERVER_LOGS-my-resource-group",
          "logPath": "/sybase/SY2/DM/repservername/repservername.log",
          "logType": "SAP_ASE_REP_SERVER_LOGS",
          "monitor": true,
          "encoding": "utf-8"
```

```
        },
        {
          "logGroupName": "SAP_ASE_RMA_AGENT_LOGS-my-resource-group",
          "logPath": "/sybase/SY2/DM/RMA-*/instances/AgentContainer/logs/",
          "logType": "SAP_ASE_RMA_AGENT_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_ASE_FAULT_MANAGER_LOGS-my-resource-group",
          "logPath": "/opt/sap/FaultManager/dev_sybdbfm",
          "logType": "SAP_ASE_FAULT_MANAGER_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        }
  ],
  "sapAsePrometheusExporter": {
    "sapAseSid": "ASE",
    "sapAsePort": "4901",
    "sapAseSecretName": "ASE_DB_CREDS",
    "prometheusPort": "9399",
    "agreeToEnableASEMonitoring": true
}
```

Amazon EC2 上的 SAP HANA

下列範例顯示 Amazon EC2 上 SAP HANA JSON 格式的元件組態。

```
{
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "hanadb_server_startup_time_variations_seconds",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_level_5_alerts_count",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_level_4_alerts_count",
          "monitor": true
```

```
        },
        {
          "alarmMetricName": "hanadb_out_of_memory_events_count",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_max_trigger_read_ratio_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_table_allocation_limit_used_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_cpu_usage_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_plan_cache_hit_ratio_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_last_data_backup_age_days",
          "monitor": true
        }
      ],
      "logs": [
        {
          "logGroupName": "SAP_HANA_TRACE-my-resourge-group",
          "logPath": "/usr/sap/HDB/HDB00/*/trace/*.trc",
          "logType": "SAP_HANA_TRACE",
          "monitor": true,
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_HANA_LOGS-my-resource-group",
          "logPath": "/usr/sap/HDB/HDB00/*/trace/*.log",
          "logType": "SAP_HANA_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        }
      ]
    }
  ],
```

```
  "hanaPrometheusExporter": {
    "hanaSid": "HDB",
    "hanaPort": "30013",
    "hanaSecretName": "HANA_DB_CREDS",
    "prometheusPort": "9668"
  }
}
```

在 Amazon EC2 上的 SAP HANA 高可用性

下列範例顯示在 Amazon EC2 上的 SAP HANA 高可用性 JSON 格式的元件組態。

```
{
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "hanadb_server_startup_time_variations_seconds",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_level_5_alerts_count",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_level_4_alerts_count",
          "monitor": true
        },
        {
          "alarmMetricName": "hanadb_out_of_memory_events_count",
          "monitor": true
        },
        {
          "alarmMetricName": "ha_cluster_pacemaker_stonith_enabled",
          "monitor": true
        }
      ],
      "logs": [
        {
          "logGroupName": "SAP_HANA_TRACE-my-resourge-group",
          "logPath": "/usr/sap/HDB/HDB00/*/trace/*.trc",
          "logType": "SAP_HANA_TRACE",
          "monitor": true,
```

```
          "encoding": "utf-8"
        },
        {
          "logGroupName": "SAP_HANA_HIGH_AVAILABILITY-my-resource-group",
          "logPath": "/var/log/pacemaker/pacemaker.log",
          "logType": "SAP_HANA_HIGH_AVAILABILITY",
          "monitor": true,
          "encoding": "utf-8"
        }
      ]
    }
  ],
  "hanaPrometheusExporter": {
    "hanaSid": "HDB",
    "hanaPort": "30013",
    "hanaSecretName": "HANA_DB_CREDS",
    "prometheusPort": "9668"
  },
  "haClusterPrometheusExporter": {
    "prometheusPort": "9664"
  }
}
```

Amazon EC2 NetWeaver 上的 SAP

下列範例顯示了在 Amazon EC2 上採用 JSON 格式 NetWeaver 的 SAP 元件組態。

```
{
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "CPUUtilization",
          "monitor": true
        },
        {
          "alarmMetricName": "StatusCheckFailed",
          "monitor": true
        },
        {
          "alarmMetricName": "disk_used_percent",
          "monitor": true
        },
```

```
        {
          "alarmMetricName": "mem_used_percent",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_ResponseTime",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_ResponseTimeDialog",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_ResponseTimeDialogRFC",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_DBRequestTime",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_LongRunners",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_AbortedJobs",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_BasisSystem",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_Database",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_Security",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_System",
          "monitor": true
        },
```

```
        {
          "alarmMetricName": "sap_alerts_QueueTime",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_Availability",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_start_service_processes",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_dispatcher_queue_now",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_dispatcher_queue_max",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_enqueue_server_locks_max",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_enqueue_server_locks_now",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_enqueue_server_locks_state",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_enqueue_server_replication_state",
          "monitor": true
        }
      ],
      "logs": [
        {
          "logGroupName": "SAP_NETWEAVER_DEV_TRACE_LOGS-NetWeaver-ML4",
          "logPath": "/usr/sap/ML4/*/work/dev_w*",
          "logType": "SAP_NETWEAVER_DEV_TRACE_LOGS",
          "monitor": true,
          "encoding": "utf-8"
```

```
        }
      ]
    }
  ],
  "netWeaverPrometheusExporter": {
    "sapSid": "ML4",
    "instanceNumbers": [
      "00",
      "11"
    ],
    "prometheusPort": "9680"
  }
}
```

Amazon EC2 上的 SAP NetWeaver 高可用性

下列範例顯示了在 Amazon EC2 上採用 JSON 格式的 SAP NetWeaver 高可用性的元件組態。

```
{
  "subComponents": [
    {
      "subComponentType": "AWS::EC2::Instance",
      "alarmMetrics": [
        {
          "alarmMetricName": "ha_cluster_corosync_ring_errors",
          "monitor": true
        },
        {
          "alarmMetricName": "ha_cluster_pacemaker_fail_count",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_HA_check_failover_config_state",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_HA_get_failover_config_HAActive",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_AbortedJobs",
          "monitor": true
        },
        {
```

```
          "alarmMetricName": "sap_alerts_Availability",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_BasisSystem",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_DBRequestTime",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_Database",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_FrontendResponseTime",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_LongRunners",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_QueueTime",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_ResponseTime",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_ResponseTimeDialog",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_ResponseTimeDialogRFC",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_Security",
          "monitor": true
        },
        {
```

```
          "alarmMetricName": "sap_alerts_Shortdumps",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_SqlError",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_alerts_System",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_enqueue_server_replication_state",
          "monitor": true
        },
        {
          "alarmMetricName": "sap_start_service_processes",
          "monitor": true
        }
      ],
      "logs": [
        {
          "logGroupName": "SAP_NETWEAVER_DEV_TRACE_LOGS-NetWeaver-PR1",
          "logPath": "/usr/sap/<SID>/D*/work/dev_w*",
          "logType": "SAP_NETWEAVER_DEV_TRACE_LOGS",
          "monitor": true,
          "encoding": "utf-8"
        }
      ]
    }
  ],
  "haClusterPrometheusExporter": {
    "prometheusPort": "9664"
  },
  "netWeaverPrometheusExporter": {
    "sapSid": "PR1",
    "instanceNumbers": [
      "11",
      "12"
    ],
    "prometheusPort": "9680"
  }
}
```

SQL Always On 可用性群組

下列範例顯示 SQL Always On 可用性群組 JSON 格式的元件組態。

```
{
  "subComponents" : [ {
    "subComponentType" : "AWS::EC2::Instance",
    "alarmMetrics" : [ {
      "alarmMetricName" : "CPUUtilization",
      "monitor" : true
    }, {
      "alarmMetricName" : "StatusCheckFailed",
      "monitor" : true
    }, {
      "alarmMetricName" : "Processor % Processor Time",
      "monitor" : true
    }, {
      "alarmMetricName" : "Memory % Committed Bytes In Use",
      "monitor" : true
    }, {
      "alarmMetricName" : "Memory Available Mbytes",
      "monitor" : true
    }, {
      "alarmMetricName" : "Paging File % Usage",
      "monitor" : true
    }, {
      "alarmMetricName" : "System Processor Queue Length",
      "monitor" : true
    }, {
      "alarmMetricName" : "Network Interface Bytes Total/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "PhysicalDisk % Disk Time",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Buffer Manager Buffer cache hit ratio",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Buffer Manager Page life expectancy",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:General Statistics Processes blocked",
      "monitor" : true
    }, {
```

```
      "alarmMetricName" : "SQLServer:General Statistics User Connections",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Locks Number of Deadlocks/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:SQL Statistics Batch Requests/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica File Bytes Received/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Log Bytes Received/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Log remaining for undo",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Log Send Queue",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Mirrored Write Transaction/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Recovery Queue",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Redo Bytes Remaining",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Redone Bytes/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Total Log requiring undo",
      "monitor" : true
    }, {
      "alarmMetricName" : "SQLServer:Database Replica Transaction Delay",
      "monitor" : true
    } ],
    "windowsEvents" : [ {
      "logGroupName" : "WINDOWS_EVENTS-Application-<RESOURCE_GROUP_NAME>",
      "eventName" : "Application",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL", "INFORMATION" ],
      "monitor" : true
```

```
    }, {
      "logGroupName" : "WINDOWS_EVENTS-System-<RESOURCE_GROUP_NAME>",
      "eventName" : "System",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL" ],
      "monitor" : true
    }, {
      "logGroupName" : "WINDOWS_EVENTS-Security-<RESOURCE_GROUP_NAME>",
      "eventName" : "Security",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL" ],
      "monitor" : true
    } ],
    "logs" : [ {
      "logGroupName" : "SQL_SERVER_ALWAYSON_AVAILABILITY_GROUP-<RESOURCE_GROUP_NAME>",
      "logPath" : "C:\\Program Files\\Microsoft SQL Server\\MSSQL**.MSSQLSERVER\\MSSQL\
\Log\\ERRORLOG",
      "logType" : "SQL_SERVER",
      "monitor" : true,
      "encoding" : "utf-8"
    } ]
  }, {
    "subComponentType" : "AWS::EC2::Volume",
    "alarmMetrics" : [ {
      "alarmMetricName" : "VolumeReadBytes",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeWriteBytes",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeReadOps",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeWriteOps",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeQueueLength",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeThroughputPercentage",
      "monitor" : true
    }, {
    "alarmMetricName" : "BurstBalance",
      "monitor" : true
    } ]
  } ]
```

```
}
```

SQL 容錯移轉叢集執行個體

下列範例顯示 SQL 容錯移轉叢集執行個體 JSON 格式的元件組態。

```
{
  "subComponents" : [ {
    "subComponentType" : "AWS::EC2::Instance",
    "alarmMetrics" : [ {
      "alarmMetricName" : "CPUUtilization",
      "monitor" : true
    }, {
      "alarmMetricName" : "StatusCheckFailed",
      "monitor" : true
    }, {
      "alarmMetricName" : "Processor % Processor Time",
      "monitor" : true
    }, {
      "alarmMetricName" : "Memory % Committed Bytes In Use",
      "monitor" : true
    }, {
      "alarmMetricName" : "Memory Available Mbytes",
      "monitor" : true
    }, {
      "alarmMetricName" : "Paging File % Usage",
      "monitor" : true
    }, {
      "alarmMetricName" : "System Processor Queue Length",
      "monitor" : true
    }, {
      "alarmMetricName" : "Network Interface Bytes Total/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "PhysicalDisk % Disk Time",
      "monitor" : true
    }, {
      "alarmMetricName" : "Bytes Received/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Normal Messages Queue Length/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Urgent Message Queue Length/se",
```

```
      "monitor" : true
    }, {
      "alarmMetricName" : "Reconnect Count",
      "monitor" : true
    }, {
      "alarmMetricName" : "Unacknowledged Message Queue Length/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Messages Outstanding",
      "monitor" : true
    }, {
      "alarmMetricName" : "Messages Sent/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Database Update Messages/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Update Messages/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Flushes/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Crypto Checkpoints Saved/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Crypto Checkpoints Restored/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Registry Checkpoints Restored/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Registry Checkpoints Saved/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Cluster API Calls/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Resource API Calls/sec",
      "monitor" : true
    }, {
      "alarmMetricName" : "Cluster Handles/sec",
      "monitor" : true
    }, {
```

```
      "alarmMetricName" : "Resource Handles/sec",
      "monitor" : true
    } ],
    "windowsEvents" : [ {
      "logGroupName" : "WINDOWS_EVENTS-Application-<RESOURCE_GROUP_NAME>",
      "eventName" : "Application",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL"],
      "monitor" : true
    }, {
      "logGroupName" : "WINDOWS_EVENTS-System-<RESOURCE_GROUP_NAME>",
      "eventName" : "System",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL", "INFORMATION" ],
      "monitor" : true
    }, {
      "logGroupName" : "WINDOWS_EVENTS-Security-<RESOURCE_GROUP_NAME>",
      "eventName" : "Security",
      "eventLevels" : [ "WARNING", "ERROR", "CRITICAL" ],
      "monitor" : true
    } ],
    "logs" : [ {
      "logGroupName" : "SQL_SERVER_FAILOVER_CLUSTER_INSTANCE-<RESOURCE_GROUP_NAME>",
      "logPath" : "\\\\amznfsxjmzbykwn.mydomain.aws\\SQLDB\\MSSQL**.MSSQLSERVER\\MSSQL\
\Log\\ERRORLOG",
      "logType" : "SQL_SERVER",
      "monitor" : true,
      "encoding" : "utf-8"
    } ]
  }, {
    "subComponentType" : "AWS::EC2::Volume",
    "alarmMetrics" : [ {
      "alarmMetricName" : "VolumeReadBytes",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeWriteBytes",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeReadOps",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeWriteOps",
      "monitor" : true
    }, {
    "alarmMetricName" : "VolumeQueueLength",
      "monitor" : true
```

```
    }, {
    "alarmMetricName" : "VolumeThroughputPercentage",
      "monitor" : true
    }, {
    "alarmMetricName" : "BurstBalance",
      "monitor" : true
    } ]
  } ]
}
```

使用 CloudFormation 範本建立和設定 CloudWatch 應用程式見解監控

您可以直接從 AWS CloudFormation 範本將「應用程式見解」監視 (包括關鍵指標和遙測) 新增至應用程式、資料庫和 Web 伺服器。

本節提供 JSON 和 YAML 格式的 AWS CloudFormation 範例範本，可協助您建立和設定應用程式深入解析監視。

若要檢視「使用者指南」中的「應AWS CloudFormation 用程式見解」資源和屬性參考，請參閱ApplicationInsights 資源類型參考。

範例範本

- 為整個 AWS CloudFormation 堆疊建立應用程式洞察應用程式
- 建立具有詳細設定的 Application Insights 應用程式
- 使用 CUSTOM 模式元件組態建立 Application Insights 應用程式
- 使用 DEFAULT 模式元件組態建立 Application Insights 應用程式
- 使用 DEFAULT_WITH_OVERWRITE 模式元件組態建立 Application Insights 應用程式

為整個 AWS CloudFormation 堆疊建立應用程式洞察應用程式

若要套用下列範本，您必須建立 AWS 資源和一或多個資源群組，以便從中建立 Application Insights 應用程式以監視這些資源。如需詳細資訊，請參閱《 AWS Resource Groups 入門》。

下列範本的前兩個部分會指定資源和資源群組。範本的最後一部分會為資源群組建立 Application Insights 應用程式，但不會設定應用程式或套用監控。如需詳細資訊，請參閱 Amazon CloudWatch 應用程式深入解析 API 參考中的CreateApplication命令詳細資訊。

JSON 格式的範本

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Test Resource Group stack",
    "Resources": {
        "EC2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId" : "ami-abcd1234efgh5678i",
                "SecurityGroupIds" : ["sg-abcd1234"]
            }
        },
        ...
        "ResourceGroup": {
            "Type": "AWS::ResourceGroups::Group",
            "Properties": {
                "Name": "my_resource_group"
            }
        },
        "AppInsightsApp": {
            "Type": "AWS::ApplicationInsights::Application",
            "Properties": {
                "ResourceGroupName": "my_resource_group"
            },
            "DependsOn" : "ResourceGroup"
        }
    }
}
```

YAML 格式的範本

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: Test Resource Group stack
Resources:
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-abcd1234efgh5678i
      SecurityGroupIds:
      - sg-abcd1234
  ...
  ResourceGroup:
    Type: AWS::ResourceGroups::Group
```

```
    Properties:
      Name: my_resource_group
  AppInsightsApp:
    Type: AWS::ApplicationInsights::Application
    Properties:
      ResourceGroupName: my_resource_group
    DependsOn: ResourceGroup
```

下列範本區段會將預設監控組態套用至 Application Insights 應用程式。如需詳細資訊，請參閱 Amazon CloudWatch 應用程式深入解析 API 參考中的CreateApplication命令詳細資訊。

在將 AutoConfigurationEnabled 設定為 true 時，則會使用該 DEFAULT 應用程式層的建議監控設定值來設定應用程式的所有元件。如需這些設定和層級的詳細資訊，請參閱 Amazon CloudWatch 應用程式洞察 API 參考UpdateComponentConfiguration中的DescribeComponentConfigurationRecommendation和。

JSON 格式的範本

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Test Application Insights Application stack",
    "Resources": {
        "AppInsightsApp": {
            "Type": "AWS::ApplicationInsights::Application",
            "Properties": {
                "ResourceGroupName": "my_resource_group",
                "AutoConfigurationEnabled": true
            }
        }
    }
}
```

YAML 格式的範本

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: Test Application Insights Application stack
Resources:
  AppInsightsApp:
    Type: AWS::ApplicationInsights::Application
    Properties:
      ResourceGroupName: my_resource_group
```

```
      AutoConfigurationEnabled: true
```

建立具有詳細設定的 Application Insights 應用程式

下列範本會執行這些動作：

- 建立具有 CloudWatch 事件通知並 OpsCenter啟用的應用程式見解應用程式。如需詳細資訊，請參閱 Amazon CloudWatch 應用程式深入解析 API 參考中的CreateApplication命令詳細資訊。
- 使用兩個標籤來標記應用程式，其中一個標籤沒有標籤值。如需詳細資訊，請參閱 Amazon CloudWatch 應用程式洞察 API 參考TagResource中的。
- 建立兩個自訂執行個體群組元件。如需詳細資訊，請參閱 Amazon CloudWatch 應用程式洞察 API 參考CreateComponent中的。
- 建立兩個日誌模式集。如需詳細資訊，請參閱 Amazon CloudWatch 應用程式洞察 API 參考CreateLogPattern中的。
- 設定 `AutoConfigurationEnabled` 為 `true`，此設定應用程式的所有元件，並使用該 `DEFAULT` 層的建議監控設定值。如需詳細資訊，請參閱 Amazon CloudWatch 應用程式洞察 API 參考DescribeComponentConfigurationRecommendation中的。

JSON 格式的範本

```
{
    "Type": "AWS::ApplicationInsights::Application",
    "Properties": {
        "ResourceGroupName": "my_resource_group",
        "CWEMonitorEnabled": true,
        "OpsCenterEnabled": true,
        "OpsItemSNSTopicArn": "arn:aws:sns:us-east-1:123456789012:my_topic",
        "AutoConfigurationEnabled": true,
        "Tags": [
            {
                "Key": "key1",
                "Value": "value1"
            },
            {
                "Key": "key2",
                "Value": ""
            }
        ],
        "CustomComponents": [
            {
```

```
                "ComponentName": "test_component_1",
                "ResourceList": [
                    "arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1234efgh5678i"
                ]
            },
            {
                "ComponentName": "test_component_2",
                "ResourceList": [
                    "arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1234efgh5678i",
                    "arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1234efgh5678i"
                ]
            }
        ],
        "LogPatternSets": [
            {
                "PatternSetName": "pattern_set_1",
                "LogPatterns": [
                    {
                        "PatternName": "deadlock_pattern",
                        "Pattern": ".*\\sDeadlocked\\sSchedulers(([^\\w].*)|($))",
                        "Rank": 1
                    }
                ]
            },
            {
                "PatternSetName": "pattern_set_2",
                "LogPatterns": [
                    {
                        "PatternName": "error_pattern",
                        "Pattern": ".*[\\s\\[]ERROR[\\s\\]].*",
                        "Rank": 1
                    },
                    {
                        "PatternName": "warning_pattern",
                        "Pattern": ".*[\\s\\[]WARN(ING)?[\\s\\]].*",
                        "Rank": 10
                    }
                ]
            }
        ]
    }
}
```

YAML 格式的範本

```
---
Type: AWS::ApplicationInsights::Application
Properties:
  ResourceGroupName: my_resource_group
  CWEMonitorEnabled: true
  OpsCenterEnabled: true
  OpsItemSNSTopicArn: arn:aws:sns:us-east-1:123456789012:my_topic
  AutoConfigurationEnabled: true
  Tags:
  - Key: key1
    Value: value1
  - Key: key2
    Value: ''
  CustomComponents:
  - ComponentName: test_component_1
    ResourceList:
    - arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1234efgh5678i
  - ComponentName: test_component_2
    ResourceList:
    - arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1234efgh5678i
    - arn:aws:ec2:us-east-1:123456789012:instance/i-abcd1234efgh5678i
  LogPatternSets:
  - PatternSetName: pattern_set_1
    LogPatterns:
    - PatternName: deadlock_pattern
      Pattern: ".*\\sDeadlocked\\sSchedulers(([^\\w].*)|($))"
      Rank: 1
  - PatternSetName: pattern_set_2
    LogPatterns:
    - PatternName: error_pattern
      Pattern: ".*[\\s\\[]ERROR[\\s\\]].*"
      Rank: 1
    - PatternName: warning_pattern
      Pattern: ".*[\\s\\[]WARN(ING)?[\\s\\]].*"
      Rank: 10
```

使用 **CUSTOM** 模式元件組態建立 Application Insights 應用程式

下列範本會執行這些動作：

- 建立 Application Insights 應用程式。如需詳細資訊，請參閱 Amazon CloudWatch 應用程式洞察 API 參考CreateApplication中的。
- 元件 my_component 會將 ComponentConfigurationMode 設定為 CUSTOM，這會導致此元件按照 CustomComponentConfiguration 中指定的組態進行設定。如需詳細資訊，請參閱 Amazon CloudWatch 應用程式洞察 API 參考UpdateComponentConfiguration中的。

JSON 格式的範本

```
{
    "Type": "AWS::ApplicationInsights::Application",
    "Properties": {
        "ResourceGroupName": "my_resource_group,
        "ComponentMonitoringSettings": [
            {
                "ComponentARN": "my_component",
                "Tier": "SQL_SERVER",
                "ComponentConfigurationMode": "CUSTOM",
                "CustomComponentConfiguration": {
                    "ConfigurationDetails": {
                        "AlarmMetrics": [
                            {
                                "AlarmMetricName": "StatusCheckFailed"
                            },
                            ...
                        ],
                        "Logs": [
                            {
                                "LogGroupName": "my_log_group_1",
                                "LogPath": "C:\\LogFolder_1\\*",
                                "LogType": "DOT_NET_CORE",
                                "Encoding": "utf-8",
                                "PatternSet": "my_pattern_set_1"
                            },
                            ...
                        ],
                        "WindowsEvents": [
                            {
                                "LogGroupName": "my_windows_event_log_group_1",
                                "EventName": "Application",
                                "EventLevels": [
                                    "ERROR",
                                    "WARNING",
```

```
                        ...
                    ],
                    "Encoding": "utf-8",
                    "PatternSet": "my_pattern_set_2"
                },
                ...
            ],
            "Alarms": [
                {
                    "AlarmName": "my_alarm_name",
                    "Severity": "HIGH"
                },
                ...
            ]
        },
        "SubComponentTypeConfigurations": [
            {
                "SubComponentType": "EC2_INSTANCE",
                "SubComponentConfigurationDetails": {
                    "AlarmMetrics": [
                        {
                            "AlarmMetricName": "DiskReadOps"
                        },
                        ...
                    ],
                    "Logs": [
                        {
                            "LogGroupName": "my_log_group_2",
                            "LogPath": "C:\\LogFolder_2\\*",
                            "LogType": "IIS",
                            "Encoding": "utf-8",
                            "PatternSet": "my_pattern_set_3"
                        },
                        ...
                    ],
                    "processes" : [
                        {
                            "processName" : "my_process",
                            "alarmMetrics" : [
                        {
                            "alarmMetricName" : "procstat cpu_usage",
                            "monitor" : true
                        }, {
                            "alarmMetricName" : "procstat memory_rss",
```

```
                                    "monitor" : true
                                }
                            ]
                        }
                    ],
                            "WindowsEvents": [
                                {
                                    "LogGroupName": "my_windows_event_log_group_2",
                                    "EventName": "Application",
                                    "EventLevels": [
                                        "ERROR",
                                        "WARNING",
                                        ...
                                    ],
                                    "Encoding": "utf-8",
                                    "PatternSet": "my_pattern_set_4"
                                },
                                ...
                            ]
                        }
                    }
                ]
            }
        }
    ]
  }
}
```

YAML 格式的範本

```
---
Type: AWS::ApplicationInsights::Application
Properties:
  ResourceGroupName: my_resource_group
  ComponentMonitoringSettings:
  - ComponentARN: my_component
    Tier: SQL_SERVER
    ComponentConfigurationMode: CUSTOM
    CustomComponentConfiguration:
      ConfigurationDetails:
        AlarmMetrics:
        - AlarmMetricName: StatusCheckFailed
        ...
```

```
        Logs:
        - LogGroupName: my_log_group_1
          LogPath: C:\LogFolder_1\*
          LogType: DOT_NET_CORE
          Encoding: utf-8
          PatternSet: my_pattern_set_1
        ...
        WindowsEvents:
        - LogGroupName: my_windows_event_log_group_1
          EventName: Application
          EventLevels:
          - ERROR
          - WARNING
          ...
          Encoding: utf-8
          PatternSet: my_pattern_set_2
        ...
        Alarms:
        - AlarmName: my_alarm_name
          Severity: HIGH
        ...
      SubComponentTypeConfigurations:
      - SubComponentType: EC2_INSTANCE
        SubComponentConfigurationDetails:
          AlarmMetrics:
          - AlarmMetricName: DiskReadOps
          ...
          Logs:
          - LogGroupName: my_log_group_2
            LogPath: C:\LogFolder_2\*
            LogType: IIS
            Encoding: utf-8
            PatternSet: my_pattern_set_3
          ...
          Processes:
          - ProcessName: my_process
            AlarmMetrics:
            - AlarmMetricName: procstat cpu_usage
              ...
            ...
          WindowsEvents:
          - LogGroupName: my_windows_event_log_group_2
            EventName: Application
            EventLevels:
```

```
            - ERROR
            - WARNING
            ...
            Encoding: utf-8
            PatternSet: my_pattern_set_4
          ...
```

使用 **DEFAULT** 模式元件組態建立 Application Insights 應用程式

下列範本會執行這些動作：

- 建立 Application Insights 應用程式。如需詳細資訊，請參閱 Amazon CloudWatch 應用程式洞察 API 參考CreateApplication中的。
- 元件 `my_component` 會將 `ComponentConfigurationMode` 設定為 `DEFAULT` 並將 `Tier` 設定為 `SQL_SERVER`，這會使用 Application Insights 建議該 `SQL_Server` 層的組態設定來設定此元件。如需詳細資訊，請參閱 Amazon CloudWatch 應用程式洞察 API 參考UpdateComponentConfiguration中的DescribeComponentConfiguration和。

JSON 格式的範本

```
{
    "Type": "AWS::ApplicationInsights::Application",
    "Properties": {
        "ResourceGroupName": "my_resource_group",
        "ComponentMonitoringSettings": [
            {
                "ComponentARN": "my_component",
                "Tier": "SQL_SERVER",
                "ComponentConfigurationMode": "DEFAULT"
            }
        ]
    }
}
```

YAML 格式的範本

```
---
Type: AWS::ApplicationInsights::Application
Properties:
  ResourceGroupName: my_resource_group
  ComponentMonitoringSettings:
```

```
  - ComponentARN: my_component
    Tier: SQL_SERVER
    ComponentConfigurationMode: DEFAULT
```

使用 **DEFAULT_WITH_OVERWRITE** 模式元件組態建立 Application Insights 應用程式

下列範本會執行這些動作：

- 建立 Application Insights 應用程式。如需詳細資訊，請參閱 Amazon CloudWatch 應用程式洞察 API 參考CreateApplication中的。
- 元件 my_component 會將 ComponentConfigurationMode 設定為 DEFAULT_WITH_OVERWRITE 並將 tier 設定為 DOT_NET_CORE，這會使用 Application Insights 建議該 DOT_NET_CORE 層的組態設定來設定此元件。覆寫的組態設定可在 DefaultOverwriteComponentConfiguration 中的下列項目中指定：
  - 在元件層級，系統會覆寫 AlarmMetrics 設定。
  - 在子元件層級，對於 EC2_Instance 類型的子元件，系統會覆寫 Logs 設定。

  如需詳細資訊，請參閱 Amazon CloudWatch 應用程式洞察 API 參考UpdateComponentConfiguration中的。

JSON 格式的範本

```
{
    "Type": "AWS::ApplicationInsights::Application",
    "Properties": {
        "ResourceGroupName": "my_resource_group",
        "ComponentMonitoringSettings": [
            {
                "ComponentName": "my_component",
                "Tier": "DOT_NET_CORE",
                "ComponentConfigurationMode": "DEFAULT_WITH_OVERWRITE",
                "DefaultOverwriteComponentConfiguration": {
                    "ConfigurationDetails": {
                        "AlarmMetrics": [
                            {
                                "AlarmMetricName": "StatusCheckFailed"
                            }
                        ]
                    },
                    "SubComponentTypeConfigurations": [
```

```
                        {
                            "SubComponentType": "EC2_INSTANCE",
                            "SubComponentConfigurationDetails": {
                                "Logs": [
                                    {
                                        "LogGroupName": "my_log_group",
                                        "LogPath": "C:\\LogFolder\\*",
                                        "LogType": "IIS",
                                        "Encoding": "utf-8",
                                        "PatternSet": "my_pattern_set"
                                    }
                                ]
                            }
                        }
                    ]
                }
            }
        ]
    }
}
```

YAML 格式的範本

```
---
Type: AWS::ApplicationInsights::Application
Properties:
  ResourceGroupName: my_resource_group
  ComponentMonitoringSettings:
  - ComponentName: my_component
    Tier: DOT_NET_CORE
    ComponentConfigurationMode: DEFAULT_WITH_OVERWRITE
    DefaultOverwriteComponentConfiguration:
      ConfigurationDetails:
        AlarmMetrics:
        - AlarmMetricName: StatusCheckFailed
      SubComponentTypeConfigurations:
      - SubComponentType: EC2_INSTANCE
        SubComponentConfigurationDetails:
          Logs:
          - LogGroupName: my_log_group
            LogPath: C:\LogFolder\*
            LogType: IIS
            Encoding: utf-8
```

```
            PatternSet: my_pattern_set
```

教學課程：設定 SAP ASE 的監控

本教學課程示範如何設定 CloudWatch 應用程式深入解析，以設定 SAP ASE 資料庫的監視。您可以使用「 CloudWatch 應用程式深入解析」自動儀表板，以視覺化方式呈現問題詳細資料、加速疑難排解，並促進 SAP ASE 資料庫的平均解決時間 (MTTR)。

SAP ASE 主題的 Application Insights

- 支援的環境
- 支援的作業系統
- 功能
- 必要條件
- 在您的 SAP ASE 資料庫上設定監控
- 管理 SAP ASE 資料庫的監控
- 設定警示閾值
- 檢視與針對 Application Insights 偵測到的 SAP ASE 問題進行疑難排解
- SAP ASE 的 Application Insights 疑難排解

支援的環境

CloudWatch 應用程式深入解析支援下列系統和模式的 AWS 資源部署。您提供並安裝 SAP ASE 資料庫軟體和支援的 SAP 應用程式軟體。

- 單一 Amazon EC2 執行個體上的一個或多個 SAP ASE 資料庫 – 單一節點、縱向擴展架構中的 SAP ASE。
- 跨可用區域 SAP ASE 資料庫高可用性設定 – 使用 SUSE/RHEL 叢集跨兩個可用區域設定高可用性的 SAP ASE。

> **Note**
>
> CloudWatch 「應用程式深入解析」僅支援單一 SAP 系統識別碼 (SID) ASE HA 環境。如果連接了多個 ASE HA SID，則僅為偵測到的第一個 SID 設定監控。

支援的作業系統

CloudWatch SAP ASE 的應用程式深入解析可在下列作業系統上支援 x86-64 架構：

- SuSE Linux 12 SP4
- SuSE Linux 12 SP5
- SuSE Linux 15
- SuSE Linux 15 SP1
- SuSE Linux 15 SP2
- SuSE Linux 15 SP3
- SuSE Linux 15 SP4
- SuSE Linux 15 SP1 For SAP
- SuSE Linux 15 SP2 For SAP
- SuSE Linux 15 SP3 For SAP
- SuSE Linux 15 SP4 For SAP
- SuSE Linux 12 SP4 For SAP
- SuSE Linux 12 SP5 For SAP
- RedHat Linux 7.6
- RedHat Linux 7.7
- RedHat Linux 7.9
- RedHat Linux 8.1
- RedHat Linux 8.4
- RedHat Linux 8.6

功能

CloudWatch SAP ASE 的應用程式深入解析提供下列功能：

- 自動偵測 SAP ASE 工作負載
- 根據靜態閾值自動建立 SAP ASE 警示
- 根據異常偵測自動建立 SAP ASE 警示
- 自動 SAP ASE 日誌模式辨識
- SAP ASE 的運作狀態儀表板

- SAP ASE 的問題儀表板

必要條件

您必須執行下列先決條件，才能設定具有「 CloudWatch 應用程式見解」的 SAP ASE 資料庫：

- SAP ASE 組態參數 – 必須在 ASE 資料庫上啟用下列組態參數："enable monitoring"、"sql text pipe max messages"、"sql text pipe active"。這可讓 CloudWatch 應用程式深入解析為您的資料庫提供完整的監控功能。如果您的 ASE 資料庫上未啟用這些設定，Application Insights 會自動讓其收集必要的指標以允許監控。
- SAP ASE 資料庫使用者 – 在 Application Insights 佈設期間提供的資料庫使用者必須具有存取下列項目的許可：
  - 主資料庫和使用者 (租用戶) 資料庫中的系統資料表
  - 監控資料表
- SAP HostCtrl — HostCtrl 在您的亞馬遜 EC2 執行個體上安裝和設定 SAP。
- Amazon CloudWatch 代理程式 — 確保您沒有在 Amazon EC2 執行個體上執行預先存在的 CloudWatch 代理程式。如果您已安裝 CloudWatch 代理程式，請務必從現有的 CloudWatch 代理程式組態檔案中移除您在 Ap CloudWatch plication Insights 中使用的資源組態，以避免合併衝突。如需詳細資訊，請參閱 手動建立或編輯 CloudWatch 代理程式組態檔。
- AWS 啟用 Systems Manager — 在您的執行個體上安裝 SSM 代理程式，並啟用已啟用 SSM 的執行個體。如需有關如何安裝 SSM Agent 的資訊，請參閱《AWS Systems Manager 使用者指南》中的使用 SSM Agent。
- Amazon EC2 執行個體角色 – 您必須連接下列 Amazon EC2 執行個體角色才能設定資料庫。
  - 您必須連接 `AmazonSSMManagedInstanceCore` 角色，以啟用 Systems Manager。如需詳細資訊，請參閱 AWS Systems Manager 以身分為基礎的政策範例。
  - 您必須附加，`CloudWatchAgentServerPolicy`才能啟用透過 CloudWatch發出的執行個體指標和記錄檔。如需詳細資訊，請參閱建立 IAM 角色和使用者以搭配 Amazon CloudWatch 代理程式使用。
  - 您必須將下列 IAM 內嵌政策連接到 Amazon EC2 執行個體角色，才能讀取存放在 AWS Secrets Manager中的密碼。如需內嵌政策的詳細資訊，請參閱《AWS Identity and Access Management 使用者指南》中的內嵌政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:ApplicationInsights-*"
        }
    ]
}
```

- AWS Resource Groups— 您必須建立一個資源群組，其中包含應用程式堆疊使用的所有相關 AWS 資源，以便將應用程式上載至「應用程式深入解析」。 CloudWatch 這包括執行 SAP ASE 資料庫的 Amazon EC2 執行個體和 Amazon EBS 磁碟區。如果每個帳戶有多個資料庫，建議您建立一個資源群組，其中包含每個 SAP ASE AWS 資料庫系統的資源。
- IAM 許可 - 對於非管理員使用者：
  - 您必須建立 AWS Identity and Access Management (IAM) 政策，讓應用程式深入解析建立服務連結角色，並將其附加至您的使用者身分識別。如需連接政策的步驟，請參閱 IAM 政策。
  - 使用者必須擁有在中建立密碼的權限， AWS Secrets Manager 才能儲存資料庫使用者認證。如需詳細資訊，請參閱範例：建立機密的許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:CreateSecret"
      ],
      "Resource": "arn:aws:secretsmanager:*:*:secret:ApplicationInsights-*"
    }
  ]
}
```

- 服務連結角色 — 應用程式深入解析使用 AWS Identity and Access Management (IAM) 服務連結角色。系統會在您於 Application Insights 主控台建立第一個 Application Insights 應用程式時，為您建立服務連結角色。如需詳細資訊，請參閱 將服務連結角色用於 CloudWatch 應用程式深入。

在您的 SAP ASE 資料庫上設定監控

使用下列步驟設定 SAP ASE 資料庫的監控

1. 開啟 CloudWatch 主控台。
2. 從左側導覽窗格中，選擇 Insights 下的 Application Insights。
3. Application Insights 頁面會顯示 Application Insights 監控的應用程式清單，以及每個應用程式的監控狀態。在右上角，選擇 Add an application (新增應用程式)。
4. 在指定應用程式詳細資訊頁面上，從資源群組的下拉清單中選取 AWS 資源群組，其中包含您的 SAP ASE 資料庫資源。如果尚未建立應用程式的資源群組，您可以透過選擇 Resource group (資源群組) 下拉單中的 Create new resource group (建立新的資源群組) 建立一個。如需建立資源群組的詳細資訊，請參閱《AWS Resource Groups 使用者指南》。
5. 在監控 CloudWatch 事件下，選取核取方塊，將應用程式洞察監控與 CloudWatch 事件整合，以取得來自 Amazon EBS、Amazon EC2 AWS CodeDeploy、Amazon ECS、 AWS Health API 和通知、Amazon RDS、Amazon S3 和的見解。 AWS Step Functions
6. 在 [整合對象] 下 AWS Systems Manager OpsCenter，選取 [產生以 AWS Systems Manager OpsCenter OpsItems 供修正動作] 旁邊的核取方塊，以檢視並在針對所選應用程式偵測到問題時取得通知。若要追蹤為解決與 AWS 資源相關的作業工作項目 (稱為 OpsItems) 而執行的作業，請提供 SNS 主題 ARN。
7. 您可以選擇性地輸入標籤，以協助您識別和組織資源。 CloudWatch 應用程式深入解析同時支援以標籤 AWS CloudFormation 為基礎和堆疊式資源群組，但群組除外。 Application Auto Scaling 如需詳細資訊，請參閱 AWS Resource Groups 和標籤使用者指南中的標籤編輯器。
8. 選擇 Next (下一個) 繼續設定監控。
9. 在 [檢閱偵測到的元件] 頁面上，會列出 CloudWatch 應用程式深入解析自動偵測的受監控元件及其工作負載。

> **Note**
>
> 包含偵測到的 SAP ASE 高可用性工作負載的元件，一個元件上僅支援一個工作負載。包含偵測到的 SAP ASE 單一節點工作負載的元件，支援多個工作負載，但您無法新增或移除工作負載。所有自動偵測到的工作負載將會受到監控。

10. 選擇下一步。
11. 在指定元件詳細資訊頁面上，輸入 SAP ASE 資料庫的使用者名稱和密碼。
12. 檢閱您的應用程式監控組態，然後選擇 Submit (提交)。

13. 應用程式詳細資訊頁面隨即開啟，您可以在其中檢視應用程式摘要、受監控元件和工作負載的清單，以及未受監控元件和工作負載。如果選取元件或工作負載旁邊的選項按鈕，則您也可以檢視組態歷史記錄、日誌模式，以及任何已建立的標籤。當您提交組態時，您的帳戶會為 SAP ASE 系統部署所有指標和警示，這可能需要最多 2 小時。

管理 SAP ASE 資料庫的監控

您可以執行下列步驟來管理 SAP ASE 資料庫的使用者憑證、指標和日誌路徑：

1. 開啟 CloudWatch 主控台。
2. 從左側導覽窗格中，選擇 Insights 下的 Application Insights。
3. Application Insights 頁面會顯示 Application Insights 監控的應用程式清單，以及每個應用程式的監控狀態。
4. 在 Monitored components (受監控元件) 下，選取元件名稱旁的選項按鈕。然後，選擇 Manage monitoring (管理監控)。
5. 在 EC2 instance group logs (EC2 執行個體群組日誌) 下，您可以更新現有的日誌路徑、日誌模式集和記錄群組名稱。此外，您可以新增最多三個額外 Application logs (應用程式日誌)。
6. 在指標下，您可以根據需求選擇 SAP ASE 指標。SAP ASE 指標名稱的字首為 `asedb`。每個元件最多可新增 60 個指標。
7. 在 ASE 組態，輸入 SAP ASE 資料庫的使用者名稱和密碼。這是 Amazon CloudWatch 代理程式用來連線至 SAP ASE 資料庫的使用者名稱和密碼。
8. 在 [自訂警示] 底下，您可以新增其他警示，以供 CloudWatch 應用程式深入解析監控。
9. 檢閱您的應用程式監控組態，然後選擇 Submit (提交)。當您提交組態時，您的帳戶會為 SAP HANA 系統更新所有指標和警示，這可能需要最多 2 小時。

設定警示閾值

CloudWatch 應用程式深入解析會自動建立 Amazon CloudWatch 指標以供監視的警示，以及該指標的閾值。當指標超過閾值達到指定的評估期間數，警示便會變更為 `ALARM` 狀態。請注意，Application Insights 不會保留這些設定。

若要編輯單一指標的警示，請執行以下步驟：

1. 開啟 CloudWatch 主控台。
2. 在導覽窗格中，選擇 Alarms (警示) > All alarms (所有警示)。

3. 選取「CloudWatch 應用程式深入解析」自動建立的警示旁邊的選項按鈕。然後選擇 Actions (動作)，然後從下拉選單中選取 Edit (編輯)。
4. 在 Metric (指標) 下編輯下列參數。
   a. 在 Statistic (統計數字) 下，選擇其中一個統計數字或預先定義的百分位數，或指定自訂的百分位數。例如 p95.45。
   b. 在 Period (期間) 下，選擇警示的評估期間。評估警示時，每個期間都會彙整為一個資料點。
5. 在 Conditions (條件) 下編輯下列參數。
   a. 選擇指標是否必須大於、小於，或等於閾值。
   b. 指定閾值。
6. 在 Additional configuration (其他組態) 下，編輯下列參數。
   a. 在 Datapoints to alarm (警示的資料點) 下，指定資料點數或評估期間 (必須處於 ALARM 狀態啟動警示)。當兩個值相符時，如果超過指定的連續週期數，系統會建立進入 ALARM 狀態的警示。若要建立 n 個的 m 個警示，請針對第一個資料點，指定低於第二個資料點的值。如需評估警示的詳細資訊，請參閱評估警示。
   b. 在 Missing data treatment (遺失資料處理方式) 下，選擇警示在遺失某些資料點時的行為。如需遺失資料處理的詳細資訊，請參閱設定 CloudWatch 警示如何處理遺失的資料。
   c. 若警示使用百分位數作為監控統計資料，則會出現一個 Percentiles with low samples (低樣本的百分位數) 方塊。選擇是要評估還是忽略具有低抽樣率的案例。若您選擇 ignore (maintain alarm state) (忽略 (維持警示狀態))，則會在抽樣大小過低時一律維持目前的警示狀態。如需低範例百分位數的詳細資訊，請參閱 以百分比為基礎的 CloudWatch 警報和低資料樣本。
7. 選擇下一步。
8. 在 Notification (通知) 下，選取 SNS 主題來在警示處於 ALARM 狀態、OK 狀態或 INSUFFICIENT_DATA 狀態時進行通知。
9. 選擇 Update alarm (更新警示)。

檢視與針對 Application Insights 偵測到的 SAP ASE 問題進行疑難排解

本節可協助您解決在 Application Insights 上設定 SAP ASE 監控時所發生的常見故障診斷問題。

SAP ASE 備份伺服器錯誤

您可透過查看動態建立的儀表板來識別錯誤訊息。儀表板會顯示 SAP ASE 備份伺服器中報告的錯誤訊息。如需有關 SAP ASE 備份伺服器日誌的詳細資訊，請參閱 SAP 文件備份伺服器錯誤日誌記錄。

SAP ASE 長時間執行的交易

識別長時間執行的交易，並確認其是否可停止或執行時間是否為有意設定。如需詳細資訊，請參閱 2180410 - 如何顯示長時間執行交易的交易日誌記錄？ - SAP ASE。

SAP ASE 使用者連線

檢閱 SAP ASE 資料庫是否已根據您想要在資料庫上執行的工作負載，相應地調整大小。如需詳細資訊，請參閱 SAP 文件中的設定使用者連線。

SAP ASE 磁碟空間

您可以查看動態建立的儀表板，識別造成問題的資料庫層。儀表板會顯示相關指標和日誌檔案程式碼片段。請務必了解磁碟增長的原因，並在適當的情況下增加實體磁碟大小、配置的磁碟空間或兩者。如需詳細資訊，請參閱 SAP 文件中的 SAP 文件磁碟大小。

SAP ASE 的 Application Insights 疑難排解

本節提供的步驟可協助您解決 Application Insights 儀表板所傳回的常見錯誤。

錯誤	傳回的錯誤	根本原因	解析度
無法新增超過 60 個監控指標。	`Component cannot have more than 60 monitored metric`	目前的指標限於每個元件 60 個監控指標。	移除不必要的指標以遵守限制。
佈設程序之後，不會顯示 SAP 指標或警示	在 AWS Systems Manager中失敗的 `AWS-ConfigureAWSPackage` 上的 run 指令。輸出會顯示錯誤：`CT-LIBRARY error:ct_connect(): protocol specific layer: external error: The attempt to`	使用者名稱和密碼可能不正確。	確認使用者名稱和密碼有效，然後重新執行上架程序。

錯誤	傳回的錯誤	根本原因	解析度
	`connect to the server failed`		

教學課程：設定 SAP HANA 的監控

本教學課程示範如何設定 CloudWatch 應用程式見解，以設定 SAP HANA 資料庫的監視。您可以使用「CloudWatch 應用程式深入解析」自動儀表板，以視覺化方式呈現問題詳細資料、加速疑難排解，並促進 SAP HANA 資料庫的平均解決時間 (MTTR)。

SAP HANA 主題的 Application Insights

- 支援的環境
- 支援的作業系統
- 功能
- 必要條件
- 設定您的 SAP HANA 資料庫以進行監控
- 管理 SAP HANA 資料庫的監控
- 檢視和疑難排解 CloudWatch 應用程式深入解析偵測到的 SAP HAN
- SAP HANA 的異常偵測
- SAP HANA 的 Application Insights 疑難排解

支援的環境

CloudWatch 應用程式深入解析支援下列系統和模式的 AWS 資源部署。您提供並安裝 SAP HANA 資料庫軟體和支援的 SAP 應用程式軟體。

- 單一 Amazon EC2 執行個體上的 SAP HANA 資料庫 – SAP HANA 在單一節點、擴充規模的架構中，最多有 24 TB 的記憶體。
- 多個 Amazon EC2 執行個體上的 SAP HANA 資料庫 – 多節點、水平擴展架構中的 SAP HANA。
- 跨可用區域 SAP HANA 資料庫高可用性設定 – 使用 SUSE/RHEL 叢集跨兩個可用區域設定高可用性的 SAP HANA。

> **Note**
>
> CloudWatch 應用程式洞見僅支援單一 SID HANA 環境。如果連接了多個 HANA SID，則僅為偵測到的第一個 SID 設定監控。

支援的作業系統

CloudWatch SAP HANA 的應用程式深入解析可在下列作業系統上支援 x86-64 架構：

- SuSE Linux 12 SP4 For SAP
- SuSE Linux 12 SP5 For SAP
- SuSE Linux 15
- SuSE Linux 15 SP1
- SuSE Linux 15 SP2
- SuSE Linux 15 For SAP
- SuSE Linux 15 SP1 For SAP
- SuSE Linux 15 SP2 For SAP
- SuSE Linux 15 SP3 For SAP
- SuSE Linux 15 SP4 For SAP
- SuSE Linux 15 SP5 For SAP
- RedHat Linux 8.6 適用於具有高可用性和更新服務的 SAP
- RedHat Linux 8.5 適用於具有高可用性和更新服務的 SAP
- RedHat Linux 8.4 適用於具有高可用性和更新服務的 SAP
- RedHat Linux 8.3 適用於具有高可用性和更新服務的 SAP
- RedHat Linux 8.2 適用於具有高可用性和更新服務的 SAP
- RedHat Linux 8.1 適用於具有高可用性和更新服務的 SAP
- RedHat Linux 7.9 適用於具有高可用性和更新服務的 SAP

功能

CloudWatch SAP HANA 的應用程式洞察提供下列功能：

- 自動偵測 SAP HANA 工作負載
- 根據靜態閾值自動建立 SAP HANA 警示
- 根據異常偵測自動建立 SAP HANA 警示
- 自動 SAP HANA 日誌模式辨識
- SAP HANA 的運作狀態儀表板
- SAP HANA 的問題儀表板

必要條件

您必須執行下列必要條件，才能使用「CloudWatch 應用程式見解」設定 SAP HANA 資料庫：

- SAP HANA — 在 Amazon EC2 實例上安裝正在運行且可訪問的 SAP HANA 數據庫 2.0 SPS05。
- SAP HANA 資料庫使用者 — 具有監視角色的資料庫使用者必須在 SYSTEM 資料庫和所有租用戶中建立。

  範例

  執行下列 SQL 命令可建立具有監控角色的使用者。

```
su - <sid>adm
hdbsql -u SYSTEM -p <SYSTEMDB password> -d SYSTEMDB
CREATE USER CW_HANADB_EXPORTER_USER PASSWORD <Monitoring user password> NO
 FORCE_FIRST_PASSWORD_CHANGE;
CREATE ROLE CW_HANADB_EXPORTER_ROLE;
GRANT MONITORING TO CW_HANADB_EXPORTER_ROLE;
GRANT CW_HANADB_EXPORTER_ROLE TO CW_HANADB_EXPORTER_USER;
```

- Python 3.8 — 在您的作業系統上安裝 Python 3.8 或更新版本。使用最新版本的 Python。如果在您的操作系統上未檢測到蟒蛇 3，則將安裝 Python 3.6。

  如需更多資訊，請參閱installation example。

  > **Note**
  >
  > 若要使用 SUSE 系統 15 SP4、 RedHat Linux 8.6 及更新版本的作業系統，必須手動安裝 Python 3.8 或更高版本。

- Pip3 — 在您的作業系統上安裝安裝程式 pip3。如果沒有在您的作業系統上偵測到 pip3，則會安裝 pip3。

- HDB 客戶端- CloudWatch 應用程序洞察使用蟒蛇驅動程序連接到 SAP HANA 數據庫。如果客戶端沒有安裝在 python3 下，請確保您有 hdbclient tar 文件版本。`2.10 or later /hana/shared/SID/hdbclient/`
- Amazon CloudWatch 代理程式 — 確保您沒有在 Amazon EC2 執行個體上執行預先存在的 CloudWatch 代理程式。如果您已安裝 CloudWatch 代理程式，請務必從現有的 CloudWatch 代理程式組態檔案中移除您在 Ap CloudWatch plication Insights 中使用的資源組態，以避免合併衝突。如需詳細資訊，請參閱 手動建立或編輯 CloudWatch 代理程式組態檔。
- AWS Systems Manager 啟用 — 在您的執行個體上安裝 SSM 代理程式，且必須為 SSM 啟用執行個體。如需如何安裝 SSM 代理程式的詳細資訊，請參閱《AWS 系統管理員使用指南》中的〈使用 SSM 代理程式〉。
- Amazon EC2 執行個體角色 – 您必須連接下列 Amazon EC2 執行個體角色才能設定資料庫。
  - 您必須連接 `AmazonSSMManagedInstanceCore` 角色，以啟用 Systems Manager。如需詳細資訊，請參閱 AWS Systems Manager 以身分為基礎的政策範例。
  - 您必須附加，`CloudWatchAgentServerPolicy`才能啟用透過 CloudWatch發出的執行個體指標和記錄檔。如需詳細資訊，請參閱建立 IAM 角色和使用者以搭配 CloudWatch代理程式使用。
  - 您必須將下列 IAM 內嵌政策連接到 Amazon EC2 執行個體角色，才能讀取存放在 AWS Secrets Manager中的密碼。如需內嵌政策的詳細資訊，請參閱《AWS Identity and Access Management 使用者指南》中的內嵌政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:ApplicationInsights-*"
        }
    ]
}
```

- AWS 資源群組 — 您必須建立一個資源群組，其中包含應用程式堆疊使用的所有相關 AWS 資源，以便將應用程式上載至「應用程式深入解析」。 CloudWatch 這包括執行 SAP HANA 資料庫的 Amazon EC2 執行個體和 Amazon EBS 磁碟區。如果每個帳戶有多個資料庫，建議您建立一個資源群組，其中包含每個 SAP HANA AWS 資料庫系統的資源。
- IAM 許可 - 對於非管理員使用者：

- 您必須建立 AWS Identity and Access Management (IAM) 政策，讓應用程式深入解析建立服務連結角色，並將其附加至您的使用者身分識別。如需連接政策的步驟，請參閱 IAM 政策。
- 使用者必須擁有在中建立密碼的權限， AWS Secrets Manager 才能儲存資料庫使用者認證。如需詳細資訊，請參閱範例：建立機密的許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:CreateSecret"
      ],
      "Resource": "arn:aws:secretsmanager:*:*:secret:ApplicationInsights-*"
    }
  ]
}
```

- 服務連結角色 — 應用程式深入解析使用 AWS Identity and Access Management (IAM) 服務連結角色。系統會在您於 Application Insights 主控台建立第一個 Application Insights 應用程式時，為您建立服務連結角色。如需詳細資訊，請參閱 將服務連結角色用於 CloudWatch 應用程式深入。

設定您的 SAP HANA 資料庫以進行監控

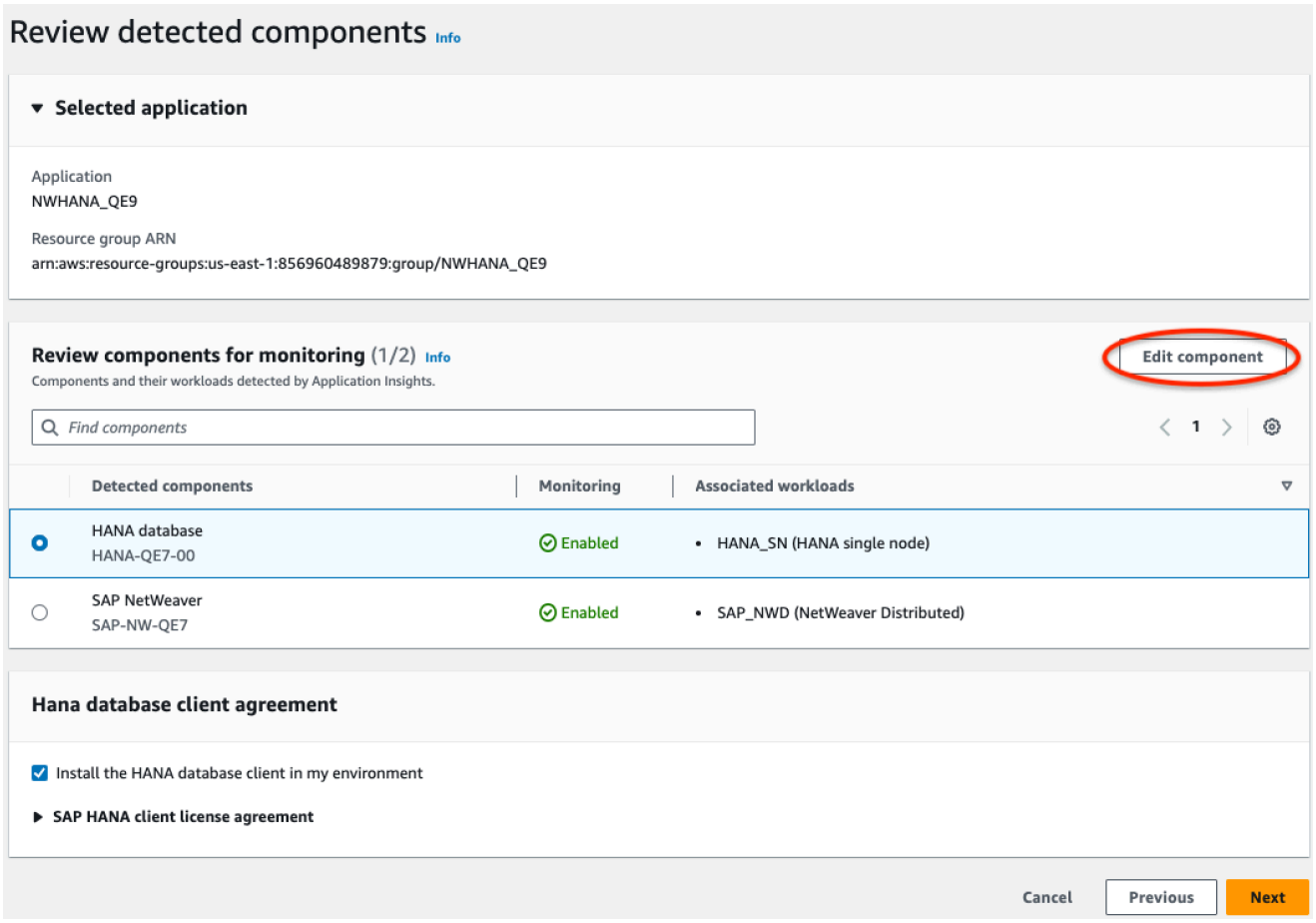
使用下列步驟設定 SAP HANA 資料庫的監控

1. 開啟 CloudWatch 主控台。
2. 從左側導覽窗格中，選擇 Insights 下的 Application Insights。
3. Application Insights 頁面會顯示 Application Insights 監控的應用程式清單，以及每個應用程式的監控狀態。在右上角，選擇 Add an application (新增應用程式)。
4. 在 Specify application details (指定應用程式詳細資訊) 頁面上，從 Resource group (資源群組) 的下拉清單中選取 AWS 資源群組，其中包含您的 SAP HANA 資料庫資源。如果尚未建立應用程式的資源群組，您可以透過選擇 Resource group (資源群組) 下拉單中的 Create new resource group (建立新的資源群組) 建立一個。如需建立資源群組的詳細資訊，請參閱《AWS Resource Groups 使用者指南》。
5. 在監控 CloudWatch 事件下，選取核取方塊，將應用程式洞察監控與 CloudWatch 事件整合，以取得來自 Amazon EBS、Amazon EC2 AWS CodeDeploy、Amazon ECS、 AWS Health API 和通知、Amazon RDS、Amazon S3 和的見解。 AWS Step Functions

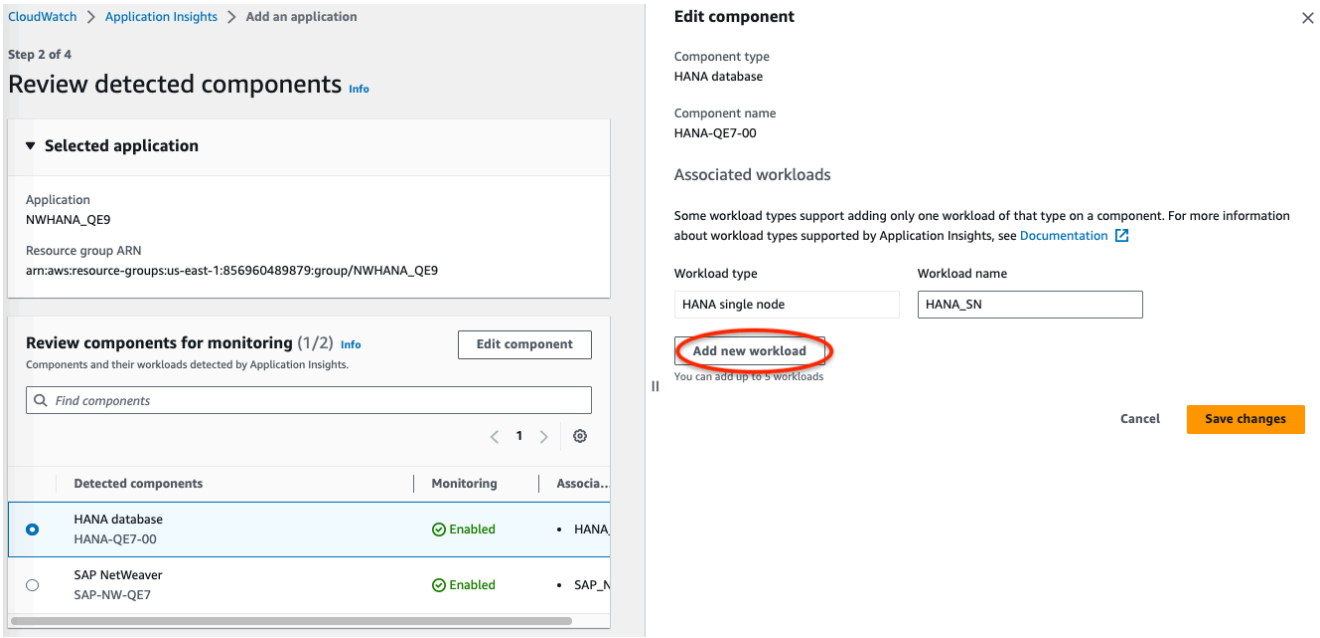
6. 在 [整合對象] 下 AWS Systems Manager OpsCenter，選取 [產生以 AWS Systems Manager OpsCenter OpsItems 供修正動作] 旁邊的核取方塊，以檢視並在針對所選應用程式偵測到問題時取得通知。若要追蹤為解決與 AWS 資源相關的作業工作項目 (稱為 OpsItems) 而執行的作業，請提供 SNS 主題 ARN。
7. 您可以選擇性地輸入標籤，以協助您識別和組織資源。 CloudWatch 應用程式深入解析同時支援以標籤 AWS CloudFormation 為基礎和堆疊式資源群組，但群組除外。 Application Auto Scaling 如需詳細資訊，請參閱 AWS Resource Groups 和標籤使用者指南中的標籤編輯器。
8. 選擇 Next (下一個) 繼續設定監控。
9. 在 [檢閱偵測到的元件] 頁面上，會列出 CloudWatch 應用程式深入解析自動偵測的受監控元件及其工作負載。

    a. 若要將工作負載新增至包含偵測到的 SAP HANA 單一節點工作負載的元件，請選取該元件，然後選擇編輯元件。

    > **Note**
    >
    > 包含偵測到的 SAP HANA 多節點或 HANA 高可用性工作負載的元件，一個元件上僅支援一個工作負載。



b. 若要新增工作負載，請選擇新增工作負載。



c. 完成編輯工作負載後，請選擇儲存變更。

10. 選擇下一步。
11. 在指定元件詳細資訊頁面上，輸入使用者的使用者名稱和密碼。
12. 檢閱您的應用程式監控組態，然後選擇 Submit (提交)。
13. 應用程式詳細資訊頁面隨即開啟，您可以在其中檢視應用程式摘要、受監控元件和工作負載的清單，以及未受監控元件和工作負載。如果選取元件或工作負載旁邊的選項按鈕，則您也可以檢視組態歷史記錄、日誌模式，以及任何已建立的標籤。當您提交組態時，您的帳戶會為 SAP HANA 系統部署所有指標和警示，這可能需要最多 2 小時。

管理 SAP HANA 資料庫的監控

您可以執行下列步驟來管理 SAP HANA 資料庫的使用者憑證、指標和日誌路徑：

1. 開啟 CloudWatch 主控台。
2. 從左側導覽窗格中，選擇 Insights 下的 Application Insights。
3. Application Insights 頁面會顯示 Application Insights 監控的應用程式清單，以及每個應用程式的監控狀態。
4. 在 Monitored components (受監控元件) 下，選取元件名稱旁的選項按鈕。然後，選擇 Manage monitoring (管理監控)。
5. 在 EC2 instance group logs (EC2 執行個體群組日誌) 下，您可以更新現有的日誌路徑、日誌模式集和記錄群組名稱。此外，您可以新增最多三個額外 Application logs (應用程式日誌)。
6. 在 Metrics (指標) 下，您可以根據需求選擇 SAP HANA 指標。SAP HANA 指標名稱的字首為 `hanadb`。每個元件最多可新增 40 個指標。
7. 在 HANA configuration (HANA 組態) 下，輸入 SAP HANA 資料庫的密碼和使用者名稱。這是 Amazon CloudWatch 代理程式用來連線到 SAP HANA 資料庫的使用者名稱和密碼。
8. 在 [自訂警示] 底下，您可以新增其他警示，以供 CloudWatch 應用程式深入解析監控。
9. 檢閱您的應用程式監控組態，然後選擇 Submit (提交)。當您提交組態時，您的帳戶會為 SAP HANA 系統更新所有指標和警示，這可能需要最多 2 小時。

檢視和疑難排解 CloudWatch 應用程式深入解析偵測到的 SAP HAN

下列各節提供的步驟可協助您解決在 Application Insights 上設定 SAP HANA 監控時所發生的常見疑難排解案例。

故障診斷主題

- SAP HANA 資料庫達到記憶體配置上限
- 磁碟已滿事件
- SAP HANA 備份已停止執行

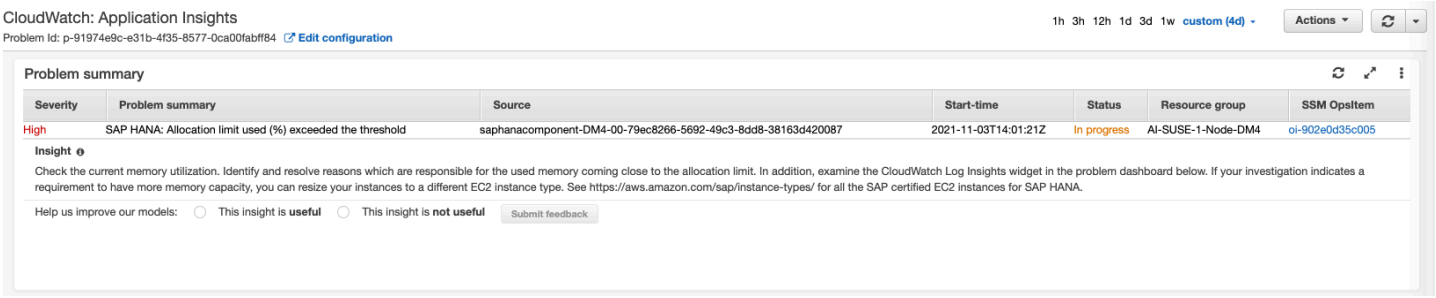
SAP HANA 資料庫達到記憶體配置上限

描述

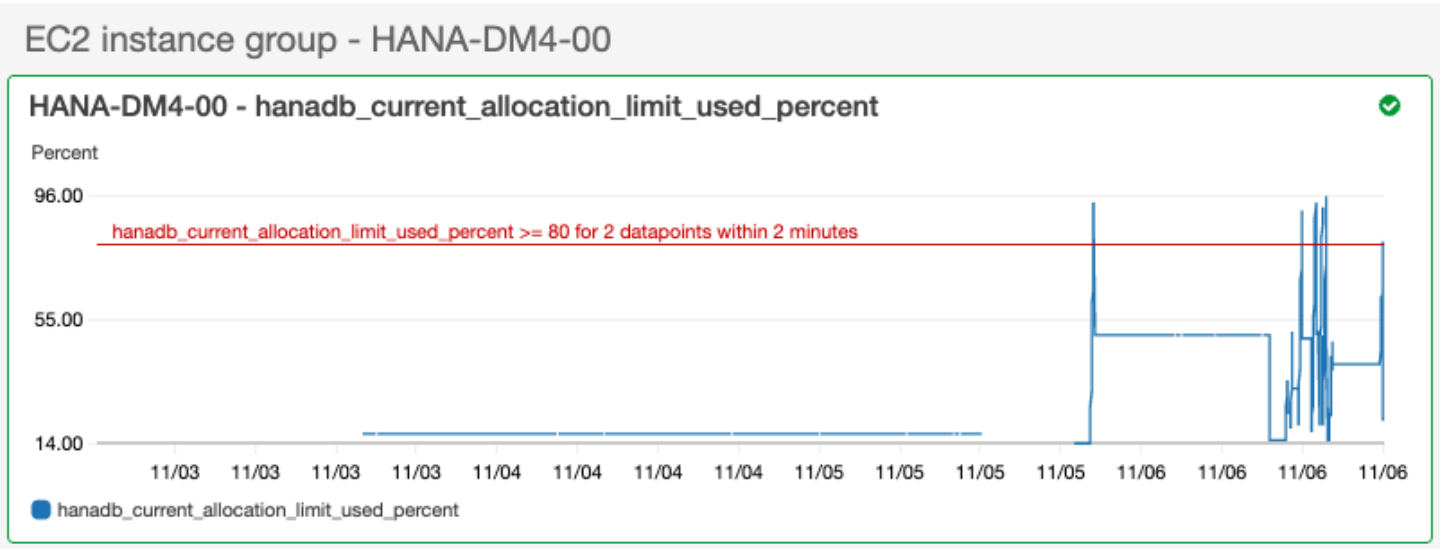
由 SAP HANA 資料庫提供技術的 SAP 應用程式因高記憶體壓力而發生故障，導致應用程式效能降低。

解析度

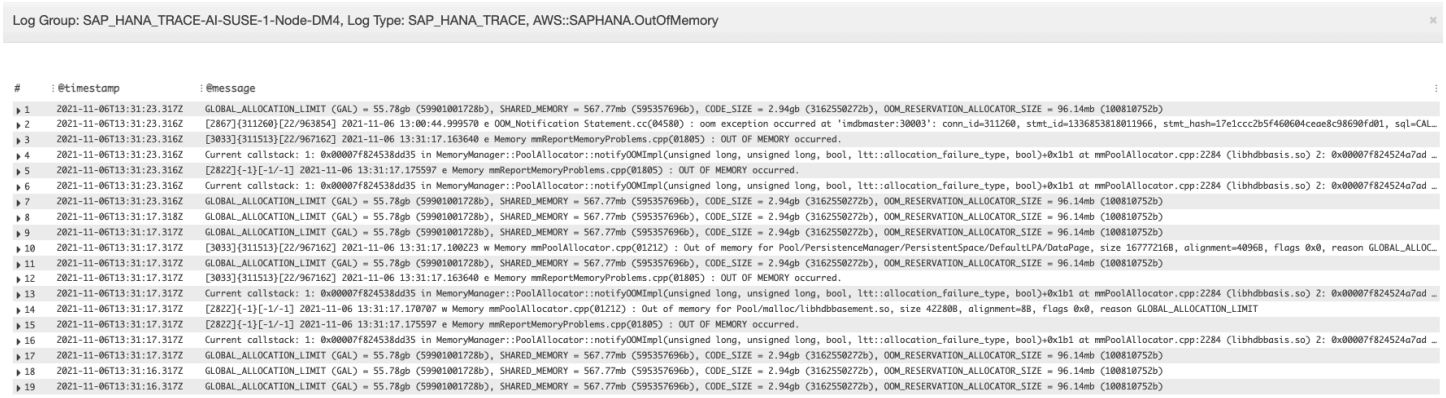
您可以檢查顯示相關指標和日誌檔案程式碼片段的動態建立儀表板，找出造成問題的應用程式層。在下列範例中，問題可能是因為 SAP HANA 系統中的大型資料負載而產生。



使用的記憶體配置超過總記憶體配置上限閾值的 80%。



日誌群組會顯示記憶體容量不足的結構描述 BNR-DATA 和資料表 IMDBMASTER_30003。此外，日誌群組會顯示問題的確切時間、目前的全域位置限制、共用記憶體、程式碼大小和 OOM 保留區配置大小。



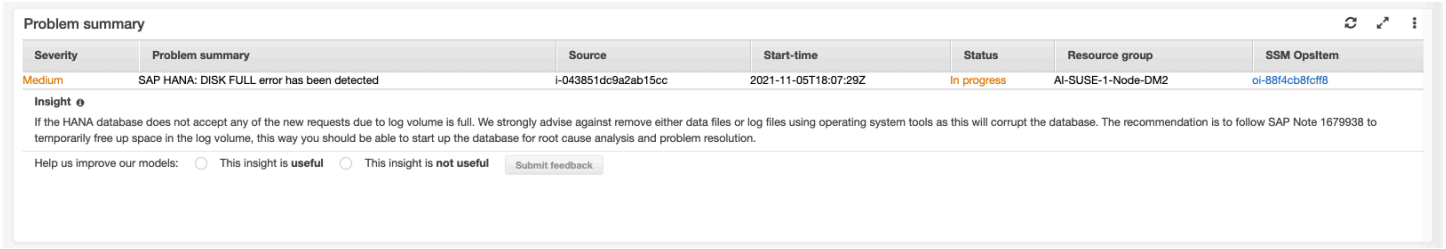
磁碟已滿事件

描述

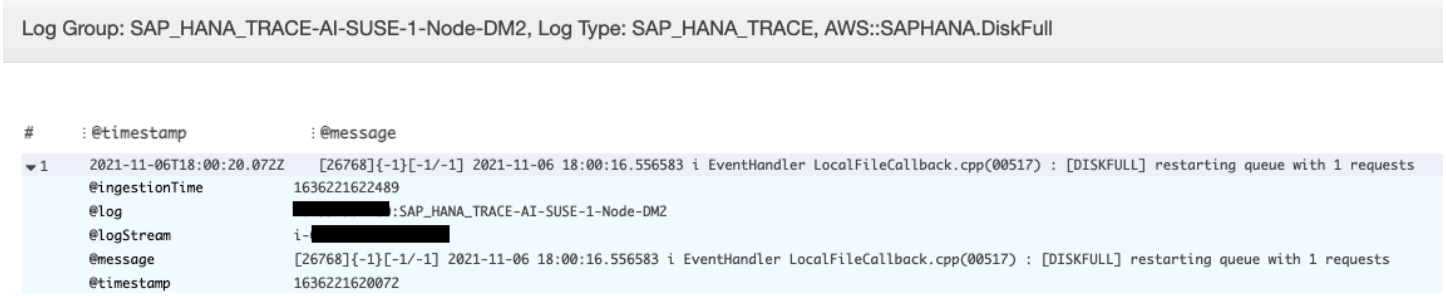
SAP HANA 資料庫提供技術的 SAP 應用程式停止回應，這會導致無法存取資料庫。

解析度

您可以檢查顯示相關指標和日誌檔案程式碼片段的動態建立儀表板，找出造成問題的資料庫層。在下列範例中，問題可能是系統管理員無法啟用自動日誌備份，造成 sap/hana/log 目錄填滿。



問題儀表板中的日誌群組小工具會顯示 DISKFULL 事件。



SAP HANA 備份已停止執行

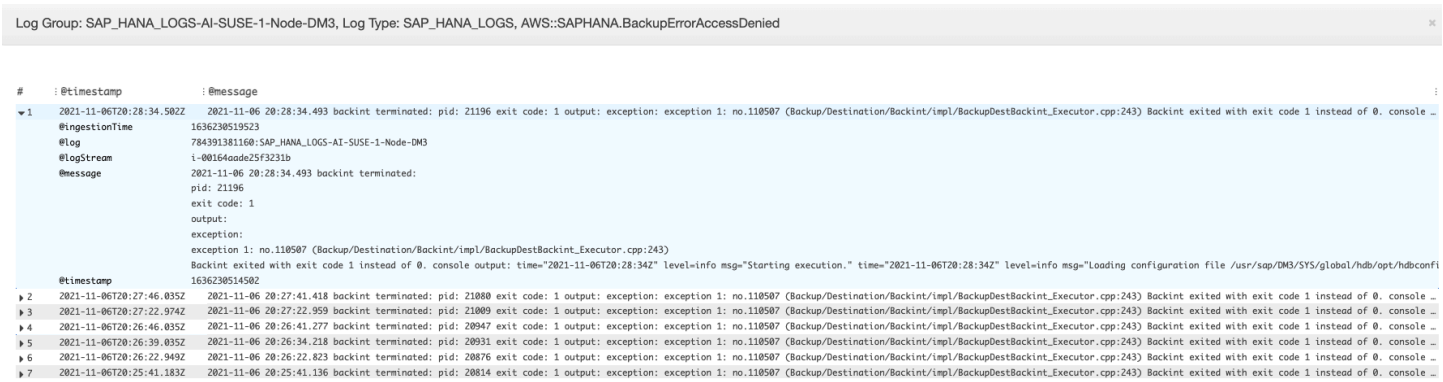
描述

SAP HANA 資料庫提供技術的 SAP 應用程式已停止運作。

解析度

您可以檢查顯示相關指標和日誌檔案程式碼片段的動態建立儀表板，找出造成問題的資料庫層。

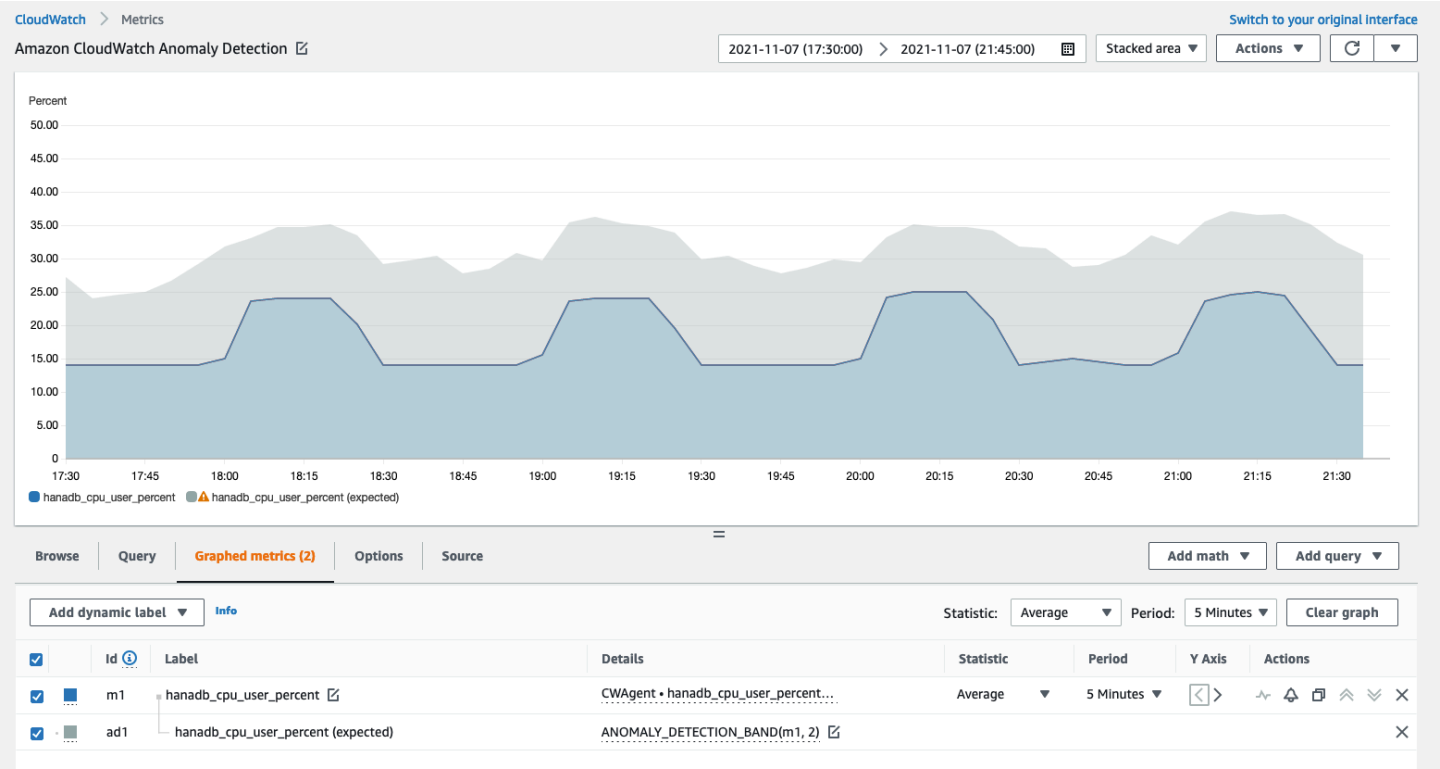
問題儀表板中的日誌群組小工具會顯示 ACCESS DENIED 事件。這包括其他資訊，例如 S3 儲存貯體、S3 儲存貯體資料夾和 S3 儲存貯體區域。



SAP HANA 的異常偵測

針對特定 SAP HANA 指標 (例如執行緒計數)，會 CloudWatch 套用統計和機器學習演算法來定義臨界值。這些演算法會在使用者介入程度最低的情況下，持續分析 SAP HANA 資料庫的指標、判斷正常基準以及表面異常情況。演算法會產生異常偵測模型，模型會產生預期值範圍，代表正常指標行為。

異常偵測演算法會考慮指標的季節性和趨勢變化。季節性變化可能是每小時、每日或每週，如下列 SAP HANA CPU 使用率範例所示。



建立模型之後， CloudWatch 異常偵測會持續評估模型，並對其進行調整，以確保其盡可能準確。這包括重新訓練模型，以調整指標值是否隨著時間而演變，或是遇到突然變化。它還包括預測，以改善季節性，尖峰或稀疏指標的模型。

SAP HANA 的 Application Insights 疑難排解

本節提供的步驟可協助您解決 Application Insights 儀表板所傳回的常見錯誤。

無法新增 60 個以上的受監控度量

輸出顯示以下錯誤。

```
Component cannot have more than 60 monitored metrics
```

根本原因 — 目前的測量結果限制為每個元件 60 個監督的測量結果。

解決方案 — 若要保持在限制之下，請移除不需要的量度。

上線程序後不會顯示任何SAP量度

使用下列資訊，瞭解為何 SAP 指標在上線程序後未顯示在儀表板上。第一個步驟是使用 Amazon EC2 執行個體的 AWS Management Console 或匯出程式日誌未顯示 SAP 指標的原因進行疑難排解。接下來，檢閱錯誤輸出以尋找解決方案。

疑難排解為何 SAP 指標在上線後未顯示

您可以使用 Amazon EC2 執行個體的 AWS Management Console 或匯出程式日誌進行疑難排解。

AWS Management Console

疑難排解使用主控台上線後不會顯示任何 SAP 指標

1. 請在以下位置開啟 AWS Systems Manager 主控台。 https://console.aws.amazon.com/systems-manager/
2. 在左側導覽窗格中，選擇 [狀態管理員]。
3. 在「關聯」下，檢查文件的狀態`AWSEC2-ApplicationInsightsCloudwatchAgentInstallAndConfigure`。如果狀態為`Failed`，請在執行 ID 下選取失敗的 ID 並檢視輸出。
4. 在「關聯」下，檢查文件的狀態`AWS-ConfigureAWSPackage`。如果狀態為`Failed`，請在執行 ID 下選取失敗的 ID 並檢視輸出。

Exporter logs from Amazon EC2 instance

使用匯出程式記錄入職後，疑難排解不會顯示 SAP 指標

1. Connect 到正在執行 SAP HANA 資料庫的 Amazon EC2 執行個體。
2. 尋找使`WORKLOAD_SHORT_NAME`用下列命令的正確命名慣例。您將在以下兩個步驟中使用此簡短名稱。

```
sudo systemctl | grep exporter
```

> **Note**
>
> 應用程式深入解析會根據執`WORKLOAD_SHORT_NAME`行中的工作負載，在服務名稱中新增尾碼。SAP HANA 單一節點、多節點和高可用性部署的簡短名稱為`HANA_SN`、`HANA_MN`、和`HANA_HA`。

3. 若要檢查匯出程式管理員服務記錄檔中是否`WORKLOAD_SHORT_NAME`有錯誤，請執行下列命令，以您在中找到的簡短名稱取代Step 2。

```
sudo journalctl -e --unit=prometheus-
hanadb_exporter_manager_WORKLOAD_SHORT_NAME.service
```

4. 如果匯出程式管理員服務記錄檔未顯示錯誤，請執行下列命令來檢查匯出程式服務記錄檔中是否有錯誤。

```
sudo journalctl -e --unit=prometheus-hanadb_exporter_WORKLOAD_SHORT_NAME.service
```

解決上線後未出現的 SAP 指標的常見根本原因

下列範例說明如何解決上線後未出現的 SAP 量度的常見根本原因。

- 輸出顯示以下錯誤。

```
Reading json config file path: /opt/aws/amazon-cloudwatch-agent/etc/amazon-
cloudwatch-agent.d/default ...
Reading json config file path: /opt/aws/amazon-cloudwatch-agent/etc/
amazon-cloudwatch-agent.d/ssm_AmazonCloudWatch-ApplicationInsights-
SSMParameterForTESTCWEC2INSTANCEi0d88867f1f3e36285.tmp ...
2023/11/30 22:25:17 Failed to merge multiple json config files.
2023/11/30 22:25:17 Failed to merge multiple json config files.
2023/11/30 22:25:17 Under path : /metrics/append_dimensions | Error : Different
 values are specified for append_dimensions
2023/11/30 22:25:17 Under path : /metrics/metrics_collected/disk | Error : Different
 values are specified for disk
2023/11/30 22:25:17 Under path : /metrics/metrics_collected/mem | Error : Different
 values are specified for mem
2023/11/30 22:25:17 Configuration validation first phase failed. Agent version: 1.0.
 Verify the JSON input is only using features supported by this version.
```

解決方案 —「應用程式深入解析」嘗試設定與現有 CloudWatch 代理程式組態檔案一部分預先設定的相同度量。移除現有代理程式組態檔下的現有檔案，/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.d/或從現有 CloudWatch 代理程式組態檔移除造成衝突的測量結果。

- 輸出顯示以下錯誤。

```
Unable to find a host with system database, for more info rerun using -v
```

解決方案 — 使用者名稱、密碼或資料庫連接埠可能不正確。確認使用者名稱、密碼和連接埠是否有效，然後重新執行上線程序。

- 輸出顯示以下錯誤。

```
This hdbcli installer is not compatible with your Python interpreter
```

解析度 — 升級 pip3 和輪子，如下列範例中所示的 Python 3.6。

```
python3.6 -m pip install --upgrade pip setuptools wheel
```

- 輸出顯示以下錯誤。

```
Unable to install hdbcli using pip3. Please try to install it
```

解決方案 — 確定您已遵循hdbclient先決條件，或在 pip3 下hdbclient手動安裝。

- 輸出顯示以下錯誤。

```
Package 'boto3' requires a different Python: 3.6.15 not in '>= 3.7'
```

解析度 — 此作業系統版本需要 Python 3.8 或更高版本。檢查 Python 3.8 的先決條件並進行安裝。

- 輸出會顯示下列其中一個安裝錯誤。

```
Can not execute `setup.py` since setuptools is not available in the build environment
```

或

```
[SSL: CERTIFICATE_VERIFY_FAILED]
```

解決方案 — 使用 SUSE Linux 指令來安裝 Python，如下列範例所示。下列範例會安裝最新版本的 Python 3.8。

```
wget https://www.python.org/ftp/python/3.8.<LATEST_RELEASE>/
Python-3.8.<LATEST_RELEASE>.tgz
tar xf Python-3.*
cd Python-3.*/
sudo zypper install make gcc-c++ gcc automake autoconf libtool
sudo zypper install zlib-devel
```

```
sudo zypper install libopenssl-devel libffi-devel
./configure --with-ensurepip=install
sudo make
sudo make install
sudo su
python3.8 -m pip install --upgrade pip setuptools wheel
```

教學課程：設定 SAP 的監督 NetWeaver

本教學課程示範如何設定 Amazon CloudWatch 應用程式深入解析，以設定 SAP 的監控 NetWeaver。您可以使用「 CloudWatch 應用程式深入解析」自動儀表板來視覺化問題詳細資料、加速疑難排解，並縮短 SAP NetWeaver 應用程式伺服器的平均解決時間 (MTTR)。

CloudWatch SAP NetWeaver 主題的應用程式洞察

- 支援的環境
- 支援的作業系統
- 功能
- 必要條件
- 設定 SAP NetWeaver 應用程式伺服器以進行監控
- 管理 SAP NetWeaver 應用程式伺服器的監控
- 檢視並疑難排解應 CloudWatch 用程式深入 NetWeaver 解析所偵測到
- 疑難排解 SAP 應用程式見解 NetWeaver

支援的環境

CloudWatch 應用程式深入解析支援下列系統和模式的 AWS 資源部署。

- SAP NetWeaver 標準系統部署。
- SAP NetWeaver 分散式部署在多個 Amazon EC2 執行個體上。
- 跨可用區域 SAP NetWeaver 高可用性設定 — SAP 使 NetWeaver 用 SUSE/RHEL 叢集在兩個可用區域中設定高可用性。

支援的作業系統

CloudWatch 下列作業系統支援 SAP NetWeaver 的應用程式深入解析：

- Oracle Linux 8
- Red Hat Enterprise Linux 7.6
- Red Hat Enterprise Linux 7.7
- Red Hat Enterprise Linux 7.9
- Red Hat Enterprise Linux 8.1
- Red Hat Enterprise Linux 8.2
- Red Hat Enterprise Linux 8.4
- Red Hat Enterprise Linux 8.6
- SUSE Linux Enterprise Server 15 for SAP
- SUSE Linux Enterprise Server 15 SP1 for SAP
- SUSE Linux Enterprise Server 15 SP2 for SAP
- SUSE Linux Enterprise Server 15 SP3 for SAP
- SUSE Linux Enterprise Server 15 SP4 for SAP
- SUSE Linux Enterprise Server 12 SP4 for SAP
- SUSE Linux Enterprise Server 12 SP5 for SAP
- SUSE Linux Enterprise Server 15，高可用性模式除外
- SUSE Linux Enterprise Server 15 SP1，高可用性模式除外
- SUSE Linux Enterprise Server 15 SP2，高可用性模式除外
- SUSE Linux Enterprise Server 15 SP3，高可用性模式除外
- SUSE Linux Enterprise Server 15 SP4，高可用性模式除外
- SUSE Linux Enterprise Server 12 SP4，高可用性模式除外
- SUSE Linux Enterprise Server 12 SP5，高可用性模式除外

功能

CloudWatch SAP NetWeaver 7.0x—7.5 倍 (包括 ABAP 平台) 的應用程式洞察提供下列功能：

- 自動 SAP NetWeaver 工作負載偵測
- 根據靜態閾值自動建立 SAP NetWeaver 警示

- 自動識別 SAP NetWeaver 日誌模式
- SAP 的 Health 儀表板 NetWeaver
- SAP 的問題儀表板 NetWeaver

必要條件

您必須執行下列先決條件，才能使用「 CloudWatch 應 NetWeaver 用程式深入解析」設

- AWS 啟用 Systems Manager — 在 Amazon EC2 執行個體上安裝 SSM 代理程式，並啟用 SSM 的執行個體。如需有關如何安裝 SSM Agent 的資訊，請參閱《AWS Systems Manager 使用者指南》中的設定 AWS Systems Manager。
- Amazon EC2 執行個體角色 — 您必須連接下列 Amazon EC2 執行個體角色，才能設定 SAP NetWeaver 監控。
  - 您必須連接 `AmazonSSMManagedInstanceCore` 角色，以啟用 Systems Manager。如需詳細資訊，請參閱 AWS Systems Manager 以身分為基礎的政策範例。
  - 您必須附加`CloudWatchAgentServerPolicy`原則，才能啟用透過 CloudWatch發出執行個體指標和記錄檔。如需詳細資訊，請參閱建立 IAM 角色和使用者以搭配 CloudWatch 代理程式使用。
- AWS 資源群組 — 您必須建立一個資源群組，其中包含應用程式堆疊使用的所有相關 AWS 資源，以便將應用程式上載至「應用程式深入解析」。 CloudWatch 這包括執行 SAP NetWeaver 應用程式伺服器的亞馬 Amazon EC2 執行個體、Amazon EFS 和 Amazon EBS 磁碟區。如果每個帳戶有多個 SAP NetWeaver 系統，建議您建立一個資源群組，其中包含每個 SAP NetWeaver 系統的 AWS 資源。如需有關建立資源群組的詳細資訊，請參閱《AWS 資源群組和標籤使用者指南》。
- IAM 許可 — 對於沒有管理存取權的使用者，您必須建立 AWS Identity and Access Management (IAM) 政策，以允許應用程式洞察建立服務連結角色，並將其附加到使用者的身分識別。如需有關如何建立政策的更多資訊，請參閱 IAM 政策。
- 服務連結角色 — 應用程式深入解析使用 AWS Identity and Access Management (IAM) 服務連結角色。系統會在您於 Application Insights 主控台建立第一個 Application Insights 應用程式時，為您建立服務連結角色。如需詳細資訊，請參閱 將服務連結角色用於 CloudWatch 應用程式深入。
- Amazon CloudWatch 代理程式 — 應用程式洞察會安裝和設定代理程式 CloudWatch 。如果您已安裝 CloudWatch 代理程式，應用程式洞察會保留您的組態。若要避免合併衝突，請從現有的代理程式組態檔案中移除您要在「應用程 CloudWatch 式深入解析」中使用的資源組態。如需詳細資訊，請參閱 手動建立或編輯 CloudWatch 代理程式組態檔。

設定 SAP NetWeaver 應用程式伺服器以進行監控

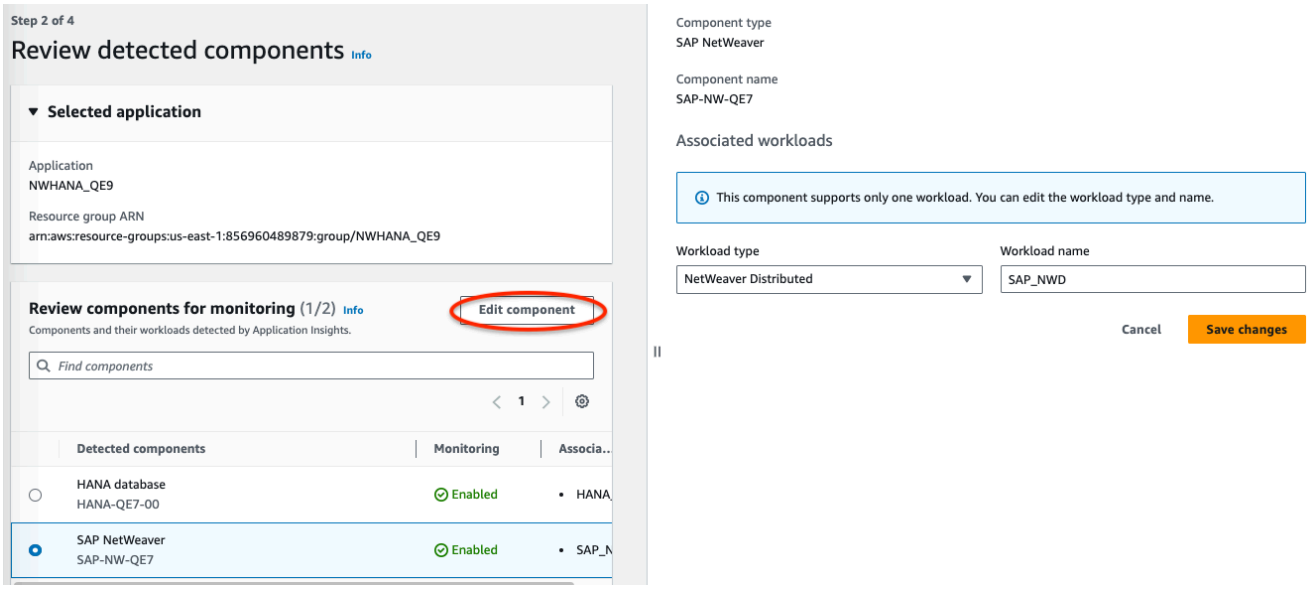
請使用下列步驟來設定 SAP NetWeaver 應用程式伺服器的監視。

設定監控

1. 開啟 CloudWatch 主控台。
2. 在左側導覽窗格中，選取 Insights 下的 Application Insights。
3. Application Insights 頁面會顯示 Application Insights 監控的應用程式清單，以及每個應用程式的監控狀態。在右上角，選取 Add an application (新增應用程式)。
4. 在 [指定應用程式詳細資訊] 頁面的 [資源群組] 下的下拉式清單中，選取您建立的包含 SAP NetWeaver 資源的資源群組。 AWS 如果尚未建立應用程式的資源群組，您可以透過選擇 Resource group (資源群組) 下拉式清單中的 Create new resource group (建立新的資源群組) 建立一個。
5. 在 Automatic monitoring of new resources (自動監控新資源) 下，選取核取方塊，以允許 Application Insights 在加入後自動監控新增至應用程式資源群組的資源。
6. 在監控 EventBridge 事件下，選取核取方塊，將應用程式洞察監控與 CloudWatch 事件整合，以取得來自 Amazon EBS、Amazon EC2 AWS CodeDeploy、Amazon ECS、 AWS Health API 和通知、Amazon RDS、Amazon S3 和的見解。 AWS Step Functions
7. 在 [整合對象] 下 AWS Systems Manager OpsCenter，選取 [產生以 AWS Systems Manager OpsCenter OpsItems 供修正動作] 旁邊的核取方塊，以檢視並在針對所選應用程式偵測到問題時取得通知。若要追蹤為解決與 AWS 資源相關的作業工作項目 (稱為 OpsItems) 而執行的作業，請提供 SNS 主題 ARN。
8. 您可以選擇性地輸入標籤，以協助您識別和組織資源。 CloudWatch 應用程式深入解析同時支援以標籤 AWS CloudFormation 為基礎和堆疊式資源群組，但群組除外。 Application Auto Scaling 如需詳細資訊，請參閱 AWS Resource Groups 和標籤使用者指南中的標籤編輯器。
9. 若要檢閱偵測到的元件，請選擇下一步。
10. 在 [檢閱偵測到的元件] 頁面上，會列出 CloudWatch 應用程式深入解析自動偵測的受監控元件及其工作負載。

    - 若要編輯工作負載類型和名稱，請選擇編輯元件。

      > **Note**
      >
      > 包含偵測到的 NetWeaver 分散式或 NetWeaver 高可用性工作負載的元件僅支援元件上的一個工作負載。



11. 選擇下一步。
12. 在 Specify component details (指定元件詳細資訊) 頁面上，選擇 Next (下一步)。
13. 檢閱您的應用程式監控組態，然後選擇提交。
14. 應用程式詳細資訊頁面隨即開啟，您可以在其中檢視應用程式摘要、儀表板、元件和工作負載。您也可以檢視 Configuration history (組態歷史記錄)、Log patterns (日誌模式)，以及任何已建立的 Tags (標籤)。在您提交應用程式之後，「CloudWatch 應用程式深入解析」會部署 SAP NetWeaver 系統的所有指標和警示，最多可能需要一個小時。

管理 SAP NetWeaver 應用程式伺服器的監控

使用下列步驟來管理 SAP NetWeaver 應用程式伺服器的監視。

管理監控

1. 開啟 CloudWatch 主控台。
2. 在左側導覽窗格中，選取 Insights 下的 Application Insights。
3. 選擇 List view (清單檢視) 索引標籤。
4. Application Insights 頁面會顯示 Application Insights 監控的應用程式清單，以及每個應用程式的監控狀態。
5. 選取您的應用程式。
6. 選擇 Components (元件) 索引標籤。

7. 在 Monitored components (受監控元件) 下，選取元件名稱旁的選項按鈕。然後，選取 Manage monitoring (管理監控)。
8. 在 Instance logs (執行個體日誌) 下，您可以更新現有的日誌路徑、日誌模式集和日誌群組名稱。此外，您可以新增最多三個額外 Application logs (應用程式日誌)。
9. 在「量度」下，您可以根 NetWeaver 據需求選取 SAP 指標。SAP NetWeaver 量度名稱前面加上。sap每個元件最多可新增 40 個指標。
10. 在 [自訂警示] 底下，您可以新增其他警示，以供 CloudWatch 應用程式深入解析監控。
11. 檢閱您的應用程式監控組態，然後選擇 Save (儲存)。當您提交組態時，您的帳戶會更新 SAP NetWeaver 系統的所有指標和警示。

檢視並疑難排解應 CloudWatch 用程式深入 NetWeaver 解析所偵測到

下列各節提供的步驟可協助您解決 NetWeaver 在應用程式深入解析上設定 SAP 監視時所發生的常見疑難排解案例。

故障診斷主題

- SAP NetWeaver 資料庫連線問題
- SAP NetWeaver 應用程式可用性問

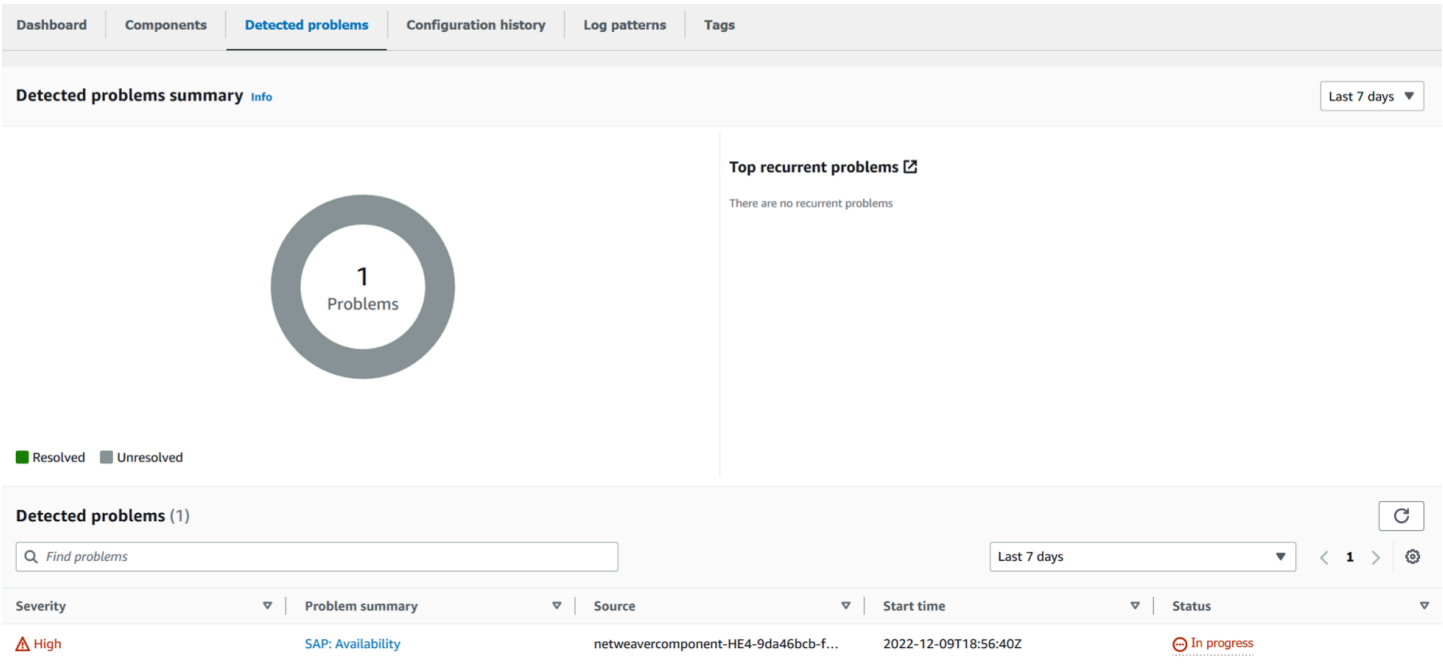
SAP NetWeaver 資料庫連線問題

描述

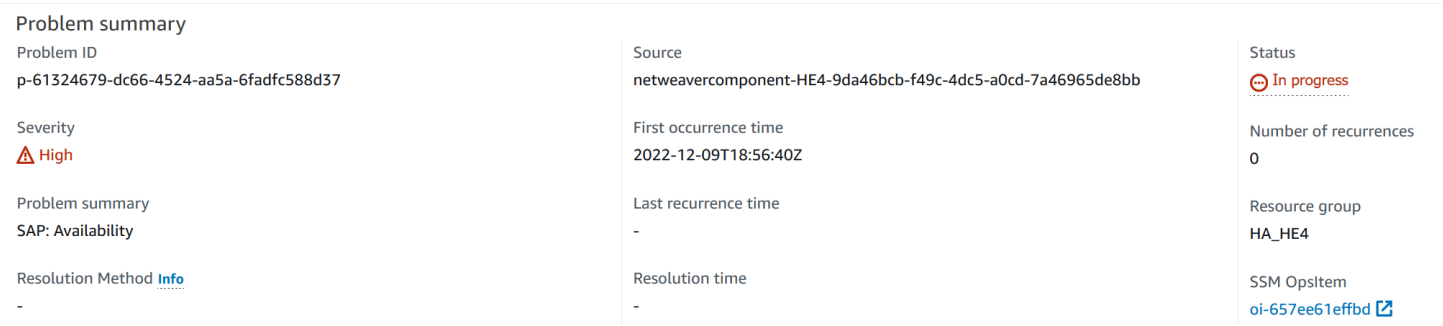
您的 SAP NetWeaver 應用程式遇到資料庫連線問題。

原因

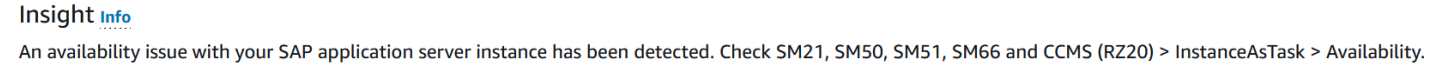
您可以移至「應用程式深入解析」主控台，並檢查 SAP CloudWatch NetWeaver 應用程式深入解析問題儀表板，以識別連線問題。選取 Problem summary (問題摘要) 下方的連結，即可查看特定問題。



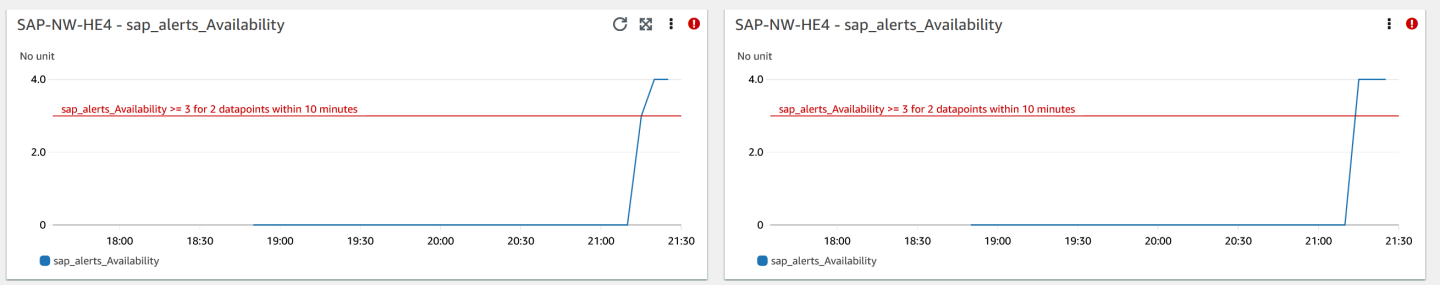
在下列範例中，在 Problem summary (問題摘要) 下，SAP：可用性是問題所在。



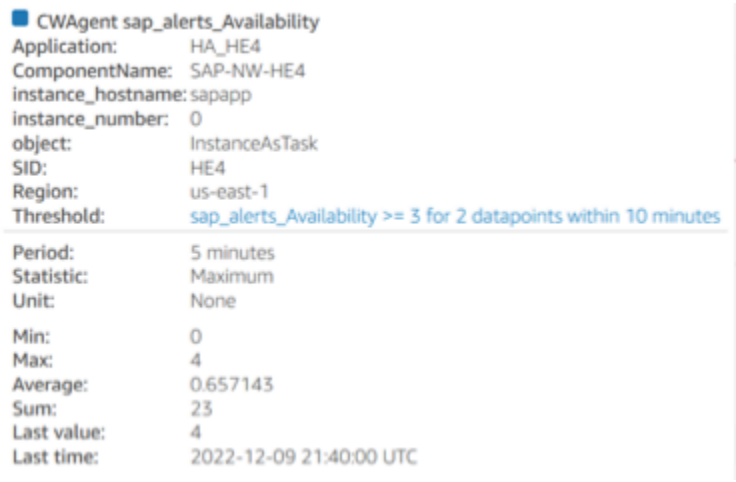
緊接在 Problem summary (問題摘要) 之後，Insight 區段會提供有關錯誤以及您可以從何處取得有關問題原因的詳細資訊。



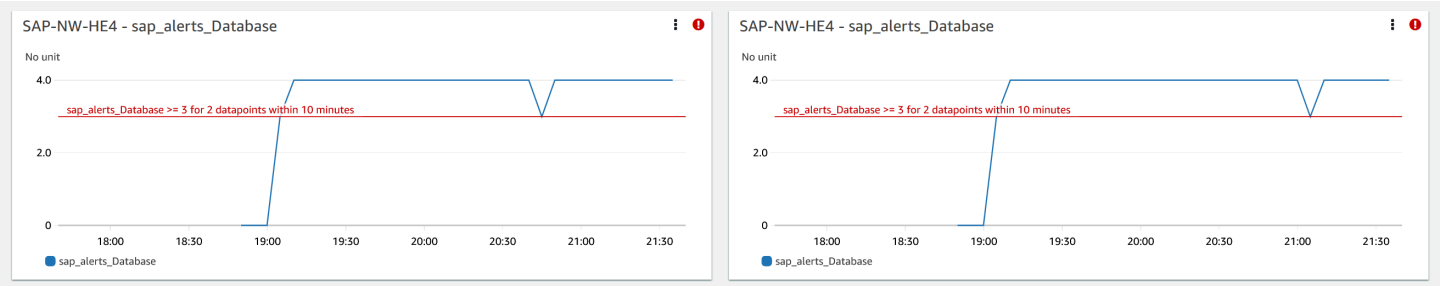
在此相同的問題儀表板上，您可以檢視問題偵測功能已進行分組的相關日誌和指標，以協助您隔離錯誤的原因。該`sap_alerts_Availability`指標會追蹤 SAP NetWeaver 系統隨時間的可用性。您可以使用歷史追蹤來關聯指標何時啟動錯誤狀態或超出警示閾值。在下列範例中，SAP NetWeaver 系統存在可用性問題。此範例會顯示兩個警示，因為有兩個 SAP 應用程式伺服器執行個體，已針對每個執行個體建立一個警示。



如需有關每個警示的詳細資訊，請將滑鼠游標暫留在 `sap_alerts_Availability` 指標名稱上。



在下列範例中，`sap_alerts_Database` 指標顯示資料庫層發生問題或故障。此警示表示 SAP NetWeaver 在連線至其資料庫或與其資料庫通訊時發生問題。



由於資料庫是 SAP 的關鍵資源 NetWeaver，因此當資料庫發生問題或故障時，您可能會收到許多相關警示。在下列範例中，因為無法使用資料庫，所以會啟動 `sap_alerts_FrontendResponseTime` 和 `sap_alerts_LongRunners` 指標。

解決方案

Application Insights 會每小時監控偵測到的問題。如果 SAP 記錄檔中沒有新的相關 NetWeaver 記錄項目，則會將較舊的記錄項目視為已解析。您必須修復與 CloudWatch警報相關的任何錯誤情況。修復錯誤條件後，當警示和日誌復原時，警示就會解決。解決所有 CloudWatch 記錄檔錯誤和警示後，Application Insights 會停止偵測錯誤，並在一小時內自動解決問題。我們建議您解決所有日誌錯誤狀況和警示，以便在問題儀表板上找到最新的問題。

在下列範例中，SAP 可用性問題已解決。

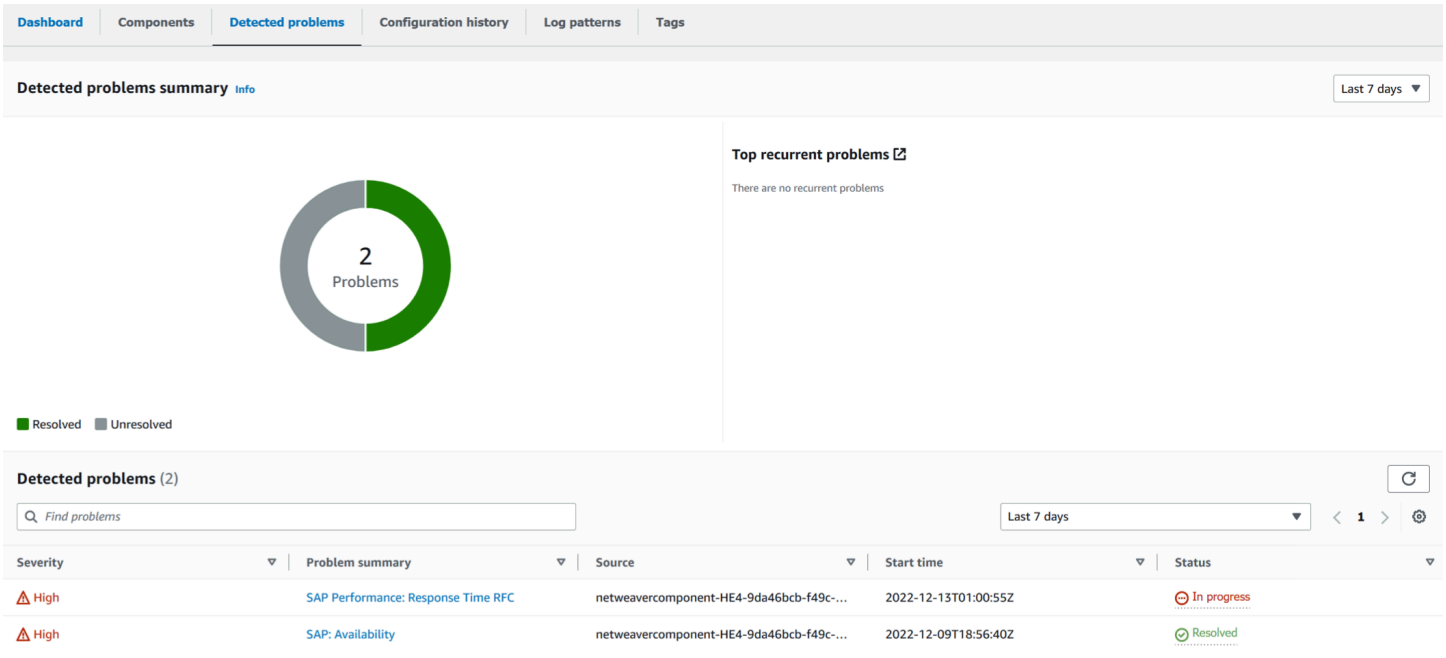
SAP NetWeaver 應用程式可用性問

描述

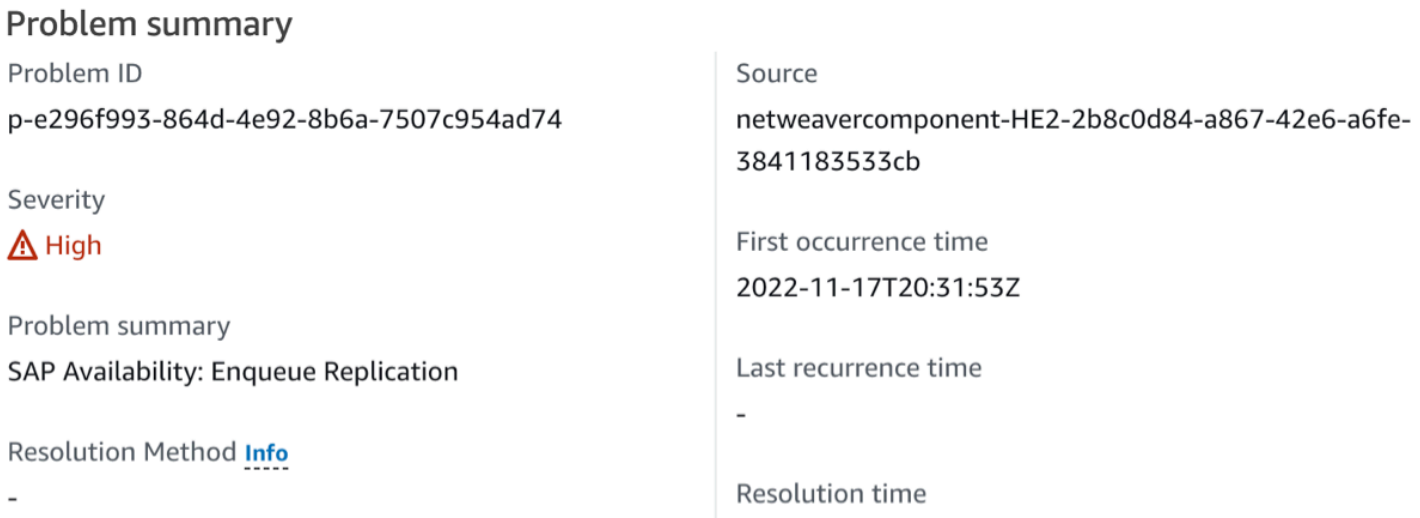
您的 SAP NetWeaver 高可用性排入佇列複寫已停止運作。

原因

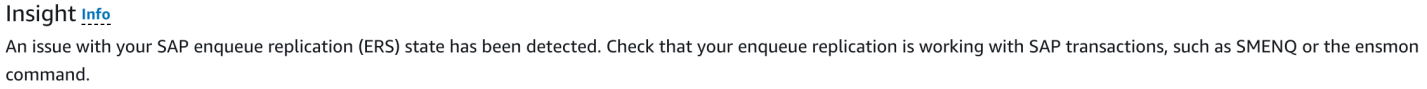
您可以移至「應用程式深入解析」主控台，並檢查 SAP CloudWatch NetWeaver 應用程式深入解析問題儀表板，以識別連線問題。選取 Problem summary (問題摘要) 下方的連結，即可查看特定問題。



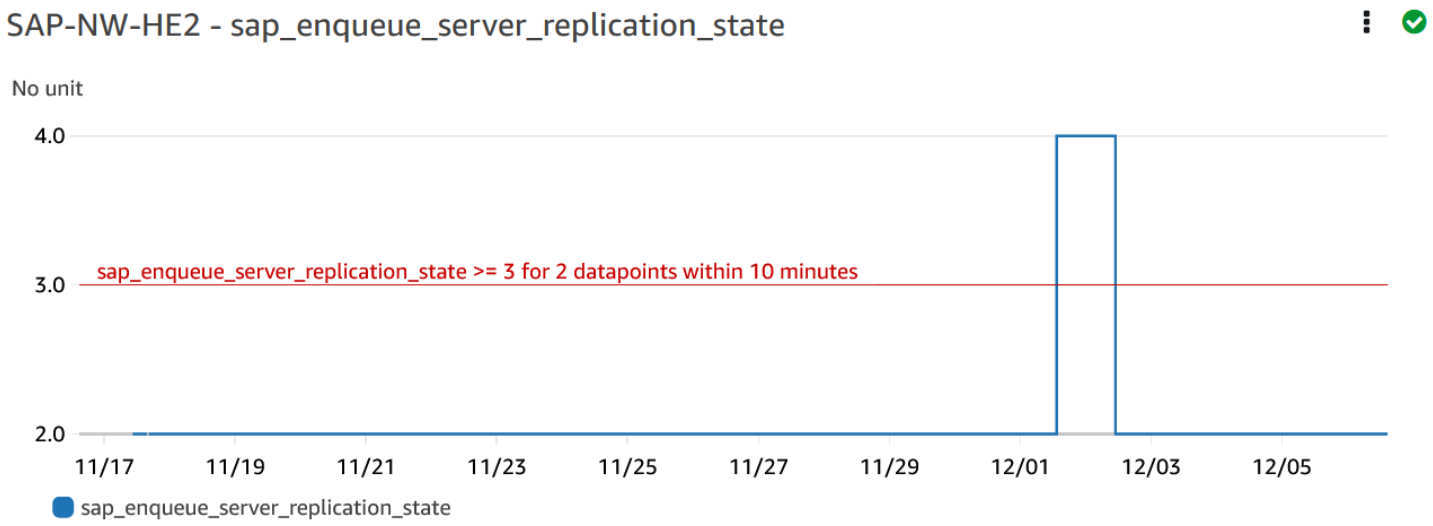
在下列範例中，在 Problem summary (問題摘要) 下，高可用性排入佇列複寫就是問題所在。



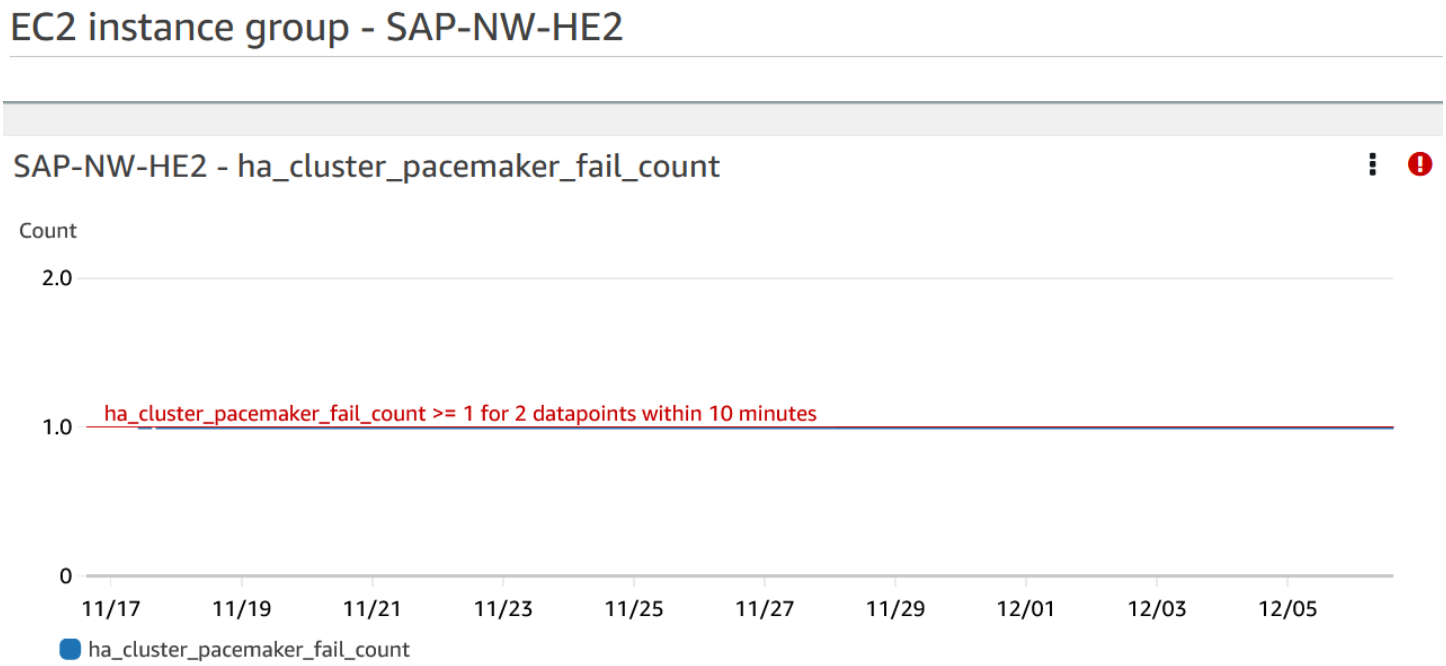
緊接在 Problem summary (問題摘要) 之後，Insight 區段會提供有關錯誤以及您可以從何處取得有關問題原因的詳細資訊。



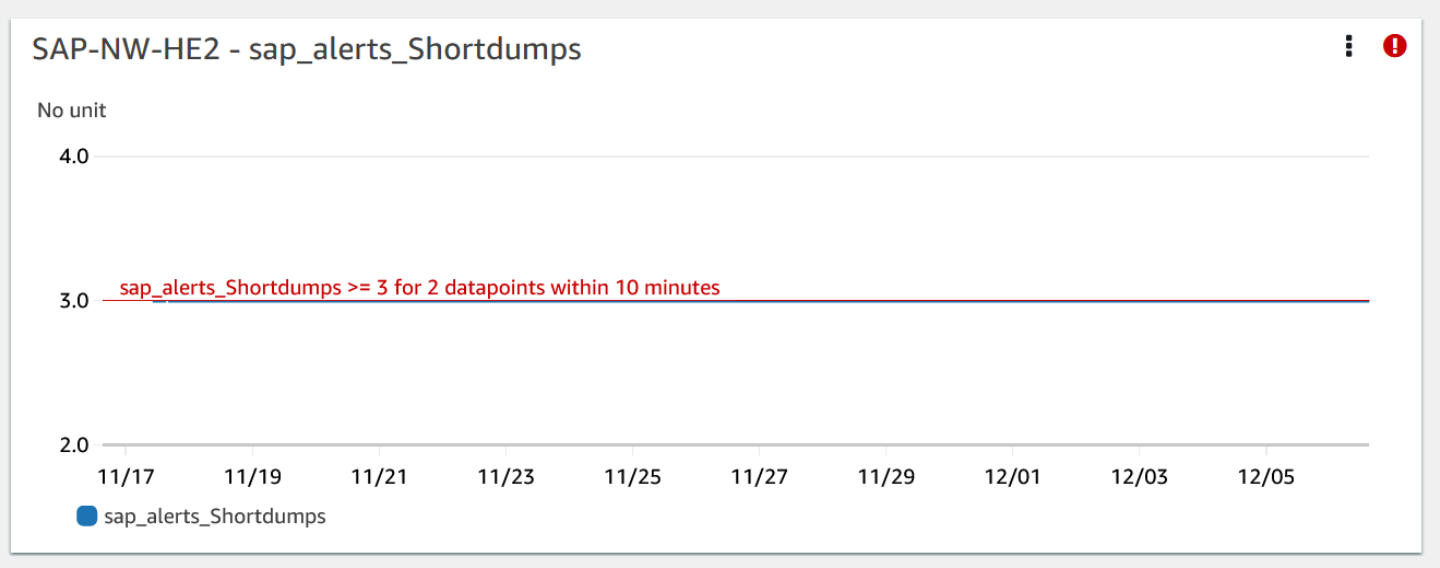
下列範例顯示問題儀表板，您可以在其中檢視日誌和指標，這些日誌和指標會分組以協助您隔離錯誤的原因。`sap_enqueue_server_replication_state` 指標會追蹤一段時間內的值。您可以使用歷史追蹤來關聯指標何時啟動錯誤狀態或超出警示閾值。



在下列範例中，`ha_cluster_pacemaker_fail_count` 指標會顯示高可用性 Pacemaker 叢集發生資源故障。失敗計數大於或等於 1 的特定 Pacemaker 資源會在元件儀表板中識別出來。

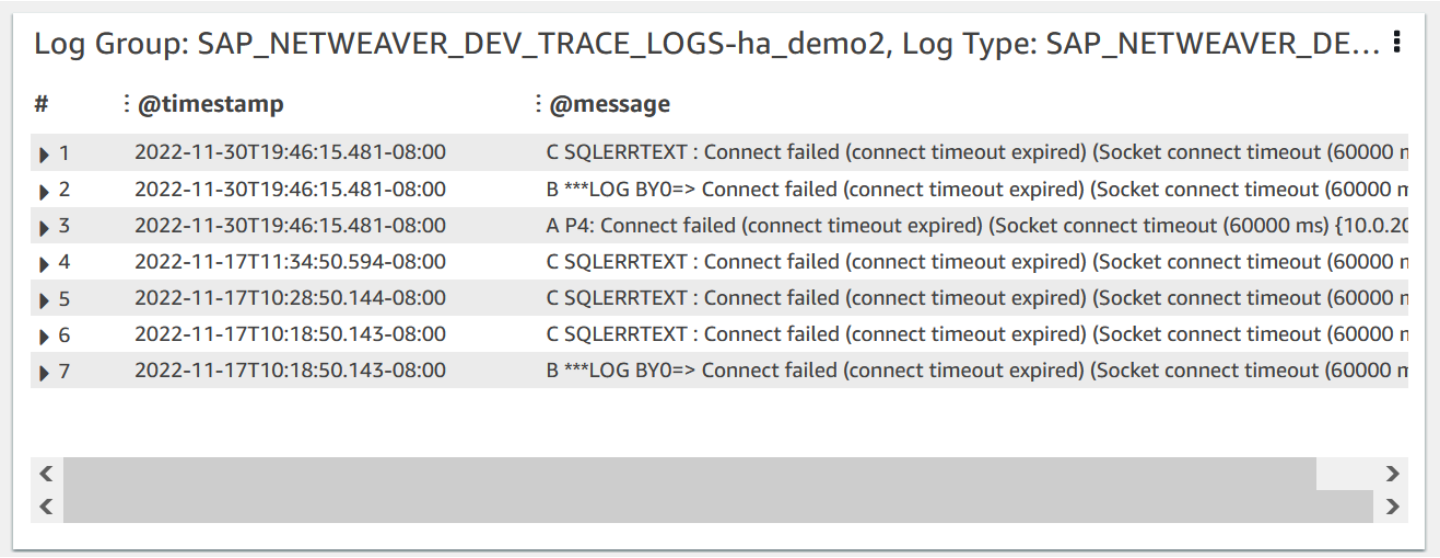


下列範例顯示 `sap_alerts_Shortdumps` 指標，表示偵測到問題時，SAP 應用程式效能已降低。

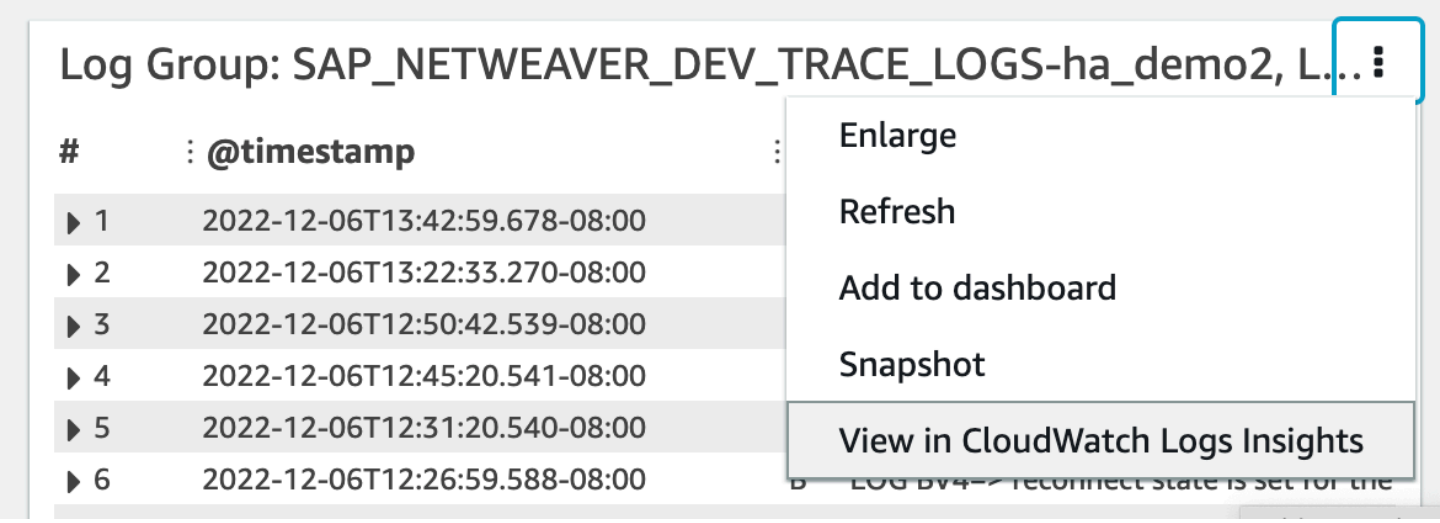


日誌

記錄項目有助於更好地瞭解偵測到問題時發生在 SAP NetWeaver 層的問題。問題儀表板中的日誌群組小工具會顯示問題的具體時間。



若要查看有關記錄的詳細資訊，請選取右上角的三個垂直點，然後選取在 CloudWatch 記錄檔見解中檢視。

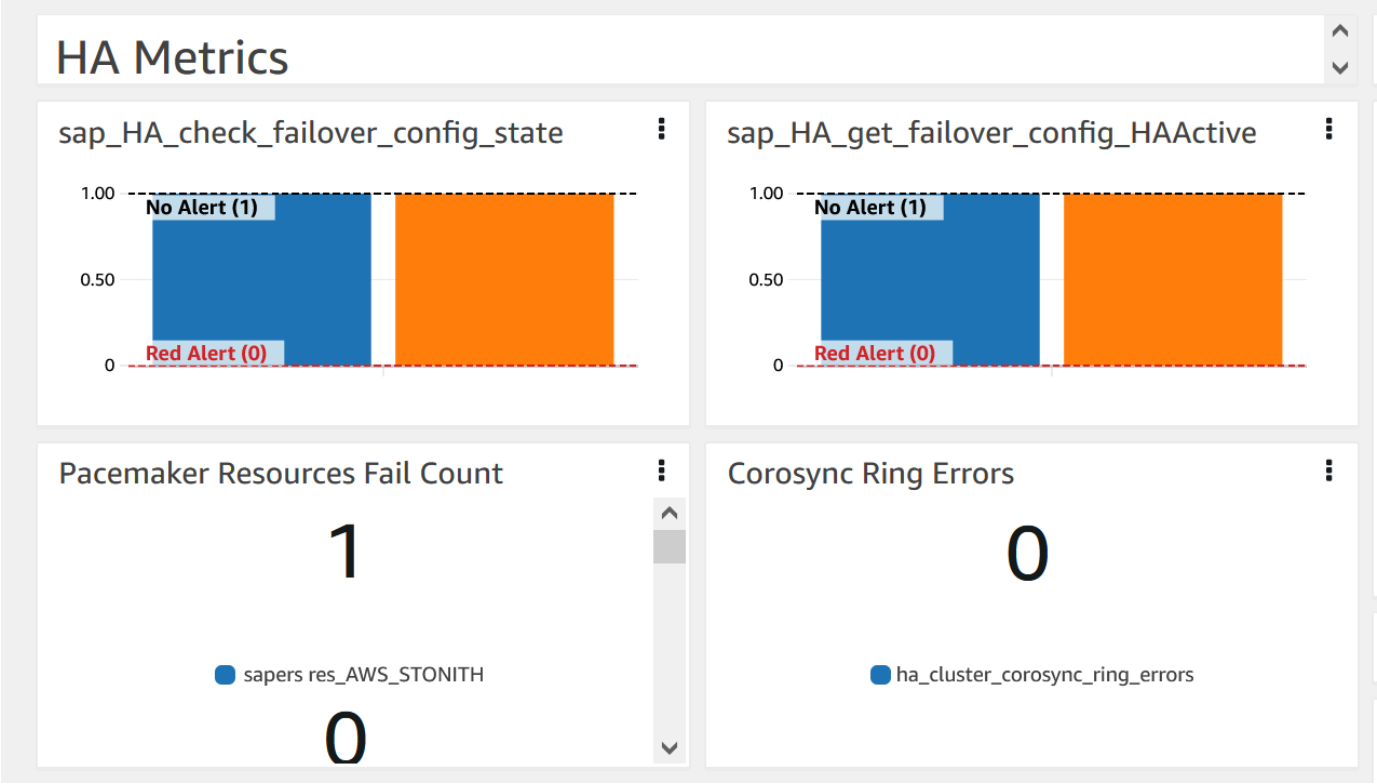


使用下列步驟來取得有關問題儀表板中顯示之指標和警示的詳細資訊。

取得指標和警示的詳細資訊

1. 開啟 CloudWatch 主控台。
2. 在左側導覽窗格中，選取 Insights 下的 Application Insights。然後，選擇 List view (清單檢視) 索引標籤，然後選擇您的應用程式。
3. 選取 Components (元件) 索引標籤。然後，選取您要取得其詳細資訊的 SAP NetWeaver 元件。

   下列範例會顯示 HA Metrics (HA 指標) 區段，其中包含顯示在問題儀表板中的 `ha_cluster_pacemaker_fail_count` 指標。



解決方案

Application Insights 會每小時監控偵測到的問題。如果 SAP 記錄檔中沒有新的相關 NetWeaver 記錄項目，則會將較舊的記錄項目視為已解析。您必須修正與此問題相關的任何錯誤情況。

針對`sap_alerts_Shortdumps`警示，您必須使用交易代碼導覽至 CCMS 警示`RZ20 # R3Abap # Shortdumps`來解決 SAP NetWeaver 系統中的警示。如需有關 CCMS 提醒的詳細資訊，請參閱 SAP 網站。解決 Shortdumps 樹中的所有 CCMS 提醒。在 SAP NetWeaver 系統中解決所有警示之後，CloudWatch 不再報告處於警示狀態的量度。

解決所有 CloudWatch 記錄檔錯誤和警示後，Application Insights 會停止偵測錯誤，並在一小時內自動解決問題。我們建議您解決所有日誌錯誤狀況和警示，以便在問題儀表板上找到最新的問題。在下列範例中，SAP Netweaver 高可用性排入佇列複寫問題已解決。

Severity	Problem summary	Source	Start time	Status
High	SAP Availability: Enqueue Replication	netweavercomponent-HE2-2b8c0...	2022-12-08T20:01:43Z	Resolved

疑難排解 SAP 應用程式見解 NetWeaver

本節提供的步驟可協助您解決 Application Insights 儀表板所傳回的常見錯誤。

無法新增超過 60 個監控指標

傳回錯誤：`Component cannot have more than 60 monitored metrics.`

根本原因：`The current metric limit is 60 monitor metrics per component.`

解決方案：移除不需要遵守限制的指標。

SAP 指標在加入程序之後不會顯示在儀表板上

根本原因：元件儀表板使用五分鐘的指標期間來彙總資料點。

解決方案：五分鐘後，所有指標都應顯示在儀表板上。

SAP 指標和警示不會顯示在儀表板上

使用下列步驟來確定為什麼 SAP 指標和警示在加入程序之後不會顯示在儀表板上。

識別有指標與警示的問題

1. 開啟 CloudWatch 主控台。
2. 在左側導覽窗格中，選取 Insights 下的 Application Insights。然後，選擇 List view (清單檢視) 索引標籤，然後選擇您的應用程式。
3. 選擇 Configuration history (組態歷史紀錄) 索引標籤。
4. 如果您看到遺失的指標資料點，請檢查與 `prometheus-sap_host_exporter` 相關的錯誤。
5. 如果在上一個步驟中找不到錯誤，Connect to your Linux instance (連線到您的 Linux 執行個體)。對於高可用性部署，請連線到主叢集 Amazon EC2 執行個體。
6. 在您的執行個體中，請使用下列命令確認匯出程式是否正在執行。預設連接埠為 9680。如果您使用其他連接埠，請使用您正在使用的連接埠取代 9680。

   ```
   curl localhost:9680/metrics
   ```

   如果沒有傳回任何資料，則匯出程式無法啟動。
7. 若要在接下來的兩個步驟中尋找要用WORKLOAD_SHORT_NAME於的正確命名慣例，請執行下列命令。

> **Note**
>
> 應用程式深入解析會根據執行中的工作負載WORKLOAD_SHORT_NAME，在服務名稱中新增尾碼。 NetWeaver 分散式、標準和高可用性部署的簡短名稱為SAP_NWDSAP_NWS、和SAP_NWH。

```
sudo systemctl | grep exporter
```

8. 若要檢查匯出程式服務記錄檔中是否有錯誤，請執行下列命令：

```
sudo journalctl -e --unit=prometheus-sap_host_exporter_WORKLOAD_SHORT_NAME.service
```

9. 若要檢查匯出程式管理員服務記錄檔中是否有錯誤，請執行下列命令：

```
sudo journalctl -e --unit=prometheus-
sap_host_exporter_manager_WORKLOAD_SHORT_NAME.service
```

> **Note**
>
> 此服務應該始終啟動並運行。

如果此命令沒有傳回錯誤，則繼續下一個步驟。

10. 若要手動啟動匯出程式，請執行下列命令。然後，檢查匯出程式輸出。

```
sudo /opt/aws/sap_host_exporter/sap_host_exporter
```

您可以在檢查錯誤後結束匯出程式程序。

根本原因：此問題有數個可能的原因。常見的原因是匯出程式無法連線到其中一個應用程式伺服器執行個體。

解決方案

請使用下列步驟將匯出程式連線至應用程式伺服器執行個體。您需要驗證 SAP 應用程式執行個體是否正在執行，並使用 SAPControl 連線至執行個體。

將匯出程式連線至應用程式伺服器執行個體

1. 在您的 Amazon EC2 執行個體中，執行下列命令以驗證 SAP 應用程式是否正在執行。

```
sapcontrol -nr <App_InstNo> -function GetProcessList
```

2. 您必須建立有效的 SAPControl 連線。如果 SAPControl 連線無法運作，請在相關 SAP 應用程式執行個體上找出問題的根本原因。
3. 若要在修正 SAPControl 連線問題後手動啟動匯出程式，請執行下列命令：

```
sudo systemctl start prometheus-sap_host_exporter.service
```

4. 如果您無法解決 SAPControl 連線問題，請使用下列程序作為暫時修正。
   a. 開啟 AWS Systems Manager 主控台。
   b. 在左側導覽窗格中，選擇 State Manager。
   c. 在「關聯」下，搜尋 SAP NetWeaver 系統的關聯。

```
Association Name: Equal: AWS-ApplicationInsights-
SSMSAPHostExporterAssociationForCUSTOMSAPNW<SID>-1
```

   d. 選取 Association ID (關聯 ID)。
   e. 選擇 Parameters (參數) 索引標籤，然後從 additionalArguments 中移除應用程式伺服器編號。
   f. 選擇 Apply association now (立即套用關聯)。

> **Note**
>
> 這是暫時的修正程式。如果對元件的監視組態進行了更新，則會重新新增執行個體。

檢視 Amazon CloudWatch 應用程式深入解析所偵測到的問題並

本節中的主題提供 Application Insights 所顯示之已偵測問題和洞察的詳細資訊。它也會針對您的帳戶或組態中所偵測到的問題，提供建議的解決方案。

故障診斷主題

- CloudWatch 主控台概觀

- Application Insights 問題摘要頁面
- CloudWatch 用戶端合併衝突失敗
- 未建立警報
- 意見回饋
- 組態錯誤

CloudWatch 主控台概觀

您可以在主控台概觀頁面的「應用程式深入解析」窗格下找到影響受監CloudWatch 控應用 CloudWatch 程式的問題概觀。如需詳細資訊，請參閱 開始使 CloudWatch 用 Amazon 應用程式深入解析。

「 CloudWatch 應用程式見解」概觀窗格會顯示下列項目

- 偵測到的問題嚴重性：高/中/低
- 問題的簡短摘要
- 問題來源
- 問題開始的時間
- 問題的解決狀態
- 受影響的資源群組

若要檢視特定問題的詳細資訊，請在 Problem Summary (問題摘要) 下，選取問題描述。詳細資訊儀表板會顯示問題和相關指標異常的洞見以及日誌錯誤的程式碼片段。您可以選取它是否有用，提供對洞察相關的意見回饋。

如果偵測到未設定的新資源，問題摘要描述會將您移至 Edit configuration (編輯組態) 精靈，以設定新的資源。您可以選擇詳細資訊儀表板右上角的 View/edit configuration (檢視/編輯組態)，檢視或編輯您的資源群組組態。

若要返回概觀，請選擇 [返回概觀]，該概觀位於 [CloudWatch 應用程式見解詳細資訊] 標題旁邊。

Application Insights 問題摘要頁面

Application Insights 問題摘要頁面

CloudWatch 「應用程式見解」會在問題摘要頁面上提供下列有關偵測到問題的資訊

- 問題的簡短摘要
- 問題的開始時間和日期
- 問題嚴重性：高/中/低
- 偵測到的問題狀態：正在進行/已解決
- 洞見：對所偵測問題及可能根本原因自動產生的洞見
- 洞察意見回饋：您提供的有關 CloudWatch 應用程式見解產生之見解實用性的回饋
- 相關觀察：詳細檢視與跨各種應用程式元件問題有關的相關日誌指標異常和錯誤程式碼片段

CloudWatch 用戶端合併衝突失敗

CloudWatch 應用程式深入解析會在客戶執行個體上安裝和設定 CloudWatch 代理程式。這包括建立具有測量結果或記錄檔組態的 CloudWatch 代理程式組態檔。如果客戶的執行個體已經有 CloudWatch 代理程式組態檔案，並且為相同的量度或記錄檔定義了不同的組態，則可能會發生合併衝突。若要解決合併衝突，請使用下列步驟：

1. 識別系統上的 CloudWatch 代理程式組態檔。如需檔案位置的詳細資訊，請參閱 CloudWatch 代理程式檔案和位置。
2. 從現有的代理程式組態檔中移除您要在「應用程 CloudWatch 式深入解析」中使用的資源組態。如果您只想使用應用程式深入解析組態，請刪除現有的 CloudWatch 代理程式組態檔案。

未建立警報

對於某些指標，Application Insights 會根據指標的先前資料點預測警示閾值。若要啟用此預測，必須符合下列條件。

- 最近的資料點 – 過去 24 小時內至少必須有 100 個資料點。資料點不需要連續，而且可以分散在 24 小時的時間範圍內。
- 歷史資料 – 從目前日期前 15 天到目前日期前 1 天，必須至少有 100 個資料點跨越時間範圍。資料點不需要連續，而且可以分散在 15 天的時間範圍內。

> **Note**
>
> 對於某些指標，Application Insights 會延遲警示的建立，直到符合前述條件為止。在此情況下，您會收到組態歷史記錄事件，說明指標缺少足夠的資料點來建立警示閾值。

意見回饋

意見回饋

您可以指出針對所偵測問題自動產生的洞見是否有用，提供意見回饋。您對洞見的意見回饋以及您的應用程式診斷 (指標異常和日誌例外狀況)，都會用來提升未來對類似問題的偵測。

組態錯誤

CloudWatch 「應用程式深入解析」會使用您的組態來建立元件的監控電話測試。當 Application Insights 偵測到您的帳戶或您的組態發生問題時，應用程式摘要的 Remarks (備註) 欄位中會提供如何解決應用程式組態問題的資訊。

下表顯示特定備註的建議解決方案。

備註	建議的解決方案	其他備註
的配額 CloudFormation 已達到。	應用程式洞察會為每個應用程式建立一個 CloudFormation 堆疊，以管 CloudWatch 理所有應用程式元件的代理程式安裝 默認情況下，每個 AWS 帳戶可以有 2000 個堆棧。請參閱 AWS CloudFormation 限制。若要解決此問題，請提高 CloudFormation 堆疊的限制。	N/A
下列執行個體上沒有 SSM 執行個體角色。	若要讓應用程式深入解析能夠在應用程式執行個體上安裝和設定 CloudWatch 代理程式，AmazonSSM ManagedInstanceCore 和 CloudWatchAgentServerPolicy 原則必須附加至執行個體角色。	應用程式深入解析會呼叫 SSM DescribeInstanceInformation API，以取得具有 SSM 權限的執行個體清單。將角色連接至執行個體後，SSM 需要一些時間才能在 DescribeInstanceInformation 結果中包含執行個體。在 SSM 將執行個體納入結果前，應用程式會一直顯示 NO_SSM_INSTANCE_ROLE 錯誤。

備註	建議的解決方案	其他備註
您可能需要設定新的元件。	Application Insights 偵測到應用程式資源群組有新元件。若要解決這個問題，請相應設定新的元件。	N/A

Amazon CloudWatch 應用程式深入解析支援的日誌和指標

下列清單顯示 Amazon CloudWatch 應用程式深入解析的支援日誌和指標。

CloudWatch 應用程式深入解析支援下列記錄：

- Microsoft Internet Information Services (IIS) 日誌
- EC2 的 SQL Server 錯誤日誌
- 自訂的 .NET 應用程式日誌，例如 Log4Net
- Windows 事件日誌，包括 Windows 日誌 (系統、應用程式和安全性) 及應用程式和服務日誌
- Amazon CloudWatch 日誌 AWS Lambda
- EC2 上 RDS MySQL、Aurora MySQL 和 MySQL 的錯誤日誌和慢速日誌
- EC2 上的 PostgreSQL RDS 和 PostgreSQL 的 Postgresql 日誌
- Amazon CloudWatch 日誌 AWS Step Functions
- API Gateway REST API 階段的執行日誌和存取日誌 (JSON、CSV 和 XML，而非 CLF)
- Prometheus JMX Exporter 日誌 (EMF)
- Amazon RDS 上的 Oracle 和 Amazon EC2 上的 Oracle 的提醒日誌與接聽程式
- 容器記錄檔從 Amazon ECS 容器路由到 CloudWatch 使用日`awslogs`誌驅動程式。
- 容器記錄檔從 Amazon ECS 容器路由到 CloudWatch 使用FireLens 容器日誌路由器。
- 容器日誌將從 Amazon Amazon EC2 上執行的 Amazon EKS 或 Kubernetes 路由到 CloudWatch 使用具有容器見解的流暢位元或流暢日誌處理器。
- SAP HANA 追蹤和錯誤日誌
- HA Pacemaker 日誌
- SAP ASE 伺服器日誌
- SAP ASE 備份伺服器日誌
- SAP ASE 複寫伺服器日誌

- SAP ASE RMA 代理程式日誌
- SAP ASE 錯誤管理員日誌
- SAP NetWeaver 開發人員追蹤記錄
- 使用代理程式專用外掛程式的 Windows 處理程序的程序指標 CloudWatch
- 託管區域的公用 DNS 查詢日誌
- Amazon Route 53 Resolver DNS 查詢記錄

CloudWatch 應用程式深入解析支援下列記錄類別：

- 標準 — Amazon CloudWatch 應用程式深入解析需要使用CloudWatch 日誌標準日誌類別來設定日誌群組，才能啟用監控。

CloudWatch 應用程式洞察支援下列應用程式元件的指標：

- Amazon Elastic Compute Cloud (EC2)
  - CloudWatch 內建量度
  - CloudWatch 代理程式度量 (Windows 伺服器)
  - CloudWatch 代理程式程序測量結果 (Windows 伺服器
  - CloudWatch 用戶端測量結果 (Linux 伺服器)
- Elastic Block Store (EBS)
- Amazon Elastic File System (Amazon EFS)
- Elastic Load Balancer (ELB)
- 應用程式 ELB
- Amazon EC2 Auto Scaling 群組
- Amazon Simple Queue Server (SQS)
- Amazon Relational Database Service (RDS)
  - RDS 資料庫執行個體
  - RDS 資料庫叢集
- AWS Lambda 函數
- Amazon DynamoDB 資料表
- Amazon S3 儲存貯體
- AWS Step Functions

  - Execution-level
  - 活動
  - Lambda 函數
  - 服務整合
  - Step Functions API
- API Gateway REST API 階段
- SAP HANA
- SAP ASE
- 在 Amazon EC2 上的 SAP ASE 高可用性
- SAP NetWeaver
- HA 叢集
- Java
- Amazon Elastic Container Service (Amazon ECS)
  - CloudWatch 內建量度
  - Container Insights 指標
  - Container Insights Prometheus 指標
- 庫伯尼特斯 AWS
  - Container Insights 指標
  - Container Insights Prometheus 指標
- Amazon FSx
- Amazon VPC
- Amazon VPC NAT 閘道
- Amazon Route 53 運作狀態檢查
- Amazon Route 53 託管區域
- Amazon Route 53 Resolver endpoint
- AWS Network Firewall 規則群組
- AWS Network Firewall 規則群組關聯
- 具有指標資料點要求的指標
  - AWS/ApplicationELB
  - AW/ AutoScaling

- AWS/EC2
- Elastic Block Store (EBS)
- AWS/ELB
- AWS/RDS
- AWS/Lambda
- AWS/SQS
- AWS/CWAgent
- AWS/DynamoDB
- AWS/S3
- AWS/狀態
- AW/ ApiGateway
- AWS/SNS

- 建議的指標
- 效能計數器指標

Amazon Elastic Compute Cloud (EC2)

CloudWatch 應用程式深入解析支援下列指標：

指標

- CloudWatch 內建量度
- CloudWatch 代理程式度量 (Windows 伺服器)
- CloudWatch 代理程式程序測量結果 (Windows 伺服器
- CloudWatch 用戶端測量結果 (Linux 伺服器)

CloudWatch 內建量度

處理器 CreditBalance

處理器 CreditUsage

處理器 SurplusCreditBalance

處理器 SurplusCreditsCharged

CPUUtilization

DiskReadBytes

DiskReadOps

DiskWriteBytes

DiskWriteOps

EBS 百分比 ByteBalance

EBSIOBalance%

EBS ReadBytes

EBS ReadOps

EBS WriteBytes

EBS WriteOps

NetworkIn

NetworkOut

NetworkPacketsIn

NetworkPacketsOut

StatusCheckFailed

StatusCheckFailed實例 (_D

StatusCheckFailed_ 系統

CloudWatch 代理程式度量 (Windows 伺服器)

擲出的 .NET CLR 異常數

每秒擲出的 .NET CLR 異常數

每秒篩選的 .NET CLR 異常數

每秒最終的 .NET CLR 異常數

每秒擲出以捕捉深度的 .NET CLR 異常

CCW 的 .NET CLR Interop 數

Stub 的 .NET CLR Interop 數

每秒 TLB 匯出的 .NET CLR Interop 數

每秒 TLB 匯入的 .NET CLR Interop 數

封送處理的 .NET CLR Interop 數

Jit 的 .NET CLR Jit 時間百分比

.NET CLR Jit 標準 Jit 失敗

.NET CLR 正在載入 - 正在載入時間百分比

.NET CLR 載入失敗的載入速率

每秒 .NET CLR LocksAndThreads 爭用速率

每秒 .NET CLR LocksAndThreads 佇列長度

.NET CLR 記憶體數總遞交的位元組

.NET CLR 記憶體在 GC 中時間佔比

網路 CLR 網路平均佇列時間 4.0.0.0 HttpWebRequest

無線 CLR 網路連線 4. HttpWebRequests 0.0.0 中止/秒

無線 CLR 網路連線 4. HttpWebRequests 0.0.0 失敗/秒

.NET CLR 網路連線 4. HttpWebRequests 0.0.0 已佇列/秒

APP_POL_WAS 總工作者處理序 Ping 失敗

ASP.NET 應用程式重新啟動

ASP.NET 應用程式 % 受管處理器時間 (估計)

ASP.NET 應用程式錯誤總計/秒

ASP.NET 應用程式執行期間未處理錯誤/秒

應用程式佇列中的 ASP.NET 應用程式請求

ASP.NET 應用程式請求/秒

ASP.NET 請求等待時間

ASP.NET 請求已排入佇列

HTTP 服務要求佇列 CurrentQueueSize

LogicalDisk % 可用空間

記憶體 % 使用中的認可位元組

記憶體可用的 MB 數

記憶體分頁/秒

網路介面位元組總數/秒

分頁檔用量 %

PhysicalDisk % 磁碟時間

PhysicalDisk 平均 磁碟佇列長度

PhysicalDisk 平均 Disk sec/Read

PhysicalDisk 平均 Disk sec/Write

PhysicalDisk 磁碟讀取位元組/秒

PhysicalDisk 磁碟讀取次數/秒

PhysicalDisk 磁碟寫入位元組/秒

PhysicalDisk 磁碟寫入次數/秒

處理器 % 閒置時間

處理器 % 中斷時間

處理器 % 處理器時間

處理器 % 使用者時間

SQLServer:Access Methods Forwarded Records/sec

SQLServer:Access Methods Full Scans/sec

SQLServer:Access Methods Page Splits/sec

SQLServer:Buffer Manager 緩衝快取命中率

SQLServer:Buffer Manager 頁面的預期壽命

SQLServer:General Statistics Processes blocked

SQLServer：一般統計資料使用者連線

SQLServer：閂鎖平均閂鎖等待時間 (毫秒)

SQLServer：鎖定平均等待時間 (毫秒)

SQLServer：鎖定鎖定逾時/秒

SQLServer：鎖定鎖定等待/秒

SQLServer：鎖定死鎖數目/秒

SQLServer：記憶體管理員記憶體授權待定

SQLServer:SQL Statistics 每秒批次要求數

SQLServer:SQL Statistics SQL Compilations/sec

SQLServer:SQL Statistics SQL Re-Compilations/sec

系統處理器佇列長度

已建立 TCPv4 連線

已建立 TCPv6 連線

W3SVC_W3WP 檔案饋取排清

W3SVC_W3WP 檔案快取遺漏

W3SVC_W3WP 請求/秒

W3SVC_W3WP URI 快取排清

W3SVC_W3WP URI 快取遺漏

接收的 Web 服務位元組/秒

傳送的 Web 服務位元組/秒

Web 服務連線嘗試/秒

Web 服務目前連線數目

Web 服務收到請求/秒

Web 服務 POST 請求/秒

接收的位元組/秒

正常訊息佇列長度/秒

紧急訊息佇列長度/秒

重新連線計數

未確認的訊息佇列長度/秒

未處理的訊息

傳送的訊息/秒

資料庫更新訊息/秒

更新訊息/秒

排清數/秒

儲存的加密檢查點/秒

還原的加密檢查點/秒

還原的登錄檢查點/秒

儲存的登錄檢查點/秒

叢集 API 呼叫/秒

資源 API 呼叫/秒

叢集處理/秒

資源處理/秒

CloudWatch 代理程式程序測量結果 (Windows 伺服器

程序指標是使用CloudWatch 代理程式 procstat 外掛程式收集的。只有執行 Windows 工作負載的 Amazon EC2 執行個體才支援程式指標。

procstat cpu_time_system

procstat cpu_time_user

procstat cpu_usage

procstat memory_rss

procstat memory_vms

procstat read_bytes

procstat write_bytes

.procstat read_count

procstat write_count

CloudWatch 用戶端測量結果 (Linux 伺服器)

cpu_time_active

cpu_time_guest

cpu_time_guest_nice

cpu_time_idle

cpu_time_iowait

cpu_time_irq

cpu_time_nice

cpu_time_softirq

cpu_time_steal

cpu_time_system

cpu_time_user

cpu_usage_active

cpu_usage_guest

cpu_usage_guest_nice

cpu_usage_idle

cpu_usage_iowait

cpu_usage_irq

cpu_usage_nice

cpu_usage_softirq

cpu_usage_steal

cpu_usage_system

cpu_usage_user

disk_free

disk_inodes_free

disk_inodes_used

disk_used

disk_used_percent

diskio_io_time

diskio_iops_in_progress

diskio_read_bytes

diskio_read_time

diskio_reads

diskio_write_bytes

diskio_write_time

diskio_writes

mem_active

mem_available

mem_available_percent

mem_buffered

mem_cached

mem_free

mem_inactive

mem_used

mem_used_percent

net_bytes_recv

net_bytes_sent

net_drop_in

net_drop_out

net_err_in

net_err_out

net_packets_recv

net_packets_sent

netstat_tcp_close

netstat_tcp_close_wait

netstat_tcp_closing

netstat_tcp_established

netstat_tcp_fin_wait1

netstat_tcp_fin_wait2

netstat_tcp_last_ack

netstat_tcp_listen

netstat_tcp_none

netstat_tcp_syn_recv

netstat_tcp_syn_sent

netstat_tcp_time_wait

netstat_udp_socket

processes_blocked

processes_dead

processes_idle

processes_paging

processes_running

processes_sleeping

processes_stopped

processes_total

processes_total_threads

processes_wait

processes_zombies

swap_free

swap_used

swap_used_percent

Elastic Block Store (EBS)

CloudWatch 應用程式深入解析支援下列指標：

VolumeReadBytes

VolumeWriteBytes

VolumeReadOps

VolumeWriteOps

VolumeTotalReadTime

VolumeTotalWriteTime

VolumeIdleTime

VolumeQueueLength

VolumeThroughputPercentage

VolumeConsumedReadWriteOps

BurstBalance

Amazon Elastic File System (Amazon EFS)

CloudWatch 應用程式深入解析支援下列指標：

BurstCreditBalance

PercentIOLimit

PermittedThroughput

MeteredIOBytes

TotalIOBytes

DataWriteIOB

DataReadIOB

MetadataIOBytes

ClientConnections

TimeSinceLastSync

StorageBytes

輸送量

PercentageOfPermittedThroughputUtilization

ThroughputIOPS

PercentThroughputDataReadIOB

PercentThroughputDataWriteIOB

PercentageOfIOPS IOB DataRead

PercentageOfIOPS IOB DataWrite

AverageDataReadIO BytesSize

AverageDataWriteIO BytesSize

Elastic Load Balancer (ELB)

CloudWatch 應用程式深入解析支援下列指標：

估計 ActiveConnectionCount

EstimatedALBConsumedLCUs

估計 NewConnectionCount

EstimatedProcessedBytes

HTTPCode_Backend_4XX

HTTPCode_Backend_5XX

HealthyHostCount

RequestCount

UnHealthyHostCount

應用程式 ELB

CloudWatch 應用程式深入解析支援下列指標：

估計 ActiveConnectionCount

EstimatedALBConsumedLCUs

估計 NewConnectionCount

EstimatedProcessedBytes

HTTPCode_Backend_4XX

HTTPCode_Backend_5XX

HealthyHostCount

Latency (延遲)

RequestCount

SurgeQueueLength

UnHealthyHostCount

Amazon EC2 Auto Scaling 群組

CloudWatch 應用程式深入解析支援下列指標：

處理器 CreditBalance

處理器 CreditUsage

處理器 SurplusCreditBalance

處理器 SurplusCreditsCharged

CPUUtilization

DiskReadBytes

DiskReadOps

DiskWriteBytes

DiskWriteOps

EBS 百分比 ByteBalance

EBSIOBalance%

EBS ReadBytes

EBS ReadOps

EBS WriteBytes

EBS WriteOps

NetworkIn

NetworkOut

NetworkPacketsIn

NetworkPacketsOut

StatusCheckFailed

StatusCheckFailed實例 (_D

StatusCheckFailed_ 系統

Amazon Simple Queue Server (SQS)

CloudWatch 應用程式深入解析支援下列指標：

ApproximateAgeOfOldestMessage

ApproximateNumberOfMessagesDelayed

ApproximateNumberOfMessagesNotVisible

ApproximateNumberOfMessagesVisible

NumberOfEmptyReceives

NumberOfMessagesDeleted

NumberOfMessagesReceived

NumberOfMessagesSent

Amazon Relational Database Service (RDS)

CloudWatch 應用程式深入解析支援下列指標：

指標

- RDS 資料庫執行個體
- RDS 資料庫叢集

RDS 資料庫執行個體

BurstBalance

處理器 CreditBalance

CPUUtilization

DatabaseConnections

DiskQueueDepth

失敗的 SQL ServerAgentJobsCount

FreeStorageSpace

FreeableMemory

NetworkReceiveThroughput

NetworkTransmitThroughput

ReadIOPS

ReadLatency

ReadThroughput

WriteIOPS

WriteLatency

WriteThroughput

RDS 資料庫叢集

ActiveTransactions

AuroraBinlogReplicaLag

AuroraReplicaLag

BackupRetentionPeriodStorageUsed

BinLogDiskUsage

BlockedTransactions

BufferCacheHitRatio

CPUUtilization

CommitLatency

CommitThroughput

DDLLatency

DDLThroughput

DMLLatency

DMLThroughput

DatabaseConnections

鎖死

DeleteLatency

DeleteThroughput

EngineUptime

FreeLocalStorage

FreeableMemory

InsertLatency

InsertThroughput

LoginFailures

NetworkReceiveThroughput

NetworkThroughput

NetworkTransmitThroughput

查詢

ResultSetCacheHitRatio

SelectLatency

SelectThroughput

SnapshotStorageUsed

TotalBackupStorageBilled

UpdateLatency

UpdateThroughput

VolumeBytesUsed

VolumeReadIOP

VolumeWriteIOP

AWS Lambda 函數

CloudWatch 應用程式深入解析支援下列指標：

錯誤

DeadLetterErrors

持續時間

限流

IteratorAge

ProvisionedConcurrencySpilloverInvocations

Amazon DynamoDB 資料表

CloudWatch 應用程式深入解析支援下列指標：

SystemErrors

UserErrors

ConsumedReadCapacityUnits

ConsumedWriteCapacityUnits

ReadThrottleEvents

WriteThrottleEvents

TimeToLiveDeletedItemCount

ConditionalCheckFailedRequests

TransactionConflict

ReturnedRecordsCount

PendingReplicationCount

ReplicationLatency

Amazon S3 儲存貯體

CloudWatch 應用程式深入解析支援下列指標：

ReplicationLatency

BytesPendingReplication

OperationsPendingReplication

4xxErrors

5xxErrors

AllRequests

GetRequests

PutRequests

DeleteRequests

HeadRequests

PostRequests

SelectRequests

ListRequests

SelectScannedBytes

SelectReturnedBytes

FirstByteLatency

TotalRequestLatency

BytesDownloaded

BytesUploaded

AWS Step Functions

CloudWatch 應用程式深入解析支援下列指標：

指標

- Execution-level
- 活動
- Lambda 函數
- 服務整合
- Step Functions API

Execution-level

ExecutionTime

ExecutionThrottled

ExecutionsFailed

ExecutionsTimedOut

ExecutionsAborted

ExecutionsSucceeded

ExecutionsStarted

活動

ActivityRunTime

ActivityScheduleTime

ActivityTime

ActivitiesFailed

ActivitiesHeartbeatTimedOut

ActivitiesTimedOut

ActivitiesScheduled

ActivitiesSucceeded

ActivitiesStarted

Lambda 函數

LambdaFunctionRunTime

LambdaFunctionScheduleTime

LambdaFunctionTime

LambdaFunctionsFailed

LambdaFunctionsTimedOut

LambdaFunctionsScheduled

LambdaFunctionsSucceeded

LambdaFunctionsStarted

服務整合

ServiceIntegrationRunTime

ServiceIntegrationScheduleTime

ServiceIntegrationTime

ServiceIntegrationsFailed

ServiceIntegrationsTimedOut

ServiceIntegrationsScheduled

ServiceIntegrationsSucceeded

ServiceIntegrationsStarted

Step Functions API

ThrottledEvents

ProvisionedBucketSize

ProvisionedRefillRate

ConsumedCapacity

API Gateway REST API 階段

CloudWatch 應用程式深入解析支援下列指標：

4XXError

5XXError

IntegrationLatency

Latency (延遲)

CacheHitCount

CacheMissCount

SAP HANA

> **Note**
>
> CloudWatch 應用程式洞見僅支援單一 SID HANA 環境。如果連接了多個 HANA SID，則僅為偵測到的第一個 SID 設定監控。

CloudWatch 應用程式深入解析支援下列指標：

hanadb_every_service_started_status

hanadb_daemon_service_started_status

hanadb_preprocessor_service_started_status

hanadb_webdispatcher_service_started_status

hanadb_compileserver_service_started_status

hanadb_nameserver_service_started_status

hanadb_server_startup_time_variations_seconds

hanadb_level_5_alerts_count

hanadb_level_4_alerts_count

hanadb_out_of_memory_events_count

hanadb_max_trigger_read_ratio_percent

hanadb_max_trigger_write_ratio_percent

hanadb_log_switch_wait_ratio_percent

hanadb_log_switch_race_ratio_percent

hanadb_time_since_last_savepoint_seconds

hanadb_disk_usage_highlevel_percent

hanadb_max_converter_page_number_count

hanadb_long_running_savepoints_count

hanadb_failed_io_reads_count

hanadb_failed_io_writes_count

hanadb_disk_data_unused_percent

hanadb_current_allocation_limit_used_percent

hanadb_table_allocation_limit_used_percent

hanadb_host_total_physical_memory_mb

hanadb_host_physical_memory_used_mb

hanadb_host_physical_memory_free_mb

hanadb_swap_memory_free_mb

hanadb_swap_memory_used_mb

hanadb_host_allocation_limit_mb

hanadb_host_total_memory_used_mb

hanadb_host_total_peak_memory_used_mb

hanadb_host_total_allocation_limit_mb

hanadb_host_code_size_mb

hanadb_host_shared_memory_allocation_mb

hanadb_cpu_usage_percent

hanadb_cpu_user_percent

hanadb_cpu_system_percent

hanadb_cpu_waitio_percent

hanadb_cpu_busy_percent

hanadb_cpu_idle_percent

hanadb_long_delta_merge_count

hanadb_unsuccessful_delta_merge_count

hanadb_successful_delta_merge_count

hanadb_row_store_allocated_size_mb

hanadb_row_store_free_size_mb

hanadb_row_store_used_size_mb

hanadb_temporary_tables_count

hanadb_large_non_compressed_tables_count

hanadb_total_non_compressed_tables_count

hanadb_longest_running_job_seconds

hanadb_average_commit_time_milliseconds

hanadb_suspended_sql_statements_count

hanadb_plan_cache_hit_ratio_percent

hanadb_plan_cache_lookup_count

hanadb_plan_cache_hit_count

hanadb_plan_cache_total_execution_microseconds

hanadb_plan_cache_cursor_duration_microseconds

hanadb_plan_cache_preparation_microseconds

hanadb_plan_cache_evicted_count

hanadb_plan_cache_evicted_microseconds

hanadb_plan_cache_evicted_preparation_count

hanadb_plan_cache_evicted_execution_count

hanadb_plan_cache_evicted_preparation_microseconds

hanadb_plan_cache_evicted_cursor_duration_microseconds

hanadb_plan_cache_evicted_total_execution_microseconds

hanadb_plan_cache_evicted_plan_size_mb

hanadb_plan_cache_count

hanadb_plan_cache_preparation_count

hanadb_plan_cache_execution_count

hanadb_network_collision_rate

hanadb_network_receive_rate

hanadb_network_transmit_rate

hanadb_network_packet_receive_rate

hanadb_network_packet_transmit_rate

hanadb_network_transmit_error_rate

hanadb_network_receive_error_rate

hanadb_time_until_license_expires_days

hanadb_is_license_valid_status

hanadb_local_running_connections_count

hanadb_local_idle_connections_count

hanadb_remote_running_connections_count

hanadb_remote_idle_connections_count

hanadb_last_full_data_backup_age_days

hanadb_last_data_backup_age_days

hanadb_last_log_backup_age_hours

hanadb_failed_data_backup_past_7_days_count

hanadb_failed_log_backup_past_7_days_count

hanadb_oldest_backup_in_catalog_age_days

hanadb_backup_catalog_size_mb

hanadb_hsr_replication_status

hanadb_hsr_log_shipping_delay_seconds

hanadb_hsr_secondary_failover_count

hanadb_hsr_secondary_reconnect_count

hanadb_hsr_async_buffer_used_mb

hanadb_hsr_secondary_active_status

hanadb_handle_count

hanadb_ping_time_milliseconds

hanadb_connection_count

hanadb_internal_connection_count

hanadb_external_connection_count

hanadb_idle_connection_count

hanadb_transaction_count

hanadb_internal_transaction_count

hanadb_external_transaction_count

hanadb_user_transaction_count

hanadb_blocked_transaction_count

hanadb_statement_count

hanadb_active_commit_id_range_count

hanadb_mvcc_version_count

hanadb_pending_session_count

hanadb_record_lock_count

hanadb_read_count

hanadb_write_count

hanadb_merge_count

hanadb_unload_count

hanadb_active_thread_count

hanadb_waiting_thread_count

hanadb_total_thread_count

hanadb_active_sql_executor_count

hanadb_waiting_sql_executor_count

hanadb_total_sql_executor_count

hanadb_data_write_size_mb

hanadb_data_write_time_milliseconds

hanadb_log_write_size_mb

hanadb_log_write_time_milliseconds

hanadb_data_read_size_mb

hanadb_data_read_time_milliseconds

hanadb_log_read_size_mb

hanadb_log_read_time_milliseconds

hanadb_data_backup_write_size_mb

hanadb_data_backup_write_time_milliseconds

hanadb_log_backup_write_size_mb

hanadb_log_backup_write_time_milliseconds

hanadb_mutex_collision_count

hanadb_read_write_lock_collision_count

hanadb_admission_control_admit_count

hanadb_admission_control_reject_count

hanadb_admission_control_queue_size_mb

hanadb_admission_control_wait_time_milliseconds

SAP ASE

CloudWatch 應用程式深入解析支援下列指標：

asedb_database_availability

asedb_trunc_log_on_chkpt_enabled

asedb_last_db_backup_age_in_days

asedb_last_transaction_log_backup_age_in_hours

asedb_suspected_database

asedb_db_space_usage_percent

asedb_db_log_space_usage_percent

asedb_locked_login

asedb_has_mixed_log_and_data

asedb_runtime_for_open_transactions

asedb_data_cache_hit_ratio

asedb_data_cache_usage

asedb_sql_cache_hit_ratio

asedb_cache_usage

asedb_run_queue_length

asedb_number_of_rollbacks

asedb_number_of_commits

asedb_number_of_transactions

asedb_outstanding_disk_io

asedb_percent_io_busy

asedb_percent_system_busy

asedb_percent_locks_active

asedb_scheduled_jobs_failed_percent

asedb_user_connections_percent

asedb_query_logical_reads

asedb_query_physical_reads

asedb_query_cpu_time

asedb_query_memory_usage

在 Amazon EC2 上的 SAP ASE 高可用性

CloudWatch 應用程式深入解析支援下列指標：

asedb_ha_replication_state

asedb_ha_replication_mode

asedb_ha_replication_latency_in_minutes

SAP NetWeaver

CloudWatch 應用程式深入解析支援下列指標：

指標	描述
皂警報 ResponseTime	來自 CCMS (RZ20) > R3 服務 > 對話 > 對話 > 的 SAP 回應時間警示。ResponseTime
皂警報 ResponseTimeDialog	來自 CCMS (RZ20) > R3 服務 > 對話 > 對話 > 的 SAP 回應時間對話方塊警示。 ResponseTimeDialog
皂液警報 _ RFC ResponseTimeDialog	來自 CCMS (RZ20) > R3 服務 > 對話方塊 > RFC 的 SAP 回應時間警示。ResponseTimeDialog
SAP_ 警報 RequestTime	來自 CCMS (RZ20) >R3 服務 > 對話 > 資料庫的 SAP 回應時間警示。RequestTime
皂警報 FrontendResponseTime	來自 CCMS (RZ20) >「R3 服務」>「對話方塊」的 SAP 回應時間警示。FrontEndResponseTime
sap_alerts_Database	SAP 系統已記錄與資料庫相關的錯誤。來自 SM21 或 CCMS (RZ20)>R3Syslog>Database 的提醒。
皂警報 QueueTime	來自 CCMS (RZ20) >「R3 服務」>「對話」的 SAP 佇列時間警示。QueueTime

指標	描述
皂警報 AbortedJobs	SAP 系統中失敗的背景作業。來自（RZ20）> R3 服務 > 背景 > 的警報。AbortedJobs
皂警報 BasisSystem	SAP 系統記錄了系統層級錯誤。來自 SM21 或中央管理系統 (RZ20) > 系統管理系統的警示。BasisSystem
sap_alerts_Security	SAP 系統記錄了與安全相關的訊息。來自 SM21 或 CCMS (RZ20)>R3Syslog>Security 的提醒。
sap_alerts_System	SAP 系統記錄了與安全或稽核相關訊息。來自 SM21 或 CCMS (RZ20)>Security>System 的提醒。
皂警報 LongRunners	您的 SAP 系統中有長時間執行的程序。來自 CCMS (RZ20) 的警示 > R3 服務 > 對話方塊 >。LongRunners
皂警報 SqlError	有 SAP 資料庫用戶端層錯誤日誌。來自中央管理系統 (RZ20) 的警示 > > DatabaseClient。AbapSql SqlError
sap_alerts_State	來自 CCMS (RZ20)>OS Collector>State 的提醒。
sap_alerts_Shortdumps	來自 ST22 和 CCMS (RZ20)>R3Abap>Shortdumps 的 Shortdumps 提醒。
sap_alerts_Availability	針對來自 SM21、SM50、SM51、SM66 及 CCMS (RZ20) > 可用性的 SAP 應用程式伺服器執行處理的使用狀態警示。InstanceAsTask
sap_dispatcher_queue_high	SAPControl Web 服務函數 `GetQueueStatistic` 會提供發送器佇列高計數。
sap_dispatcher_queue_max	SAPControl Web 服務函數 `GetQueueStatistic` 會提供發送器佇列計數上限。

指標	描述
sap_dispatcher_queue_now	SAPControl Web 服務函數 GetQueueStatistic 會提供發送器佇列當前計數。
sap_dispatcher_queue_reads	SAPControl Web 服務函數 GetQueueStatistic 會提供發送器佇列讀取計數。
sap_dispatcher_queue_writes	SAPControl Web 服務函數 GetQueueStatistic 會提供發送器佇列寫入計數。
sap_enqueue_server_arguments_high	SAPControl Web 服務函數 EnqGetStatistic 會提供高排入佇列引數數量。
sap_enqueue_server_arguments_max	SAPControl Web 服務函數 EnqGetStatistic 會提供排入佇列引數數量上限。
sap_enqueue_server_arguments_now	SAPControl Web 服務函數 EnqGetStatistic 會提供當前排入佇列引數數量。
sap_enqueue_server_arguments_state	SAPControl Web 服務函數 EnqGetStatistic 會提供排入佇列引數狀態。
sap_enqueue_server_backup_requests	SAPControl Web 服務函數 EnqGetStatistic 會提供排入佇列備份請求數量。
sap_enqueue_server_cleanup_requests	SAPControl Web 服務函數 EnqGetStatistic 會提供排入佇列清除請求數量。
sap_enqueue_server_dequeue_all_requests	SAPControl Web 服務函數 EnqGetStatistic 會提供所有移出佇列請求數量。
sap_enqueue_server_dequeue_errors	SAPControl Web 服務函數 EnqGetStatistic 會提供移出佇列錯誤數量。
sap_enqueue_server_dequeue_requests	SAPControl Web 服務函數 EnqGetStatistic 會提供移出佇列請求數量。
sap_enqueue_server_enqueue_errors	SAPControl Web 服務函數 EnqGetStatistic 會提供排入佇列錯誤數量。

指標	描述
sap_enqueue_server_enqueue_rejects	SAPControl Web 服務函數 EnqGetStatistic 會提供排入佇列拒絕數量。
sap_enqueue_server_enqueue_requests	SAPControl Web 服務函數 EnqGetStatistic 會提供排入佇列請求數量。
sap_enqueue_server_lock_time	SAPControl Web 服務函數 EnqGetStatistic 會提供排入佇列鎖定時間。
sap_enqueue_server_lock_wait_time	SAPControl Web 服務函數 EnqGetStatistic 會提供排入佇列鎖定等待時間。
sap_enqueue_server_locks_high	SAPControl Web 服務函數 EnqGetStatistic 會提供高排入佇列鎖定數量。
sap_enqueue_server_locks_max	SAPControl Web 服務函數 EnqGetStatistic 會提供排入佇列鎖定數量上限。
sap_enqueue_server_locks_now	SAPControl Web 服務函數 EnqGetStatistic 會提供當前排入佇列鎖定數量。
sap_enqueue_server_locks_state	SAPControl Web 服務函數 EnqGetStatistic 會提供排入佇列鎖定狀態。
sap_enqueue_server_owner_high	SAPControl Web 服務函數 EnqGetStatistic 會提供高排入佇列擁有者。
sap_enqueue_server_owner_max	SAPControl Web 服務函數 EnqGetStatistic 會提供最大排入佇列擁有者。
sap_enqueue_server_owner_now	SAPControl Web 服務函數 EnqGetStatistic 會提供當前排入佇列擁有者。
sap_enqueue_server_owner_state	SAPControl Web 服務函數 EnqGetStatistic 會提供排入佇列擁有者狀態。
sap_enqueue_server_replication_state	SAPControl Web 服務函數 EnqGetStatistic 會提供排入佇列複寫狀態。

指標	描述
sap_enqueue_server_reporting_requests	SAPControl Web 服務函數 EnqGetStatistic 會提供報告請求狀態。
sap_enqueue_server_server_time	SAPControl Web 服務函數 EnqGetStatistic 會提供排入佇列伺服器時間。
sap_HA_check_failover_config_state	SAPControl Web 服務功能 HACheckFailoverConfig 會提供 SAP 高可用性狀態。
sap_HA_get_failover_config_HAActive	SAPControl Web 服務函數 HAGetFailoverConfig 會提供 SAP 高可用性叢集組態和狀態。
sap_start_service_processes	SAPControl Web 服務函數 GetProcessList 會提供顯示 disp+work、IGS、gwrd、icman、訊息伺服器和排入佇列伺服器程序的狀態。

HA 叢集

CloudWatch 應用程式深入解析支援下列指標：

ha_cluster_pacemaker_stonith_enabled

ha_cluster_corosync_quorate

hanadb_webdispatcher_service_started_status

ha_cluster_pacemaker_nodes

ha_cluster_corosync_ring_errors

ha_cluster_pacemaker_fail_count

Java

CloudWatch 應用程式深入解析支援下列指標：

java_lang_memory_heapmemoryusage_used

java_lang_memory_heapmemoryusage_committed

java_lang_operatingsystem_openfiledescriptorcount

java_lang_operatingsystem_maxfiledescriptorcount

java_lang_operatingsystem_freephysicalmemorysize

java_lang_operatingsystem_freeswapspacesize

java_lang_threading_threadcount

java_lang_threading_daemonthreadcount

java_lang_classloading_loadedclasscount

java_lang_garbagecollector_collectiontime_copy

java_lang_garbagecollector_collectiontime_ps_scavenge

java_lang_garbagecollector_collectiontime_parnew

java_lang_garbagecollector_collectiontime_marksweepcompact

java_lang_garbagecollector_collectiontime_ps_marksweep

java_lang_garbagecollector_collectiontime_concurrentmarksweep

java_lang_garbagecollector_collectiontime_g1_young_generation

java_lang_garbagecollector_collectiontime_g1_old_generation

java_lang_garbagecollector_collectiontime_g1_mixed_generation

java_lang_operatingsystem_committedvirtualmemorysize

Amazon Elastic Container Service (Amazon ECS)

CloudWatch 應用程式深入解析支援下列指標：

指標

- CloudWatch 內建量度
- Container Insights 指標

- Container Insights Prometheus 指標

CloudWatch 內建量度

CPUReservation

CPUUtilization

MemoryReservation

MemoryUtilization

GPUReservation

Container Insights 指標

ContainerInstanceCount

CpuUtilized

CpuReserved

DeploymentCount

DesiredTaskCount

MemoryUtilized

MemoryReserved

NetworkRxBytes

NetworkTxBytes

PendingTaskCount

RunningTaskCount

ServiceCount

StorageReadBytes

StorageWriteBytes

TaskCount

TaskSetCount

instance_cpu_limit

instance_cpu_reserved_capacity

instance_cpu_usage_total

instance_cpu_utilization

instance_filesystem_utilization

instance_memory_limit

instance_memory_reserved_capacity

instance_memory_utilization

instance_memory_working_set

執行個體網路總數 (_O) 位元組

instance_number_of_running_tasks

Container Insights Prometheus 指標

Javer JMX 指標

java_lang_memory_heapmemoryusage_used

java_lang_memory_heapmemoryusage_committed

java_lang_operatingsystem_openfiledescriptorcount

java_lang_operatingsystem_maxfiledescriptorcount

java_lang_operatingsystem_freephysicalmemorysize

java_lang_operatingsystem_freeswapspacesize

java_lang_threading_threadcount

java_lang_classloading_loadedclasscount

java_lang_threading_daemonthreadcount

java_lang_garbagecollector_collectiontime_copy

java_lang_garbagecollector_collectiontime_ps_scavenge

java_lang_garbagecollector_collectiontime_parnew

java_lang_garbagecollector_collectiontime_marksweepcompact

java_lang_garbagecollector_collectiontime_ps_marksweep

java_lang_garbagecollector_collectiontime_concurrentmarksweep

java_lang_garbagecollector_collectiontime_g1_young_generation

java_lang_garbagecollector_collectiontime_g1_old_generation

java_lang_garbagecollector_collectiontime_g1_mixed_generation

java_lang_operatingsystem_committedvirtualmemorysize

庫伯尼特斯 AWS

CloudWatch 應用程式深入解析支援下列指標：

指標

- Container Insights 指標
- Container Insights Prometheus 指標

Container Insights 指標

cluster_failed_node_count

cluster_node_count

namespace_number_of_running_pods

node_cpu_limit

node_cpu_reserved_capacity

node_cpu_usage_total

node_cpu_utilization

node_filesystem_utilization

node_memory_limit

node_memory_reserved_capacity

node_memory_utilization

node_memory_working_set

node_network_total_bytes

node_number_of_running_containers

node_number_of_running_pods

pod_cpu_reserved_capacity

pod_cpu_utilization

pod_cpu_utilization_over_pod_limit

pod_memory_reserved_capacity

pod_memory_utilization

pod_memory_utilization_over_pod_limit

pod_network_rx_bytes

pod_network_tx_bytes

service_number_of_running_pods

Container Insights Prometheus 指標

Javer JMX 指標

java_lang_memory_heapmemoryusage_used

java_lang_memory_heapmemoryusage_committed

java_lang_operatingsystem_openfiledescriptorcount

java_lang_operatingsystem_maxfiledescriptorcount

java_lang_operatingsystem_freephysicalmemorysize

java_lang_operatingsystem_freeswapspacesize

java_lang_threading_threadcount

java_lang_classloading_loadedclasscount

java_lang_threading_daemonthreadcount

java_lang_garbagecollector_collectiontime_copy

java_lang_garbagecollector_collectiontime_ps_scavenge

java_lang_garbagecollector_collectiontime_parnew

java_lang_garbagecollector_collectiontime_marksweepcompact

java_lang_garbagecollector_collectiontime_ps_marksweep

java_lang_garbagecollector_collectiontime_concurrentmarksweep

java_lang_garbagecollector_collectiontime_g1_young_generation

java_lang_garbagecollector_collectiontime_g1_old_generation

java_lang_garbagecollector_collectiontime_g1_mixed_generation

java_lang_operatingsystem_committedvirtualmemorysize

Amazon FSx

CloudWatch 應用程式深入解析支援下列指標：

DataReadBytes

DataWriteBytes

DataReadOperations

DataWriteOperations

MetadataOperations

FreeStorageCapacity

FreeDataStorageCapacity

LogicalDiskUsage

PhysicalDiskUsage

Amazon VPC

CloudWatch 應用程式深入解析支援下列指標：

NetworkAddressUsage

NetworkAddressUsagePeered

VPC FirewallQueryVolume

Amazon VPC NAT 閘道

CloudWatch 應用程式深入解析支援下列指標：

ErrorPortAllocation

IdleTimeoutCount

Amazon Route 53 運作狀態檢查

CloudWatch 應用程式深入解析支援下列指標：

ChildHealthCheckHealthyCount

ConnectionTime

HealthCheckPercentageHealthy

HealthCheckStatus

SSL HandshakeTime

TimeToFirstByte

Amazon Route 53 託管區域

CloudWatch 應用程式深入解析支援下列指標：

DNSQueries

DNSSC InternalFailure

DNSSC KeySigningKeysNeedingAction

DNSSC KeySigningKeyMaxNeedingActionAge

DNSSC KeySigningKeyAge

Amazon Route 53 Resolver endpoint

CloudWatch 應用程式深入解析支援下列指標：

EndpointHealthy英尼康

EndpointUnHealthy英尼康

InboundQueryVolume

OutboundQueryVolume

OutboundQueryAggregateVolume

AWS Network Firewall 規則群組

CloudWatch 應用程式深入解析支援下列指標：

FirewallRuleGroupQueryVolume

AWS Network Firewall 規則群組關聯

CloudWatch 應用程式深入解析支援下列指標：

FirewallRuleGroupVpcQueryVolume

具有指標資料點要求的指標

針對沒有明顯預設警示閾值的指標，Application Insights 會等到指標有足夠的資料點可預測合理的警示閾值。「 CloudWatch 應用程式深入解析」在建立警示之前檢查的量度資料點需求如下：

- 指標至少有過去 15 天到過去 2 天的 100 個資料點。
- 指標至少有最後一天的 100 個資料點。

以下指標會遵循以下資料點要求。請注意， CloudWatch 代理程式指標最多需要一小時才能建立警示。

指標

- AWS/ApplicationELB
- AW/ AutoScaling
- AWS/EC2
- Elastic Block Store (EBS)
- AWS/ELB
- AWS/RDS
- AWS/Lambda
- AWS/SQS
- AWS/CWAgent
- AWS/DynamoDB
- AWS/S3
- AWS/狀態
- AW/ ApiGateway
- AWS/SNS

AWS/ApplicationELB

ActiveConnectionCount

ConsumedLCUs

HTTPCode_ELB_4XX_Count

HTTPCode_Target_2XX_Count

HTTPCode_Target_3XX_Count

HTTPCode_Target_4XX_Count

HTTPCode_Target_5XX_Count

NewConnectionCount

ProcessedBytes

TargetResponseTime

UnHealthyHostCount

AW/ AutoScaling

GroupDesiredCapacity

GroupInServiceInstances

GroupMaxSize

GroupMinSize

GroupPendingInstances

GroupStandbyInstances

GroupTerminatingInstances

GroupTotalInstances

AWS/EC2

處理器 CreditBalance

處理器 CreditUsage

處理器 SurplusCreditBalance

處理器 SurplusCreditsCharged

CPUUtilization

DiskReadBytes

DiskReadOps

DiskWriteBytes

DiskWriteOps

EBS 百分比 ByteBalance

EBSIOBalance%

EBS ReadBytes

EBS ReadOps

EBS WriteBytes

EBS WriteOps

NetworkIn

NetworkOut

NetworkPacketsIn

NetworkPacketsOut

Elastic Block Store (EBS)

VolumeReadBytes

VolumeWriteBytes

VolumeReadOps

VolumeWriteOps

VolumeTotalReadTime

VolumeTotalWriteTime

VolumeIdleTime

VolumeQueueLength

VolumeThroughputPercentage

VolumeConsumedReadWriteOps

BurstBalance

AWS/ELB

估計 ActiveConnectionCount

EstimatedALBConsumedLCUs

估計 NewConnectionCount

EstimatedProcessedBytes

HTTPCode_Backend_4XX

HTTPCode_Backend_5XX

HealthyHostCount

Latency (延遲)

RequestCount

SurgeQueueLength

UnHealthyHostCount

AWS/RDS

ActiveTransactions

AuroraBinlogReplicaLag

AuroraReplicaLag

BackupRetentionPeriodStorageUsed

BinLogDiskUsage

BlockedTransactions

處理器 CreditBalance

CommitLatency

CommitThroughput

DDLLatency

DDLThroughput

DMLLatency

DMLThroughput

DatabaseConnections

鎖死

DeleteLatency

DeleteThroughput

DiskQueueDepth

EngineUptime

FreeLocalStorage

FreeStorageSpace

FreeableMemory

InsertLatency

InsertThroughput

LoginFailures

NetworkReceiveThroughput

NetworkThroughput

NetworkTransmitThroughput

查詢

ReadIOPS

ReadThroughput

SelectLatency

SelectThroughput

SnapshotStorageUsed

TotalBackupStorageBilled

UpdateLatency

UpdateThroughput

VolumeBytesUsed

VolumeReadIOP

VolumeWriteIOP

WriteIOPS

WriteThroughput

AWS/Lambda

錯誤

DeadLetterErrors

持續時間

限流

IteratorAge

ProvisionedConcurrencySpilloverInvocations

AWS/SQS

ApproximateAgeOfOldestMessage

ApproximateNumberOfMessagesDelayed

ApproximateNumberOfMessagesNotVisible

ApproximateNumberOfMessagesVisible

NumberOfEmptyReceives

NumberOfMessagesDeleted

NumberOfMessagesReceived

NumberOfMessagesSent

AWS/CWAgent

LogicalDisk % 可用空間

記憶體 % 使用中的認可位元組

記憶體可用的 MB 數

網路介面位元組總數/秒

分頁檔用量 %

PhysicalDisk % 磁碟時間

PhysicalDisk 平均 Disk sec/Read

PhysicalDisk 平均 Disk sec/Write

PhysicalDisk 磁碟讀取位元組/秒

PhysicalDisk 磁碟讀取次數/秒

PhysicalDisk 磁碟寫入位元組/秒

PhysicalDisk 磁碟寫入次數/秒

處理器 % 閒置時間

處理器 % 中斷時間

處理器 % 處理器時間

處理器 % 使用者時間

SQLServer:Access Methods Forwarded Records/sec

SQLServer:Access Methods Page Splits/sec

SQLServer:Buffer Manager 緩衝快取命中率

SQLServer:Buffer Manager 頁面的預期壽命

SQLServer:已接收的資料庫複本檔案位元組/秒

SQLServer:已接收的資料庫複本日誌位元組/秒

SQL Server:剩餘可復原的資料庫複本日誌

SQL Server:資料庫複本日誌傳送佇列

SQLServer:資料庫複本鏡像寫入交易/秒

SQL Server:資料庫複本復原佇列

SQL Server:剩餘的資料庫複本重做位元組

SQL Server:資料庫複本已重做位元組/秒

SQL Server:需要復原的資料庫複本日誌合計

SQL Server:資料庫複本交易延遲

SQLServer:General Statistics Processes blocked

SQLServer:SQL Statistics 每秒批次要求數

SQLServer:SQL Statistics SQL Compilations/sec

SQLServer:SQL Statistics SQL Re-Compilations/sec

系統處理器佇列長度

已建立 TCPv4 連線

已建立 TCPv6 連線

AWS/DynamoDB

ConsumedReadCapacityUnits

ConsumedWriteCapacityUnits

ReadThrottleEvents

WriteThrottleEvents

TimeToLiveDeletedItemCount

ConditionalCheckFailedRequests

TransactionConflict

ReturnedRecordsCount

PendingReplicationCount

ReplicationLatency

AWS/S3

ReplicationLatency

BytesPendingReplication

OperationsPendingReplication

4xxErrors

5xxErrors

AllRequests

GetRequests

PutRequests

DeleteRequests

HeadRequests

PostRequests

SelectRequests

ListRequests

SelectScannedBytes

SelectReturnedBytes

FirstByteLatency

TotalRequestLatency

BytesDownloaded

BytesUploaded

AWS/狀態

ActivitiesScheduled

ActivitiesStarted

ActivitiesSucceeded

ActivityScheduleTime

ActivityRuntime

ActivityTime

LambdaFunctionsScheduled

LambdaFunctionsStarted

LambdaFunctionsSucceeded

LambdaFunctionScheduleTime

LambdaFunctionRuntime

LambdaFunctionTime

ServiceIntegrationsScheduled

ServiceIntegrationsStarted

ServiceIntegrationsSucceeded

ServiceIntegrationScheduleTime

ServiceIntegrationRuntime

ServiceIntegrationTime

ProvisionedRefillRate

ProvisionedBucketSize

ConsumedCapacity

ThrottledEvents

AW/ ApiGateway

4XXError

IntegrationLatency

Latency (延遲)

DataProcessed

CacheHitCount

CacheMissCount

AWS/SNS

NumberOfNotificationsDelivered

NumberOfMessagesPublished

NumberOfNotificationsFailed

NumberOfNotificationsFilteredOut

NumberOfNotificationsFilteredOut-InvalidAttributes

NumberOfNotificationsFilteredOut-NoMessageAttributes

NumberOfNotificationsRedrivenToDlq

NumberOfNotificationsFailedToRedriveToDlq

短信 SuccessRate

建議的指標

下表列出每個元件類型的建議指標。

元件類型	工作負載類型	建議的指標
EC2 執行個體 (Windows 伺服器)	預設/自訂	CPUUtilization StatusCheckFailed 處理器 % 處理器時間 記憶體 % 使用中的認可位元組 LogicalDisk % 可用空間 記憶體可用的 MB 數
	Active Directory	CPUUtilization StatusCheckFailed 處理器 % 處理器時間 記憶體 % 使用中的認可位元組 記憶體可用的 MB 數 資料庫 ==> 執行個體資料庫快取 % 命中率 DirectoryServices DRA 擱置中複寫作業 DirectoryServices DRA 擱置複寫同步 DNS 遞迴查詢失敗/秒 LogicalDisk 平均 磁碟佇列長度

元件類型	工作負載類型	建議的指標
	Java Application	CPUUtilization StatusCheckFailed 處理器 % 處理器時間 記憶體 % 使用中的認可位元組 記憶體可用的 MB 數 java_lang_threading_threadcount java_lang_classloading_loadedclasscount java_lang_memory_heapmemoryusage_used java_lang_memory_heapmemoryusage_committed java_lang_operatingsystem_freephysicalmemorysize java_lang_operatingsystem_freeswapspacesize

元件類型	工作負載類型	建議的指標
	Microsoft IIS/.NET Web 前端	CPUUtilization StatusCheckFailed 處理器 % 處理器時間 記憶體 % 使用中的認可位元組 記憶體可用的 MB 數 每秒擲出的 .NET CLR 異常數 .NET CLR 記憶體數總遞交的位元組 .NET CLR 記憶體在 GC 中時間佔比 應用程式佇列中的 ASP.NET 應用程式請求 ASP.NET 請求已排入佇列 ASP.NET 應用程式重新啟動

元件類型	工作負載類型	建議的指標
	Microsoft SQL 伺服器資料庫層	CPUUtilization StatusCheckFailed 處理器 % 處理器時間 記憶體 % 使用中的認可位元組 記憶體可用的 MB 數 分頁檔用量 % 系統處理器佇列長度 網路介面位元組總數/秒 PhysicalDisk % 磁碟時間 SQLServer:Buffer Manager 緩衝快取命中率 SQLServer:Buffer Manager 頁面的預期壽命 SQLServer:General Statistics Processes Blocked SQLServer：一般統計資料使用者連線 SQLServer：鎖定死鎖數目/秒 SQLServer:SQL Statistics Batch Requests/Sec

元件類型	工作負載類型	建議的指標
	MySQL	CPUUtilization StatusCheckFailed 處理器 % 處理器時間 記憶體 % 使用中的認可位元組 LogicalDisk % 可用空間 記憶體可用的 MB 數
	.NET 背景工作集區/中層	CPUUtilization StatusCheckFailed 處理器 % 處理器時間 記憶體 % 使用中的認可位元組 記憶體可用的 MB 數 每秒擲出的 .NET CLR 異常數 .NET CLR 記憶體數總遞交的位元組 .NET CLR 記憶體在 GC 中時間佔比
	.NET 核心層	CPUUtilization StatusCheckFailed 處理器 % 處理器時間 記憶體 % 使用中的認可位元組 記憶體可用的 MB 數

元件類型	工作負載類型	建議的指標
	Oracle	CPUUtilization StatusCheckFailed 處理器 % 處理器時間 記憶體 % 使用中的認可位元組 LogicalDisk % 可用空間 記憶體可用的 MB 數
	Postgres	CPUUtilization StatusCheckFailed 處理器 % 處理器時間 記憶體 % 使用中的認可位元組 LogicalDisk % 可用空間 記憶體可用的 MB 數

元件類型	工作負載類型	建議的指標
	SharePoint	CPUUtilization StatusCheckFailed 處理器 % 處理器時間 記憶體 % 使用中的認可位元組 記憶體可用的 MB 數 ASP.NET 應用程式快取 API 裁剪 ASP.NET 請求已遭拒絕 ASP.NET 工作程序重新啟動 記憶體分頁/秒 SharePoint 發布緩存發布緩存刷新/秒 SharePoint 基礎執行時間/頁請求 SharePoint 以磁碟為基礎的快取記憶體壓縮總數 SharePoint 磁碟式快取 Blob 快取命中率 SharePoint 磁碟式快取 Blob 快取填滿率 SharePoint 磁碟式快取 Blob 快取記憶體清除/秒 ASP.NET 請求已排入佇列

元件類型	工作負載類型	建議的指標
		應用程式佇列中的 ASP.NET 應用程式請求 ASP.NET 應用程式重新啟動 LogicalDisk 平均 Disk sec/ Write LogicalDisk 平均 Disk sec/ Read 處理器 % 中斷時間
EC2 執行個體 (Linux 伺服器)	預設/自訂	CPUUtilization StatusCheckFailed disk_used_percent mem_used_percent

元件類型	工作負載類型	建議的指標
	Java Application	CPUUtilization StatusCheckFailed disk_used_percent mem_used_percent java_lang_threading_threadcount java_lang_classloading_loadedclasscount java_lang_memory_heapmemoryusage_used java_lang_memory_heapmemoryusage_committed java_lang_operatingsystem_freephysicalmemorysize java_lang_operatingsystem_freeswapspacesize
	.NET Core 層或 SQL Server Database 層	CPUUtilization StatusCheckFailed disk_used_percent mem_used_percent

元件類型	工作負載類型	建議的指標
	Oracle	CPUUtilization StatusCheckFailed disk_used_percent mem_used_percent
	Postgres	CPUUtilization StatusCheckFailed disk_used_percent mem_used_percent

元件類型	工作負載類型	建議的指標
EC2 執行個體群組	SAP HANA 多節點或單一節點	• hanadb_server_startup_time_variations_seconds • hanadb_level_5_alerts_count • hanadb_level_4_alerts_count • hanadb_out_of_memory_events_count • hanadb_max_trigger_read_ratio_percent • hanadb_max_trigger_write_ratio_percent • hanadb_log_switch_race_ratio_percent • hanadb_time_since_last_savepoint_seconds • hanadb_disk_usage_highlevel_percent • hanadb_current_allocation_limit_used_percent • hanadb_table_allocation_limit_used_percent • hanadb_cpu_usage_percent • hanadb_plan_cache_hit_ratio_percent • hanadb_last_data_backup_age_days

元件類型	工作負載類型	建議的指標
EBS 磁碟區	任何	VolumeReadBytes VolumeWriteBytes VolumeReadOps VolumeWriteOps VolumeQueueLength VolumeThroughputPercentage VolumeConsumedReadWriteOps BurstBalance
傳統 ELB	任何	HTTPCode_Backend_4XX HTTPCode_Backend_5XX Latency (延遲) SurgeQueueLength UnHealthyHostCount
應用程式 ELB	任何	HTTPCode_Target_4XX_Count HTTPCode_Target_5XX_Count TargetResponseTime UnHealthyHostCount

元件類型	工作負載類型	建議的指標
RDS 資料庫執行個體	任何	CPUUtilization ReadLatency WriteLatency BurstBalance 失敗的 SQL ServerAgentJobsCount
RDS 資料庫叢集	任何	CPUUtilization CommitLatency DatabaseConnections 鎖死 FreeableMemory NetworkThroughput VolumeBytesUsed
Lambda 功能	任何	持續時間 錯誤 IteratorAge ProvisionedConcurrencySpilloverInvocations 限流

元件類型	工作負載類型	建議的指標
SQS 佇列	任何	ApproximateAgeOfOldestMessage ApproximateNumberOfMessagesVisible NumberOfMessagesSent
Amazon DynamoDB 資料表	任何	SystemErrors UserErrors ConsumedReadCapacityUnits ConsumedWriteCapacityUnits ReadThrottleEvents WriteThrottleEvents ConditionalCheckFailedRequests TransactionConflict

元件類型	工作負載類型	建議的指標
Amazon S3 儲存貯體	任何	如果啟用具有複寫時間控制 (RTC) 的複寫組態： ReplicationLatency BytesPendingReplication OperationsPendingReplication 如果請求指標已開啟： 5xxErrors 4xxErrors BytesDownloaded BytesUploaded

元件類型	工作負載類型	建議的指標
AWS Step Functions	任何	一般 • ExecutionThrottled • ExecutionsAborted • ProvisionedBucketSize • ProvisionedRefillRate • ConsumedCapacity 如果狀態機器類型為 **EXPRESS** 或日誌群組層級為 **OFF** • ExecutionsFailed • ExecutionsTimedOut 如果狀態機器具有 Lambda 函數 • LambdaFunctionsFailed • LambdaFunctionsTimedOut 如果狀態機器有活動 • ActivitiesFailed • ActivitiesTimedOut • ActivitiesHeartbeatTimedOut 如果狀態機器有服務整合 • ServiceIntegrationsFailed • ServiceIntegrationsTimedOut

元件類型	工作負載類型	建議的指標
API Gateway REST API 階段	任何	• 4XXErrors • 5XXErrors • Latency (延遲)

元件類型	工作負載類型	建議的指標
ECS 叢集	任何	CpuUtilized MemoryUtilized NetworkRxBytes NetworkTxBytes RunningTaskCount PendingTaskCount StorageReadBytes StorageWriteBytes CPUReservation (僅限 EC2 啟動類型) CPUUtilization (僅限 EC2 啟動類型) MemoryReservation (僅適用於 EC2 啟動類型) MemoryUtilization (僅適用於 EC2 啟動類型) GPUReservation (僅限 EC2 啟動類型) instance_cpu_utilization (僅限 EC2 啟動類型) instance_filesystem_utilization (僅限 EC2 啟動類型) instance_memory_utilization (僅限 EC2 啟動類型)

元件類型	工作負載類型	建議的指標
		instance_network_total_bytes (僅限 EC2 啟動類型)

元件類型	工作負載類型	建議的指標
	Java Application	CpuUtilized MemoryUtilized NetworkRxBytes NetworkTxBytes RunningTaskCount PendingTaskCount StorageReadBytes StorageWriteBytes CPUReservation (僅限 EC2 啟動類型) CPUUtilization (僅限 EC2 啟動類型) MemoryReservation (僅適用於 EC2 啟動類型) MemoryUtilization (僅適用於 EC2 啟動類型) GPUReservation (僅限 EC2 啟動類型) instance_cpu_utilization (僅限 EC2 啟動類型) instance_filesystem_utilization (僅限 EC2 啟動類型) instance_memory_utilization (僅限 EC2 啟動類型)

元件類型	工作負載類型	建議的指標
		instance_network_total_bytes (僅限 EC2 啟動類型) java_lang_threading_threadcount java_lang_classloading_loadedclasscount java_lang_memory_heapmemoryusage_used java_lang_memory_heapmemoryusage_committed java_lang_operatingsystem_freephysicalmemorysize java_lang_operatingsystem_freeswapspacesize
ECS 服務	任何	CPUUtilization MemoryUtilization CpuUtilized MemoryUtilized NetworkRxBytes NetworkTxBytes RunningTaskCount PendingTaskCount StorageReadBytes StorageWriteBytes

元件類型	工作負載類型	建議的指標
	Java Application	CPUUtilization MemoryUtilization CpuUtilized MemoryUtilized NetworkRxBytes NetworkTxBytes RunningTaskCount PendingTaskCount StorageReadBytes StorageWriteBytes java_lang_threading_threadcount java_lang_classloading_loadedclasscount java_lang_memory_heapmemoryusage_used java_lang_memory_heapmemoryusage_committed java_lang_operatingsystem_freephysicalmemorysize java_lang_operatingsystem_freeswapspacesize

元件類型	工作負載類型	建議的指標
EKS 叢集	任何	cluster_failed_node_count node_cpu_reserved_capacity node_cpu_utilization node_filesystem_utilization node_memory_reserved_capacity node_memory_utilization node_network_total_bytes pod_cpu_reserved_capacity pod_cpu_utilization pod_cpu_utilization_over_pod_limit pod_memory_reserved_capacity pod_memory_utilization pod_memory_utilization_over_pod_limit pod_network_rx_bytes pod_network_tx_bytes

元件類型	工作負載類型	建議的指標
	Java Application	cluster_failed_node_count node_cpu_reserved_capacity node_cpu_utilization node_filesystem_utilization node_memory_reserved_capacity node_memory_utilization node_network_total_bytes pod_cpu_reserved_capacity pod_cpu_utilization pod_cpu_utilization_over_pod_limit pod_memory_reserved_capacity pod_memory_utilization pod_memory_utilization_over_pod_limit pod_network_rx_bytes pod_network_tx_bytes java_lang_threading_threadcount java_lang_classloading_loadedclasscount

元件類型	工作負載類型	建議的指標
		java_lang_memory_heapmemoryusage_used java_lang_memory_heapmemoryusage_committed java_lang_operatingsystem_freephysicalmemorysize java_lang_operatingsystem_freeswapspacesize

元件類型	工作負載類型	建議的指標
EC2 上的 Kubernetes 叢集	任何	cluster_failed_node_count node_cpu_reserved_capacity node_cpu_utilization node_filesystem_utilization node_memory_reserved_capacity node_memory_utilization node_network_total_bytes pod_cpu_reserved_capacity pod_cpu_utilization pod_cpu_utilization_over_pod_limit pod_memory_reserved_capacity pod_memory_utilization pod_memory_utilization_over_pod_limit pod_network_rx_bytes pod_network_tx_bytes

元件類型	工作負載類型	建議的指標
	Java Application	cluster_failed_node_count node_cpu_reserved_capacity node_cpu_utilization node_filesystem_utilization node_memory_reserved_capacity node_memory_utilization node_network_total_bytes pod_cpu_reserved_capacity pod_cpu_utilization pod_cpu_utilization_over_pod_limit pod_memory_reserved_capacity pod_memory_utilization pod_memory_utilization_over_pod_limit pod_network_rx_bytes pod_network_tx_bytes java_lang_threading_threadcount java_lang_classloading_loadedclasscount

元件類型	工作負載類型	建議的指標
		java_lang_memory_heapmemoryusage_used java_lang_memory_heapmemoryusage_committed java_lang_operatingsystem_freephysicalmemorysize java_lang_operatingsystem_freeswapspacesize

下表列出每個元件類型的建議程序和程序測量結果。 CloudWatch 對於未在執行個體上執行的程序，「應用程式深入解析」不建議進行程序監視。

元件類型	工作負載類型	建議程式	建議的指標

EC2 執行個體 (Windows 伺服器)	Microsoft IIS/.NET Web 前端	`w3wp`	`procstat cpu_usage ,`
			
			`procstat memory_rss ,`
			
			`procstat memory_vms ,`
			
			`procstat read_bytes ,`
			
			`procstat write_bytes`

	Microsoft SQL 伺服器資料庫層	`SQLAgent`	`procstat cpu_usage ,`
			
			`procstat memory_rss ,`

元件類型	工作負載類型	建議程式	建議的指標
			procstat memory_vms , procstat read_bytes , procstat write_bytes
		sqlservr	procstat cpu_usage , procstat memory_rss , procstat memory_vms , procstat read_bytes , procstat write_bytes
		sqlwriter	procstat cpu_usage , procstat memory_rss
		Reporting ServicesS ervice	procstat cpu_usage , procstat memory_rss

元件類型	工作負載類型	建議程式	建議的指標
		MsDtsServr	procstat cpu_usage , procstat memory_rss , procstat memory_vms , procstat read_bytes , procstat write_bytes
		Msmdsrv	procstat cpu_usage , procstat memory_rss , procstat memory_vms , procstat read_bytes , procstat write_bytes

元件類型	工作負載類型	建議程式	建議的指標
	.NET 背景工作集區/中層	w3wp	procstat cpu_usage , procstat memory_rss , procstat memory_vms , procstat read_bytes , procstat write_bytes
	.NET 核心層	w3wp	procstat cpu_usage , procstat memory_rss , procstat memory_vms , procstat read_bytes , procstat write_bytes

效能計數器指標

只有在 Windows 執行個體上安裝對應的效能計數器集時，才建議執行個體使用效能計數器指標。

效能計數器指標名稱	效能計數器集名稱
擲出的 .NET CLR 異常數	.NET CLR 異常情形

效能計數器指標名稱	效能計數器集名稱
每秒擲出的 .NET CLR 異常數	.NET CLR 異常情形
每秒篩選的 .NET CLR 異常數	.NET CLR 異常情形
每秒最終的 .NET CLR 異常數	.NET CLR 異常情形
每秒擲出以捕捉深度的 .NET CLR 異常	.NET CLR 異常情形
CCW 的 .NET CLR Interop 數	.NET CLR Interop
Stub 的 .NET CLR Interop 數	.NET CLR Interop
每秒 TLB 匯出的 .NET CLR Interop 數	.NET CLR Interop
每秒 TLB 匯入的 .NET CLR Interop 數	.NET CLR Interop
封送處理的 .NET CLR Interop 數	.NET CLR Interop
Jit 的 .NET CLR Jit 時間百分比	.NET CLR Jit
.NET CLR Jit 標準 Jit 失敗	.NET CLR Jit
.NET CLR 正在載入 - 正在載入時間百分比	.NET CLR 正在載入
.NET CLR 載入失敗的載入速率	.NET CLR 正在載入
.net CLR LocksAndThreads 爭用率/秒	网络 CLR LocksAndThreads
每秒 .NET CLR LocksAndThreads 佇列長度	网络 CLR LocksAndThreads
.NET CLR 記憶體數總遞交的位元組	.NET CLR 記憶體
.NET CLR 記憶體在 GC 中時間佔比	.NET CLR 記憶體
網路 CLR 網路平均佇列時間 4.0.0.0 HttpWebRequest	.NET CLR Networking 4.0.0.0
無線 CLR 網路 4. HttpWebRequests 0.0.0 中止/秒	.NET CLR Networking 4.0.0.0

效能計數器指標名稱	效能計數器集名稱
.NET CLR 網路連線 4. HttpWebRequests 0.0.0 失敗/秒	.NET CLR Networking 4.0.0.0
.NET CLR 網路連線 4. HttpWebRequests 0.0.0 已佇列/秒	.NET CLR Networking 4.0.0.0
APP_POL_WAS 總工作者處理序 Ping 失敗	APP_POOL_WAS
ASP.NET 應用程式重新啟動	ASP.NET
ASP.NET 請求已遭拒絕	ASP.NET
ASP.NET 工作程序重新啟動	ASP.NET
ASP.NET 應用程式快取 API 裁剪	ASP.NET 應用程式
ASP.NET 應用程式 % 受管處理器時間 (估計)	ASP.NET 應用程式
ASP.NET 應用程式錯誤總計/秒	ASP.NET 應用程式
ASP.NET 應用程式執行期間未處理錯誤/秒	ASP.NET 應用程式
應用程式佇列中的 ASP.NET 應用程式請求	ASP.NET 應用程式
ASP.NET 應用程式請求/秒	ASP.NET 應用程式
ASP.NET 請求等待時間	ASP.NET
ASP.NET 請求已排入佇列	ASP.NET
資料庫 ==> 執行個體資料庫快取 % 命中率	資料庫 ==> 執行個體
資料庫 ==> 執行個體輸入/輸出資料庫讀取平均延遲	資料庫 ==> 執行個體
資料庫 ==> 執行個體輸入/輸出資料庫讀取/秒	資料庫 ==> 執行個體
資料庫 ==> 執行個體輸入/輸出日誌寫入平均延遲	資料庫 ==> 執行個體

效能計數器指標名稱	效能計數器集名稱
DirectoryServices DRA 擱置中複寫作業	DirectoryServices
DirectoryServices DRA 擱置複寫同步	DirectoryServices
DirectoryServices LDAP 連結時間	DirectoryServices
DNS 遞迴查詢/秒	DNS
DNS 遞迴查詢失敗/秒	DNS
收到的 DNS TCP 查詢/秒	DNS
收到的 DNS 查詢總數/秒	DNS
傳送的 DNS 回應總數/秒	DNS
收到的 DNS UDP 查詢/秒	DNS
HTTP 服務要求佇列 CurrentQueueSize	HTTP 服務要求佇列
LogicalDisk % 可用空間	LogicalDisk
LogicalDisk 平均 Disk sec/Write	LogicalDisk
LogicalDisk 平均 Disk sec/Read	LogicalDisk
LogicalDisk 平均 磁碟佇列長度	LogicalDisk
記憶體 % 使用中的認可位元組	記憶體
記憶體可用的 MB 數	記憶體
記憶體分頁/秒	記憶體
記憶體長期平均備用快取生命週期 (s)	記憶體
網路介面位元組總數/秒	網路介面
收到的網路介面位元組/秒	網路介面

效能計數器指標名稱	效能計數器集名稱
傳送的網路介面位元組/秒	網路介面
網路介面電流頻寬	網路介面
分頁檔用量 %	分頁檔
PhysicalDisk % 磁碟時間	PhysicalDisk
PhysicalDisk 平均 磁碟佇列長度	PhysicalDisk
PhysicalDisk 平均 Disk Sec/Read	PhysicalDisk
PhysicalDisk 平均 Disk Sec/Write	PhysicalDisk
PhysicalDisk 磁碟讀取位元組/秒	PhysicalDisk
PhysicalDisk 磁碟讀取次數/秒	PhysicalDisk
PhysicalDisk 磁碟寫入位元組/秒	PhysicalDisk
PhysicalDisk 磁碟寫入次數/秒	PhysicalDisk
處理器 % 閒置時間	處理器
處理器 % 中斷時間	處理器
處理器 % 處理器時間	處理器
處理器 % 使用者時間	處理器
SharePoint 磁碟式快取 Blob 快取填滿率	SharePoint 磁碟式快取
SharePoint 磁碟式快取 Blob 快取記憶體清除/秒	SharePoint 磁碟式快取
SharePoint 磁碟式快取 Blob 快取命中率	SharePoint 磁碟式快取
SharePoint 以磁碟為基礎的快取記憶體壓縮總數	SharePoint 磁碟式快取
SharePoint 基礎執行時間/頁請求	SharePoint 基金會

效能計數器指標名稱	效能計數器集名稱
SharePoint 發布緩存發布緩存刷新/秒	SharePoint 發佈快取
安全全系統範圍統計數字 Kerberos 身分驗證	安全全系統範圍統計數字
安全全系統範圍統計數字 NTLM 身分驗證	安全全系統範圍統計數字
SQLServer:Access Methods Forwarded Records/Sec	SQLServer:Access Methods
SQLServer:Access Methods Full Scans/Sec	SQLServer:Access Methods
SQLServer:Access Methods Page Splits/Sec	SQLServer:Access Methods
SQLServer:Buffer Manager 緩衝快取命中率	SQLServer:Buffer Manager
SQLServer:Buffer Manager 頁面的預期壽命	SQLServer:Buffer Manager
SQLServer:已接收的資料庫複本檔案位元組/秒	SQLServer:Database Replica
SQLServer:已接收的資料庫複本日誌位元組/秒	SQLServer:Database Replica
SQL Server:剩餘可復原的資料庫複本日誌	SQLServer:Database Replica
SQL Server:資料庫複本日誌傳送佇列	SQLServer:Database Replica
SQLServer:資料庫複本鏡像寫入交易/秒	SQLServer:Database Replica
SQL Server:資料庫複本復原佇列	SQLServer:Database Replica
SQL Server:剩餘的資料庫複本重做位元組	SQLServer:Database Replica
SQL Server:資料庫複本已重做位元組/秒	SQLServer:Database Replica
SQL Server:需要復原的資料庫複本日誌合計	SQLServer:Database Replica
SQL Server:資料庫複本交易延遲	SQLServer:Database Replica
SQLServer:General Statistics Processes Blocked	SQLServer:General Statistics

效能計數器指標名稱	效能計數器集名稱
SQLServer：一般統計資料使用者連線	SQLServer:General Statistics
SQLServer：閂鎖平均閂鎖等待時間 (毫秒)	SQLServer：閂鎖
SQLServer：鎖定平均等待時間 (毫秒)	SQLServer：鎖定
SQLServer：鎖定鎖定逾時/秒	SQLServer：鎖定
SQLServer：鎖定鎖定等待/秒	SQLServer：鎖定
SQLServer：鎖定死鎖數目/秒	SQLServer：鎖定
SQLServer：記憶體管理員記憶體授權待定	SQLServer：記憶體管理員
SQLServer:SQL Statistics Batch Requests/Sec	SQLServer:SQL Statistics
SQLServer:SQL Statistics SQL Compilations/Sec	SQLServer:SQL Statistics
SQLServer:SQL Statistics SQL Re-Compilations/Sec	SQLServer:SQL Statistics
系統處理器佇列長度	系統
已建立 TCPv4 連線	TCPv4
已建立 TCPv6 連線	TCPv6
W3SVC_W3WP 檔案饋取排清	W3SVC_W3WP
W3SVC_W3WP 檔案快取遺漏	W3SVC_W3WP
W3SVC_W3WP 請求/秒	W3SVC_W3WP
W3SVC_W3WP URI 快取排清	W3SVC_W3WP
W3SVC_W3WP URI 快取遺漏	W3SVC_W3WP
接收的 Web 服務位元組/秒	Web 服務

效能計數器指標名稱	效能計數器集名稱
傳送的 Web 服務位元組/秒	Web 服務
Web 服務連線嘗試/秒	Web 服務
Web 服務目前連線數目	Web 服務
Web 服務收到請求/秒	Web 服務
Web 服務 POST 請求/秒	Web 服務

使用主控台中的資源健全狀況檢 CloudWatch 視

您可以使用資源運作狀態檢視，在單一檢視中跨應用程式自動探索、管理及視覺化主機的運作狀態和效能。您可以透過效能維度 (例如 CPU 或記憶體) 視覺化其主機的運作狀態，並使用篩選條件在單一檢視中並排顯示數百台主機。您可以依標籤或使用案例 (例如相同 Auto Scaling 群組中的主機或使用相同負載平衡器的主機) 進行篩選。

必要條件

若要確認您獲得資源運作狀態檢視的完整效益，請檢查您是否具備下列先決條件。

- 若要查看主機的記憶體使用率並將其用作篩選器，您必須在主機上安裝 CloudWatch 代理程式，並將其設定為在預設`CWAgent`命名空間 CloudWatch 中將記憶體指標傳送至。在 Linux 和 macOS 執行個體上， CloudWatch 代理程式必須傳送指`mem_used_percent`標。在 Windows 執行個體上，代理程式必須傳送 `Memory % Committed Bytes In Use` 指標。如果您使用精靈建立 CloudWatch 代理程式組態檔，並選取任何預先定義的測量結果集，就會包含這些測量結果。 CloudWatch 代理程式收集的指標會以自訂指標計費。如需詳細資訊，請參閱 安裝 CloudWatch 代理程式。

  當您使用 CloudWatch 代理程式收集這些記憶體測量結果以與資源健全狀況檢視搭配使用時，您必須在 CloudWatch 代理程式組態檔中包含下列區段。此區段包含預設維度設定，依預設會建立，因此請勿將此區段的任何部分變更為與下列範例所示不同的任何部分。

```
"append_dimensions": {
  "ImageId": "${aws:ImageId}",
  "InstanceId": "${aws:InstanceId}",
  "InstanceType": "${aws:InstanceType}",
  "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
```

```
},
```

- 若要檢視資源運作狀態檢視中可用的所有資訊，您必須登入擁有下列許可的帳戶。如果您使用較少的許可登入，您仍然可以使用資源運作狀態檢視，但是某些效能資料將無法看見。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "autoscaling:Describe*",
                "cloudwatch:Describe*",
                "cloudwatch:Describe*",
                "cloudwatch:Get*",
                "cloudwatch:List*",
                "logs:Get*",
                "logs:Describe*",
                "sns:Get*",
                "sns:List*",
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeRegions"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

若要檢視您帳戶中的資源運作狀態

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇基礎設施監控，然後選擇資源運作狀態。

   隨即出現資源運作狀態頁面，並為您的帳戶中的每個主機顯示一個方形。每個方形都會根據該主機的目前狀態和 Color by (顏色依據) 的設定著色。具有警示符號的主機方形有一個或多個警示目前處於 ALARM 狀態。

   您可以在單一檢視中看到多達 500 台主機。如果您的帳戶中有更多主機，請使用此程序的步驟 6 中的篩選條件設定。

3. 若要變更用來顯示每個主機運作狀態的條件，請選擇 Color by (顏色依據) 的設定。您可以選擇 CPU Utilization (CPU 使用率)、Memory Utilization (記憶體使用率) 或 Status check (狀態檢查)。記憶體使用率測量結果僅適用於執行 CloudWatch 代理程式並設定為收集記憶體測量結果並將其傳送至預設`CWAgent`命名空間的主機。如需詳細資訊，請參閱 使用 CloudWatch 代理程式收集指標、記錄和追蹤。
4. 若要變更網格中運作狀態指示器所使用的閾值和顏色，請選擇網格上方的齒輪圖示。
5. 若要切換是否要在主機網格中顯示警示，請選擇或清除 Show alarms across all metrics (顯示所有指標的警示)。
6. 若要將映射中的主機分割為群組，選擇 Group by (分組依據) 的分組條件。
7. 若要將檢視縮小至較少的主機，請選擇 Filter by (篩選依據) 的篩選條件。您可以依標籤和資源群組 (例如 Auto Scaling 群組、執行個體類型、安全群組等) 進行篩選。
8. 若要排序主機，請選擇 Sort by (排序依據) 的排序條件。您可以依狀態檢查結果、執行個體狀態、CPU 或記憶體使用率，以及處於 ALARM 狀態的警示數目來排序。
9. 若要查看有關主機的詳細資訊，請選擇代表該主機的方形。隨即會顯示快顯窗格。若要深入了解該主機的相關資訊，請選擇 View dashboard (檢視儀表板) 或 View on list (檢視清單)。

CloudWatch 跨帳戶可觀察性

透過 Amazon CloudWatch 跨帳戶觀察功能，您可以監控和疑難排解跨區域內多個帳戶的應用程式。在任何連結帳戶中無縫搜尋、視覺化和分析您的指標、日誌、追蹤、應用程式洞察應用程式和 Internet Monitor 監視器監視器，而無需帳戶界限。

設置一個或多個 AWS 帳戶作為監控帳戶，並將其與多個源帳戶鏈接。監控帳戶是一個中央 AWS 帳戶，可以檢視來源帳戶產生的可觀察性資料並與之互動。來源帳戶是個別 AWS 帳戶，可為駐留在其中的資源產生可觀測性資料。來源帳戶會與監控帳戶共用其可觀察性資料。共用的可觀察性資料可以包括下列類型的遙測：

- Amazon 的指標 CloudWatch。您可以選擇與監督帳戶共用所有命名空間的測量結果，或篩選為命名空間的子集。
- Amazon CloudWatch 日誌中的日誌群組。您可以選擇與監視帳戶共用所有記錄群組，或篩選為記錄群組的子集。
- 追蹤中 AWS X-Ray
- Amazon 應用程式洞察中的 CloudWatch 應用
- 監視器在 CloudWatch 互聯網監控

若要在監視帳戶和來源帳戶之間建立連結，您可以使用 CloudWatch主控台。或者，使用 AWS CLI 和 API 中的「觀察性存取管理員」命令。如需詳細資訊，請參閱 Observability Access Manager API Reference (《Observability Access Manager API 參考》)。

接收器是代表監控帳戶中連接點的資源。來源帳戶可以連結至接收器，以共用可觀察性資料。每個帳戶每個區域可以有一個接收器。每個接收器都由其所在的監控帳戶管理。可觀察性連結是代表來源帳戶與監控帳戶之間所建立連結的資源。連結由來源帳戶管理。

有關設置 CloudWatch 跨帳戶觀察性的視頻演示，請參閱以下視頻。

下一個主題說明如何在監視帳戶和來源 CloudWatch 帳戶中設定跨帳戶可觀察性。如需跨帳戶跨區域資料 CloudWatch 面板的相關資訊，請參閱。跨帳戶跨 CloudWatch 區域主控台

將 Organizations 用於來源帳戶

將來源帳戶連結至您的監控帳戶有兩個選項。您可以使用其中一個或同時使用兩個選項。

- 用於 AWS Organizations 將組織或組織單位中的帳號連結至監視帳戶。

- 將個別 AWS 帳戶 Connect 到監控帳戶。

我們建議您使用「組 Organizations」，以便稍後在組織中建立的新 AWS 帳戶自動登入，以跨帳戶可觀察性作為來源帳戶。

連結監控帳戶和來源帳戶的詳細資料

- 每個監控帳戶可以連結到多達 100,000 個來源帳戶。
- 每個來源帳戶最多可與五個監控帳戶共用資料。
- 您可以將單一帳戶同時設定為監控帳戶和來源帳戶。如果您這樣做，此帳戶只會將本身的可觀察性資料傳送至其連結的監控帳戶。它不會從其來源帳戶轉送資料。
- 監控帳戶會指定可與其共用的遙測類型。來源帳戶會指定要共用的遙測類型。
  - 如果監控帳戶中選取的遙測類型多於來源帳戶中選取的遙測類型，則兩個帳戶之間會連結。只會共用兩個帳戶中都選取的資料類型。
  - 如果來源帳戶中選取的遙測類型多於監控帳戶中的遙測類型，則連結建立會失敗，且不會共用任何內容。
  - 指標名稱不會顯示在監控帳戶主控台中，直到該指標在建立連結後發出新的資料點。
- 若要移除帳戶之間的連結，請從來源帳戶執行此動作。
- 若要刪除監控帳戶中的接收器，您必須先移除接收監控帳戶的所有連結。

定價

中 CloudWatch 的跨帳戶可觀察性無需支付額外的日誌和指標費用，並且第一個跟踪副本是免費的。如需有關定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

內容

- 連結監控帳戶與來源帳戶
  - 必要的許可
  - 設定概觀
  - 步驟 1：設定監控帳戶
  - 步驟 2：(選擇性) 下載 AWS CloudFormation 範本或網址
  - 步驟 3：連結來源帳戶
    - 使用 AWS CloudFormation 範本將組織或組織單位中的所有帳戶設定為來源帳戶
    - 使用 AWS CloudFormation 範本設定個別來源帳戶

    - 使用 URL 來設定個別來源帳戶
- 管理監控帳戶和來源帳戶
  - 將更多來源帳戶連結至現有監控帳戶
  - 移除監控帳戶與來源帳戶之間的連結
  - 檢視監控帳戶的相關資訊

連結監控帳戶與來源帳戶

本節中的主題會說明如何設定監控帳戶和來源帳戶之間的連結。

我們建議您建立一個新 AWS 帳戶作為組織的監控帳戶。

內容

- 必要的許可
- 設定概觀
- 步驟 1：設定監控帳戶
- 步驟 2：(選擇性) 下載 AWS CloudFormation 範本或網址
- 步驟 3：連結來源帳戶
  - 使用 AWS CloudFormation 範本將組織或組織單位中的所有帳戶設定為來源帳戶
  - 使用 AWS CloudFormation 範本設定個別來源帳戶
  - 使用 URL 來設定個別來源帳戶

必要的許可

若要在監控帳戶和來源帳戶之間建立連結，您必須擁有特定許可並登入帳戶。

- 若要設定監控帳戶：您必須在監控帳戶中擁有完整的系統管理員存取權，或者您必須擁有下列許可並登入該帳戶：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSinkModification",
            "Effect": "Allow",
```

```
            "Action": [
                "oam:CreateSink",
                "oam:DeleteSink",
                "oam:PutSinkPolicy",
                "oam:TagResource"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadOnly",
            "Effect": "Allow",
            "Action": ["oam:Get*", "oam:List*"],
            "Resource": "*"
        }
    ]
}
```

- 範圍為特定監控帳戶的來源帳戶：若要僅針對某個指定監控帳戶建立、更新和管理連結，您必須擁有下列許可並登入該帳戶。在此範例中，監控帳戶為 999999999999。

  如果連結不會共用全部五種資源類型 (指標、記錄檔、追蹤、應用程式深入解析應用程式和網際網路監視器監視器) `cloudwatch:Link`、`logs:Link`，您可以視需要省略`xray:Link`、`applicationinsights:Link`、、、或`internetmonitor:Link`。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "oam:CreateLink",
                "oam:UpdateLink",
                "oam:DeleteLink",
                "oam:GetLink",
                "oam:TagResource"
            ],
            "Effect": "Allow",
            "Resource": "arn:*:oam:*:*:link/*"
        },
        {
            "Action": [
                "oam:CreateLink",
                "oam:UpdateLink"
            ],
```

```
            "Effect": "Allow",
            "Resource": "arn:*:oam:*:*:sink/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": [
                        "999999999999"
                    ]
                }
            }
        },
        {
            "Action": "oam:ListLinks",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "cloudwatch:Link",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "logs:Link",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "xray:Link",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
             "Action": "applicationinsights:Link",
             "Effect": "Allow",
             "Resource": "*"
         },
        {
             "Action": "internetmonitor:Link",
             "Effect": "Allow",
             "Resource": "*"
        }
    ]
}
```

- 具有連結至任何監視帳戶之權限的來源帳戶 — 若要建立任何現有監視帳戶接收器的連結，並共用指標、記錄群組、追蹤、Application Insights 應用程式和 Internet Monitor 監視器，您必須以完整管理員權限登入來源帳戶，或使用下列權限登入該帳戶

  如果連結不會共用全部五種資源類型 (指標、記錄檔、追蹤、應用程式深入解析應用程式和網際網路監視器監視器) `cloudwatch:Link`、`logs:Link`，您可以視需要省略`xray:Link`、`applicationinsights:Link`、`internetmonitor:Link`、或`oam:Link`。

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": [
                "oam:CreateLink",
                "oam:UpdateLink"
            ],
            "Resource": [
                "arn:aws:oam:*:*:link/*",
                "arn:aws:oam:*:*:sink/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:List*",
                "oam:Get*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:DeleteLink",
                "oam:GetLink",
                "oam:TagResource"
            ],
            "Resource": "arn:aws:oam:*:*:link/*"
        },
        {
            "Action": "cloudwatch:Link",
            "Effect": "Allow",
            "Resource": "*"
        },
```

```
        {
            "Action": "xray:Link",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "logs:Link",
            "Effect": "Allow",
            "Resource": "*"
        },
        {
             "Action": "applicationinsights:Link",
             "Effect": "Allow",
             "Resource": "*"
        },
        {
             "Action": "internetmonitor:Link",
             "Effect": "Allow",
             "Resource": "*"
        }
    ]
}
```

設定概觀

下列高階步驟說明如何設定 CloudWatch 跨帳戶可觀察性。

> **Note**
>
> 我們建議您建立新 AWS 帳戶作為組織的監控帳戶使用。

1. 設定專用的監控帳戶。
2. (選擇性) 下載 AWS CloudFormation 範本或複製 URL 以連結來源帳戶。
3. 將來源帳戶連結至監控帳戶。

完成這些步驟後，您可以使用監控帳戶來檢視來源帳戶的可觀察性資料。

步驟 1：設定監控帳戶

請依照本節中的步驟將帳戶設定為 CloudWatch 跨 AWS 帳戶觀察性的監控帳戶。

必要條件

- 如果您要將 AWS Organizations 組織中的帳戶設定為來源帳戶，請取得組織路徑或組織 ID。
- 如果您未將 Organizations 用於來源帳戶：請取得來源帳戶的帳戶 ID。

若要將帳戶設定為監控帳戶，您必須具備特定許可。如需詳細資訊，請參閱 必要的許可。

設定監控帳戶

1. 登入您想要用作監控帳戶的帳戶。
2. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
3. 在左側的導覽窗格中，選擇設定。
4. 在 Monitoring account configuration (監控帳戶組態) 中，選擇 Configure (設定)。
5. 對於選取資料，請選擇此監視帳戶是否能夠檢視記錄、指標、追蹤、應用程式深入解析-應用程式和網際網路監視器- 監視其所連結來源帳戶的資料。
6. 在 List source accounts (列出來源帳戶) 中，輸入此監控帳戶可檢視的來源帳戶。若要識別來源帳戶，請輸入個別帳戶 ID、組織路徑或組織 ID。如果您輸入組織路徑或組織 ID，系統會允許此監控帳戶檢視該組織中所有連結帳戶的可觀察性資料。

   以逗號分隔的此清單中的項目。

   > **Important**
   >
   > 當您輸入組織路徑時，請遵循確切的格式。ou-id 必須以 (斜線字元/) 結尾。例如：`o-a1b2c3d4e5/r-f6g7h8i9j0example/ou-def0-awsbbbb/`

7. 在 Define a label to identify your source account (定義標籤來識別來源帳戶) 中，指定當您使用監控帳戶檢視來源帳戶時，是否要使用帳戶名稱或電子郵件地址來識別來源帳戶。
8. 選擇 Configure (設定)。

> **Important**
>
> 設定來源帳戶之後，監控帳戶和來源帳戶之間的連結才會完成。如需詳細資訊，請參閱下列區段。

步驟 2：(選擇性) 下載 AWS CloudFormation 範本或網址

若要將來源帳戶連結至監控帳戶，建議您使用 AWS CloudFormation 範本或 URL。

- 如果您要連結整個組織，則會 CloudWatch 提供 AWS CloudFormation 範本。
- 如果您要連結個別帳號 — 請使用 AWS CloudFormation 範本或提 CloudWatch 供的 URL。

若要使用 AWS CloudFormation 範本，您必須在這些步驟中下載範本。將監視帳戶與至少一個來源帳戶連結後，即無法再下載 AWS CloudFormation 範本。

下載 AWS CloudFormation 範本或複製 URL 以將來源帳戶連結至監視帳戶

1. 登入您想要用作監控帳戶的帳戶。
2. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
3. 在左側的導覽窗格中，選擇設定。
4. 在 Monitoring account configuration (監控帳戶組態) 中，選擇 Resources to link accounts (要連結帳戶的資源)。
5. 執行以下任意一項：
   - 選擇 AWS 組織以取得範本，並用來將組織中的帳戶連結至此監視帳戶。
   - 選擇 Any account (任何帳戶) 以取得用於將個別帳戶設定為來源帳戶的範本或 URL。
6. 執行以下任意一項：
   - 如果您選擇AWS 組織，請選擇 [下載 CloudFormation 範本]。
   - 如果您選擇 [任何帳戶]，請選擇 [下載 CloudFormation 範本] 或 [複製 URL]。
7. (選擇性) 重複步驟 5-6 以下載 AWS CloudFormation 範本和 URL。

步驟 3：連結來源帳戶

請依照這些章節中的步驟，將來源帳戶連結至監控帳戶。

若要將監控帳戶與來源帳戶連結，您必須具備特定許可。如需詳細資訊，請參閱 必要的許可。

使用 AWS CloudFormation 範本將組織或組織單位中的所有帳戶設定為來源帳戶

這些步驟假設您已透過執行中的步驟來下載必要的 AWS CloudFormation 範本步驟 2：(選擇性) 下載 AWS CloudFormation 範本或網址。

使用 AWS CloudFormation 範本將組織或組織單位中的帳號連結至監視帳戶

1. 登入組織的管理帳戶。
2. 開啟主 AWS CloudFormation 控台，網址為 https://console.aws.amazon.com/cloudformation。
3. 在左側導覽列中，選擇StackSets。
4. 檢查您是否已登入所需的「區域」，然後選擇「建立」 StackSet。
5. 選擇 Next (下一步)。
6. 選擇 Template is ready (範本已準備就緒)，然後選擇 Upload a template file (上傳範本檔案)。
7. 選擇 Choose file (選擇檔案)，選擇您從監控帳戶下載的範本，然後選擇 Open (開啟)。
8. 選擇下一步。
9. 在「指定 StackSet 詳細資訊」中，輸入的名稱， StackSet 然後選擇「下一步」。
10. 在 Add stacks to stack set (將堆疊新增至堆疊集) 中，選擇 Deploy new stacks (部署新堆疊)。
11. 在 Deployment targets (部署目標) 中，選擇是部署到整個組織還是部署至指定的組織單位。
12. 對於「指定區域」，請選擇要將 CloudWatch 跨帳戶觀察性部署到哪些區域。
13. 選擇下一步。
14. 在 Review (檢閱) 頁面上，確認您選取的選項，然後選擇 Submit (提交)。
15. 在 Stack instances (堆疊執行個體) 索引標籤中，重新整理畫面，直到您看到堆疊執行個體的狀態為 CREATE_COMPLETE。

使用 AWS CloudFormation 範本設定個別來源帳戶

這些步驟假設您已透過執行中的步驟來下載必要的 AWS CloudFormation 範本步驟 2：(選擇性) 下載 AWS CloudFormation 範本或網址。

若要使用 AWS CloudFormation 範本來設定 CloudWatch跨帳戶觀察性的個別來源科目，請執行下列步驟：

1. 登入來源帳戶。
2. 開啟主 AWS CloudFormation 控台，網址為 https://console.aws.amazon.com/cloudformation。

3. 在左側導覽窗格中，選擇 Stacks (堆疊)。
4. 檢查您是否已登入到所需的區域，然後選擇 Create stack (建立堆疊)、With new resources (standard) (使用新資源 (標準))。
5. 選擇下一步。
6. 選擇 Upload a template file (上傳範本檔案)。
7. 選擇 Choose file (選擇檔案)，選擇您從監控帳戶下載的範本，然後選擇 Open (開啟)。
8. 選擇下一步。
9. 在 Specify Stack details (指定堆疊詳細資訊) 中，輸入堆疊名稱，然後選擇 Next (下一步)。
10. 在 Configure stack options (設定堆疊選項) 頁面，選擇 Next (下一步)。
11. 在 Review (檢閱) 頁面，選擇 Submit (提交)。
12. 在堆疊的狀態頁面上，重新整理畫面，直到您看到堆疊的狀態為 CREATE_COMPLETE。
13. 若要使用相同範本將更多來源帳戶連結至此監控帳戶，請登出此帳戶並登入下一個來源帳戶。然後重複步驟 2-12。

使用 URL 來設定個別來源帳戶

這些步驟假設您已透過執行 步驟 2：(選擇性) 下載 AWS CloudFormation 範本或網址 中的步驟複製了必要的 URL。

若要使用 URL 將個別來源帳戶連結至監控帳戶

1. 登入您想要用作來源帳戶的帳戶。
2. 輸入您從監控帳戶複製的 URL。

   您會看到 CloudWatch 設定頁面，其中已填入一些資訊。
3. 對於選取資料，請選擇此來源帳戶是否將記錄、指標、追蹤、應用程式深入解析-應用程式和網際網路監控-監控資料至此監視帳戶。

   對於記錄和指標，您可以選擇要與監視帳戶共用所有資源，還是共用子集。

   a. (選擇性) 若要與監控帳戶共用此帳戶的記錄群組子集，請選取 [記錄]，然後選擇 [篩選記錄]。然後使用 [篩選記錄檔] 方塊建構查詢，以尋找您要共用的記錄群組。該查詢將使用術語`LogGroupName`和一個或多個以下操作數。

      - `=` 和 `!=`
      - `AND`

- OR
- ^表示喜歡，並!^表示不喜歡。這些只能用作前綴搜索。%在您要搜尋和包含的字串結尾加入。
- IN以及NOT IN，使用圓括號 (( ))

完整的查詢不得超過 2000 個字元，且限制為五個條件運算元。條件運算元為AND和。OR其他操作數的數量沒有限制。

> **Tip**
>
> 選擇 [檢視範例查詢] 以查看常見查詢格式的正確語法。

b. (選擇性) 若要與監督帳戶共用此帳戶的測量結果命名空間子集，請選取量度，然後選擇篩選指標。然後使用「篩選度量」方塊來建構查詢，以尋找您要共用的度量命名空間。使用術語Namespace和下列一或多個運算元。

- = 和 !=
- AND
- OR
- LIKE 與 NOT LIKE。這些只能用作前綴搜索。%在您要搜尋和包含的字串結尾加入。
- IN以及NOT IN，使用圓括號 (( ))

完整的查詢不得超過 2000 個字元，且限制為五個條件運算元。條件運算元為AND和。OR其他操作數的數量沒有限制。

> **Tip**
>
> 選擇 [檢視範例查詢] 以查看常見查詢格式的正確語法。

4. 請勿更改 Enter monitoring account configuration ARN (輸入監控帳戶組態 ARN) 中的 ARN。
5. Define a label to identify your source account (定義標來識別來源帳戶) 區段會預先填入監控帳戶的標籤選項。或者，您也可以選擇 Edit (編輯) 來進行變更。
6. 選擇 Link (連結)。
7. 在方塊中輸入 **Confirm**，然後選擇 Confirm (確認)。

8. 若要使用相同 URL 將更多來源帳戶連結至此監控帳戶，請登出此帳戶並登入下一個來源帳戶。然後重複步驟 2-7。

管理監控帳戶和來源帳戶

設定監控帳戶和來源帳戶後，您可以使用這些章節中的步驟來進行管理。

內容

- 將更多來源帳戶連結至現有監控帳戶
- 移除監控帳戶與來源帳戶之間的連結
- 檢視監控帳戶的相關資訊

將更多來源帳戶連結至現有監控帳戶

請依照本節中的步驟，將其他來源帳戶的連結新增至現有的監控帳戶。

每個來源帳戶最多可連結至五個監控帳戶。每個監控帳戶可以連結到多達 100,000 個來源帳戶。

若要管理來源帳戶，您必須擁有特定許可。如需詳細資訊，請參閱 必要的許可。

將更多來源帳戶新增至監控帳戶

1. 登入至監控帳戶。
2. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
3. 在左側的導覽窗格中，選擇設定。
4. 在 Monitoring account configuration (監控帳戶組態) 中，選擇 Manage source accounts (管理來源帳戶)。
5. 選擇 Configuration policy (組態政策) 索引標籤。
6. 在 Configuration policy (組態政策) 方塊中，在 Principal (主體) 命令列中新增新的來源帳戶 ID。

   例如，假設 Principal (主體) 命令列目前為下列內容：

   ```
   "Principal": {"AWS": ["111111111111", "222222222222"]}
   ```

   若要新增 999999999999 作為第三個來源帳戶，請將該命令列編輯為下列內容：

```
"Principal": {"AWS": ["111111111111", "222222222222", "999999999999"]}
```

7. 選擇更新。
8. 選擇 Configuration details (組態詳細資訊) 索引標籤。
9. 選擇監控帳戶接收器 ARN 旁邊的複製圖示。
10. 登入您想要用作新來源帳戶的帳戶。
11. 貼上您在步驟 9 中複製的監視帳戶接收器 ARN。

    您會看到 CloudWatch 設定頁面，其中已填入一些資訊。
12. 在選取的資料中，選擇此來源帳戶是否將日誌、指標、追蹤和 Application Insights - 應用程式資料傳送至其連結的監控帳戶。
13. 請勿更改 Enter monitoring account configuration ARN (輸入監控帳戶組態 ARN) 中的 ARN。
14. Define a label to identify your source account (定義標來識別來源帳戶) 區段會預先填入監控帳戶的標籤選項。或者，您也可以選擇 Edit (編輯) 來進行變更。
15. 選擇 Link (連結)。
16. 在方塊中輸入 **`Confirm`**，然後選擇 Confirm (確認)。

移除監控帳戶與來源帳戶之間的連結

請依照本節中的步驟，停止將某個來源帳戶內的資料傳送至監控帳戶。

您必須擁有管理來源帳戶所需的許可，才能完成此任務。如需詳細資訊，請參閱 必要的許可。

移除來源帳戶與監控帳戶之間的連結

1. 登入來源帳戶。
2. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
3. 在左側的導覽窗格中，選擇設定。
4. 在 Source account information (來源帳戶資訊) 中，選擇 View monitoring accounts (檢視監控帳戶)。
5. 選取您想要停止共用資料的監控帳戶旁邊的核取方塊。
6. 選擇 Stop sharing data (停止共享資料)，Confirm (確認)。
7. 登入至監控帳戶。
8. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。

9. 選擇設定。
10. 在 Monitoring account information (監控帳戶資訊) 中，選擇 View configuration (檢視組態)。
11. 在 Policy (政策) 方塊中，從 Principal (主體) 命令列中刪除來源帳戶 ID，然後選擇 Update (更新)。

檢視監控帳戶的相關資訊

請依照本節中的步驟，檢視監控帳戶的跨帳戶設定。

若要管理監控帳戶，您必須擁有特定許可。如需詳細資訊，請參閱 必要的許可。

管理監控帳戶

1. 登入至監控帳戶。
2. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
3. 在左側的導覽窗格中，選擇設定。
4. 在 Monitoring account configuration (監控帳戶組態) 中，選擇 Manage source accounts (管理來源帳戶)。
5. 若要檢視允許此帳戶成為監控帳戶的 Observability Access Manager 政策，請選擇 Configuration policy (組態政策) 索引標籤。
6. 若要檢視連結至此監控帳戶的來源帳戶，請選擇 Linked source accounts (連結的來源帳戶) 索引標籤。
7. 若要檢視監控帳戶接收器 ARN，以及此監控帳戶在連結的來源帳戶中可以檢視的資料類型，請選擇 Linked source accounts (連結的來源帳戶) 索引標籤。

從其他資料來源中查詢指標

您可以使 CloudWatch 用從其他資料來源查詢、視覺化和建立指標的警示。若要這麼做，請連線 CloudWatch 到其他資料來源。這可讓您在 CloudWatch 主控台內獲得單一整合的監視體驗。無論資料儲存在何處，您都可以統一檢視基礎架構和應用程式指標，協助您更快找出問題並進行解決。

使用 CloudWatch 精靈連線至資料來源之後， CloudWatch 建立用於部署和設定函數的 AWS CloudFormation 堆疊。 AWS Lambda 每次查詢資料來源時，該 Lambda 函數都會隨需執行。查 CloudWatch 詢產生器會即時顯示可查詢的元素清單，例如量度、資料表、欄位或標籤。當您進行選擇時，查詢建置器會以所選來源的母語預先填入查詢。

CloudWatch 提供引導式精靈，讓您連線至下列資料來源。對於這些資料來源，您需要提供基本資訊以識別資料來源和憑證。您也可以透過建立自己的 Lambda 函數，手動建立其他資料來源的連接器。

- Amazon OpenSearch 服務 — 從您的 OpenSearch 服務日誌和追蹤衍生指標。
- Amazon Managed Service for Prometheus – 使用 PromQL 查詢這些指標。
- Amazon RDS for MySQL – 使用 SQL 將存放在 Amazon RDS 資料表中的資料轉換為指標。
- Amazon RDS for PostgreSQL – 使用 SQL 將存放在 Amazon RDS 資料表中的資料轉換為指標。
- Amazon S3 CSV 檔案 – 顯示 Amazon S3 儲存貯體中存放的 CSV 檔案中的指標資料。
- Microsoft Azure Monitor – 從 Microsoft Azure Monitor 帳戶查詢指標。
- Prometheus – 使用 PromQL 查詢這些指標。

建立資料來源的連接器之後，請參閱 從另一個資料來源建立指標圖表 以取得有關從資料來源中繪製指標圖形的資訊。如需有關從資料來源中設定指標警示的相關資訊，請參閱 根據連線的資料來源建立警示。

主題

- 管理資料來源的存取
- 使用精靈連線至預先建立的資料來源
- 建立資料來源的自訂連接器
- 使用自訂資料來源
- 刪除資料來源的連接器

管理資料來源的存取

CloudWatch 用 AWS CloudFormation 於在您的帳戶中建立必要的資源。建議您在授予`CreateStack`權限給 IAM 使用者時，使用此`cloudformation:TemplateUrl`條件來控制對 AWS CloudFormation 範本的存取。

> **Warning**
>
> 被授予資料來源調用許可的任何使用者都可以從該資料來源中查詢指標，即使該使用者沒有資料來源的直接 IAM 許可。例如，如果您將 Amazon Managed Service for Prometheus 資料來源 Lambda 函數的 `lambda:InvokeFunction` 許可授予給使用者，該使用者將能夠從對應的 Amazon Managed Service for Prometheus 工作區中查詢指標，即使您沒有授予該工作區的直接 IAM 存取權。

您可以在 CloudWatch 「設定主控台」的「建立堆疊」頁面上找到資料來源的範本 URL。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect" : "Allow",
      "Action" : [ "cloudformation:CreateStack" ],
      "Resource" : "*",
      "Condition" : {
        "StringEquals" : {
          "cloudformation:TemplateUrl" : [ data-source-template-url ]
        }
      }
    }
  ]
}
```

如需控制 AWS CloudFormation 存取權的詳細資訊，請參閱使用 AWS Identity and Access Management 控制存取

使用精靈連線至預先建立的資料來源

本主題提供使用精靈連線 CloudWatch 至下列資料來源的指示。

- Amazon OpenSearch 服務
- Amazon Managed Service for Prometheus
- Amazon RDS for MySQL
- Amazon RDS for PostgreSQL
- Amazon S3 CSV 檔案
- Microsoft Azure Monitor
- Prometheus

本節後面的小節包含有關管理和查詢這些資料來源的備註。

建立資料來源的連接器

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇設定。
3. 選擇指標資料來源索引標籤。
4. 選擇 Create data source (建立資料來源)。
5. 選取需要的來源，然後選擇下一步。
6. 輸入資料來源的名稱。
7. 根據選擇的資料來源，輸入其他必要資訊。這可能包括用於存取資料來源的憑證和資料來源識別資訊，例如 Prometheus 工作區名稱、資料庫名稱或 Amazon S3 儲存貯體名稱。對於 AWS 服務，精靈會探索資源，並將其填入選取下拉式清單中。

   如需有關您正在使用之資料來源的詳細說明，請參閱此程序後的小節。
8. 若要 CloudWatch 連線至 VPC 中的資料來源，請選擇 [使用 VPC]，然後選取要使用的 VPC。然後選取子網路和安全群組。
9. 選擇 [我確認 AWS CloudFormation 將建立 IAM 資源]。此資源是 Lambda 函數執行角色。
10. 選擇 Create data source (建立資料來源)。

    您剛剛添加的新源代碼在 AWS CloudFormation 堆棧完成創建之前不會出現。若要檢查進度，您可以選擇 [檢視我的 CloudFormation 堆疊狀態]。或者，可以選擇重新整理圖示來更新此清單。

    當新資料來源出現在此清單中時，即已準備好可供使用。您可以從 CloudWatch 指標中選擇「查詢」，開始使用它進行查詢。如需詳細資訊，請參閱 從另一個資料來源建立指標圖表。

Amazon Managed Service for Prometheus

更新資料來源組態

- 可執行下列操作來手動更新資料來源：
  - 若要更新 Amazon Managed Service for Prometheus 工作區 ID，請更新資料來源連接器 Lambda 函數的 `AMAZON_PROMETHEUS_WORKSPACE_ID` 環境變數。
  - 若要更新 VPC 組態，請參閱設定 VPC 存取 (主控台) 以取得詳細資訊。

查詢資料來源

- 查詢 Amazon Managed Service for Prometheus 時，在多來源查詢索引標籤中選取資料來源並選取 Amazon Managed Service for Prometheus 連接器後，可以使用查詢助手來查找指標和標籤，並提供簡單的 PromQL 查詢。也可以使用 PromQL 查詢編輯器來建置 PromQL 查詢。
- CloudWatch 資料來源連接器不支援多行查詢。執行查詢時，或當您使用查詢建立警示或儀表板小工具時，每個換行符都會取代為空格。在某些情況下，這可能會使查詢無效。例如，如果查詢包含單行註釋，它將無效。如果您嘗試使用命令列或「基礎設施即程式碼」的多行查詢建立儀表板或警示，API 將以剖析錯誤拒絕該動作。

Amazon OpenSearch 服務

建立資料來源

如果 OpenSearch 網域已啟用 FGAC，您必須將連接器 Lambda 函數的執行角色對應至服務中 OpenSearch 的使用者。如需詳細資訊，請參閱 OpenSearch 服務說明文件中管理權限中的〈將使用者對應至角色〉一節。

如果您的 OpenSearch 網域只能在 Virtual Private Cloud (VPC) (VPC) 中存取，則需要在名為的 Lambda 函`AMAZON_OPENSEARCH_ENDPOINT`數中手動包含新的環境變數。此變數的值應為 OpenSearch 端點的根網域。您可以移除`https://` OpenSearch 服務主控台中列`<region>.es.amazonaws.com`出的網域端點，以取得此根網域。例如，如果您的網域端點是`https://sample-domain.us-east-1.es.amazonaws.com`，則根網域會是`sample-domain`。

更新資料來源

- 可執行下列操作來手動更新資料來源：

- 若要更新 OpenSearch 服務網域，請更新資料來源連接器 Lambda 函數的AMAZON_OPENSEARCH_DOMAIN_NAME環境變數。
- 若要更新 VPC 組態，請參閱設定 VPC 存取 (主控台) 以取得詳細資訊。

查詢資料來源

- 查詢 OpenSearch Service 時，在「多重來源查詢」索引標籤中選取資料來源之後，請執行下列動作：
  - 選取要查詢的索引。
  - 選取指標名稱 (文件中的任何數字欄位) 和「統計」。
  - 選取時間軸 (文件中的任何日期欄位)。
  - 選取要套用的篩選條件 (文件中的任何字串欄位)。
  - 選擇圖形查詢。

Amazon RDS for PostgreSQL 和 Amazon RDS for MySQL

建立資料來源

- 如果只能在 VPC 中存取資料來源，則必須包含連接器的 VPC 組態，如 使用精靈連線至預先建立的資料來源 中所述。如果資料來源要連線到 VPC 以取得憑證，則必須在 VPC 中設定端點。如需詳細資訊，請參閱使用 AWS Secrets Manager VPC 端點。

  此外，您必須為 Amazon RDS 服務建立 VPC 端點。如需詳細資訊，請參閱 Amazon RDS API 和介面 VPC 端點 (AWS PrivateLink)。

更新資料來源

- 可執行下列操作來手動更新資料來源：
  - 若要更新資料庫執行個體，請更新資料來源連接器 Lambda 函數的 RDS_INSTANCE 環境變數。
  - 若要更新用於連線至 Amazon RDS 的使用者名稱和密碼，請使用 AWS Secrets Manager。您可以在資料來源 Lambda 函數的環境變數 RDS_SECRET 中找到用於資料來源之密碼的 ARN。如需有關在 AWS Secrets Manager中更新密碼的詳細資訊，請參閱修改 AWS Secrets Manager 密碼。
  - 若要更新 VPC 組態，請參閱設定 VPC 存取 (主控台) 以取得詳細資訊。

查詢資料來源

- 查詢 Amazon RDS 時，在多來源查詢索引標籤中選取資料來源並選取 Amazon RDS 連接器後，可以使用資料庫探索器來檢視可用的資料庫、資料表和資料欄。也可以使用 SQL 編輯器建立 SQL 查詢。

  可以在查詢中使用下列變數：

  - `$start.iso` – ISO 日期格式的開始時間
  - `$end.iso` – ISO 日期格式的結束時間
  - `$period` – 所選時段 (以秒為單位)

  例如，您可以查詢 `SELECT value, timestamp FROM table WHERE timestamp BETWEEN $start.iso and $end.iso`
- CloudWatch 資料來源連接器不支援多行查詢。執行查詢時，或當您使用查詢建立警示或儀表板小工具時，每個換行符都會取代為空格。在某些情況下，這可能會使查詢無效。例如，如果查詢包含單行註釋，它將無效。如果您嘗試使用命令列或「基礎設施即程式碼」的多行查詢建立儀表板或警示，API 將以剖析錯誤拒絕該動作。

> **Note**
>
> 如果在結果中找不到日期欄位，則每個數值欄位的值會相加為單個值，並在提供的時間範圍內繪製。如果時間戳記與中選取的期間不一致 CloudWatch，系統會使用中的期間自動彙總資料，`SUM`並與中 CloudWatch的期間對齊。

Amazon S3 CSV 檔案

查詢資料來源

- 查詢 Amazon S3 CSV 檔案時，在多來源查詢索引標籤中選取資料來源並選取 Amazon S3 連接器之後，可選取 Amazon S3 儲存貯體和金鑰。

  CSV 檔案必須以下列方式格式化：

  - 時間戳記必須是第一欄。
  - 該表必須有一個標題行。標頭是用來命名您的指標。時間戳記欄的標題將會被忽略，只會使用量度欄的標題。
  - 時間戳記必須為 ISO 日期格式。

- 量度必須是數值欄位。

```
Timestamp, Metric-1, Metric-2, ...
```

以下是範例：

timestamp	CPU (%)	Memory (%) (記憶體 (%))	儲存 (%)
2023-11-23T17:09:41+00:00	1	2	3
2023-11-23T17:04:41+00:00	4	5	6
2023-11-23T16:59:41+00:00	7	8	9
2023-11-23T16:54:41+00:00	10	11	12

> **Note**
>
> 如果未提供時間戳記，則每個指標的值會相加為單個值，並在提供的時間範圍內繪製。如果時間戳記與中選取的期間不一致 CloudWatch，系統會使用中的期間自動彙總資料，SUM並與中CloudWatch的期間對齊。

Microsoft Azure Monitor

建立資料來源

- 您必須提供租用戶 ID、用戶端 ID 及用戶端密碼，才能連線至 Microsoft Azure Monitor。認證將儲存在中 AWS Secrets Manager。如需詳細資訊，請參閱 Microsoft 文件中的建立可存取資源的 Microsoft Entra 應用程式和服務主體。

更新資料來源

- 可執行下列操作來手動更新資料來源：
  - 若要更新用於連線至 Azure Monitor 的租用戶 ID、用戶端 ID 和用戶端密碼，您可以在資料來源 Lambda 函數中找到用於資料來源之密碼的 ARN 作為 AZURE_CLIENT_SECRET 環境變數。如需更新中密碼的詳細資訊 AWS Secrets Manager，請參閱修改 AWS Secrets Manager 密碼。

查詢資料來源

- 查詢 Azure Monitor 時，在多來源查詢索引標籤中選取資料來源並選取 Azure Monitor 連接器之後，可以指定 Azure 訂閱以及資源群組和資源。然後，可以選取指標命名空間、指標和彙總，並依維度進行篩選。

Prometheus

建立資料來源

- 必須提供 Prometheus 端點以及查詢 Prometheus 所需的使用者和密碼。認證將儲存在中 AWS Secrets Manager。
- 如果只能在 VPC 中存取資料來源，則必須包含連接器的 VPC 組態，如 使用精靈連線至預先建立的資料來源 中所述。如果要連線資料來源以取得憑證，則必須在 VPC 中設定端點。如需詳細資訊，請參閱使用 AWS Secrets Manager VPC 端點。

更新資料來源組態

- 可執行下列操作來手動更新資料來源：
  - 若要更新 Prometheus 端點，請在資料來源 Lambda 函數中將新端點指定為 PROMETHEUS_API_ENDPOINT 環境變數。
  - 若要更新用於連線至 Prometheus 的使用者名稱和密碼，您可以在資料來源 Lambda 函數中找到用於資料來源之密碼的 ARN 作為 PROMETHEUS_API_SECRET 環境變數。如需更新中密碼的詳細資訊 AWS Secrets Manager，請參閱修改 AWS Secrets Manager 密碼。
  - 若要更新 VPC 組態，請參閱設定 VPC 存取 (主控台) 以取得詳細資訊。

查詢資料來源

> **Important**
>
> Prometheus CloudWatch 度量類型與指標不同，並且可以通過 Prometheus 獲得的許多指標都是根據設計累積的。當您查詢 Prometheus 測量結果時， CloudWatch 不會將任何額外的轉換套用至資料：如果您只指定測量結果名稱或標籤，則顯示的值將會是累計值。如需詳細資訊，請參閱 Prometheus 文件中的指標類型。
> 若要將 Prometheus 指標資料視為離散值 (例如 CloudWatch 指標)，您必須先編輯查詢，然後再執行查詢。例如，您可能需要透過 Prometheus 指標名稱新增對 rate 函數的呼叫。如需有關 rate 函數和其他 Prometheus 函數的文件，請參閱 Prometheus 文件中的 rate()。

CloudWatch 資料來源連接器不支援多行查詢。執行查詢時，或當您使用查詢建立警示或儀表板小工具時，每個換行符都會取代為空格。在某些情況下，這可能會使查詢無效。例如，如果查詢包含單行註釋，它將無效。如果您嘗試使用命令列或「基礎設施即程式碼」的多行查詢建立儀表板或警示，API 將以剖析錯誤拒絕該動作。

可用更新的通知

Amazon 可能會不時通知您，建議您使用較新的可用版本更新連接器，並提供如何執行此操作的指示。

建立資料來源的自訂連接器

若要將自訂資料來源連線到 CloudWatch，您有兩種選擇：

- 使用提供的範例範本開始使 CloudWatch 用。您可以使用任何一個 JavaScript 或 Python 與此模板。這些範本包含範例 Lambda 程式碼，這些程式碼在您建立 Lambda 函數時很有用。然後，可以從範本中修改 Lambda 函數，以連線到自訂資料來源。
- 從頭開始建立 AWS Lambda 函數，以實作資料來源連接器、資料查詢以及準備供使用的時間序列 CloudWatch。如果需要，此函數必須預先彙總或合併資料點，並對齊要與之相容的週期和時間戳記。 CloudWatch

內容

- 使用範本
- 從頭開始建立自訂資料來源
  - 步驟 1：建立函數
    - GetMetricData 事件

    - DescribeGetMetricData 事件
    - CloudWatch 警報的重要注意事項
    - (選擇性) 用 AWS Secrets Manager 來儲存認證
    - (選用) 連線至 VPC 中的資料來源
- 步驟 2：建立 Lambda 許可政策
- 步驟 3：將資源標籤附接至 Lambda 函數

使用範本

使用範本可建立範例 Lambda 函數，並協助您更快地建置自訂連接器。這些範例函數為建置自訂連接器所涉及的許多常見案例提供範例程式碼。可以在使用範本建立連接器之後檢查 Lambda 程式碼，然後修改它以用於連線至資料來源。

此外，如果您使用範本，請注 CloudWatch 意建立 Lambda 許可政策，並將資源標籤附加至 Lambda 函數。

使用範本建立自訂資料來源的連接器

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇設定。
3. 選擇指標資料來源索引標籤。
4. 選擇 Create data source (建立資料來源)。
5. 選擇自訂 - 開始使用範本的選項按鈕，然後選擇下一步。
6. 輸入資料來源的名稱。
7. 選取列出的其中一個範本。
8. 選擇 Node.js 或 Python。
9. 選擇 Create data source (建立資料來源)。

   在 AWS CloudFormation 堆疊完成建立之前，您剛新增的新自訂來源才會出現。若要檢查進度，您可以選擇 [檢視我的 CloudFormation 堆疊狀態]。或者，可以選擇重新整理圖示來更新此清單。

   當新資料來源出現在此清單中時，就可以在主控台中進行測試並修改。
10. (選用) 若要在主控台中查詢來自此來源的測試資料，請遵循 從另一個資料來源建立指標圖表 中的指示。
11. 根據需求修改 Lambda 函數。

a. 在導覽窗格中，選擇設定。

b. 選擇指標資料來源索引標籤。

c. 針對您想要修改的來源，選擇在 Lambda 主控台中檢視。

您現在可以修改函數以存取資料來源。如需詳細資訊，請參閱 步驟 1：建立函數。

> **Note**
>
> 透過使用範本，當您撰寫 Lambda 函數時，無需遵循 步驟 2：建立 Lambda 許可政策 或 步驟 3：將資源標籤附接至 Lambda 函數 中的指示。這些步驟是由執行的， CloudWatch 因為您使用了範本。

從頭開始建立自訂資料來源

請依照本節中的步驟建立連線 CloudWatch 至資料來源的 Lambda 函數。

步驟 1：建立函數

自訂資料來源連接器必須支援來自的`GetMetricData`事件 CloudWatch。或者，您也可以實作`DescribeGetMetricData`事件，以便在 CloudWatch 主控台中向使用者提供如何使用連接器的說明文件。`DescribeGetMetricData`回應也可以用來設 CloudWatch 定自訂查詢建置器中使用的預設值。

CloudWatch 提供程式碼片段做為範例，以協助您開始使用。如需詳細資訊，請參閱 https://github.com/aws-samples/ 的範例儲存庫cloudwatch-data-source-samples。

限制條件

- 來自 Lambda 的回應必須小於 6 Mb。如果回應超過 6 Mb，則 `GetMetricData` 回應會將 Lambda 函數標記為 `InternalError`，且不會傳回任何資料。
- Lambda 函數必須在 10 秒內完成其執行，以達到視覺化和儀表板目的；或在 4.5 秒內完成執行，起到警示作用。如果執行時間超過該時間，則 `GetMetricData` 回應會將 Lambda 函數標記為 `InternalError`，且不會傳回任何資料。
- Lambda 函數必須使用 Epoch 時間戳記 (以秒為單位) 來傳送其輸出。

- 如果 Lambda 函數未重新取樣資料，而是傳回與 CloudWatch 使用者要求的開始時間和週期長度不相符的資料，則會忽略該資料。 CloudWatch會從任何視覺效果或警示中捨棄額外資料。任何非開始時間和結束時間之間的資料也會被捨棄。

  例如，如果使用者需要間隔為 5 分鐘從上午 10 點到 11 點之間的資料，則傳回資料的有效時間範圍為「10:00:00 到 10:04:59」和「10:05:00 到 10:09:59」。必須傳回包含 `10:00 value1`、`10:05 value2` 等的時間序列。例如，如果函數傳回 `10:03 valueX`，它會被丟棄，因為 10:03 與請求的開始時間和間隔不符。
- CloudWatch 資料來源連接器不支援多行查詢。執行查詢時，或當您使用查詢建立警示或儀表板小工具時，每個換行符都會取代為空格。在某些情況下，這可能會使查詢無效。

GetMetricData 事件

請求承載

以下是作為 Lambda 函數的輸入而傳送的 `GetMetricData` 請求承載範例。

```
{
  "EventType": "GetMetricData",
  "GetMetricDataRequest": {
    "StartTime": 1697060700,
    "EndTime": 1697061600,
    "Period": 300,
    "Arguments": ["serviceregistry_external_http_requests{host_cluster!=\"prod\"}"]
  }
}
```

- StartTime— 指定要傳回之最早資料的時間戳記。類型是時間戳記 Epoch 秒。
- EndTime— 指定要傳回之最新資料的時間戳記。類型是時間戳記 Epoch 秒。
- 間隔 – 指標資料的每個彙總所代表的秒數。最小值為 60 秒。類型為秒。
- 引數 – 傳遞至 Lambda 指標數學運算式的引數陣列。如需有關傳遞引數的詳細資訊，請參閱 如何將引數傳遞給 Lambda 函數。

回應承載

以下是 Lambda 函數傳回的 `GetMetricData` 回應承載範例。

```
{
```

```
   "MetricDataResults": [
      {
         "StatusCode": "Complete",
         "Label": "CPUUtilization",
         "Timestamps": [ 1697060700, 1697061000, 1697061300 ],
         "Values": [ 15000, 14000, 16000 ]
      }
   ]
}
```

回應承載將包含 MetricDataResults 欄位或 Error 欄位，但不能同時包含兩者。

MetricDataResults 欄位是一系列 MetricDataResult 類型的時間序列欄位。每個時間序列欄位可以包含下列欄位。

- StatusCode— (選用) Complete 表示傳回要求時間範圍內的所有資料點。 PartialData表示傳回一組不完整的資料點。若省略，則預設值為 Complete。

  有效值：Complete | InternalError | PartialData | Forbidden
- 訊息 – 可選訊息清單，其中包含有關傳回資料的其他資訊。

  類型：具有Code和Value字符串的MessageData對象數組。
- 標籤 – 與資料相關聯的人類可讀標籤。

  類型：字串
- 時間戳記 – 資料點的時間戳記，格式為 Epoch 時間。時間戳記的數目始終符合值的數目，而且 Timestamps[x] 的值為 Values[x]。

  類型：時間戳記陣列
- 值 – 指標的資料點值，對應於 Timestamps。值的數目始終符合時間戳記的數目，而且 Timestamps[x] 的值為 Values[x]。

  類型：雙精度陣列

如需有關 Error 物件的詳細資訊，請參閱下列各節。

錯誤回應格式

您可以選擇使用錯誤回應來提供有關錯誤的詳細資訊。建議您在發生驗證錯誤時傳回「程式碼驗證」錯誤，例如當參數遺失或類型錯誤時。

以下是 Lambda 函數想要引發 GetMetricData 驗證例外狀況時的回應範例。

```
{
   "Error": {
      "Code": "Validation",
      "Value": "Invalid Prometheus cluster"
   }
}
```

以下是當 Lambda 函數指出由於存取問題而無法傳回資料時的回應範例。回應會轉譯成單一時間序列，且狀態碼為 Forbidden。

```
{
   "Error": {
      "Code": "Forbidden",
      "Value": "Unable to access ..."
   }
}
```

以下是 Lambda 函數引發整體 InternalError 例外狀況時的範例，它會轉譯為單一時間序列，且具有狀態碼 InternalError 和訊息。每當錯誤代碼具有Validation或以外的值時Forbidden，CloudWatch 假設它是一個通用的內部錯誤。

```
{
   "Error": {
      "Code": "PrometheusClusterUnreachable",
      "Value": "Unable to communicate with the cluster"
   }
}
```

DescribeGetMetricData 事件

請求承載

以下是 DescribeGetMetricData 請求承載的範例。

```
{
  "EventType": "DescribeGetMetricData"
}
```

回應承載

以下是 DescribeGetMetricData 回應承載的範例。

```
{
    "Description": "Data source connector",
    "ArgumentDefaults": [{
        Value: "default value"
     }]
}
```

- 描述 – 如何使用資料來源連接器的描述。此說明將出現在 CloudWatch 控制台中。支援 Markdown。

  類型：字串
- ArgumentDefaults— 使用預先填入自訂資料來源建置器的可選引數預設值陣列。

  如果返回[{ Value: "default value 1"}, { Value: 10}]，則 CloudWatch 控制台中的查詢生成器顯示兩個輸入，第一個輸入為「默認值 1」，第二個輸入為 10。

  如果未提供 ArgumentDefaults，則會顯示單一輸入，且類型預設為 String。

  類型：包含「值」和「類型」的物件陣列。
- 錯誤 – (選用) 錯誤欄位可包含在任何回應中。可在 GetMetricData 事件 中看到範例。

CloudWatch 警報的重要注意事項

如果您要使用資料來源設定 CloudWatch 警示，則應將其設定為每分鐘將時間戳記報告資料至 CloudWatch。如需有關從連線的資料來源中建立指標警示的詳細資訊和其他考量，請參閱 根據連線的資料來源建立警示。

(選擇性) 用 AWS Secrets Manager 來儲存認證

如果您的 Lambda 函數需要使用認證來存取資料來源，建議您使 AWS Secrets Manager 用儲存這些認證，而不是將其硬式編碼到 Lambda 函數中。如需 AWS Secrets Manager 搭配 Lambda 搭配使用的詳細資訊，請參閱在 AWS Lambda 函數中使用 AWS Secrets Manager 密碼。

(選用) 連線至 VPC 中的資料來源

如果資料來源位於 Amazon Virtual Private Cloud 所管理的 VPC 中，則必須設定 Lambda 函數才能存取它。如需詳細資訊，請參閱將傳出網路連線至 VPC 中的資源。

可能還需要設定 VPC 服務端點，才能存取諸如 AWS Secrets Manager等服務。如需詳細資訊，請參閱使用介面 VPC 端點存取 AWS 服務。

步驟 2：建立 Lambda 許可政策

您必須使用建立政策陳述式來 CloudWatch 授與使用您所建立之 Lambda 函數的權限。您可以使用 AWS CLI 或 Lambda 主控台建立政策陳述式。

若要使用 AWS CLI 建立政策陳述式

- 輸入以下命令。將 *123456789012* 取代為您的帳戶識別碼、以您的 Lambda 函數*my-data-source-function*的名稱取代，並以任意唯一值取代 *MyDataSource-DataSourcePermission 12* 34。

```
aws lambda add-permission --function-name my-data-source-function --statement-
id MyDataSource-DataSourcePermission1234 --action lambda:InvokeFunction --principal
 lambda.datasource.cloudwatch.amazonaws.com --source-account 123456789012
```

步驟 3：將資源標籤附接至 Lambda 函數

主 CloudWatch 控台會使用標籤來判斷哪些 Lambda 函數是資料來源連接器。當您使用其中一個精靈建立資料來源時，標籤會由設定該資料來源的 AWS CloudFormation 堆疊自動套用。當您自行建立資料來源時，可以針對 Lambda 函數使用下列標籤。這會讓您的連接器在您查詢指標時出現在 CloudWatch 主控台的 [資料來源] 下拉式清單中。

- 索引鍵為 `cloudwatch:datasource` 和值為 `custom` 的標籤。

使用自訂資料來源

建立資料來源之後，您可以使用其來查詢來自該來源的資料，以便將其視覺化並設定警示。如果使用範本來建立自訂資料來源連接器，或已新增 步驟 3：將資源標籤附接至 Lambda 函數 中列出的標籤，則可以遵循 從另一個資料來源建立指標圖表 中的步驟進行查詢。

也可以使用指標數學函數 LAMBDA 進行查詢，如下節所述。

如需有關從資料來源中建立指標警示的相關資訊，請參閱 根據連線的資料來源建立警示。

如何將引數傳遞給 Lambda 函數

將引數傳遞至自訂資料來源的建議方式是在查詢資料來源時使用 CloudWatch 主控台中的查詢產生器。

您也可以在 CloudWatch 指標數學中使用新LAMBDA運算式，使用 Lambda 函數從資料來源擷取資料。

```
LAMBDA("LambdaFunctionName" [, optional-arg]*)
```

optional-arg 最多為 20 個字串、數字或布林值。例如，param、3.14 或 true。

> **Note**
>
> CloudWatch 資料來源連接器不支援多行字串。執行查詢時，或當您使用查詢建立警示或儀表板小工具時，每個換行符都會取代為空格。在某些情況下，這可能會使查詢無效。

使用 LAMBDA 指標數學函數時，可以提供函數名稱 ("MyFunction")。如果資源政策允許，也可以使用特定版本的函數 ("MyFunction:22") 或 Lambda 函數別名 ("MyFunction:MyAlias")。您無法使用 *

以下是呼叫 LAMBDA 函數的一些範例。

```
LAMBDA("AmazonOpenSearchDataSource", "MyDomain", "some-query")
```

```
LAMBDA("MyCustomDataSource", true, "fuzzy", 99.9)
```

LAMBDA 指標數學函數會傳回時間序列清單，它可傳回給請求者，或與其他指標數學函數結合使用。以下是 LAMBDA 與其他指標數學函數結合使用的範例。

```
FILL(LAMBDA("AmazonOpenSearchDataSource", "MyDomain", "some-query"), 0)
```

刪除資料來源的連接器

若要刪除資料來源的連接器，請遵循本節中的指示。

若要刪除資料來源的連接器

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇設定。
3. 選擇指標資料來源索引標籤。
4. CloudFormation在您要刪除的資料來源列中選擇「管理」。

   您將被帶到 AWS CloudFormation 控制台。

5. 在具有資料來源名稱的區段中，選擇刪除。
6. 在確認彈出視窗中，選擇刪除。

使用 CloudWatch 代理程式收集指標、記錄和追蹤

統一的 CloudWatch 代理程式可讓您執行下列作業：

- 從 Amazon EC2 執行個體跨作業系統收集內部系統層級指標。除了 EC2 執行個體的指標外，指標還可以包含訪客指標。可收集的其他指標會在 CloudWatch代理程式收集的測量結果中列出。
- 從現場部署伺服器收集系統層級指標。這些可能包括混合式環境中的伺服器，以及不受管理的伺服器 AWS。
- 使用 `StatsD` 和 `collectd` 通訊協定從您的應用程式或服務擷取自訂指標。Linux 伺服器和執行 Windows 的伺服器都支援 `StatsD`。`collectd` 則僅有 Linux 伺服器支援。
- 從 Amazon EC2 執行個體和執行 Linux 或 Windows Server 的內部部署伺服器收集日誌。

> **Note**
>
> CloudWatch 代理程式不支援從 FIFO 管道收集記錄檔。

- 版本 1.300031.0 及更高版本可用於啟 CloudWatch 用應用程式訊號。如需詳細資訊，請參閱 Application Signals。
- 版本 1.300025.0 及更高版本可以從 OpenTelemetryX-Ray 客戶端 SDK 收集跟踪，並將其發送到 X-Ray。

  使用 CloudWatch 代理程式可讓您收集追蹤，而不需要執行個別的追蹤收集常駐程式，有助於減少執行和管理的代理程式數目。

您可以儲存和檢視您透過 CloudWatch 代理程式收集的指標， CloudWatch 就像您可以處理任何其他 CloudWatch 指標一樣。雖然您可以在設定 CloudWatch 代理程式時指定不同的命名空間`CWAgent`，但代理程式收集的測量結果預設命名空間為。

統一 CloudWatch 代理程式收集的日誌會處理並儲存在 Amazon CloudWatch Logs 中，就像舊版日誌代理程式收集的 CloudWatch 日誌一樣。如需 CloudWatch 日誌定價的相關資訊，請參閱 Amazon CloudWatch 定價。

CloudWatch 代理程式收集的指標會以自訂指標計費。如需有關 CloudWatch 指標定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

CloudWatch 代理程式是在 MIT 授權下開放原始碼，並託管於 GitHub. 如果您想要建置、自訂或貢獻 CloudWatch 代理程式，請參閱 GitHub 儲存庫以取得最新指示。如果您認為自己發現了潛在的安全問

題，請不要將其發佈到 GitHub 或任何公共論壇上。相反，請直接按照漏洞報告或電子郵件 AWS 安全中的說明進行操作。

本節中的步驟說明如何在 Amazon EC2 執行個體和現場部署伺服器上安裝統一 CloudWatch 代理程式。如需 CloudWatch 代理程式可收集之測量結果的詳細資訊，請參閱 CloudWatch代理程式收集的測量結果。

支援的作業系統

下列作業系統的 x86-64 架構支援 CloudWatch 代理程式。此處所列出每個主要版本的所有次要版本更新也支援此代理程式。

- Amazon Linux 2023
- Amazon Linux 2
- Ubuntu 服務器版本 23.10，22.04，20.04，18.04，16.04 和 14.04
- CentOS 9、8 和 7 版
- Red Hat Enterprise Linux (RHEL) 9、8 和 7 版
- Debian 版本 12、11 和 10 版
- SUSE Linux Enterprise Server (SLES) 15 和 12 版
- 甲骨文版本 9、8 和 7
- AlmaLinux 版本 9 和 8
- Rocky Linux 版本 9 和 8
- 下面的 macOS 計算機:EC2 M1 Mac1 實例, 和計算機運行 macOS 14 (索諾瑪), macOS 13 (文圖拉), 和 macOS 12 (蒙特雷)
- 視窗服務器 2022，視窗服務器 2019 和視窗服務器 2016 的 64 位版本
- 64 位元 Windows 10

下列作業系統上的 AMD64 架構都支援代理程式。此處所列出每個主要版本的所有次要版本更新也支援此代理程式。

- Amazon Linux 2023
- Amazon Linux 2
- Ubuntu 服務器版本 23.10，22.04，20.04，18.04 和 16.04
- CentOS 9 和 8 版
- Red Hat Enterprise Linux (RHEL) 9、8 和 7 版

- Debian 版本 12、11 和 10 版
- SUSE Linux Enterprise Server 15
- 下面的 macOS 計算機:macOS 14 (索諾瑪), 蘋果系統 13 (文圖拉), 和 macOS 12 (蒙特雷)

安裝程序概觀

您可以使用命令列手動下載並安裝 CloudWatch 代理程式，也可以將其與 SSM 整合。使用其中一種方法安裝 CloudWatch 代理程式的一般流程如下：

1. 建立 IAM 角色或使用者，讓代理程式從伺服器收集指標，並選擇性地與之整合 AWS Systems Manager。
2. 下載代理程式套件。
3. 修改 CloudWatch 代理程式組態檔並指定要收集的測量結果。
4. 在您的伺服器上安裝及啟動代理程式。當您在 EC2 執行個體上安裝代理程式時，您可以連接您在步驟 1 建立的 IAM 角色。當您在內部部署伺服器上安裝代理程式時，您可以指定一個具名描述檔，其中包含您在步驟 1 中所建立 IAM 使用者的登入資料。

目錄

- 安裝 CloudWatch 代理程式
- 建立 CloudWatch 代理程式組態檔
- 使用 Amazon CloudWatch 可觀測 EKS 附加元件安裝 CloudWatch 代理程式
- CloudWatch代理程式收集的測量結果
- CloudWatch代理程式的常見案例
- 疑難排解 CloudWatch 代理

安裝 CloudWatch 代理程式

該 CloudWatch 代理程式可在 Amazon Linux 2023 和 Amazon Linux 2 中以套件形式提供。如果您使用的是這些作業系統之一，您可以輸入下列指令來安裝套件。您還必須確保附加到執行個體的 IAM 角色已CloudWatchAgentServerPolicy附加。如需詳細資訊，請參閱 建立 IAM 角色以搭配 Amazon EC2 執行個體上的 CloudWatch 代理程式使用。

```
sudo yum install amazon-cloudwatch-agent
```

在包括 Linux 和 Windows 伺服器在內的所有支援作業系統上，您都可以使用具有 Amazon S3 下載連結的命令列、使用 Amazon EC2 Systems Manager 或使用 AWS CloudFormation 範本下載和安裝 CloudWatch 代理程式。如需詳細資訊，請參閱後續章節。

目錄

- 使用命令列安裝 CloudWatch 代理程式
- 使用安裝 CloudWatch 代理程式 AWS Systems Manager
- 使用在新執行個體上安裝 CloudWatch代理程式 AWS CloudFormation
- CloudWatch 代理程式認證偏
- 驗證代 CloudWatch 理程式套件的簽章

使用命令列安裝 CloudWatch 代理程式

請使用下列主題下載、設定和安裝 CloudWatch 代理程式套件。

主題

- 使用命令列下載並設定 CloudWatch代理程式
- 建立 IAM 角色和使用者以搭配 CloudWatch 代理程式使用
- 在伺服器上安裝和執行 CloudWatch 代理程式

使用命令列下載並設定 CloudWatch代理程式

使用下列步驟下載 CloudWatch 代理程式套件、建立 IAM 角色或使用者，以及選擇性地修改通用組態檔案。

下載 CloudWatch 代理程式套件

> **Note**
>
> 若要下載 CloudWatch 代理程式，您的連線必須使用 TLS 1.2 或更新版本。

該 CloudWatch 代理程式可在 Amazon Linux 2023 和 Amazon Linux 2 中以套件形式提供。如果您使用的是此作業系統，則可以輸入下列命令來安裝套件。您還必須確保附加到執行個體的 IAM 角色已CloudWatchAgentServerPolicy附加。如需詳細資訊，請參閱 建立 IAM 角色和使用者以搭配 CloudWatch 代理程式使用。

```
sudo yum install amazon-cloudwatch-agent
```

在所有支援的作業系統上，您可以使用命令列下載並安裝 CloudWatch 代理程式。

每個下載連結都有一個一般連結及針對每個區域的連結。例如，對於 Amazon Linux 2023 和 Amazon Linux 2 和 x86-64 架構，三個有效的下載鏈接是：

- `https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`
- `https://amazoncloudwatch-agent-us-east-1.s3.us-east-1.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`
- `https://amazoncloudwatch-agent-eu-central-1.s3.eu-central-1.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`

您也可以下載有關代理程式最新變更的 README 檔案，以及指出可供下載之版本編號的檔案。這些檔案位於下列位置：

- `https://amazoncloudwatch-agent.s3.amazonaws.com/info/latest/RELEASE_NOTES` 或 `https://amazoncloudwatch-agent-region.s3.region.amazonaws.com/info/latest/RELEASE_NOTES`
- `https://amazoncloudwatch-agent.s3.amazonaws.com/info/latest/CWAGENT_VERSION` 或 `https://amazoncloudwatch-agent-region.s3.region.amazonaws.com/info/latest/CWAGENT_VERSION`

架構	平台	下載連結	簽章檔案連結
x86-64	Amazon Linux 2023 和 Amazon Linux 2	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent. 每分鐘	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/ amazon-cloudwatch-agent .rpm.sig
		https://amazoncloudwatch-agent-##. ##. 亞馬遜亞馬遜亞馬	https://amazoncloudwatch-agent-##. ##. 亞馬遜亞馬遜亞

架構	平台	下載連結	簽章檔案連結
		遜 /amd64 /最新/ amazon-cloudwatch-agent 轉	馬遜/amd64 /最新/ .rpm.sig amazon-cloudwatch-agent
x86-64	Centos	https://amazoncloudwatch-agent.s3.amazonaws.com/centos/amd64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ##. 亞馬遜amazon-cloudwatch-agent公司/百分之六十四/最新/轉	https://amazoncloudwatch-agent.s3.amazonaws.com/centos/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ##. 亞馬遜amazon-cloudwatch-agent公司/百分之六十四/最新/.rpm.sig
x86-64	Redhat	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/amd64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ##. #####/紅帽 /amd64 /最新/ amazon-cloudwatch-agent 轉	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ##. 亞馬遜amazon-cloudwatch-agent公司/紅帽 /amd64 /最新/ .rpm.sig
x86-64	SUSE	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/amd64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ##. 亞馬遜公司/蘇/amazon-cloudwatch-agent安 64 /最新/ 轉	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ##. 亞馬遜amazon-cloudwatch-agent公司/蘇聯/安德 64 /最新/ .rpm.sig

架構	平台	下載連結	簽章檔案連結
x86-64	Debian	https://amazoncloudwatch-agent.s3.amazonaws.com/debian/amd64/latest/ amazon-cloudwatch-agent .deb https://amazoncloudwatch-agent-##. ##. ###. COM /德比安/amd64 /最新/ amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/debian/amd64/latest/amazon-cloudwatch-agent. 德博 https://amazoncloudwatch-agent-##. ##. ##### / debi an /amd64 /最新/.deb. sig amazon-cloudwatch-agent
x86-64	Ubuntu	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/amd64/latest/ amazon-cloudwatch-agent .deb https://amazoncloudwatch-agent-##. ##. 亞馬遜公司/企業/amazon-cloudwatch-agent上午 64 /最新/	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/amd64/latest/amazon-cloudwatch-agent. 德博 https://amazoncloudwatch-agent-##. ##. #####/##/日期 64 /最新/.deb.sig amazon-cloudwatch-agent
x86-64	Oracle	https://amazoncloudwatch-agent.s3.amazonaws.com/oracle_linux/amd64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ## .###/# /amd64/最新/ amazon-cloudwatch-agent 轉	https://amazoncloudwatch-agent.s3.amazonaws.com/oracle_linux/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ## .###/###_ lin ux/amd64 /最新/.rpm.sig amazon-cloudwatch-agent

架構	平台	下載連結	簽章檔案連結
x86-64	macOS	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/amd64/latest/ amazon-cloudwatch-agent .pkg https://amazoncloudwatch-agent-##. ## .###/###/ AMD64 /最新/ .pkg amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/amd64/latest/ www.amazon-cloudwatch-agent pkg.sig https://amazoncloudwatch-agent-##. ##. ###/### #/ amd64 /最新/ .pkg.sig amazon-cloudwatch-agent
x86-64	Windows	https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent. 微星 https://amazoncloudwatch-agent-##. ##. ###/#口/ amd64 /最新/. amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent. 微笑 https://amazoncloudwatch-agent-##. ## .###/## / amd64 /最新/.msi.sig amazon-cloudwatch-agent
ARM64	Amazon Linux 2023 和 Amazon Linux 2	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/arm64/latest/ amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ##. 亞馬遜亞馬遜亞馬遜 /arm64 /最新/ amazon-cloudwatch-agent 轉	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/arm64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ##. 亞馬遜亞馬遜亞馬遜 /arm64 /最新/.rpm.sig amazon-cloudwatch-agent

架構	平台	下載連結	簽章檔案連結
ARM64	Redhat	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/arm64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ########/ 紅帽/ARM64 /最新/ amazon-cloudwatch-agent 轉	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/arm64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ##. 亞馬遜amazon-cloudwatch-agent公司/紅帽 / arm64 /最新/ .rpm.sig
ARM64	Ubuntu	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/arm64/latest/ amazon-cloudwatch-agent .deb https://amazoncloudwatch-agent-##. ##. 亞馬遜公司/管理/ARM64 /最新/ amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/arm64/latest/amazon-cloudwatch-agent. 德博 https://amazoncloudwatch-agent-##. ##. ##### / ubun tu/ arm64 /最新/.deb.sig amazon-cloudwatch-agent
ARM64	SUSE	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/arm64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ##. ###. COM / 蘇打/arm64 /最新/ amazon-cloudwatch-agent 轉	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/arm64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ##. ###. COM / #打/arm64 /最新/ .rpm.sig amazon-cloudwatch-agent

架構	平台	下載連結	簽章檔案連結
ARM64	MacOS	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/arm64/latest/ amazon-cloudwatch-agent .pkg https://amazoncloudwatch-agent-*##. ##. ###. COM* /達爾文/ arm64 /最新/ amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/arm64/latest/ www.amazon-cloudwatch-agent pkg.sig https://amazoncloudwatch-agent-*##. ##*。亞馬遜/達爾文/arm64 /最新/ amazon-cloudwatch-agent

使用命令列下載並安裝 CloudWatch 代理程式套件

1. 下載代 CloudWatch 理程式。

   在 Linux 伺服器上，輸入以下資訊。對於 *download-link*，請使用上表中適當的下載連結。

   ```
   wget download-link
   ```

   在執行 Windows Server 的伺服器上，下載以下檔案：

   ```
   https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent.msi
   ```

2. 下載套件之後，您可以選擇驗證套件簽章。如需詳細資訊，請參閱 [驗證代 CloudWatch 理程式套件的簽章](#)。
3. 安裝套件。若您在 Linux 伺服器上下載了 RPM 套件，請變更到包含該套件的目錄，然後輸入以下資訊：

   ```
   sudo rpm -U ./amazon-cloudwatch-agent.rpm
   ```

   若您在 Linux 伺服器上下載了 DEB 套件，請變更到包含該套件的目錄，然後輸入以下資訊：

   ```
   sudo dpkg -i -E ./amazon-cloudwatch-agent.deb
   ```

若您在執行 Windows Server 的伺服器上下載了 MSI 套件，請變更到包含該套件的目錄，然後輸入以下資訊：

```
msiexec /i amazon-cloudwatch-agent.msi
```

此命令也可以在內部使用 PowerShell。如需 MSI 命令選項的詳細資訊，請參閱 Microsoft Windows 文件中的命令列選項。

若您在 macOS 伺服器上下載了 PKG 套件，請變更到包含該套件的目錄，然後輸入以下資訊：

```
sudo installer -pkg ./amazon-cloudwatch-agent.pkg -target /
```

建立及修改代理程式組態檔案

下載 CloudWatch 代理程式之後，您必須先建立組態檔，然後才能在任何伺服器上啟動代理程式。如需詳細資訊，請參閱 建立 CloudWatch 代理程式組態檔。

建立 IAM 角色和使用者以搭配 CloudWatch 代理程式使用

存取資 AWS 源需要權限。您可以建立 IAM 角色、IAM 使用者或兩者，以授與 CloudWatch 代理程式寫入指標所需的許可 CloudWatch。若您要在 Amazon EC2 執行個體上使用代理程式，您必須建立 IAM 角色。若您要在內部部署伺服器上使用代理程式，您必須建立 IAM 使用者。

> **Note**
>
> 我們最近使用 Amazon 所建立的新 `CloudWatchAgentServerPolicy` 及 `CloudWatchAgentAdminPolicy` 政策修改了下列程序，而無須要求客戶自行建立這些政策。針對寫入檔案及從參數存放區下載檔案，Amazon 所建立的政策只支援名稱開頭為 `AmazonCloudWatch-` 的檔案。如果您的 CloudWatch 代理程式組態檔案名稱開頭不是`AmazonCloudWatch-`，則無法使用這些原則將檔案寫入參數存放區或從參數存放區下載。

如果您要在 Amazon EC2 執行個體上執行 CloudWatch 代理程式，請使用下列步驟建立必要的 IAM 角色。此角色提供從執行個體讀取資訊並將其寫入的權限 CloudWatch。

建立在 EC2 執行個體上執行 CloudWatch 代理程式所需的 IAM 角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側的導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 確定在 Trusted entity type (受信任實體類型) 下，選取 AWS service (服務)。
4. 對於 Use case (使用案例)，在 Common use cases (常用使用案例) 下，選擇 EC2。
5. 選擇下一步。
6. 在策略清單中，選取旁邊的核取方塊CloudWatchAgentServerPolicy。如有需要，請使用搜尋方塊來尋找政策。
7. (選擇性) 如果代理程式正在將追蹤傳送至 X-Ray，您還需要為AWSXRayDaemonWriteAccess原則指定角色。因此，請在清單中找到該政策，然後選取其旁邊的核取方塊。
8. 選擇下一步。
9. 在角色名稱中，輸入角色的名稱，例如*CloudWatchAgentServerRole*。您可以選擇性地給予它一個描述。然後選擇 Create role (建立角色)。

   現在已建立角色。
10. (選擇性) 如果代理程式要將記錄檔傳送至 CloudWatch 記錄檔，而您希望代理程式能夠為這些記錄群組設定保留原則，則需要將`logs:PutRetentionPolicy`權限新增至角色。如需詳細資訊，請參閱 允許 CloudWatch 代理程式設定記錄保留原則。

如果您要在內部部署伺服器上執行 CloudWatch 代理程式，請使用下列步驟建立必要的 IAM 使用者。

> **Warning**
>
> 此案例需要具有程式設計存取權限和長期登入資料的 IAM 使用者，這會帶來安全風險。為了減輕此風險，我們建議您僅向這些使用者提供執行工作所需的權限，並在不再需要這些使用者時移除這些使用者。如有必要，可更新存取金鑰。如需詳細資訊，請參閱 IAM 使用者指南中的更新存取金鑰。

建立代理程式在內部部署伺服器上執 CloudWatch 行所需的 IAM 使用者

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側導覽窗格中，依次選擇 Users (使用者)、Add user (新增使用者)。

3. 為新使用者輸入使用者名稱。
4. 選取 Access key - Programmatic access (存取金鑰 - 程式設計存取)，然後選擇 Next: Permissions (下一步：許可)。
5. 選擇直接連接現有政策。
6. 在策略清單中，選取旁邊的核取方塊CloudWatchAgentServerPolicy。如有需要，請使用搜尋方塊來尋找政策。
7. (選擇性) 如果代理程式將追蹤至 X-Ray，您還需要為AWSXRayDaemonWriteAccess原則指定角色。因此，請在清單中找到該政策，然後選取其旁邊的核取方塊。
8. 選擇下一步：標籤。
9. (選用) 為新 IAM 使用者建立標籤，然後選擇 Next:Review (下一步：檢閱)。
10. 確認列出的是正確的政策，然後選擇 Create user (建立使用者)。
11. 在新使用者名稱的旁邊，選擇 Show (顯示)。將存取金鑰和秘密金鑰複製至檔案，以便在安裝代理程式時使用。選擇關閉。

允許 CloudWatch 代理程式設定記錄保留原則

您可以設定 CloudWatch 代理程式，為其傳送記錄事件的記錄群組設定保留原則。如果您這樣做，您必須將 `logs:PutRetentionPolicy` 授予代理程式使用的 IAM 角色或使用者。代理程式使用 IAM 角色在 Amazon EC2 執行個體上執行，並為內部部署伺服器使用 IAM 使用者。

授與 CloudWatch 代理程式的 IAM 角色設定記錄保留政策的權限

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側導覽窗格中，選擇 Roles (角色)。
3. 在搜尋方塊中，輸入 CloudWatch 代理程式 IAM 角色名稱的開頭。您可以在建立角色時選擇此名稱。其可能被命名為 `CloudWatchAgentServerRole`。

   如果您看到該角色，請選擇角色的名稱。
4. 在 Permissions (許可) 索引標籤中，依次選擇 Add permissions (新增許可)、Create inline policy (建立內嵌政策)。
5. 選擇 JSON 索引標籤並將以下政策複製到方塊中，以便替換方塊中的預設 JSON：

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:PutRetentionPolicy",
      "Resource": "*"
    }
  ]
}
```

6. 選擇檢閱政策。
7. 對於 Name (名稱)，輸入 **CloudWatchAgentPutLogsRetention** 或類似的內容，然後選擇 Create policy (建立政策)。

授與 CloudWatch 代理程式的 IAM 使用者設定記錄保留政策的權限

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側導覽窗格中，選擇 Users (使用者)。
3. 在搜尋方塊中，輸入 CloudWatch 代理程式 IAM 使用者名稱的開頭。您可以在建立使用者時選擇此名稱。

   如果您看到該使用者，請選擇使用者的名稱。
4. 針對 Permissions (許可) 索引標籤，選擇 Add inline policy (新增內嵌政策)。
5. 選擇 JSON 索引標籤並將以下政策複製到方塊中，以便替換方塊中的預設 JSON：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:PutRetentionPolicy",
      "Resource": "*"
    }
  ]
}
```

6. 選擇檢閱政策。
7. 對於 Name (名稱)，輸入 **CloudWatchAgentPutLogsRetention** 或類似的內容，然後選擇 Create policy (建立政策)。

在伺服器上安裝和執行 CloudWatch 代理程式

在您建立您希望的代理程式組態檔案，並建立 IAM 角色或 IAM 使用者後，請使用下列步驟來在您的伺服器上安裝及執行代理程式，並使用該組態。首先，請先將 IAM 角色或 IAM 使用者連接到將執行代理程式的伺服器。接著，在該伺服器上，下載代理程式套件，並使用您建立的代理程式組態啟動它。

使用 S3 下載連結下載 CloudWatch 代理程式套件

> **Note**
>
> 若要下載 CloudWatch 代理程式，您的連線必須使用 TLS 1.2 或更新版本。

您需要在每個您將執行代理程式的伺服器上安裝代理程式。

Amazon

該 CloudWatch 代理程式可在 Amazon Linux 2023 和 Amazon Linux 2 中以套件形式提供。如果您使用的是此作業系統，則可以輸入下列命令來安裝套件。您還必須確保附加到執行個體的 IAM 角色已CloudWatchAgentServerPolicy附加。如需詳細資訊，請參閱建立 IAM 角色以搭配 Amazon EC2 執行個體上的 CloudWatch 代理程式使用。

```
sudo yum install amazon-cloudwatch-agent
```

所有作業系統

在所有支援的作業系統上，您可以使用具有 Amazon S3 下載連結的命令列下載和安裝 CloudWatch 代理程式，如下列步驟所述。

每個下載連結都有一個一般連結及針對每個區域的連結。例如，對於 Amazon Linux 2023 和 Amazon Linux 2 和 x86-64 架構，三個有效的下載鏈接是：

- `https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`
- `https://amazoncloudwatch-agent-us-east-1.s3.us-east-1.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`
- `https://amazoncloudwatch-agent-eu-central-1.s3.eu-central-1.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`

架構	平台	下載連結	簽章檔案連結
x86-64	Amazon Linux 2023 和 Amazon Linux 2	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ##. 亞馬遜亞馬遜亞馬遜 /amd64 /最新/ amazon-cloudwatch-agent 轉	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ##. 亞馬遜亞馬遜亞馬遜/amd64 /最新/ .rpm.sig amazon-cloudwatch-agent
x86-64	Centos	https://amazoncloudwatch-agent.s3.amazonaws.com/centos/amd64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ##. 亞馬遜amazon-cloudwatch-agent公司/百分之六十四/最新/轉	https://amazoncloudwatch-agent.s3.amazonaws.com/centos/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ##. 亞馬遜amazon-cloudwatch-agent公司/百分之六十四/最新/.rpm.sig
x86-64	Redhat	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/amd64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ##. #####/紅帽 /amd64 /最新/ amazon-cloudwatch-agent 轉	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ##. 亞馬遜amazon-cloudwatch-agent公司/紅帽 /amd64 /最新/ .rpm.sig
x86-64	SUSE	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/amd64/latest/amazon-cloudwatch-agent. 每分鐘	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/amd64/latest/ amazon-cloudwatch-agent .rpm.sig

架構	平台	下載連結	簽章檔案連結
		https://amazoncloudwatch-agent-##. ##. 亞馬遜公司/蘇/amazon-cloudwatch-agent安 64 /最新/ 轉	https://amazoncloudwatch-agent-##. ##. 亞馬遜amazon-cloudwatch-agent公司/蘇聯/安德 64 /最新/ .rpm.sig
x86-64	Debian	https://amazoncloudwatch-agent.s3.amazonaws.com/debian/amd64/latest/ amazon-cloudwatch-agent .deb https://amazoncloudwatch-agent-##. ##. ###. COM /德比安/amd64 /最新/ amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/debian/amd64/latest/amazon-cloudwatch-agent. 德博 https://amazoncloudwatch-agent-##. ##. ##### / debi an /amd64 /最新/.deb. sig amazon-cloudwatch-agent
x86-64	Ubuntu	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/amd64/latest/ amazon-cloudwatch-agent .deb https://amazoncloudwatch-agent-##. ##. 亞馬遜公司/企業/amazon-cloudwatch-agent上午 64 /最新/	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/amd64/latest/amazon-cloudwatch-agent. 德博 https://amazoncloudwatch-agent-##. ##. #####/##/日期 64 /最新/.deb.sig amazon-cloudwatch-agent
x86-64	Oracle	https://amazoncloudwatch-agent.s3.amazonaws.com/oracle_linux/amd64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ## .###/# /amd64/ 最新/ amazon-cloudwatch-agent 轉	https://amazoncloudwatch-agent.s3.amazonaws.com/oracle_linux/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ## .###/###_ lin ux/amd64 /最新/.rpm.sig amazon-cloudwatch-agent

架構	平台	下載連結	簽章檔案連結
x86-64	macOS	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/amd64/latest/ amazon-cloudwatch-agent .pkg https://amazoncloudwatch-agent-##. ## .###/###/ AMD64 /最新/ .pkg amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/amd64/latest/ www.amazon-cloudwatch-agent pkg.sig https://amazoncloudwatch-agent-##. ##. ###/### #/ amd64 /最新/ .pkg.sig amazon-cloudwatch-agent
x86-64	Windows	https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent. 微星 https://amazoncloudwatch-agent-##. ##. ###/#口/ amd64 /最新/. amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent. 微笑 https://amazoncloudwatch-agent-##. ## .###/## / amd64 /最新/.msi.sig amazon-cloudwatch-agent
ARM64	Amazon Linux 2023 和 Amazon Linux 2	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/arm64/latest/ amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ##. 亞馬遜亞馬遜亞馬遜 /arm64 /最新/ amazon-cloudwatch-agent 轉	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/arm64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ##. 亞馬遜亞馬遜亞馬遜 /arm64 /最新/.rpm.sig amazon-cloudwatch-agent

架構	平台	下載連結	簽章檔案連結
ARM64	Redhat	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/arm64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-*##*. *########/* 紅帽/ARM64 /最新/ amazon-cloudwatch-agent 轉	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/arm64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-*##*. *##*. 亞馬遜amazon-cloudwatch-agent公司/紅帽 / arm64 /最新/ .rpm.sig
ARM64	Ubuntu	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/arm64/latest/ amazon-cloudwatch-agent .deb https://amazoncloudwatch-agent-*##*. *##*. 亞馬遜公司/管理/ARM64 /最新/ amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/arm64/latest/amazon-cloudwatch-agent. 德博 https://amazoncloudwatch-agent-*##*. *##*. *#####* / *ubun* tu/ arm64 /最新/.deb.sig amazon-cloudwatch-agent
ARM64	SUSE	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/arm64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-*##*. *##*. *###*. *COM* / 蘇打/arm64 /最新/ amazon-cloudwatch-agent 轉	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/arm64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-*##*. *##*. *###*. *COM* / *#*打/arm64 /最新/ .rpm.sig amazon-cloudwatch-agent

使用命令列在 Amazon EC2 執行個體上安裝 CloudWatch 代理程式

1. 下載代 CloudWatch 理程式。針對 Linux 伺服器，請輸入以下資訊。對於 *download-link*，請使用上表中適當的下載連結。

```
wget download-link
```

針對執行 Windows Server 的伺服器，請下載以下檔案：

```
https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/amazon-
cloudwatch-agent.msi
```

2. 下載套件之後，您可以選擇驗證套件簽章。如需詳細資訊，請參閱 驗證代 CloudWatch 理程式套件的簽章。
3. 安裝套件。若您在 Linux 伺服器上下載了 RPM 套件，請變更到包含該套件的目錄，然後輸入以下資訊：

```
sudo rpm -U ./amazon-cloudwatch-agent.rpm
```

若您在 Linux 伺服器上下載了 DEB 套件，請變更到包含該套件的目錄，然後輸入以下資訊：

```
sudo dpkg -i -E ./amazon-cloudwatch-agent.deb
```

若您在執行 Windows Server 的伺服器上下載了 MSI 套件，請變更到包含該套件的目錄，然後輸入以下資訊：

```
msiexec /i amazon-cloudwatch-agent.msi
```

此命令也可以在內部使用 PowerShell。如需 MSI 命令選項的詳細資訊，請參閱 Microsoft Windows 文件中的命令列選項。

(在 EC2 執行個體上安裝) 連接 IAM 角色

若要讓 CloudWatch 代理程式從執行個體傳送資料，您必須將 IAM 角色附加至執行個體。要附加的角色是CloudWatchAgentServerRole。您之前應該已建立此角色。如需更多資訊，請參閱建立 IAM 角色和使用者以搭配 CloudWatch 代理程式使用。

如需有關將 IAM 角色連接至執行個體的詳細資訊，請參閱《Amazon EC2 Windows 執行個體使用者指南》中的將 IAM 角色連接至執行個體。

(在內部部署伺服器上安裝) 指定 IAM 登入資料和 AWS 區域

若要讓 CloudWatch 代理程式能夠從內部部署伺服器傳送資料，您必須指定先前建立之 IAM 使用者的存取金鑰和秘密金鑰。如需建立此使用者的詳細資訊，請參閱建立 IAM 角色和使用者以搭配 CloudWatch 代理程式使用。

您也必須使用 AWS 組態檔 AWS 區[AmazonCloudWatchAgent]段中的region欄位，指定要傳送量度的「地區」，如下列範例所示。

```
[profile AmazonCloudWatchAgent]
region = us-west-1
```

以下是使用aws configure命令為 CloudWatch 代理程式建立具名設定檔的範例。此範例假設您使用名為 AmazonCloudWatchAgent 的預設設定檔。

建立 CloudWatch 代理程式的 AmazonCloudWatchAgent 設定檔

1. 如果您尚未這樣做，請在伺服器 AWS Command Line Interface 上安裝。如需詳細資訊，請參閱安裝 AWS CLI。
2. 在 Linux 伺服器上，輸入以下命令並依提示操作：

   ```
   sudo aws configure --profile AmazonCloudWatchAgent
   ```

   在 Windows Server 上，以系統管理員身分開啟 PowerShell ，輸入下列命令，然後依照提示執行。

   ```
   aws configure --profile AmazonCloudWatchAgent
   ```

驗證網際網路存取

您的 Amazon EC2 執行個體必須具有輸出網際網路存取權，才能將資料傳送至 CloudWatch 或 CloudWatch 記錄 如需有關如何設定網際網路存取的詳細資訊，請參閱《Amazon VPC 使用者指南》中的網際網路閘道。

在您的代理上設定的端點和連接埠如下所示：

- 如果您使用代理程式收集指標，則必須將適當區域的 CloudWatch端點新增至允許清單。這些端點列在 Amazon CloudWatch 端點和配額中。

- 如果您使用代理程式收集記錄檔，則必須將適當區域的 CloudWatch 記錄檔端點新增至允許清單。這些端點列在 Amazon CloudWatch 日誌端點和配額中。
- 若您使用 Systems Manager 安裝代理程式或參數存放區來存放組態檔案，您必須針對適當的區域將 Systems Manager 端點新增至允許清單。這些端點會列在 AWS Systems Manager 端點和配額中。

(選用) 修改代理或區域資訊的常見組態

CloudWatch 代理程式包含一個名為的組態檔common-config.toml。您可以選擇性地使用此檔案來指定代理和區域資訊。

在執行 Linux 的伺服器上，此檔案位於 /opt/aws/amazon-cloudwatch-agent/etc 目錄。在執行 Windows Server 的伺服器上，此檔案位於 C:\ProgramData\Amazon \AmazonCloudWatchAgent 目錄。

> **Note**
>
> 我們建議您在內部部署模式下執行 CloudWatch 代理程式時，使用該common-config.toml檔案提供共用組態和登入資料，而且當您在 Amazon EC2 上執行，而且想要重複使用現有的共用登入資料設定檔和檔案時，這也很有用。透過啟用它common-config.toml具有額外的優點：如果您的共用認證檔案在到期後使用更新的認證輪換，則代理程式會自動取得新認證，而不需要重新啟動。

預設的 common-config.toml 如下。

```
# This common-config is used to configure items used for both ssm and cloudwatch access


## Configuration for shared credential.
## Default credential strategy will be used if it is absent here:
##            Instance role is used for EC2 case by default.
##            AmazonCloudWatchAgent profile is used for the on-premises case by
 default.
# [credentials]
#    shared_credential_profile = "{profile_name}"
#    shared_credential_file= "{file_name}"

## Configuration for proxy.
## System-wide environment-variable will be read if it is absent here.
## i.e. HTTP_PROXY/http_proxy; HTTPS_PROXY/https_proxy; NO_PROXY/no_proxy
```

```
## Note: system-wide environment-variable is not accessible when using ssm run-command.
## Absent in both here and environment-variable means no proxy will be used.
# [proxy]
#    http_proxy = "{http_url}"
#    https_proxy = "{https_url}"
#    no_proxy = "{domain}"
```

最初所有行都會標示為註解。若要設定登入資料設定檔或代理設定，請移除該行的 # 並指定值。您可以手動編輯此檔案，或使用 Systems Manager 中的 RunShellScript 執行命令：

- shared_credential_profile— 對於內部部署伺服器，此行指定要用來傳送資料的 IAM 使用者登入資料設定檔 CloudWatch。若您將此行標示為註解，則會使用 AmazonCloudWatchAgent。如需建立此描述檔的詳細資訊，請參閱(在內部部署伺服器上安裝) 指定 IAM 登入資料和 AWS 區域。

  CloudWatch 在 EC2 執行個體上，您可以使用此行讓 CloudWatch 代理程式將資料從此執行個體傳送到不同的 AWS 區域。若要執行此作業，請指定一個具名描述檔，其中包含指定要傳送對象區域名稱的 region 欄位。

  如果您指定 shared_credential_profile，即必須也要移除 [credentials] 行開頭中的 #。
- shared_credential_file – 若要讓代理程式在位於預設路徑以外路徑的檔案中尋找憑證，請在此處指定該完整路徑及檔案名稱。Linux 的預設路徑是 /root/.aws，Windows Server 的預設路徑是 C:\\Users\\Administrator\\.aws。

  以下第一個範例顯示適用於 Linux 伺服器的 shared_credential_file 行語法，第二個範例則適用於 Windows Server 有效。在 Windows Server 上，您必須跳脫 \ 字元。

  ```
  shared_credential_file= "/usr/username/credentials"
  ```

  ```
  shared_credential_file= "C:\\Documents and Settings\\username\\.aws\\credentials"
  ```

  如果您指定 shared_credential_file，即必須也要移除 [credentials] 行開頭中的 #。
- 代理設定 – 若您的伺服器使用 HTTP 或 HTTPS 代理來和 AWS 服務聯絡，請在 http_proxy 和 https_proxy 欄位中指定那些代理。如有必須排除在代理之外的 URL，請在 no_proxy 欄位中指定並以逗號分隔。

使用命令列啟動 CloudWatch 代理程式

請依照下列步驟使用命令列在伺服器上啟動 CloudWatch 代理程式。

使用命令列在伺服器上啟動 CloudWatch 代理程式

1. 將您希望使用的代理程式組態檔案複製到您要執行代理程式的伺服器。請注意您複製的目標路徑名稱。
2. 在此命令中，`-a fetch-config`會使代理程式載入最新版本的 CloudWatch 代理程式組態檔，並`-s`啟動代理程式。

   輸入以下其中一個命令。以代理程式組態檔的路徑取*`configuration-file-path`*代。如果您使用精靈建立，則這個檔案稱為 `config.json`；如果您以手動方式建立，則可能稱為 `amazon-cloudwatch-agent.json`。

   在執行 Linux 的 EC2 執行個體上，輸入以下命令：

   ```
   sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:configuration-file-path
   ```

   在執行 Linux 的現場部署伺服器上，輸入以下內容：

   ```
   sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m onPremise -s -c file:configuration-file-path
   ```

   在執行 Windows 伺服器的 EC2 執行個體上，從 PowerShell 主控台輸入以下內容：

   ```
   & "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a fetch-config -m ec2 -s -c file:configuration-file-path
   ```

   在執行 Windows Server 的內部部署伺服器上，從 PowerShell 主控台輸入下列內容：

   ```
   & "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -a fetch-config -m onPremise -s -c file:configuration-file-path
   ```

使用安裝 CloudWatch 代理程式 AWS Systems Manager

您可以使用下列主題來安裝和執行 CloudWatch 代理程式 AWS Systems Manager。

主題

- 建立 IAM 角色和使用者以搭配 CloudWatch 代理程式使用

- 下載並設定 CloudWatch 代理程式
- 使用 CloudWatch代理程式組態在 EC2 執行個體上安裝代理程式
- 在內部部署伺服器上安裝 CloudWatch 代理

建立 IAM 角色和使用者以搭配 CloudWatch 代理程式使用

存取資 AWS 源需要權限。您可以建立 IAM 角色和使用者，其中包含 CloudWatch 代理程式寫入指標所需的許可，以 CloudWatch 及讓 CloudWatch 代理程式與 Amazon EC2 和通訊所需的許可 AWS Systems Manager。若是 Amazon EC2 執行個體，您可以使用 IAM 角色；若是內部部署伺服器，您可以使用 IAM 使用者。

一個角色或使用者可讓 CloudWatch 代理程式安裝在伺服器上，並將指標傳送至 CloudWatch。需要另一個角色或使用者才能將 CloudWatch 代理程式組態儲存在 Systems Manager 參數存放區中。參數存放區可讓多部伺服器使用一個 CloudWatch 代理程式組態。

寫入參數存放區的能力是一項廣泛且強大的許可。建議您只在需要的時候使用它，並且建議您不要將它連接到您部署中的多個執行個體。如果您將 CloudWatch 代理程式組態儲存在參數存放區中，建議您執行下列操作：

- 設定一個執行此程式組態的執行個體。
- 只在此執行個體使用具備寫入許可的 IAM 角色寫入參數存放區。
- 僅在處理和儲存 CloudWatch 代理程式組態檔案時，使用具有許可的 IAM 角色寫入參數存放區。

> **Note**
>
> 我們最近使用 Amazon 所建立的新 `CloudWatchAgentServerPolicy` 及 `CloudWatchAgentAdminPolicy` 政策修改了下列程序，而無須要求客戶自行建立這些政策。若要使用這些政策將代理程式組態檔寫入參數存放區，然後從參數存放區下載檔案，您的代理程式組態檔案必須具有開頭為 `AmazonCloudWatch-` 的名稱。如果您的 CloudWatch 代理程式組態檔案名稱開頭不是`AmazonCloudWatch-`，則無法使用這些原則將檔案寫入參數存放區或從參數存放區下載檔案。

建立 IAM 角色以搭配 Amazon EC2 執行個體上的 CloudWatch 代理程式使用

第一個程序會建立 IAM 角色，您必須將該角色連接至執行 CloudWatch 代理程式的每個 Amazon EC2 執行個體。此角色提供從執行個體讀取資訊並將其寫入的權限 CloudWatch。

第二個程序會建立 IAM 角色，您必須將該角色連接至用於建立 CloudWatch 代理程式組態檔的 Amazon EC2 執行個體。如果您要將此檔案存放在 Systems Manager 參數存放區，讓其他伺服器可以使用它，這是必要的步驟。除了從執行個體讀取資訊並寫入資訊的權限外，此角色還提供寫入參數存放區的權限 CloudWatch。此角色包含足以執行 CloudWatch 代理程式以及寫入參數存放區的權限。

> **Note**
>
> 參數存放區支援標準和進階層的參數。這些參數層與「 CloudWatch 代理程式」預先定義的測量結果集可用的「基本」、「標準」和「進階」詳細資訊層次無關。

建立每部伺服器執行 CloudWatch 代理程式所需的 IAM 角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 在 Select type of trusted entity (選擇可信任執行個體類型) 下，選擇 AWS service (服務)。
4. 即刻在 Common use cases (常用使用案例) 下，選擇 EC2，然後選擇 Next: Permissions (下一步：許可)。
5. 在策略清單中，選取旁邊的核取方塊CloudWatchAgentServerPolicy。如有需要，請使用搜尋方塊來尋找政策。
6. 若要使用 Systems Manager 來安裝或設定 CloudWatch 代理程式，請選取 Amazon ManagedInstanceCore SSM 旁邊的核取方塊。此 AWS 受管原則可讓執行個體使用 Systems Manager 服務核心功能。如有需要，請使用搜尋方塊來尋找政策。如果您只透過命令列啟動和設定代理程式，便不需要此政策。
7. 選擇下一步：標籤。
8. (選用) 新增一或多個標籤鍵/值對來組織、追蹤或控制存取此角色，然後選擇 Next: Review (下一步：檢視)。
9. 對於 Role name (角色名稱)，輸入新角色的名稱，例如 **`CloudWatchAgentServerRole`** 或另一個您喜好的名稱。
10. (選用) 針對 Role description (角色描述)，輸入描述。
11. 確認CloudWatchAgentServerPolicy並選擇性地ManagedInstanceCore顯示在策略旁邊。
12. 選擇建立角色。

    現在已建立角色。

以下程序會建立也能寫入參數存放區的 IAM 角色。您可以使用此角色在參數存放區內存放代理程式組態檔案，以便其他伺服器可以擷取此組態檔案。

寫入參數存放區的許可可提供廣泛的存取。此角色不應連接到您所有的伺服器，並且應只能讓管理員使用它。在您建立代理程式組態檔案，並將它複製到參數存放區後，建議您從執行個體分離此角色，並改為使用 `CloudWatchAgentServerRole`。

若要為管理員建立寫入參數存放區的 IAM 角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 在 Select type of trusted entity (選擇可信任執行個體類型) 下，選擇 AWS service (服務)。
4. 緊接在 Choose the service that will use this role (選擇將使用此角色的服務) 下，選擇 EC2，然後選擇 Next: Permissions (下一步：許可)。
5. 在策略清單中，選取旁邊的核取方塊CloudWatchAgentAdminPolicy。如有需要，請使用搜尋方塊來尋找政策。
6. 若要使用 Systems Manager 來安裝或設定 CloudWatch 代理程式，請選取 Amazon ManagedInstanceCore SSM 旁邊的核取方塊。此 AWS 受管原則可讓執行個體使用 Systems Manager 服務核心功能。如有需要，請使用搜尋方塊來尋找政策。如果您只透過命令列啟動和設定代理程式，便不需要此政策。
7. 選擇下一步：標籤。
8. (選用) 新增一或多個標籤鍵/值對來組織、追蹤或控制存取此角色，然後選擇 Next: Review (下一步：檢視)。
9. 對於 Role name (角色名稱)，輸入新角色的名稱，例如 **`CloudWatchAgentAdminRole`** 或另一個您喜好的名稱。
10. (選用) 針對 Role description (角色描述)，輸入描述。
11. 確認CloudWatchAgentAdminPolicy並選擇性地ManagedInstanceCore顯示在策略旁邊。
12. 選擇建立角色。

    現在已建立角色。

建立 IAM 使用者，以搭配內部部署伺服器上的 CloudWatch 代理程式

第一個程序會建立執行 CloudWatch 代理程式所需的 IAM 使用者。此使用者提供將資料傳送至的權限 CloudWatch。

第二個程序會建立可在建立 CloudWatch代理程式設定檔時使用的 IAM 使用者。使用此程序在 Systems Manager 參數存放區內存放此檔案，以便其他伺服器可以使用此檔案。除了寫入資料的權限外，此使用者還提供寫入參數存放區的權限 CloudWatch。

> **Note**
>
> 參數存放區支援標準和進階層的參數。這些參數層與「 CloudWatch 代理程式」預先定義的測量結果集可用的「基本」、「標準」和「進階」詳細資訊層次無關。

建立 CloudWatch 代理程式寫入資料所需的 IAM 使用者 CloudWatch

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 使用者，然後選擇 新增使用者。
3. 為新使用者輸入使用者名稱。
4. 針對 Access type (存取類型)，選取 Programmatic access (程式設計存取)，然後選擇 Next: Permissions (下一步：許可)。
5. 針對 Set permissions (設定許可)，選擇 Attach existing policies directly (直接連接現有政策)。
6. 在策略清單中，選取旁邊的核取方塊CloudWatchAgentServerPolicy。如有需要，請使用搜尋方塊來尋找政策。
7. 若要使用 Systems Manager 來安裝或設定 CloudWatch 代理程式，請選取 Amazon ManagedInstanceCore SSM 旁邊的核取方塊。此 AWS 受管原則可讓執行個體使用 Systems Manager 服務核心功能。(如有需要，請使用搜尋方塊來尋找此政策。如果您只透過命令列啟動和設定代理，便不需要此政策。)
8. 選擇下一步：標籤。
9. (選用) 新增一或多個標籤鍵/值對來組織、追蹤或控制存取此角色，然後選擇 Next: Review (下一步：檢視)。
10. 確認列出的是正確的政策，然後選擇 Create user (建立使用者)。
11. 在新使用者列中，選擇 Show (顯示)。將存取金鑰和秘密金鑰複製至檔案，以便在安裝代理程式時使用。選擇關閉。

以下程序會建立也能寫入參數存放區的 IAM 使用者。如果您要將代理程式組態檔案存放於參數存放區，以便其他伺服器可以使用此組態檔案，您必須使用此 IAM 使用者。此 IAM 使用者提供寫入參數

存放區的許可。此使用者也提供從執行個體讀取資訊並將其寫入的權限 CloudWatch。寫入 Systems Manager 參數存放區的許可可提供廣泛的存取。此 IAM 使用者不應連接到您所有的伺服器，並且應只能讓管理員使用它。您應該只有在將代理程式組態檔案存放於參數存放區時，才使用此 IAM 使用者。

建立將設定檔儲存在參數存放區並將資訊傳送至所需的 IAM 使用者 CloudWatch

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 使用者，然後選擇 新增使用者。
3. 為新使用者輸入使用者名稱。
4. 針對 Access type (存取類型)，選取 Programmatic access (程式設計存取)，然後選擇 Next: Permissions (下一步：許可)。
5. 針對 Set permissions (設定許可)，選擇 Attach existing policies directly (直接連接現有政策)。
6. 在策略清單中，選取旁邊的核取方塊CloudWatchAgentAdminPolicy。如有需要，請使用搜尋方塊來尋找政策。
7. 若要使用 Systems Manager 來安裝或設定 CloudWatch 代理程式，請選取 Amazon ManagedInstanceCore SSM 旁邊的核取方塊。此 AWS 受管原則可讓執行個體使用 Systems Manager 服務核心功能。(如有需要，請使用搜尋方塊來尋找此政策。如果您只透過命令列啟動和設定代理，便不需要此政策。)
8. 選擇下一步：標籤。
9. (選用) 新增一或多個標籤鍵/值對來組織、追蹤或控制存取此角色，然後選擇 Next: Review (下一步：檢視)。
10. 確認列出的是正確的政策，然後選擇 Create user (建立使用者)。
11. 在新使用者列中，選擇 Show (顯示)。將存取金鑰和秘密金鑰複製至檔案，以便在安裝代理程式時使用。選擇關閉。

下載並設定 CloudWatch 代理程式

本節說明如何使用 Systems Manager 下載代理程式，以及如何建立您的代理程式組態檔案。在您可以使用 Systems Manager 下載代理程式前，您必須確保執行個體已針對 Systems Manager 設定正確。

安裝或更新 SSM Agent

在 Amazon EC2 執行個體上， CloudWatch 代理程式要求執行個體執行的是 2.2.93.0 版或更新版本。安裝 CloudWatch 代理程式之前，請先在執行個體上更新或安裝 SSM Agent (如果尚未安裝)。

如需在執行 Linux 的執行個體上安裝或更新 SSM Agent 的資訊，請參閱《AWS Systems Manager 使用者指南》中的在 Linux 執行個體上安裝和設定 SSM Agent。

如需有關安裝或更新 SSM Agent 的資訊，請參閱《AWS Systems Manager 使用者指南》中的使用 SSM Agent。

(選用) 驗證 Systems Manager 先決條件

驗證網際網路存取

您的 Amazon EC2 執行個體必須具有輸出網際網路存取權，才能將資料傳送至 CloudWatch 或 CloudWatch 記錄 如需有關如何設定網際網路存取的詳細資訊，請參閱《Amazon VPC 使用者指南》中的網際網路閘道。

在您的代理上設定的端點和連接埠如下所示：

- 如果您使用代理程式收集指標，則必須允許列出適當區域的 CloudWatch端點。這些端點列在中的 Amazon CloudWatch 中Amazon Web Services 一般參考。
- 如果您使用代理程式收集記錄檔，則必須允許列出適當區域的 CloudWatch 記錄檔端點。這些端點會列在中的 Amazon CloudWatch 日誌中Amazon Web Services 一般參考。
- 如果您使用 Systems Manager 安裝代理程式或參數存放區來存放組態檔案，您必須針對適當的區域將 Systems Manager 端點列入允許名單。這些端點會列在 Amazon Web Services 一般參考 的 AWS Systems Manager 中。

使用下列步驟，使用系統管理員下載 CloudWatch 代理程式套件。

使用系統管理員下載 CloudWatch 代 Systems Manager 程式

1. 開啟 Systems Manager 主控台，網址為 https://console.aws.amazon.com/systems-manager/。
2. 在導覽窗格中，選擇 執行命令。

   -或-

   如果 AWS Systems Manager 首頁開啟，請向下捲動並選擇「瀏覽執行命令」。
3. 選擇 執行命令。
4. 在 [命令] 文件清單中，選擇 [AWS AWSPackage 設定]。
5. 在 [目標] 區域中，選擇要安裝 CloudWatch代理程式的執行個體。如果看不到特定的執行個體，它可能未設定為受管執行個體供 Systems Manager 使用。如需詳細資訊，請參閱使AWS Systems Manager 用指南中的〈設定混合式環境AWS Systems Manager〉。

6. 在 Action (動作) 清單中，選擇 Install (安裝)。
7. 在 Name (名稱) 欄位中，輸入 *AmazonCloudWatchAgent*。
8. 保留將 Version (版本) 設為 latest (最新) 以安裝代理程式的最新版本。
9. 選擇執行。
10. 或者，在 Targets and outputs (目標和輸出) 區域中，選取執行個體名稱旁的按鈕，然後選擇 View output (檢視輸出)。Systems Manager 應該會顯示代理程式已成功安裝。

建立及修改代理程式組態檔案

下載 CloudWatch 代理程式之後，您必須先建立組態檔，然後才能在任何伺服器上啟動代理程式。

若您要將代理程式組態檔案儲存在 Systems Manager 參數存放區中，您必須使用 EC2 執行個體才能儲存到參數存放區。此外，您必須先將 `CloudWatchAgentAdminRole` IAM 角色連接到該執行個體。如需有關連接角色的詳細資訊，請參閱《Amazon EC2 Windows 執行個體使用者指南》中的將 IAM 角色連接至執行個體。

如需有關建立 CloudWatch 代理程式組態檔的詳細資訊，請參閱建立 CloudWatch 代理程式組態檔。

使用 CloudWatch代理程式組態在 EC2 執行個體上安裝代理程式

在參數存放區中儲存 CloudWatch 代理程式組態之後，您可以在其他伺服器上安裝代理程式時使用該組態。

主題

- 將 IAM 角色連接至執行個體
- 在 Amazon EC2 執行個體上下載 CloudWatch代理程式套件
- (選擇性) 修改 CloudWatch 代理程式的一般組態和具名設定檔
- 啟動 CloudWatch 代理程式

將 IAM 角色連接至執行個體

您必須將 CloudWatchAgentServerRoleIAM 角色附加到 EC2 執行個體，才能在執行個體上執行 CloudWatch 代理程式。此角色可讓 CloudWatch 代理程式在執行個體上執行動作。您之前應該已建立此角色。如需更多資訊，請參閱建立 IAM 角色和使用者以搭配 CloudWatch 代理程式使用。

如需詳細資訊，請參閱《Amazon EC2 Windows 執行個體使用者指南》中的將 IAM 角色連接至執行個體。

在 Amazon EC2 執行個體上下載 CloudWatch代理程式套件

您需要在每個您將執行代理程式的伺服器上安裝代理程式。該 CloudWatch 代理程式可在 Amazon Linux 2023 和 Amazon Linux 2 中以套件形式提供。如果您使用的是此作業系統，則可以輸入下列命令來安裝套件。您還必須確保附加到執行個體的 IAM 角色已CloudWatchAgentServerPolicy附加。如需詳細資訊，請參閱 建立 IAM 角色以搭配 Amazon EC2 執行個體上的 CloudWatch 代理程式使用。

```
sudo yum install amazon-cloudwatch-agent
```

在所有支援的作業系統上，您可以使用 Systems Manager 執行命令或 Amazon S3 下載連結下載 CloudWatch 代理程式套件。如需有關使用 Simple Storage Service (Amazon S3) 下載連結的資訊，請參閱 下載 CloudWatch 代理程式套件。

> **Note**
>
> 當您安裝或更新 CloudWatch 代理程式時，僅支援「解除安裝並重新安裝」選項。您無法使用 In-place update (就地更新) 選項。

使用系統管理器在 Amazon EC2 執行個體上下載 CloudWatch 代 Systems Manager 程式

在您可以使用 Systems Manager 安裝 CloudWatch 代理程式之前，您必須確定 Systems Manager 已正確設定執行個體。

安裝或更新 SSM Agent

在 Amazon EC2 執行個體上， CloudWatch 代理程式要求執行個體執行的是 2.2.93.0 版或更新版本。安裝 CloudWatch 代理程式之前，請先在執行個體上更新或安裝 SSM Agent (如果尚未安裝)。

如需在執行 Linux 的執行個體上安裝或更新 SSM Agent 的資訊，請參閱《AWS Systems Manager 使用者指南》中的在 Linux 執行個體上安裝和設定 SSM Agent。

如需在執行 Windows Server 的執行個體上安裝或更新 SSM Agent 的資訊，請參閱《AWS Systems Manager 使用者指南》中的在 Windows 執行個體上安裝和設定 SSM Agent。

(選用) 驗證 Systems Manager 先決條件

在您使用 Systems Manager 執行命令來安裝和設定 CloudWatch 代理程式之前，請確認您的執行個體符合 Systems Manager 的最低需求。如需詳細資訊，請參閱《AWS Systems Manager 使用者指南中的 設定 AWS Systems Manager。

驗證網際網路存取

您的 Amazon EC2 執行個體必須具有輸出網際網路存取權，才能將資料傳送至 CloudWatch 或 CloudWatch 記錄。如需有關如何設定網際網路存取的詳細資訊，請參閱《Amazon VPC 使用者指南》中的網際網路閘道。

下載 CloudWatch 代理程式套件

Systems Manager 執行命令可讓您管理執行個體的組態。您指定 Systems Manager 文件、指定參數，然後在一或多個執行個體上執行命令。在執行個體上的 SSM Agent 代理程式，會依指定來處理指令和設定執行個體。

使用執行命令下載 CloudWatch 代理程式

1. 開啟 Systems Manager 主控台，網址為 https://console.aws.amazon.com/systems-manager/。
2. 在導覽窗格中，選擇 執行命令。

   -或-

   如果 AWS Systems Manager 首頁開啟，請向下捲動並選擇「瀏覽執行命令」。
3. 選擇 執行命令。
4. 在 [命令] 文件清單中，選擇 [AWS-AWSPackage 設定]。
5. 在 [目標] 區域中，選擇要在其上安裝 CloudWatch 代理程式的執行個體。如果您沒看到特定執行個體，可能是因為它未設定用於執行命令。如需詳細資訊，請參閱《AWS Systems Manager 使用者指南》中的在混合環境中設定 AWS Systems Manager。
6. 在 Action (動作) 清單中，選擇 Install (安裝)。
7. 在 Name (名稱) 方塊中，輸入 *AmazonCloudWatchAgent*。
8. 保留將 Version (版本) 設為 latest (最新) 以安裝代理程式的最新版本。
9. 選擇執行。
10. 或者，在 Targets and outputs (目標和輸出) 區域中，選取執行個體名稱旁的按鈕，然後選擇 View output (檢視輸出)。Systems Manager 應該會顯示代理程式已成功安裝。

(選擇性) 修改 CloudWatch 代理程式的一般組態和具名設定檔

CloudWatch 代理程式包含一個名為的組態檔`common-config.toml`。您可以使用此檔案選擇性地指定代理和區域資訊。

在執行 Linux 的伺服器上，此檔案位於 /opt/aws/amazon-cloudwatch-agent/etc 目錄。在執行 Windows Server 的伺服器上，此檔案位於 C:\ProgramData\Amazon\AmazonCloudWatchAgent 目錄。

預設的 common-config.toml 如下：

```
# This common-config is used to configure items used for both ssm and cloudwatch access


## Configuration for shared credential.
## Default credential strategy will be used if it is absent here:
##            Instance role is used for EC2 case by default.
##            AmazonCloudWatchAgent profile is used for onPremise case by default.
# [credentials]
#    shared_credential_profile = "{profile_name}"
#    shared_credential_file= "{file_name}"

## Configuration for proxy.
## System-wide environment-variable will be read if it is absent here.
## i.e. HTTP_PROXY/http_proxy; HTTPS_PROXY/https_proxy; NO_PROXY/no_proxy
## Note: system-wide environment-variable is not accessible when using ssm run-command.
## Absent in both here and environment-variable means no proxy will be used.
# [proxy]
#    http_proxy = "{http_url}"
#    https_proxy = "{https_url}"
#    no_proxy = "{domain}"
```

最初所有行都會標示為註解。若要設定登入資料設定檔或代理設定，請移除該行的 # 並指定值。您可以手動編輯此檔案，或使用 Systems Manager 中的 RunShellScript 執行命令：

- shared_credential_profile— 對於內部部署伺服器，此行指定要用來傳送資料的 IAM 使用者登入資料設定檔 CloudWatch。若您將此行標示為註解，則會使用 AmazonCloudWatchAgent。

  CloudWatch 在 EC2 執行個體上，您可以使用此行讓 CloudWatch 代理程式將資料從此執行個體傳送到不同的 AWS 區域。若要執行此作業，請指定一個具名描述檔，其中包含指定要傳送對象區域名稱的 region 欄位。

  如果您指定 shared_credential_profile，即必須也要移除 [credentials] 行開頭中的 #。
- shared_credential_file – 若要讓代理程式在位於預設路徑以外路徑的檔案中尋找憑證，請在此處指定該完整路徑及檔案名稱。Linux 的預設路徑是 /root/.aws，Windows Server 的預設路徑是 C:\\Users\\Administrator\\.aws。

以下第一個範例顯示適用於 Linux 伺服器的 shared_credential_file 行語法，第二個範例則適用於 Windows Server 有效。在 Windows Server 上，您必須跳脫 \ 字元。

```
shared_credential_file= "/usr/username/credentials"
```

```
shared_credential_file= "C:\\Documents and Settings\\username\\.aws\\credentials"
```

如果您指定 shared_credential_file，即必須也要移除 [credentials] 行開頭中的 #。

- 代理設定 – 若您的伺服器使用 HTTP 或 HTTPS 代理來和 AWS 服務聯絡，請在 http_proxy 和 https_proxy 欄位中指定那些代理。如有必須排除在代理之外的 URL，請在 no_proxy 欄位中指定並以逗號分隔。

啟動 CloudWatch 代理程式

您可以使用 Systems Manager 執行命令或命令列啟動代理程式。

使用系統管理器運行命令啟動 CloudWatch 代理

依照以下步驟使用 Systems Manager 執行命令啟動代理程式。

使用執行命令啟動 CloudWatch 代理程式

1. 開啟 Systems Manager 主控台，網址為 https://console.aws.amazon.com/systems-manager/。
2. 在導覽窗格中，選擇 執行命令。

   -或-

   如果 AWS Systems Manager 首頁開啟，請向下捲動並選擇「瀏覽執行命令」。
3. 選擇 執行命令。
4. 在「命令」文件清單中，選擇 AmazonCloudWatch-ManageAgent。
5. 在 [目標] 區域中，選擇您安裝 CloudWatch 代理程式的執行個體。
6. 在 Action (動作) 清單中，選擇 configure (設定)。
7. 在 Optional Configuration Source (選用組態來源) 清單中，選擇 ssm。
8. 在選用組態位置方塊中，輸入您建立並儲存至 Systems Manager 參數存放區的 Systems Manager 參數名稱和代理程式組態檔案名稱，如 建立 CloudWatch 代理程式組態檔 中所述。

9. 完成這些步驟之後，在 Optional Restart (選用重新啟動) 清單中選擇 yes (是) 以啟動代理程式。
10. 選擇執行。
11. 或者，在 Targets and outputs (目標和輸出) 區域中，選取執行個體名稱旁的按鈕，然後選擇 View output (檢視輸出)。Systems Manager 應該會顯示代理程式已成功啟動。

使用命令列在 Amazon EC2 執行個體上啟動 CloudWatch 代理程式

請依照下列步驟使用命令列在 Amazon EC2 執行個體上安裝 CloudWatch 代理程式。

使用命令列在 Amazon EC2 執行個體上啟動 CloudWatch 代理程式

- 在此命令中，`-a fetch-config`會使代理程式載入最新版本的 CloudWatch 代理程式組態檔，並`-s`啟動代理程式。

  Linux 和 macOS：如果您已將組態檔案儲存於 Systems Manager 參數存放區，請輸入以下內容：

  ```
  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
  config -m ec2 -s -c ssm:configuration-parameter-store-name
  ```

  Linux 和 macOS：如果您已將組態檔案儲存於本機電腦，請輸入以下命令。以代理程式組態檔的路徑取*configuration-file-path*代。如果您使用精靈建立，則這個檔案稱為 `config.json`；如果您以手動方式建立，則可能稱為 `amazon-cloudwatch-agent.json`。

  ```
  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
  config -m ec2 -s -c file:configuration-file-path
  ```

  Windows Server：如果您將代理程式組態檔儲存在「Systems Manager 參數存放區」中，請從 PowerShell 主控台輸入下列內容：

  ```
  & "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -
  a fetch-config -m ec2 -s -c ssm:configuration-parameter-store-name
  ```

  Windows Server：如果您已將代理程式組態檔儲存在本機電腦上，請從 PowerShell 主控台輸入下列內容：

  ```
  & "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1"
   -a fetch-config -m ec2 -s -c file:"C:\Program Files\Amazon\AmazonCloudWatchAgent
  \config.json"
  ```

在內部部署伺服器上安裝 CloudWatch 代理

如果您已在一部電腦上下載 CloudWatch 代理程式，並建立所需的代理程式組態檔，則可以使用該組態檔將代理程式安裝在其他內部部署伺服器上。

在內部部署伺服器上下載 CloudWatch 代理程式

您可以使用 Systems Manager 執行命令或 Amazon S3 下載連結來下載 CloudWatch 代理程式套件。如需有關使用 Simple Storage Service (Amazon S3) 下載連結的資訊，請參閱 下載 CloudWatch 代理程式套件。

使用 Systems Manager 下載

若要使用 Systems Manager 執行命令，您必須使用 Amazon EC2 Systems Manager 註冊您的內部部署伺服器。如需詳細資訊，請參閱《AWS Systems Manager 使用者指南》中的在混合環境中設定 Systems Manager。

如果您已經註冊伺服器，請將 SSM Agent 更新為最新版本。

如需在執行 Linux 的伺服器上更新 SSM Agent 的相關資訊，請參閱《AWS Systems Manager 使用者指南》中的在混合環境 (Linux) 中安裝 SSM Agent。

如需在執行 Windows Server 的伺服器上更新 SSM Agent 的相關資訊，請參閱《AWS Systems Manager 使用者指南》中的在混合環境 (Windows) 中安裝 SSM Agent。

使用 SSM 代理程式在內部部署伺服器上下載 CloudWatch 代理程式套件

1. 開啟 Systems Manager 主控台，網址為 https://console.aws.amazon.com/systems-manager/。
2. 在導覽窗格中，選擇 執行命令。

   -或-

   如果 AWS Systems Manager 首頁開啟，請向下捲動並選擇「瀏覽執行命令」。
3. 選擇 執行命令。
4. 在 [命令] 文件清單中，選取 AWS AWSPackage 設定旁邊的按鈕。
5. 在「目標」區域中，選取要安裝 CloudWatch 代理程式的伺服器。若您沒有看到特定伺服器，可能是因為它尚未設定用於執行命令。如需詳細資訊，請參閱《AWS Systems Manager 使用者指南》中的在混合環境中設定 AWS Systems Manager。
6. 在 Action (動作) 清單中，選擇 Install (安裝)。

7. 在 Name (名稱) 方塊中，輸入 *AmazonCloudWatchAgent*。
8. 將 Version (版本) 維持空白，以安裝代理程式的最新版本。
9. 選擇執行。

下載代理程式套件之後，下一個步驟是設定然後啟動。

(在內部部署伺服器上安裝) 指定 IAM 登入資料和 AWS 區域

若要讓 CloudWatch 代理程式能夠從內部部署伺服器傳送資料，您必須指定先前建立之 IAM 使用者的存取金鑰和秘密金鑰。如需建立此使用者的詳細資訊，請參閱建立 IAM 角色和使用者以搭配 CloudWatch 代理程式使用。

您也必須使用欄位指定要傳送量度的「 AWS 地區`region`」。

以下是此檔案的範例。

```
[AmazonCloudWatchAgent]
aws_access_key_id=my_access_key
aws_secret_access_key=my_secret_key
region = us-west-1
```

針對 *my_access_key* 和 *my_secret_key*，使用沒有寫入 Systems Manager 參數存放區許可的 IAM 使用者金鑰。如需 CloudWatch 代理程式所需 IAM 使用者的詳細資訊，請參閱建立 IAM 使用者，以搭配內部部署伺服器上的 CloudWatch 代理程式。

若您將此描述檔命名為 `AmazonCloudWatchAgent`，您便不需要執行更多作業。您也可以選擇給予它一個不同的名稱，並將名稱指定為 `common-config.toml` 檔案中 `shared_credential_profile` 的值，如下一節所述。

以下是使用aws configure命令為 CloudWatch 代理程式建立具名設定檔的範例。此範例假設您使用 `AmazonCloudWatchAgent` 的預設描述檔名稱。

建立 CloudWatch 代理程式的 AmazonCloudWatchAgent 設定檔

1. 如果您尚未這樣做，請在伺服器 AWS Command Line Interface 上安裝。如需詳細資訊，請參閱安裝 AWS CLI。
2. 在 Linux 伺服器上，輸入以下命令並依提示操作：

```
sudo aws configure --profile AmazonCloudWatchAgent
```

在 Windows Server 上，以系統管理員身分開啟 PowerShell ，輸入下列命令，然後依照提示執行。

```
aws configure --profile AmazonCloudWatchAgent
```

(選擇性) 修改 CloudWatch 代理程式的一般組態和具名設定檔

CloudWatch 代理程式包含一個名為的組態檔common-config.toml。您可以選擇性地使用此檔案來指定代理和區域資訊。

在執行 Linux 的伺服器上，此檔案位於 /opt/aws/amazon-cloudwatch-agent/etc 目錄。在執行 Windows Server 的伺服器上，此檔案位於 C:\ProgramData\Amazon \AmazonCloudWatchAgent 目錄。

預設的 common-config.toml 如下：

```
# This common-config is used to configure items used for both ssm and cloudwatch access


## Configuration for shared credential.
## Default credential strategy will be used if it is absent here:
##            Instance role is used for EC2 case by default.
##            AmazonCloudWatchAgent profile is used for onPremise case by default.
# [credentials]
#    shared_credential_profile = "{profile_name}"
#    shared_credential_file= "{file_name}"

## Configuration for proxy.
## System-wide environment-variable will be read if it is absent here.
## i.e. HTTP_PROXY/http_proxy; HTTPS_PROXY/https_proxy; NO_PROXY/no_proxy
## Note: system-wide environment-variable is not accessible when using ssm run-command.
## Absent in both here and environment-variable means no proxy will be used.
# [proxy]
#    http_proxy = "{http_url}"
#    https_proxy = "{https_url}"
#    no_proxy = "{domain}"
```

最初所有行都會標示為註解。若要設定登入資料設定檔或代理設定，請移除該行的 # 並指定值。您可以手動編輯此檔案，或使用 Systems Manager 中的 RunShellScript 執行命令：

- shared_credential_profile— 對於內部部署伺服器，此行指定要用來傳送資料的 IAM 使用者登入資料設定檔 CloudWatch。若您將此行標示為註解，則會使用 AmazonCloudWatchAgent。如需建立此描述檔的詳細資訊，請參閱(在內部部署伺服器上安裝) 指定 IAM 登入資料和 AWS 區域。

  CloudWatch 在 EC2 執行個體上，您可以使用此行讓 CloudWatch 代理程式將資料從此執行個體傳送到不同的 AWS 區域。若要執行此作業，請指定一個具名描述檔，其中包含指定要傳送對象區域名稱的 region 欄位。

  如果您指定 shared_credential_profile，即必須也要移除 [credentials] 行開頭中的 #。
- shared_credential_file – 若要讓代理程式在位於預設路徑以外路徑的檔案中尋找憑證，請在此處指定該完整路徑及檔案名稱。Linux 的預設路徑是 /root/.aws，Windows Server 的預設路徑是 C:\\Users\\Administrator\\.aws。

  以下第一個範例顯示適用於 Linux 伺服器的 shared_credential_file 行語法，第二個範例則適用於 Windows Server 有效。在 Windows Server 上，您必須跳脫 \ 字元。

  ```
  shared_credential_file= "/usr/username/credentials"
  ```

  ```
  shared_credential_file= "C:\\Documents and Settings\\username\\.aws\\credentials"
  ```

  如果您指定 shared_credential_file，即必須也要移除 [credentials] 行開頭中的 #。
- 代理設定 – 若您的伺服器使用 HTTP 或 HTTPS 代理來和 AWS 服務聯絡，請在 http_proxy 和 https_proxy 欄位中指定那些代理。如有必須排除在代理之外的 URL，請在 no_proxy 欄位中指定並以逗號分隔。

啟動 CloudWatch 代理程式

您可以使用 Systems Manager 執行命令或命令列來啟動 CloudWatch 代理程式。

使用 SSM 代理程式在內部部署伺服器上啟動 CloudWatch 代理程式

1. 開啟 Systems Manager 主控台，網址為 https://console.aws.amazon.com/systems-manager/。
2. 在導覽窗格中，選擇 執行命令。

   -或-

   如果 AWS Systems Manager 首頁開啟，請向下捲動並選擇「瀏覽執行命令」。
3. 選擇 執行命令。

4. 在 [指令] 文件清單中，選取 [AmazonCloudWatch-] 旁邊的按鈕ManageAgent。
5. 在 Targets (目標) 區域中，選取您安裝代理程式的執行個體。
6. 在 Action (動作) 清單中，選擇 configure (設定)。
7. 在 Mode (模式) 清單中，選擇 onPremise (現場部署)。
8. 在 Optional Configuration Location (選用組態位置) 方塊中，輸入您使用精靈建立並存放於參數存放區的代理程式組態檔案名稱。
9. 選擇執行。

   代理程式以您在組態檔案中指定的組態進行啟動。

使用命令列在內部部署伺服器上啟動 CloudWatch 代理程式

- 在此命令中，`-a fetch-config`會使代理程式載入最新版本的 CloudWatch 代理程式組態檔，並`-s`啟動代理程式。

  Linux：如果您已將組態檔案儲存於 Systems Manager 參數存放區，請輸入以下內容：

  ```
  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
  config -m onPremise -s -c ssm:configuration-parameter-store-name
  ```

  Linux：如果您已將組態檔案儲存於本機電腦，請輸入以下命令：以代理程式組態檔的路徑取*configuration-file-path*代。如果您使用精靈建立，則這個檔案稱為 `config.json`；如果您以手動方式建立，則可能稱為 `amazon-cloudwatch-agent.json`。

  ```
  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
  config -m onPremise -s -c file:configuration-file-path
  ```

  Windows Server：如果您將代理程式組態檔儲存在「Systems Manager 參數存放區」中，請從 PowerShell 主控台輸入下列內容：

  ```
  & "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -
  a fetch-config -m onPremise -s -c ssm:configuration-parameter-store-name
  ```

  Windows Server：如果您已將代理程式組態檔儲存在本機電腦上，請從 PowerShell 主控台輸入下列內容。以代理程式組態檔的路徑取*configuration-file-path*代。如果您使用精靈建立，則這個檔案稱為 `config.json`；如果您以手動方式建立，則可能稱為 `amazon-cloudwatch-agent.json`。

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1" -
a fetch-config -m onPremise -s -c file:configuration-file-path
```

使用在新執行個體上安裝 CloudWatch代理程式 AWS CloudFormation

Amazon 已上傳數個 AWS CloudFormation 範本， GitHub 以協助您在新的 Amazon EC2 執行個體上安裝和更新 CloudWatch 代理程式。如需有關使用的詳細資訊 AWS CloudFormation，請參閱什麼是 AWS CloudFormation？ 。

範本位置是使用將 Amazon CloudWatch 代理程式部署到 EC2 執行個體 AWS CloudFormation。此位置同時包含 `inline` 和 `ssm` 目錄。每個目錄都包含適用於 Linux 和 Windows 執行個體的範本。

- `inline`目錄中的範本會將 CloudWatch 代理程式組態內嵌在 AWS CloudFormation 範本中。根據預設，Linux 範本會收集 `mem_used_percent` 和 `swap_used_percent` 指標，而 Windows 範本則會收集 `Memory % Committed Bytes In Use` 和 `Paging File % Usage`。

  若要修改這些範本以收集不同指標，請修改範本的以下區段。以下範例來自適用於 Linux 伺服器的範本。請依照代理程式組態檔案的格式和語法來完成這些變更。如需詳細資訊，請參閱 手動建立或編輯 CloudWatch 代理程式組態檔。

```
{
   "metrics":{
      "append_dimensions":{
         "AutoScalingGroupName":"${!aws:AutoScalingGroupName}",
         "ImageId":"${!aws:ImageId}",
         "InstanceId":"${!aws:InstanceId}",
         "InstanceType":"${!aws:InstanceType}"
      },
      "metrics_collected":{
         "mem":{
            "measurement":[
               "mem_used_percent"
            ]
         },
         "swap":{
            "measurement":[
               "swap_used_percent"
            ]
         }
```

```
        }
    }
}
```

> **Note**
>
> 在內嵌範本中，所有預留位置變數前面必須有驚嘆號 (!) 作為跳脫字元。您可以在範例範本中看到。如果您新增其他預留位置變數，務必在名稱前面加上驚嘆號。

- `ssm` 目錄中的範本會從參數存放區載入代理程式組態檔案。若要使用這些範本，您必須先建立組態檔案並將其上傳到參數存放區。然後，在範本中提供檔案的參數存放區名稱。您可以手動或使用精靈來建立組態檔案。如需詳細資訊，請參閱 建立 CloudWatch 代理程式組態檔。

您可以使用這兩種類型的範本來安裝 CloudWatch 代理程式和更新代理程式組態。

教學課程：使用 AWS CloudFormation 內嵌範本安裝和設定 CloudWatch 代理程式

本教學將逐步引導您如 AWS CloudFormation 何在新的 Amazon EC2 執行個體上安裝 CloudWatch 代理程式。本教學會使用內嵌範本在執行 Amazon Linux 2 的新執行個體上安裝，不需要使用 JSON 組態檔案或參數存放區。內嵌範本在範本中包含代理程式組態。在此教學課程中，您將使用範本包含的預設代理程式組態。

完成安裝代理程式的程序之後，教學課程會繼續說明如何更新代理程式。

用 AWS CloudFormation 於在新執行個體上安裝 CloudWatch 代理程式

1. 從中下載範本 GitHub。在本教學中，下載 Amazon Linux 2 的內嵌範本，如下所示：

   ```
   curl -O https://raw.githubusercontent.com/awslabs/aws-cloudformation-templates/
   master/aws/solutions/AmazonCloudWatchAgent/inline/amazon_linux.template
   ```

2. 請在以下位置開啟 AWS CloudFormation 主控台。 https://console.aws.amazon.com/cloudformation
3. 選擇建立堆疊。
4. 針對 Choose template (選擇範本)，選取 Upload a template to Simple Storage Service (Amazon S3) (將範本上傳至 Simple Storage Service (Amazon S3))，選擇已下載的範本，然後選擇 Next (下一步)。
5. 在 Specify Details (指定詳細資訊) 頁面上，填寫下列參數，然後選擇 Next (下一步)：

- 堆疊名稱：為您的堆疊選擇一個 AWS CloudFormation 堆疊名稱。
- IamRole：選擇具有寫入 CloudWatch 指標、記錄和追蹤權限的 IAM 角色。如需詳細資訊，請參閱 建立 IAM 角色以搭配 Amazon EC2 執行個體上的 CloudWatch 代理程式使用。
- InstanceAMI：選擇在您要啟動堆疊區域中有效的 AMI。
- InstanceType：選擇有效的執行個體類型。
- KeyName：若要啟用 SSH 存取新執行個體，請選擇現有的 Amazon EC2 key pair。如果您還沒有 Amazon EC2 金鑰對，可以在 AWS Management Console中建立一個。如需詳細資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的 Amazon EC2 金鑰對。
- SSHLocation：指定可用於透過 SSH 連接到執行個體的 IP 地址範圍。預設值允許從任何 IP 地址存取。

6. 在 Options (選項) 頁面上，您可以選擇為堆疊資源加上標籤。選擇下一步。
7. 在 Review (審核) 頁面上，審核您的資訊，確認該堆疊可能會建立 IAM 資源，並選擇 Create (建立)。

   如果您重新整理主控台，您會看到新堆疊的狀態為 CREATE_IN_PROGRESS。
8. 執行個體建立後，您就可以在 Amazon EC2 主控台看到它。或者，您可以連線到主機並檢查進度。

   使用以下命令確認已安裝代理程式：

   ```
   rpm -qa amazon-cloudwatch-agent
   ```

   使用以下命令確認代理程式正在執行：

   ```
   ps aux | grep amazon-cloudwatch-agent
   ```

下一個程序示範 AWS CloudFormation 如何使用內嵌範本更新 CloudWatch 代理程式。預設內嵌範本會收集 mem_used_percent 指標。在此教學課程中，您將變更代理程式組態來停止收集該指標。

用來更 AWS CloudFormation 新 CloudWatch 代理程式

1. 在您於上個程序下載的範本中，移除下列幾行，然後儲存範本：

   ```
   "mem": {
   ```

```
    "measurement": [
        "mem_used_percent"
      ]
},
```

2. 請在以下位置開啟 AWS CloudFormation 主控台。 https://console.aws.amazon.com/cloudformation
3. 在 AWS CloudFormation 儀表板上，選取您建立的堆疊，然後選擇 [更新堆疊]。
4. 針對 Select Template (選取範本)，選取 Upload a template to Simple Storage Service (Amazon S3) (將範本上傳至 Simple Storage Service (Amazon S3))，選擇您修改的範本，然後選擇 Next (下一步)。
5. 在 Options (選項) 頁面上，選擇 Next (下一步)，然後選擇 Next (下一步)。
6. 在 Review (檢閱) 頁面上，檢閱您的資訊，然後選擇 Update (更新)。

   一段時間後，您會看到 `UPDATE_COMPLETE`。

教學課程：使用 AWS CloudFormation 和參數存放區安裝 CloudWatch 代理程式

本教學將逐步引導您如 AWS CloudFormation 何在新的 Amazon EC2 執行個體上安裝 CloudWatch 代理程式。此教學安裝使用您在參數存放區中建立並儲存的代理程式組態檔案，在執行 Amazon Linux 2 的新執行個體上安裝。

完成安裝代理程式的程序之後，教學課程會繼續說明如何更新代理程式。

使用 AWS CloudFormation 參數存放區中的組態在新執行個體上安裝 CloudWatch 代理程式

1. 如果您尚未這麼做，請將 CloudWatch 代理程式套件下載到其中一部電腦，以便您可以建立代理程式設定檔。如需詳細資訊及使用參數存放區下載代理程式，請參閱 下載並設定 CloudWatch 代理程式。如需使用命令列下載套件的詳細資訊，請參閱使用命令列下載並設定 CloudWatch代理程式。
2. 建立代理程式組態檔案並將其儲存在參數存放區中。如需詳細資訊，請參閱 建立 CloudWatch 代理程式組態檔。
3. 從 GitHub 下列步驟下載範本：

```
curl -O https://raw.githubusercontent.com/awslabs/aws-cloudformation-templates/
master/aws/solutions/AmazonCloudWatchAgent/ssm/amazon_linux.template
```

4. 請在以下位置開啟 AWS CloudFormation 主控台。 https://console.aws.amazon.com/cloudformation
5. 選擇建立堆疊。
6. 針對 Choose template (選擇範本)，選取 Upload a template to Simple Storage Service (Amazon S3) (將範本上傳至 Simple Storage Service (Amazon S3))，選擇您下載的範本，然後選擇 Next (下一步)。
7. 在 Specify Details (指定詳細資訊) 頁面上，據此填寫下列參數，然後選擇 Next (下一步)。
   - 堆疊名稱：為您的堆疊選擇一個 AWS CloudFormation 堆疊名稱。
   - IamRole：選擇具有寫入 CloudWatch 指標、記錄和追蹤權限的 IAM 角色。如需詳細資訊，請參閱 建立 IAM 角色以搭配 Amazon EC2 執行個體上的 CloudWatch 代理程式使用。
   - InstanceAMI：選擇在您要啟動堆疊區域中有效的 AMI。
   - InstanceType：選擇有效的執行個體類型。
   - KeyName：若要啟用 SSH 存取新執行個體，請選擇現有的 Amazon EC2 key pair。如果您還沒有 Amazon EC2 金鑰對，可以在 AWS Management Console中建立一個。如需詳細資訊，請參閱《Amazon EC2 Linux 執行個體使用者指南》中的 Amazon EC2 金鑰對。
   - SSHLocation：指定可用於透過 SSH 連接到執行個體的 IP 地址範圍。預設值允許從任何 IP 地址存取。
   - SSMKey：指定您在參數存放區中建立並儲存的代理程式組態檔案。
8. 在 Options (選項) 頁面上，您可以選擇為堆疊資源加上標籤。選擇下一步。
9. 在 Review (審核) 頁面上，審核您的資訊，確認該堆疊可能會建立 IAM 資源，並選擇 Create (建立)。

   如果您重新整理主控台，您會看到新堆疊的狀態為 CREATE_IN_PROGRESS。
10. 執行個體建立後，您就可以在 Amazon EC2 主控台看到它。或者，您可以連線到主機並檢查進度。

    使用以下命令確認已安裝代理程式：

    ```
    rpm -qa amazon-cloudwatch-agent
    ```

    使用以下命令確認代理程式正在執行：

    ```
    ps aux | grep amazon-cloudwatch-agent
    ```

下一個程序示範如 AWS CloudFormation 何使用您儲存在參數存放區中的代理程式組態來更新代理程式。 CloudWatch

使用 AWS CloudFormation 來使用參數存放區中的組態更新 CloudWatch 代理程式

1. 將參數存放區中存放的代理程式組態檔案變更為您希望的新組態。
2. 在您在the section called “教學課程：使用 AWS CloudFormation 和參數存放區安裝 CloudWatch 代理程式”主題中下載的 AWS CloudFormation 範本中，變更版本號碼。例如，您可能將 `VERSION=1.0` 變更為 `VERSION=2.0`。
3. 請在以下位置開啟 AWS CloudFormation 主控台。 https://console.aws.amazon.com/cloudformation
4. 在 AWS CloudFormation 儀表板上，選取您建立的堆疊，然後選擇 [更新堆疊]。
5. 針對 Select Template (選取範本)，選取 Upload a template to Simple Storage Service (Amazon S3) (將範本上傳至 Simple Storage Service (Amazon S3))，選取您剛修改的範本，然後選擇 Next (下一步)。
6. 在 Options (選項) 頁面上，選擇 Next (下一步)，然後選擇 Next (下一步)。
7. 在 Review (檢閱) 頁面上，檢閱您的資訊，然後選擇 Update (更新)。

   一段時間後，您會看到 `UPDATE_COMPLETE`。

疑難排解安裝代 CloudWatch 理程式 AWS CloudFormation

本節可協助您疑難排解使用安裝和更新 CloudWatch 代理程式時發生的問題 AWS CloudFormation。

當更新失敗時偵測

如果您使用 AWS CloudFormation 更新 CloudWatch 代理程式組態並使用無效的組態設定，則代理程式會停止將任何指標傳送至 CloudWatch。檢查代理程式組態更新是否成功的快速方法是查看 `cfn-init-cmd.log` 檔案。在 Linux 伺服器上，此檔案位於 `/var/log/cfn-init-cmd.log`。在 Windows 執行個體上，此檔案位於 `C:\cfn\log\cfn-init-cmd.log`。

遺失指標

如果您在安裝或更新代理程式之後沒看到預料會看到的指標，請確認已將代理程式設定為收集該指標。若要執行此作業，請檢查 `amazon-cloudwatch-agent.json` 檔案，以確定其中列出該指標，而且您查看的是正確的指標命名空間。如需詳細資訊，請參閱 CloudWatch 代理程式檔案和位置。

CloudWatch 代理程式認證偏

本節概述 CloudWatch 代理程式在與其他 AWS 服務和 API 通訊時用來取得認證的認證提供者鏈結。順序如下。下列清單中第二到第五的偏好設定與 AWS SDK 中定義的偏好設定順序相同。如需詳細資訊，請參閱 SDK 文件中的指定認證。

1. CloudWatch 代理程式檔案中定義的共用組態和認證`common-config.toml`檔案。如需詳細資訊，請參閱 (選用) 修改代理或區域資訊的常見組態。
2. AWS SDK 環境變數

> **Important**
>
> 在 Linux 上，如果您使用指令碼執行 CloudWatch 代理程式，則`amazon-cloudwatch-agent-ctl`指令碼會以`systemd`服務的形式啟動代理程式。在此情況下，代理程式無法存取環境變數 (例如`AWS_ACCESS_KEY_ID`、和`AWS_SECRET_ACCESS_KEY`)。`HOME`

3. 在中找到的共用組態和認證檔 `$HOME/%USERPROFILE%`

> **Note**
>
> 該 CloudWatch 代理程式會`.aws/credentials`在中`$HOME`尋找 Linux 和 MacOS，並`%USERPROFILE%`尋找視窗。與 AWS SDK 不同的是，如果無法存取環境變數，則 CloudWatch 代理程式沒有後援方法來判斷主目錄。這種行為的差異在於保持與早期 AWS SDK 實現的向後兼容性。
>
> 此外，與中找到的共用認證不同`common-config.toml`，如果 AWS SDK 衍生的共用認證過期並輪替，則代理程式不會自動拾取更新的認證，而需要重新啟動 CloudWatch 代理程式才能執行此操作。

4. 如果存在使用 Amazon 彈性容器服務任務定義或 RunTask API 作業的應用程式，則為任務的 AWS Identity and Access Management 角色。
5. 連接至 Amazon EC2 執行個體的執行個體設定檔。

最佳作法是，建議您在使用 CloudWatch 代理程式時，依下列順序指定認證。

1. 如果您的應用程式使用 Amazon 彈性容器服務任務定義或 RunTask API 作業，請將 IAM 角色用於任務。
2. 如果您的應用程式在 Amazon EC2 執行個體上執行，請使用 IAM 角色。

3. 使用 CloudWatch 代理程式`common-config.toml`檔案來指定認證檔案。此認證檔案與其他 AWS SDK 和. AWS CLI如果您已經在使用共用認證檔案，也可以將其用於此目的。如果您使用 CloudWatch 代理程式的`common-config.toml`檔案提供，則可確保代理程式在過期時會耗用輪換的認證，並取代，而不需要您重新啟動代理程式。
4. 使用環境變數。如果您在 Amazon EC2 執行個體以外的電腦上進行開發工作，則設定環境變數非常有用。

> **Note**
>
> 如果您如中所述將遙測傳送至其他帳戶[將指標、日誌和追蹤傳送到不同帳戶]，則 CloudWatch 代理程式會使用本節所述的認證提供者鏈結來取得初始認證集。接著，當假設 CloudWatch 代理程式組態檔`role_arn`中指定的 IAM 角色時，會使用這些登入資料。

驗證代 CloudWatch 理程式套件的簽章

Linux 伺服器上的 CloudWatch 代理程式套件包含 GPG 簽章檔案。您可以使用公開金鑰來驗證代理程式下載檔案是否為原版且未經修改。

針對 Windows Server，您可以使用 MSI 來驗證簽章。

對於 macOS 電腦，簽章會包含在代理程式下載套件中。

若要尋找正確的簽章檔案，請參閱下表。針對每個架構和作業系統，各自都有一般連結和區域的連結。例如，對於 Amazon Linux 2023 和 Amazon Linux 2 和 x86-64 架構，其中三個有效鏈接是：

- `https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm.sig`
- `https://amazoncloudwatch-agent-us-east-1.s3.us-east-1.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`
- `https://amazoncloudwatch-agent-eu-central-1.s3.eu-central-1.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent.rpm`

> **Note**
>
> 若要下載 CloudWatch 代理程式，您的連線必須使用 TLS 1.2 或更新版本。

架構	平台	下載連結	簽章檔案連結
x86-64	Amazon Linux 2023 和 Amazon Linux 2	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ##. 亞馬遜亞馬遜亞馬遜 /amd64 /最新/ amazon-cloudwatch-agent 轉	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ##. 亞馬遜亞馬遜亞馬遜/amd64 /最新/ .rpm.sig amazon-cloudwatch-agent
x86-64	Centos	https://amazoncloudwatch-agent.s3.amazonaws.com/centos/amd64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ##. 亞馬遜amazon-cloudwatch-agent公司/百分之六十四/最新/轉	https://amazoncloudwatch-agent.s3.amazonaws.com/centos/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ##. 亞馬遜amazon-cloudwatch-agent公司/百分之六十四/最新/.rpm.sig
x86-64	Redhat	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/amd64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ##. #####/紅帽 /amd64 /最新/ amazon-cloudwatch-agent 轉	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ##. 亞馬遜amazon-cloudwatch-agent公司/紅帽 /amd64 /最新/ .rpm.sig

架構	平台	下載連結	簽章檔案連結
x86-64	SUSE	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/amd64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ##. 亞馬遜公司/蘇/amazon-cloudwatch-agent安 64 /最新/ 轉	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ##. 亞馬遜amazon-cloudwatch-agent公司/蘇聯/安德 64 /最新/ .rpm.sig
x86-64	Debian	https://amazoncloudwatch-agent.s3.amazonaws.com/debian/amd64/latest/ amazon-cloudwatch-agent .deb https://amazoncloudwatch-agent-##. ##. ###. COM /德比安/amd64 /最新/ amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/debian/amd64/latest/amazon-cloudwatch-agent. 德博 https://amazoncloudwatch-agent-##. ##. ##### / debi an /amd64 /最新/.deb. sig amazon-cloudwatch-agent
x86-64	Ubuntu	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/amd64/latest/ amazon-cloudwatch-agent .deb https://amazoncloudwatch-agent-##. ##. 亞馬遜公司/企業/amazon-cloudwatch-agent上午 64 /最新/	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/amd64/latest/amazon-cloudwatch-agent. 德博 https://amazoncloudwatch-agent-##. ##. #####/##/日期 64 /最新/.deb.sig amazon-cloudwatch-agent

架構	平台	下載連結	簽章檔案連結
x86-64	Oracle	https://amazoncloudwatch-agent.s3.amazonaws.com/oracle_linux/amd64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ## .###/#_lin ux/amd64 /最新/轉速 amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/oracle_linux/amd64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ## .###/###_lin ux/amd64 /最新/.rpm.sig amazon-cloudwatch-agent
x86-64	macOS	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/amd64/latest/ amazon-cloudwatch-agent .pkg https://amazoncloudwatch-agent-##. ## .###/###/AMD64 /最新/ .pkg amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/darwin/amd64/latest/ www.amazon-cloudwatch-agent pkg.sig https://amazoncloudwatch-agent-##. ##. ###/## #/ amd64 /最新/ .pkg.sig amazon-cloudwatch-agent
x86-64	Windows	https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent. 微星 https://amazoncloudwatch-agent-##. ##. ###/#口/amd64 /最新/. amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/windows/amd64/latest/amazon-cloudwatch-agent. 微笑 https://amazoncloudwatch-agent-##. ## .###/## / amd64 /最新/.msi.sig amazon-cloudwatch-agent

架構	平台	下載連結	簽章檔案連結
ARM64	Amazon Linux 2023 和 Amazon Linux 2	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/arm64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ##. 亞馬遜亞馬遜亞馬遜 /arm64 /最新/ amazon-cloudwatch-agent 轉	https://amazoncloudwatch-agent.s3.amazonaws.com/amazon_linux/arm64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ##. 亞馬遜亞馬遜亞馬遜 /arm64 /最新/.rpm.sig amazon-cloudwatch-agent
ARM64	Redhat	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/arm64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-##. ########/ 紅帽/ARM64 /最新/ amazon-cloudwatch-agent 轉	https://amazoncloudwatch-agent.s3.amazonaws.com/redhat/arm64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-##. ##. 亞馬遜amazon-cloudwatch-agent公司/紅帽 / arm64 /最新/ .rpm.sig
ARM64	Ubuntu	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/arm64/latest/ amazon-cloudwatch-agent .deb https://amazoncloudwatch-agent-##. ##. 亞馬遜公司/管理/ARM64 /最新/ amazon-cloudwatch-agent	https://amazoncloudwatch-agent.s3.amazonaws.com/ubuntu/arm64/latest/amazon-cloudwatch-agent. 德博 https://amazoncloudwatch-agent-##. ##. ##### / ubun tu/ arm64 /最新/.deb.sig amazon-cloudwatch-agent

架構	平台	下載連結	簽章檔案連結
ARM64	SUSE	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/arm64/latest/amazon-cloudwatch-agent. 每分鐘 https://amazoncloudwatch-agent-*##*. *##*. *###*. *COM* / 蘇打/arm64 /最新/ amazon-cloudwatch-agent 轉	https://amazoncloudwatch-agent.s3.amazonaws.com/suse/arm64/latest/ amazon-cloudwatch-agent .rpm.sig https://amazoncloudwatch-agent-*##*. *##*. *###*. *COM* / *#*打/arm64 /最新/ .rpm.sig amazon-cloudwatch-agent

驗證 Linux 伺服器上的 CloudWatch 代理程式套件

1. 下載公開金鑰。

```
shell$ wget https://amazoncloudwatch-agent.s3.amazonaws.com/assets/amazon-
cloudwatch-agent.gpg
```

2. 將公開金鑰匯入至您的 keyring。

```
shell$  gpg --import amazon-cloudwatch-agent.gpg
gpg: key 3B789C72: public key "Amazon CloudWatch Agent" imported
gpg: Total number processed: 1
gpg: imported: 1 (RSA: 1)
```

   請記下金鑰值，在後續步驟您將會用到它。在前面的範例中，金鑰值為 3B789C72。

3. 執行以下命令以驗證指紋，請以三個步驟的值取代 *key-value*：

```
shell$  gpg --fingerprint key-value
pub   2048R/3B789C72 2017-11-14
      Key fingerprint = 9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
uid                  Amazon CloudWatch Agent
```

   指紋字串應等於：

   9376 16F3 450B 7D80 6CBD 9725 D581 6730 3B78 9C72

   若指紋字串不相符，請勿安裝代理程式。請聯絡 Amazon Web Services。

驗證指紋後，您可以使用它來驗證 CloudWatch 代理程式套件的簽章。

4. 使用 wget 下載套件簽章檔案。若要決定正確的簽章檔案，請參閱上表。

```
wget Signature File Link
```

5. 若要驗證簽章，請執行 gpg --verify。

```
shell$ gpg --verify signature-filename agent-download-filename
gpg: Signature made Wed 29 Nov 2017 03:00:59 PM PST using RSA key ID 3B789C72
gpg: Good signature from "Amazon CloudWatch Agent"
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: 9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
```

如果輸出包含 BAD `signature` 片語，請檢查您是否已正確執行程序。如果您持續收到此回應，請聯絡 Amazon Web Services，並避免使用已下載的檔案。

請注意有關信任的警告。只有您或您信任者所簽章的金鑰才能信任。這不表示該簽章是無效的，只是您尚未驗證該公有金鑰。

驗證執行 Windows 伺服器之伺服器上的 CloudWatch 代理程式套件

1. 從 https://gnupg.org/download/ 下載並安裝適用於 Windows 的 GnuPG。安裝時，請包括外圍程序擴展（GpgEx）選項。

   您可以在視窗中執行其餘步驟 PowerShell。

2. 下載公開金鑰。

```
PS> wget https://amazoncloudwatch-agent.s3.amazonaws.com/assets/amazon-cloudwatch-
agent.gpg -OutFile amazon-cloudwatch-agent.gpg
```

3. 將公開金鑰匯入至您的 keyring。

```
PS>  gpg --import amazon-cloudwatch-agent.gpg
gpg: key 3B789C72: public key "Amazon CloudWatch Agent" imported
gpg: Total number processed: 1
gpg: imported: 1 (RSA: 1)
```

請記下金鑰的值，因為您的下一個步驟將需要它。在前面的範例中，金鑰值為 3B789C72。

4. 執行以下命令以驗證指紋，請以三個步驟的值取代 *key-value*：

```
PS>  gpg --fingerprint key-value
pub   rsa2048 2017-11-14 [SC]
      9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
uid           [ unknown] Amazon CloudWatch Agent
```

指紋字串應等於：

```
9376 16F3 450B 7D80 6CBD 9725 D581 6730 3B78 9C72
```

若指紋字串不相符，請勿安裝代理程式。請聯絡 Amazon Web Services。

驗證指紋後，您可以使用它來驗證 CloudWatch 代理程式套件的簽章。

5. 使用 wget 下載套件簽章檔案。如果要判斷正確的簽章檔案，請參閱CloudWatch 代理程式下載連結。
6. 若要驗證簽章，請執行 gpg --verify。

```
PS> gpg --verify sig-filename agent-download-filename
gpg: Signature made 11/29/17 23:00:45 Coordinated Universal Time
gpg:                using RSA key D58167303B789C72
gpg: Good signature from "Amazon CloudWatch Agent" [unknown]
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: 9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
```

如果輸出包含 BAD `signature` 片語，請檢查您是否已正確執行程序。如果您持續收到此回應，請聯絡 Amazon Web Services，並避免使用已下載的檔案。

請注意有關信任的警告。只有您或您信任者所簽章的金鑰才能信任。這不表示該簽章是無效的，只是您尚未驗證該公有金鑰。

驗證 macOS 電腦上的 CloudWatch 代理程式套件

- 有兩種方法可在 macOS 上進行簽章驗證。
  - 執行下列命令，驗證指紋：

```
pkgutil --check-signature amazon-cloudwatch-agent.pkg
```

您應該會看到類似以下的結果。

```
Package "amazon-cloudwatch-agent.pkg":
       Status: signed by a developer certificate issued by Apple for
 distribution
       Signed with a trusted timestamp on: 2020-10-02 18:13:24 +0000
       Certificate Chain:
       1. Developer ID Installer: AMZN Mobile LLC (94KV3E626L)
       Expires: 2024-10-18 22:31:30 +0000
       SHA256 Fingerprint:
       81 B4 6F AF 1C CA E1 E8 3C 6F FB 9E 52 5E 84 02 6E 7F 17 21 8E FB
       0C 40 79 13 66 8D 9F 1F 10 1C

 ------------------------------------------------------------------------
       2. Developer ID Certification Authority
       Expires: 2027-02-01 22:12:15 +0000
       SHA256 Fingerprint:
       7A FC 9D 01 A6 2F 03 A2 DE 96 37 93 6D 4A FE 68 09 0D 2D E1 8D 03
       F2 9C 88 CF B0 B1 BA 63 58 7F

 ------------------------------------------------------------------------
       3. Apple Root CA
       Expires: 2035-02-09 21:40:36 +0000
       SHA256 Fingerprint:
       B0 B1 73 0E CB C7 FF 45 05 14 2C 49 F1 29 5E 6E DA 6B CA ED 7E 2C
       68 C5 BE 91 B5 A1 10 01 F0 24
```

- 或者，下載並使用 .sig 檔案。若要使用這個方法，請依照以下步驟。
  - 輸入以下命令來將 GPG 應用程式安裝到您的 macOS 主機。

    ```
    brew install GnuPG
    ```

- 使用 curl 下載套件簽章檔案。如果要判斷正確的簽章檔案，請參閱CloudWatch 代理程式下載連結。
- 若要驗證簽章，請執行 gpg --verify。

```
PS> gpg --verify sig-filename agent-download-filename
gpg: Signature made 11/29/17 23:00:45 Coordinated Universal Time
gpg:                using RSA key D58167303B789C72
gpg: Good signature from "Amazon CloudWatch Agent" [unknown]
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: 9376 16F3 450B 7D80 6CBD  9725 D581 6730 3B78 9C72
```

如果輸出包含 BAD `signature` 片語，請檢查您是否已正確執行程序。如果您持續收到此回應，請聯絡 Amazon Web Services，並避免使用已下載的檔案。

請注意有關信任的警告。只有您或您信任者所簽章的金鑰才能信任。這不表示該簽章是無效的，只是您尚未驗證該公有金鑰。

建立 CloudWatch 代理程式組態檔

在任何伺服器上執行 CloudWatch 代理程式之前，您必須先建立一或多個 CloudWatch 代理程式組態檔。

代理程式組態檔案是一種 JSON 檔案，可指定代理程式收集的指標、日誌和追蹤，包含自訂指標。您可以使用精靈來建立，或從零開始自行建立。您也可以使用精靈來開始建立組態檔案，然後手動修改。如果您以手動方式建立或修改檔案，程序將更為複雜，但您對於收集的指標將有更多的控制權，而且可以指定精靈中無法使用的指標。

無論何時，只要變更代理程式組態檔案，您就必須重新啟動代理程式使變更生效。若要重新啟動代理程式，請按照 啟動 CloudWatch 代理程式 中的說明操作。

在您建立一個組態檔案後，您可以手動將它儲存為 JSON 檔案並在您的伺服器上安裝代理程式時使用此檔案。或者，若是您在伺服器上安裝代理程式時需要使用 Systems Manager，您可以將它存放在 Systems Manager 參數存放區中。

CloudWatch 代理程式支援使用多個組態檔。如需詳細資訊，請參閱 多個 CloudWatch 代理程式組態檔。

代理程式收集的指標、記錄檔和追蹤會 CloudWatch 產生費用。如需有關定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

目錄

- 使用精靈建立 CloudWatch 代理程式組態檔
- 手動建立或編輯 CloudWatch 代理程式組態檔

使用精靈建立 CloudWatch 代理程式組態檔

代理程式組態檔精靈會詢問一系列問題amazon-cloudwatch-agent-config-wizard，以協助您根據需求設定 CloudWatch 代理程式。

必要的認證

如果您在啟動精靈之前擁有認證和組態檔案，精靈可以自動偵測 AWS 認證和要使用的 AWS 區域。如需有關這些檔案的詳細資訊，請參閱《AWS Systems Manager 使用者指南》中的組態與憑證檔案。

在 AWS 認證檔案中，精靈會檢查預設認證，並尋找如下所示的AmazonCloudWatchAgent區段：

```
[AmazonCloudWatchAgent]
aws_access_key_id = my_access_key
aws_secret_access_key = my_secret_key
```

精靈會顯示預設登入資料、來自 AmazonCloudWatchAgent 的登入資料，以及 Others 選項。您可以選取要使用的登入資料。若您選擇 Others，您可以輸入登入資料。

針對 *my_access_key* 和 *my_secret_key*，使用具有寫入 Systems Manager 參數存放區許可的 IAM 使用者金鑰。如需 CloudWatch 代理程式所需 IAM 使用者的詳細資訊，請參閱建立 IAM 使用者，以搭配內部部署伺服器上的 CloudWatch 代理程式。

在 AWS 組態檔案中，如果代理程式與區[default]段不同，您可以指定代理程式傳送量度的目標區域。預設為將指標發佈至 Amazon EC2 執行個體所在的區域。如果指標應發佈至不同的區域，請在此指定區域。在下列範例中，會將指標發佈至 us-west-1 區域。

```
[AmazonCloudWatchAgent]
region = us-west-1
```

執行 CloudWatch 代理程式設定精靈

建立 CloudWatch 代理程式組態檔

1. 輸入下列內容以啟動 CloudWatch 代理程式設定精靈：

   ```
   sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-config-wizard
   ```

   在執行 Windows Server 的伺服器上，執行下列命令來啟動精靈：

```
cd "C:\Program Files\Amazon\AmazonCloudWatchAgent"
```

```
.\amazon-cloudwatch-agent-config-wizard.exe
```

2. 回答問題來為您的伺服器自訂組態檔案。
3. 如果您將組態檔案存放於本機，組態檔案 `config.json` 會存放在 Linux 伺服器上的 `/opt/aws/amazon-cloudwatch-agent/bin/` 中，在 Windows Server 上則存放於 `C:\Program Files\Amazon\AmazonCloudWatchAgent` 中。您接著可以將此檔案複製到其他您希望安裝代理程式的伺服器。

   如果您要使用 Systems Manager 來安裝和設定代理程式，在提示是否要將檔案存放於 Systems Manager 參數存放區時，請務必回答 Yes (是)。即使您未使用 SSM 代理程式來安裝代理程式，也可以選擇將檔案儲存在參數存放區中 CloudWatch 。若要能夠將檔案存放於參數存放區，您必須使用具有足夠許可的 IAM 角色。如需詳細資訊，請參閱 建立 IAM 角色和使用者以搭配 CloudWatch 代理程式使用。

CloudWatch 代理程式預先定義的

精靈已使用預先定義的指標集進行設定，包括不同的詳細資訊層級。這些指標集如下表所示。如需這些指標的詳細資訊，請參閱 CloudWatch代理程式收集的測量結果。

> **Note**
>
> 參數存放區支援標準和進階層的參數。這些參數層與這些表格中所述的基本、標準及進階層級的指標詳細資訊無關。

在 Linux 上執行的 Amazon EC2 執行個體

詳細資訊層級	包含的指標
基本	Mem：mem_used_percent 磁碟：disk_used_percent `disk_used_percent` 等這類 `disk` 指標會包含 `Partition` 的維度，表示所產生的自訂指標數依存於與您執行個體相關聯的分割區數。您擁有的磁碟

詳細資訊層級	包含的指標
	分割區數量取決於您使用的 AMI，以及連接到伺服器的 Amazon EBS 磁碟區數量。
標準	CPU：cpu_usage_idle 、cpu_usage_iowait 、cpu_usage_user 、cpu_usage_system Disk：disk_used_percent 、disk_inodes_free Diskio：diskio_io_time Mem：mem_used_percent Swap：swap_used_percent
Advanced (進階)	CPU：cpu_usage_idle 、cpu_usage_iowait 、cpu_usage_user 、cpu_usage_system Disk：disk_used_percent 、disk_inodes_free Diskio：diskio_io_time 、diskio_write_bytes 、diskio_read_bytes 、diskio_writes 、diskio_reads Mem：mem_used_percent Netstat：netstat_tcp_established 、netstat_tcp_time_wait Swap：swap_used_percent

執行 Linux 的現場部署伺服器

詳細資訊層級	包含的指標
基本	Disk：disk_used_percent Diskio：diskio_write_bytes 、diskio_read_bytes 、diskio_writes 、diskio_reads Mem：mem_used_percent

詳細資訊層級	包含的指標
	Net：net_bytes_sent 、net_bytes_recv 、net_packets_sent 、net_packets_recv Swap：swap_used_percent
標準	CPU：cpu_usage_idle 、cpu_usage_iowait Disk：disk_used_percent 、disk_inodes_free Diskio：diskio_io_time 、diskio_write_bytes 、diskio_read_bytes 、diskio_writes 、diskio_reads Mem：mem_used_percent Net：net_bytes_sent 、net_bytes_recv 、net_packets_sent 、net_packets_recv Swap：swap_used_percent
Advanced (進階)	CPU：cpu_usage_guest 、cpu_usage_idle 、cpu_usage_iowait 、cpu_usage_steal 、cpu_usage_user 、cpu_usage_system Disk：disk_used_percent 、disk_inodes_free Diskio：diskio_io_time 、diskio_write_bytes 、diskio_read_bytes 、diskio_writes 、diskio_reads Mem：mem_used_percent Net：net_bytes_sent 、net_bytes_recv 、net_packets_sent 、net_packets_recv Netstat：netstat_tcp_established 、netstat_tcp_time_wait Swap：swap_used_percent

在 Windows Server 上執行的 Amazon EC2 執行個體

> **Note**
>
> 此表格列出的指標名稱會顯示在主控台中檢視時的指標顯示方式。實際的指標名稱可能不包含第一個字。例如，LogicalDisk % Free Space 的實際指標名稱只有 % Free Space。

詳細資訊層級	包含的指標
基本	Memory：Memory % Committed Bytes In Use LogicalDisk: LogicalDisk % Free Space
標準	Memory：Memory % Committed Bytes In Use Paging：Paging File % Usage Processor：Processor % Idle Time、Processor % Interrupt Time、Processor % User Time PhysicalDisk: PhysicalDisk % Disk Time LogicalDisk: LogicalDisk % Free Space
Advanced (進階)	Memory：Memory % Committed Bytes In Use Paging：Paging File % Usage Processor：Processor % Idle Time、Processor % Interrupt Time、Processor % User Time LogicalDisk: LogicalDisk % Free Space PhysicalDisk: PhysicalDisk % Disk Time，PhysicalDisk Disk Write Bytes/sec，PhysicalDisk Disk Read Bytes/sec，PhysicalDisk Disk Writes/sec，PhysicalDisk Disk Reads/sec TCP：TCPv4 Connections Established、TCPv6 Connections Established

執行 Windows Server 的現場部署伺服器

> **Note**
>
> 此表格列出的指標名稱會顯示在主控台中檢視時的指標顯示方式。實際的指標名稱可能不包含第一個字。例如，LogicalDisk % Free Space 的實際指標名稱只有 % Free Space。

詳細資訊層級	包含的指標
基本	Paging：Paging File % Usage Processor：Processor % Processor Time LogicalDisk:LogicalDisk % Free Space PhysicalDisk: PhysicalDisk Disk Write Bytes/sec ，PhysicalDisk Disk Read Bytes/sec ，PhysicalDisk Disk Writes/sec ，PhysicalDisk Disk Reads/sec Memory：Memory % Committed Bytes In Use Network Interface：Network Interface Bytes Sent/sec、Network Interface Bytes Received/sec 、Network Interface Packets Sent/sec、Network Interface Packets Received/sec
標準	Paging：Paging File % Usage Processor：Processor % Processor Time、Processor % Idle Time、Processor % Interrupt Time LogicalDisk: LogicalDisk % Free Space PhysicalDisk: PhysicalDisk % Disk Time ，PhysicalDisk Disk Write Bytes/sec ，PhysicalDisk Disk Read Bytes/sec ，PhysicalDisk Disk Writes/sec ，PhysicalDisk Disk Reads/sec Memory：Memory % Committed Bytes In Use

詳細資訊層級	包含的指標
	Network Interface：Network Interface Bytes Sent/sec、Network Interface Bytes Received/sec 、Network Interface Packets Sent/sec、Network Interface Packets Received/sec
Advanced (進階)	Paging：Paging File % Usage Processor：Processor % Processor Time、Processor % Idle Time、Processor % Interrupt Time、Processor % User Time LogicalDisk: LogicalDisk % Free Space PhysicalDisk: PhysicalDisk % Disk Time ，PhysicalDisk Disk Write Bytes/sec ，PhysicalDisk Disk Read Bytes/sec ，PhysicalDisk Disk Writes/sec ，PhysicalDisk Disk Reads/sec Memory：Memory % Committed Bytes In Use Network Interface：Network Interface Bytes Sent/sec、Network Interface Bytes Received/sec 、Network Interface Packets Sent/sec、Network Interface Packets Received/sec TCP：TCPv4 Connections Established 、TCPv6 Connections Established

手動建立或編輯 CloudWatch 代理程式組態檔

CloudWatch 代理程式組態檔案是一個 JSON 檔案，其中包含四個區段agentmetricslogs、traces、和，說明如下：

- agent 區段包括代理程式整體組態的各個欄位。
- 此metrics區段會指定收集和發佈至的自訂量度 CloudWatch。如果您僅使用代理程式來收集日誌，您可以省略檔案中的 metrics 區段。
- 此logs區段會指定要將哪些記錄檔發佈至 CloudWatch 記錄檔。如果伺服器執行 Windows Server，這部分可以包含來自 Windows 事件記錄的事件。
- 此traces段落會指定收集並傳送至的追蹤來源 AWS X-Ray。

以下章節說明此 JSON 檔案的架構和欄位。您也可以查看此組態檔案的結構描述定義。結構描述定義位於 Linux 伺服器上的 *installation-directory*/doc/amazon-cloudwatch-agent-schema.json，以及執行 Windows Server 的伺服器上的 *installation-directory*/amazon-cloudwatch-agent-schema.json。

如果您手動建立或編輯 代理程式組態檔案，您可以給予它任何名稱。若要簡化故障診斷，我們建議您在 Linux 伺服器上將它命名為 /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json，且在執行 Windows Server 的伺服器上將它命名為 $Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json。在您建立檔案後，您可以將此檔案複製到其他您希望安裝代理程式的伺服器。

> **Note**
>
> 代理程式收集的指標、記錄檔和追蹤會 CloudWatch 產生費用。如需有關定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

CloudWatch 代理程式組態檔案：代理程式

agent 區段可以包含下列欄位。精靈不會建立 agent 區段。反之，精靈會省略它並在此區段的所有欄位中使用預設值。

- metrics_collection_interval – 選用。指定此組態檔案中指定的所有指標的收集頻率。您可以針對特定指標類型覆寫此值。

  此值是以秒數指定。例如，指定每隔 10 秒收集 10 個原因指標，也就是將其設定為每 5 分鐘收集 300 個指定指標。

  如果您將此值設為低於 60 秒，每個指標都將以高解析度指標進行收集。如需高解析度指標的詳細資訊，請參閱 高解析度指標。

  預設值為 60。

- region— 指定監控 Amazon EC2 執行個體時要用於 CloudWatch 端點的區域。收集的指標會傳送到這個區域，例如 us-west-1。如果您省略此欄位，代理程式會將指標傳送至 Amazon EC2 執行個體所在的區域。

  如果您要監控現場部署伺服器，將不會使用此欄位；代理程式會從 AWS 組態檔案的 AmazonCloudWatchAgent 描述檔中讀取區域。

- `credentials`— 指定將指標、記錄和追蹤傳送至其他 AWS 帳戶時要使用的 IAM 角色。若有指定，這個欄位會有一個 `role_arn` 參數。
  - `role_arn` – 指定將指標、日誌和追蹤傳送給不同 AWS 帳戶時，用於身分驗證的 IAM 角色的 Amazon Resource Name (ARN)。如需詳細資訊，請參閱 將指標、日誌和追蹤傳送到不同帳戶。
- `debug` – 選用。指定使用偵錯記錄檔訊息執行 CloudWatch 代理程式。預設值為 `false`。
- `aws_sdk_log_level` – 選用。僅在代理程式版本 1.247350.0 及更新版本中受支援。 CloudWatch

  您可以指定此欄位，讓代理程式對 AWS SDK 端點執行記錄。此欄位的值可包含下列選項中的一或多項。如有多個選項，請使用 | 字元進行分隔。
  - `LogDebug`
  - `LogDebugWithSigning`
  - `LogDebugWithHTTPBody`
  - `LogDebugRequestRetries`
  - `LogDebugWithEventStreamBody`

  如需這些選項的更多資訊，請參閱LogLevelType。
- `logfile`— 指定 CloudWatch 代理程式寫入記錄訊息的位置。如果您指定空白字串、日誌將移至 stderr。如果您未指定此選項，預設位置如下：
  - Linux：`/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log`
  - Windows Server：`c:\\ProgramData\\Amazon\\CloudWatchAgent\\Logs\\amazon-cloudwatch-agent.log`

  CloudWatch 代理程式會自動輪替所建立的記錄檔。日誌檔案大小達到 100 MB 時即輪換。代理最多保留 7 天輪換的日誌檔，而且最多保留五個已輪換的備份日誌檔。備份日誌檔案的檔案名稱會附加時間戳記。時間戳記會顯示檔案輪換的日期和時間：例如，`amazon-cloudwatch-agent-2018-06-08T21-01-50.247.log.gz`。
- `omit_hostname` – 選用。根據預設，主機名稱會發佈為代理收集的指標維度，除非您使用 `metrics` 區段中的 `append_dimensions` 欄位。將 `omit_hostname` 設定為 `true`，以防止主機名稱發佈為維度，即使您並未使用 `append_dimensions`。預設值為 `false`。
- `run_as_user` – 選用。指定用來執行 CloudWatch 代理程式的使用者。如果不指定此參數，則會使用根使用者。此選項只適用於 Linux 伺服器。

  如果您指定此選項，則使用者必須在啟動 CloudWatch 代理程式之前存在。如需詳細資訊，請參閱 以不同的使用者身分執行 CloudWatch 代理程式。

- user_agent – 選用。指定 CloudWatch 代理程式對 CloudWatch 後端進行 API 呼叫時所使用的user-agent字串。預設值是由代理程式版本、用來編譯代理程式的 Go 程式設計語言版本、執行時間作業系統和架構、建置時間以及啟用的外掛程式組成的字串。
- usage_data - 選用。根據預設，每當 CloudWatch 代理程式發佈指標或記錄到 CloudWatch 時，代理程式就會傳送有關本身的健全狀況和效能資料 CloudWatch 此資料不會產生任何費用。您可以針對 usage_data 指定 false，防止代理程式傳送此資料。如果您省略此參數，會使用預設值 true，且代理程式會傳送運作狀態和效能資料。

  如果將此值設定為 false，必須停止並重新啟動代理程式，設定才會生效。

以下是 agent 區段的範例。

```
"agent": {
   "metrics_collection_interval": 60,
   "region": "us-west-1",
   "logfile": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log",
   "debug": false,
   "run_as_user": "cwagent"
  }
```

CloudWatch 代理程式組態檔：測量結果段

Linux 和視窗常見的欄位

在執行 Linux 或 Windows Server 的伺服器上，metrics 區段包含下列欄位：

- namespace – 選用。用於代理程式所收集指標的命名空間。預設值為 CWAgent。長度上限為 255 個字元。以下是範例：

  ```
  {
    "metrics": {
      "namespace": "Development/Product1Metrics",
     ......
     },
  }
  ```

- append_dimensions – 選用。將 Amazon EC2 指標維度新增至代理程式收集的所有指標。這也會導致代理程式無法將主機名稱發佈為維度。

下列清單顯示了 append_dimensions 的唯一支援的鍵/值對。其他鍵/值對會被忽略。代理程式支援這些鍵值組，如以下清單中所示。您無法變更鍵值，為其發布其他維度名稱。

- "ImageId":"${aws:ImageId}" 會將執行個體的 AMI ID 設定為 ImageId 維度的值。
- "InstanceId":"${aws:InstanceId}" 會將執行個體的執行個體 ID 設定為 InstanceId 維度的值。
- "InstanceType":"${aws:InstanceType}" 會將執行個體的執行個體類型設定為 InstanceType 維度的值。
- "AutoScalingGroupName":"${aws:AutoScalingGroupName}" 會將執行個體的 Auto Scaling 群組名稱設定為 AutoScalingGroupName 維度的值。

欲將維度以任意鍵/值對附加至指標中，請針對該類型指標，在此欄位中使用 append_dimensions 參數。

如果您指定的值取決於 Amazon EC2 中繼資料，而且您使用代理，那麼您必須確保伺服器可以存取 Amazon EC2 的端點。如需有關這些端點的詳細資訊，請參閱 Amazon Web Services 一般參考中的 Amazon Elastic Compute Cloud (Amazon EC2)。

- aggregation_dimensions – 選用。指定彙整所收集指標的維度。例如，如果您彙整 AutoScalingGroupName 維度上的指標，來自各 Auto Scaling 群組的所有執行個體的指標將會彙總並可整體檢視。

  您可以彙整一或多個維度的指標。例如，為單一執行個體 ID、單一執行個體類型及兩個維度組合指定 [["InstanceId"], ["InstanceType"], ["InstanceId","InstanceType"]] 彙總指標。

  您也可以指定 [] 將所有指標彙整至一個集合，無視所有維度。

- endpoint_override – 指定 FIPS 端點或私有連結，作為代理程式傳送指標的目標端點。指定此選項和設定私有連結可讓您將指標傳送到 Amazon VPC 端點。如需詳細資訊，請參閱什麼是 Amazon VPC？。

  endpoint_override 的值必須是 URL 字串。

  例如，組態檔案中 metrics 區段的下列部分會將代理程式設定為在傳送指標時使用 VPC 端點。

```
{
  "metrics": {
    "endpoint_override": "vpce-XXXXXXXXXXXXXXXXXXXXXXXX.monitoring.us-
east-1.vpce.amazonaws.com",
```

```
    ......
    },
}
```

- `metrics_collected` – 必要。指定要收集哪些指標，包括要透過 StatsD 或 `collectd` 收集的自訂指標。此區段包含數個子區段。

  `metrics_collected` 區段的內容取決於此組態檔案適用於執行 Linux 或 Windows Server 的伺服器。

- `force_flush_interval` – 指定指標在傳送到伺服器之前停留在記憶體緩衝區內的最長時間 (以秒為單位)。不論此欄位的設定如何，如果緩衝區中的指標大小達到 1 MB 或有 1000 個不同的指標，系統便會立即將指標傳送到伺服器。

  預設值為 60。

- `credentials` – 指定當將指標傳送給不同帳戶時要使用的 IAM 角色。若有指定，這個欄位會有一個 `role_arn` 參數。
  - `role_arn` – 指定當傳送指標給不同帳戶時，用於身分驗證的 IAM 角色的 ARN。如需詳細資訊，請參閱 將指標、日誌和追蹤傳送到不同帳戶。若在此處指定，此值會覆寫組態檔案中 `agent` 區段指定的 `role_arn` (若有的話)。

Linux 區段

在執行 Linux 的伺服器上，組態檔案的 `metrics_collected` 區段也可以包含下列欄位。

其中的許多欄位可包含 `measurement` 區段，列出您想要對該資源收集的指標。這些 `measurement` 區段可以指定完整的指標名稱 (例如 `swap_used`)，或只在部分指標名稱中附加資源的類型。例如，在 `diskio` 的 `measurement` 區段中指定 `reads` 導致收集 `diskio_reads` 指標。

- `collectd` – 選用。指定您想要使用 `collectd` 通訊協定來擷取自訂指標。您可以使用collectd軟體將度量傳送至 CloudWatch 代理程式。如需有關 collectd 可用組態選項的詳細資訊，請參閱 使用 collectd 擷取自訂指標。
- `cpu` – 選用。指定要收集哪些 CPU 指標。此區段僅適用於 Linux 執行個體。對於欲收集的 CPU 指標，您必須納入至少其中一個 `resources` 和 `totalcpu` 欄位。此區段可以包含下列欄位：
  - `drop_original_metrics` – 選用。如果您使用 `metrics` 區段中的 `aggregation_dimensions` 欄位，將指標彙總為彙總結果，則代理程式預設會同時傳送為每個維度值分開的彙總指標和原始指標。如果您不想將原始量度傳送至 CloudWatch，可以使用量度清單來指定此參數。與此參數一起指定的量度沒有依維度報告的量度 CloudWatch。而是只報告彙總指標。這樣可以減少代理程式收集的指標數量，降低您的成本。

- resources - 選用。請以值 * 指定此欄位，以便收集 per-cpu 指標。唯一允許的值為：*。
- totalcpu – 選用。指定是否要報告彙總所有 cpu 核心的 cpu 指標。預設值為 true。
- measurement – 指定要收集的 CPU 指標陣列。可能值為 time_active、time_guest、time_guest_nice、time_idle、time_iowait、time_irq、tim 和 usage_user。如果您包含 cpu，即需要此欄位。

  根據預設，cpu_usage_* 指標的單位是 Percent，cpu_time_* 指標則沒有單位。

  在各個個別指標的項目中，您可以選擇性指定以下一或兩個項目：

  - rename – 為此指標指定不同的名稱。
  - unit – 指定此指標使用的單位，並覆寫指標的 None 預設單位。您指定的單位必須是有效的 CloudWatch 公制單位，如中的Unit說明所示MetricDatum。
- metrics_collection_interval – 選用。指定收集 cpu 指標的頻率，以覆寫組態檔案中 agent 區段指定的全域 metrics_collection_interval。

  此值是以秒數指定。例如，指定每隔 10 秒收集 10 個原因指標，也就是將其設定為每 5 分鐘收集 300 個指定指標。

  如果您將此值設為低於 60 秒，每個指標都將以高解析度指標進行收集。如需高解析度指標的詳細資訊，請參閱 高解析度指標。
- append_dimensions – 選用。其他僅用於 cpu 指標的維度。若您指定此欄位，則除了使用全域 append_dimensions 欄位中指定的維度 (用於代理程式收集的所有類型指標) 之外，也會使用您在此欄位指定的內容。

- disk – 選用。指定要收集哪些磁碟指標。此區段僅適用於 Linux 執行個體。此區段可以包含下列欄位：
  - drop_original_metrics – 選用。如果您使用 metrics 區段中的 aggregation_dimensions 欄位，將指標彙總為彙總結果，則代理程式預設會同時傳送為每個維度值分開的彙總指標和原始指標。如果您不想將原始量度傳送至 CloudWatch，可以使用量度清單來指定此參數。與此參數一起指定的量度沒有依維度報告的量度 CloudWatch。而是只報告彙總指標。這樣可以減少代理程式收集的指標數量，降低您的成本。
  - resources – 選用。指定磁碟掛載點的陣列。此欄位限制 CloudWatch 只能從列出的掛載點收集測量結果。您可以指定 * 作為值，收集來自所有掛載點的指標。預設值為收集來自所有掛載點的指標。

- measurement – 指定要收集的磁碟指標陣列。可能值為 free、total、used、used_percent、inodes_free、inodes_used 及 inodes_total。如果您包含 disk，即需要此欄位。

  > **Note**
  >
  > disk 指標會包含 Partition 的維度，表示所產生的自訂指標數依存於與您執行個體相關聯的分割區數。您擁有的磁碟分割區數量取決於您使用的 AMI，以及連接到伺服器的 Amazon EBS 磁碟區數量。

  若要檢視各個 disk 指標的預設單位，請參閱 CloudWatch代理程式在 Linux 和 macOS 執行個體上收集的指標。

  在各個個別指標的項目中，您可以選擇性指定以下一或兩個項目：

  - rename – 為此指標指定不同的名稱。
  - unit – 指定此指標使用的單位，並覆寫指標的 None 之 None 的預設單位。您指定的單位必須是有效的 CloudWatch 公制單位，如中的Unit說明所示MetricDatum。
- ignore_file_system_types – 指定收集磁碟指標時要排除的檔案系統類型。包括 sysfs、devtmpfs 等的有效值。
- drop_device – 將此設為 true 會導致 Device 未包含在磁碟指標的維度中。

  防止 Device 作為維度使用，對於使用 Nitro 系統的執行個體而言會很有用，因為在這些執行個體上，當執行個體重新啟動時，每個磁碟掛載的裝置名稱都會變更。這可能會導致指標中的資料不一致，並導致警示根據這些指標進入 INSUFFICIENT DATA 狀態。

  預設值為 false。
- metrics_collection_interval – 選用。指定收集磁碟指標的頻率，以覆寫組態檔案中 agent 區段指定的全域 metrics_collection_interval。

  此值是以秒數指定。

  如果您將此值設為低於 60 秒，每個指標都將以高解析度指標進行收集。如需詳細資訊，請參閱 高解析度指標。
- append_dimensions – 選用。其他僅用於磁碟指標的維度。如果您指定此欄位，除了使用 append_dimensions 欄位中指定的維度 (用於代理程式收集的所有類型指標) 之外，也會使用您在此欄位指定的內容。

- diskio – 選用。指定要收集哪些磁碟 i/o 指標。此區段僅適用於 Linux 執行個體。此區段可以包含下列欄位：
  - drop_original_metrics – 選用。如果您使用 metrics 區段中的 aggregation_dimensions 欄位，將指標彙總為彙總結果，則代理程式預設會同時傳送為每個維度值分開的彙總指標和原始指標。如果您不想將原始量度傳送至 CloudWatch，可以使用量度清單來指定此參數。與此參數一起指定的量度沒有依維度報告的量度 CloudWatch。而是只報告彙總指標。這樣可以減少代理程式收集的指標數量，降低您的成本。
  - resources - 選用。如果您指定裝置陣列，則只會從這些裝置 CloudWatch 收集指標。否則，會收集所有裝置的指標。您也可以指定 * 作為值，收集來自所有裝置的指標。
  - measurement – 指定要收集的 diskio 指標陣列。可能值為 reads、writes、read_bytes、write_bytes、read_time、write_time、io_time 及 iops_in_progress。如果您包含 diskio，即需要此欄位。

    在各個個別指標的項目中，您可以選擇性指定以下一或兩個項目：
    - rename – 為此指標指定不同的名稱。
    - unit – 指定此指標使用的單位，並覆寫指標的 None 之 None 的預設單位。您指定的單位必須是有效的 CloudWatch 公制單位，如中的Unit說明所示MetricDatum。
  - metrics_collection_interval – 選用。指定收集 diskio 指標的頻率，以覆寫組態檔案中 agent 區段指定的全域 metrics_collection_interval。

    此值是以秒數指定。

    如果您將此值設為低於 60 秒，每個指標都將以高解析度指標進行收集。如需高解析度指標的詳細資訊，請參閱 高解析度指標。
  - append_dimensions – 選用。其他僅用於 diskio 指標的維度。如果您指定此欄位，除了使用 append_dimensions 欄位中指定的維度 (用於代理程式收集的所有類型指標) 之外，也會使用您在此欄位指定的內容。
- swap – 選用。指定要收集哪些 swap 記憶體指標。此區段僅適用於 Linux 執行個體。此區段可以包含下列欄位：
  - drop_original_metrics – 選用。如果您使用 metrics 區段中的 aggregation_dimensions 欄位，將指標彙總為彙總結果，則代理程式預設會同時傳送為每個維度值分開的彙總指標和原始指標。如果您不想將原始量度傳送至 CloudWatch，可以使用量度清單來指定此參數。與此參數一起指定的量度沒有依維度報告的量度 CloudWatch。而是只報告彙總指標。這樣可以減少代理程式收集的指標數量，降低您的成本。

- measurement – 指定要收集的切換指標陣列。可能值為 free、used 及 used_percent。如果您包含 swap，即需要此欄位。

  若要檢視各個 swap 指標的預設單位，請參閱 CloudWatch代理程式在 Linux 和 macOS 執行個體上收集的指標。

  在各個個別指標的項目中，您可以選擇性指定以下一或兩個項目：

  - rename – 為此指標指定不同的名稱。
  - unit – 指定此指標使用的單位，並覆寫指標的 None 之 None 的預設單位。您指定的單位必須是有效的 CloudWatch 公制單位，如中的Unit說明所示MetricDatum。
- metrics_collection_interval – 選用。指定收集 swap 指標的頻率，以覆寫組態檔案中 agent 區段指定的全域 metrics_collection_interval。

  此值是以秒數指定。

  如果您將此值設為低於 60 秒，每個指標都將以高解析度指標進行收集。如需高解析度指標的詳細資訊，請參閱 高解析度指標。
- append_dimensions – 選用。其他僅用於 swap 指標的維度。如果您指定此欄位，除了使用全域 append_dimensions 欄位中指定的維度 (用於代理程式收集的所有類型指標) 之外，也會使用您在此欄位指定的內容。以高解析度指標收集。

- mem – 選用。指定要收集哪些記憶體指標。此區段僅適用於 Linux 執行個體。此區段可以包含下列欄位：
  - drop_original_metrics – 選用。如果您使用 metrics 區段中的 aggregation_dimensions 欄位，將指標彙總為彙總結果，則代理程式預設會同時傳送為每個維度值分開的彙總指標和原始指標。如果您不想將原始量度傳送至 CloudWatch，可以使用量度清單來指定此參數。與此參數一起指定的量度沒有依維度報告的量度 CloudWatch。而是只報告彙總指標。這樣可以減少代理程式收集的指標數量，降低您的成本。
  - measurement – 指定要收集的記憶體指標陣列。可能值為 active、available、available_percent、buffered、cached、free、inactive、total、u 及 used_percent。如果您包含 mem，即需要此欄位。

    若要檢視各個 mem 指標的預設單位，請參閱 CloudWatch代理程式在 Linux 和 macOS 執行個體上收集的指標。

    在各個個別指標的項目中，您可以選擇性指定以下一或兩個項目：

    - rename – 為此指標指定不同的名稱。

    - unit – 指定此指標使用的單位，並覆寫指標的 None 預設單位。您指定的單位必須是有效的 CloudWatch 公制單位，如中的Unit說明所示MetricDatum。
  - metrics_collection_interval – 選用。指定收集 mem 指標的頻率，以覆寫組態檔案中 agent 區段指定的全域 metrics_collection_interval。

    此值是以秒數指定。

    如果您將此值設為低於 60 秒，每個指標都將以高解析度指標進行收集。如需高解析度指標的詳細資訊，請參閱 高解析度指標。
  - append_dimensions – 選用。其他僅用於 mem 指標的維度。若您指定此欄位，則除了使用 append_dimensions 欄位中指定的維度 (用於代理程式收集的所有類型指標) 之外，也會使用您在此欄位指定的內容。
- net – 選用。指定要收集哪些網路指標。此區段僅適用於 Linux 執行個體。此區段可以包含下列欄位：
  - drop_original_metrics – 選用。如果您使用 metrics 區段中的 aggregation_dimensions 欄位，將指標彙總為彙總結果，則代理程式預設會同時傳送為每個維度值分開的彙總指標和原始指標。如果您不想將原始量度傳送至 CloudWatch，可以使用量度清單來指定此參數。與此參數一起指定的量度沒有依維度報告的量度 CloudWatch。而是只報告彙總指標。這樣可以減少代理程式收集的指標數量，降低您的成本。
  - resources - 選用。如果您指定網路介面陣列，則只會從這些介面 CloudWatch 收集指標。否則，會收集所有裝置的指標。您也可以指定 * 作為值，收集來自所有介面的指標。
  - measurement – 指定要收集的聯網指標陣列。可能值為 bytes_sent、bytes_recv、drop_in、drop_out、err_in、err_out、packets_sent 及 packets_recv。如果您包含 net，即需要此欄位。

    若要檢視各個 net 指標的預設單位，請參閱 CloudWatch代理程式在 Linux 和 macOS 執行個體上收集的指標。

    在各個個別指標的項目中，您可以選擇性指定以下一或兩個項目：
    - rename – 為此指標指定不同的名稱。
    - unit – 指定此指標使用的單位，並覆寫指標的 None 預設單位。您指定的單位必須是有效的 CloudWatch 公制單位，如中的Unit說明所示MetricDatum。
  - metrics_collection_interval – 選用。指定收集 net 指標的頻率，以覆寫組態檔案中 agent 區段指定的全域 metrics_collection_interval。

此值是以秒數指定。例如，指定每隔 10 秒收集 10 個原因指標，也就是將其設定為每 5 分鐘收集 300 個指定指標。

如果您將此值設為低於 60 秒，每個指標都將以高解析度指標進行收集。如需高解析度指標的詳細資訊，請參閱 高解析度指標。

- append_dimensions – 選用。其他僅用於 net 指標的維度。如果您指定此欄位，則除了使用 append_dimensions 欄位中指定的維度 (用於代理程式收集的所有類型指標) 之外，也會使用您在此欄位指定的內容。

- netstat – 選用。指定要收集 TCP 連線狀態和 UDP 連接指標。此區段僅適用於 Linux 執行個體。此區段可以包含下列欄位：
  - drop_original_metrics – 選用。如果您使用 metrics 區段中的 aggregation_dimensions 欄位，將指標彙總為彙總結果，則代理程式預設會同時傳送為每個維度值分開的彙總指標和原始指標。如果您不想將原始量度傳送至 CloudWatch，可以使用量度清單來指定此參數。與此參數一起指定的量度沒有依維度報告的量度 CloudWatch。而是只報告彙總指標。這樣可以減少代理程式收集的指標數量，降低您的成本。
  - measurement – 指定要收集的 netstat 指標陣列。可能值為 tcp_close、tcp_close_wait、tcp_closing、tcp_established、tcp_fin_wait1、tcp_fi 及 udp_socket。如果您包含 netstat，即需要此欄位。

    若要檢視各個 netstat 指標的預設單位，請參閱 CloudWatch代理程式在 Linux 和 macOS 執行個體上收集的指標。

    在各個個別指標的項目中，您可以選擇性指定以下一或兩個項目：
    - rename – 為此指標指定不同的名稱。
    - unit – 指定此指標使用的單位，並覆寫指標的 None 預設單位。您指定的單位必須是有效的 CloudWatch 公制單位，如中的Unit說明所示MetricDatum。
  - metrics_collection_interval – 選用。指定收集 netstat 指標的頻率，以覆寫組態檔案中 agent 區段指定的全域 metrics_collection_interval。

    此值是以秒數指定。

    如果您將此值設為低於 60 秒，每個指標都將以高解析度指標進行收集。如需高解析度指標的詳細資訊，請參閱 高解析度指標。
  - append_dimensions – 選用。其他僅用於 netstat 指標的維度。如果您指定此欄位，則除了使用 append_dimensions 欄位中指定的維度 (用於代理程式收集的所有類型指標) 之外，也會使用您在此欄位指定的內容。

- processes – 選用。指定要收集哪些程序指標。此區段僅適用於 Linux 執行個體。此區段可以包含下列欄位：
  - drop_original_metrics – 選用。如果您使用 metrics 區段中的 aggregation_dimensions 欄位，將指標彙總為彙總結果，則代理程式預設會同時傳送為每個維度值分開的彙總指標和原始指標。如果您不想將原始量度傳送至 CloudWatch，可以使用量度清單來指定此參數。與此參數一起指定的量度沒有依維度報告的量度 CloudWatch。而是只報告彙總指標。這樣可以減少代理程式收集的指標數量，降低您的成本。
  - measurement – 指定要收集的程序指標陣列。可能值為 blocked、dead、idle、paging、running、sleeping、stopped、total、total_threads、w 及 zombies。如果您包含 processes，即需要此欄位。

    所有 processes 指標的預設單位皆為 None。

    在各個個別指標的項目中，您可以選擇性指定以下一或兩個項目：
    - rename – 為此指標指定不同的名稱。
    - unit – 指定此指標使用的單位，並覆寫指標的 None 預設單位。您指定的單位必須是有效的 CloudWatch 公制單位，如中的Unit說明所示MetricDatum。
  - metrics_collection_interval – 選用。指定收集程序指標的頻率，以覆寫組態檔案中 agent 區段指定的全域 metrics_collection_interval。

    此值是以秒數指定。例如，指定每隔 10 秒收集 10 個原因指標，也就是將其設定為每 5 分鐘收集 300 個指定指標。

    如果您將此值設為低於 60 秒，每個指標都將以高解析度指標進行收集。如需詳細資訊，請參閱 高解析度指標。
  - append_dimensions – 選用。其他僅用於程序指標的維度。如果您指定此欄位，則除了使用 append_dimensions 欄位中指定的維度 (用於代理程式收集的所有類型指標) 之外，也會使用您在此欄位指定的內容。
- nvidia_gpu – 選用。指定要收集哪些 NVIDIA GPU 指標。此區段僅適用於設定了 NVIDIA GPU 加速器並安裝了 NVIDIA 系統管理介面 (nvidia-smi) 之主機上的 Linux 執行個體。

  收集的 NVIDIA GPU 指標字首是字串 nvidia_smi_，可用來區分其與為其他加速器類型收集的指標。此區段可以包含下列欄位：
  - drop_original_metrics – 選用。如果您使用 metrics 區段中的 aggregation_dimensions 欄位，將指標彙總為彙總結果，則代理程式預設會同時傳送為每個維度值分開的彙總指標和原始指標。如果您不想將原始量度傳送至 CloudWatch，可以使用量度清

單來指定此參數。與此參數一起指定的量度沒有依維度報告的量度 CloudWatch。而是只報告彙總指標。這樣可以減少代理程式收集的指標數量，降低您的成本。

- `measurement` – 指定要收集的 NVIDIA GPU 指標陣列。如需此處要使用之可能值的清單，請參閱 收集 NVIDIA GPU 指標 的資料表中的 Metric (指標) 資料欄。

  在每個指標的項目中，您可以選擇性指定以下一個或兩個項目：

  - `rename` – 為此指標指定不同的名稱。
  - `unit` – 指定此指標使用的單位，並覆寫指標的 None 預設單位。您指定的單位必須是有效的 CloudWatch 公制單位，如中的`Unit`說明所示MetricDatum。
- `metrics_collection_interval` - 選用。指定 NVIDIA GPU 指標的收集頻率，並覆寫組態檔案的 `agent` 區段中指定的全域 `metrics_collection_interval`。

- `procstat` – 選用。指定您想要從個別的程序擷取指標。如需有關 procstat 可用組態選項的詳細資訊，請參閱 使用 procstat 外掛程式收集程序指標。
- `statsd` – 選用。指定您想要使用 StatsD 通訊協定來擷取自訂指標。 CloudWatch 代理程式充當通訊協定的常駐程式。您可以使用任何標準用`StatsD`戶端將度量傳送至 CloudWatch 代理程式。如需有關 StatsD 可用組態選項的詳細資訊，請參閱 使用 StatsD 擷取自訂指標 。
- `ethtool` – 選用。指定您想要使用 `ethtool` 外掛程式來擷取網路指標。此外掛程式可以匯入標準 ethtool 公用程式收集的指標，以及來自 Amazon EC2 執行個體的網路效能指標。如需有關 ethtool 可用組態選項的詳細資訊，請參閱 收集網路效能指標。

以下是適用於 Linux 伺服器的 `metrics` 區段的範例。在此範例中，會收集三個 CPU 指標、三個 netstat 指標、三個程序指標以及一個磁碟指標，且會將代理程式設定為從 `collectd` 用戶端接收其他指標。

```
"metrics": {
    "aggregation_dimensions" : [["AutoScalingGroupName"], ["InstanceId",
 "InstanceType"],[]],
    "metrics_collected": {
      "collectd": {},
      "cpu": {
        "resources": [
          "*"
        ],
        "measurement": [
          {"name": "cpu_usage_idle", "rename": "CPU_USAGE_IDLE", "unit": "Percent"},
          {"name": "cpu_usage_nice", "unit": "Percent"},
          "cpu_usage_guest"
```

```
        ],
        "totalcpu": false,
        "drop_original_metrics": [ "cpu_usage_guest" ],
        "metrics_collection_interval": 10,
        "append_dimensions": {
          "test": "test1",
          "date": "2017-10-01"
        }
      },
      "netstat": {
        "measurement": [
          "tcp_established",
          "tcp_syn_sent",
          "tcp_close"
        ],
        "metrics_collection_interval": 60
      },
       "disk": {
        "measurement": [
          "used_percent"
        ],
        "resources": [
          "*"
        ],
        "drop_device": true
      },
      "processes": {
        "measurement": [
          "running",
          "sleeping",
          "dead"
        ]
      }
    },
    "append_dimensions": {
      "ImageId": "${aws:ImageId}",
      "InstanceId": "${aws:InstanceId}",
      "InstanceType": "${aws:InstanceType}",
      "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
    }
  }
```

Windows Server

在 Windows Server 的 metrics_collected 區段中，您可以擁有各個 Windows 效能物件的子區段，例如 Memory、Processor 及 LogicalDisk。如需有關哪些物件和計數器可供使用的資訊，請參閱 Microsoft Windows 文件中的「效能計數器」。

在各物件的子區段中，您指定要收集的 measurement 計數器陣列。您在組態檔案中指定的每個物件都需要 measurement 陣列。您也可以指定 resources 欄位來命名要收集指標的執行個體。您也可以為 resources 指定 *，為每個執行個體收集單獨的指標。如果您省略具有執行個體之計數器的 resources，所有執行個體的資料將彙總為一組資料。如果您忽略resources沒有執行個體的計數器， CloudWatch 代理程式不會收集計數器。若要判斷計數器是否具有執行個體，您可以使用下列其中一個命令。

Powershell：

```
Get-Counter -ListSet *
```

命令列 (非 Powershell)：

```
TypePerf.exe -q
```

在每個物件區段，您也可以指定以下可選的欄位：

- metrics_collection_interval – 選用。指定為此物件收集指標的頻率，以覆寫組態檔案中 agent 區段指定的全域 metrics_collection_interval。

  此值是以秒數指定。例如，指定每隔 10 秒收集 10 個原因指標，也就是將其設定為每 5 分鐘收集 300 個指定指標。

  如果您將此值設為低於 60 秒，每個指標都將以高解析度指標進行收集。如需詳細資訊，請參閱 高解析度指標。
- append_dimensions – 選用。指定僅用於此物件指標的其他維度。如果您指定此欄位，則除了使用全域 append_dimensions 欄位中指定的維度 (用於代理程式收集的所有類型指標) 之外，也會使用您在此欄位指定的內容。
- drop_original_metrics - 選用。如果您使用 metrics 區段中的 aggregation_dimensions 欄位，將指標彙總為彙總結果，則代理程式預設會同時傳送為每個維度值分開的彙總指標和原始指標。如果您不想將原始量度傳送至 CloudWatch，可以使用量度清單來指定此參數。與此參數一起指定的量度沒有依維度報告的量度 CloudWatch。而是只報告彙總指標。這樣可以減少代理程式收集的指標數量，降低您的成本。

在每個計數器區段，您也可以指定以下可選的欄位：

- `rename`— 指定此測量結果要使用 CloudWatch 的不同名稱。
- `unit` – 指定用於此指標的單位。您指定的單位必須是有效的 CloudWatch 公制單位，如中的`Unit`說明所示MetricDatum。

`metrics_collected` 中還可以包含其他兩個選用區段：

- `statsd` – 讓您使用 StatsD 通訊協定擷取自訂指標。 CloudWatch 代理程式充當通訊協定的常駐程式。您可以使用任何標準用`StatsD`戶端將度量傳送至 CloudWatch代理程式。如需詳細資訊，請參閱 使用 StatsD 擷取自訂指標 。
- `procstat` – 讓您可以從個別程序擷取指標。如需詳細資訊，請參閱 使用 procstat 外掛程式收集程序指標。

以下是適用於 Windows Server 的 `metrics` 區段範例。在此範例中，會收集許多 Windows 指標，且也會將電腦設為從 `StatsD` 用戶端接收其他指標。

```
"metrics": {
    "metrics_collected": {
      "statsd": {},
      "Processor": {
        "measurement": [
          {"name": "% Idle Time", "rename": "CPU_IDLE", "unit": "Percent"},
          "% Interrupt Time",
          "% User Time",
          "% Processor Time"
        ],
        "resources": [
          "*"
        ],
        "append_dimensions": {
          "d1": "win_foo",
          "d2": "win_bar"
        }
      },
      "LogicalDisk": {
        "measurement": [
          {"name": "% Idle Time", "unit": "Percent"},
          {"name": "% Disk Read Time", "rename": "DISK_READ"},
          "% Disk Write Time"
```

```
        ],
        "resources": [
          "*"
        ]
      },
      "Memory": {
        "metrics_collection_interval": 5,
        "measurement": [
          "Available Bytes",
          "Cache Faults/sec",
          "Page Faults/sec",
          "Pages/sec"
        ],
        "append_dimensions": {
          "d3": "win_bo"
        }
      },
      "Network Interface": {
        "metrics_collection_interval": 5,
        "measurement": [
          "Bytes Received/sec",
          "Bytes Sent/sec",
          "Packets Received/sec",
          "Packets Sent/sec"
        ],
        "resources": [
          "*"
        ],
        "append_dimensions": {
          "d3": "win_bo"
        }
      },
      "System": {
        "measurement": [
          "Context Switches/sec",
          "System Calls/sec",
          "Processor Queue Length"
        ],
        "append_dimensions": {
          "d1": "win_foo",
          "d2": "win_bar"
        }
      }
    },
```

```
    "append_dimensions": {
      "ImageId": "${aws:ImageId}",
      "InstanceId": "${aws:InstanceId}",
      "InstanceType": "${aws:InstanceType}",
      "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
    },
    "aggregation_dimensions" : [["ImageId"], ["InstanceId", "InstanceType"], ["d1"],[]]
    }
  }
```

CloudWatch 代理程式組態檔：記錄檔段落

`logs` 區段可包含下列欄位：

- `logs_collected` – 若包含 `logs` 區段，則為必要項目。指定要從伺服器收集哪些日誌檔和 Windows 事件日誌。它可以包含兩個欄位，`files` 及 `windows_events`。
  - `files`— 指定 CloudWatch 代理程式要收集的一般記錄檔。它包含一個欄位 `collect_list`，可進一步定義這些檔案。
    - `collect_list` – 若包含 `files`，則為必要項目。包含項目陣列，各項目指定一個要收集的日誌。每個項目可以包含下列欄位：
      - `file_path`— 指定要上傳至防護記錄的 CloudWatch 記錄檔路徑。接受標準 Unix glob 匹配規則，以及將 ** 作為超級星號。例如，指定 `/var/log/**.log` 以收集 `/var/log` 樹狀目錄中的所有 `.log` 檔案。如需更多範例，請參閱 Glob Library。

        您也可以使用標準星號作為標準萬用字元。例如，`/var/log/system.log*` 符合檔案 (如 `/var/log` 中的 `system.log_1111`、`system.log_2222` 等等)。

        只有最新的檔案會根據檔案修改時間推送至 CloudWatch 記錄檔。我們建議您使用萬用字元來指定一系列的相同類型的檔案，例如 `access_log.2018-06-01-01` 和 `access_log.2018-06-01-02`，而不是多種的檔案，例如 `access_log_80` 和 `access_log_443`。若要指定多種種類的檔案，可將另一個日誌串流項目新增至代理程式組態檔案，讓每個種類的日誌檔案進入不同的日誌串流。
      - `auto_removal` – 選用。如果是這樣`true`， CloudWatch 代理程式會在讀取此記錄檔後自動刪除該記錄檔，並且已輪替。通常記錄檔會在全部內容上傳至 CloudWatch 記錄檔之後刪除，但是如果代理程式到達 EOF (檔案結尾)，並偵測到另一個符合相同檔案的較新記錄檔`file_path`，則代理程式會刪除舊檔案，因此您必須先確定已完成寫入 OLD 檔案，然後再建立新檔案。RUST 追蹤程式庫存在已知的不相容性，因為它可能會建立新的記錄檔，然後仍會嘗試寫入 OLD 記錄檔。

代理程式只會從建立多個檔案的日誌檔案中移除完整檔案，例如為每個日期建立不同檔案的日誌檔案。如果日誌檔案持續寫入單一檔案，則不會移除該日誌檔案。

如果您已經有記錄檔案輪換或移除方法，建議您省略此欄位或將其設定為 `false`。

如果您省略此欄位，則會使用 `false` 的預設值。

- `log_group_name` – 選用。指定要用作 CloudWatch 記錄檔中記錄群組名稱的項目。

  建議您使用此欄位來指定日誌群組名稱以防止混淆。若省略 `log_group_name`，則會使用 `file_path` 值 (包含最後的點) 作為日誌群組名稱。例如，如果檔案路徑是 `/tmp/TestLogFile.log.2017-07-11-14`，日誌群組名稱即為 `/tmp/TestLogFile.log`。

  如果指定日誌群組名稱，您可以使用 `{instance_id}`、`{hostname}`、`{local_hostname}` 和 `{ip_address}` 作為名稱中的變數。`{hostname}` 會從 EC2 中繼資料擷取主機名稱，而 `{local_hostname}` 會使用網路組態檔案中的主機名稱。

  若您使用這些變數來建立多個不同的日誌群組，請注意每個區域每個帳戶的上限為 1,000,000 個日誌群組。

  可用的字元為 a–z、A–Z、0–9、'_' (底線)、'-' (連字號)、'/' (正斜線) 和 '.' (句點)。

- `log_group_class` - 選用。指定哪些日誌群組類別要用於新日誌群組。如需有關日誌群組類別的詳細資訊，請參閱日誌類別。

  有效值為 `STANDARD` 和 `INFREQUENT_ACCESS`。如果您省略此欄位，預設為使用 `STANDARD`。

  > **Important**
  >
  > 建立日誌群組後，其類別即無法變更。

- `log_stream_name` - 選用。指定在記錄檔中用作記 CloudWatch 錄資料流名稱的項目。作為名稱的一部分，您可以使用 `{instance_id}`、`{hostname}`、`{local_hostname}` 和 `{ip_address}` 作為名稱中的變數。`{hostname}` 會從 EC2 中繼資料擷取主機名稱，而 `{local_hostname}` 會使用網路組態檔案中的主機名稱。

  如果省略此欄位，則會使用全域 `logs` 區段中的 `log_stream_name` 參數值。如果該數值也省略，則會使用 `{instance_id}` 的預設值。

若日誌串流尚未存在，則會自動建立。

- `retention_in_days` – 選用。指定在指定日誌群組中保留日誌事件的天數。
  - 如果代理程式目前正在建立日誌群組，且您忽略此欄位，則此新日誌群組保留期將設為永不過期。
  - 如果此日誌群組已存在，並且您指定此欄位，則會使用您指定的新保留期。如果您因為日誌群組已存在而忽略此欄位，則日誌群組的保留期不會變更。

    當 CloudWatch 代理程式精靈用`-1`來建立代理程式組態檔，而您未指定記錄保留值時，代理程式精靈會使用此欄位的預設值。此精靈設定的`-1`值指定記錄群組中的事件永遠不會過期。然而，手動將此值編輯為 `-1` 並不會有任何作用。

  有效值為：
  1、3、5、7、14、30、60、90、120、150、180、365、400、545、731、1827、2192、2557、29
  和 3653。

  如果將代理程式設定為將多個日誌串流寫入同一日誌群組，則在某個位置指定 `retention_in_days` 將為整個日誌群組設定日誌保留期。如果您為多個位置的同一日誌群組指定 `retention_in_days`，則如果所有這些值都相等，則會設定保留期。但是，如果在多個位置為同一日誌群組指定不同的 `retention_in_days` 值，則不會設定日誌保留期，且代理程式將停止並傳回錯誤。

  > **Note**
  >
  > 代理程式的 IAM 角色或 IAM 使用者必須具有 `logs:PutRetentionPolicy`，以便其能夠設定保留政策。如需詳細資訊，請參閱 [允許 CloudWatch 代理程式設定記錄保留原則](#)。

  > **Warning**
  >
  > 如果您為已存在的日誌群組設定 `retention_in_days`，則該日誌群組中在您指定的天數之前發佈的所有日誌都將遭到刪除。例如，將其設定為 3 會導致超過 3 天之前的所有日誌遭到刪除。

- `filters` - 選用。可以包含項目陣列，每個項目都指定正則表達式和篩選條件類型，以指定是發佈還是捨棄與篩選條件相符的日誌項目。如果您省略此欄位，記錄檔中的所有記錄都會發佈至 CloudWatch 記錄檔。如果您包含此欄位，則代理程式會使用您指定的所有篩選器處理每

個記錄檔訊息，並且只會將通過所有篩選器的記錄事件發佈至 CloudWatch 記錄檔。未通過所有篩選器的記錄項目仍會保留在主機的記錄檔中，但不會傳送至 CloudWatch 記錄檔。

篩選條件陣列中的每個項目可以包含下列欄位：

- `type` – 表示篩選條件的類型。有效值為 `include` 和 `exclude`。使用時`include`，記錄項目必須符合要發行至 CloudWatch 記錄的運算式。使用時`exclude`，符合篩選器的每個記錄項目不會傳送至 CloudWatch 記錄檔。
- `expression` – 遵循 RE2 語法的正則表達式字串。

> **Note**
>
> CloudWatch 代理程式不會檢查您提供的任何規則運算式的效能，也不會限制規則運算式評估的執行時間。我們建議您小心不要編寫評估代價過高的表達式。如需可能問題的詳細資訊，請參閱規則運算式拒絕服務- ReDo S

例如，下列 CloudWatch 代理程式設定檔摘錄會將 PUT 和 POST 要求的記錄檔發佈到 CloudWatch 記錄檔，但不包括來自 Firefox 的記錄檔。

```
"collect_list": [
  {
    "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/test.log",
    "log_group_name": "test.log",
    "log_stream_name": "test.log",
    "filters": [
      {
        "type": "exclude",
        "expression": "Firefox"
      },
      {
        "type": "include",
        "expression": "P(UT|OST)"
      }
    ]
  },
  .....
]
```

> **Note**
>
> 組態檔案中篩選條件的順序對效能來說很重要。在上述範例中，代理程式將捨棄與 Firefox 相符的所有日誌，然後才開始評估第二個篩選條件。若要由多個篩選條件評估較少個日誌項目，請將您希望排除更多日誌的篩選條件放置在組態檔案中的第一位。

- timezone – 選用。指定將時間戳記放在日誌事件時要使用的時區。有效值為 UTC 和 Local。預設值為 Local。

  如果您未指定 timestamp_format 的值，則此參數會被忽略。

- timestamp_format – 選用。使用純文字和以 % 開頭的特殊符號，指定時間戳記格式。如果您省略此欄位，將使用目前時間。如果您使用此欄位，您可以使用以下清單中的符號作為格式的一部分，

  如果單一日誌項目包含兩個符合格式的時間戳記，則會使用第一個時間戳記。

  這個符號清單與舊版 CloudWatch Logs 代理程式所使用的清單不同。如需這些差異的摘要，請參閱 統一 CloudWatch 代理程式與舊版記錄代理程式之間的時間戳 CloudWatch 記差。

  %y

  年，不包含世紀的填充零的十進位數字。例如，19 表示 2019。

  %Y

  年，包含世紀的十進位數字。例如 2019。

  %b

  月，當地的縮寫名稱

  %B

  月，當地的完整名稱

  %m

  月，填充零的十進位數字

  %-m

  月，十進位數字 (非填充零)

%d

日，填充零的十進位數字

%-d

日，十進位數字 (非填充零)

%A

完整的週間日名稱，例如 Monday

%a

週間日名稱的縮寫，例如 Mon

%H

小時 (24 小時制)，填充零的十進位數字

%I

小時 (12 小時制)，填充零的十進位數字

%-I

小時 (12 小時制)，十進位數字 (非填充零)

%p

AM 或 PM

%M

分鐘，填充零的十進位數字

%-M

分鐘，十進位數字 (非填充零)

%S

秒鐘，填充零的十進位數字

%-S

秒鐘，十進位數字 (非填充零)

%f

分數秒為十進位數字 (1-9 的數字)，在左側填補零。

%Z

時區，例如 PST

%z

時區，以本地時間與 UTC 之間的偏移表示。例如 -0700。只有此格式受支援。例如，-07:00 不是有效的格式。

- multi_line_start_pattern – 指定用於識別日誌訊息開始處的模式。日誌訊息是由符合模式的一列及不符合模式的任何後續幾列所組成。

  如果您省略此欄位，會停用多行模式，而開頭使用非空白字元的任何行皆可結束之前的日誌訊息，並開始新的日誌訊息。

  如果您包含此欄位，即可指定 {timestamp_format} 使用與您的時間戳記格式相同的規則表達式。否則，您可以為 CloudWatch Logs 指定不同的規則運算式，以用來決定多行項目的起始行。

- encoding – 指定日誌檔案的編碼，以便正確讀取檔案。如果您指定不正確的編碼，可能導致資料遺失，因為無法解碼的字元會被替換為其他的字元。

  預設值為 utf-8。以下是所有的可能值：

  ascii, big5, euc-jp, euc-kr, gbk, gb18030, ibm866, iso2022-jp, iso8859-2, iso8859-3, iso8859-4, iso8859-5, iso8859-6, iso8859-7, iso8859-8, iso8859-8-i, iso8859-10, iso8859-13, iso8859-14, iso8859-15, iso8859-16, koi8-r, koi8-u, macintosh, shift_jis, utf-8, utf-16, utf-16le, UTF-16, UTF-16LE, windows-874, windows-1250, windows-1251, windows-1252, windows-1253, windows-1254, windows-1255, windows-1256, windows-1257, windows-1258, x-mac-cyrillic

- windows_events 區段會指定要從執行 Windows Server 的伺服器收集的 Windows 事件類型。它包括以下欄位：
  - collect_list – 若包含 windows_events，則為必要項目。指定要收集的 Windows 事件類型和層級。每個要收集的記錄在此區段都有一個項目，其中可包含下列欄位：

- `event_name` – 指定要記錄的 Windows 事件類型。這和 Windows 事件記錄的通道名稱相同：例如，`System`、`Security`、`Application` 等。要記錄的每個 Windows 事件類型都需要此欄位。

  > **Note**
  >
  > 從 Windows 記錄通道 CloudWatch 擷取訊息時，會根據其`Full Name`內容查詢記錄通道。同時，Windows 事件檢視器導覽窗格會顯示日誌頻道的 `Log Name` 屬性。`Full Name` 和 `Log Name` 並不總是相符。若要確認頻道的 `Full Name`，在 Windows 事件檢視器中按一下滑鼠右鍵，然後開啟 Properties (屬性)。

- `event_levels` – 指定要記錄日誌的事件層級。您必須指定要記錄的每個層級。可能的值包括 `INFORMATION`、`WARNING`、`ERROR`、`CRITICAL` 及 `VERBOSE`。要記錄的每個 Windows 事件類型都需要此欄位。
- `log_group_name` – 必要。指定要用作 CloudWatch 記錄檔中記錄群組名稱的項目。
- `log_stream_name` - 選用。指定在記錄檔中用作記 CloudWatch 錄資料流名稱的項目。作為名稱的一部分，您可以使用 `{instance_id}`、`{hostname}`、`{local_hostname}` 和 `{ip_address}` 作為名稱中的變數。`{hostname}` 會從 EC2 中繼資料擷取主機名稱，而 `{local_hostname}` 會使用網路組態檔案中的主機名稱。

  如果省略此欄位，則會使用全域 `logs` 區段中的 `log_stream_name` 參數值。如果該數值也省略，則會使用 `{instance_id}` 的預設值。

  若日誌串流尚未存在，則會自動建立。
- `event_format` – 選用。指定在 CloudWatch 記錄檔中儲存 Windows 事件時要使用的格式。`xml`使用 XML 格式，就像在視窗事件檢視器中一樣。 `text`使用舊版 CloudWatch Logs 代理程式格式。
- `retention_in_days` - 選用。指定在指定日誌群組中保留 Windows 事件的天數。
  - 如果代理程式目前正在建立日誌群組，且您忽略此欄位，則此新日誌群組保留期將設為永不過期。
  - 如果此日誌群組已存在，並且您指定此欄位，則會使用您指定的新保留期。如果您因為日誌群組已存在而忽略此欄位，則日誌群組的保留期不會變更。

    當 CloudWatch 代理程式精靈用`-1`來建立代理程式組態檔，而您未指定記錄保留值時，代理程式精靈會使用此欄位的預設值。這個由精靈設定的 `-1` 值會指定日誌群組中的事件不會過期。然而，手動將此值編輯為 `-1` 並不會有任何作用。

有效值為：
1、3、5、7、14、30、60、90、120、150、180、365、400、545、731、1827、2192、2557、29
和 3653。

如果將代理程式設定為將多個日誌串流寫入同一日誌群組，則在某個位置指定 `retention_in_days` 將為整個日誌群組設定日誌保留期。如果您為多個位置的同一日誌群組指定 `retention_in_days`，則如果所有這些值都相等，則會設定保留期。但是，如果在多個位置為同一日誌群組指定不同的 `retention_in_days` 值，則不會設定日誌保留期，且代理程式將停止並傳回錯誤。

> **Note**
>
> 代理程式的 IAM 角色或 IAM 使用者必須具有 `logs:PutRetentionPolicy`，以便其能夠設定保留政策。如需詳細資訊，請參閱 允許 CloudWatch 代理程式設定記錄保留原則。

> **Warning**
>
> 如果您為已存在的日誌群組設定 `retention_in_days`，則該日誌群組中在您指定的天數之前發佈的所有日誌都將遭到刪除。例如，將其設定為 3 會導致超過 3 天之前的所有日誌遭到刪除。

- `log_stream_name` – 必要。指定預設的日誌串流名稱，以用於在 `collect_list` 項目內的 `log_stream_name` 參數中沒有定義個別日誌串流名稱的任何日誌或 Windows 事件。
- `endpoint_override` – 指定 FIPS 端點或私有連結，作為代理程式傳送日誌的目標端點。指定此欄位和設定私有連結，可讓您將日誌傳送到 Amazon VPC 端點。如需詳細資訊，請參閱什麼是 Amazon VPC？。

  `endpoint_override` 的值必須是 URL 字串。

  例如，組態檔案中 logs 區段的下列部分會將代理程式設定為在傳送日誌時使用 VPC 端點。

```
{
  "logs": {
    "endpoint_override": "vpce-XXXXXXXXXXXXXXXXXXXXXXXX.logs.us-
east-1.vpce.amazonaws.com",
    ......
```

```
    },
}
```

- force_flush_interval – 指定日誌在傳送到伺服器之前停留在記憶體緩衝區內的最長時間 (以秒為單位)。不論此欄位的設定如何，如果緩衝區中的日誌大小達到 1 MB，系統會立即將日誌傳送到伺服器。預設值為 5。

  如果您使用代理程式以內嵌指標格式來報告高解析度指標，並在這些指標上設定警示，請保留此參數的預設值 5 設定。否則，系統在報告指標時會產生延遲，從而導致對部分或不完整的資料發出警示。

- credentials— 指定將記錄傳送至其他 AWS 帳戶時要使用的 IAM 角色。若有指定，這個欄位會有一個 role_arn 參數。
  - role_arn— 指定將記錄傳送至其他 AWS 帳戶時用於驗證的 IAM 角色 ARN。如需詳細資訊，請參閱 將指標、日誌和追蹤傳送到不同帳戶。此處若有指定，會覆寫組態檔案中 agent 區段指定的 role_arn (如果有)。
- metrics_collected— 此欄位可包含指定代理 CloudWatch 程式收集日誌以啟用應用程式訊號和容器見解等使用案例，並增強 Amazon EKS 的可觀察性。
  - app_signals(選擇性) 指定您要啟用CloudWatch 應用程式訊號如需此組態的詳細資訊，請參閱啟用 CloudWatch 應用程式信。
  - kubernetes – 此欄位可包含 enhanced_container_insights 參數，您可使用該參數啟用 Container Insights 搭配 Amazon EKS 的增強可觀測性。
    - enhanced_container_insights – 將此項設定為 true，以啟用 Container Insights 搭配 Amazon EKS 的增強可觀測性。如需詳細資訊，請參閱 Container Insights 搭配 Amazon EKS 的增強可觀測性。
    - accelerated_compute_metrics— 將此項設定false為選擇不在 Amazon EKS 叢集上收集 Nvidia GPU 指標。如需詳細資訊，請參閱 GPU 指標。
  - emf — 若要收集日誌中內嵌的指標，已不再需要新增此 emf 欄位。這是一個舊版欄位，指定要收集內嵌指標格式之日誌的代理程式。您可以從這些記錄產生測量結果資料。如需詳細資訊，請參閱在日誌中內嵌指標。

以下是 logs 區段的範例。

```
"logs":
   {
       "logs_collected": {
           "files": {
```

```
                "collect_list": [
                    {
                        "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\
\Logs\\amazon-cloudwatch-agent.log",
                        "log_group_name": "amazon-cloudwatch-agent.log",
                        "log_stream_name": "my_log_stream_name_1",
                        "timestamp_format": "%H: %M: %S%y%b%-d"
                    },
                    {
                        "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\
\Logs\\test.log",
                        "log_group_name": "test.log",
                        "log_stream_name": "my_log_stream_name_2"
                    }
                ]
            },
            "windows_events": {
                "collect_list": [
                    {
                        "event_name": "System",
                        "event_levels": [
                            "INFORMATION",
                            "ERROR"
                        ],
                        "log_group_name": "System",
                        "log_stream_name": "System"
                    },
                    {
                        "event_name": "CustomizedName",
                        "event_levels": [
                            "INFORMATION",
                            "ERROR"
                        ],
                        "log_group_name": "CustomizedLogGroup",
                        "log_stream_name": "CustomizedLogStream"
                    }
                ]
            }
        },
        "log_stream_name": "my_log_stream_name",
        "metrics_collected": {
            "kubernetes": {
                 "enhanced_container_insights": true
       }
```

```
    }
  }
```

CloudWatch 代理程式組態檔：追蹤區段

透過將traces區段新增至 CloudWatch 代理程式組態檔案，您可以啟用 CloudWatch 應用程式訊號，或從 X-Ray 和 OpenTelemetry 儀器 SDK 收集追蹤，然後將它們傳送至 X-Ray。

> **Important**
>
> 客服人員的 IAM 角色或 IAM 使用者必須具有將追蹤資料傳送至 X-Ray 的AWSXrayWriteOnlyAccess政策。如需詳細資訊，請參閱 建立 IAM 角色和使用者以搭配 CloudWatch 代理程式使用。

如需收集追蹤的快速入門，您可以只將下列項目新增至 CloudWatch 代理程式組態檔。

```
"traces_collected": {
        "xray": {
        },
        "otlp": {
        }
      }
```

如果您將上一個區段新增至 CloudWatch 代理程式組態檔並重新啟動代理程式，這會導致代理程式開始使用下列預設選項和值收集追蹤。如需有關這些參數的更多資訊，請參閱本節後面的參數定義。

```
"traces_collected": {
        "xray": {
            "bind_address": "127.0.0.1:2000",
            "tcp_proxy": {
              "bind_address": "127.0.0.1:2000"
            }
        },
        "otlp": {
            "grpc_endpoint": "127.0.0.1:4317",
            "http_endpoint": "127.0.0.1:4318"
        }
      }
```

traces 區段可以包含下列欄位：

- `traces_collected` – 若包含 `traces` 區段，則為必要項目。指定要從哪些 SDK 中收集追蹤。它可以包含下列欄位：
  - `app_signals` - 選用。指定您要啟用CloudWatch 應用程式訊號如需有關此組態的更多資訊，請參閱啟用 CloudWatch 應用程式信。
  - `xray` - 選用。指定您要從 X-Ray SDK 中收集追蹤。此區段可以包含下列欄位：
    - `bind_address` – 選用。指定 CloudWatch 代理程式用來偵聽 X-Ray 追蹤的 UDP 位址。格式是 `ip:port`。此地址必須與 X-Ray SDK 中設定的地址相符。

      如果您省略此欄位，預設為使用 `127.0.0.1:2000`。
    - `tcp_proxy` - 選用。設定用於支援 X-Ray 遠端取樣的代理地址。如需詳細資訊，請參閱 X-Ray 文件中的設定取樣規則。

      此區段可以包含下列欄位。
      - `bind_address` - 選用。指定 CloudWatch 代理程式應設定 Proxy 的 TCP 位址。格式是 `ip:port`。此地址必須與 X-Ray SDK 中設定的地址相符。

        如果您省略此欄位，預設為使用 `127.0.0.1:2000`。
  - `otlp` - 選用。指定您要從 OpenTelemetry SDK 收集追蹤。有關 AWS 發行版的更多信息 OpenTelemetry，請參閱 AWS . OpenTelemetry 如需有關適用於 OpenTelemetry SDK 的發 AWS 行版的詳細資訊，請參閱簡介。

    此區段可以包含下列欄位：
    - `grpc_endpoint` – 選用。指定 CloudWatch 代理程式用來偵聽使用 GrPC 遠端程序呼叫傳送之 OpenTelemetry 追蹤的位址。格式是 `ip:port`。此位址必須符合 OpenTelemetry SDK 中 GrPC 匯出器所設定的位址。

      如果您省略此欄位，預設為使用 `127.0.0.1:4317`。
    - `http_endpoint` - 選用。指定 CloudWatch 代理程式用來監聽透過 HTTP 傳送之 OTLP 追蹤的位址。格式是 `ip:port`。此位址必須符合 OpenTelemetry SDK 中 HTTP 匯出器所設定的位址。

      如果您省略此欄位，預設為使用 `127.0.0.1:4318`。
- `concurrency` - 選用。指定可用於上傳追蹤的 X-Ray 的同時呼叫最大數目。預設值為 8
- `local_mode` - 選用。如果為 `true`，則代理程式不會收集 Amazon EC2 執行個體中繼資料。預設為 `false`。

- endpoint_override - 選用。指定要用作 CloudWatch 代理程式傳送追蹤的端點的 FIPS 端點或私人連結。指定此欄位並設定私有連結，可讓您將追蹤傳送到 Amazon VPC 端點。如需詳細資訊，請參閱什麼是 Amazon VPC。

  endpoint_override 的值必須是 URL 字串。
- region_override - 選用。指定要用於 X-Ray 端點的區域。 CloudWatch 代理程式會將追蹤傳送至指定區域中的 X-Ray。如果忽略此欄位，代理程式會將追蹤傳送至 Amazon EC2 執行個體所在的區域。

  如果在此處指定「區域」，則其會優先於組態檔案 agent 部分中的 region 參數。
- proxy_override - 選用。指定代 CloudWatch 理程式傳送要求至 X-Ray 時要使用的 Proxy 伺服器位址。必須將代理伺服器的通訊協定指定為此地址的一部分。
- credentials— 指定將追蹤傳送至其他 AWS 帳戶時要使用的 IAM 角色。若有指定，這個欄位會有一個 role_arn 參數。
  - role_arn— 指定將追蹤傳送至其他 AWS 帳戶時用於驗證的 IAM 角色 ARN。如需詳細資訊，請參閱 將指標、日誌和追蹤傳送到不同帳戶。此處若有指定，會覆寫組態檔案中 agent 區段指定的 role_arn (如果有)。

CloudWatch 代理程式設定檔：完整範例

以下是 Linux 伺服器的完整 CloudWatch 代理程式組態檔範例。

measurement 部分中所列、您想收集的指標項目，可以是特定完整的指標名稱，或只在指標名稱附加到類型的資源。例如，在 diskio 的 measurement 區段中指定 reads 或 diskio_reads部分將導致收集 diskio_reads 指標。

此範例包含在 measurement 區段中兩種指定指標的方式。

```
{
  "agent": {
    "metrics_collection_interval": 10,
    "logfile": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log"
  },
  "metrics": {
    "namespace": "MyCustomNamespace",
    "metrics_collected": {
      "cpu": {
        "resources": [
          "*"
```

```
            ],
            "measurement": [
              {"name": "cpu_usage_idle", "rename": "CPU_USAGE_IDLE", "unit":
 "Percent"},
              {"name": "cpu_usage_nice", "unit": "Percent"},
              "cpu_usage_guest"
            ],
            "totalcpu": false,
            "metrics_collection_interval": 10,
            "append_dimensions": {
              "customized_dimension_key_1": "customized_dimension_value_1",
              "customized_dimension_key_2": "customized_dimension_value_2"
            }
          },
          "disk": {
            "resources": [
              "/",
              "/tmp"
            ],
            "measurement": [
              {"name": "free", "rename": "DISK_FREE", "unit": "Gigabytes"},
              "total",
              "used"
            ],
             "ignore_file_system_types": [
              "sysfs", "devtmpfs"
            ],
            "metrics_collection_interval": 60,
            "append_dimensions": {
              "customized_dimension_key_3": "customized_dimension_value_3",
              "customized_dimension_key_4": "customized_dimension_value_4"
            }
          },
          "diskio": {
            "resources": [
              "*"
            ],
            "measurement": [
              "reads",
              "writes",
              "read_time",
              "write_time",
              "io_time"
            ],
```

```
            "metrics_collection_interval": 60
          },
          "swap": {
            "measurement": [
              "swap_used",
              "swap_free",
              "swap_used_percent"
            ]
          },
          "mem": {
            "measurement": [
              "mem_used",
              "mem_cached",
              "mem_total"
            ],
            "metrics_collection_interval": 1
          },
          "net": {
            "resources": [
              "eth0"
            ],
            "measurement": [
              "bytes_sent",
              "bytes_recv",
              "drop_in",
              "drop_out"
            ]
          },
          "netstat": {
            "measurement": [
              "tcp_established",
              "tcp_syn_sent",
              "tcp_close"
            ],
            "metrics_collection_interval": 60
          },
          "processes": {
            "measurement": [
              "running",
              "sleeping",
              "dead"
            ]
          }
        },
```

```
        "append_dimensions": {
          "ImageId": "${aws:ImageId}",
          "InstanceId": "${aws:InstanceId}",
          "InstanceType": "${aws:InstanceType}",
          "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
        },
        "aggregation_dimensions" : [["ImageId"], ["InstanceId", "InstanceType"],
 ["d1"],[]],
        "force_flush_interval" : 30
      },
      "logs": {
        "logs_collected": {
          "files": {
            "collect_list": [
              {
                "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-
agent.log",
                "log_group_name": "amazon-cloudwatch-agent.log",
                "log_stream_name": "amazon-cloudwatch-agent.log",
                "timezone": "UTC"
              },
              {
                "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/test.log",
                "log_group_name": "test.log",
                "log_stream_name": "test.log",
                "timezone": "Local"
              }
            ]
          }
        },
        "log_stream_name": "my_log_stream_name",
        "force_flush_interval" : 15,
        "metrics_collected": {
           "kubernetes": {
                "enhanced_container_insights": true
      }
    }
  }
}
```

以下是執行 Windows Server 之伺服器的完整 CloudWatch 代理程式組態檔範例。

```
{
```

```
      "agent": {
        "metrics_collection_interval": 60,
        "logfile": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\\amazon-
cloudwatch-agent.log"
      },
      "metrics": {
        "namespace": "MyCustomNamespace",
        "metrics_collected": {
          "Processor": {
            "measurement": [
              {"name": "% Idle Time", "rename": "CPU_IDLE", "unit": "Percent"},
              "% Interrupt Time",
              "% User Time",
              "% Processor Time"
            ],
            "resources": [
              "*"
            ],
            "append_dimensions": {
              "customized_dimension_key_1": "customized_dimension_value_1",
              "customized_dimension_key_2": "customized_dimension_value_2"
            }
          },
          "LogicalDisk": {
            "measurement": [
              {"name": "% Idle Time", "unit": "Percent"},
              {"name": "% Disk Read Time", "rename": "DISK_READ"},
              "% Disk Write Time"
            ],
            "resources": [
              "*"
            ]
          },
          "customizedObjectName": {
            "metrics_collection_interval": 60,
            "customizedCounterName": [
              "metric1",
              "metric2"
            ],
            "resources": [
              "customizedInstances"
            ]
          },
          "Memory": {
```

```
            "metrics_collection_interval": 5,
            "measurement": [
              "Available Bytes",
              "Cache Faults/sec",
              "Page Faults/sec",
              "Pages/sec"
            ]
          },
          "Network Interface": {
            "metrics_collection_interval": 5,
            "measurement": [
              "Bytes Received/sec",
              "Bytes Sent/sec",
              "Packets Received/sec",
              "Packets Sent/sec"
            ],
            "resources": [
              "*"
            ],
            "append_dimensions": {
              "customized_dimension_key_3": "customized_dimension_value_3"
            }
          },
          "System": {
            "measurement": [
              "Context Switches/sec",
              "System Calls/sec",
              "Processor Queue Length"
            ]
          }
        },
        "append_dimensions": {
          "ImageId": "${aws:ImageId}",
          "InstanceId": "${aws:InstanceId}",
          "InstanceType": "${aws:InstanceType}",
          "AutoScalingGroupName": "${aws:AutoScalingGroupName}"
        },
        "aggregation_dimensions" : [["ImageId"], ["InstanceId", "InstanceType"],
["d1"],[]]
      },
      "logs": {
        "logs_collected": {
          "files": {
            "collect_list": [
```

```
            {
              "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\
\amazon-cloudwatch-agent.log",
              "log_group_name": "amazon-cloudwatch-agent.log",
              "timezone": "UTC"
            },
            {
              "file_path": "c:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\Logs\
\test.log",
              "log_group_name": "test.log",
              "timezone": "Local"
            }
          ]
        },
        "windows_events": {
          "collect_list": [
            {
              "event_name": "System",
              "event_levels": [
                "INFORMATION",
                "ERROR"
              ],
              "log_group_name": "System",
              "log_stream_name": "System",
              "event_format": "xml"
            },
            {
              "event_name": "CustomizedName",
              "event_levels": [
                "WARNING",
                "ERROR"
              ],
              "log_group_name": "CustomizedLogGroup",
              "log_stream_name": "CustomizedLogStream",
              "event_format": "xml"
            }
          ]
        }
      },
      "log_stream_name": "example_log_stream_name"
    }
  }
```

手動儲存 CloudWatch 代理程式組態檔

如果您手動建立或編輯 CloudWatch 代理程式組態檔，您可以為其指定任何名稱。若要簡化故障診斷，我們建議您在 Linux 伺服器上將它命名為 `/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json`，且在執行 Windows Server 的伺服器上將它命名為 `$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json`。在您建立檔案後，您可以將此檔案複製到其他您希望執行代理程式的伺服器。

將 CloudWatch 代理程式組態檔上傳至 Systems Manager 參數存放區

如果您打算使用 SSM 代理程式在伺服器上安裝 CloudWatch 代理程式，在您手動編輯 CloudWatch 代理程式組態檔之後，您可以將其上傳到 Systems Manager 參數存放區。若要這樣做，您可以使用 Systems Manager `put-parameter` 命令。

若要能夠將檔案存放於參數存放區，您必須使用具有足夠許可的 IAM 角色。如需詳細資訊，請參閱 建立 IAM 角色和使用者以搭配 CloudWatch 代理程式使用。

使用以下命令，其中的 *`parameter name`* (參數名稱) 是參數存放區中要用於此檔案的名稱，而 *`configuration_file_pathname`* 是您已編輯的組態檔案的路徑和檔案名稱。

```
aws ssm put-parameter --name "parameter name" --type "String" --value
 file://configuration_file_pathname
```

啟用 CloudWatch 應用程式信

使用「應用 CloudWatch 程式信號」自動檢測您的應用程式，以 AWS 便根據業務目標追蹤應用程式效能。Application Signals 為您提供 Java 應用程式、其相依性及其優勢的統一、以應用程式為中心的檢視。如需詳細資訊，請參閱 Application Signals。

CloudWatch Application Signals 會利用 CloudWatch 代理程式從自動檢測的應用程式接收指標和追蹤，選擇性地套用規則以降低高基數，然後將處理的遙測發佈至。 CloudWatch您可以使用 CloudWatch 代理程式組態檔，為「應用程式訊號」專用的代理程式提供自訂組態。首先，代理程式組態檔`app_signals`區段中的區`metrics_collected`段下方會顯示`logs`區段，指定代理程式將從您自動檢測的應用 CloudWatch 程式接收指標。同樣地，代理程式組態檔`app_signals`區段中的`traces_collected`區段下方會顯示`traces`區段，指定代理程 CloudWatch 式已啟用以接收來自自動檢測之應用程式的追蹤。此外，您可以選擇性地傳遞自訂組態規則，以減少發佈本節所述的高基數遙測。

- 對於 Amazon EKS 叢集，當您安裝 Amazon 可 CloudWatch 觀測性 EKS 附加元件時， CloudWatch 代理程式預設會啟用，以從您的自動檢測應用程式接收指標和追蹤。如果您想要選擇性地傳遞自訂組

態規則，可以在建立或使用其他組態更新 Amazon EKS 附加元件時，在自訂代理程式組態中傳遞至該附加元件來實現，如 (選用) 額外組態 中所述。

- 對於包括 Amazon EC2 在內的其他支援平台，您必須使用 CloudWatch 代理程式組態啟動代理程式，該代理程式組態透過指定區app_signals段和選擇性地指定本節稍後所述的任何自訂組態規則來啟動代理程式。

以下是代理程 CloudWatch 式組態檔中與 CloudWatch 應用程式訊號相關之欄位的概觀。

- logs
  - metrics_collected— 此欄位可包含指定代理 CloudWatch 程式收集日誌以啟用應用程式訊號和容器見解等使用案例，並增強 Amazon EKS 的可觀察性。

    > **Note**
    >
    > 之前，此區段也用於指定代理程式來收集採用內嵌指標格式的日誌。不再需要這些設定。

    - app_signals(選擇性) 指定您要啟用「 CloudWatch 應用程式信號」，以便從自動檢測的應用程式接收指標，以便利 CloudWatch 應用程式訊號。
      - rules (選用) 規則陣列，用於有條件地選取指標和追蹤，並套用動作以處理高基數案例。每個規則可以包含下列欄位：
        - rule_name (選用) 規則的名稱。
        - selectors (選用) 一組指標和追蹤維度比對器。每個選取器都必須提供下列欄位：
          - dimension 如果 selectors 不為空，則為必填欄位。這會指定要用作篩選條件的指標和追蹤維度。
          - match 如果 selectors 不為空，則為必填欄位。用於比對指定的維度值的萬用字元模式。
        - action (選用) 要套用至符合指定選取器之指標和追蹤的動作。action 的值必須是下列其中一個關鍵字：
          - keep指定僅傳送度量和追蹤 (如 CloudWatch 果符合) selectors。
          - drop 指定丟棄與 selectors 相符的指標和追蹤。
          - replace 指定取代與 selectors 相符的指標和追蹤維度。它們根據 replacements 部分進行取代。

- replacements 在 action 為 replace 時需要。當 action 為 replace 時，維度和值對陣列將套用至符合指定 selectors 的指標和追蹤。每個取代都必須提供下列欄位：
  - target_dimension 如果 replacements 不為空，則為必填欄位。指定需要取代的維度。
  - value 如果 replacements 不為空，則為必填欄位。用來取代 target_dimension 原始值的值。
- limiter(選擇性) 使用此區段可限制應用程式訊號傳送至多少量度和維度 CloudWatch，以最佳化您的成本。
  - disabled(選擇性) 如果true，則會停用量度限制功能。預設為 false。
  - drop_threshold(選擇性) 一個 CloudWatch 代理程式可匯出的一個循環間隔內，每個服務的不同測量結果數目上限。預設值為 500。
  - rotation_interval(選擇性) 限制器重設量度記錄以進行區別計數的間隔。這表示為帶有數字序列和單位後綴的字符串。支持分數。支援的單位尾碼為s、、m、hms、us和 ns

    預設值1h為一小時。
  - log_dropped_metrics(選擇性) 指定卸除「應用程式訊號」度量時， CloudWatch 代理程式是否應將記錄寫入代理程式記錄檔。預設值為 false。

    > **Note**
    >
    > 若要啟動此記錄，agent區段中的debug參數也必須設定為true。

- traces
  - traces_collected
    - app_signals 選用。指定此選項可讓 CloudWatch 代理程式從自動檢測的應用程式接收追蹤，以便於應用程式 CloudWatch 訊號。

> **Note**
>
> 即使在 logs 區段中所包含的 metrics_collected 區段下指定自訂 app_signals 規則，它們也會隱式地套用至 traces_collected 區段。相同的規則集將套用至指標和追蹤。

當有多個具有不同動作的規則時，它們會依照下列順序套用：keep、drop、然後 replace。

以下是套用自訂規則的完整 CloudWatch 代理程式組態檔範例。

```
{
  "logs": {
    "metrics_collected": {
      "app_signals": {
        "rules": [
          {
            "rule_name": "keep01",
            "selectors": [
              {
                "dimension": "Service",
                "match": "pet-clinic-frontend"
              },
              {
                "dimension": "RemoteService",
                "match": "customers-service"
              }
            ],
            "action": "keep"
          },
          {
            "rule_name": "drop01",
            "selectors": [
              {
                "dimension": "Operation",
                "match": "GET /api/customer/owners/*"
              }
            ],
            "action": "drop"
          },
          {
            "rule_name": "replace01",
            "selectors": [
              {
                "dimension": "Operation",
                "match": "PUT /api/customer/owners/*/pets/*"
              },
              {
                "dimension": "RemoteOperation",
                "match": "PUT /owners"
              }
            ],
            "replacements": [
```

```
              {
                "target_dimension": "Operation",
                "value": "PUT /api/customer/owners/{ownerId}/pets{petId}"
              }
            ],
            "action": "replace"
          }
        ]
      }
    }
  },
  "traces": {
    "traces_collected": {
      "app_signals": {}
    }
  }
}
```

對於上一個範例組態檔案，rules 會按如下方式處理：

1. 規則 keep01 可確保維度 Service 為 pet-clinic-frontend 和維度 RemoteService 為 customers-service 的任何指標和追蹤被保留。
2. 對於套用 keep01 後已處理的指標和追蹤，drop01 規則可確保捨棄維度 Operation 為 GET /api/customer/owners/* 的指標和追蹤。
3. 對於套用 drop01 後已處理的指標和追蹤，replace01 規則會更新維度 Operation 為 PUT /api/customer/owners/*/pets/* 和維度 RemoteOperation 為 PUT /owners 的指標和追蹤，而它們的 Operation 維度現在會被取代為 PUT /api/customer/owners/{ownerId}/pets{petId}。

以下是 CloudWatch 組態檔案的完整範例，此檔案會透過將度量限制變更為 100、啟用捨棄的量度記錄，並將輪換間隔設定為兩小時，來管理「應用程式信號」中的基數。

```
{
    "logs": {
        "metrics_collected": {
            "app_signals": {
                "limiter": {
                    "disabled": false,
                    "drop_threshold": 100,
                    "rotation_interval": "2h",
```

```
                    "log_dropped_metrics": true
                }
            }
        },
        "traces": {
            "traces_collected": {
                "app_signals": {}
            }
        }
    }
}
```

收集網路效能指標

在 Linux 上執行的 EC2 執行個體，使用彈性網路轉接器 (ENA)，可會發佈網路效能指標。版本 1.246396.0 及更新版本的 CloudWatch 代理程式可讓您將這些網路效能指標匯入。 CloudWatch當您將這些網路效能指標匯入時 CloudWatch，會以 CloudWatch 自訂指標的形式收費。

如需有關 ENA 驅動程式的詳細資訊，請參閱在 Linux 執行個體上使用彈性網路轉接器 (ENA) 啟用增強型聯網和在 Windows 執行個體上使用彈性網路轉接器 (ENA) 啟用增強型聯網。

設定網路效能指標集合的方式會因 Linux 伺服器和 Windows 伺服器而異。

下表列出 ENA 轉接器啟用的這些網路效能指標。當 CloudWatch 代理程式將這些指標 CloudWatch 從 Linux 執行個體匯入時，會`ethtool_`在每個指標名稱的開頭加上前面。

指標	描述
Linux 伺服器的名稱：**`bw_in_allowance_exceeded`** Windows 伺服器的名稱：**`Aggregate inbound BW allowance exceeded`**	因傳入的彙總頻寬超過執行個體的上限而排入佇列及/或丟棄的封包數目。 只有在您已將此測量結果列在 CloudWatch 代理程式組態檔`ethtool`段落的子`metrics_collected` 段落時，才會收集此測量結果。如需詳細資訊，請參閱收集網路效能指標 單位：無
Linux 伺服器的名稱：**`bw_out_allowance_exceeded`**	因傳出的彙總頻寬超過執行個體的上限而排入佇列及/或丟棄的封包數目。

指標	描述
Windows 伺服器的名稱：**Aggregate outbound BW allowance exceeded**	只有在您已將此測量結果列在 CloudWatch 代理程式組態檔`ethtool`段落的子`metrics_collected`段落時，才會收集此測量結果。如需詳細資訊，請參閱收集網路效能指標 單位：無
Linux 伺服器的名稱：**conntrack_allowance_available** Windows 伺服器的名稱：**Available connection tracking allowance**	報告在達到該執行個體類型的追蹤連線限額之前，執行個體可建立的追蹤連線數目。此指標僅適用於使用 Linux 驅動程式適用於彈性網路介面卡 (ENA) 的硝基 EC2 執行個體，以及從 2.6.0 版開始使用 Windows 驅動程式彈性網路介面卡 (ENA) 的電腦上可用。 只有在您已將此測量結果列在 CloudWatch 代理程式組態檔`ethtool`段落的子`metrics_collected`段落時，才會收集此測量結果。如需詳細資訊，請參閱收集網路效能指標 單位：無
Linux 伺服器的名稱：**ena_srd_mode** Windows 伺服器的名稱：**ena srd mode**	說明已啟用哪些 ENA 快速功能。如需有關 ENA Express 的詳細資訊，請參閱在 Linux 執行個體上使用 ENA Express 改善網路效能的值如下： • 0 = ENA Express 關閉，UDP 關閉 • 1 = ENA Express 開啟，UDP 關閉 • 2 = ENA Express 關閉，UDP 開啟 **Note** 只有當初啟用 ENA Express，並且已將 UDP 設定為使用它時，才會發生這種情況。會保留 UDP 流量的先前值。 • 3 = ENA Express 開啟，UDP 開啟

指標	描述
Linux 伺服器的名稱：**ena_srd_eligible_tx_pkts** Windows 伺服器的名稱：**ena srd eligible tx pkts**	在指定時間範圍內傳送符合可 AWS 擴充可靠資格資料包 (SRD) 要求的網路封包數目，如下所示： • 支援傳送和接收執行個體類型。 • 傳送和接收執行個體都必須設定 ENA Express。 • 傳送和接收執行個體必須位於相同的子網路上。 • 執行個體之間的網路路徑不得包含中介軟體方塊。ENA Express 目前不支援中介軟體。
Linux 伺服器的名稱：**ena_srd_tx_pkts** Windows 伺服器的名稱：**ena srd tx pkts**	指定期間內傳輸的 SRD 封包數目。
Linux 伺服器的名稱：**ena_srd_rx_pkts** Windows 伺服器的名稱：**ena srd rx pkts**	指定期間內接收到的 SRD 封包數目。
Linux 伺服器的名稱：**ena_srd_resource_utilization** Windows 伺服器的名稱：**ena srd resource utilization**	執行個體已耗用的並行 SRD 連線所允許的最大記憶體使用率百分比。

指標	描述
Linux 伺服器的名稱：**linklocal_allowance_exceeded** Windows 伺服器的名稱：**Link local packet rate allowance exceeded**	由於本機代理伺服器服務的流量 PPS 超過網路介面上限而丟棄的封包數目。這會影響 DNS 服務、執行個體中繼資料服務和 Amazon Time Sync Service 的流量。 只有在您已將此測量結果列在 CloudWatch 代理程式組態檔`ethtool`段落的子`metrics_collected` 段落時，才會收集此測量結果。如需詳細資訊，請參閱收集網路效能指標 單位：無
Linux 伺服器的名稱：**linklocal_allowance_exceeded** Windows 伺服器的名稱：**Link local packet rate allowance exceeded**	由於本機代理伺服器服務的流量 PPS 超過網路介面上限而丟棄的封包數目。這會影響 DNS 服務、執行個體中繼資料服務和 Amazon Time Sync Service 的流量。 只有在您已將此測量結果列在 CloudWatch 代理程式組態檔`ethtool`段落的子`metrics_collected` 段落時，才會收集此測量結果。如需詳細資訊，請參閱收集網路效能指標 單位：無
Linux 伺服器的名稱：**pps_allowance_exceeded** Windows 伺服器的名稱：**PPS allowance exceeded**	因雙向 PPS 超過執行個體的上限而排入佇列及/或丟棄的封包數目。 只有在您已將此測量結果列在 CloudWatch 代理程式組態檔`ethtool`段落的子`metrics_collected` 段落時，才會收集此測量結果。如需詳細資訊，請參閱收集網路效能指標 單位：無

Linux 設定

在 Linux 伺服器上，ethtool 外掛程式可讓您將網路效能指標匯入。 CloudWatch

ethtool 是一個標準的 Linux 公用程序，可以收集有關 Linux 伺服器上乙太網路裝置的統計資料。它收集的統計資料取決於網路裝置和驅動程式。這些統計資料的範例包括 `tx_cnt`、`rx_bytes`、`tx_errors` 及 `align_errors`。將 ethtool 外掛程式與 CloudWatch 代理程式搭配使用時，也可以將 CloudWatch這些統計資料以及本節前面列出的 EC2 網路效能指標匯入。

> **Tip**
>
> 要尋找作業系統和網路裝置上可用的統計資訊，請使用 `ethtool -S` 命令。

當 CloudWatch 代理程式匯入量度時 CloudWatch，會在所有匯入量度的名稱中新增`ethtool_`首碼。因此，標準的 ethtool 統計信息`rx_bytes`被調`ethtool_rx_bytes`用 CloudWatch，並調`ethtool_bw_in_allowance_exceeded`用 EC2 網絡性能指標`bw_in_allowance_exceeded`。 CloudWatch

在 Linux 伺服器上，若要匯入 ethtool 度量，請在 CloudWatch 代理程式組態檔的`metrics_collected`區段中新增區段。`ethtool`ethtool 區段可以包含下列子區段：

- interface_include— 包含此區段會導致代理程式僅從本區段中列出名稱的介面收集指標。如果您省略此區段，則會從未列於 `interface_exclude` 的所有乙太網路介面收集指標。

  預設乙太網路介面為 `eth0`。
- interface_exclude— 如果您包含此區段，請列出您不想從中收集指標的乙太網路介面。

  ethtool 外掛程式總是忽略迴路介面。
- 量度 _ 包含 — 此段落會列出要匯入的測量結果。 CloudWatch它可以包含 ethtool 和 Amazon EC2 高解析度網路指標收集的標準統計資料。

下列範例顯示 CloudWatch 代理程式組態檔的一部分。此組態會收集標準的 ethtool 指標 `rx_packets` 和 `tx_packets`，以及僅來自 `eth1` 介面的 Amazon EC2 網路效能指標。

如需 CloudWatch 代理程式組態檔的詳細資訊，請參閱 手動建立或編輯 CloudWatch 代理程式組態檔。

```
"metrics": {
    "append_dimensions": {
      "InstanceId": "${aws:InstanceId}"
    },
```

```
    "metrics_collected": {
      "ethtool": {
        "interface_include": [
          "eth1"
        ],
        "metrics_include": [
          "rx_packets",
          "tx_packets",
          "bw_in_allowance_exceeded",
          "bw_out_allowance_exceeded",
          "conntrack_allowance_exceeded",
          "linklocal_allowance_exceeded",
          "pps_allowance_exceeded"
         ]
      }
   }
}
```

Windows 設定

在 Windows 伺服器上，網路效能測量結果可透過 Windows 效能計數器取得， CloudWatch 代理程式已從中收集測量結果。因此，您不需要外掛程式即可從 Windows 伺服器收集這些指標。

以下範組態檔案是用來從 Windows 收集網路效能指標。如需編輯 CloudWatch 代理程式組態檔的詳細資訊，請參閱 手動建立或編輯 CloudWatch 代理程式組態檔。

```
{
    "metrics": {
        "append_dimensions": {
            "InstanceId": "${aws:InstanceId}"
        },
        "metrics_collected": {
            "ENA Packets Shaping": {
                "measurement": [
                    "Aggregate inbound BW allowance exceeded",
                    "Aggregate outbound BW allowance exceeded",
                    "Connection tracking allowance exceeded",
                    "Link local packet rate allowance exceeded",
                    "PPS allowance exceeded"
                ],
                "metrics_collection_interval": 60,
                "resources": [
                    "*"
```

```
                ]
            }
        }
    }
}
```

檢視網路效能指標

將網路效能指標匯入後 CloudWatch，您可以將這些指標視為時間序列圖表，並建立警示以監視這些指標，並在超出您指定的臨界值時通知您。下列程序顯示如何以時間序列圖形檢視 ethtool 指標。如需設定警示的詳細資訊，請參閱 使用 Amazon CloudWatch 警報。

由於所有這些量度都是彙總計數器，因此您可以使用度 CloudWatch 量數學函數，例如`RATE(METRICS())`在圖表中計算這些量度的比率，或使用它們來設定警示。如需指標數學函數的詳細資訊，請參閱 使用指標數學。

在 CloudWatch 主控台中檢視網路效能測量結果

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇 指標。
3. 選擇用於代理程式所收集指標的命名空間。依預設，這是 CWAgent，但您可能在 CloudWatch 代理程式組態檔中指定了不同的命名空間。
4. 選擇指標維度 (例如 Per-Instance Metrics (每個執行個體指標))。
5. All metrics (所有指標) 索引標籤會顯示命名空間中該維度的所有指標。您可以執行下列作業：
   a. 若要將指標圖形化，請勾選指標旁的核取方塊。若要選擇所有指標，請勾選表格標題列中的核取方塊。
   b. 若要將資料表排序，請使用直欄標題。
   c. 若要依資源篩選，請選擇資源 ID，然後選擇 Add to search (新增至搜尋)。
   d. 若要依指標篩選，請選擇指標名稱，然後選擇 Add to search (新增至搜尋)。
6. (選擇性) 若要將此圖形新增至 CloudWatch 儀表板，請選擇 [動作]，然後選擇 [新增至儀表板]。

收集 NVIDIA GPU 指標

您可以使用 CloudWatch 代理程式從 Linux 伺服器收集 NVIDIA GPU 指標。若要進行設定，請在 CloudWatch 代理程式組態檔的`metrics_collected`區段中新增區段。`nvidia_gpu`如需詳細資訊，請參閱 Linux 區段。

此外，執行個體必須已安裝 NVIDIA 驅動程式。某些 Amazon Machine Image (AMI) 上預先安裝了 NVIDIA 驅動程式。或者，可以手動安裝驅動程式。如需詳細資訊，請參閱在 Linux 執行個體上安裝 NVIDIA 驅動程式。

可以收集以下指標。所有這些測量結果都是否收集的 CloudWatch `Unit`，但是您可以將參數新增至 CloudWatch 代理程式組態檔，以指定每個測量結果的單位。如需詳細資訊，請參閱 Linux 區段。

指標	測量結果名稱 CloudWatch	描述
`utilization_gpu`	`nvidia_smi_utilization_gpu`	在過去的抽樣週期內，GPU 上的一個或多個核心執行的時間百分比。
`temperature_gpu`	`nvidia_smi_temperature_gpu`	核心 GPU 溫度 (以攝氏度為單位)。
`power_draw`	`nvidia_smi_power_draw`	整個電路板的最後一次測量功耗 (以瓦特為單位)。
`utilization_memory`	`nvidia_smi_utilization_memory`	在過去的抽樣週期內讀取或寫入全域 (裝置) 記憶體的時間百分比。
`fan_speed`	`nvidia_smi_fan_speed`	裝置風扇目前計劃執行的最大風扇速度百分比。
`memory_total`	`nvidia_smi_memory_total`	記錄的記憶體總計 (MB)。
`memory_used`	`nvidia_smi_memory_used`	使用的記憶體 (MB)。
`memory_free`	`nvidia_smi_memory_free`	可用的記憶體 (MB)。
`pcie_link_gen_current`	`nvidia_smi_pcie_link_gen_current`	目前連結版本。

指標	測量結果名稱 CloudWatch	描述
pcie_link_width_current	nvidia_smi_pcie_link_width_current	目前連結頻寬。
encoder_stats_session_count	nvidia_smi_encoder_stats_session_count	目前編碼器工作階段數。
encoder_stats_average_fps	nvidia_smi_encoder_stats_average_fps	每秒編碼影格的移動平均值。
encoder_stats_average_latency	nvidia_smi_encoder_stats_average_latency	編碼延遲的移動平均值 (以微秒為單位)。
clocks_current_graphics	nvidia_smi_clocks_current_graphics	圖形 (著色器) 時脈的目前頻率。
clocks_current_sm	nvidia_smi_clocks_current_sm	串流多處理器 (SM) 時脈的目前頻率。
clocks_current_memory	nvidia_smi_clocks_current_memory	記憶體時脈的目前頻率。
clocks_current_video	nvidia_smi_clocks_current_video	影片 (編碼器與解碼器) 時脈的目前頻率。

所有這些指標均採用以下維度進行收集：

維度	描述
index	此伺服器上 GPU 的唯一識別碼。表示裝置的 NVIDIA 管理庫 (NVML) 索引。
name	GPU 的類型。例如 NVIDIA Tesla A100
host	伺服器主機名稱。

使用 procstat 外掛程式收集程序指標

procstat 外掛程式可讓您從個別的程序收集指標。它在 Linux 服務器和運行支持的 Windows 服務器版本的服務器上支持。

主題

- 設定監測器的 CloudWatch 代理程式
- Procstat 收集的指標
- 檢視 CloudWatch 代理程式匯入的程序測量結果

設定監測器的 CloudWatch 代理程式

若要使用 procstat 外掛程式，請在 CloudWatch 代理程式設定檔的metrics_collected區段中新增區段。procstat有三種方法可以指定要監控的程序。您只能使用其中一種方法，但您可以使用該方法來指定一或多個要監控的程序。

- pid_file：依程序所建立的程序識別碼 (PID) 檔案的名稱來選取程序。
- exe：使用規則表達式比對規則，選取程序名稱符合指定字串的程序。符合是「內含」的符合，表示如果指定必須符合的詞彙為 agent，則具有 cloudwatchagent 一類名稱的處理程序就符合該詞彙。如需詳細資訊，請參閱語法。

- pattern：依用於啟動程序的命令列來選取程序。系統會選取命令列符合以規則表達式比對規則所指定之字串的所有程序。整個命令列會經過檢查，包括命令中使用的參數和選項。

  符合是「內含」的符合，表示如果指定必須符合的詞彙為 -c，則含有 -config 一類參數的處理程序就符合該詞彙。
- drop_original_metrics - 選用。如果您使用 metrics 區段中的 aggregation_dimensions 欄位，將指標彙總為彙總結果，則代理程式預設會同時傳送為每個維度值分開的彙總指標和原始指標。如果您不想將原始量度傳送至 CloudWatch，可以使用量度清單來指定此參數。與此參數一起指定的量度沒有依維度報告的量度 CloudWatch。而是只報告彙總指標。這樣可以減少代理程式收集的指標數量，降低您的成本。

CloudWatch 代理程式只會使用其中一種方法，即使您包含上述其中一個以上的章節也一樣。如果您指定多個區段， CloudWatch 代理程式會使用該pid_file區段 (如果存在)。如果不存在，則使用 exe 區段。

在 Linux 伺服器上，您在 exe 或 pattern 區段中指定的字串會當做規則表達式來評估。在執行 Windows Server 的伺服器上，這些字串會當做 WMI 查詢來評估。例如，即改為 pattern: "%apache%"。如需詳細資訊，請參閱 LIKE 運算子。

無論您採用哪一種方法，您可以包含選用的 metrics_collection_interval 參數，以指定收集這些指標的頻率 (以秒為單位)。如果您省略此參數，則會使用預設值的 60 秒。

在以下幾節的範例中，procstat 區段是代理程式組態檔案的 metrics_collected 區段包含的唯一區段。實際組態檔案在 metrics_collected 中也可能包含其他區段。如需詳細資訊，請參閱 手動建立或編輯 CloudWatch 代理程式組態檔。

使用 pid_file 來設定

以下範例 procstat 區段監控 PID 檔案 example1.pid 和 example2.pid 的建立程序。從每個程序收集不同的指標。從建立 example2.pid 的程序收集的指標是每 10 秒收集一次，而從 example1.pid 程序收集的指標是每 60 秒 (預設值) 收集一次。

```
{
    "metrics": {
        "metrics_collected": {
            "procstat": [
                {
                    "pid_file": "/var/run/example1.pid",
                    "measurement": [
                        "cpu_usage",
```

```
                        "memory_rss"
                    ]
                },
                {
                    "pid_file": "/var/run/example2.pid",
                    "measurement": [
                        "read_bytes",
                        "read_count",
                        "write_bytes"
                    ],
                    "metrics_collection_interval": 10
                }
            ]
        }
    }
}
```

使用 exe 來設定

以下範例 procstat 區段監控名稱符合字串 agent 或 plugin 的所有程序。從每個程序收集相同的指標。

```
{
    "metrics": {
        "metrics_collected": {
            "procstat": [
                {
                    "exe": "agent",
                    "measurement": [
                        "cpu_time",
                        "cpu_time_system",
                        "cpu_time_user"
                    ]
                },
                {
                    "exe": "plugin",
                    "measurement": [
                        "cpu_time",
                        "cpu_time_system",
                        "cpu_time_user"
                    ]
                }
            ]
```

```
        }
    }
}
```

使用 pattern 來設定

以下範例 procstat 區段監控命令列符合字串 config 或 -c 的所有程序。從每個程序收集相同的指標。

```
{
    "metrics": {
        "metrics_collected": {
            "procstat": [
                {
                    "pattern": "config",
                    "measurement": [
                        "rlimit_memory_data_hard",
                        "rlimit_memory_data_soft",
                        "rlimit_memory_stack_hard",
                        "rlimit_memory_stack_soft"
                    ]
                },
                {
                    "pattern": "-c",
                    "measurement": [
                        "rlimit_memory_data_hard",
                        "rlimit_memory_data_soft",
                        "rlimit_memory_stack_hard",
                        "rlimit_memory_stack_soft"
                    ]
                }
            ]
        }
    }
}
```

Procstat 收集的指標

下表列出您可以使用 procstat 外掛程式收集的指標。

CloudWatch 代理程式會新增procstat至下列測量結果名稱的開頭。根據從 Linux 伺服器或執行 Windows Server 的伺服器收集指標而定，而有不同的語法。例如，cpu_time 指標從 Linux 收集時會顯示為 procstat_cpu_time，而從 Windows Server 收集時會顯示為 procstat cpu_time。

指標名稱	可用位置	描述
cpu_time	Linux	程序使用 CPU 的時間量。這個指標是以百分之一秒來測量。 單位：計數
cpu_time_guest	Linux	程序處於訪客模式的時間長度。這個指標是以百分之一秒來測量。 類型：浮點數 單位：無
cpu_time_guest_nice	Linux	程序以良好訪客狀態執行的時間長度。這個指標是以百分之一秒來測量。 類型：浮點數 單位：無
cpu_time_idle	Linux	程序處於閒置模式的時間長度。這個指標是以百分之一秒來測量。 類型：浮點數

指標名稱	可用位置	描述
		單位：無
cpu_time_iowait	Linux	程序等待 I/O 操作完成所需的時間。這個指標是以百分之一秒來測量。 類型：浮點數 單位：無
cpu_time_irq	Linux	程序處於服務中斷狀態的時間長度。這個指標是以百分之一秒來測量。 類型：浮點數 單位：無
cpu_time_nice	Linux	程序處於良好模式的時間長度。這個指標是以百分之一秒來測量。 類型：浮點數 單位：無

指標名稱	可用位置	描述
cpu_time_soft_irq	Linux	程序處於服務軟體中斷狀態的時間長度。這個指標是以百分之一秒來測量。 類型：浮點數 單位：無
cpu_time_steal	Linux	在虛擬化環境中執行時，在其他作業系統上執行所花費的時間。這個指標是以百分之一秒來測量。 類型：浮點數 單位：無

指標名稱	可用位置	描述
cpu_time_stolen	Linux、Windows Server	程序處於遭竊時間狀態的時間長度，也是在虛擬化環境中的其他作業系統上所花費的時間。這個指標是以百分之一秒來測量。 類型：浮點數 單位：無
cpu_time_system	Linux、Windows Server、macOS	程序處於系統模式的時間量。這個指標是以百分之一秒來測量。 類型：浮點數 單位：計數
cpu_time_user	Linux、Windows Server、macOS	程序處於使用者模式的時間量。這個指標是以百分之一秒來測量。 單位：計數

指標名稱	可用位置	描述
cpu_usage	Linux、Windows Server、macOS	程序在任何容量中運作的時間百分比。 單位：百分比
memory_data	Linux、macOS	程序用在資料上的記憶體數量。 單位：位元組
memory_locked	Linux、macOS	程序已鎖定的記憶體數量。 單位：位元組
memory_rss	Linux、Windows Server、macOS	程序正在使用的真實記憶體 (常駐集) 數量。 單位：位元組
memory_stack	Linux、macOS	程序正在使用的堆疊記憶體數量。 單位：位元組
memory_swap	Linux、macOS	程序正在使用的切換記憶體數量。 單位：位元組

指標名稱	可用位置	描述
memory_vms	Linux、Windows Server、macOS	程序正在使用的虛擬記憶體數量。 單位：位元組
num_fds	Linux	此程序已開啟的檔案描述項數量。 單位：無
num_threads	Linux、Windows、macOS	程序中的執行緒數量。 單位：無
pid	Linux、Windows Server、macOS	程序識別符 (ID)。 單位：無

指標名稱	可用位置	描述
pid_count	Linux、Windows Server、macOS	與程序相關聯的程序 ID 數目。 在 Linux 伺服器和 macOS 電腦上，此指標的完整名稱是 procstat_lookup_pid_count ，在 Windows Server 上則是 procstat_lookup pid_count 。 單位：無
read_bytes	Linux、Windows Server	程序已從磁碟讀取的位元組數目。 單位：位元組
write_bytes	Linux、Windows Server	程序已寫入磁碟的位元組數目。 單位：位元組
read_count	Linux、Windows Server	程序已執行的磁碟讀取操作次數。 單位：無

指標名稱	可用位置	描述
rlimit_realtime_priority_hard	Linux	可為此程序設定之即時優先順序的硬性限制。 單位：無
rlimit_realtime_priority_soft	Linux	可為此程序設定之即時優先順序的彈性限制。 單位：無
rlimit_signals_pending_hard	Linux	此程序可排入佇列之訊號數量上限的硬性限制。 單位：無
rlimit_signals_pending_soft	Linux	此程序可排入佇列之訊號數量上限的彈性限制。 單位：無
rlimit_nice_priority_hard	Linux	此程序可設定之最高良好優先順序的硬性限制。 單位：無

指標名稱	可用位置	描述
rlimit_nice_priority_soft	Linux	此程序可設定之最高良好優先順序的彈性限制。 單位：無
rlimit_num_fds_hard	Linux	此程序可開啟之檔案描述項數量上限的硬性限制。 單位：無
rlimit_num_fds_soft	Linux	此程序可開啟之檔案描述項數量上限的彈性限制。 單位：無
write_count	Linux、Windows Server	程序已執行的磁碟寫入操作次數。 單位：無
involuntary_context_switches	Linux	程序非自願切換內容的次數。 單位：無
voluntary_context_switches	Linux	程序自願切換內容的次數。 單位：無

指標名稱	可用位置	描述
realtime_priority	Linux	程序目前使用即時優先順序的情形。 單位：無
nice_priority	Linux	程序目前使用適當優先順序的情形。 單位：無
signals_pending	Linux	等待由程序處理的訊號數目。 單位：無
rlimit_cpu_time_hard	Linux	程序的硬性 CPU 時間資源限制。 單位：無
rlimit_cpu_time_soft	Linux	程序的彈性 CPU 時間資源限制。 單位：無
rlimit_file_locks_hard	Linux	程序的硬性檔案鎖定資源限制。 單位：無

指標名稱	可用位置	描述
rlimit_file_locks_soft	Linux	程序的彈性檔案鎖定資源限制。 單位：無
rlimit_memory_data_hard	Linux	程序上用於資料的記憶體方面的硬性資源限制。 單位：位元組
rlimit_memory_data_soft	Linux	程序上用於資料的記憶體方面的彈性資源限制。 單位：位元組
rlimit_memory_locked_hard	Linux	程序上關於鎖定記憶體的硬性資源限制。 單位：位元組
rlimit_memory_locked_soft	Linux	程序上關於鎖定記憶體的彈性資源限制。 單位：位元組
rlimit_memory_rss_hard	Linux	程序上關於實體記憶體的硬性資源限制。 單位：位元組

指標名稱	可用位置	描述
rlimit_memory_rss_soft	Linux	程序上關於實體記憶體的彈性資源限制。 單位：位元組
rlimit_memory_stack_hard	Linux	程序堆疊上的硬性資源限制。 單位：位元組
rlimit_memory_stack_soft	Linux	程序堆疊上的彈性資源限制。 單位：位元組
rlimit_memory_vms_hard	Linux	程序上關於虛擬記憶體的硬性資源限制。 單位：位元組
rlimit_memory_vms_soft	Linux	程序上關於虛擬記憶體的彈性資源限制。 單位：位元組

檢視 CloudWatch 代理程式匯入的程序測量結果

將流程指標匯入之後 CloudWatch，您可以將這些指標視為時間序列圖表，並建立警示以監視這些指標，並在超出您指定的臨界值時通知您。下列程序顯示如何以時間序列圖形檢視程序指標。如需設定警示的詳細資訊，請參閱 使用 Amazon CloudWatch 警報。

若要在 CloudWatch 主控台中檢視程序指標

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇 指標。
3. 選擇用於代理程式所收集指標的命名空間。依預設，這是 CWAgent，但您可能在 CloudWatch 代理程式組態檔中指定了不同的命名空間。
4. 選擇指標維度 (例如 Per-Instance Metrics (每個執行個體指標))。
5. All metrics (所有指標) 索引標籤會顯示命名空間中該維度的所有指標。您可以執行下列作業：
   a. 若要將指標圖形化，請勾選指標旁的核取方塊。若要選擇所有指標，請勾選表格標題列中的核取方塊。
   b. 若要將資料表排序，請使用直欄標題。
   c. 若要依資源篩選，請選擇資源 ID，然後選擇 Add to search (新增至搜尋)。
   d. 若要依指標篩選，請選擇指標名稱，然後選擇 Add to search (新增至搜尋)。
6. (選擇性) 若要將此圖形新增至 CloudWatch 儀表板，請選擇動作 > 新增至儀表板。

使用 StatsD 擷取自訂指標

您可以使用 CloudWatch代理程式搭配StatsD通訊協定，從應用程式或服務擷取其他自訂指標。StatSD 是一種普遍的開放原始碼解決方案，可以從各種應用程式收集指標。StatSD 對於檢測您自己的指標特別有用。如需將 CloudWatch 代理程式和 StatsD 一起使用的範例，請參閱如何使用 Amazon 代理程式更好地監控自訂應用程 CloudWatch 式指標。

StatsD在 Linux 伺服器和執行視窗伺服器的伺服器上均受支援。 CloudWatch 支援以下StatsD格式：

```
MetricName:value|type|@sample_rate|#tag1:
  value,tag1...
```

- MetricName – 字串，不含冒號、直線、# 字元或 @ 字元。
- value – 這可以是整數或浮點數。
- type – 指定 c (用於計數器)、g (用於測量儀)、ms (用於計時器)、h (用於長條圖) 或 s (用於 set)。
- sample_rate – (選用) 介於 0 和 1 的浮點數，包含 0 和 1。僅適用於計數器、長條圖、計時器指標。預設值為 1 (抽樣 100% 的時間)。

- `tags`— (選用) 以逗號分隔的標籤清單。 StatsD標籤與中的標註類似 CloudWatch。對鍵/值標籤使用冒號，例如 `env:prod`。

您可以使用任何遵循此格式的用StatsD戶端，將度量傳送至 CloudWatch 代理程式。如需有關某些可用用StatsD戶端的詳細資訊，請參閱中的 StatsD 用戶端頁面 GitHub。

若要收集這些自訂指標，請將 `"statsd": {}` 這一行新增到代理程式組態檔案的 `metrics_collected` 區段。您可以手動加入這一行。如果您使用精靈來建立組態檔案，它會替您完成這項作業。如需詳細資訊，請參閱 建立 CloudWatch 代理程式組態檔。

StatsD 預設的組態可適用於大部分的使用者。有多個選用欄位，您可以視需要將其新增至代理程式組態檔的 statsd 區段：

- `service_address`— CloudWatch 代理程式應監聽的服務位址。格式是 *`ip`*`:`*`port`*。如果您省略 IP 地址，代理程式會監聽所有可用介面。僅支援 UDP 格式，因此您不需要指定 UDP 前綴。

  預設值為 `:8125`。
- `metrics_collection_interval` – StatsD 外掛程式執行和收集指標的頻率 (以秒為單位)。預設值為 10 秒。範圍介於 1 – 172,000 之間。
- `metrics_aggregation_interval`— 將指標 CloudWatch彙總到單一資料點的頻率 (以秒為單位)。預設值為 60 秒。

  例如，如果`metrics_collection_interval`為 10 且`metrics_aggregation_interval`為 60，則每 10 秒 CloudWatch 收集一次資料。每分鐘之後，該分鐘的六個資料讀數會彙總成單一資料點，並傳送至該資料點 CloudWatch。

  範圍介於 0 – 172,000 之間。將 `metrics_aggregation_interval` 設為 0，會停止彙整 StatsD 指標。
- `allowed_pending_messages` – 允許排入佇列的 UDP 訊息數量。當佇列已滿時，StatsD 伺服器會開始捨棄封包。預設值為 10000。
- `drop_original_metrics` - 選用。如果您使用 `metrics` 區段中的 `aggregation_dimensions` 欄位，將指標彙總為彙總結果，則代理程式預設會同時傳送為每個維度值分開的彙總指標和原始指標。如果您不想將原始量度傳送至 CloudWatch，可以使用量度清單來指定此參數。與此參數一起指定的量度沒有依維度報告的量度 CloudWatch。而是只報告彙總指標。這樣可以減少代理程式收集的指標數量，降低您的成本。

以下範例是代理程式組態檔案的 statsd 區段，其中使用預設的連接埠和自訂收集以及彙總間隔。

```
{
   "metrics":{
      "metrics_collected":{
         "statsd":{
            "service_address":":8125",
            "metrics_collection_interval":60,
            "metrics_aggregation_interval":300
         }
      }
   }
}
```

檢視代理程式匯入的 CloudWatch StatsD 測量結果

將 StatsD 指標匯入後 CloudWatch，您可以將這些指標視為時間序列圖表，並建立警示以監視這些指標，並在超出您指定的閾值時通知您。下列程序顯示如何以時間序列圖形檢視 StatsD 指標。如需設定警示的詳細資訊，請參閱 使用 Amazon CloudWatch 警報。

若要在主 CloudWatch 控台中檢視 StatsD 指標

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇 指標。
3. 選擇用於代理程式所收集指標的命名空間。依預設，這是 CWAgent，但您可能在 CloudWatch 代理程式組態檔中指定了不同的命名空間。
4. 選擇指標維度 (例如 Per-Instance Metrics (每個執行個體指標))。
5. All metrics (所有指標) 索引標籤會顯示命名空間中該維度的所有指標。您可以執行下列作業：
   a. 若要將指標圖形化，請勾選指標旁的核取方塊。若要選擇所有指標，請勾選表格標題列中的核取方塊。
   b. 若要將資料表排序，請使用直欄標題。
   c. 若要依資源篩選，請選擇資源 ID，然後選擇 Add to search (新增至搜尋)。
   d. 若要依指標篩選，請選擇指標名稱，然後選擇 Add to search (新增至搜尋)。
6. (選擇性) 若要將此圖形新增至 CloudWatch 儀表板，請選擇動作 > 新增至儀表板。

使用 collectd 擷取自訂指標

您可以使用帶有 collectd 通訊協定的 CloudWatch代理程式，從應用程式或服務擷取其他指標，此通訊協定僅在 Linux 伺服器上受支援。collectd 是一種流行的開放原始碼解決方案，其中包含外掛程式，可

以收集各種應用程式的系統統計資料。透過將 CloudWatch 代理程式已經收集的系統指標與 collectd 的其他指標相結合，您可以更好地監控、分析和疑難排解系統和應用程式。如需 collectd 的詳細資訊，請參閱 collectd - 系統統計資料收集協助程式。

您可以使用 collectd 軟體將度量傳送至 CloudWatch 代理程式。對於 collectd 度量， CloudWatch 代理程式充當伺服器，而 collectd 外掛程式則充當用戶端。

每個伺服器上不會自動安裝 collectd 軟體。在執行 Amazon Linux 2 的伺服器上，按照下列步驟安裝 collectd

```
sudo amazon-linux-extras install collectd
```

如需有關如何在其他系統上安裝 collectd 的資訊，請參閱 collectd 的下載頁面。

若要收集這些自訂指標，請將 "collectd": {} 這一行新增至代理程式組態檔案的 metrics_collected 區段。您可以手動加入這一行。如果您使用精靈來建立組態檔案，它會替您完成。如需詳細資訊，請參閱建立 CloudWatch 代理程式組態檔。

也有選用的參數。如果您使用 collectd，且未使用 `/etc/collectd/auth_file` 作為 collectd_auth_file，則必須設定其中一些選項。

- 服務位址： CloudWatch 代理程式應監聽的服務位址。格式是 "udp://*ip*:*port*。預設值為 udp://127.0.0.1:25826。
- name_prefix：每個 collectd 指標名稱開頭的字首。預設值為 `collectd_`。長度上限為 255 個字元。
- collectd_security_level：設定網路通訊的安全層級。預設設定為 encrypt (加密)。

  encrypt (加密) 可指定只接受有加密的資料。sign (簽署) 可指定只接受已簽署和加密的資料。none (無) 可指定接受所有資料。如果您為 collectd_auth_file 指定值，若有可能，加密的資料會解密。

  如需詳細資訊，請參閱 collectd Wiki 中的 Client 設定和可能的互動。
- collectd_auth_file 設置一個檔案，其中的使用者名稱對應到密碼。這些密碼用於驗證簽章以及解密加密的網路封包。如有提供，即可驗證已簽署的資料，並將加密的封包解密。否則，會在未檢查簽章的情況下接受已簽署的資料，且無法解密加密的資料。

  預設值為 `/etc/collectd/auth_file`。

  如果 collectd_security_level (collectd_security_level) 設為 none (無)，則這是選用參數。如果您將 collectd_security_level (collectd_security_level) 設為 `encrypt` 或 sign (簽署)，則必須指定 collectd_auth_file (collectd_auth_file)。

授權檔案的格式是，每一行一個使用者名稱，加一個冒號和任意數量的空格，之後再加上密碼。例如：

```
user1: user1_password

user2: user2_password
```

- collectd_typesdb: 包含資料集描述項的一或多個檔案的清單。清單必須以方括號圍起來，即使清單中只有一個項目。清單中的每個項目都必須加上雙引號。如果有多個項目，以逗號分隔。Linux 伺服器的預設值是 `["/usr/share/collectd/types.db"]`。macOs 電腦上的預設值取決於 collectd 的版本。例如：`["/usr/local/Cellar/collectd/5.12.0/share/collectd/types.db"]`。

  如需詳細資訊，請參閱 https://www.collectd.org/documentation/manpages/types.db.html。
- 指標 _ 聚合間隔：將指標彙總到單一資料點的頻率 (以秒為單 CloudWatch位)。預設值為 60 秒。範圍介於 0 至 172,000 之間。若設定為 0，會停止對 collectd 指標進行彙總。

以下是適用於代理程式組態檔案 collectd 區段的範例。

```
{
   "metrics":{
      "metrics_collected":{
         "collectd":{
            "name_prefix":"My_collectd_metrics_",
            "metrics_aggregation_interval":120
         }
      }
   }
}
```

檢視代理程式匯入的 CloudWatch 收集測量結果

將 collectd 指標匯入後 CloudWatch，您可以將這些指標作為時間序列圖表檢視，並建立警示以監視這些指標，並在超出您指定的臨界值時通知您。下列程序顯示如何以時間序列圖形檢視 collectd 指標。如需設定警示的詳細資訊，請參閱 使用 Amazon CloudWatch 警報。

若要在主 CloudWatch 控台中檢視收集量度

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇 指標。

3. 選擇用於代理程式所收集指標的命名空間。依預設，這是 CWAgent，但您可能在 CloudWatch 代理程式組態檔中指定了不同的命名空間。
4. 選擇指標維度 (例如 Per-Instance Metrics (每個執行個體指標))。
5. All metrics (所有指標) 索引標籤會顯示命名空間中該維度的所有指標。您可以執行下列作業：
   a. 若要將指標圖形化，請勾選指標旁的核取方塊。若要選擇所有指標，請勾選表格標題列中的核取方塊。
   b. 若要將資料表排序，請使用直欄標題。
   c. 若要依資源篩選，請選擇資源 ID，然後選擇 Add to search (新增至搜尋)。
   d. 若要依指標篩選，請選擇指標名稱，然後選擇 Add to search (新增至搜尋)。
6. (選擇性) 若要將此圖形新增至 CloudWatch 儀表板，請選擇動作 > 新增至儀表板。

在 Amazon EC2 執行個體上安裝和設定 Prometheus 指標集合

以下各節說明如何在 EC2 執行個體上透過 Prometheus 監控安裝 CloudWatch 代理程式，以及如何設定代理程式以抓取其他目標。它也提供了設定範例工作負載的教學課程，以便搭配 Prometheus 監控使用測試。

如需 CloudWatch 代理程式支援之作業系統的相關資訊，請參閱 使用 CloudWatch 代理程式收集指標、記錄和追蹤

VPC 安全群組要求

如果您使用 VPC，則適用下列需求。

- Prometheus 工作負載的安全群組輸入規則必須向 CloudWatch 代理程式開啟 Prometheus 連接埠，以便透過私有 IP 擷取 Prometheus 度量。
- 代理程式的安全性群組輸出規則必須允許 CloudWatch 代理程式透過 CloudWatch 私有 IP 連線至 Prometheus 工作負載的連接埠。

主題

- 步驟 1：安裝 CloudWatch 代理程式
- 步驟 2：湊集 Prometheus 來源並匯入指標
- 範例：為 Prometheus 指標測試設定 Java/JMX 範例工作負載

步驟 1：安裝 CloudWatch 代理程式

第一步是在 EC2 執行個體上安裝 CloudWatch 代理程式。如需說明，請參閱安裝 CloudWatch 代理程式。

步驟 2：湊集 Prometheus 來源並匯入指標

具有 Prometheus 監控的 CloudWatch 代理需要兩種配置來抓取 Prometheus 指標。其中一個是 Prometheus 文件的 <scrape_config> 中記錄的標準 Prometheus 湊集組態。另一個用於代 CloudWatch 理配置。

Prometheus 湊集組態

該 CloudWatch 代理支持標準的 Prometheus 抓取配置，如 Prometheus 文檔https://prometheus.io/docs/prometheus/latest/configuration/configuration/#scrape_config<scrape_config>中所述。您可以編輯此區段來更新已存在於此檔案中的組態，並新增其他 Prometheus 湊集目標。範例組態檔案包含下列全域組態行：

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus.yaml
global:
  scrape_interval: 1m
  scrape_timeout: 10s
scrape_configs:
- job_name: MY_JOB
  sample_limit: 10000
  file_sd_configs:
    - files: ["C:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\prometheus_sd_1.yaml",
 "C:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\\prometheus_sd_2.yaml"]
```

`global` 區段會指定在所有組態環境中都有效的參數。它們也可作為其他組態區段的預設值。它包含以下參數：

- `scrape_interval`— 定義湊集目標的頻率。
- `scrape_timeout`— 定義湊集請求逾時之前要等待的時間。

`scrape_configs` 區段會指定一組定義如何湊集它們的目標和參數。它包含以下參數：

- `job_name`— 預設指派給湊集指標的任務名稱。
- `sample_limit`— 將被接受的湊集樣本數量的每個湊集限制。

- `file_sd_configs`— 檔案服務探索組態的清單。它讀取一組包含零個或多個靜態組態清單的檔案。`file_sd_configs` 區段包含 `files` 參數，該參數可定義從中擷取目標群組的檔案模式。

CloudWatch 代理程式支援下列服務探索組態類型。

`static_config` 允許指定目標清單及其常用標籤集。這是一種可在湊集組態中指定靜態目標的正式方式。

以下是從本地主機湊集 Prometheus 指標的範例靜態組態。如果 Prometheus 連接埠對代理程式執行所在的伺服器開啟，也可以從其他伺服器中湊集指標。

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus_sd_1.yaml
- targets:
    - 127.0.0.1:9404
  labels:
    key1: value1
    key2: value2
```

此範例包含下列參數：

- `targets`— 由靜態組態湊集的目標。
- `labels`— 指派給從目標湊集之所有指標的標籤。

`ec2_sd_config` 允許從 Amazon EC2 執行個體擷取湊集目標。 以下是範例 `ec2_sd_config` 從 EC2 執行個體清單中湊集 Prometheus 指標。這些執行個體的 Prometheus 連接埠必須開啟代理程式執行所在的伺服器。 CloudWatch 執行 CloudWatch 代理程式之 EC2 執行個體的 IAM 角色必須包含`ec2:DescribeInstance`許可。例如，您可以將受管政策 AmazonEC2 附加ReadOnlyAccess到執行代理程式的執行個體 CloudWatch 。

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus.yaml
global:
  scrape_interval: 1m
  scrape_timeout: 10s
scrape_configs:
  - job_name: MY_JOB
    sample_limit: 10000
    ec2_sd_configs:
      - region: us-east-1
        port: 9404
```

```
        filters:
          - name: instance-id
            values:
              - i-98765432109876543
              - i-12345678901234567
```

此範例包含下列參數：

- `region`— 目標 EC2 執行個體所在的 AWS 區域。如果保留此為空白，會使用執行個體中繼資料中的區域。
- `port`— 從其中湊集指標的連接埠。
- `filters`— 用來篩選執行個體清單的選用篩選條件。本範例會根據 EC2 執行個體 ID 進行篩選。如需可篩選的更多條件，請參閱 DescribeInstances。

CloudWatch Prometheus 的代理程式組態

CloudWatch 代理程式組態檔案包含`logs`和下的`prometheus`區段`metrics_collected`。其包含以下參數。

- cluster_name— 指定要在日誌事件中新增為標籤的叢集名稱。此欄位為選用欄位。
- log_group_name— 為湊集的 Prometheus 指標指定日誌檔案群組名稱。
- prometheus_config_path— 指定 Prometheus 湊集組態檔案路徑。
- emf_processor— 指定內嵌指標格式處理器組態。如需內嵌指標格式的詳細資訊，請參閱在日誌中內嵌指標。

  `emf_processor` 區段可包含以下參數：

  - metric_declaration_dedup— 設定為 true，已啟用內嵌指標格式的重複資料刪除函數。
  - metric_namespace — 指定發出之測量結果的測量結果命名空間。 CloudWatch
  - metric_unit— 指定指標名稱:指標單位映射。如需支援的尺規單位的資訊，請參閱 MetricDatum。
  - metric_declaration— 是以要產生之內嵌指標格式來指定日誌陣列的區段。根據預設，CloudWatch代理程式從中匯入的每個 Prometheus 來源都有`metric_declaration`區段。這些區段各包括下列欄位：
    - `source_labels` 指定由 `label_matcher` 行檢查的標籤值。
    - `label_matcher` 是一個規則表達式，會檢查 `source_labels` 中列出的標籤值。符合的量度會啟用以包含在傳送至的內嵌量度格式中 CloudWatch。
    - `metric_selectors`是一個規則運算式，用來指定要收集並傳送至的測量結果 CloudWatch。

- dimensions是要作為每個選取量度之 CloudWatch 維度使用的標籤清單。

以下是 Prometheus 的 CloudWatch 代理程式組態範例。

```
{
   "logs":{
      "metrics_collected":{
         "prometheus":{
            "cluster_name":"prometheus-cluster",
            "log_group_name":"Prometheus",
            "prometheus_config_path":"C:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\
\prometheus.yaml",
            "emf_processor":{
               "metric_declaration_dedup":true,
               "metric_namespace":"CWAgent-Prometheus",
               "metric_unit":{
                  "jvm_threads_current": "Count",
                  "jvm_gc_collection_seconds_sum": "Milliseconds"
               },
               "metric_declaration":[
                  {
                     "source_labels":[
                        "job", "key2"
                     ],
                     "label_matcher":"MY_JOB;^value2",
                     "dimensions":[
                        [
                           "key1", "key2"
                        ],
                        [
                           "key2"
                        ]
                     ],
                     "metric_selectors":[
                        "^jvm_threads_current$",
                        "^jvm_gc_collection_seconds_sum$"
                     ]
                  }
               ]
            }
         }
      }
   }
```

```
}
```

之前的範例會在符合下列條件時，設定內嵌指標格式區段，以作為日誌事件傳送：

- 標籤的值 job 是 MY_JOB
- 標籤的值 key2 是 value2
- Prometheus 指標 jvm_threads_current 和 jvm_gc_collection_seconds_sum 包含 job 和 key2 標籤。

傳送的日誌事件包含下列反白顯示的區段。

```
{
    "CloudWatchMetrics": [
        {
            "Metrics": [
                {
                    "Unit": "Count",
                    "Name": "jvm_threads_current"
                },
                {
                    "Unit": "Milliseconds",
                    "Name": "jvm_gc_collection_seconds_sum"
                }
            ],
            "Dimensions": [
                [
                    "key1",
                    "key2"
                ],
                [
                    "key2"
                ]
            ],
            "Namespace": "CWAgent-Prometheus"
        }
    ],
    "ClusterName": "prometheus-cluster",
    "InstanceId": "i-0e45bd06f196096c8",
    "Timestamp": "1607966368109",
    "Version": "0",
    "host": "EC2AMAZ-PDDOIUM",
```

```
    "instance": "127.0.0.1:9404",
    "jvm_threads_current": 2,
    "jvm_gc_collection_seconds_sum": 0.006000000000000002,
    "prom_metric_type": "gauge",
    ...
}
```

範例：為 Prometheus 指標測試設定 Java/JMX 範例工作負載

JMX Exporter 是官方的 Prometheus 匯出工具，可以湊集 JMX mBeans 並將其公開為 Prometheus 指標。如需詳細資訊，請參閱 prometheus/jmx_exporter。

CloudWatch 代理程式可以從 EC2 執行個體上的 JMX 匯出程式，從 Java 虛擬機器 (JVM)、Hjava 和 Tomcat (卡塔利娜) 收集預先定義的 Prometheus 指標。

步驟 1：安裝 CloudWatch 代理程式

第一步是在 EC2 執行個體上安裝 CloudWatch 代理程式。如需說明，請參閱安裝 CloudWatch 代理程式。

步驟 2：啟動 Java/JMX 工作負載

下一步是啟動 Java/JMX 工作負載。

首先，從以下位置下載最新的 JMX Exporter jar 檔案：prometheus/jmx_exporter。

為您的範例應用程式使用 jar

以下章節中的範例命令使用 `SampleJavaApplication-1.0-SNAPSHOT.jar` 作為 jar 檔案。用您的應用程式的 jar 取代命令的這些部分。

準備 JMX Exporter 組態

`config.yaml` 檔案是 JMX Exporter 組態檔案。如需詳細資訊，請參閱 JMX Exporter 文件中的組態。

以下是 Java 和 Tomcat 的範例組態：

```
---
lowercaseOutputName: true
lowercaseOutputLabelNames: true

rules:
```

```
- pattern: 'java.lang<type=OperatingSystem><>(FreePhysicalMemorySize|
TotalPhysicalMemorySize|FreeSwapSpaceSize|TotalSwapSpaceSize|SystemCpuLoad|
ProcessCpuLoad|OpenFileDescriptorCount|AvailableProcessors)'
  name: java_lang_OperatingSystem_$1
  type: GAUGE

- pattern: 'java.lang<type=Threading><>(TotalStartedThreadCount|ThreadCount)'
  name: java_lang_threading_$1
  type: GAUGE

- pattern: 'Catalina<type=GlobalRequestProcessor, name=\"(\w+-\w+)-(\d+)\"><>(\w+)'
  name: catalina_globalrequestprocessor_$3_total
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina global $3
  type: COUNTER

- pattern: 'Catalina<j2eeType=Servlet, WebModule=//([-a-zA-Z0-9+&@#/%?=~_|!:.,;]*[-
a-zA-Z0-9+&@#/%=~_|]), name=([-a-zA-Z0-9+/$%~_-|!.]*), J2EEApplication=none,
 J2EEServer=none><>(requestCount|maxTime|processingTime|errorCount)'
  name: catalina_servlet_$3_total
  labels:
    module: "$1"
    servlet: "$2"
  help: Catalina servlet $3 total
  type: COUNTER

- pattern: 'Catalina<type=ThreadPool, name="(\w+-\w+)-(\d+)"><>(currentThreadCount|
currentThreadsBusy|keepAliveCount|pollerThreadCount|connectionCount)'
  name: catalina_threadpool_$3
  labels:
    port: "$2"
    protocol: "$1"
  help: Catalina threadpool $3
  type: GAUGE

- pattern: 'Catalina<type=Manager, host=([-a-zA-Z0-9+&@#/%?=~_|!:.,;]*[-a-zA-
Z0-9+&@#/%=~_|]), context=([-a-zA-Z0-9+/$%~_-|!.]*)><>(processingTime|sessionCounter|
rejectedSessions|expiredSessions)'
  name: catalina_session_$3_total
  labels:
    context: "$2"
    host: "$1"
```

```
  help: Catalina session $3 total
  type: COUNTER

- pattern: ".*"
```

使用 Prometheus 匯出程式啟動 Java 應用程式

啟動範例應用程式 這會將 Prometheus 指標發射至連接埠 9404。請務必以範例 java 應用程式的正確資訊取代進入點 com.gubupt.sample.app.App。

若使用的是 Linux，請輸入下列命令。

```
$ nohup java -javaagent:./jmx_prometheus_javaagent-0.14.0.jar=9404:./config.yaml -cp
  ./SampleJavaApplication-1.0-SNAPSHOT.jar com.gubupt.sample.app.App &
```

若使用的是 Windows，請輸入下列命令。

```
PS C:\> java -javaagent:.\jmx_prometheus_javaagent-0.14.0.jar=9404:.\config.yaml -cp  .
\SampleJavaApplication-1.0-SNAPSHOT.jar com.gubupt.sample.app.App
```

驗證 Prometheus 指標發射

驗證正在發射 Prometheus 指標。

若使用的是 Linux，請輸入下列命令。

```
$ curl localhost:9404
```

若使用的是 Windows，請輸入下列命令。

```
PS C:\> curl  http://localhost:9404
```

Linux 上的範例輸出：

```
StatusCode        : 200
StatusDescription : OK
Content           : # HELP jvm_classes_loaded The number of classes that are currently
 loaded in the JVM
                    # TYPE jvm_classes_loaded gauge
                    jvm_classes_loaded 2526.0
```

```
                    # HELP jvm_classes_loaded_total The total number of class...
RawContent        : HTTP/1.1 200 OK
                    Content-Length: 71908
                    Content-Type: text/plain; version=0.0.4; charset=utf-8
                    Date: Fri, 18 Dec 2020 16:38:10 GMT

                    # HELP jvm_classes_loaded The number of classes that are
 currentl...
Forms             : {}
Headers           : {[Content-Length, 71908], [Content-Type, text/plain; version=0.0.4;
 charset=utf-8], [Date, Fri, 18
                    Dec 2020 16:38:10 GMT]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : System.__ComObject
RawContentLength  : 71908
```

步驟 3：配置 CloudWatch 代理以抓取 Prometheus 指標

接下來，在代理程式設定檔中設定 Prometheus 抓取設定 CloudWatch 。

若要為 Java/JMx 範例設定 Prometheus 湊集組態

1. 設定 `file_sd_config` 和 `static_config` 的組態。

   若使用的是 Linux，請輸入下列命令。

   ```
   $ cat /opt/aws/amazon-cloudwatch-agent/var/prometheus.yaml
   global:
     scrape_interval: 1m
     scrape_timeout: 10s
   scrape_configs:
     - job_name: jmx
       sample_limit: 10000
       file_sd_configs:
         - files: [ "/opt/aws/amazon-cloudwatch-agent/var/prometheus_file_sd.yaml" ]
   ```

   若使用的是 Windows，請輸入下列命令。

   ```
   PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus.yaml
   global:
     scrape_interval: 1m
   ```

```
  scrape_timeout: 10s
scrape_configs:
  - job_name: jmx
    sample_limit: 10000
    file_sd_configs:
      - files: [ "C:\\ProgramData\\Amazon\\AmazonCloudWatchAgent\
\prometheus_file_sd.yaml" ]
```

2. 設定湊集目標組態。

   若使用的是 Linux，請輸入下列命令。

```
$ cat /opt/aws/amazon-cloudwatch-agent/var/prometheus_file_sd.yaml
- targets:
  - 127.0.0.1:9404
  labels:
    application: sample_java_app
    os: linux
```

   若使用的是 Windows，請輸入下列命令。

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus_file_sd.yaml
- targets:
  - 127.0.0.1:9404
  labels:
    application: sample_java_app
    os: windows
```

3. 透過 `ec2_sc_config` 設定 Prometheus 湊集組態。以正確的 EC2 執行個體 ID 取代 *your-ec2-instance-id*。

   若使用的是 Linux，請輸入下列命令。

```
$ cat .\prometheus.yaml
global:
  scrape_interval: 1m
  scrape_timeout: 10s
scrape_configs:
  - job_name: jmx
    sample_limit: 10000
    ec2_sd_configs:
      - region: us-east-1
```

```
        port: 9404
        filters:
          - name: instance-id
            values:
              - your-ec2-instance-id
```

若使用的是 Windows，請輸入下列命令。

```
PS C:\ProgramData\Amazon\AmazonCloudWatchAgent> cat prometheus_file_sd.yaml
- targets:
  - 127.0.0.1:9404
  labels:
    application: sample_java_app
    os: windows
```

4. 設定 CloudWatch 代理程式組態。首先，導覽至正確的目錄。若使用的是 Linux，則為 `/opt/aws/amazon-cloudwatch-agent/var/cwagent-config.json`。若使用的是 Windows，則為 `C:\ProgramData\Amazon\AmazonCloudWatchAgent\cwagent-config.json`。

   以下是定義了 Java/JHX Prometheus 指標的範例組態。請務必以正確的路徑取代 *path-to-Prometheus-Scrape-Configuration-file*。

```
{
  "agent": {
    "region": "us-east-1"
  },
  "logs": {
    "metrics_collected": {
      "prometheus": {
        "cluster_name": "my-cluster",
        "log_group_name": "prometheus-test",
        "prometheus_config_path": "path-to-Prometheus-Scrape-Configuration-file",
        "emf_processor": {
          "metric_declaration_dedup": true,
          "metric_namespace": "PrometheusTest",
          "metric_unit":{
            "jvm_threads_current": "Count",
            "jvm_classes_loaded": "Count",
            "java_lang_operatingsystem_freephysicalmemorysize": "Bytes",
            "catalina_manager_activesessions": "Count",
            "jvm_gc_collection_seconds_sum": "Seconds",
            "catalina_globalrequestprocessor_bytesreceived": "Bytes",
```

```
            "jvm_memory_bytes_used": "Bytes",
            "jvm_memory_pool_bytes_used": "Bytes"
          },
          "metric_declaration": [
            {
              "source_labels": ["job"],
              "label_matcher": "^jmx$",
              "dimensions": [["instance"]],
              "metric_selectors": [
                "^jvm_threads_current$",
                "^jvm_classes_loaded$",
                "^java_lang_operatingsystem_freephysicalmemorysize$",
                "^catalina_manager_activesessions$",
                "^jvm_gc_collection_seconds_sum$",
                "^catalina_globalrequestprocessor_bytesreceived$"
              ]
            },
            {
              "source_labels": ["job"],
              "label_matcher": "^jmx$",
              "dimensions": [["area"]],
              "metric_selectors": [
                "^jvm_memory_bytes_used$"
              ]
            },
            {
              "source_labels": ["job"],
              "label_matcher": "^jmx$",
              "dimensions": [["pool"]],
              "metric_selectors": [
                "^jvm_memory_pool_bytes_used$"
              ]
            }
          ]
        }
      }
    },
    "force_flush_interval": 5
  }
}
```

5. 輸入下列其中一個命令，以重新啟動 CloudWatch 代理程式。

   若使用的是 Linux，請輸入下列命令。

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
config -m ec2 -s -c file:/opt/aws/amazon-cloudwatch-agent/var/cwagent-config.json
```

若使用的是 Windows，請輸入下列命令。

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1"
 -a fetch-config -m ec2 -s -c file:C:\ProgramData\Amazon\AmazonCloudWatchAgent
\cwagent-config.json
```

檢視 Prometheus 指標和日誌

您現在可以檢視收集的 Java/JMX 指標。

若要檢視範例 JAW/JMX 工作負載的指標

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在執行叢集的區域中，在左側的導覽窗格中選擇 Metrics (指標)。找到PrometheusTest命名空間以查看指標。
3. 若要查看記 CloudWatch 錄事件，請在導覽窗格中選擇 [記錄群組]。事件位於日誌群組 prometheus-test。

使用 Amazon CloudWatch 可觀測 EKS 附加元件安裝 CloudWatch 代理程式

Amazon 可 CloudWatch 觀測性 EKS 附加元件會在 Amazon EKS 叢集上安裝 CloudWatch 代理程式和流暢位元代理程式，並且容器洞見增強了 Amazon EKS 和應用程式訊號預設啟用的可觀測性。CloudWatch 使用附加元件，您可從 Amazon EKS 叢集中收集基礎設施指標、應用程式效能遙測資料和容器日誌。

使用 Container Insights 搭配 Amazon EKS 的增強可觀測性，Container Insights 指標會按觀測，而不是存放或擷取的指標計費。對於 Application Signals，帳單以發往應用程式的傳入請求、來自應用程式的傳出請求以及每個已設定的服務水準目標 (SLO) 來計費。收到的每個傳入請求都會產生一個應用程式訊號，而發出的每個傳出請求也會產生一個應用程式訊號。每個 SLO 會在每個測量週期建立兩個應用程式訊號。如需有關 CloudWatch 定價的詳細資訊，請參閱 Amazon CloudWatch 定價。

Amazon EKS 附加元件可在 Amazon EKS 叢集中的 Linux 和 Windows 工作者節點上啟用容器洞見。若要在 Windows 上啟用容器洞見，您必須使用 Amazon EKS 附加元件 1.5.0 版或更新版本。目前，Amazon EKS 叢集中的視窗不支援應用程式訊號。

使用 Kubernetes 版本 1.23 或更新版本執行的 Amazon EKS 叢集上支援 Amazon 可 CloudWatch 觀測 EKS 附加元件。

安裝附加元件時，您還必須授予 IAM 許可，才能讓 CloudWatch 代理程式將指標、記錄和追蹤傳送至 CloudWatch。有兩種方式可以進行：

- 將政策連接至工作節點的 IAM 角色。此選項會授與工作者節點的權限，以便將遙測傳送至 CloudWatch。
- 針對代理程式 Pod 的服務帳戶使用 IAM 角色，並將政策連接至此角色。這僅適用於 Amazon EKS 叢集。此選項僅提 CloudWatch 供適當代理程式網繭的存取權。

選項 1：在工作節點上使用 IAM 許可進行安裝

若要使用此方法，請先輸入下列命令，將 CloudWatchAgentServerPolicyIAM 政策附加到您的背景工作節點。在此命令中，請以 Kubernetes 工作者節點所使用的 IAM 角色取*my-worker-node-role*代。

```
aws iam attach-role-policy \
--role-name my-worker-node-role \
--policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy
```

然後安裝 Amazon CloudWatch 觀測 EKS 附加組件。若要安裝附加元件，您可以使用 AWS CLI、主控台或 Terraform。 AWS CloudFormation

AWS CLI

若要使用安裝 Amazon CloudWatch 可觀測 EKS 附 AWS CLI 加元件

輸入以下命令。使用您叢集的名稱取代 *my-cluster-name*。

```
aws eks create-addon --addon-name amazon-cloudwatch-observability --cluster-name my-cluster-name
```

Amazon EKS console

若要使用 Amazon EKS 主控台新增 Amazon CloudWatch 可觀測 EKS 附加元件

1. 在 https://console.aws.amazon.com/eks/home#/clusters 開啟 Amazon EKS 主控台。
2. 在左側導覽窗格中選擇 Clusters (叢集)。
3. 選擇您要為其設定 Amazon 可 CloudWatch 觀測性 EKS 附加元件的叢集名稱。
4. 選擇附加元件索引標籤。
5. 選擇取得更多附加元件。
6. 在選取附加元件頁面上，執行下列動作：
   a. 在「Amazon EKS-外掛程式」區段中，選取「Amazon 可 CloudWatch 觀測性」核取方塊。
   b. 選擇下一步。
7. 在設定選取的附加元件設定頁面上，執行以下操作：
   a. 選取您要使用的版本。
   b. 針對選取 IAM 角色，選取從節點繼承
   c. (選用) 您可以展開選用組態設定。對於衝突解決方法，如果您選取覆寫，就可以使用 Amazon EKS 附加元件的設定來覆寫現有附加元件的一種或多種設定。若未啟用此選項，而且有設定與現有設定發生衝突，則操作會失敗。您可以使用產生的錯誤訊息對此衝突進行疑難排解。在選取此選項之前，請確認 Amazon EKS 附加元件未管理您需要自我管理的設定。
   d. 選擇下一步。
8. 在檢閱並新增頁面上，選擇建立。附加元件安裝完成後，您會看到已安裝的附加元件。

AWS CloudFormation

用於安裝 Amazon CloudWatch 可觀測 EKS 附 AWS CloudFormation 加元件

使用您叢集的名稱取代 *my-cluster-name*。如需詳細資訊，請參閱 AWS::EKS::Addon。

```
{
    "Resources": {
        "EKSAddOn": {
            "Type": "AWS::EKS::Addon",
```

```
            "Properties": {
                "AddonName": "amazon-cloudwatch-observability",
                "ClusterName": "my-cluster-name"
            }
        }
    }
}
```

Terraform

若要使用地形安裝 Amazon 可 CloudWatch 觀測 EKS 附加元件

使用您叢集的名稱取代 *my-cluster-name*。如需詳細資訊，請參閱 Resource: aws_eks_addon。

```
resource "aws_eks_addon" "example" {
  addon_name   = "amazon-cloudwatch-observability"
  cluster_name = "my-cluster-name"
}
```

選項 2：使用 IAM 服務帳戶角色進行安裝

使用此方法之前，請驗證下列先決條件：

- 您具備的功能性 Amazon EKS 叢集，內含在支援 Container Insights 的其中一個 AWS 區域 中附接的節點。如需支援區域的清單，請參閱 Container Insights。
- 您已安裝 `kubectl` 並設定叢集。如需詳細資訊，請參閱《Amazon EKS 使用者指南》中的安裝 kubectl。
- 您已安裝 `eksctl`。如需詳細資訊，請參閱 Amazon EKS 使用者指南中的安裝或更新 eksctl。

使用 IAM 服務帳戶角色安裝 Amazon CloudWatch 可觀測 EKS 附加元件

1. 如果叢集尚未提供 OpenID Connect (OIDC) 供應商，請輸入下列命令建立一個。如需詳細資訊，請參閱 Amazon EKS 使用者指南中的設定要擔任 IAM 角色的 Kubernetes 服務帳戶。

   ```
   eksctl utils associate-iam-oidc-provider --cluster my-cluster-name --approve
   ```

2. 輸入以下命令以建立附加CloudWatchAgentServerPolicy政策的 IAM 角色，並將代理程式服務帳戶設定為使用 OIDC 擔任該角色。以叢集*my-cluster-name*的名稱取代，並取代為您要*my-*

*service-account-role*與服務帳戶建立關聯的角色名稱。如果角色不存在，eksctl 會為您建立。

```
eksctl create iamserviceaccount \
  --name cloudwatch-agent \
  --namespace amazon-cloudwatch --cluster my-cluster-name \
  --role-name my-service-account-role \
  --attach-policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
  --role-only \
  --approve
```

3. 輸入下列命令來安裝附加元件。以叢集*my-cluster-name*的名稱取代，將 *111122223333* 取代為您的帳戶 ID，並取代為上一個步驟中建立*my-service-account-role*的 IAM 角色。

```
aws eks create-addon --addon-name amazon-cloudwatch-observability --cluster-
name my-cluster-name --service-account-role-arn arn:aws:iam::111122223333:role/my-
service-account-role
```

(選用) 額外組態

選擇不收集容器記錄

依預設，附加元件會使用 Fluent Bit 從所有網繭收集容器記錄檔，然後將記錄傳送至 CloudWatch 記錄檔。如需有關收集哪些日誌的資訊，請參閱 設定 Fluent Bit。

若要選擇退出容器記錄的收集，請在建立或更新附加元件時傳遞下列選項：

```
--configuration-values '{ "containerLogs": { "enabled": false } }'
```

選擇退出 GPU 指標收集

從 CloudWatch 代理程式的 1.300034.0 版開始，容器洞見預設會從 EKS 工作負載收集 NVIDIA GPU 指標。這些測量結果列在中的表格中GPU 指標。

您可以將 CloudWatch 代理程式組態檔案中的`accelerated_compute_metrics`選項設定為，選擇不收集 NVIDIA GPU 指標`false`。此選項位於`kubernetes`組 CloudWatch 態檔案中的`metrics_collected`區段中。以下是退出設定的範例。

```
{
  "agent": {
```

```
    "region": "us-east-1"
  },
  "logs": {
    "metrics_collected": {
      "emf": {
      },
      "kubernetes": {
        "enhanced_container_insights": true,
        "accelerated_compute_metrics": false
      }
    },
    "force_flush_interval": 5,
  }
}
```

使用自訂 CloudWatch 代理程式組態

若要使用 CloudWatch 代理程式收集其他指標、記錄檔或追蹤，您可以指定自訂組態，同時保持容器深入解析和 CloudWatch 應用程式訊號的啟用狀態。若要這麼做，請在建立或更新 EKS 附加元件時使用的進階組態的代理程式金鑰下，將代理程式組態檔案嵌入組態金鑰中。 CloudWatch 以下表示未提供任何其他組態時的預設代理程式組態。

> **Important**
>
> 您使用其他組態設定提供的任何自訂組態會覆寫代理程式使用的預設組態。請注意，不要意外停用預設啟用的功能，例如具有增強可觀察性的容器見解和 CloudWatch 應用程式訊號。在需要提供自訂代理程式組態的案例中，建議您使用下列預設組態作為基準，然後進行相應的修改。

```
--configuration-values '{
  "agent": {
    "config": {
      "logs": {
        "metrics_collected": {
          "app_signals": {},
          "kubernetes": {
            "enhanced_container_insights": true
          }
        }
      },
```

```
      "traces": {
        "traces_collected": {
          "app_signals": {}
        }
      }
    }

}'
```

下列範例顯示 Windows 上代理程式的預設 CloudWatch 代理程式組態。Windows 上的 CloudWatch 代理程式不支援自訂組態。

```
{
  "logs": {
    "metrics_collected": {
      "kubernetes": {
        "enhanced_container_insights": true
      },
    }
  }
}
```

管理許可 Webhook TLS 憑證

Amazon 可 CloudWatch 觀測性 EKS 附加元件會利用 Kubernetes 許可網路掛鉤來驗證和變更和`Instrumentation`自訂資源 (CR) 請求，`AmazonCloudWatchAgent`並在啟用應用程式訊號時選擇性地在叢集上使用 Kubernetes 網繭請求。 CloudWatch 在 Kubernetes 中，Webhook 需要 API 伺服器設定為信任的 TLS 憑證，以確保安全通訊。

根據預設，Amazon 可 CloudWatch 觀測性 EKS 附加元件會自動產生由此 CA 簽署的自我簽署 CA 和 TLS 憑證，以保護 API 伺服器與 Webhook 伺服器之間的通訊安全。此自動產生的憑證預設有效期為 10 年，且不會在到期時自動續訂。此外，每次升級或重新安裝附加元件時，都會重新產生 CA 服務包和憑證，進而重設到期日。如果您想要變更自動產生的憑證預設到期日，可以在建立或更新附加元件時使用下列其他組態。以天為單*`expiry-in-days`*位取代您想要的到期時間。

```
--configuration-values '{ "admissionWebhooks": { "autoGenerateCert":
 { "expiryDays": expiry-in-days } } }'
```

為了獲得更安全且功能豐富的憑證授權機構解決方案，附加元件可選擇支援 cert-manager，這是在 Kubernetes 中廣泛採用的 TLS 憑證管理解決方案，它可簡化獲取、更新、管理和使用這些憑證的程

序。它可確保憑證有效且最新，並嘗試在到期前的設定時間更新憑證。cert-manager 也有助於從各種支援來源 (包括 AWS Certificate Manager 私有憑證授權機構) 發行憑證。

建議您檢閱叢集上的 TLS 憑證管理最佳實務，並建議您選擇使用適合生產環境的 cert-manager。請注意，如果您選擇啟用憑證管理員來管理許可 Webhook TLS 憑證，則必須先在 Amazon EKS 叢集上預先安裝憑證管理員，然後再安裝 Amazon EKS 附加元件。 CloudWatch 請參閱 cert-manager 文件以進一步了解可用的安裝選項。安裝之後，您可以在建立或更新附加元件時，使用下列額外組態，選擇 cert-manager 來管理許可 Webhook TLS 憑證。

```
--configuration-values '{ "admissionWebhooks": { "certManager": { "enabled":
 true } } }'
```

本節中討論的進階組態預設會使用 SelfSigned發行者。

收集 Amazon EBS 磁碟區 ID

若您想要在效能日誌中收集 Amazon EBS 磁碟區 ID，必須將另一項政策新增至附接到工作節點或服務帳戶的 IAM 角色。新增以下內容作為內嵌政策。如需詳細資訊，請參閱新增和移除 IAM 身分許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:DescribeVolumes"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

疑難排解 Amazon CloudWatch 可觀測 EKS 附加元件

使用下列資訊協助疑難排解 Amazon CloudWatch 可觀測性 EKS 附加元件的問題。

更新和刪除 Amazon CloudWatch 可觀測 EKS 插件

如需更新或刪除 Amazon 可 CloudWatch 觀測性 EKS 附加元件的相關指示，請參閱管理 Amazon EKS 附加元件。使用 `amazon-cloudwatch-observability` 作為附加元件的名稱。

驗證 Amazon CloudWatch 可觀測性 EKS 附加元件所使用的 CloudWatch 代理程式版本

Amazon CloudWatch Observability EKS 附加元件會安裝類型的自訂資源，`AmazonCloudWatchAgent`以控制叢集上 CloudWatch代理程式精靈集的行為，包括正在使用的代理程式版本。 CloudWatch 可以透過輸入下列命令，取得叢集 u 上安裝的所有 `AmazonCloudWatchAgent` 自訂資源清單：

```
kubectl get amazoncloudwatchagent -A
```

在此命令的輸出中，您應該能夠檢查 CloudWatch 代理程式的版本。或者，您也可以描述 `amazoncloudwatchagent` 資源或叢集上執行的其中一個 `cloudwatch-agent-*` Pod，以檢查正在使用的映像。

管理附加元件 ConfigurationConflict 時處理

當您安裝或更新 Amazon CloudWatch Observability EKS 附加元件時，如果您注意到某種類型`Health IssueConfigurationConflict`的失敗，說明開頭為`Conflicts found when trying to apply. Will not continue due to resolve conflicts mode`，很可能是因為您已經在叢集上安 ClusterRoleBinding 裝了 CloudWatch 代理程式及其相關元件，例如 ServiceAccount、ClusterRole 和。當附加元件嘗試安裝 CloudWatch 代理程式及其相關元件時，如果偵測到內容有任何變更，依預設會無法安裝或更新，以避免覆寫叢集上的資源狀態。

如果您嘗試上載 Amazon CloudWatch Observability EKS 附加元件，但發現此失敗，建議您刪除先前安裝在叢集上的現有 CloudWatch 代理程式設定，然後安裝 EKS 附加元件。請務必備份您對原始 CloudWatch 代理程式設定所做的任何自訂 (例如自訂代理程式組態)，並在下次安裝或更新時將這些自訂提供給 Amazon CloudWatch Ob Servability EKS 附加元件。如果您之前已安裝 CloudWatch 代理程式以便上線至容器深入解析，請參閱以取得刪除容器見解的 CloudWatch 代理程式和流利位元詳細資訊。

或者，附加元件也支援衝突解決組態選項，該選項可指定 `OVERWRITE`。您可以使用此選項覆寫叢集上的衝突來繼續安裝或更新附加元件。如果您使用 Amazon EKS 主控台，則在建立或更新附加元件時選擇可選組態設定，可找到衝突解決方法。如果您使用的是 AWS CLI，您可以`--resolve-conflicts OVERWRITE`將指令提供給以建立或更新附加元件。

CloudWatch代理程式收集的測量結果

您可以在伺服器上安裝 CloudWatch 代理程式，從伺服器收集指標。您可以在 Amazon EC2 執行個體和內部部署伺服器，以及執行 Linux、Windows Server 或 macOS 的電腦上安裝代理程式。如果您在

Amazon EC2 執行個體上安裝代理程式，它會收集 Amazon EC2 執行個體上預設啟用的指標以外的其他指標。

如需在執行個體上安裝 CloudWatch 代理程式的詳細資訊，請參閱使用 CloudWatch 代理程式收集指標、記錄和追蹤。

CloudWatch 代理程式會直接收集此段落中討論的所有測量結果。

CloudWatch代理程式在 Windows 伺服器執行處理上收集的測量

在執行 Windows Server 的伺服器上，安裝 CloudWatch 代理程式可讓您收集與 Windows 效能監視器中計數器相關聯的度量。這些計數器的 CloudWatch 度量名稱是透過在物件名稱和計數器名稱之間加上空格來建立的。例如，`Processor`物件的`% Interrupt Time`計數器會`Processor % Interrupt Time`在中指定度量名稱 CloudWatch。如需有關 Windows 效能監控計數器的詳細資訊，請參閱 Microsoft Windows Server 文件。

雖然您可以在設定 CloudWatch 代理程式時指定不同的命名空間`CWAgent`，但代理程式收集的測量結果預設命名空間為。

CloudWatch代理程式在 Linux 和 macOS 執行個體上收集的指標

下表列出您可以在 Linux 伺服器和 macOS 電腦上使用 CloudWatch 代理程式收集的指標。

指標	描述
`cpu_time_active`	CPU 在任何容量中作用所花的時間。這個指標是以百分之一秒來測量。 單位：無
`cpu_time_guest`	CPU 執行訪客作業系統虛擬 CPU 所花的時間。這個指標是以百分之一秒來測量。 單位：無
`cpu_time_guest_nice`	CPU 為訪客作業系統執行虛擬 CPU 的時間量，此優先順序較低，且可能被其他程序中斷。這個指標是以百分之一秒來測量。 單位：無

指標	描述
cpu_time_idle	CPU 閒置所花費的時間。這個指標是以百分之一秒來測量。 單位：無
cpu_time_iowait	CPU 等待 I/O 操作完成所花費的時間。這個指標是以百分之一秒來測量。 單位：無
cpu_time_irq	CPU 服務中斷所花費的時間。這個指標是以百分之一秒來測量。 單位：無
cpu_time_nice	CPU 在使用者模式中處理低優先順序程序的時間量，可輕易被高優先順序的程序中斷。這個指標是以百分之一秒來測量。 單位：無
cpu_time_softirq	CPU 服務軟體中斷所花費的時間。這個指標是以百分之一秒來測量。 單位：無
cpu_time_steal	CPU 在遭竊的時間中所費的時間，也是在虛擬化環境中其他作業系統所花的時間。這個指標是以百分之一秒來測量。 單位：無
cpu_time_system	CPU 在系統模式中所花費的時間。這個指標是以百分之一秒來測量。 單位：無

指標	描述
cpu_time_user	CPU 在使用者模式中所花費的時間。這個指標是以百分之一秒來測量。 單位：無
cpu_usage_active	CPU 在任何容量中作用所花的時間百分比。 單位：百分比
cpu_usage_guest	CPU 執行訪客作業系統虛擬 CPU 的時間百分比。 單位：百分比
cpu_usage_guest_nice	CPU 為訪客作業系統執行虛擬 CPU 的時間百分比，其優先順序較低，且可能被其他程序中斷。 單位：百分比
cpu_usage_idle	CPU 閒置時間的百分比。 單位：百分比
cpu_usage_iowait	CPU 等待 I/O 操作完成的時間百分比。 單位：百分比
cpu_usage_irq	CPU 服務中斷的時間百分比。 單位：百分比
cpu_usage_nice	CPU 在使用者模式中處理低優先順序程序的時間百分比，可輕易被高優先順序的程序中斷。 單位：百分比
cpu_usage_softirq	CPU 服務軟體中斷的時間百分比。 單位：百分比

指標	描述
cpu_usage_steal	CPU 在遭奪取時間中的時間百分比，或是在虛擬化環境中其他作業系統內所花的時間。 單位：百分比
cpu_usage_system	CPU 在系統模式中的時間百分比。 單位：百分比
cpu_usage_user	CPU 在使用者模式中的時間百分比。 單位：百分比
disk_free	磁碟上的可用空間。 單位：位元組
disk_inodes_free	磁碟上可用的索引節點數量。 單位：計數
disk_inodes_total	磁碟上預留的所有索引節點數量。 單位：計數
disk_inodes_used	磁碟上使用的索引節點數量。 單位：計數
disk_total	磁碟上的總空間，包括已使用的和可用空間。 單位：位元組
disk_used	磁碟上的已使用空間。 單位：位元組
disk_used_percent	已使用總磁碟空間的百分比。 單位：百分比

指標	描述
diskio_iops_in_progress	已發至裝置驅動程式但尚未完成的 I/O 請求數量。 單位：計數
diskio_io_time	磁碟有 I/O 請求排入佇列的時間量。 單位：毫秒 此指標應使用的唯一統計資訊為 Sum。請勿選擇 Average。
diskio_reads	磁碟讀取操作的數量。 單位：計數 此指標應使用的唯一統計資訊為 Sum。請勿選擇 Average。
diskio_read_bytes	讀取自磁碟的位元組數目。 單位：位元組 此指標應使用的唯一統計資訊為 Sum。請勿選擇 Average。
diskio_read_time	讀取請求在磁碟上等待的時間。多個同時等待的讀取請求都會增加數量。例如，如果有 5 個請求平均都等待了 100 毫秒，則會報告 500 個。 單位：毫秒 此指標應使用的唯一統計資訊為 Sum。請勿選擇 Average。

指標	描述
diskio_writes	磁碟寫入操作的數量。 單位：計數 此指標應使用的唯一統計資訊為 Sum。請勿選擇 Average。
diskio_write_bytes	寫入至磁碟的位元組數目。 單位：位元組 此指標應使用的唯一統計資訊為 Sum。請勿選擇 Average。
diskio_write_time	寫入請求在磁碟上等待的時間。多個同時等待的寫入請求都會增加數量。例如，如果有 8 個請求平均都等待了 1000 毫秒，則會報告 8000 個。 單位：毫秒 此指標應使用的唯一統計資訊為 Sum。請勿選擇 Average。
ethtool_bw_in_allowance_exceeded	因傳入的彙總頻寬超過執行個體的上限而排入佇列及/或丟棄的封包數目。 只有在您已將此測量結果列在 CloudWatch 代理程式組態檔ethtool段落的子metrics_collected 段落時，才會收集此測量結果。如需詳細資訊，請參閱收集網路效能指標 單位：無

指標	描述
ethtool_bw_out_allowance_exceeded	因傳出的彙總頻寬超過執行個體的上限而排入佇列及/或丟棄的封包數目。 只有在您已將此測量結果列在 CloudWatch 代理程式組態檔ethtool段落的子metrics_collected 段落時，才會收集此測量結果。如需詳細資訊，請參閱收集網路效能指標 單位：無
ethtool_conntrack_allowance_exceeded	因為連線追蹤超過執行個體的上限且無法建立新的連線，而丟棄的封包數目。這可能會導致傳送或傳回執行個體流量的封包遺失。 只有在您已將此測量結果列在 CloudWatch 代理程式組態檔ethtool段落的子metrics_collected 段落時，才會收集此測量結果。如需詳細資訊，請參閱收集網路效能指標 單位：無
ethtool_linklocal_allowance_exceeded	由於本機代理伺服器服務的流量 PPS 超過網路介面上限而丟棄的封包數目。這會影響 DNS 服務、執行個體中繼資料服務和 Amazon Time Sync Service 的流量。 只有在您已將此測量結果列在 CloudWatch 代理程式組態檔ethtool段落的子metrics_collected 段落時，才會收集此測量結果。如需詳細資訊，請參閱收集網路效能指標 單位：無

指標	描述
ethtool_pps_allowance_exceeded	因雙向 PPS 超過執行個體的上限而排入佇列及/或丟棄的封包數目。 只有在您已將此測量結果列在 CloudWatch 代理程式組態檔ethtool段落的子metrics_collected 段落時，才會收集此測量結果。如需詳細資訊，請參閱收集網路效能指標。 單位：無
mem_active	在最後一個取樣期間以一些方式使用的記憶體數量。 單位：位元組
mem_available	可用的記憶體數量，可以立即指定到程序。 單位：位元組
mem_available_percent	可用的記憶體百分比，可以立即指定到程序。 單位：百分比
mem_buffered	用於緩衝區的記憶體數量。 單位：位元組
mem_cached	用於檔案快取的記憶體數量。 單位：位元組
mem_free	未使用的記憶體數量。 單位：位元組
mem_inactive	在最後一個抽象期間以某些方式而未使用的記憶體數量。 單位：位元組

指標	描述
mem_total	記憶體總量。 單位：位元組
mem_used	目前使用中的記憶體數量。 單位：位元組
mem_used_percent	目前使用中的記憶體百分比。 單位：百分比
net_bytes_recv	網路介面收到的位元組數目。 單位：位元組 此指標應使用的唯一統計資訊為 Sum。請勿選擇 Average。
net_bytes_sent	網路介面傳送的位元組數目。 單位：位元組 此指標應使用的唯一統計資訊為 Sum。請勿選擇 Average。
net_drop_in	此網路介面所接收且已捨棄的封包數量。 單位：計數 此指標應使用的唯一統計資訊為 Sum。請勿選擇 Average。
net_drop_out	此網路介面所傳輸且已捨棄的封包數量。 單位：計數 此指標應使用的唯一統計資訊為 Sum。請勿選擇 Average。

指標	描述
net_err_in	此網路介面偵測到的接收錯誤數量。 單位：計數 此指標應使用的唯一統計資訊為 Sum。請勿選擇 Average。
net_err_out	此網路介面偵測到的傳送錯誤數量。 單位：計數 此指標應使用的唯一統計資訊為 Sum。請勿選擇 Average。
net_packets_sent	此網路介面傳送的封包數目。 單位：計數 此指標應使用的唯一統計資訊為 Sum。請勿選擇 Average。
net_packets_recv	此網路介面接收的封包數目。 單位：計數 此指標應使用的唯一統計資訊為 Sum。請勿選擇 Average。
netstat_tcp_close	無狀態的 TCP 連接數目。 單位：計數
netstat_tcp_close_wait	從用戶端等待終止請求的 TCP 連線數量。 單位：計數
netstat_tcp_closing	具用戶端認可，等待終止請求的 TCP 連線數量。 單位：計數

指標	描述
netstat_tcp_established	已建立的 TCP 連線數量。 單位：計數
netstat_tcp_fin_wait1	在關閉連線程序期間，處於 FIN_WAIT1 狀態的 TCP 連線數量。 單位：計數
netstat_tcp_fin_wait2	在關閉連線程序期間，處於 FIN_WAIT2 狀態的 TCP 連線數量。 單位：計數
netstat_tcp_last_ack	等待用戶端傳送連線終止訊息認可的 TCP 連接數量。這是連線關閉前的最後一個狀態。 單位：計數
netstat_tcp_listen	目前偵聽連線請求的 TCP 連接埠數量。 單位：計數
netstat_tcp_none	非使用中用戶端的 TCP 連接數目。 單位：計數
netstat_tcp_syn_sent	在傳送連線請求後，等待符合連線請求的 TCP 連線數目。 單位：計數
netstat_tcp_syn_recv	在傳送並接收連線請求後，等待連線請求認可的 TCP 連線數目。 單位：計數

指標	描述
netstat_tcp_time_wait	目前正在等待以確定用戶端收到連線終止請求的 TCP 連接數目。 單位：計數
netstat_udp_socket	目前 UDP 連線數量。 單位：計數
processes_blocked	封鎖的程序數量。 單位：計數
processes_dead	無效的程序數量，在 Linux 上以 X 狀態代碼指出。 此指標不會在 macOS 電腦上收集。 單位：計數
processes_idle	閒置的程序數量 (睡眠超過 20 秒)。僅適用於 FreeBSD 執行個體。 單位：計數
processes_paging	正在分頁的程序數量，在 Linux 上以 W 狀態代碼指出。 此指標不會在 macOS 電腦上收集。 單位：計數
processes_running	執行中的程序數量，以 R 狀態代碼指出。 單位：計數
processes_sleeping	睡眠中的程序數量，以 S 狀態代碼指出。 單位：計數

指標	描述
processes_stopped	停止的程序數量，以 T 狀態代碼指出。 單位：計數
processes_total	在執行個體上的程序總數。 單位：計數
processes_total_threads	構成程序的執行緒總數。此指標只適用於 Linux 執行個體。 此指標不會在 macOS 電腦上收集。 單位：計數
processes_wait	正在分頁的程序數量，在 FreeBSD 執行個體上以 W 狀態代碼指出。此指標僅適用於 FreeBSD 執行個體，不適用於 Linux、Windows Server 或 macOS 執行個體。 單位：計數
processes_zombies	殭屍程序數量，以 Z 狀態代碼指出。 單位：計數
swap_free	未使用的切換空間數量。 單位：位元組
swap_used	目前使用中的交換空間數量。 單位：位元組
swap_used_percent	目前使用中的交換空間百分比。 單位：百分比

CloudWatch 代理程式收集的記憶體測量結果定義

CloudWatch 代理程式收集記憶體指標時，來源為主機的記憶體管理子系統。舉例來說，Linux 核心會在 `/proc` 中公開由作業系統維護的資料。對記憶體而言，該資料位於 `/proc/meminfo` 中。

每個不同的作業系統和架構對於各種程序所使用的資源都有不同的計算。如需詳細資訊，請參閱下列區段。

在每個收集間隔期間，每個執行處 CloudWatch 理上的代理程式都會收集執行處理資源，並計算該執行處理中執行的所有處理作業所使用的資源。此資訊會回報至 CloudWatch 量度。您可以在 CloudWatch 代理程式組態檔中設定收集間隔的長度。如需詳細資訊，請參閱 CloudWatch 代理程式組態檔案：代理程式。

下列清單說明如何定義 CloudWatch 代理程式收集的記憶體測量結果。

- 作用中記憶體 – 程序正在使用的記憶體。換句話說，就是目前正在執行的應用程式所使用的記憶體。
- 可用記憶體 – 系統不需要進入交換狀態，即可立即提供給程序的記憶體 (也稱為虛擬記憶體)。
- 緩衝記憶體 – 以不同速度和優先順序運作之硬體裝置或程式程序所共用的資料區域。
- 快取記憶體 – 可儲存會在 CPU 接下來可能需要的程式作業中重複使用的程式指示和資料。
- 閒置記憶體 – 尚未使用且隨時可用的記憶體。系統可在需要時完全自由使用。
- 非作用中記憶體 – 「最近」未存取的頁面。
- 記憶體總計 – 實際實體記憶體 RAM 的大小。
- 已使用記憶體 – 程式和程序目前正在使用的記憶體。

主題

- Linux：收集的指標和使用的計算
- macOS：收集的指標和使用的計算
- Windows：收集的指標
- 範例：在 Linux 上計算記憶體指標

Linux：收集的指標和使用的計算

收集的指標和單位：

- 作用中 (位元組)

- 可用 (位元組)
- 可用百分比 (百分比)
- 緩衝 (位元組)
- 快取 (位元組)
- 閒置 (位元組)
- 非作用中 (位元組)
- 總計 (位元組)
- 已使用 (位元組)
- 已使用百分比 (百分比)

已使用記憶體 = 記憶體總計 - 可用記憶體 - 快取記憶體 - 緩衝記憶體

記憶體總計 = 已使用記憶體 + 可用記憶體 + 快取記憶體 + 緩衝記憶體

macOS：收集的指標和使用的計算

收集的指標和單位：

- 作用中 (位元組)
- 可用 (位元組)
- 可用百分比 (百分比)
- 閒置 (位元組)
- 非作用中 (位元組)
- 總計 (位元組)
- 已使用 (位元組)
- 已使用百分比 (百分比)

可用記憶體 = 閒置記憶體 + 非作用中記憶體

已使用記憶體 = 記憶體總計 - 可用記憶體

記憶體總計 = 可用記憶體 - 已使用記憶體

Windows：收集的指標

以下列出在 Windows 主機上收集的指標。所有這些指標的 `Unit` 均為 `None`。

- 可用位元組
- 快取錯誤數/秒
- 頁面錯誤數/秒
- 頁數/秒

沒有用於 Windows 度量的計算，因為 CloudWatch 代理程式會剖析效能計數器的事件。

範例：在 Linux 上計算記憶體指標

舉例來說，假設在 Linux 主機上輸入 **cat /proc/meminfo** 命令會顯示下列結果：

```
MemTotal:       3824388 kB
MemFree:         462704 kB
MemAvailable:   2157328 kB
Buffers:         126268 kB
Cached:         1560520 kB
SReclaimable:    289080 kB>
```

在此範例中， CloudWatch 代理程式將收集下列值。 CloudWatch 代理程式收集和報告的所有值都以位元組為單位。

- mem_total：3916173312 位元組
- mem_available: 2 個位元組 (+ 快取記憶體) MemFree
- mem_free：473808896 位元組
- mem_cached：1893990400 位元組 (cached + SReclaimable
- mem_used：1419075584 位元組 (MemTotal – (MemFree + Buffers + (Cached + SReclaimable)))
- mem_buffered：129667072 位元組
- mem_available_percent：56.41%
- mem_used_percent：36.24% (mem_used / mem_total) * 100

CloudWatch代理程式的常見案例

下列各節概述如何完成 CloudWatch 代理程式的一般設定和自訂工作。

主題

- 以不同的使用者身分執行 CloudWatch 代理程式
- CloudWatch 代理程式處理稀疏記錄檔的方式
- 將自訂維度新增至 CloudWatch 代理程式收集的指標
- 多個 CloudWatch 代理程式組態檔
- 彙總或彙總代理程式收集的 CloudWatch 測量結果
- 使用 CloudWatch 代理程式收集高解析度量
- 將指標、日誌和追蹤傳送到不同帳戶
- 統一 CloudWatch 代理程式與舊版記錄代理程式之間的時間戳 CloudWatch 記差

以不同的使用者身分執行 CloudWatch 代理程式

在 Linux 伺服器上，依預設 CloudWatch 會以 root 使用者身分執行。若要讓代理程式以不同的使用者身分執行，請使用 CloudWatch 代理程式組態檔中`agent`區段中的`run_as_user`參數。此選項僅供 Linux 伺服器使用。

如果您已使用根使用者身分執行代理，並想要變更為使用不同的使用者身分，請使用下列程序之一。

若要在執行 CloudWatch Linux 的 EC2 執行個體上以不同使用者身分執行代理程式

1. 下載並安裝新的 CloudWatch 代理程式套件。如需詳細資訊，請參閱 下載 CloudWatch 代理程式套件。
2. 建立新的 Linux 使用者，或使用 RPM 或 DEB 檔案所建立名為 `cwagent` 的預設使用者。
3. 以下列方式之一提供此使用者的登入資料：
   - 如果檔案`.aws/credentials`存在於 root 使用者的主目錄中，您必須為要用來執行 CloudWatch 代理程式的使用者建立認證檔案。這個登入資料檔案會是 `/home/`*`username`*`/.aws/credentials`。然後，將 `common-config.toml` 中的 `shared_credential_file` 參數值設為登入資料檔案的路徑名稱。如需詳細資訊，請參閱 (選用) 修改代理或區域資訊的常見組態。
   - 如果檔案 `.aws/credentials` 不存在於根使用者的主目錄中，您可執行下列操作之一：
     - 為您要用來執行 CloudWatch 代理程式的使用者建立認證檔案。這個登入資料檔案會是 `/home/`*`username`*`/.aws/credentials`。然後，將 `common-config.toml` 中的 `shared_credential_file` 參數值設為登入資料檔案的路徑名稱。如需詳細資訊，請參閱 (選用) 修改代理或區域資訊的常見組態。

- 不要建立憑證檔案，而是將 IAM 角色連接到執行個體。代理會使用這個角色作為登入資料提供者。

4. 在 CloudWatch 代理程式組態檔中，在`agent`區段中新增下列行：

```
"run_as_user": "username"
```

視需要對組態檔案執行其他修改。如需詳細資訊，請參閱 建立 CloudWatch 代理程式組態檔

5. 提供使用者必要的許可。使用者必須擁有要收集之日誌檔的讀取 (r) 許可，而且必須擁有日誌檔路徑中每個目錄的 Execute (x) 許可。
6. 使用您剛才修改的組態檔案啟動代理。

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
config -m ec2 -s -c file:configuration-file-path
```

在執行 Linux 的內部部署伺服器上以不同的使用者身分執行 CloudWatch 代理程式

1. 下載並安裝新的 CloudWatch 代理程式套件。如需詳細資訊，請參閱 下載 CloudWatch 代理程式套件。
2. 建立新的 Linux 使用者，或使用 RPM 或 DEB 檔案所建立名為 `cwagent` 的預設使用者。
3. 將此使用者的登入資料存放在使用者可以存取的路徑，例如，`/home/username/.aws/credentials`。
4. 將 `common-config.toml` 中的 `shared_credential_file` 參數值設為登入資料檔案的路徑名稱。如需詳細資訊，請參閱 (選用) 修改代理或區域資訊的常見組態。
5. 在 CloudWatch 代理程式組態檔中，在`agent`區段中新增下列行：

```
"run_as_user": "username"
```

視需要對組態檔案執行其他修改。如需詳細資訊，請參閱 建立 CloudWatch 代理程式組態檔

6. 提供使用者必要的許可。使用者必須擁有要收集之日誌檔的讀取 (r) 許可，而且必須擁有日誌檔路徑中每個目錄的 Execute (x) 許可。
7. 使用您剛才修改的組態檔案啟動代理。

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
config -m ec2 -s -c file:configuration-file-path
```

CloudWatch 代理程式處理稀疏記錄檔的方式

稀疏檔案是同時具有空白區塊和實際內容的檔案。稀疏檔案透過將代表空白區塊的簡短資訊 (而不是組成區塊的實際空白位元組) 寫入磁碟，以更有效率的方式使用磁碟空間。這使稀疏檔案的實際大小往往比其表面上的大小小得多。

不過， CloudWatch 代理程式處理疏鬆檔案的方式與處理一般檔案的方式不同。當代理程式讀取稀疏檔案時，空白區塊會被視為填滿空位元組的「真實」區塊。因此， CloudWatch 代理程式會將與稀疏檔案明顯大小相同的位元組發佈到 CloudWatch。

將 CloudWatch 代理程式設定為發佈稀疏檔案可能會造成高於預期的 CloudWatch 成本，因此建議您不要這麼做。例如，`/var/logs/lastlog`在 Linux 中通常是一個非常稀疏的文件，我們建議您不要將其發佈到 CloudWatch.

將自訂維度新增至 CloudWatch 代理程式收集的指標

若要新增自訂維度，例如代理程式收集的標籤指標，請將 `append_dimensions` 欄位新增至列出這些指標的代理程式組態檔案區段。

例如，以下組態檔案範例區段將名為 `stackName` 的自訂維度與 `Prod` 值，新增至代理程式收集的 `cpu` 和 `disk` 指標。

```
"cpu":{
  "resources":[
    "*"
  ],
  "measurement":[
    "cpu_usage_guest",
    "cpu_usage_nice",
    "cpu_usage_idle"
  ],
  "totalcpu":false,
  "append_dimensions":{
    "stackName":"Prod"
  }
},
"disk":{
  "resources":[
    "/",
    "/tmp"
  ],
```

```
  "measurement":[
    "total",
    "used"
  ],
  "append_dimensions":{
    "stackName":"Prod"
  }
}
```

請記住，無論何時，只要變更代理程式組態檔案，您就必須重新啟動代理程式，才能使變更生效。

多個 CloudWatch 代理程式組態檔

在 Linux 伺服器和 Windows 伺服器上，您都可以將 CloudWatch 代理程式設定為使用多個組態檔。例如，您可以使用收集一組指標、日誌和追蹤的常見組態檔，而您一向會從基礎設施的所有伺服器收集它們。然後，您可以使用從某些應用程式或某些情況下收集指標的其他組態檔案。

若要設定，請先建立您要使用的組態檔案。將在相同伺服器上一起使用的任何組態檔案都必須具有不同的檔案名稱。您可以將組態檔案存放在伺服器或參數存放區中。

使用`fetch-config`選項啟動 CloudWatch 代理程式，並指定第一個組態檔。若要在執行中的代理程式附加第二個組態檔案，使用相同命令但請搭配 `append-config` 選項。會收集組態檔案中列出的所有指標、日誌和追蹤。下列範例命令說明使用組態儲存為檔案的這種情況。第一行使用 `infrastructure.json` 組態檔案來啟動代理程式，第二個行則附加 `app.json` 組態檔案。

下列範例指令適用於 Linux。

```
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2
 -s -c file:/tmp/infrastructure.json
```

```
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a append-config -m
 ec2 -s -c file:/tmp/app.json
```

下列範例指令適用於 Windows 伺服器。

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1"
 -a fetch-config -m ec2 -s -c file:"C:\Program Files\Amazon\AmazonCloudWatchAgent
\infrastructure.json"
```

```
& "C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1"
 -a append-config -m ec2 -s -c file:"C:\Program Files\Amazon\AmazonCloudWatchAgent
\app.json"
```

以下範例組態檔案說明如何使用此功能。第一個組態檔案用於基礎設施中的所有伺服器，第二個組態檔案僅收集來自特定應用程式的日誌，並附加到執行該應用程式的伺服器。

infrastructure.json

```
{
  "metrics": {
    "metrics_collected": {
      "cpu": {
        "resources": [
          "*"
        ],
        "measurement": [
          "usage_active"
        ],
        "totalcpu": true
      },
      "mem": {
         "measurement": [
           "used_percent"
        ]
      }
    }
  },
  "logs": {
    "logs_collected": {
      "files": {
        "collect_list": [
          {
            "file_path": "/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-
agent.log",
            "log_group_name": "amazon-cloudwatch-agent.log"
          },
          {
            "file_path": "/var/log/messages",
            "log_group_name": "/var/log/messages"
          }
        ]
      }
```

```
    }
  }
}
```

app.json

```
{
    "logs": {
        "logs_collected": {
            "files": {
                "collect_list": [
                    {
                        "file_path": "/app/app.log*",
                        "log_group_name": "/app/app.log"
                    }
                ]
            }
        }
    }
}
```

附加到組態的任何組態檔案都必須具有彼此不同的檔案名稱，且該名稱也必須和初始組態檔案不同。若您搭配具有和代理程式已正在使用組態檔案相同檔案名稱的組態檔案，使用 `append-config`，則附加命令會覆寫來自第一個組態檔案的資訊，而非附加到其中。即使具有相同檔案名稱的兩個組態檔案位於不同的檔案路徑上，也是如此。

上面的範例顯示了兩個組態檔案的使用，但可以附加到代理程式組態的組態檔案數沒有限制。您也可以混合使用位於伺服器的組態檔案和位於參數存放區的組態。

彙總或彙總代理程式收集的 CloudWatch 測量結果

若要彙整或累計代理程式收集的指標，請將 `aggregation_dimensions` 欄位新增至代理程式組態檔案中該指標的區段。

例如，以下組態檔案片段累計 `AutoScalingGroupName` 維度上的指標。每個 Auto Scaling 群組的所有執行個體的指標都將彙總並可整體檢視。

```
"metrics": {
  "cpu":{...}
  "disk":{...}
  "aggregation_dimensions" : [["AutoScalingGroupName"]]
```

```
}
```

除了以 Auto Scaling 群組名稱彙整之外，若要彙整每個 InstanceId 和 InstanceType 維度的組合，請新增以下內容。

```
"metrics": {
  "cpu":{...}
  "disk":{...}
  "aggregation_dimensions" : [["AutoScalingGroupName"], ["InstanceId", "InstanceType"]]
}
```

或者，若要將指標彙整至一個集合，請使用 []。

```
"metrics": {
  "cpu":{...}
  "disk":{...}
  "aggregation_dimensions" : [[]]
}
```

請記住，無論何時，只要變更代理程式組態檔案，您就必須重新啟動代理程式，才能使變更生效。

使用 CloudWatch 代理程式收集高解析度量

metrics_collection_interval 欄位指定收集指標的時間間隔 (以秒為單位)。藉由為此欄位指定小於 60 的值，就會以高解析度指標來收集指標。

例如，若您的指標都必須是高解析度，且每 10 秒收集一次，請為 metrics_collection_interval 區段下方的 agent 指定 10 作為值，以作為全域指標的收集間隔。

```
"agent": {
  "metrics_collection_interval": 10
}
```

或者，以下範例會設定每秒收集 cpu 指標一次，而所有其他指標則每分鐘收集一次。

```
"agent":{
  "metrics_collection_interval": 60
},
"metrics":{
  "metrics_collected":{
```

```
    "cpu":{
      "resources":[
        "*"
      ],
      "measurement":[
        "cpu_usage_guest"
      ],
      "totalcpu":false,
      "metrics_collection_interval": 1
    },
    "disk":{
      "resources":[
        "/",
        "/tmp"
      ],
      "measurement":[
        "total",
        "used"
      ]
    }
  }
}
```

請記住，無論何時，只要變更代理程式組態檔案，您就必須重新啟動代理程式，才能使變更生效。

將指標、日誌和追蹤傳送到不同帳戶

如果要讓 CloudWatch 代理程式將度量、記錄檔或追蹤傳送至其他帳戶，請在傳送伺服器上的代理程式組態檔中指定role_arn參數。role_arn 值會指定當將資料傳送給目標帳戶時，代理程式使用之目標帳戶的 IAM 角色。將指標或日誌交付給目標帳戶時，此角色可讓傳送帳戶擔任在目標帳戶中的對應角色。

您也可以在代理程式組態檔案中指定單獨的 role_arn 字串：一個用於傳送指標，一個用於傳送日誌，一個用於傳送追蹤。

以下範例是組態檔案 agent 區段的一部分，這部分會設定代理程式在將資料傳送給不同帳戶時使用 CrossAccountAgentRole。

```
{
  "agent": {
    "credentials": {
      "role_arn": "arn:aws:iam::123456789012:role/CrossAccountAgentRole"
```

```
    }
  },
  .....
}
```

或者，以下範例設定在傳送指標、日誌和追蹤時傳送帳戶所用的不同角色：

```
"metrics": {
    "credentials": {
     "role_arn": "RoleToSendMetrics"
    },
    "metrics_collected": {....
```

```
"logs": {
    "credentials": {
    "role_arn": "RoleToSendLogs"
    },
    ....
```

需要政策

當您在代理程式組態檔案中指定 `role_arn` 時，也必須確定傳送和目標帳戶的 IAM 角色擁有特定的政策。傳送和目標帳戶的角色都應該要有 `CloudWatchAgentServerPolicy`。如需將此政策指派給角色的詳細資訊，請參閱 建立 IAM 角色以搭配 Amazon EC2 執行個體上的 CloudWatch 代理程式使用。

傳送帳戶的角色也必須包含下列政策。編輯角色時，您可以將此政策新增至 IAM 主控台中的 Permissions (許可) 標籤。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": [
                "arn:aws:iam::target-account-ID:role/agent-role-in-target-account"
```

```
            ]
        }
    ]
}
```

目標帳戶中的角色必須包含以下政策，才能辨識傳送帳戶所使用的 IAM 角色。編輯角色時，您可以將此政策新增至 IAM 主控台中的 Trust relationships (信任關係) 標籤。您在目標帳戶中新增此政策的角色，就是您在 建立 IAM 角色和使用者以搭配 CloudWatch 代理程式使用 中建立的角色。這個角色是在傳送帳戶使用的政策中以 *agent-role-in-target-account* 指定的角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::sending-account-ID:role/role-in-sender-account"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

統一 CloudWatch 代理程式與舊版記錄代理程式之間的時間戳 CloudWatch 記差

與先前的 Logs CloudWatch 代理程式相比，代理程式支援時間戳 CloudWatch 記格式的一組不同符號。這些差異如下表所示。

兩種代理程式都支援的符號	僅由統一 CloudWatch 代理程式支援的符號	僅舊版 CloudWatch 記錄代理程式支援的符號
%A、%a、%b、%B、%d、%f、%H、%I、%m、%M、%p、%S、%y、%Y、%Z、%z	%-d、%-I、%-m、%-M、%-S	%c、%j、%U、%W、%w

如需有關新 CloudWatch 代理程式支援之符號含義的詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的 CloudWatch 代理程式組態檔：記錄一節。如需 CloudWatch 日誌代理程式支援之符號的相關資訊，請參閱 Amazon Lo CloudWatch gs 使用者指南中的代理程式組態檔。

疑難排解 CloudWatch 代理

使用下列資訊可協助疑難排解 CloudWatch 代理程式的問題。

主題

- CloudWatch 代理程式命令列參數
- 使用執行命令安裝 CloudWatch 代理程式失敗
- CloudWatch 代理程式無法啟動
- 確認 CloudWatch 代理程式正在執行
- CloudWatch 代理程式無法啟動，錯誤提到 Amazon EC2 區域
- CloudWatch 代理程式無法在 Windows 伺服器上啟動
- 指標在哪裡？
- CloudWatch 代理程式需要很長時間才能在容器中執行，或記錄躍點限制錯誤
- 我已更新代理程式設定，但在 CloudWatch 主控台中看不到新的指標或記錄檔
- CloudWatch 代理程式檔案和位置
- 尋找 CloudWatch 代理程式版本的資訊
- CloudWatch代理程式產生的記錄檔
- 停止並重新啟動 CloudWatch 代理程式

CloudWatch 代理程式命令列參數

若要查看代理程式支援的完整參數清單，請在已安裝該 CloudWatch 代理程式的電腦的命令列中輸入下列指令：

```
amazon-cloudwatch-agent-ctl -help
```

使用執行命令安裝 CloudWatch 代理程式失敗

若要使用 Systems Manager 執行命令來安裝 CloudWatch 代理程式，目標伺服器上的 SSM 代理程式必須是 2.2.93.0 或更新版本。如果您的 SSM Agent 不是正確的版本，您可能會看到錯誤，其中包含以下訊息：

```
no latest version found for package AmazonCloudWatchAgent on platform linux
```

```
failed to download installation package reliably
```

如需有關更新 SSM Agent 版本的資訊，請參閱《AWS Systems Manager 使用者指南》中的安裝和設定 SSM Agent。

CloudWatch 代理程式無法啟動

如果 CloudWatch 代理程式無法啟動，您的組態可能有問題。組態資訊會記錄在 `configuration-validation.log` 檔案中。此檔案在 Linux 伺服器上位於 `/opt/aws/amazon-cloudwatch-agent/logs/configuration-validation.log`，在執行 Windows Server 的伺服器上則位於 `$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\configuration-validation.log`。

確認 CloudWatch 代理程式正在執行

您可以查詢代理程式，找出 CloudWatch 代理程式是否正在執行中或已停止。您可以使用 AWS Systems Manager 從遠端執行此操作。您也可以使用命令列，但只會檢查本機伺服器。

使用執行命令查詢 CloudWatch 代理程式的狀態

1. 開啟 Systems Manager 主控台，網址為 https://console.aws.amazon.com/systems-manager/。
2. 在導覽窗格中，選擇 執行命令。

   -或-

   如果 AWS Systems Manager 首頁開啟，請向下捲動並選擇「瀏覽執行命令」。
3. 選擇 執行命令。
4. 在 [指令] 文件清單中，選擇 [AmazonCloudWatch-] 旁邊的按鈕ManageAgent。
5. 在 Action (動作) 清單中，選擇 status (狀態)。
6. 針對 Optional Configuration Source (選用組態來源)，選擇 default (預設) 並將 Optional Configuration Location (選用組態位置) 維持空白。
7. 在 Target (目標) 區域，選擇要檢查的執行個體。
8. 選擇執行。

如果代理程式正在執行，輸出會如下所示。

```
{
       "status": "running",
       "starttime": "2017-12-12T18:41:18",
       "version": "1.73.4"
}
```

如果代理程式已停止，"status" 欄位將顯示 "stopped"。

使用命令列在本機查詢 CloudWatch 代理程式的狀態

- 在 Linux 伺服器上，輸入以下資訊：

  ```
  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -m ec2 -a
   status
  ```

  在執行 Windows 伺服器的伺服器上，以系統管理員身分輸入下列內 PowerShell 容：

  ```
  & $Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1 -m
   ec2 -a status
  ```

CloudWatch 代理程式無法啟動，錯誤提到 Amazon EC2 區域

若代理程式未啟動，而且錯誤訊息提及 Amazon EC2 區域端點，則您可能已將代理程式設定為需要存取 Amazon EC2 端點，卻沒有授予該存取權限。

例如，您若指定代理程式組態檔案中 append_dimensions 參數的值 (取決於 Amazon EC2 中繼資料)，而且您使用代理，那麼您必須確保伺服器可以存取 Amazon EC2 的端點。如需有關這些端點的詳細資訊，請參閱 Amazon Web Services 一般參考中的 Amazon Elastic Compute Cloud (Amazon EC2)。

CloudWatch 代理程式無法在 Windows 伺服器上啟動

若使用的是 Windows Server，您可能會看到下列錯誤：

```
Start-Service : Service 'Amazon CloudWatch Agent (AmazonCloudWatchAgent)' cannot be
 started due to the following
error: Cannot start service AmazonCloudWatchAgent on computer '.'.
At C:\Program Files\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1:113
 char:12
+     $svc | Start-Service
```

```
+             ~~~~~~~~~~~~~~
    + CategoryInfo          : OpenError:
 (System.ServiceProcess.ServiceController:ServiceController) [Start-Service],
   ServiceCommandException
    + FullyQualifiedErrorId :
 CouldNotStartService,Microsoft.PowerShell.Commands.StartServiceCommand
```

若要修正此問題，請先確定伺服器服務正在執行中。如果代理程式嘗試在未執行伺服器服務時啟動，就可以看到此錯誤。

如果伺服器服務已在執行中，可能是下列問題。在某些 Windows 伺服器安裝上， CloudWatch 代理程式需要 30 秒以上的時間才能啟動。由於 Windows Server 依預設只允許服務有 30 秒的時間進行啟動，因此這會導致代理程式失敗，並出現類似以下的錯誤：

若要修正此問題，請增加服務逾時值。如需詳細資訊，請參閱服務未啟動，且事件 7000 和 7011 已記錄在 Windows 事件記錄檔中。

指標在哪裡？

如果 CloudWatch 代理程式一直在執行，但您無法在 AWS Management Console 或中找到其收集的指標 AWS CLI，請確認您使用的是正確的命名空間。根據預設，代理程式所收集指標的命名空間為 `CWAgent`。您可以使用代理程式組態檔案中 `metrics` 區段的 `namespace` 欄位來自訂此命名空間。如果看不到您預期的指標，請檢查組態檔案以確認所使用的命名空間。

當您第一次下載 CloudWatch 代理程式套件時，代理程式組態檔為`amazon-cloudwatch-agent.json`。此檔案位於您當初執行設定精靈的目錄，或者您可能已將它移到不同的目錄。如果您使用設定精靈，精靈輸出的代理程式組態檔案名為 `config.json`。如需有關組態檔案的詳細資訊，包括 `namespace` 欄位，請參閱 CloudWatch 代理程式組態檔：測量結果段。

CloudWatch 代理程式需要很長時間才能在容器中執行，或記錄躍點限制錯誤

當您以容器服務的形式執行 CloudWatch 代理程式，並希望將 Amazon EC2 指標維度新增至代理程式收集的所有指標時，您可能會在代理程式的 v1.247354.0 版中看到下列錯誤：

```
2022-06-07T03:36:11Z E! [processors.ec2tagger] ec2tagger: Unable to retrieve Instance
 Metadata Tags. This plugin must only be used on an EC2 instance.
2022-06-07T03:36:11Z E! [processors.ec2tagger] ec2tagger: Please increase hop limit
 to 2 by following this document https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/
configuring-instance-metadata-options.html#configuring-IMDS-existing-instances.
2022-06-07T03:36:11Z E! [telegraf] Error running agent: could not initialize processor
 ec2tagger: EC2MetadataRequestError: failed to get EC2 instance identity document
```

```
caused by: EC2MetadataError: failed to make EC2Metadata request
        status code: 401, request id:
caused by:
```

如果代理程式試圖在沒有適當跳轉限制的情況下從容器內的 IMDSv2 取得中繼資料，您可能會看到此錯誤。在比 v1.247354.0 更早的代理程式版本中，您可能會遇見此問題但看不到此日誌訊息。

若要解決此問題，請依照設定執行個體中繼資料選項中的指示將跳轉限制增加至 2。

我已更新代理程式設定，但在 CloudWatch 主控台中看不到新的指標或記錄檔

如果您更新 CloudWatch 代理程式組態檔，下次啟動代理程式時，您必須使用此選**fetch-config**項。例如，如果您將更新的檔案存放在本機電腦上，請輸入下列命令：

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -
s -m ec2 -c file:configuration-file-path
```

CloudWatch 代理程式檔案和位置

下表列出 CloudWatch 代理程式所安裝並搭配使用的檔案，以及它們在執行 Linux 或 Windows Server 的伺服器上的位置。

檔案	Linux 位置	Windows Server 位置
控制啟動、停用和重新啟動代理程式的控制指令碼。	`/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl` 或 `/usr/bin/amazon-cloudwatch-agent-ctl`	`$Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1`
代理程式寫入的日誌檔。您可能需要在聯繫時附加此信息 AWS Support。	`/opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log` 或 `/var/log/amazon/amazon-cloudwatch-agent/`	`$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\amazon-cloudwatch-agent.log`

檔案	Linux 位置	Windows Server 位置
	amazon-cloudwatch-agent.log	
代理程式組態驗證檔案。	/opt/aws/amazon-cloudwatch-agent/logs/configuration-validation.log 或 /var/log/amazon/amazon-cloudwatch-agent/configuration-validation.log	$Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\configuration-validation.log
在精靈建立它後用於立即設定代理程式的 JSON 檔案。如需詳細資訊，請參閱 建立 CloudWatch 代理程式組態檔。	/opt/aws/amazon-cloudwatch-agent/bin/config.json	$Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\config.json
若此組態檔案已從參數存放區下載，則為用於設定代理程式的 JSON 檔案。	/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json 或 /etc/amazon/amazon-cloudwatch-agent/amazon-cloudwatch-agent.json	$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json
TOML 檔案，用於指定代理程式使用的區域和登入資料資訊，將會覆寫系統預設值。	/opt/aws/amazon-cloudwatch-agent/etc/common-config.toml 或 /etc/amazon/amazon-cloudwatch-agent/common-config.toml	$Env:ProgramData\Amazon\AmazonCloudWatchAgent\common-config.toml

檔案	Linux 位置	Windows Server 位置
包含轉換後的 JSON 組態檔案內容的 TOML 檔案。amazon-cloudwatch-agent-ctl 指令碼會產生此檔案。使用者不應直接修改此檔案。這對於驗證 JSON 到 TOML 的轉換是否成功非常有用。	/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.toml 或 /etc/amazon/amazon-cloudwatch-agent/amazon-cloudwatch-agent.toml	$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.toml
包含轉換後的 JSON 組態檔案內容的 YAML 檔案。amazon-cloudwatch-agent-ctl 指令碼會產生此檔案。不應直接修改此檔案。此檔案對於驗證 JSON 到 TOML 的轉換是否成功非常有用。	/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.yaml or /etc/amazon/amazon-cloudwatch-agent/amazon-cloudwatch-agent.yaml	$Env:ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.yaml

尋找 CloudWatch 代理程式版本的資訊

若要尋找 Linux 伺服器上 CloudWatch 代理程式的版本號碼，請輸入下列命令：

```
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a status
```

若要尋找 Windows 伺服器上 CloudWatch 代理程式的版本號碼，請輸入下列命令：

```
& $Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1 -m ec2
 -a status
```

> **Note**
>
> 使用此命令是尋找 CloudWatch 代理程式版本的正確方法。如果您使用控制面板中的程式和功能，您會看到不正確的版本編號。

您也可以下載有關代理程式最新變更的 README 檔案，以及指出目前可供下載之版本編號的檔案。這些檔案位於下列位置：

- https://amazoncloudwatch-agent.s3.amazonaws.com/info/latest/RELEASE_NOTES 或 https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/info/latest/RELEASE_NOTES
- https://amazoncloudwatch-agent.s3.amazonaws.com/info/latest/CWAGENT_VERSION 或 https://amazoncloudwatch-agent-*region*.s3.*region*.amazonaws.com/info/latest/CWAGENT_VERSION

CloudWatch代理程式產生的記錄檔

代理程式會在執行時產生日誌。此日誌包含故障診斷資訊。此日誌是 amazon-cloudwatch-agent.log 檔案。此檔案在 Linux 伺服器上位於 /opt/aws/amazon-cloudwatch-agent/logs/amazon-cloudwatch-agent.log，在執行 Windows Server 的伺服器上則位於 $Env:ProgramData\Amazon\AmazonCloudWatchAgent\Logs\amazon-cloudwatch-agent.log。

您可以設定代理程式在 amazon-cloudwatch-agent.log 檔案中記錄其他詳細資訊。在代理程式組態檔的agent區段中，將debug欄位設定為true，然後重新設定並重新啟動 CloudWatch 代理程式。若要停用此額外資訊的記錄，請將 debug 欄位設為 false。然後，重新設定並重新啟動代理程式。如需詳細資訊，請參閱 手動建立或編輯 CloudWatch 代理程式組態檔。

在代理程式版本 1.247350.0 及更新版本中，您可以選擇性地將 CloudWatch 代理程式組態檔agent區段中的aws_sdk_log_level欄位設定為下列一或多個選項。如有多個選項，請使用 | 字元進行分隔。

- LogDebug
- LogDebugWithSigning
- LogDebugWithHTTPBody
- LogDebugRequestRetries
- LogDebugWithEventStreamBody

如需這些選項的更多資訊，請參閱LogLevelType。

停止並重新啟動 CloudWatch 代理程式

您可以使用 AWS Systems Manager 或命令列手動停止 CloudWatch 代理程式。

使用執行命令停止 CloudWatch 代理程式

1. 開啟 Systems Manager 主控台，網址為 https://console.aws.amazon.com/systems-manager/。
2. 在導覽窗格中，選擇 執行命令。

   -或-

   如果 AWS Systems Manager 首頁開啟，請向下捲動並選擇「瀏覽執行命令」。
3. 選擇 執行命令。
4. 在「命令」文件清單中，選擇 AmazonCloudWatch-ManageAgent。
5. 在 [目標] 區域中，選擇您安裝 CloudWatch 代理程式的執行個體。
6. 在 Action (動作) 清單中，選擇 stop (停止)。
7. 將 Optional Configuration Source (選用組態來源) 和 Optional Configuration Location (選用組態位置) 維持空白。
8. 選擇執行。

使用命令列在本機停止 CloudWatch 代理程式

- 在 Linux 伺服器上，輸入以下資訊：

  ```
  sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -m ec2 -a
   stop
  ```

  在執行 Windows 伺服器的伺服器上，以系統管理員身分輸入下列內 PowerShell 容：

  ```
  & $Env:ProgramFiles\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1 -m
   ec2 -a stop
  ```

若要重新啟動代理程式，請按照 啟動 CloudWatch 代理程式 中的說明操作。

在日誌中內嵌指標

內 CloudWatch 嵌指標格式可讓您以寫入記錄檔的日誌形式非同步產生自訂指標。 CloudWatch 您可以在詳細的日誌事件數據旁嵌入自定義指標，并 CloudWatch 自動提取自定義指標，以便您可以對其進行可視化和警報，以實時檢測事件。此外，您可以使用 CloudWatch Logs Insights 查詢與擷取指標相關聯的詳細記錄事件，以提供操作事件根本原因的深入見解。

內嵌指標格式可協助您從暫時性資源 (例如 Lambda 函數和容器) 產生可行的自訂指標。透過使用內嵌指標格式從這些暫時性資源傳送日誌，您現在可以輕鬆建立自訂指標，而無需檢測或維護單獨的程式碼，同時取得強大的日誌資料分析功能。

無需設定即可使用內嵌指標格式。您可以按照嵌入式指標格式規範來構建日誌，或者使用我們的客戶端庫生成它們，然後使用 PutLogEvents API 或CloudWatch 代理將其發送到 CloudWatch 日誌。

擷取日誌和存檔日誌，以及產生自訂指標時會產生費用。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

> **Note**
>
> 請在設定您的指標擷取時注意，因為這會影響您的自訂指標用量和對應的帳單。如果您不小心建立了以高基數維度 (例如 `requestId`) 為基礎的指標，根據設計，內嵌指標格式會建立自訂指標，對應至每個獨特的維度組合。如需詳細資訊，請參閱維度。

主題

- 使用內嵌指標格式發布日誌
- 在主控台中檢視您的指標和日誌
- 在以內嵌指標格式建立的指標上設定警示

使用內嵌指標格式發布日誌

您可以使用下列方法產生內嵌指標格式日誌：

- 使用開放原始碼的用戶端程式庫產生和傳送日誌。
- 使用內嵌指標格式規格手動產生記錄檔，然後使用CloudWatch 代理程式或 PutLogEvents API 傳送記錄檔。

主題

- 使用用戶端程式庫建立內嵌指標格式日誌
- 規格：內嵌指標格式
- 使用 PutLogEvents API 傳送手動建立的內嵌指標格式記錄
- 使用 CloudWatch 代理程式傳送內嵌的量度格式記錄
- 將嵌入式度量格式與 AWS 發行版搭配使用 OpenTelemetry

使用用戶端程式庫建立內嵌指標格式日誌

Amazon 提供開放原始碼的用戶端程式庫，可讓您用來建立內嵌指標格式日誌。目前這些程式庫適用於下列清單中的語言。可以在我們用戶端程式庫的 /examples 中找到不同設定的完整範例。

程式庫和如何使用這些程式庫的說明可在 GitHub 上找到。使用下列連結。

- Node.Js

> **Note**
>
> 若是 Node.js，需要 4.1.1+、3.0.2+、2.0.7+ 版本，才能搭配使用 Lambda JSON 日誌格式。在此類 Lambda 環境中使用舊版會導致指標遺失。
> 如需詳細資訊，請參閱存取的 Amazon CloudWatch 日誌 AWS Lambda。

- Python
- Java
- C#

用戶端程式庫的目的是要使用 CloudWatch 代理程式開箱即用。產生的內嵌度量格式記錄檔會傳送至 CloudWatch 代理程式，然後為您彙總並發佈至 CloudWatch 記錄檔。

> **Note**
>
> 使用 Lambda 時，不需要代理程式即可將記錄傳送至 CloudWatch。記錄到標準輸出的任何內容都會透過 Lambda 記錄代理程式傳送至 CloudWatch 日誌。

規格：內嵌指標格式

內 CloudWatch 嵌度量格式是 JSON 規格，用於指示 CloudWatch 記錄檔自動擷取內嵌在結構化記錄事件中的指標值。您可以使 CloudWatch 用在擷取的量度值上繪製和建立警示。

內嵌指標格式規格慣例

本格式規格中的關鍵字「必須」、「禁止」、「必要」、「可」、「不可」、「應」、「不應」、「建議」、「能」和「選用」應按照 Key Words RFC2119 中的說明解釋。

術語「JSON」，「JSON 文本」，「JSON 值」，「成員」，「元素」，「對象」，「數組」，「數字」，「字符串」，「布爾值」，「真」，「假」和「空」將被解釋為JavaScript 對象表示法 RFC8259 中定義。

> **Note**
>
> 如果您打算在使用內嵌指標格式建立的指標上建立警示，請參閱在以內嵌指標格式建立的指標上設定警示以取得相關建議。

內嵌指標格式文件結構

本節說明內嵌指標格式文件的結構。嵌入式度量格式文件是在JavaScript 物件標記 RFC8259 中定義的。

除非另有說明，否則此規格定義的物件「禁止」包含任何其他成員。未經此規格辨識的成員「必須」遭到忽略。此規格中定義的成員區分大小寫。

內嵌的量度格式受到與標準 CloudWatch 記錄事件相同的限制，且大小上限為 256 KB。

借助內嵌指標格式，您可以透過在您帳戶的 `AWS/Logs` 命名空間中發佈的指標追蹤 EMF 日誌的處理進度。這些指標可用於追蹤從 EMF 產生指標失敗的情形，以及失敗是否因剖析或驗證造成。如需詳細資訊，請參閱使用 CloudWatch 指標監視。

根節點

該 LogEvent 消息必須是有效的 JSON 對象，在消 LogEvent 息字符串的開頭或結尾沒有其他數據。如需 LogEvent 結構的詳細資訊，請參閱InputLogEvent。

內嵌指標格式文件「必須」在根節點上包含以下最上層成員。這是一個 中繼資料物件 物件。

```
{
 "_aws": {
    "CloudWatchMetrics": [ ... ]
  }
}
```

根節點必須包含所有由 MetricDirective 物件 中參考定義的 目標成員 成員。

根節點「能」包含任何其他未包含在以上需求內的成員。這些成員的值「必須」是有效的 JSON 類型。

中繼資料物件

成_aws員可用來表示有關承載的中繼資料，通知下游服務應如何處理. LogEvent 其值「必須」是物件且「必須」包含下列成員：

- CloudWatchMetrics— MetricDirective 物件 用於指示 CloudWatch 從的根節點擷取度量的陣列。LogEvent

  ```
  {
    "_aws": {
      "CloudWatchMetrics": [ ... ]
    }
  }
  ```

- 時間戳記— 數字，表示用於從事件擷取的指標的時間戳記。其值「必須」表示為 UTC 時間 1970 年 1 月 1 日上午 00:00:00 之後的毫秒數。

  ```
  {
    "_aws": {
      "Timestamp": 1559748430481
    }
  }
  ```

MetricDirective 物件

MetricDirective 物件會指示下游服務 LogEvent 包含將擷取並發佈至 CloudWatch的量度。MetricDirectives 「必須」包含下列成員：

- 命名空間 — 字串，代表測量結果的 CloudWatch 命名空間。

- 維度— A DimensionSet 陣列。
- 指標— MetricDefinition 物件的陣列。此陣列不得包含超過 100 個 MetricDefinition 物件。

DimensionSet 陣列

A DimensionSet 是字串陣列，其中包含將套用至文件中所有度量的維度索引鍵。此陣列中的值「必須」也是根節點上的成員，稱為 目標成員。

A 不 DimensionSet 得包含超過 30 個維度索引鍵。A DimensionSet 可能是空的。

目標成員「必須」擁有字串值。此值不得包含超過 1,024 個字元。目標成員定義了將作為指標身分一部分發佈的維度。每個 DimensionSet 使用的項目都會在中建立新量度 CloudWatch。如需維度的詳細資訊，請參閱維度和維度。

```
{
 "_aws": {
   "CloudWatchMetrics": [
     {
       "Dimensions": [ [ "functionVersion" ] ],
       ...
     }
   ]
 },
 "functionVersion": "$LATEST"
}
```

> **Note**
>
> 請在設定您的指標擷取時注意，因為這會影響您的自訂指標用量和對應的帳單。如果您不小心建立了以高基數維度 (例如 `requestId`) 為基礎的指標，根據設計，內嵌指標格式會建立自訂指標，對應至每個獨特的維度組合。如需詳細資訊，請參閱維度。

MetricDefinition 物件

A MetricDefinition 是「必須」包含下列成員的物件：

- 名稱— 字串 參考值 至指標 目標成員。指標目標「必須」是數值或數值的陣列。

一個 MetricDefinition 對象可以包含以下成員：

- 單位— 選用字串值，表示對應指標的測量單位。值應該是有效的 CloudWatch 公制單位。如需有效單位的資訊，請參閱MetricDatum。如果沒有提供值，則會使用預設值 NONE。
- StorageResolution— 選擇性整數值，代表對應量度的儲存解析度。將此值設定為 1 會將此量度指定為高解析度量，以便 CloudWatch 儲存小於一秒的分鐘解析度的量度。將此值設定為 60 會將此量度指定為標準解析度，並以 1 分鐘的解析度 CloudWatch 儲存。值應該是有效的 CloudWatch 支援解析度，1 或 60。若沒有提供值，則會使用預設值 60。

  如需高解析度指標的詳細資訊，請參閱 高解析度指標。

> **Note**
>
> 如果您打算在使用內嵌指標格式建立的指標上建立警示，請參閱在以內嵌指標格式建立的指標上設定警示以取得相關建議。

```
{
  "_aws": {
    "CloudWatchMetrics": [
      {
        "Metrics": [
          {
            "Name": "Time",
            "Unit": "Milliseconds",
            "StorageResolution": 60
          }
        ],
        ...
      }
    ]
  },
  "Time": 1
}
```

參考值

參考值是參考根節點上 目標成員 成員的字串值。這些參考「不應」與 RFC6901 中說明的 JSON 指標混淆。目標值不可為巢狀。

目標成員

有效目標「必須」是根節點上的成員，且不可為巢狀物件。例如，"A.a" 的 _reference_ 值「必須」符合以下成員：

```
{ "A.a" }
```

其「禁止」與巢狀成員相符：

```
{ "A": { "a" } }
```

目標成員的有效值取決於參考目標成員的項目。指標目標「必須」是數值或數值的陣列。數值陣列指標目標不得超過 100 個成員。維度目標「必須」擁有字串值。

內嵌指標格式範例和 JSON 結構描述

以下是內嵌指標格式的有效範例。

```
{
  "_aws": {
    "Timestamp": 1574109732004,
    "CloudWatchMetrics": [
      {
        "Namespace": "lambda-function-metrics",
        "Dimensions": [["functionVersion"]],
        "Metrics": [
          {
            "Name": "time",
            "Unit": "Milliseconds",
            "StorageResolution": 60
          }
        ]
      }
    ]
  },
  "functionVersion": "$LATEST",
  "time": 100,
  "requestId": "989ffbf8-9ace-4817-a57c-e4dd734019ee"
}
```

您可以使用以下結構描述來驗證內嵌指標格式文件。

```
{
    "type": "object",
    "title": "Root Node",
    "required": [
        "_aws"
    ],
    "properties": {
        "_aws": {
            "$id": "#/properties/_aws",
            "type": "object",
            "title": "Metadata",
            "required": [
                "Timestamp",
                "CloudWatchMetrics"
            ],
            "properties": {
                "Timestamp": {
                    "$id": "#/properties/_aws/properties/Timestamp",
                    "type": "integer",
                    "title": "The Timestamp Schema",
                    "examples": [
                        1565375354953
                    ]
                },
                "CloudWatchMetrics": {
                    "$id": "#/properties/_aws/properties/CloudWatchMetrics",
                    "type": "array",
                    "title": "MetricDirectives",
                    "items": {
                        "$id": "#/properties/_aws/properties/CloudWatchMetrics/items",
                        "type": "object",
                        "title": "MetricDirective",
                        "required": [
                            "Namespace",
                            "Dimensions",
                            "Metrics"
                        ],
                        "properties": {
                            "Namespace": {
                                "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/Namespace",
                                "type": "string",
                                "title": "CloudWatch Metrics Namespace",
```

```
                                "examples": [
                                    "MyApp"
                                ],
                                "pattern": "^(.*)$",
                                "minLength": 1,
                                "maxLength": 1024
                            },
                            "Dimensions": {
                                "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/Dimensions",
                                "type": "array",
                                "title": "The Dimensions Schema",
                                "minItems": 1,
                                "items": {
                                    "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Dimensions/items",
                                    "type": "array",
                                    "title": "DimensionSet",
                                    "minItems": 0,
                                    "maxItems": 30,
                                    "items": {
                                        "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Dimensions/items/items",
                                        "type": "string",
                                        "title": "DimensionReference",
                                        "examples": [
                                            "Operation"
                                        ],
                                        "pattern": "^(.*)$",
                                        "minLength": 1,
                                        "maxLength": 250
}
                                }
                            },
                            "Metrics": {
                                "$id": "#/properties/_aws/properties/CloudWatchMetrics/
items/properties/Metrics",
                                "type": "array",
                                "title": "MetricDefinitions",
                                "items": {
                                    "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Metrics/items",
                                    "type": "object",
                                    "title": "MetricDefinition",
```

```
                                    "required": [
                                        "Name"
                                    ],
                                    "properties": {
                                        "Name": {
                                            "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Metrics/items/properties/Name",
                                            "type": "string",
                                            "title": "MetricName",
                                            "examples": [
                                                "ProcessingLatency"
                                            ],
                                            "pattern": "^(.*)$",
                                            "minLength": 1,
                                            "maxLength": 1024
                                        },
                                        "Unit": {
                                            "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Metrics/items/properties/Unit",
                                            "type": "string",
                                            "title": "MetricUnit",
                                            "examples": [
                                                "Milliseconds"
                                            ],
                                            "pattern": "^(Seconds|Microseconds|
Milliseconds|Bytes|Kilobytes|Megabytes|Gigabytes|Terabytes|Bits|Kilobits|Megabits|
Gigabits|Terabits|Percent|Count|Bytes\\/Second|Kilobytes\\/Second|Megabytes\\/Second|
Gigabytes\\/Second|Terabytes\\/Second|Bits\\/Second|Kilobits\\/Second|Megabits\\/
Second|Gigabits\\/Second|Terabits\\/Second|Count\\/Second|None)$"
                                        },
                                        "StorageResolution": {
                                            "$id": "#/properties/_aws/properties/
CloudWatchMetrics/items/properties/Metrics/items/properties/StorageResolution",
                                            "type": "integer",
                                            "title": "StorageResolution",
                                            "examples": [
                                                60
                                            ]
                                        }
                                    }
                                }
                            }
                        }
                    }
```

```
                }
            }
        }
    }
}
```

使用 PutLogEvents API 傳送手動建立的內嵌指標格式記錄

您可以使用記錄 PutLogEvents API 將內嵌指標格式 CloudWatch 記錄傳送至 CloudWatch 記錄檔。呼叫時 PutLogEvents，您可以選擇性地包含下列 HTTP 標頭，以指示應擷取指標的 CloudWatch 記錄檔，但不再需要這樣做。

```
x-amzn-logs-format: json/emf
```

以下是使用適用於 Java 2.x 的 AWS SDK 的完整範例：

```
package org.example.basicapp;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsRequest;
import software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.InputLogEvent;
import software.amazon.awssdk.services.cloudwatchlogs.model.PutLogEventsRequest;

import java.util.Collections;

public class EmbeddedMetricsExample {
        public static void main(String[] args) {

                final String usage = "To run this example, supply a Region code (eg.
 us-east-1), log group, and stream name as command line arguments"
                                + "Ex: PutLogEvents <region-id> <log-group-name>
 <stream-name>";

                if (args.length != 3) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String regionId = args[0];
                String logGroupName = args[1];
```

```
                String logStreamName = args[2];

                CloudWatchLogsClient logsClient =
 CloudWatchLogsClient.builder().region(Region.of(regionId)).build();

                // Build a JSON log using the EmbeddedMetricFormat.
                long timestamp = System.currentTimeMillis();
                String message = "{" +
                                 "  \"_aws\": {" +
                                 "    \"Timestamp\": " + timestamp  + "," +
                                 "    \"CloudWatchMetrics\": [" +
                                 "      {" +
                                 "        \"Namespace\": \"MyApp\"," +
                                 "        \"Dimensions\": [[\"Operation\"], [\"Operation
\", \"Cell\"]]," +
                                 "        \"Metrics\": [{ \"Name\": \"ProcessingLatency
\", \"Unit\": \"Milliseconds\", \"StorageResolution\": 60 }]" +
                                 "      }" +
                                 "    ]" +
                                 "  }," +
                                 "  \"Operation\": \"Aggregator\"," +
                                 "  \"Cell\": \"001\"," +
                                 "  \"ProcessingLatency\": 100" +
                                 "}";
                InputLogEvent inputLogEvent = InputLogEvent.builder()
                        .message(message)
                        .timestamp(timestamp)
                        .build();

                // Specify the request parameters.
                PutLogEventsRequest putLogEventsRequest = PutLogEventsRequest.builder()
                        .logEvents(Collections.singletonList(inputLogEvent))
                        .logGroupName(logGroupName)
                        .logStreamName(logStreamName)
                        .build();

                logsClient.putLogEvents(putLogEventsRequest);

                System.out.println("Successfully put CloudWatch log event");
        }

}
```

> **Note**
>
> 借助內嵌指標格式，您可以透過在您帳戶的 AWS/Logs 命名空間中發佈的指標追蹤 EMF 日誌的處理進度。這些指標可用於追蹤從 EMF 產生指標失敗的情形，以及失敗是否因剖析或驗證造成。如需詳細資訊，請參閱使用 CloudWatch 指標監視。

使用 CloudWatch 代理程式傳送內嵌的量度格式記錄

若要使用此方法，請先為您要傳送內嵌指標格式記錄檔的服務安裝 CloudWatch 代理程式，然後您就可以開始傳送事件。

CloudWatch 代理程式必須是 1.230621.0 或更新版本。

> **Note**
>
> 您不需要安裝 CloudWatch 代理程式即可從 Lambda 函數傳送記錄。
> Lambda 函數逾時不會自動處理。這表示如果您的函數在排清指標前逾時，便不會擷取該呼叫的指標。

安裝 CloudWatch代理程式

為每個要傳送內嵌指標格式記錄的服務安裝 CloudWatch 代理程式。

在 EC2 上安裝 CloudWatch代理程式

首先，在執行個體上安裝 CloudWatch 代理程式。如需詳細資訊，請參閱 安裝 CloudWatch 代理程式。

一旦您安裝了代理程式，請設定代理程式在 UDP 或 TCP 連接埠上接聽內嵌指標格式日誌。以下是此組態的範例，此組態會在預設通訊端 tcp:25888 上接聽。如需代理程式組態的詳細資訊，請參閱 手動建立或編輯 CloudWatch 代理程式組態檔。

```
{
  "logs": {
    "metrics_collected": {
      "emf": { }
    }
  }
```

```
}
```

在 Amazon ECS 上安裝 CloudWatch代理程式

在 Amazon ECS 上部署 CloudWatch 代理程式最簡單的方法是將代理程式做為附屬程式執行，並在與應用程式相同的任務定義中定義代理程式。

建立代理程式組態檔案

在本機建立 CloudWatch 代理程式設定檔。在此範例中，相對檔案路徑將會是 `amazon-cloudwatch-agent.json`。

如需代理程式組態的詳細資訊，請參閱 手動建立或編輯 CloudWatch 代理程式組態檔。

```
{
  "logs": {
    "metrics_collected": {
      "emf": { }
    }
  }
}
```

將組態推送至 SSM 參數存放區

輸入下列命令，將 CloudWatch 代理程式組態檔推送至系 AWS Systems Manager (SSM) 參數存放區。

```
aws ssm put-parameter \
    --name "cwagentconfig" \
    --type "String" \
    --value "`cat amazon-cloudwatch-agent.json`" \
    --region "{{region}}"
```

設定任務定義

設定您的工作定義以使用 CloudWatch 代理程式並公開 TCP 或 UDP 連接埠。您應使用的範例任務定義取決於您的網路模式。

請注意，`webapp` 會指定 `AWS_EMF_AGENT_ENDPOINT` 環境變數。此環境變數會由程式庫使用，且應指向代理程式正在接聽的端點。此外，`cwagent` 會指定 `CW_CONFIG_CONTENT` 為 “valueFrom” 參數，其指向您在先前步驟中建立的 SSM 組態。

本節包含一個橋接模式的範例，以及一個託管模式或 awsvpc 模式的範例。如需如何在 Amazon ECS 上設定 CloudWatch 代理程式的更多範例，請參閱 Github 範例儲存庫

以下是橋接模式的範例。啟用橋接模式聯網時，代理程式需要使用 `links` 參數連結至您的應用程式，並且必須使用容器名稱來定址。

```
{
  "containerDefinitions": [
          {
              "name": "webapp",
              "links": [ "cwagent" ],
              "image": "my-org/web-app:latest",
              "memory": 256,
              "cpu": 256,
              "environment": [{
                "name": "AWS_EMF_AGENT_ENDPOINT",
                "value": "tcp://cwagent:25888"
              }],
          },
          {
              "name": "cwagent",
              "mountPoints": [],
              "image": "public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest",
              "memory": 256,
              "cpu": 256,
              "portMappings": [{
                  "protocol": "tcp",
                  "containerPort": 25888
              }],
              "environment": [{
                "name": "CW_CONFIG_CONTENT",
                "valueFrom": "cwagentconfig"
              }],
          }
      ],
}
```

以下是託管模式或 awsvpc 模式的範例。在這些網路模式上執行時，代理程式可以透過 `localhost` 定址。

```
{
  "containerDefinitions": [
```

```
            {
                "name": "webapp",
                "image": "my-org/web-app:latest",
                "memory": 256,
                "cpu": 256,
                "environment": [{
                  "name": "AWS_EMF_AGENT_ENDPOINT",
                  "value": "tcp://127.0.0.1:25888"
                }],
            },
            {
                "name": "cwagent",
                "mountPoints": [],
                "image": "public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest",
                "memory": 256,
                "cpu": 256,
                "portMappings": [{
                    "protocol": "tcp",
                    "containerPort": 25888
                }],
                "environment": [{
                  "name": "CW_CONFIG_CONTENT",
                  "valueFrom": "cwagentconfig"
                }],
            }
        ],
}
```

> **Note**
>
> 在 awsvpc 模式中，您必須將公用 IP 位址提供給虛擬私人雲端 (僅限 Fargate)、設定 NAT 閘道或設定記錄 VPC 端點 CloudWatch 。如需設定 NAT 的詳細資訊，請參閱 NAT 閘道。如需有關設定 CloudWatch 記錄檔 VPC 端點的詳細資訊，請參閱搭配介面 VPC 端點使用 CloudWatch 記錄檔。
>
> 以下是如何將公有 IP 地址指派給使用 Fargate 啟動類型任務的範例。
>
> ```
> aws ecs run-task \
> --cluster {{cluster-name}} \
> --task-definition cwagent-fargate \
> --region {{region}} \
> --launch-type FARGATE \
> ```

```
--network-configuration
 "awsvpcConfiguration={subnets=[{{subnetId}}],securityGroups=[{{sgId}}],assignPublicIp=ENA
```

確認許可

確認執行您任務的 IAM 角色具有從 SSM 參數存放區讀取的許可。您可以通過附加亞馬遜 SS ReadOnlyAccess M 策略來添加此權限。若要執行此作業，請輸入以下命令。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/AmazonSSMReadOnlyAccess
 \
--role-name CWAgentECSExecutionRole
```

在 Amazon E CloudWatch KS 上安裝代理

如果您已經在此叢集上安裝了 CloudWatch 容器見解，則可以略過此程序的部分內容。

許可

如果您尚未安裝 Container Insights，首先請先確認您的 Amazon EKS 節點具有適當的 IAM 許可。他們應該有CloudWatchAgentServerPolicy附件。如需詳細資訊，請參閱 確認 先決條件。

創建 ConfigMap

為代理 ConfigMap 程式建立。 ConfigMap 也會告知代理程式在 TCP 或 UDP 連接埠上接聽。使用以下內容 ConfigMap。

```
# cwagent-emf-configmap.yaml
apiVersion: v1
data:
  # Any changes here must not break the JSON format
  cwagentconfig.json: |
    {
      "agent": {
        "omit_hostname": true
      },
      "logs": {
        "metrics_collected": {
          "emf": { }
        }
      }
```

```
    }
kind: ConfigMap
metadata:
  name: cwagentemfconfig
  namespace: default
```

如果您已安裝容器見解，請將以下"emf": { }行新增至您現有的 ConfigMap。

套用 ConfigMap

輸入下列指令以套用 ConfigMap。

```
kubectl apply -f cwagent-emf-configmap.yaml
```

部署代理程式

若要將 CloudWatch 代理程式部署為附屬，請將代理程式新增至您的網繭定義，如下列範例所示。

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp
  namespace: default
spec:
  containers:
    # Your container definitions go here
    - name: web-app
      image: my-org/web-app:latest
    # CloudWatch Agent configuration
    - name: cloudwatch-agent
      image: public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest
      imagePullPolicy: Always
      resources:
        limits:
          cpu: 200m
          memory: 100Mi
        requests:
          cpu: 200m
          memory: 100Mi
      volumeMounts:
        - name: cwagentconfig
          mountPath: /etc/cwagentconfig
```

```
      ports:
  # this should match the port configured in the ConfigMap
        - protocol: TCP
          hostPort: 25888
          containerPort: 25888
  volumes:
    - name: cwagentconfig
      configMap:
        name: cwagentemfconfig
```

使用 CloudWatch 代理程式傳送內嵌的量度格式記錄

當您安裝並執行 CloudWatch 代理程式時，您可以透過 TCP 或 UDP 傳送內嵌的指標格式記錄檔。透過代理程式傳送日誌時有兩個需求：

- 日誌必須包含 LogGroupName 索引鍵，告知代理程式要使用的日誌群組。
- 每個日誌事件都必須在單獨的一行上。換句話說，日誌事件不能包含換行 (\n) 字元。

日誌事件也必須遵循內嵌指標格式規格。如需詳細資訊，請參閱 規格：內嵌指標格式 。

如果您打算在使用內嵌指標格式建立的指標上建立警示，請參閱在以內嵌指標格式建立的指標上設定警示以取得相關建議。

以下是從 Linux bash shell 手動傳送日誌事件的範例。您可以改為使用您所選擇程式設計語言提供的 UDP 通訊端介面。

```
echo '{"_aws":{"Timestamp":1574109732004,"LogGroupName":"Foo","CloudWatchMetrics":
[{"Namespace":"MyApp","Dimensions":[["Operation"]],"Metrics":
[{"Name":"ProcessingLatency","Unit":"Milliseconds","StorageResolution":60}]}]},"Operation":"Agg
 \
> /dev/udp/0.0.0.0/25888
```

> **Note**
>
> 借助內嵌指標格式，您可以透過在您帳戶的 AWS/Logs 命名空間中發佈的指標追蹤 EMF 日誌的處理進度。這些指標可用於追蹤從 EMF 產生指標失敗的情形，以及失敗是否因剖析或驗證造成。如需詳細資訊，請參閱使用 CloudWatch 指標監視。

將嵌入式度量格式與 AWS 發行版搭配使用 OpenTelemetry

您可以使用內嵌的度量格式做為 OpenTelemetry 專案的一部分。 OpenTelemetry 是一項開放原始碼計劃，透過提供單一規格和 API，消除供應商特定格式之間的界限和限制，以便追蹤、記錄和指標。如需詳細資訊，請參閱OpenTelemetry。

將內嵌指標格式與 OpenTelemetry 需要兩個元件搭配使用： OpenTelemetry符合規範的資料來源，以及啟用與 CloudWatch 嵌入式指標格式記錄搭配使用的 AWS Distro (適用於 OpenTelemetry 收集器)。

我們已預先設定 OpenTelemetry 元件的重新分配 (由維護) AWS，以便盡可能簡化上線。 OpenTelemetry 有關使用嵌入式指標格式的更多信息，除了其他 AWS 服務外，請參閱AWS . OpenTelemetry

如需語言支援和使用方式的其他資訊，請參閱 Github 上的AWS Observability。

在主控台中檢視您的指標和日誌

產生擷取指標的內嵌指標格式記錄後，您可以使用 CloudWatch主控台來檢視指標。內嵌指標包含您在產生日誌時指定的維度。此外，您使用用戶端程式庫產生的內嵌指標也具備下列預設維度：

- ServiceType
- ServiceName
- LogGroup

檢視從內嵌指標格式日誌產生的指標

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 指標。
3. 選取您在產生內嵌指標時，為其指定的命名空間。如果您使用用戶端程式庫產生測量結果，但未指定命名空間，請選取aws-embedded-metrics。這是使用用戶端程式庫所產生內嵌指標的預設命名空間。
4. 選取量度維度 (例如，ServiceName)。
5. All metrics (所有指標) 索引標籤會顯示命名空間中該維度的所有指標。您可以執行下列作業：
   a. 若要將資料表排序，請使用直欄標題。
   b. 若要將指標圖形化，請勾選指標旁的核取方塊。若要選擇所有指標，請勾選表格標題列中的核取方塊。

c. 若要依資源篩選，請選擇資源 ID，然後選擇 Add to search (新增至搜尋)。

d. 若要依指標篩選，請選擇指標名稱，然後選擇 Add to search (新增至搜尋)。

使用記錄檔見解查詢 CloudWatch 記錄

您可以使用 CloudWatch Logs Insights 提供操作事件根本原因的深入見解，查詢與擷取指標相關聯的詳細記錄事件。從您日誌擷取指標的其中一個好處是您稍後可以使用唯一指標 (指標名稱加上唯一的維度集) 和指標值來篩選您的日誌，以取得促成彙整指標值事件的事件內容。

例如，若要取得受影響的要求識別碼或 x 射線追蹤識別碼，您可以在 CloudWatch 記錄深入解析中執行下列查詢。

```
filter Latency > 1000 and Operation = "Aggregator"
| fields RequestId, TraceId
```

您也可以在高基數索引鍵上執行查詢時間彙整，例如尋找受事件影響的客戶。下列的範例示範了這一點。

```
filter Latency > 1000 and Operation = "Aggregator"
| stats count() by CustomerId
```

如需詳細資訊，請參閱使用日誌深入分析分析 CloudWatch 記錄檔

在以內嵌指標格式建立的指標上設定警示

一般來說，在透過內嵌指標格式產生的指標上建立警示時，其模式與在任何其他指標上建立警示的模式相同。如需詳細資訊，請參閱 使用 Amazon CloudWatch 警報。

內嵌指標格式指標產生取決於您的記錄發佈流程，因為記錄檔必須由 CloudWatch 記錄檔處理才能轉換為指標。因此，您必須即時發佈記日誌，才能在評估警示的期間內建立指標資料點。

如果您計劃使用內嵌指標格式傳送高解析度指標，並在這些指標上建立警示，我們建議您以 5 秒或更少的時間間隔將 CloudWatch 記錄清除至記錄檔，以避免產生額外延遲，而導致部分或遺失資料發生警報。如果您使用的是 CloudWatch 代理程式，可以透過在代理程 CloudWatch 式組態檔中設定`force_flush_interval`參數來調整清除間隔。此值預設為 5 秒。

如果您在無法控制日誌排清間隔的其他平台上使用 Lambda，請考慮使用「N 個中有 M 個」警示來控制用於警示的資料點數量。如需更多詳細資訊，請參閱 評估警示。

AWS 發佈指 CloudWatch標的服務

下列 AWS 服務會將量度發佈至 CloudWatch。如需指標和維度的相關詳細資訊，請參閱指定的文件。

服務	命名空間	文件
AWS Amplify	AWS/AmplifyHosting	監控
Amazon API Gateway	AWS/ApiGateway	使用 Amazon 監控 API 執行 CloudWatch
Amazon AppFlow	AWS/AppFlow	AppFlow 用 Amazon 監控 Amazon CloudWatch
AWS 應用程式遷移	AWS/MGN	使用 Amazon 監控應用程式遷移服務 CloudWatch
AWS 應用亞軍	AWS/AppRunner	檢視報告給的應用程式執行器服務 CloudWatch
AppStream 2.0	AWS/AppStream	監控 Amazon AppStream 2.0 資源
AWS AppSync	AWS/AppSync	CloudWatch 度量
Amazon Athena	AWS/Athena	使用 CloudWatch 指標監視 Athena 查詢
Amazon Aurora	AWS/RDS	Amazon Aurora 指標
AWS Backup	AWS/Backup	監督 AWS Backup 測量結果 CloudWatch
Amazon Bedrock	AWS/Bedrock	使用 Amazon 監控 Amazon 基岩 CloudWatch
AWS Billing and Cost Management	AWS/Billing	使用提醒和通知監控費用

服務	命名空間	文件
Amazon Braket	AWS/Braket/By Device	使用 Amazon 監控 Amazon Braket CloudWatch
AWS Certificate Manager	AWS/CertificateManager	支援的 CloudWatch 指標
AWS 私有 CA	AWS/ACMPrivateCA	支援的 CloudWatch 指標
AWS Chatbot	AWS/Chatbot	AWS Chatbot 使用 Amazon 監控 CloudWatch
Amazon Chime	AWS/ChimeVoiceConnector	Amazon Chime 使用 Amazon 監控 CloudWatch
Amazon Chime SDK	AWS/ChimeSDK	服務指標
AWS Client VPN	AWS/ClientVPN	使用 Amazon 監控 CloudWatch
Amazon CloudFront	AWS/CloudFront	監視 CloudFront 活動使用 CloudWatch
AWS CloudHSM	AWS/CloudHSM	獲取 CloudWatch 指標
Amazon CloudSearch	AWS/CloudSearch	使用 Amazon 監控 Amazon CloudSearch 域 CloudWatch
AWS CloudTrail	AWS/CloudTrail	支援的 CloudWatch 指標
CloudWatch 代理	CWAgent 或自訂命名空間	CloudWatch 代理程式收集的測量結果

服務	命名空間	文件
CloudWatch 測量結果流	AWS/CloudWatch/MetricStreams	使用指標監視 CloudWatch 指標串流
CloudWatch 朗姆酒	AWS/RUM	CloudWatch 您可以使用 CloudWatch RUM 收集的指標
CloudWatch Synthetics	CloudWatchSynthetics	CloudWatch 加那利群島發布的指標
Amazon CloudWatch 日誌	AWS/Logs	以測量 CloudWatch 結果監督使用
AWS CodeBuild	AWS/CodeBuild	監控 AWS CodeBuild
Amazon 評論 CodeGuru 家		使用 CodeGuru Amazon 監控審稿 CloudWatch
Amazon Kendra		使用 Amazon 監控亞馬遜肯德拉 CloudWatch
Amazon CodeWhisperer	AWS/CodeWhisperer	Amazon CodeWhisperer 使用 Amazon 監控 CloudWatch
Amazon Cognito	AWS/Cognito	監控 Amazon Cognito
Amazon Comprehend	AWS/Comprehend	監控 Amazon Comprehend 端點
AWS Config	AWS/Config	AWS Config 使用狀況和成功量度
Amazon Connect	AWS/Connect	Amazon 指標中監控 CloudWatch Amazon Connect
Amazon Data Lifecycle Manager	AWS/DataLifecycleManager	使用 Amazon 監控您的政策 CloudWatch

服務	命名空間	文件
AWS DataSync	AWS/DataSync	監控任務
Amazon DataZone		DataZone 用 Amazon 監控 Amazon CloudWatch
Amazon DevOps Guru	AWS/DevOps-Guru	Amazon DevOps Guru 使用 Amazon 監控 CloudWatch
AWS Database Migration Service	AWS/DMS	監視 AWS DMS 工作
AWS Direct Connect	AWS/DX	使用 Amazon 監控 CloudWatch
AWS Directory Service	AWS/DirectoryService	使用 Amazon CloudWatch 指標判斷何時新增網域控制站
Amazon DocumentDB	AWS/DocDB	Amazon DocumentDB 指標
Amazon DynamoDB	AWS/DynamoDB	DynamoDB 指標和維度
DynamoDB Accelerator (DAX)	AWS/DAX	檢視 DAX 指標和維度
Amazon EC2	AWS/EC2	使用以下方式監視您 CloudWatch
Amazon EC2 Elastic Graphics	AWS/ElasticGPUs	使用 CloudWatch 指標監視彈性圖形
Amazon EC2 Spot 機群	AWS/EC2Spot	CloudWatch 競價型艦隊的指標

服務	命名空間	文件
Amazon EC2 Auto Scaling	AWS/AutoScaling	使用以下方式監控您的 Auto Scaling 集 CloudWatch
AWS Elastic Beanstalk	AWS/ElasticBeanstalk	為環境發佈 Amazon CloudWatch 自訂指標
Amazon Elastic Block Store	AWS/EBS	Amazon E CloudWatch BS 的 Amazon 指標
Amazon Elastic Container Registry	AWS/ECR	Amazon ECR 儲存庫指標
Amazon Elastic Container Service	AWS/ECS	Amazon ECS 指標 CloudWatch
Amazon ECS 透過 CloudWatch 容器洞察	ECS/ContainerInsights	Amazon ECS Container Insights 指標
Amazon ECS 叢集自動擴展	AWS/ECS/ManagedScaling	Amazon ECS 叢集自動擴展
AWS Elastic Disaster Recovery		CloudWatch DRS 的測量結果
Amazon Elastic File System	AWS/EFS	使用監控 CloudWatch
Amazon Elastic Inference	AWS/ElasticInference	使用 CloudWatch 指標監控 Amazon Elastic Inference

服務	命名空間	文件
透過 CloudWatch 容器洞察的 Amazon EKS	`ContainerInsights`	Amazon EKS 和 Kubernetes Container Insights 指標
Elastic Load Balancing	`AWS/ApplicationELB`	CloudWatch Application Load Balancer 的指標
Elastic Load Balancing	`AWS/NetworkELB`	CloudWatch Network Load Balancer 的指標
Elastic Load Balancing	`AWS/GatewayELB`	CloudWatch 閘道 Load Balancer 的指標
Elastic Load Balancing	`AWS/ELB`	CloudWatch Classic Load Balancer 的指標
Amazon Elastic Transcoder	`AWS/ElasticTranscoder`	使用 Amazon 監控 CloudWatch
Amazon ElastiCache	`AWS/ElastiCache`	監督測 CloudWatch 量結果的使用
Amazon ElastiCache 的雷迪斯	`AWS/ElastiCache`	監督測 CloudWatch 量結果的使用
Amazon OpenSearch 服務	`AWS/ES`	使用 Amazon 監控 OpenSearch 叢集指標 CloudWatch
Amazon EMR	`AWS/ElasticMapReduce`	監視指標 CloudWatch
AWS Elemental MediaConnect	`AWS/MediaConnect`	MediaConnect 使用 Amazon 監控 CloudWatch

服務	命名空間	文件
AWS Elemental MediaConvert	AWS/MediaConvert	使用 CloudWatch 測量結果檢視 AWS Elemental MediaConvert 資源的測量結果
AWS Elemental MediaLive	AWS/MediaLive	使用 Amazon CloudWatch 指標監控活動
AWS Elemental MediaPackage	AWS/MediaPackage	AWS Elemental MediaPackage 使用 Amazon CloudWatch 指標監控
AWS Elemental MediaStore	AWS/MediaStore	AWS Elemental MediaStore 使用 Amazon CloudWatch 指標監控
AWS Elemental MediaTailor	AWS/MediaTailor	AWS Elemental MediaTailor 使用 Amazon 監控 CloudWatch
Amazon EventBridge	AWS/Events	監控 Amazon EventBridge
Amazon FinSpace		記錄和監控
Amazon Forecast		CloudWatch Amazon Forecast 指標
Amazon Fraud Detector		用 Amazon 監控 Amazon Fraud Detector CloudWatch
Amazon FSx for Lustre	AWS/FSx	監控 Amazon FSx for Lustre
Amazon FSx for OpenZFS	AWS/FSx	使用 Amazon 監控 CloudWatch
Amazon FSx for Windows File Server	AWS/FSx	監控 Amazon FSx for Windows File Server

服務	命名空間	文件
Amazon FSx NetApp	AWS/FSx	使用 Amazon 監控 CloudWatch
Amazon FSx for OpenZFS	AWS/FSx	使用 Amazon 監控 CloudWatch
Amazon GameLift	AWS/GameLift	監控 GameLift Amazon CloudWatch
AWS Global Accelerator	AWS/GlobalAccelerator	使用 CloudWatch Amazon AWS Global Accelerator
AWS Glue	Glue	AWS Glue 使用測 CloudWatch 量結果監
AWS Ground Station	AWS/GroundStation	指標使用 Amazon CloudWatch
AWS HealthLake	AWS/HealthLake	HealthLake 使用監控 CloudWatch
Amazon Inspector	AWS/Inspector	使用監控 Amazon Inspector CloudWatch
Amazon Interactive Video Service	AWS/IVS	使用 Amazon 監控 Amazon IVS CloudWatch
Amazon Interactive Video Service Chat	AWS/IVSChat	使用 Amazon 監控 Amazon IVS CloudWatch
AWS IoT	AWS/IoT	AWS IoT 指標和維度
AWS IoT Analytics	AWS/IoTAnalytics	命名空間、指標與維度
AWS IoT FleetWise	AWS/IoTFleetWise	FleetWise 用 Amazon 監控 AWS IoT CloudWatch

服務	命名空間	文件
AWS IoT SiteWise	AWS/IoTSiteWise	AWS IoT SiteWise 使用 Amazon CloudWatch 指標監控
AWS IoT TwinMaker	AWS/IoTTwinMaker	AWS IoT TwinMaker 使用 Amazon CloudWatch 指標監控
AWS IoT 一鍵式		使用 AWS IoT Amazon 監控一鍵 CloudWatch
AWS Key Management Service	AWS/KMS	使用監控 CloudWatch
Amazon Keyspaces (適用於 Apache Cassandra)	AWS/Cassandra	Amazon Keyspaces 指標和維度
Amazon Kendra		使用 Amazon 監控亞馬遜肯德拉 CloudWatch
Amazon Managed Service for Apache Flink	AWS/KinesisAnalytics	適用於 SQL 應用程式的 Apache Flink 的受管理服務：使用 CloudWatch 適用於 Apache Flink 的 Managed Service for Apache Flink：檢視 Amazon Managed Service for Apache Flink 指標和維度
Amazon 數據 Firehose	AWS/Firehose	使 CloudWatch 用指標監控 Firehose
Amazon Kinesis Data Streams	AWS/Kinesis	使用 Amazon 監控亞馬遜室運動數據流 CloudWatch
Amazon Kinesis Video Streams	AWS/KinesisVideo	監控 Kinesis Video Streams 量度 CloudWatch
AWS Lambda	AWS/Lambda	AWS Lambda 指標
Amazon Lex	AWS/Lex	與 Amazon 監控亞馬遜 Lex CloudWatch

服務	命名空間	文件
AWS License Manager	AWSLicenseManager/licenseUsage AWS/LicenseManager/LinuxSubscriptions	使用 Amazon 監控授權使用情況 CloudWatch Linux 訂閱的使用量指標和 Amazon CloudWatch 警示
Amazon Location Service	AWS/Location	Amazon 定 Location Service 指標匯出到 Amazon CloudWatch
Amazon Lookout for Equipment	AWS/lookoutequipment	監控 Lookout for Equipment 與 Amazon CloudWatch
Amazon Lookout for Metrics	AWS/LookoutMetrics	監視 Lookout for Metrics 與 Amazon CloudWatch
Amazon Lookout for Vision	AWS/LookoutVision	監控 Lookout for Vision 與 Amazon CloudWatch
AWS 大型主機現代化		使用 Amazon 監控 AWS 大型主機現代化 CloudWatch
Amazon Machine Learning	AWS/ML	使用 CloudWatch 指標監控 Amazon ML
Amazon Managed Blockchain	AWS/managedblockchain	在 Amazon Managed Blockchain 上使用 Hyperledger Fabric 對等節點指標

服務	命名空間	文件
Amazon Managed Service for Prometheus	AWS/Prometheus	Amazon CloudWatch 指標
Amazon Managed Streaming for Apache Kafka	AWS/Kafka	使用 Amazon 監控 Amazon MSK CloudWatch
Amazon Managed Streaming for Apache Kafka	AWS/KafkaConnect	監視 MSK 連線
Amazon Managed Workflows for Apache Airflow	AWS/MWAA	適用於 Amazon MWAA 的容器、佇列和資料庫指標
Amazon MemoryDB for Redis	AWS/MemoryDB	監控 CloudWatch 指標
Amazon MQ	AWS/AmazonMQ	使用 Amazon 監控 Amazon MQ 經紀人 CloudWatch
Amazon Neptune	AWS/Neptune	使用監控 Neptune CloudWatch
AWS Network Firewall	AWS/NetworkFirewall	AWS Network Firewall Amazon 的指標 CloudWatch
AWS 網路管理員	AWS/NetworkManager	CloudWatch 內部部署資源的指標

服務	命名空間	文件
Amazon Nimble Studio	AWS/NimbleStudio	使用 Amazon 監控靈活的工作室 CloudWatch
AWS HealthOmics	AWS/Omics	AWS HealthOmics 使用 Amazon 監控 CloudWatch
AWS OpsWorks	AWS/OpsWorks	使用 Amazon 監控堆棧 CloudWatch
AWS Outposts	AWS/Outposts	CloudWatch 度量 AWS Outposts
AWS Panorama	AWS/PanoramaDeviceMetrics	使用 Amazon 監控設備和應用程式 CloudWatch
Amazon Personalize	AWS/Personalize	CloudWatch Amazon Personalize 化指標
Amazon Pinpoint	AWS/Pinpoint	檢視 Amazon Pinpoint 度量 CloudWatch
Amazon Polly	AWS/Polly	CloudWatch 與 Amazon Polly 集成
AWS PrivateLink	AWS/PrivateLinkEndpoints	CloudWatch 度量 AWS PrivateLink
AWS PrivateLink	AWS/PrivateLinkServices	CloudWatch 度量 AWS PrivateLink
AWS 私人 5G	AWS/Private5G	Amazon CloudWatch 指標
Amazon QLDB	AWS/QLDB	在 Amazon 監控數據 QuickSight

服務	命名空間	文件
Amazon QuickSight	AWS/QuickSight	使用 Amazon 監控 CloudWatch
Amazon Redshift	AWS/Redshift	Amazon Redshift 效能資料
Amazon Relational Database Service	AWS/RDS	使用 Amazon 監控 Amazon RDS 指標 CloudWatch
Amazon Rekognition	AWS/Rekognition	使用 Amazon 監控 Rekognition CloudWatch
AWS re:Post 私人	AWS/rePostPrivate	與 Amazon AWS re:Post 私人監控 CloudWatch
AWS RoboMaker	AWS/Robomaker	AWS RoboMaker 使用 Amazon 監控 CloudWatch
Amazon Route 53	AWS/Route53	監控 Amazon Route 53
Route 53 Application Recovery Controller	AWS/Route53RecoveryReadiness	將 Amazon CloudWatch 搭配應用程式復原控制器
Amazon SageMaker	AWS/SageMaker	SageMaker 使用監控 CloudWatch
Amazon SageMaker 模型構建管道	AWS/SageMaker/ModelBuildingPipeline	SageMaker 管道指標

服務	命名空間	文件
AWS Secrets Manager	AWS/SecretsManager	監控 Secrets Manager 與 Amazon CloudWatch
Amazon Security Lake	AWS/SecurityLake	CloudWatch Amazon 安全湖的指標
Service Catalog	AWS/ServiceCatalog	Service Catalog CloudWatch 度量
AWS Shield Advanced	AWS/DDoSProtection	使用監控 CloudWatch
Amazon Simple Email Service	AWS/SES	從中擷取 Amazon SES 事件資料 CloudWatch
AWS SimSpace Weaver	AWS/simspaceweaver	AWS SimSpace Weaver 使用 Amazon 監控 CloudWatch
Amazon Simple Notification Service	AWS/SNS	使用監控 Amazon SNS CloudWatch
Amazon Simple Queue Service	AWS/SQS	使用監控 Amazon SQS 佇列 CloudWatch
Amazon S3	AWS/S3	使用 Amazon 監控指標 CloudWatch
S3 Storage Lens	AWS/S3/Storage-Lens	監控 S3 儲存鏡頭指標 CloudWatch
Amazon Simple Workflow Service	AWS/SWF	Amazon SWF 指標 CloudWatch
AWS Step Functions	AWS/States	監視 Step Functions 使用 CloudWatch

服務	命名空間	文件
AWS Storage Gateway	AWS/StorageGateway	使用 Amazon CloudWatch 指標
AWS Systems Manager 執行命令	AWS/SSM-RunCommand	監督執行命令測量結果 CloudWatch
Amazon Textract	AWS/Textract	CloudWatch Amazon Textract 的指標
Amazon Timestream	AWS/Timestream	Timestream 指標和維度
AWS Transfer for SFTP	AWS/Transfer	AWS SFTP CloudWatch 度量
Amazon Transcribe	AWS/Transcribe	Amazon Transcribe 使用 Amazon 監控 CloudWatch
Amazon Translate	AWS/Translate	CloudWatch Amazon Translate 的指標和維度
AWS Trusted Advisor	AWS/TrustedAdvisor	建立 Trusted Advisor 警示 CloudWatch
Amazon VPC	AWS/NATGateway	監視您的 NAT 閘道 CloudWatch
Amazon VPC	AWS/TransitGateway	CloudWatch 您的交通閘道的指標
Amazon VPC	AWS/VPN	使用監控 CloudWatch
Amazon VPC IP 地址管理員	AWS/IPAM	使用 Amazon 創建警報 CloudWatch

服務	命名空間	文件
AWS WAF	AWS/WAFV2 對於 AWS WAF 資源 WAF對於 AWS WAF 傳統資源	使用監控 CloudWatch
Amazon WorkMail	AWS/WorkMail	Amazon WorkMail 使用 Amazon 監控 CloudWatch
Amazon WorkSpaces	AWS/WorkSpaces	監視您的 WorkSpaces 使用 CloudWatch 指標
Amazon WorkSpaces 網站	AWS/WorkSpacesWeb	用 Amazon 監控 Amazon WorkSpaces 網站 CloudWatch

AWS 使用量度

CloudWatch 收集跟踪某些 AWS 資源和 API 使用情況的指標。這些指標會在 AWS/Usage 命名空間中發佈。中的使用量度可 CloudWatch 讓您透過視覺化主 CloudWatch 控台中的指標、建立自訂儀表板、偵測 CloudWatch 異常偵測活動中的變更，以及設定警示，以便在使用量接近閾值時提醒您，藉此主動管理使用情況。

有些 AWS 服務會將這些使用量度與 Service Quotas 整合。對於這些服務，您可以用 CloudWatch 來管理帳戶對服務配額的使用情況。如需詳細資訊，請參閱 視覺化您的服務配額和設定警示。

主題

- 視覺化您的服務配額和設定警示
- AWS API 使用量指標
- CloudWatch 使用量度

視覺化您的服務配額和設定警示

對於某些 AWS 服務，您可以使用使用量度在 CloudWatch 圖形和儀表板上以視覺化方式呈現您目前的服務使用情況。您可以使用 CloudWatch 量度數學函數，在圖形上顯示這些資源的服務配額。您也可以設定警示，在您的用量接近服務配額時發出警示。如需有關服務配額的詳細資訊，請參閱《Service Quotas 使用者指南》中的什麼是 Service Quotas。

如果您登入的帳戶已設定為 CloudWatch 跨帳戶可觀察性的監視帳戶，則可以使用該監視帳戶將服務配額視覺化，並針對連結至該監視帳戶的來源帳戶中的指標設定警示。如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。

目前，下列服務會將其用量指標與 Service Quotas 整合：

- AWS CloudHSM
- Amazon Chime SDK
- Amazon CloudWatch
- Amazon CloudWatch 日誌
- Amazon DynamoDB
- Amazon EC2
- Amazon Elastic Container Registry
- Elastic Load Balancing

- AWS Fargate
- AWS Fault Injection Service
- AWS 互動視頻服務
- AWS Key Management Service
- Amazon 數據 Firehose
- Amazon Location Service
- Amazon Managed Blockchain（AMB）查詢
- AWS RoboMaker
- Amazon SageMaker

視覺化服務配額並選擇是否設定警示

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇 指標。
3. 在 [所有量度] 索引標籤上，選擇 [使用狀況]，然後選擇 [依 AWS 資源]

   服務配額用量指標清單隨即出現。
4. 選取其中一個指標旁的核取方塊。

   此圖表會顯示您目前該 AWS 資源的使用量。
5. 若要將服務配額新增至圖表，請執行下列動作：

   a. 選擇 Graphed metrics (圖表化指標) 標籤。

   b. 選擇 Math expression (數學表達式)、Start with an empty expression (以空表達式開始)。在新資料列的 Details (詳細資訊) 下，輸入 **`SERVICE_QUOTA(m1)`**。

      圖表中會新增一行，顯示指標所示資源的服務配額。
6. 若要查看目前用量的配額百分比，請新增表達式或變更目前的 SERVICE_QUOTA 表達式。要使用的新表達式是 **`m1/SERVICE_QUOTA(m1)*100`**。
7. (選用) 若要設定接近服務配額的警示通知，請執行下列動作：

   a. 在 **`m1/SERVICE_QUOTA(m1)*100`** 資料列的 Actions (動作) 下，選擇警示圖示。它看起來像一個鈴鐺。

      警示建立頁面隨即出現。

b. 確定在 Conditions (條件) 下，Threshold type (閾值類型) 為 Static (靜態)，而 Whenever Expression1 is (每當 Expression1) 設為 Greater (大於)。在 than (比較值) 下輸入 **80**。當您的用量超過配額的 80% 時，即會建立進入 ALARM 狀態的警示。

c. 選擇下一步。

d. 在下一頁中，選取 Amazon SNS 主題或建立新主題，然後選擇 Next (下一步)。當警示進入 ALARM 狀態時，即會通知您選取的主題。

e. 在下一頁中，輸入警示的名稱和說明，然後選擇 Next (下一步)。

f. 選擇 Create alarm (建立警示)。

AWS API 使用量指標

大多數支援 AWS CloudTrail 記錄的 API 也會向 CloudWatch. 中的 API 使用量度可 CloudWatch 讓您透過視覺化主 CloudWatch 控台中的指標、建立自訂儀表板、使用 CloudWatch 異常偵測偵測活動中的變更，以及設定在使用量接近閾值時提醒的警示，主動管理 API 使用情況。

下表列出報告 API 使用量度的服務 CloudWatch，以及用於`Service`維度查看該服務之使用量度的值。

服務	**`Service`** 維度的數值
AWS Identity and Access Management Access Analyzer	`Access Analyzer`
AWS Account Management	`Account Management`
企業版 Alexa	`A4B`
Amazon API Gateway	`API Gateway`
AWS App Mesh	`App Mesh`
AWS AppConfig	`AWS AppConfig`
Amazon AppFlow	`AppFlow`
Application Auto Scaling	`Application Auto Scaling`

服務	**Service** 維度的數值
Application Discovery Service	`Application Discovery Service`
Amazon AppStream	`AppStream`
AppStream 2.0 Image Builder	`Image Builder`
Amazon Athena	`Athena`
AWS Audit Manager	`Audit Manager`
AWS Backup	`Backup`
AWS Batch	`Batch`
Amazon Braket	`Braket`
AWS 預算	`Budgets`
AWS Certificate Manager	`Certificate Manager`
Amazon Chime SDK	`ChimeSDK`
Amazon 雲端目錄	`Cloud Directory`
AWS Cloud Map	`Cloud Map`
AWS CloudFormation	`CloudFormation`
AWS CloudHSM	`CloudHSM`
Amazon CloudSearch	`CloudSearch`
AWS CloudShell	`CloudShell`
AWS CloudTrail	`CloudTrail`
Amazon CloudWatch	`CloudWatch`
Amazon CloudWatch 日誌	`Logs`

服務	**Service** 維度的數值
Amazon CloudWatch 應用洞察	`CloudWatch Application Insights`
AWS CodeBuild	`CodeBuild`
AWS CodeCommit	`CodeCommit`
Amazon CodeGuru 分析器	`CodeGuru Profiler`
AWS CodePipeline	`CodePipeline`
AWS CodeStar	`CodeStar`
AWS CodeStar 通知	`CodeStar Notifications`
AWS CodeStar 連接	`CodeStar Connections`
Amazon Cognito 身分集區	`Cognito Identity Pools`
Amazon Cognito Sync	`Cognito Sync`
Amazon Comprehend	`Comprehend`
Amazon Comprehend Medical	`Comprehend Medical`
AWS Compute Optimizer	`ComputeOptimzier`
Amazon Connect	`Connect`
Amazon Connect Customer Profiles	`Customer Profiles`
AWS 成本和使用情況報告	`Cost and Usage Report`
AWS Cost Explorer	`Cost Explorer`
AWS Data Exchange	`Data Exchange`
AWS 資料生命週期管	`Data Lifecycle Manager`
AWS Database Migration Service	`Database Migration Service`

服務	**Service** 維度的數值
AWS DataSync	`DataSync`
AWS DeepLens	`AWS DeepLens`
Amazon Detective	`Detective`
Device Advisor	`Device Advisor`
AWS Direct Connect	`Direct Connect`
AWS Directory Service	`Directory Service`
DynamoDB Accelerator	`DynamoDBAccelerator`
Amazon EC2	`EC2`
EC2 Auto Scaling	`EC2 Auto Scaling`
Amazon Elastic Container Registry	`ECR Public`
Amazon Elastic Container Service	`ECS`
Amazon Elastic File System	`EFS`
Amazon Elastic Kubernetes Service	`EKS`
AWS Elastic Beanstalk	`Elastic Beanstalk`
Amazon Elastic Inference	`Elastic Inference`
Elastic Load Balancing	`Elastic Load Balancing`
Amazon EMR	`EMR Containers`
AWS Firewall Manager	`Firewall Manager`
Amazon FSx	`FSx`
Amazon GameLift	`GameLift`

服務	**Service** 維度的數值
AWS Glue DataBrew	`DataBrew`
Amazon Managed Grafana	`Grafana`
AWS IoT Greengrass	`Greengrass`
AWS Ground Station	`Ground Station`
AWS Health API 和通知	`AWS Health APIs And Notifications`
Amazon Interactive Video Service	`IVS`
AWS IoT Core	`IoT`
AWS IoT 一鍵式	`IoT 1-Click`
AWS IoT Events	`IoT Events`
AWS IoT RoboRunner	`IoT RoboRunner`
AWS IoT SiteWise	`IoT Sitewise`
AWS IoT Wireless	`IoT Wireless`
Amazon Kendra	`Kendra`
Amazon Keyspaces (適用於 Apache Cassandra)	`Keyspaces`
Amazon Managed Service for Apache Flink	`Kinesis Analytics`
Amazon 數據 Firehose	`Firehose`
Kinesis Video Streams	`Kinesis Video Streams`
AWS Key Management Service	`KMS`
AWS Lambda	`Lambda`
AWS Launch Wizard	`Launch Wizard`

服務	**Service** 維度的數值
Amazon Lex	`Amazon Lex`
Amazon Lightsail	`Lightsail`
Amazon Location Service	`Location`
Amazon Lookout for Vision	`Lookout for Vision`
Amazon Machine Learning	`Amazon Machine Learning`
Amazon Macie	`Macie`
Amazon Managed Blockchain（AMB）查詢	`Amazon Managed Blockchain Query`
AWS Managed Services	`AWS Managed Services`
AWS Marketplace Commerce Analytics	`Marketplace Analytics Service`
AWS Elemental MediaConnect	`MediaConnect`
AWS Elemental MediaConvert	`MediaConvert`
AWS Elemental MediaLive	`MediaLive`
AWS Elemental MediaStore	`Mediastore`
AWS Elemental MediaTailor	`MediaTailor`
AWS Mobile Hub	`Mobile Hub`
AWS Network Firewall	`Network Firewall`
AWS OpsWorks	`OpsWorks`
AWS OpsWorks 用於組態管理	`OPsWorks CM`
AWS Outposts	`Outposts`
AWS Organizations	`Organizations`

服務	**Service** 維度的數值
Amazon RDS Performance Insights	`Performance Insights`
Amazon Pinpoint	`Pinpoint`
AWS Private Certificate Authority	`Private Certificate Authority`
Amazon Managed Service for Prometheus	`Prometheus`
AWS Proton	`Proton`
Amazon Quantum Ledger Database (Amazon QLDB)	`QLDB`
Amazon RDS	`RDS`
Amazon Redshift	`Redshift Data API`
Amazon Rekognition	`Rekognition`
AWS Resource Access Manager	`Resource Access Manager`
AWS Resource Groups	`Resource Groups`
AWS Resource Groups Tagging API	`Resource Groups Tagging API`
AWS RoboMaker	`RoboMaker`
Amazon Route 53 網域	`Route 53 Domains`
Amazon Route 53 Resolver	`Route 53 Resolver`
Amazon S3	`S3`
Amazon S3 Glacier	`Amazon S3 Glacier`
Amazon SageMaker 運行	`Sagemaker`
Savings Plans	`Savings Plans`
AWS Secrets Manager	`Secrets Manager`

服務	**Service** 維度的數值
AWS Security Hub	`Security Hub`
AWS Server Migration Service	`AWS Server Migration Service`
AWS Service Catalog AppRegistry	`Service Catalog AppRegistry`
Service Quotas	`Service Quotas`
AWS Shield	`Shield`
AWS 簽署者	`Signer`
Amazon Simple Notification Service	`SNS`
Amazon Simple Email Service	`SES`
Amazon Simple Queue Service	`SQS`
身分存放區	`Identity Store`
Storage Gateway	`Storage Gateway`
AWS Support	`Support`
Amazon Simple Workflow Service	`SWF`
Amazon Textract	`Textract`
AWS IoT Things Graph	`ThingsGraph`
Amazon Timestream	`Timestream`
Amazon Transcribe	`Transcribe`
Amazon Translate	`Translate`
Amazon Transcribe 串流轉錄	`Transcribe Streaming`
AWS Transfer Family	`Transfer`

服務	**Service** 維度的數值
AWS WAF	`WAF`
Amazon WorkDocs	`Amazon WorkDocs`
Amazon WorkLink	`WorkLink`
Amazon WorkMail	`Amazon WorkMail`
Amazon WorkSpaces	`Workspaces`
AWS X-Ray	`X-Ray`

某些服務也會報告其他 API 的用量指標。若要查看 API 是否向其報告使用量度 CloudWatch，請使用 CloudWatch 主控台查看該服務在`AWS/Usage`命名空間中報告的指標。

若要查看將使用量度報告至的服務 API 清單 CloudWatch

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇 指標。
3. 在 [所有量度] 索引標籤上，選擇 [使用狀況]，然後選擇 [依 AWS 資源]
4. 在指標清單附近的搜尋方塊中，輸入服務的名稱。指標會依您輸入的服務進行篩選。

CloudWatch 使用量度

CloudWatch 收集跟踪某些 AWS 資源使用情況的指標。這些量度對應於 AWS 服務配額。追蹤這些指標可協助您主動管理配額。如需詳細資訊，請參閱 視覺化您的服務配額和設定警示。

服務配額用量指標位於 `AWS/Usage` 命名空間，每分鐘收集一次。

目前，此命名空間中唯一 CloudWatch 發佈的測量結果名稱是`CallCount`。此指標會與維度 `Resource`、`Service` 和 `Type` 一起發佈。`Resource` 維度指定要追蹤之 API 操作的名稱。例如，具有維度的`CallCount`量度`"Service": "CloudWatch"`，`"Type": "API"`並`"Resource": "PutMetricData"`指出在您的帳戶中呼叫 CloudWatch `PutMetricData` API 作業的次數。

`CallCount` 指標沒有指定的單位。指標最實用的統計資訊是 `SUM`，代表 1 分鐘期間的總操作計數。

指標

指標	描述
CallCount	在您的帳戶中執行的指定操作數目。

Dimensions (尺寸)

維度	描述
Service	包含資源的 AWS 服務名稱。對於 CloudWatch 使用狀況測量結果，此維度的值為CloudWatch 。
Class	正在追蹤的資源類別。 CloudWatch API 使用量度使用此維度的值為None。
Type	正在追蹤的資源類型。目前，當 Service 維度為 CloudWatch ，Type 的唯一有效值為 API。
Resource	API 操作的名稱。有效值包括以下項目：DeleteAlarms 、DeleteDashboards 、DescribeAlarmHistory 、DescribeAlarms 、GetDashboard 、GetMetricData 、GetMetricStatistics 、ListMetrics 、PutDashboard 和 PutMetricData

CloudWatch 教程

下列案例說明 Amazon 的使用方式 CloudWatch。在第一個案例中，您可以使用 CloudWatch主控台建立帳單警示，以追蹤您的 AWS 使用情況，並讓您知道何時超過特定支出閾值。在第二個更進階的案例中，您可以使用 AWS Command Line Interface (AWS CLI) 為名為的假設應用程式發佈單一量度。GetStarted

案例

- 監控您的預估費用
- 發佈指標

案例：使用以下方式監控估計費用 CloudWatch

在這個案例中，您會建立 Amazon CloudWatch 警示來監控您的估計費用。當您啟用 AWS 帳戶的預估費用監視時，系統會計算估計費用，並以指標資料的 CloudWatch 形式，每天多次傳送至。

帳單指標資料存放在美國東部 (維吉尼亞北部) 區域，並且會反映全球費用。此資料包括您使用的每項服務 AWS 的估計費用，以及估計的整體 AWS 費用總額。

您可以選擇在費用超過特定閾值時，透過電子郵件接收提醒。這些警示由觸發， CloudWatch 並使用 Amazon Simple Notification Service (Amazon SNS) 傳送訊息。

> **Note**
>
> 如需分析已收取 CloudWatch 費用的相關資訊，請參閱CloudWatch 帳單和成本。

任務

- 步驟 1：啟用帳單提醒
- 步驟 2：建立帳單警示
- 步驟 3：檢查警示狀態
- 步驟 4：編輯帳單警示
- 步驟 5：刪除帳單警示

步驟 1：啟用帳單提醒

您必須先啟用帳單警示，才能建立預估費用警示，以便您可以使用帳單指標資料監控預估 AWS 費用並建立警示。在啟用帳單提醒之後，將無法停用資料收集，但您可以刪除已建立的任何帳單警示。

在您第一次啟用帳單提醒之後，大約需要 15 分鐘才能查看帳單資料和設定帳單警示。

要求

- 您必須使用根使用者憑證登入，或以已獲得帳單資訊檢視許可的使用者身分登入。
- 針對合併帳單帳戶，以付款帳戶身分登入，即可看到各個連結帳戶的帳單資料。除了整合帳戶，您可以檢視各個連結帳戶的總計預估費用及各項服務預估費用。
- 在合併帳單帳戶中，只有在付款人帳戶啟用 Receive Billing Alers (接收帳單提醒) 喜好設定時，才會擷取成員連結的帳戶指標。若您變更管理/付款人帳戶的帳戶，則必須在新的管理/付款人帳戶中啟用帳單提醒。
- 該帳戶不得是 Amazon 合作夥伴網路 (APN) 的一部分，因為 APN 帳戶的計費指標並未發佈到 CloudWatch 。如需詳細資訊，請參閱 AWS 合作夥伴網路。

啟用監控預估費用

1. 請在以下位置開啟 AWS Billing 主控台。 https://console.aws.amazon.com/billing/
2. 在導覽窗格中，選擇 Billing preferences (帳單偏好設定)。
3. 依警示偏好設定選擇編輯。
4. 選擇「接收 CloudWatch 帳單通知」。
5. 選擇 Save preferences (儲存喜好設定)。

步驟 2：建立帳單警示

> **Important**
>
> 建立帳單警示前，您必須將區域設定為美國東部 (維吉尼亞北部)。帳單指標資料會存放在此區域，並且會呈現全球費用。此外，您還須為帳戶或在管理/付款人帳戶中啟用帳單提醒 (若您使用的是合併帳單)。如需詳細資訊，請參閱步驟 1：啟用帳單提醒。

在此程序中，您會建立警示，當您的預估費用 AWS 超過定義的臨界值時傳送通知。

使用 CloudWatch 主控台建立帳單警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Alarms (警示)，然後選擇 All alarms (所有警示)。
3. 選擇 Create alarm (建立警示)。
4. 選擇 Select metric (選取指標)。在 Browse (瀏覽) 中，選擇 Billing (計費)，然後選擇 Total Estimated Charge (預估費用總金額)。

   > **Note**
   >
   > 如果您未看到帳單/預估費用總金額指標，請啟用帳單警示，並將區域變更為美國東部 (維吉尼亞北部)。如需詳細資訊，請參閱 啟用帳單提醒。

5. 選取EstimatedCharges測量結果的方塊，然後選擇 「選取測量結果」。
6. 對於 Statistic (統計數字)，選擇 Maximum (最大值)。
7. 在 Period (時段) 中，選擇 6 hours (6 小時)。
8. 對於閾值類型，選擇靜態。
9. 對於無論何時如 EstimatedCharges 此。 ，選擇 [更大]。
10. 針對於...，定義您要讓警示觸發的值。例如，**200** 美元。

    EstimatedCharges指標值僅以美元 (USD) 為單位，貨幣轉換由 Amazon 服務有限責任公司提供。如需詳細資訊，請參閱什麼是 AWS Billing？ 。

    > **Note**
    >
    > 定義閾值後，預覽圖表會顯示當月的預估費用。

11. 選擇其他組態並執行下列操作：
    - 在 Datapoints to alarm (要警示的資料點) 中，指定 1 out of 1 (1 傳出 1)。
    - 在 Missing data treatment (遺失資料處理) 中，選擇 Treat missing data as missing (將遺失資料視為遺失)。
12. 選擇下一步。
13. 在通知下，確定已選取警示中。接著，指定當警示處於 ALARM 狀態時要通知的 Amazon SNS 主題。Amazon SNS 主題可以包含您的電子郵件地址，以便您在帳單金額超過自己指定的閾值時收到電子郵件。

您可以選取現有的 Amazon SNS 主題、建立新的 Amazon SNS 主題，或使用主題 ARN 來通知其他帳戶。若希望警報針對相同警示狀態或不同警示狀態傳送多個通知，請選擇 Add notification (新增通知)。

14. 選擇下一步。
15. 在 Name and description (名稱和描述) 下，輸入警示的名稱。
    - (選用) 輸入警示的描述。
16. 選擇下一步。
17. 在 Preview and create (預覽並建立) 下，請檢查組態是否正確無誤，然後選擇 Create alarm (建立警示)。

步驟 3：檢查警示狀態

現在，查看您剛才建立的帳單警示的狀態。

檢查警示狀態

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 如有必要，請將 Region (區域) 變更為美國東部 (維吉尼亞北部)。帳單指標資料存放於此區域，而且會反映全球費用。
3. 在導覽窗格中，選擇警示。
4. 選取警示旁的核取方塊。直到確認訂閱為止，它會顯示為「待確認」。確認訂閱後，請重新整理主控台以顯示更新後的狀態。

步驟 4：編輯帳單警示

例如，您可能希望將 AWS 每月花費的金額從 200 美元增加到 400 美元。您可以編輯現有的帳單警示，提高觸發警示的金額門檻。

編輯帳單警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 如有必要，請將 Region (區域) 變更為美國東部 (維吉尼亞北部)。帳單指標資料存放於此區域，而且會反映全球費用。
3. 在導覽窗格中，選擇警示。

4. 選取警示旁的核取方塊，然後選擇 Actions (動作)、Modify (修改)。
5. 針對每當月的總 AWS 費用超過時，請指定觸發警示並傳送電子郵件通知所必須超過的新金額。
6. 選擇 Save Changes (儲存變更)。

步驟 5：刪除帳單警示

如果您不再需要帳單警示，可以刪除它。

刪除帳單警示

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 如有必要，請將 Region (區域) 變更為美國東部 (維吉尼亞北部)。帳單指標資料存放於此區域，而且會反映全球費用。
3. 在導覽窗格中，選擇警示。
4. 選取警示旁的核取方塊，然後選擇 Actions (動作)、Delete (刪除)。
5. 出現確認提示時，選擇 Yes, Delete (是，刪除)。

案例:將量度發佈至 CloudWatch

在這個案例中，您可以使用 AWS Command Line Interface (AWS CLI) 發佈名為的假設應用程式的單一量度。GetStarted如果您尚未安裝和設定 AWS CLI，請參閱《使用者指南》 AWS Command Line Interface中的〈AWS Command Line Interface 使用〉進行設定。

任務

- 步驟 1：定義資料組態
- 步驟 2：將量度新增至 CloudWatch
- 步驟 3：從中獲取統計信息 CloudWatch
- 步驟 4：使用主控台檢視圖形

步驟 1：定義資料組態

在此案例中，您發佈資料點以追蹤應用程式的請求延遲。為指標和命名空間選擇符合您需求的名稱。在此範例中，為指標命名，RequestLatency並將所有資料點放入GetStarted命名空間中。

您發佈數個資料點，共同代表三個小時的延遲資料。原始資料包含分散於三小時內的 15 個請求延遲讀數。每個讀取皆以毫秒為單位：

- 第一小時：87、51、125、235
- 第二小時：121、113、189、65、89
- 第三小時：100、47、133、98、100、328

您可以將資料以單一資料點的 CloudWatch 形式發佈至，也可以作為一組彙總的資料點 (稱為統計資料集) 發佈至。您可以將指標最短彙總為一分鐘的精細程度。您可以使用四個預先定義的索引鍵 CloudWatch ，將彙總的資料點發佈成一組統計資料：SumMinimumMaximum、、和SampleCount。

您將第一小時的資料點發佈為單一資料點。對於第二小時至第三小時的資料，您每小時彙總資料點並發佈一組統計。金鑰值如下表所示。

小時	原始資料	總和	下限	最大	SampleCount
1	87				
1	51				
1	125				
1	235				
2	121, 113, 189, 65, 89	577	65	189	5
3	100, 47, 133, 98, 100, 328	806	47	328	6

步驟 2：將量度新增至 CloudWatch

在您定義資料組態之後，就可以新增資料。

若要將資料點發佈至 CloudWatch

1. 在命令提示字元中，執行下列命put-metric-data令以新增第一個小時的資料。以過去兩小時的時間戳記取代範例時間戳記，以國際標準時間 (UTC) 為準。

```
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted
 \
--timestamp 2016-10-14T20:30:00Z --value 87 --unit Milliseconds
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted
 \
--timestamp 2016-10-14T20:30:00Z --value 51 --unit Milliseconds
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted
 \
--timestamp 2016-10-14T20:30:00Z --value 125 --unit Milliseconds
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted
 \
--timestamp 2016-10-14T20:30:00Z --value 235 --unit Milliseconds
```

2. 使用第一個小時之後 1 小時的時間戳記，新增第二個小時的資料。

```
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted
 \
--timestamp 2016-10-14T21:30:00Z --statistic-values
 Sum=577,Minimum=65,Maximum=189,SampleCount=5 --unit Milliseconds
```

3. 新增第三個小時的資料，省略時間戳記以預設為目前時間。

```
aws cloudwatch put-metric-data --metric-name RequestLatency --namespace GetStarted
 \
--statistic-values Sum=806,Minimum=47,Maximum=328,SampleCount=6 --unit Milliseconds
```

步驟 3：從中獲取統計信息 CloudWatch

現在您已將量度發佈至 CloudWatch，您可以使用get-metric-statistics命令擷取以這些度量為基礎的統計資料，如下所示。請務必指定過去夠久以前的 `--start-time` 和 `--end-time`，以涵蓋您所發佈的最早時間戳記。

```
aws cloudwatch get-metric-statistics --namespace GetStarted --metric-name
 RequestLatency --statistics Average \
--start-time 2016-10-14T00:00:00Z --end-time 2016-10-15T00:00:00Z --period 60
```

下列為範例輸出：

```
{
```

```
 "Datapoints": [],
 "Label": "Request:Latency"
}
```

步驟 4：使用主控台檢視圖形

將量度發佈到之後 CloudWatch，您可以使用主 CloudWatch 控台檢視統計圖表。

在主控台檢視您的統計數字圖形

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在 Navigation (導覽) 窗格中，選擇 Metrics (指標)。
3. 在 [所有量度] 索引標籤的搜尋方塊中，輸入RequestLatency並按 Enter。
4. 選取RequestLatency測量結果的核取方塊。上方窗格將顯示指標資料的圖形。

如需更多詳細資訊，請參閱 建立指標圖形。

搭 CloudWatch 配 AWS SDK 使用

AWS 軟件開發套件（SDK）可用於許多流行的編程語言。每個 SDK 都提供 API、程式碼範例和說明文件，讓開發人員能夠更輕鬆地以偏好的語言建置應用程式。

SDK 文件	代碼範例
AWS SDK for C++	AWS SDK for C++ 程式碼範例
AWS CLI	AWS CLI 程式碼範例
AWS SDK for Go	AWS SDK for Go 程式碼範例
AWS SDK for Java	AWS SDK for Java 程式碼範例
AWS SDK for JavaScript	AWS SDK for JavaScript 程式碼範例
適用於 Kotlin 的 AWS SDK	適用於 Kotlin 的 AWS SDK 程式碼範例
AWS SDK for .NET	AWS SDK for .NET 程式碼範例
AWS SDK for PHP	AWS SDK for PHP 程式碼範例
AWS Tools for PowerShell	PowerShell 程式碼範例的工具
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 程式碼範例
AWS SDK for Ruby	AWS SDK for Ruby 程式碼範例
適用於 Rust 的 AWS SDK	適用於 Rust 的 AWS SDK 程式碼範例
適用於 SAP ABAP 的 AWS SDK	適用於 SAP ABAP 的 AWS SDK 程式碼範例
適用於 Swift 的 AWS SDK	適用於 Swift 的 AWS SDK 程式碼範例

如需特定的範例 CloudWatch，請參閱 CloudWatch 使用 AWS SDK 的程式碼範例。

可用性範例

找不到所需的內容嗎？ 請使用本頁面底部的提供意見回饋連結申請程式碼範例。

CloudWatch 使用 AWS SDK 的程式碼範例

下列程式碼範例顯示如何搭 CloudWatch 配 AWS 軟體開發套件 (SDK) 使用。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境和跨服務範例中查看內容中的動作。

Scenarios (案例) 是向您展示如何呼叫相同服務中的多個函數來完成特定任務的程式碼範例。

Cross-service examples (跨服務範例) 是跨多個 AWS 服務執行的應用程式範例。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含入門相關資訊和舊版 SDK 的詳細資訊。

開始使用

你好 CloudWatch

下列程式碼範例會示範如何開始使用 CloudWatch。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace CloudWatchActions;

public static class HelloCloudWatch
{
```

```
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the Amazon CloudWatch service.
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonCloudWatch>()
            ).Build();

        // Now the client is available for injection.
        var cloudWatchClient =
 host.Services.GetRequiredService<IAmazonCloudWatch>();

        // You can use await and any of the async methods to get a response.
        var metricNamespace = "AWS/Billing";
        var response = await cloudWatchClient.ListMetricsAsync(new
 ListMetricsRequest
        {
            Namespace = metricNamespace
        });
        Console.WriteLine($"Hello Amazon CloudWatch! Following are some metrics
 available in the {metricNamespace} namespace:");
        Console.WriteLine();
        foreach (var metric in response.Metrics.Take(5))
        {
            Console.WriteLine($"\tMetric: {metric.MetricName}");
            Console.WriteLine($"\tNamespace: {metric.Namespace}");
            Console.WriteLine($"\tDimensions: {string.Join(", ",
 metric.Dimensions.Select(m => $"{m.Name}:{m.Value}"))}");
            Console.WriteLine();
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListMetrics中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloService {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <namespace>\s

                Where:
                  namespace - The namespace to filter against (for example, AWS/
EC2).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String namespace = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        listMets(cw, namespace);
        cw.close();
    }

    public static void listMets(CloudWatchClient cw, String namespace) {
        try {
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                    .flatMap(r -> r.metrics().stream())
                    .forEach(metrics -> System.out.println(" Retrieved metric is:
 " + metrics.metricName()));

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考ListMetrics中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
           <namespace>
        Where:
           namespace - The namespace to filter against (for example, AWS/EC2).
    """

    if (args.size != 1) {
        println(usage)
        exitProcess(0)
    }

    val namespace = args[0]
    listAllMets(namespace)
}

suspend fun listAllMets(namespaceVal: String?) {
    val request = ListMetricsRequest {
        namespace = namespaceVal
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.listMetricsPaginated(request)
            .transform { it.metrics?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.metricName}")
                println("Namespace is ${obj.namespace}")
            }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK ListMetrics中的 Kotlin API 參考。

程式碼範例

- CloudWatch 使用 AWS SDK 的動作
  - 搭DeleteAlarms配 AWS 開發套件或 CLI 使用
  - 搭DeleteAnomalyDetector配 AWS 開發套件或 CLI 使用
  - 搭DeleteDashboards配 AWS 開發套件或 CLI 使用
  - 搭DescribeAlarmHistory配 AWS 開發套件或 CLI 使用
  - 搭DescribeAlarms配 AWS 開發套件或 CLI 使用
  - 搭DescribeAlarmsForMetric配 AWS 開發套件或 CLI 使用
  - 搭DescribeAnomalyDetectors配 AWS 開發套件或 CLI 使用
  - 搭DisableAlarmActions配 AWS 開發套件或 CLI 使用
  - 搭EnableAlarmActions配 AWS 開發套件或 CLI 使用
  - 搭GetDashboard配 AWS 開發套件或 CLI 使用
  - 搭GetMetricData配 AWS 開發套件或 CLI 使用
  - 搭GetMetricStatistics配 AWS 開發套件或 CLI 使用
  - 搭GetMetricWidgetImage配 AWS 開發套件或 CLI 使用
  - 搭ListDashboards配 AWS 開發套件或 CLI 使用
  - 搭ListMetrics配 AWS 開發套件或 CLI 使用
  - 搭PutAnomalyDetector配 AWS 開發套件或 CLI 使用
  - 搭PutDashboard配 AWS 開發套件或 CLI 使用
  - 搭PutMetricAlarm配 AWS 開發套件或 CLI 使用
  - 搭PutMetricData配 AWS 開發套件或 CLI 使用
- CloudWatch 使用 AWS SDK 的案例
  - 使用 AWS SDK 開始使用 CloudWatch 警示
  - 使用 AWS SDK 開始使用 CloudWatch 指標、儀表板和警示
  - 使用 AWS SDK 管理 CloudWatch 指標和警示
- 使用 SDK 的跨服務 CloudWatch 範 AWS 例
  - 使用開發套件監控 Amazon DynamoDB 的效能 AWS

CloudWatch 使用 AWS SDK 的動作

下列程式碼範例示範如何使用 AWS SDK 執 CloudWatch行個別動作。這些摘錄會呼叫 CloudWatch API，是來自必須在內容中執行的大型程式碼摘錄。每個範例都包含一個連結 GitHub，您可以在其中找到設定和執行程式碼的指示。

下列範例僅包含最常使用的動作。如需完整清單，請參閱 Amazon CloudWatch API 參考資料。

範例

- 搭DeleteAlarms配 AWS 開發套件或 CLI 使用
- 搭DeleteAnomalyDetector配 AWS 開發套件或 CLI 使用
- 搭DeleteDashboards配 AWS 開發套件或 CLI 使用
- 搭DescribeAlarmHistory配 AWS 開發套件或 CLI 使用
- 搭DescribeAlarms配 AWS 開發套件或 CLI 使用
- 搭DescribeAlarmsForMetric配 AWS 開發套件或 CLI 使用
- 搭DescribeAnomalyDetectors配 AWS 開發套件或 CLI 使用
- 搭DisableAlarmActions配 AWS 開發套件或 CLI 使用
- 搭EnableAlarmActions配 AWS 開發套件或 CLI 使用
- 搭GetDashboard配 AWS 開發套件或 CLI 使用
- 搭GetMetricData配 AWS 開發套件或 CLI 使用
- 搭GetMetricStatistics配 AWS 開發套件或 CLI 使用
- 搭GetMetricWidgetImage配 AWS 開發套件或 CLI 使用
- 搭ListDashboards配 AWS 開發套件或 CLI 使用
- 搭ListMetrics配 AWS 開發套件或 CLI 使用
- 搭PutAnomalyDetector配 AWS 開發套件或 CLI 使用
- 搭PutDashboard配 AWS 開發套件或 CLI 使用
- 搭PutMetricAlarm配 AWS 開發套件或 CLI 使用
- 搭PutMetricData配 AWS 開發套件或 CLI 使用

搭`DeleteAlarms`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`DeleteAlarms`。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用警示
- 開始使用指標、儀表板和警示
- 管理指標和警示

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/// <summary>
/// Delete a list of alarms from CloudWatch.
/// </summary>
/// <param name="alarmNames">A list of names of alarms to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteAlarms(List<string> alarmNames)
{
    var deleteAlarmsResult = await _amazonCloudWatch.DeleteAlarmsAsync(
        new DeleteAlarmsRequest()
        {
            AlarmNames = alarmNames
        });

    return deleteAlarmsResult.HttpStatusCode == HttpStatusCode.OK;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DeleteAlarms中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

包括必需的檔案。

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DeleteAlarmsRequest.h>
#include <iostream>
```

刪除警示。

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::DeleteAlarmsRequest request;
        request.AddAlarmNames(alarm_name);

        auto outcome = cw.DeleteAlarms(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to delete CloudWatch alarm:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully deleted CloudWatch alarm " << alarm_name
                << std::endl;
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考DeleteAlarms中的。

CLI

AWS CLI

刪除警示

以下示例使用命delete-alarms令刪除名為「my CloudWatch alarm」的 Amazon 警報：

```
aws cloudwatch delete-alarms --alarm-names myalarm
```

輸出：

```
This command returns to the prompt if successful.
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteAlarms中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
public class DeleteAlarm {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to delete (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarmName = args[0];
        Region region = Region.US_EAST_2;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        deleteCWAlarm(cw, alarmName);
        cw.close();
    }

    public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
        try {
            DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
                    .alarmNames(alarmName)
                    .build();

            cw.deleteAlarms(request);
            System.out.printf("Successfully deleted alarm %s", alarmName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DeleteAlarms中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

匯入 SDK 和用戶端模組，然後呼叫 API。

```
import { DeleteAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DeleteAlarmsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

在單獨的模組中建立用戶端並將其匯出。

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考DeleteAlarms中的。

適用於 JavaScript (v2) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

匯入 SDK 和用戶端模組，然後呼叫 API。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmNames: ["Web_Server_CPU_Utilization"],
};

cw.deleteAlarms(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考DeleteAlarms中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun deleteAlarm(alarmNameVal: String) {
    val request = DeleteAlarmsRequest {
        alarmNames = listOf(alarmNameVal)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DeleteAlarms中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
```

```
        """
        self.cloudwatch_resource = cloudwatch_resource


    def delete_metric_alarms(self, metric_namespace, metric_name):
        """
        Deletes all of the alarms that are currently watching the specified
metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace,
metric_name)
            metric.alarms.delete()
            logger.info(
                "Deleted alarms for metric %s.%s.", metric_namespace, metric_name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete alarms for metric %s.%s.",
                metric_namespace,
                metric_name,
            )
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DeleteAlarms中的 Python (博托 3) API 參考。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    TRY.
        lo_cwt->deletealarms(
          it_alarmnames = it_alarm_names
        ).
        MESSAGE 'Alarms deleted.' TYPE 'I'.
      CATCH /aws1/cx_cwtresourcenotfound .
        MESSAGE 'Resource being accessed is not found.' TYPE 'E'.
    ENDTRY.
```

- 如需 API 詳細資訊，請參閱 AWS SDK DeleteAlarms中的 SAP ABAP API 參考資料。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteAnomalyDetector配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteAnomalyDetector。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用指標、儀表板和警示

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Delete a single metric anomaly detector.
    /// </summary>
    /// <param name="anomalyDetector">The anomaly detector to delete.</param>
    /// <returns>True if successful.</returns>
```

```
   public async Task<bool> DeleteAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
   {
       var deleteAnomalyDetectorResponse = await
_amazonCloudWatch.DeleteAnomalyDetectorAsync(
           new DeleteAnomalyDetectorRequest()
           {
               SingleMetricAnomalyDetector = anomalyDetector
           });

       return deleteAnomalyDetectorResponse.HttpStatusCode == HttpStatusCode.OK;
   }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DeleteAnomalyDetector中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   public static void deleteAnomalyDetector(CloudWatchClient cw, String
fileName) {
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
           String customMetricName =
rootNode.findValue("customMetricName").asText();

           SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                   .metricName(customMetricName)
```

```
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

            DeleteAnomalyDetectorRequest request =
 DeleteAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

            cw.deleteAnomalyDetector(request);
            System.out.println("Successfully deleted the Anomaly Detector.");

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DeleteAnomalyDetector中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal = SingleMetricAnomalyDetector {
```

```
        metricName = customMetricName
        namespace = customMetricNamespace
        stat = "Maximum"
    }

    val request = DeleteAnomalyDetectorRequest {
        singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DeleteAnomalyDetector中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteDashboards配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteDashboards。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用指標、儀表板和警示

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Delete a list of CloudWatch dashboards.
    /// </summary>
    /// <param name="dashboardNames">List of dashboard names to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDashboards(List<string> dashboardNames)
    {
        var deleteDashboardsResponse = await
_amazonCloudWatch.DeleteDashboardsAsync(
            new DeleteDashboardsRequest()
            {
                DashboardNames = dashboardNames
            });

        return deleteDashboardsResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考[DeleteDashboards](中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫]。

```
    public static void deleteDashboard(CloudWatchClient cw, String dashboardName)
{
        try {
            DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
                    .dashboardNames(dashboardName)
                    .build();
            cw.deleteDashboards(dashboardsRequest);
            System.out.println(dashboardName + " was successfully deleted.");

        } catch (CloudWatchException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DeleteDashboards中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest = DeleteDashboardsRequest {
        dashboardNames = listOf(dashboardName)
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DeleteDashboards中的 Kotlin API 參考。

PowerShell

適用的工具 PowerShell

範例 1：刪除指定的儀表板，在繼續之前促進確認。若要略過確認，請將-Force 參數新增至指令。

```
Remove-CWDashboard -DashboardName Dashboard1
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteDashboards式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DescribeAlarmHistory配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DescribeAlarmHistory。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用指標、儀表板和警示

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Describe the history of an alarm for a number of days in the past.
    /// </summary>
    /// <param name="alarmName">The name of the alarm.</param>
    /// <param name="historyDays">The number of days in the past.</param>
    /// <returns>The list of alarm history data.</returns>
    public async Task<List<AlarmHistoryItem>> DescribeAlarmHistory(string
alarmName, int historyDays)
    {
        List<AlarmHistoryItem> alarmHistory = new List<AlarmHistoryItem>();
        var paginatedAlarmHistory =
_amazonCloudWatch.Paginators.DescribeAlarmHistory(
            new DescribeAlarmHistoryRequest()
            {
                AlarmName = alarmName,
```

```
                EndDateUtc = DateTime.UtcNow,
                HistoryItemType = HistoryItemType.StateUpdate,
                StartDateUtc = DateTime.UtcNow.AddDays(-historyDays)
            });

        await foreach (var data in paginatedAlarmHistory.AlarmHistoryItems)
        {
            alarmHistory.Add(data);
        }
        return alarmHistory;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考[DescribeAlarmHistory](#)中的。

CLI

AWS CLI

若要擷取警示的歷史記錄

以下示例使用命describe-alarm-history令來檢索名為「myalarm」的 Amazon CloudWatch 警報的歷史記錄：

```
aws cloudwatch describe-alarm-history --alarm-name "myalarm" --history-item-type
 StateUpdate
```

輸出：

```
{
    "AlarmHistoryItems": [
        {
            "Timestamp": "2014-04-09T18:59:06.442Z",
            "HistoryItemType": "StateUpdate",
            "AlarmName": "myalarm",
            "HistoryData": "{\"version\":\"1.0\",\"oldState\":{\"stateValue
\":\"ALARM\",\"stateReason\":\"testing purposes\"},\"newState\":{\"stateValue
\":\"OK\",\"stateReason\":\"Threshold Crossed: 2 datapoints were not
 greater than the threshold (70.0). The most recent datapoints: [38.958,
 40.292].\",\"stateReasonData\":{\"version\":\"1.0\",\"queryDate\":
\"2014-04-09T18:59:06.419+0000\",\"startDate\":\"2014-04-09T18:44:00.000+0000\",
```

```
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[38.958,40.292],
\"threshold\":70.0}}}",
            "HistorySummary": "Alarm updated from ALARM to OK"
        },
        {
            "Timestamp": "2014-04-09T18:59:05.805Z",
            "HistoryItemType": "StateUpdate",
            "AlarmName": "myalarm",
            "HistoryData": "{\"version\":\"1.0\",\"oldState\":{\"stateValue
\":\"OK\",\"stateReason\":\"Threshold Crossed: 2 datapoints were
 not greater than the threshold (70.0). The most recent datapoints:
 [38.839999999999996, 39.714].\",\"stateReasonData\":{\"version\":
\"1.0\",\"queryDate\":\"2014-03-11T22:45:41.569+0000\",\"startDate\":
\"2014-03-11T22:30:00.000+0000\",\"statistic\":\"Average\",\"period\":300,
\"recentDatapoints\":[38.839999999999996,39.714],\"threshold\":70.0}},\"newState
\":{\"stateValue\":\"ALARM\",\"stateReason\":\"testing purposes\"}}",
            "HistorySummary": "Alarm updated from OK to ALARM"
        }
    ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[DescribeAlarmHistory](#)中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
   public static void getAlarmHistory(CloudWatchClient cw, String fileName,
String date) {
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String alarmName = rootNode.findValue("exampleAlarmName").asText();
```

```
            Instant start = Instant.parse(date);
            Instant endDate = Instant.now();
            DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                    .startDate(start)
                    .endDate(endDate)
                    .alarmName(alarmName)
                    .historyItemType(HistoryItemType.ACTION)
                    .build();

            DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
            List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
            if (historyItems.isEmpty()) {
                System.out.println("No alarm history data found for " + alarmName
+ ".");
            } else {
                for (AlarmHistoryItem item : historyItems) {
                    System.out.println("History summary: " +
item.historySummary());
                    System.out.println("Time stamp: " + item.timestamp());
                }
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[DescribeAlarmHistory](#)中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
suspend fun getAlarmHistory(fileName: String, date: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest = DescribeAlarmHistoryRequest {
        startDate = aws.smithy.kotlin.runtime.time.Instant(start)
        endDate = aws.smithy.kotlin.runtime.time.Instant(endDateVal)
        alarmName = alarmNameVal
        historyItemType = HistoryItemType.Action
    }

    CloudWatchClient { credentialsProvider = EnvironmentCredentialsProvider();
 region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
                for (item in historyItems) {
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DescribeAlarmHistory中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**DescribeAlarms**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DescribeAlarms。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用警示
- 開始使用指標、儀表板和警示

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Describe the current alarms, optionally filtered by state.
    /// </summary>
    /// <param name="stateValue">Optional filter for alarm state.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarms(StateValue? stateValue =
null)
    {
        List<MetricAlarm> alarms = new List<MetricAlarm>();
        var paginatedDescribeAlarms =
_amazonCloudWatch.Paginators.DescribeAlarms(
            new DescribeAlarmsRequest()
            {
                StateValue = stateValue
            });

        await foreach (var data in paginatedDescribeAlarms.MetricAlarms)
        {
            alarms.Add(data);
        }
        return alarms;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DescribeAlarms中的。

CLI

AWS CLI

列出有關警示的資訊

下列範例使用 describe-alarms 命令提供名為 "myalarm" 的警示相關資訊：

```
aws cloudwatch describe-alarms --alarm-names "myalarm"
```

輸出：

```
{
    "MetricAlarms": [
        {
            "EvaluationPeriods": 2,
            "AlarmArn": "arn:aws:cloudwatch:us-
east-1:123456789012:alarm:myalarm",
            "StateUpdatedTimestamp": "2014-04-09T18:59:06.442Z",
            "AlarmConfigurationUpdatedTimestamp": "2012-12-27T00:49:54.032Z",
            "ComparisonOperator": "GreaterThanThreshold",
            "AlarmActions": [
                "arn:aws:sns:us-east-1:123456789012:myHighCpuAlarm"
            ],
            "Namespace": "AWS/EC2",
            "AlarmDescription": "CPU usage exceeds 70 percent",
            "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2014-04-09T18:59:06.419+0000\",\"startDate\":\"2014-04-09T18:44:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[38.958,40.292],
\"threshold\":70.0}",
            "Period": 300,
            "StateValue": "OK",
            "Threshold": 70.0,
            "AlarmName": "myalarm",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-0c986c72"
                }
            ],
            "Statistic": "Average",
            "StateReason": "Threshold Crossed: 2 datapoints were not greater than
 the threshold (70.0). The most recent datapoints: [38.958, 40.292].",
```

```
            "InsufficientDataActions": [],
            "OKActions": [],
            "ActionsEnabled": true,
            "MetricName": "CPUUtilization"
        }
    ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DescribeAlarms中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    public static void describeAlarms(CloudWatchClient cw) {
        try {
            List<AlarmType> typeList = new ArrayList<>();
            typeList.add(AlarmType.METRIC_ALARM);

            DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
                    .alarmTypes(typeList)
                    .maxRecords(10)
                    .build();

            DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
            List<MetricAlarm> alarmList = response.metricAlarms();
            for (MetricAlarm alarm : alarmList) {
                System.out.println("Alarm name: " + alarm.alarmName());
                System.out.println("Alarm description: " +
 alarm.alarmDescription());
            }
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DescribeAlarms中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest = DescribeAlarmsRequest {
        alarmTypes = typeList
        maxRecords = 10
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DescribeAlarms中的 Kotlin API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
require "aws-sdk-cloudwatch"

# Lists the names of available Amazon CloudWatch alarms.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   list_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def list_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms
  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts alarm.alarm_name
    end
  else
    puts "No alarms found."
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考DescribeAlarms中的。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    TRY.
        oo_result = lo_cwt->describealarms(                " oo_result is
 returned for testing purposes. "
          it_alarmnames = it_alarm_names
        ).
        MESSAGE 'Alarms retrieved.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- 如需 API 詳細資訊，請參閱 AWS SDK DescribeAlarms中的 SAP ABAP API 參考資料。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`DescribeAlarmsForMetric`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`DescribeAlarmsForMetric`。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用指標、儀表板和警示
- 管理指標和警示

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Describe the current alarms for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarmsForMetric(string
metricNamespace, string metricName)
    {
        var alarmsResult = await _amazonCloudWatch.DescribeAlarmsForMetricAsync(
            new DescribeAlarmsForMetricRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName
            });

        return alarmsResult.MetricAlarms;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DescribeAlarmsForMetric中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

包括必需的檔案。

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DescribeAlarmsRequest.h>
#include <aws/monitoring/model/DescribeAlarmsResult.h>
#include <iomanip>
#include <iostream>
```

說明該警示。

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::DescribeAlarmsRequest request;
        request.SetMaxRecords(1);

        bool done = false;
        bool header = false;
        while (!done)
        {
            auto outcome = cw.DescribeAlarms(request);
            if (!outcome.IsSuccess())
            {
                std::cout << "Failed to describe CloudWatch alarms:" <<
                    outcome.GetError().GetMessage() << std::endl;
                break;
            }

            if (!header)
            {
                std::cout << std::left <<
                    std::setw(32) << "Name" <<
                    std::setw(64) << "Arn" <<
                    std::setw(64) << "Description" <<
                    std::setw(20) << "LastUpdated" <<
                    std::endl;
                header = true;
            }

            const auto &alarms = outcome.GetResult().GetMetricAlarms();
            for (const auto &alarm : alarms)
            {
                std::cout << std::left <<
```

```
                    std::setw(32) << alarm.GetAlarmName() <<
                    std::setw(64) << alarm.GetAlarmArn() <<
                    std::setw(64) << alarm.GetAlarmDescription() <<
                    std::setw(20) <<
                    alarm.GetAlarmConfigurationUpdatedTimestamp().ToGmtString(
                        SIMPLE_DATE_FORMAT_STR) <<
                    std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考DescribeAlarmsForMetric中的。

CLI

AWS CLI

顯示與指標相關聯的警示資訊

下列範例使用 `describe-alarms-for-metric` 命令，顯示與 Amazon EC2 CPUUtilization 指標和 ID 為 i-0c986c72 的執行個體相關聯之任何警示的資訊：

```
aws cloudwatch describe-alarms-for-metric --metric-name CPUUtilization --
namespace AWS/EC2 --dimensions Name=InstanceId,Value=i-0c986c72
```

輸出：

```
{
    "MetricAlarms": [
        {
            "EvaluationPeriods": 10,
            "AlarmArn": "arn:aws:cloudwatch:us-
east-1:111122223333:alarm:myHighCpuAlarm2",
            "StateUpdatedTimestamp": "2013-10-30T03:03:51.479Z",
            "AlarmConfigurationUpdatedTimestamp": "2013-10-30T03:03:50.865Z",
            "ComparisonOperator": "GreaterThanOrEqualToThreshold",
            "AlarmActions": [
                "arn:aws:sns:us-east-1:111122223333:NotifyMe"
```

```
            ],
            "Namespace": "AWS/EC2",
            "AlarmDescription": "CPU usage exceeds 70 percent",
            "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2013-10-30T03:03:51.479+0000\",\"startDate\":\"2013-10-30T02:08:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":
[40.698,39.612,42.432,39.796,38.816,42.28,42.854,40.088,40.760000000000005,41.316],
\"threshold\":70.0}",
            "Period": 300,
            "StateValue": "OK",
            "Threshold": 70.0,
            "AlarmName": "myHighCpuAlarm2",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-0c986c72"
                }
            ],
            "Statistic": "Average",
            "StateReason": "Threshold Crossed: 10 datapoints were not
 greater than or equal to the threshold (70.0). The most recent datapoints:
 [40.760000000000005, 41.316].",
            "InsufficientDataActions": [],
            "OKActions": [],
            "ActionsEnabled": true,
            "MetricName": "CPUUtilization"
        },
        {
            "EvaluationPeriods": 2,
            "AlarmArn": "arn:aws:cloudwatch:us-
east-1:111122223333:alarm:myHighCpuAlarm",
            "StateUpdatedTimestamp": "2014-04-09T18:59:06.442Z",
            "AlarmConfigurationUpdatedTimestamp": "2014-04-09T22:26:05.958Z",
            "ComparisonOperator": "GreaterThanThreshold",
            "AlarmActions": [
                "arn:aws:sns:us-east-1:111122223333:HighCPUAlarm"
            ],
            "Namespace": "AWS/EC2",
            "AlarmDescription": "CPU usage exceeds 70 percent",
            "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":
\"2014-04-09T18:59:06.419+0000\",\"startDate\":\"2014-04-09T18:44:00.000+0000\",
\"statistic\":\"Average\",\"period\":300,\"recentDatapoints\":[38.958,40.292],
\"threshold\":70.0}",
            "Period": 300,
```

```
            "StateValue": "OK",
            "Threshold": 70.0,
            "AlarmName": "myHighCpuAlarm",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-0c986c72"
                }
            ],
            "Statistic": "Average",
            "StateReason": "Threshold Crossed: 2 datapoints were not greater than
 the threshold (70.0). The most recent datapoints: [38.958, 40.292].",
            "InsufficientDataActions": [],
            "OKActions": [],
            "ActionsEnabled": false,
            "MetricName": "CPUUtilization"
        }
    ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DescribeAlarmsForMetric中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    public static void checkForMetricAlarm(CloudWatchClient cw, String fileName)
{
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
```

```
            String customMetricName =
rootNode.findValue("customMetricName").asText();
            boolean hasAlarm = false;
            int retries = 10;

            DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            while (!hasAlarm && retries > 0) {
                DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
                hasAlarm = response.hasMetricAlarms();
                retries--;
                Thread.sleep(20000);
                System.out.println(".");
            }
            if (!hasAlarm)
                System.out.println("No Alarm state found for " + customMetricName
+ " after 10 retries.");
            else
                System.out.println("Alarm state found for " + customMetricName +
".");

        } catch (CloudWatchException | IOException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DescribeAlarmsForMetric中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

匯入 SDK 和用戶端模組，然後呼叫 API。

```
import { DescribeAlarmsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DescribeAlarmsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

在單獨的模組中建立用戶端並將其匯出。

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考DescribeAlarmsForMetric中的。

適用於 JavaScript (v2) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

cw.describeAlarms({ StateValue: "INSUFFICIENT_DATA" }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    // List the names of all current alarms in the console
    data.MetricAlarms.forEach(function (item, index, array) {
      console.log(item.AlarmName);
    });
  }
});
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考DescribeAlarmsForMetric中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest = DescribeAlarmsForMetricRequest {
        metricName = customMetricName
        namespace = customMetricNamespace
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
        if (!hasAlarm) println("No Alarm state found for $customMetricName after
 10 retries.") else println("Alarm state found for $customMetricName.")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DescribeAlarmsForMetric中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def get_metric_alarms(self, metric_namespace, metric_name):
        """
        Gets the alarms that are currently watching the specified metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :returns: An iterator that yields the alarms.
        """
        metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
        alarm_iter = metric.alarms.all()
        logger.info("Got alarms for metric %s.%s.", metric_namespace,
 metric_name)
        return alarm_iter
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DescribeAlarmsForMetric中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @example
#   describe_metric_alarms(Aws::CloudWatch::Client.new(region: 'us-east-1'))
def describe_metric_alarms(cloudwatch_client)
  response = cloudwatch_client.describe_alarms

  if response.metric_alarms.count.positive?
    response.metric_alarms.each do |alarm|
      puts "-" * 16
      puts "Name:           " + alarm.alarm_name
      puts "State value:    " + alarm.state_value
      puts "State reason:   " + alarm.state_reason
      puts "Metric:         " + alarm.metric_name
      puts "Namespace:      " + alarm.namespace
      puts "Statistic:      " + alarm.statistic
      puts "Period:         " + alarm.period.to_s
      puts "Unit:           " + alarm.unit.to_s
      puts "Eval. periods:  " + alarm.evaluation_periods.to_s
      puts "Threshold:      " + alarm.threshold.to_s
      puts "Comp. operator: " + alarm.comparison_operator

      if alarm.key?(:ok_actions) && alarm.ok_actions.count.positive?
        puts "OK actions:"
        alarm.ok_actions.each do |a|
          puts "  " + a
        end
      end

      if alarm.key?(:alarm_actions) && alarm.alarm_actions.count.positive?
```

```
        puts "Alarm actions:"
        alarm.alarm_actions.each do |a|
          puts "  " + a
        end
      end

      if alarm.key?(:insufficient_data_actions) &&
          alarm.insufficient_data_actions.count.positive?
        puts "Insufficient data actions:"
        alarm.insufficient_data_actions.each do |a|
          puts "  " + a
        end
      end

      puts "Dimensions:"
      if alarm.key?(:dimensions) && alarm.dimensions.count.positive?
        alarm.dimensions.each do |d|
          puts "  Name: " + d.name + ", Value: " + d.value
        end
      else
        puts "  None for this alarm."
      end
    end
  else
    puts "No alarms found."
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end

# Example usage:
def run_me
  region = ""

  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:   ruby cw-ruby-example-show-alarms.rb REGION"
    puts "Example: ruby cw-ruby-example-show-alarms.rb us-east-1"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    region = "us-east-1"
  # Otherwise, use the values as specified at the command prompt.
  else
```

```
    region = ARGV[0]
  end

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)
  puts "Available alarms:"
  describe_metric_alarms(cloudwatch_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考DescribeAlarmsForMetric中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DescribeAnomalyDetectors配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DescribeAnomalyDetectors。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用指標、儀表板和警示

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Describe anomaly detectors for a metric and namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
```

```
   /// <param name="metricName">The metric of the anomaly detectors.</param>
   /// <returns>The list of detectors.</returns>
   public async Task<List<AnomalyDetector>> DescribeAnomalyDetectors(string
metricNamespace, string metricName)
   {
       List<AnomalyDetector> detectors = new List<AnomalyDetector>();
       var paginatedDescribeAnomalyDetectors =
_amazonCloudWatch.Paginators.DescribeAnomalyDetectors(
           new DescribeAnomalyDetectorsRequest()
           {
               MetricName = metricName,
               Namespace = metricNamespace
           });

       await foreach (var data in
paginatedDescribeAnomalyDetectors.AnomalyDetectors)
       {
           detectors.Add(data);
       }

       return detectors;
   }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DescribeAnomalyDetectors中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
```

```
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();
            DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                    .maxResults(10)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
            List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
            for (AnomalyDetector detector : anomalyDetectorList) {
                System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
                System.out.println("State: " + detector.stateValue());
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[DescribeAnomalyDetectors](#)中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest = DescribeAnomalyDetectorsRequest {
        maxResults = 10
        metricName = customMetricName
        namespace = customMetricNamespace
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DescribeAnomalyDetectors中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DisableAlarmActions配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DisableAlarmActions。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用警示
- 管理指標和警示

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Disable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DisableAlarmActions(List<string> alarmNames)
    {
        var disableAlarmActionsResult = await
_amazonCloudWatch.DisableAlarmActionsAsync(
            new DisableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return disableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DisableAlarmActions中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

包括必需的檔案。

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DisableAlarmActionsRequest.h>
#include <iostream>
```

停用警示動作。

```
        Aws::CloudWatch::CloudWatchClient cw;

        Aws::CloudWatch::Model::DisableAlarmActionsRequest
 disableAlarmActionsRequest;
        disableAlarmActionsRequest.AddAlarmNames(alarm_name);

        auto disableAlarmActionsOutcome =
 cw.DisableAlarmActions(disableAlarmActionsRequest);
        if (!disableAlarmActionsOutcome.IsSuccess())
        {
            std::cout << "Failed to disable actions for alarm " << alarm_name <<
                ": " << disableAlarmActionsOutcome.GetError().GetMessage() <<
                std::endl;
        }
        else
        {
            std::cout << "Successfully disabled actions for alarm " <<
                alarm_name << std::endl;
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考DisableAlarmActions中的。

CLI

AWS CLI

停用警示的動作

下列範例使用 `disable-alarm-actions` 命令來停用名為 myalarm 之警示的所有動作：

```
aws cloudwatch disable-alarm-actions --alarm-names myalarm
```

如果成功，此命令會回到提示字元。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DisableAlarmActions中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import
 software.amazon.awssdk.services.cloudwatch.model.DisableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DisableAlarmActions {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to disable (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String alarmName = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        disableActions(cw, alarmName);
        cw.close();
    }

    public static void disableActions(CloudWatchClient cw, String alarmName) {
        try {
            DisableAlarmActionsRequest request =
 DisableAlarmActionsRequest.builder()
                    .alarmNames(alarmName)
                    .build();

            cw.disableAlarmActions(request);
            System.out.printf("Successfully disabled actions on alarm %s",
 alarmName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DisableAlarmActions中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

匯入 SDK 和用戶端模組，然後呼叫 API。

```
import { DisableAlarmActionsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new DisableAlarmActionsCommand({
    AlarmNames: process.env.CLOUDWATCH_ALARM_NAME, // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

在單獨的模組中建立用戶端並將其匯出。

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考DisableAlarmActions中的。

適用於 JavaScript (v2) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

匯入 SDK 和用戶端模組，然後呼叫 API。

```
// Load the AWS SDK for Node.js
```

```
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

cw.disableAlarmActions(
  { AlarmNames: ["Web_Server_CPU_Utilization"] },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考DisableAlarmActions中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun disableActions(alarmName: String) {

    val request = DisableAlarmActionsRequest {
        alarmNames = listOf(alarmName)
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.disableAlarmActions(request)
        println("Successfully disabled actions on alarm $alarmName")
    }
```

```
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DisableAlarmActions中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def enable_alarm_actions(self, alarm_name, enable):
        """
        Enables or disables actions on the specified alarm. Alarm actions can be
        used to send notifications or automate responses when an alarm enters a
        particular state.

        :param alarm_name: The name of the alarm.
        :param enable: When True, actions are enabled for the alarm. Otherwise,
 they
                       disabled.
        """
        try:
            alarm = self.cloudwatch_resource.Alarm(alarm_name)
            if enable:
                alarm.enable_actions()
            else:
                alarm.disable_actions()
```

```
            logger.info(
                "%s actions for alarm %s.",
                "Enabled" if enable else "Disabled",
                alarm_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't %s actions alarm %s.",
                "enable" if enable else "disable",
                alarm_name,
            )
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DisableAlarmActions中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
# Disables an alarm in Amazon CloudWatch.
#
# Prerequisites.
#
# - The alarm to disable.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm to disable.
# @return [Boolean] true if the alarm was disabled; otherwise, false.
# @example
#   exit 1 unless alarm_actions_disabled?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
```

```
#     'ObjectsInBucket'
#   )
def alarm_actions_disabled?(cloudwatch_client, alarm_name)
  cloudwatch_client.disable_alarm_actions(alarm_names: [alarm_name])
  return true
rescue StandardError => e
  puts "Error disabling alarm actions: #{e.message}"
  return false
end

# Example usage:
def run_me
  alarm_name = "ObjectsInBucket"
  alarm_description = "Objects exist in this bucket for more than 1 day."
  metric_name = "NumberOfObjects"
  # Notify this Amazon Simple Notification Service (Amazon SNS) topic when
  # the alarm transitions to the ALARM state.
  alarm_actions = ["arn:aws:sns:us-
east-1:111111111111:Default_CloudWatch_Alarms_Topic"]
  namespace = "AWS/S3"
  statistic = "Average"
  dimensions = [
    {
      name: "BucketName",
      value: "doc-example-bucket"
    },
    {
      name: "StorageType",
      value: "AllStorageTypes"
    }
  ]
  period = 86_400 # Daily (24 hours * 60 minutes * 60 seconds = 86400 seconds).
  unit = "Count"
  evaluation_periods = 1 # More than one day.
  threshold = 1 # One object.
  comparison_operator = "GreaterThanThreshold" # More than one object.
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = "us-east-1"

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  if alarm_created_or_updated?(
    cloudwatch_client,
    alarm_name,
```

```
    alarm_description,
    metric_name,
    alarm_actions,
    namespace,
    statistic,
    dimensions,
    period,
    unit,
    evaluation_periods,
    threshold,
    comparison_operator
  )
    puts "Alarm '#{alarm_name}' created or updated."
  else
    puts "Could not create or update alarm '#{alarm_name}'."
  end

  if alarm_actions_disabled?(cloudwatch_client, alarm_name)
    puts "Alarm '#{alarm_name}' disabled."
  else
    puts "Could not disable alarm '#{alarm_name}'."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考DisableAlarmActions中的。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    "Disables actions on the specified alarm. "
    TRY.
```

```
        lo_cwt->disablealarmactions(
          it_alarmnames = it_alarm_names
        ).
        MESSAGE 'Alarm actions disabled.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- 如需 API 詳細資訊，請參閱 AWS SDK DisableAlarmActions中的 SAP ABAP API 參考資料。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭EnableAlarmActions配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用EnableAlarmActions。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 管理指標和警示

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Enable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
```

```
    public async Task<bool> EnableAlarmActions(List<string> alarmNames)
    {
        var enableAlarmActionsResult = await
_amazonCloudWatch.EnableAlarmActionsAsync(
            new EnableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return enableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考EnableAlarmActions中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

包括必需的檔案。

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/EnableAlarmActionsRequest.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

啟用警示動作。

```
    Aws::CloudWatch::CloudWatchClient cw;
    Aws::CloudWatch::Model::PutMetricAlarmRequest request;
    request.SetAlarmName(alarm_name);
    request.SetComparisonOperator(
        Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
```

```
    request.SetEvaluationPeriods(1);
    request.SetMetricName("CPUUtilization");
    request.SetNamespace("AWS/EC2");
    request.SetPeriod(60);
    request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
    request.SetThreshold(70.0);
    request.SetActionsEnabled(false);
    request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
    request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);
    request.AddAlarmActions(actionArn);

    Aws::CloudWatch::Model::Dimension dimension;
    dimension.SetName("InstanceId");
    dimension.SetValue(instanceId);
    request.AddDimensions(dimension);

    auto outcome = cw.PutMetricAlarm(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to create CloudWatch alarm:" <<
            outcome.GetError().GetMessage() << std::endl;
        return;
    }

    Aws::CloudWatch::Model::EnableAlarmActionsRequest enable_request;
    enable_request.AddAlarmNames(alarm_name);

    auto enable_outcome = cw.EnableAlarmActions(enable_request);
    if (!enable_outcome.IsSuccess())
    {
        std::cout << "Failed to enable alarm actions:" <<
            enable_outcome.GetError().GetMessage() << std::endl;
        return;
    }

    std::cout << "Successfully created alarm " << alarm_name <<
        " and enabled actions on it." << std::endl;
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考EnableAlarmActions中的。

CLI

AWS CLI

啟用警示的所有動作

下列範例使用 enable-alarm-actions 命令來啟用名為 myalarm 之警示的所有動作：

```
aws cloudwatch enable-alarm-actions --alarm-names myalarm
```

如果成功，此命令會回到提示字元。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考EnableAlarmActions中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import
 software.amazon.awssdk.services.cloudwatch.model.EnableAlarmActionsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class EnableAlarmActions {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:
                  <alarmName>

                Where:
                  alarmName - An alarm name to enable (for example, MyAlarm).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alarm = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        enableActions(cw, alarm);
        cw.close();
    }

    public static void enableActions(CloudWatchClient cw, String alarm) {
        try {
            EnableAlarmActionsRequest request =
 EnableAlarmActionsRequest.builder()
                    .alarmNames(alarm)
                    .build();

            cw.enableAlarmActions(request);
            System.out.printf("Successfully enabled actions on alarm %s", alarm);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考EnableAlarmActions中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

匯入 SDK 和用戶端模組，然後呼叫 API。

```
import { EnableAlarmActionsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  const command = new EnableAlarmActionsCommand({
    AlarmNames: [process.env.CLOUDWATCH_ALARM_NAME], // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

在單獨的模組中建立用戶端並將其匯出。

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考EnableAlarmActions中的。

適用於 JavaScript (v2) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

匯入 SDK 和用戶端模組，然後呼叫 API。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmName: "Web_Server_CPU_Utilization",
  ComparisonOperator: "GreaterThanThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: true,
  AlarmActions: ["ACTION_ARN"],
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

cw.putMetricAlarm(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
```

```
    console.log("Alarm action added", data);
    var paramsEnableAlarmAction = {
      AlarmNames: [params.AlarmName],
    };
    cw.enableAlarmActions(paramsEnableAlarmAction, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Alarm action enabled", data);
      }
    });
  }
});
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考EnableAlarmActions中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun enableActions(alarm: String) {

    val request = EnableAlarmActionsRequest {
        alarmNames = listOf(alarm)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.enableAlarmActions(request)
        println("Successfully enabled actions on alarm $alarm")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK EnableAlarmActions中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def enable_alarm_actions(self, alarm_name, enable):
        """
        Enables or disables actions on the specified alarm. Alarm actions can be
        used to send notifications or automate responses when an alarm enters a
        particular state.

        :param alarm_name: The name of the alarm.
        :param enable: When True, actions are enabled for the alarm. Otherwise,
 they
                       disabled.
        """
        try:
            alarm = self.cloudwatch_resource.Alarm(alarm_name)
            if enable:
                alarm.enable_actions()
            else:
                alarm.disable_actions()
            logger.info(
                "%s actions for alarm %s.",
                "Enabled" if enable else "Disabled",
```

```
                alarm_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't %s actions alarm %s.",
                "enable" if enable else "disable",
                alarm_name,
            )
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件EnableAlarmActions中的 Python (博托 3) API 參考。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    "Enable actions on the specified alarm."
    TRY.
        lo_cwt->enablealarmactions(
          it_alarmnames = it_alarm_names
        ).
        MESSAGE 'Alarm actions enabled.' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
        DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
        MESSAGE lv_error TYPE 'E'.
    ENDTRY.
```

- 如需 API 詳細資訊，請參閱 AWS SDK EnableAlarmActions中的 SAP ABAP API 參考資料。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetDashboard配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetDashboard。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/// <summary>
/// Get information on a dashboard.
/// </summary>
/// <param name="dashboardName">The name of the dashboard.</param>
/// <returns>A JSON object with dashboard information.</returns>
public async Task<string> GetDashboard(string dashboardName)
{
    var dashboardResponse = await _amazonCloudWatch.GetDashboardAsync(
        new GetDashboardRequest()
        {
            DashboardName = dashboardName
        });

    return dashboardResponse.DashboardBody;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考GetDashboard中的。

PowerShell

適用的工具 PowerShell

範例 1：傳回 arn 指定儀表板主體。

```
Get-CWDashboard -DashboardName Dashboard1
```

輸出：

```
DashboardArn                                                DashboardBody
------------                                                -------------
arn:aws:cloudwatch::123456789012:dashboard/Dashboard1 {...
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetDashboard式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetMetricData配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetMetricData。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用指標、儀表板和警示

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Get data for CloudWatch metrics.
```

```
    /// </summary>
    /// <param name="minutesOfData">The number of minutes of data to include.</
param>
    /// <param name="useDescendingTime">True to return the data descending by
 time.</param>
    /// <param name="endDateUtc">The end date for the data, in UTC.</param>
    /// <param name="maxDataPoints">The maximum data points to include.</param>
    /// <param name="dataQueries">Optional data queries to include.</param>
    /// <returns>A list of the requested metric data.</returns>
    public async Task<List<MetricDataResult>> GetMetricData(int minutesOfData,
 bool useDescendingTime, DateTime? endDateUtc = null,
        int maxDataPoints = 0, List<MetricDataQuery>? dataQueries = null)
    {
        var metricData = new List<MetricDataResult>();
        // If no end time is provided, use the current time for the end time.
        endDateUtc ??= DateTime.UtcNow;
        var timeZoneOffset =
 TimeZoneInfo.Local.GetUtcOffset(endDateUtc.Value.ToLocalTime());
        var startTimeUtc = endDateUtc.Value.AddMinutes(-minutesOfData);
        // The timezone string should be in the format +0000, so use the timezone
 offset to format it correctly.
        var timeZoneString = $"{timeZoneOffset.Hours:D2}
{timeZoneOffset.Minutes:D2}";
        var paginatedMetricData = _amazonCloudWatch.Paginators.GetMetricData(
            new GetMetricDataRequest()
            {
                StartTimeUtc = startTimeUtc,
                EndTimeUtc = endDateUtc.Value,
                LabelOptions = new LabelOptions { Timezone = timeZoneString },
                ScanBy = useDescendingTime ? ScanBy.TimestampDescending :
 ScanBy.TimestampAscending,
                MaxDatapoints = maxDataPoints,
                MetricDataQueries = dataQueries,
            });

        await foreach (var data in paginatedMetricData.MetricDataResults)
        {
            metricData.Add(data);
        }
        return metricData;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考GetMetricData中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   public static void getCustomMetricData(CloudWatchClient cw, String fileName)
{
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
           String customMetricName =
rootNode.findValue("customMetricName").asText();

           // Set the date.
           Instant nowDate = Instant.now();

           long hours = 1;
           long minutes = 30;
           Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
                   ChronoUnit.MINUTES);

           Metric met = Metric.builder()
                   .metricName(customMetricName)
                   .namespace(customMetricNamespace)
                   .build();

           MetricStat metStat = MetricStat.builder()
                   .stat("Maximum")
                   .period(1)
                   .metric(met)
                   .build();
```

```
            MetricDataQuery dataQUery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();

            List<MetricDataQuery> dq = new ArrayList<>();
            dq.add(dataQUery);

            GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(date2)
                    .metricDataQueries(dq)
                    .build();

            GetMetricDataResponse response = cw.getMetricData(getMetReq);
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                System.out.println("The label is " + item.label());
                System.out.println("The status code is " +
item.statusCode().toString());
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[GetMetricData](#)中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(
        minutes,
        ChronoUnit.MINUTES
    )

    val met = Metric {
        metricName = customMetricName
        namespace = customMetricNamespace
    }

    val metStat = MetricStat {
        stat = "Maximum"
        period = 1
        metric = met
    }

    val dataQUery = MetricDataQuery {
        metricStat = metStat
        id = "foo2"
        returnData = true
    }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq = GetMetricDataRequest {
        maxDatapoints = 10
        scanBy = ScanBy.TimestampDescending
        startTime = aws.smithy.kotlin.runtime.time.Instant(nowDate)
        endTime = aws.smithy.kotlin.runtime.time.Instant(date2)
        metricDataQueries = dq
    }
```

```
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK GetMetricData中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetMetricStatistics配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetMetricStatistics。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用指標、儀表板和警示
- 管理指標和警示

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Get billing statistics using a call to a wrapper class.
    /// </summary>
    /// <returns>A collection of billing statistics.</returns>
```

```
    private static async Task<List<Datapoint>> SetupBillingStatistics()
    {
        // Make a request for EstimatedCharges with a period of one day for the
past seven days.
        var billingStatistics = await _cloudWatchWrapper.GetMetricStatistics(
            "AWS/Billing",
            "EstimatedCharges",
            new List<string>() { "Maximum" },
            new List<Dimension>() { new Dimension { Name = "Currency", Value =
"USD" } },
            7,
            86400);

        billingStatistics = billingStatistics.OrderBy(n => n.Timestamp).ToList();

        return billingStatistics;
    }

    /// <summary>
    /// Wrapper to get statistics for a specific CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <param name="statistics">The list of statistics to include.</param>
    /// <param name="dimensions">The list of dimensions to include.</param>
    /// <param name="days">The number of days in the past to include.</param>
    /// <param name="period">The period for the data.</param>
    /// <returns>A list of DataPoint objects for the statistics.</returns>
    public async Task<List<Datapoint>> GetMetricStatistics(string
metricNamespace,
        string metricName, List<string> statistics, List<Dimension> dimensions,
int days, int period)
    {
        var metricStatistics = await _amazonCloudWatch.GetMetricStatisticsAsync(
            new GetMetricStatisticsRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName,
                Dimensions = dimensions,
                Statistics = statistics,
                StartTimeUtc = DateTime.UtcNow.AddDays(-days),
                EndTimeUtc = DateTime.UtcNow,
                Period = period
            });
```

```
        return metricStatistics.Datapoints;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考GetMetricStatistics中的。

CLI

AWS CLI

取得每個 EC2 執行個體的 CPU 使用率

下列範例使用 `get-metric-statistics` 命令來取得 ID 為 i-abcdef 之 EC2 執行個體的 CPU 使用率。

```
aws cloudwatch get-metric-statistics --metric-name CPUUtilization --start-time
 2014-04-08T23:18:00Z --end-time 2014-04-09T23:18:00Z --period 3600 --namespace
 AWS/EC2 --statistics Maximum --dimensions Name=InstanceId,Value=i-abcdef
```

輸出：

```
{
    "Datapoints": [
        {
            "Timestamp": "2014-04-09T11:18:00Z",
            "Maximum": 44.79,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T20:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T19:18:00Z",
            "Maximum": 50.85,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T09:18:00Z",
            "Maximum": 47.92,
```

```
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T03:18:00Z",
            "Maximum": 76.84,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T21:18:00Z",
            "Maximum": 48.96,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T14:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T08:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T16:18:00Z",
            "Maximum": 45.55,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T06:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T13:18:00Z",
            "Maximum": 45.08,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T05:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T18:18:00Z",
```

```
            "Maximum": 46.88,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T17:18:00Z",
            "Maximum": 52.08,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T07:18:00Z",
            "Maximum": 47.92,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T02:18:00Z",
            "Maximum": 51.23,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T12:18:00Z",
            "Maximum": 47.67,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-08T23:18:00Z",
            "Maximum": 46.88,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T10:18:00Z",
            "Maximum": 51.91,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T04:18:00Z",
            "Maximum": 47.13,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T15:18:00Z",
            "Maximum": 48.96,
            "Unit": "Percent"
        },
        {
```

```
            "Timestamp": "2014-04-09T00:18:00Z",
            "Maximum": 48.16,
            "Unit": "Percent"
        },
        {
            "Timestamp": "2014-04-09T01:18:00Z",
            "Maximum": 49.18,
            "Unit": "Percent"
        }
    ],
    "Label": "CPUUtilization"
}
```

指定多個維度

下列範例說明如何指定多個維度。每個維度都指定為「名稱/值」對，名稱和值之間用逗號分隔。使用空格分隔多個維度。如果單一指標包含多個維度，則必須為每個定義的維度指定值。

如需使用`get-metric-statistics`命令的更多範例，請參閱 Amazon CloudWatch 開發人員指南中的取得指標的統計資料。

```
aws cloudwatch get-metric-statistics --metric-name Buffers --
namespace MyNameSpace --dimensions Name=InstanceID,Value=i-abcdef
 Name=InstanceType,Value=m1.small --start-time 2016-10-15T04:00:00Z --end-time
 2016-10-19T07:00:00Z --statistics Average --period 60
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[GetMetricStatistics](#)中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
    public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
 nameSpace, String metVal,
```

```
            String metricOption, String date, Dimension myDimension) {
        try {
            Instant start = Instant.parse(date);
            Instant endDate = Instant.now();

            GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
                    .endTime(endDate)
                    .startTime(start)
                    .dimensions(myDimension)
                    .metricName(metVal)
                    .namespace(nameSpace)
                    .period(86400)
                    .statistics(Statistic.fromValue(metricOption))
                    .build();

            GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
            List<Datapoint> data = response.datapoints();
            if (!data.isEmpty()) {
                for (Datapoint datapoint : data) {
                    System.out
                            .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
                }
            } else {
                System.out.println("The returned data list is empty");
            }

        } catch (CloudWatchException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考GetMetricStatistics中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun getAndDisplayMetricStatistics(nameSpaceVal: String, metVal: String,
 metricOption: String, date: String, myDimension: Dimension) {
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest = GetMetricStatisticsRequest {
        endTime = aws.smithy.kotlin.runtime.time.Instant(endDate)
        startTime = aws.smithy.kotlin.runtime.time.Instant(start)
        dimensions = listOf(myDimension)
        metricName = metVal
        namespace = nameSpaceVal
        period = 86400
        statistics = listOf(Statistic.fromValue(metricOption))
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK GetMetricStatistics中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def get_metric_statistics(self, namespace, name, start, end, period,
 stat_types):
        """
        Gets statistics for a metric within a specified time span. Metrics are
 grouped
        into the specified period.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param start: The UTC start time of the time span to retrieve.
        :param end: The UTC end time of the time span to retrieve.
        :param period: The period, in seconds, in which to group metrics. The
 period
                       must match the granularity of the metric, which depends on
                       the metric's age. For example, metrics that are older than
                       three hours have a one-minute granularity, so the period
 must
                       be at least 60 and must be a multiple of 60.
        :param stat_types: The type of statistics to retrieve, such as average
 value
                           or maximum value.
        :return: The retrieved statistics for the metric.
```

```
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            stats = metric.get_statistics(
                StartTime=start, EndTime=end, Period=period,
Statistics=stat_types
            )
            logger.info(
                "Got %s statistics for %s.", len(stats["Datapoints"]),
stats["Label"]
            )
        except ClientError:
            logger.exception("Couldn't get statistics for %s.%s.", namespace,
name)
            raise
        else:
            return stats
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件GetMetricStatistics中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetMetricWidgetImage配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetMetricWidgetImage。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用指標、儀表板和警示

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Get an image for a metric graphed over time.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metric">The name of the metric.</param>
    /// <param name="stat">The name of the stat to chart.</param>
    /// <param name="period">The period to use for the chart.</param>
    /// <returns>A memory stream for the chart image.</returns>
    public async Task<MemoryStream> GetTimeSeriesMetricImage(string
metricNamespace, string metric, string stat, int period)
    {
        var metricImageWidget = new
        {
            title = "Example Metric Graph",
            view = "timeSeries",
            stacked = false,
            period = period,
            width = 1400,
            height = 600,
            metrics = new List<List<object>>
                { new() { metricNamespace, metric, new { stat } } }
        };

        var metricImageWidgetString =
JsonSerializer.Serialize(metricImageWidget);
        var imageResponse = await _amazonCloudWatch.GetMetricWidgetImageAsync(
            new GetMetricWidgetImageRequest()
            {
                MetricWidget = metricImageWidgetString
            });

        return imageResponse.MetricWidgetImage;
```

```
    }

    /// <summary>
    /// Save a metric image to a file.
    /// </summary>
    /// <param name="memoryStream">The MemoryStream for the metric image.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The path to the file.</returns>
    public string SaveMetricImage(MemoryStream memoryStream, string metricName)
    {
        var metricFileName = $"{metricName}_{DateTime.Now.Ticks}.png";
        using var sr = new StreamReader(memoryStream);
        // Writes the memory stream to a file.
        File.WriteAllBytes(metricFileName, memoryStream.ToArray());
        var filePath = Path.Join(AppDomain.CurrentDomain.BaseDirectory,
            metricFileName);
        return filePath;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考[GetMetricWidgetImage](中的。)

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫]。

```
   public static void getAndOpenMetricImage(CloudWatchClient cw, String
fileName) {
       System.out.println("Getting Image data for custom metric.");
       try {
           String myJSON = "{\n" +
                   "  \"title\": \"Example Metric Graph\",\n" +
                   "  \"view\": \"timeSeries\",\n" +
                   "  \"stacked \": false,\n" +
                   "  \"period\": 10,\n" +
                   "  \"width\": 1400,\n" +
```

```
                    "  \"height\": 600,\n" +
                    "  \"metrics\": [\n" +
                    "    [\n" +
                    "      \"AWS/Billing\",\n" +
                    "      \"EstimatedCharges\",\n" +
                    "      \"Currency\",\n" +
                    "      \"USD\"\n" +
                    "    ]\n" +
                    "  ]\n" +
                    "}";

            GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
                    .metricWidget(myJSON)
                    .build();

            GetMetricWidgetImageResponse response =
cw.getMetricWidgetImage(imageRequest);
            SdkBytes sdkBytes = response.metricWidgetImage();
            byte[] bytes = sdkBytes.asByteArray();
            File outputFile = new File(fileName);
            try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
                outputStream.write(bytes);
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[GetMetricWidgetImage](#)中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
            ]
        ]
        }"""

    val imageRequest = GetMetricWidgetImageRequest {
        metricWidget = myJSON
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK GetMetricWidgetImage中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ListDashboards配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListDashboards。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/// <summary>
/// Get a list of dashboards.
/// </summary>
/// <returns>A list of DashboardEntry objects.</returns>
public async Task<List<DashboardEntry>> ListDashboards()
{
    var results = new List<DashboardEntry>();
    var paginateDashboards = _amazonCloudWatch.Paginators.ListDashboards(
        new ListDashboardsRequest());
    // Get the entire list using the paginator.
    await foreach (var data in paginateDashboards.DashboardEntries)
    {
        results.Add(data);
    }

    return results;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListDashboards中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    public static void listDashboards(CloudWatchClient cw) {
        try {
            ListDashboardsIterable listRes = cw.listDashboardsPaginator();
            listRes.stream()
                    .flatMap(r -> r.dashboardEntries().stream())
                    .forEach(entry -> {
                        System.out.println("Dashboard name is: " +
entry.dashboardName());
                        System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
                    });

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考ListDashboards中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK ListDashboards中的 Kotlin API 參考。

PowerShell

適用的工具 PowerShell

範例 1：傳回您帳戶的儀表板集合。

```
Get-CWDashboardList
```

輸出：

```
DashboardArn DashboardName LastModified        Size
------------ ------------- ------------        ----
arn:...      Dashboard1    7/6/2017 8:14:15 PM 252
```

範例 2：傳回名稱以字首「dev」開頭的帳戶的儀表板集合。

```
Get-CWDashboardList -DashboardNamePrefix dev
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程ListDashboards式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListMetrics**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListMetrics。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用指標、儀表板和警示
- 管理指標和警示

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// List metrics available, optionally within a namespace.
    /// </summary>
    /// <param name="metricNamespace">Optional CloudWatch namespace to use when
listing metrics.</param>
    /// <param name="filter">Optional dimension filter.</param>
    /// <param name="metricName">Optional metric name filter.</param>
    /// <returns>The list of metrics.</returns>
    public async Task<List<Metric>> ListMetrics(string? metricNamespace = null,
DimensionFilter? filter = null, string? metricName = null)
    {
        var results = new List<Metric>();
        var paginateMetrics = _amazonCloudWatch.Paginators.ListMetrics(
            new ListMetricsRequest
            {
                Namespace = metricNamespace,
                Dimensions = filter != null ? new List<DimensionFilter>
{ filter } : null,
                MetricName = metricName
            });
        // Get the entire list using the paginator.
        await foreach (var metric in paginateMetrics.Metrics)
        {
            results.Add(metric);
        }
```

```
        return results;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListMetrics中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

包括必需的檔案。

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/ListMetricsRequest.h>
#include <aws/monitoring/model/ListMetricsResult.h>
#include <iomanip>
#include <iostream>
```

列出指標。

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::ListMetricsRequest request;

        if (argc > 1)
        {
            request.SetMetricName(argv[1]);
        }

        if (argc > 2)
        {
            request.SetNamespace(argv[2]);
        }
```

```
        bool done = false;
        bool header = false;
        while (!done)
        {
            auto outcome = cw.ListMetrics(request);
            if (!outcome.IsSuccess())
            {
                std::cout << "Failed to list CloudWatch metrics:" <<
                    outcome.GetError().GetMessage() << std::endl;
                break;
            }

            if (!header)
            {
                std::cout << std::left << std::setw(48) << "MetricName" <<
                    std::setw(32) << "Namespace" << "DimensionNameValuePairs" <<
                    std::endl;
                header = true;
            }

            const auto &metrics = outcome.GetResult().GetMetrics();
            for (const auto &metric : metrics)
            {
                std::cout << std::left << std::setw(48) <<
                    metric.GetMetricName() << std::setw(32) <<
                    metric.GetNamespace();
                const auto &dimensions = metric.GetDimensions();
                for (auto iter = dimensions.cbegin();
                    iter != dimensions.cend(); ++iter)
                {
                    const auto &dimkv = *iter;
                    std::cout << dimkv.GetName() << " = " << dimkv.GetValue();
                    if (iter + 1 != dimensions.cend())
                    {
                        std::cout << ", ";
                    }
                }
                std::cout << std::endl;
            }

            const auto &next_token = outcome.GetResult().GetNextToken();
            request.SetNextToken(next_token);
            done = next_token.empty();
```

```
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考ListMetrics中的。

CLI

AWS CLI

列出 Amazon SNS 的指標

下列 `list-metrics` 範例顯示 Amazon SNS 的指標。

```
aws cloudwatch list-metrics \
    --namespace "AWS/SNS"
```

輸出：

```
{
    "Metrics": [
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
            "MetricName": "PublishSize"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
            ],
            "MetricName": "PublishSize"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
```

```
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
            "MetricName": "NumberOfNotificationsFailed"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
            "MetricName": "NumberOfNotificationsDelivered"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "NotifyMe"
                }
            ],
            "MetricName": "NumberOfMessagesPublished"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
            ],
            "MetricName": "NumberOfMessagesPublished"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
```

```
            ],
            "MetricName": "NumberOfNotificationsDelivered"
        },
        {
            "Namespace": "AWS/SNS",
            "Dimensions": [
                {
                    "Name": "TopicName",
                    "Value": "CFO"
                }
            ],
            "MetricName": "NumberOfNotificationsFailed"
        }
    ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListMetrics中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListMetrics {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <namespace>\s

                Where:
                  namespace - The namespace to filter against (for example, AWS/
EC2).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String namespace = args[0];
        Region region = Region.US_EAST_1;
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .build();

        listMets(cw, namespace);
        cw.close();
    }

    public static void listMets(CloudWatchClient cw, String namespace) {
        boolean done = false;
        String nextToken = null;

        try {
            while (!done) {

                ListMetricsResponse response;
                if (nextToken == null) {
                    ListMetricsRequest request = ListMetricsRequest.builder()
                            .namespace(namespace)
                            .build();

                    response = cw.listMetrics(request);
```

```
                } else {
                    ListMetricsRequest request = ListMetricsRequest.builder()
                            .namespace(namespace)
                            .nextToken(nextToken)
                            .build();

                    response = cw.listMetrics(request);
                }

                for (Metric metric : response.metrics()) {
                    System.out.printf("Retrieved metric %s",
 metric.metricName());
                    System.out.println();
                }

                if (response.nextToken() == null) {
                    done = true;
                } else {
                    nextToken = response.nextToken();
                }
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[ListMetrics](#)中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#)中設定和執行。

匯入 SDK 和用戶端模組，然後呼叫 API。

```
import { ListMetricsCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

export const main = () => {
  // Use the AWS console to see available namespaces and metric names. Custom
 metrics can also be created.
  // https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
viewing_metrics_with_cloudwatch.html
  const command = new ListMetricsCommand({
    Dimensions: [
      {
        Name: "LogGroupName",
      },
    ],
    MetricName: "IncomingLogEvents",
    Namespace: "AWS/Logs",
  });

  return client.send(command);
};
```

在單獨的模組中建立用戶端並將其匯出。

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考ListMetrics中的。

適用於 JavaScript (v2) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  Dimensions: [
    {
      Name: "LogGroupName" /* required */,
    },
  ],
  MetricName: "IncomingLogEvents",
  Namespace: "AWS/Logs",
};

cw.listMetrics(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Metrics", JSON.stringify(data.Metrics));
  }
});
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考ListMetrics中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
```

```
    val metList = ArrayList<String>()
    val request = ListMetricsRequest {
        namespace = namespaceVal
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
    return metList
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK ListMetrics中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def list_metrics(self, namespace, name, recent=False):
        """
        Gets the metrics within a namespace that have the specified name.
```

```
        If the metric has no dimensions, a single metric is returned.
        Otherwise, metrics for all dimensions are returned.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param recent: When True, only metrics that have been active in the last
                       three hours are returned.
        :return: An iterator that yields the retrieved metrics.
        """
        try:
            kwargs = {"Namespace": namespace, "MetricName": name}
            if recent:
                kwargs["RecentlyActive"] = "PT3H"  # List past 3 hours only
            metric_iter = self.cloudwatch_resource.metrics.filter(**kwargs)
            logger.info("Got metrics for %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't get metrics for %s.%s.", namespace, name)
            raise
        else:
            return metric_iter
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件ListMetrics中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
# Lists available metrics for a metric namespace in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric.
# @example
#   list_metrics_for_namespace(
```

```
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC'
#   )
def list_metrics_for_namespace(cloudwatch_client, metric_namespace)
  response = cloudwatch_client.list_metrics(namespace: metric_namespace)

  if response.metrics.count.positive?
    response.metrics.each do |metric|
      puts "  Metric name: #{metric.metric_name}"
      if metric.dimensions.count.positive?
        puts "    Dimensions:"
        metric.dimensions.each do |dimension|
          puts "      Name: #{dimension.name}, Value: #{dimension.value}"
        end
      else
        puts "No dimensions found."
      end
    end
  else
    puts "No metrics found for namespace '#{metric_namespace}'. " \
      "Note that it could take up to 15 minutes for recently-added metrics " \
      "to become available."
  end
end

# Example usage:
def run_me
  metric_namespace = "SITE/TRAFFIC"
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = "us-east-1"

  cloudwatch_client = Aws::CloudWatch::Client.new(region: region)

  # Add three datapoints.
  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    "UniqueVisitors",
    "SiteName",
    "example.com",
    5_885.0,
    "Count"
  )
```

```
  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    "UniqueVisits",
    "SiteName",
    "example.com",
    8_628.0,
    "Count"
  )

  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    "PageViews",
    "PageURL",
    "example.html",
    18_057.0,
    "Count"
  )

  puts "Metrics for namespace '#{metric_namespace}':"
  list_metrics_for_namespace(cloudwatch_client, metric_namespace)
end

run_me if $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考ListMetrics中的。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    "The following list-metrics example displays the metrics for Amazon
 CloudWatch."
    TRY.
```

```
        oo_result = lo_cwt->listmetrics(            " oo_result is returned for
testing purposes. "
          iv_namespace = iv_namespace
        ).
        DATA(lt_metrics) = oo_result->get_metrics( ).
        MESSAGE 'Metrics retrieved.' TYPE 'I'.
      CATCH /aws1/cx_cwtinvparamvalueex .
        MESSAGE 'The specified argument was not valid.' TYPE 'E'.
    ENDTRY.
```

- 如需 API 詳細資訊，請參閱 AWS SDK ListMetrics中的 SAP ABAP API 參考資料。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutAnomalyDetector配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutAnomalyDetector。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用指標、儀表板和警示

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Add an anomaly detector for a single metric.
    /// </summary>
    /// <param name="anomalyDetector">A single metric anomaly detector.</param>
    /// <returns>True if successful.</returns>
```

```
    public async Task<bool> PutAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var putAlarmDetectorResult = await
_amazonCloudWatch.PutAnomalyDetectorAsync(
            new PutAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });

        return putAlarmDetectorResult.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考PutAnomalyDetector中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
```

```
                    .stat("Maximum")
                    .build();

            PutAnomalyDetectorRequest anomalyDetectorRequest =
 PutAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

            cw.putAnomalyDetector(anomalyDetectorRequest);
            System.out.println("Added anomaly detector for metric " +
 customMetricName + ".");

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[PutAnomalyDetector](#)中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal = SingleMetricAnomalyDetector {
        metricName = customMetricName
        namespace = customMetricNamespace
```

```
            stat = "Maximum"
        }

        val anomalyDetectorRequest = PutAnomalyDetectorRequest {
            singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
        }

        CloudWatchClient { region = "us-east-1" }.use { cwClient ->
            cwClient.putAnomalyDetector(anomalyDetectorRequest)
            println("Added anomaly detector for metric $customMetricName.")
        }
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK PutAnomalyDetector中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutDashboard配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutDashboard。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用指標、儀表板和警示

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
```

```
    /// Set up a dashboard using a call to the wrapper class.
    /// </summary>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A list of validation messages.</returns>
    private static async Task<List<DashboardValidationMessage>> SetupDashboard(
        string customMetricNamespace, string customMetricName, string
dashboardName)
    {
        // Get the dashboard model from configuration.
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);

        // Add a new metric to the dashboard.
        newDashboard.Widgets.Add(new Widget
        {
            Height = 8,
            Width = 8,
            Y = 8,
            X = 0,
            Type = "metric",
            Properties = new Properties
            {
                Metrics = new List<List<object>>
                    { new() { customMetricNamespace, customMetricName } },
                View = "timeSeries",
                Region = "us-east-1",
                Stat = "Sum",
                Period = 86400,
                YAxis = new YAxis { Left = new Left { Min = 0, Max = 100 } },
                Title = "Custom Metric Widget",
                LiveData = true,
                Sparkline = true,
                Trend = true,
                Stacked = false,
                SetPeriodToTimeRange = false
            }
        });

        var newDashboardString = JsonSerializer.Serialize(newDashboard,
            new JsonSerializerOptions
            { DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull });
        var validationMessages =
```

```
            await _cloudWatchWrapper.PutDashboard(dashboardName,
 newDashboardString);

        return validationMessages;
    }

    /// <summary>
    /// Wrapper to create or add to a dashboard with metrics.
    /// </summary>
    /// <param name="dashboardName">The name for the dashboard.</param>
    /// <param name="dashboardBody">The metric data in JSON for the dashboard.</
param>
    /// <returns>A list of validation messages for the dashboard.</returns>
    public async Task<List<DashboardValidationMessage>> PutDashboard(string
 dashboardName,
        string dashboardBody)
    {
        // Updating a dashboard replaces all contents.
        // Best practice is to include a text widget indicating this dashboard
 was created programmatically.
        var dashboardResponse = await _amazonCloudWatch.PutDashboardAsync(
            new PutDashboardRequest()
            {
                DashboardName = dashboardName,
                DashboardBody = dashboardBody
            });

        return dashboardResponse.DashboardValidationMessages;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考PutDashboard中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
       try {
           PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
                   .dashboardName(dashboardName)
                   .dashboardBody(readFileAsString(fileName))
                   .build();

           PutDashboardResponse response = cw.putDashboard(dashboardRequest);
           System.out.println(dashboardName + " was successfully created.");
           List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
           if (messages.isEmpty()) {
               System.out.println("There are no messages in the new Dashboard");
           } else {
               for (DashboardValidationMessage message : messages) {
                   System.out.println("Message is: " + message.message());
               }
           }

       } catch (CloudWatchException | IOException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考PutDashboard中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun createDashboardWithMetrics(dashboardNameVal: String, fileNameVal:
 String) {
```

```
    val dashboardRequest = PutDashboardRequest {
        dashboardName = dashboardNameVal
        dashboardBody = readFileAsString(fileNameVal)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK PutDashboard中的 Kotlin API 參考。

PowerShell

適用的工具 PowerShell

範例 1：建立或更新名為「儀表板 1」的儀表板，以並排包含兩個公制小器具。

```
$dashBody = @"
{
    "widgets":[
        {
             "type":"metric",
             "x":0,
             "y":0,
             "width":12,
             "height":6,
             "properties":{
                "metrics":[
                   [
                      "AWS/EC2",
```

```
                    "CPUUtilization",
                    "InstanceId",
                    "i-012345"
                 ]
              ],
              "period":300,
              "stat":"Average",
              "region":"us-east-1",
              "title":"EC2 Instance CPU"
           }
        },
        {
           "type":"metric",
           "x":12,
           "y":0,
           "width":12,
           "height":6,
           "properties":{
              "metrics":[
                 [
                    "AWS/S3",
                    "BucketSizeBytes",
                    "BucketName",
                    "MyBucketName"
                 ]
              ],
              "period":86400,
              "stat":"Maximum",
              "region":"us-east-1",
              "title":"MyBucketName bytes"
           }
        }
    ]
}
"@

Write-CWDashboard -DashboardName Dashboard1 -DashboardBody $dashBody
```

範例 2：建立或更新儀表板，並將描述儀表板的內容傳送至指令程式。

```
$dashBody = @"
{
...
```

```
}
"@

$dashBody | Write-CWDashboard -DashboardName Dashboard1
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程PutDashboard式參考中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutMetricAlarm配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutMetricAlarm。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用警示
- 開始使用指標、儀表板和警示
- 管理指標和警示

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/// <summary>
/// Add a metric alarm to send an email when the metric passes a threshold.
/// </summary>
/// <param name="alarmDescription">A description of the alarm.</param>
/// <param name="alarmName">The name for the alarm.</param>
/// <param name="comparison">The type of comparison to use.</param>
/// <param name="metricName">The name of the metric for the alarm.</param>
/// <param name="metricNamespace">The namespace of the metric.</param>
/// <param name="threshold">The threshold value for the alarm.</param>
```

```
   /// <param name="alarmActions">Optional actions to execute when in an alarm
state.</param>
   /// <returns>True if successful.</returns>
   public async Task<bool> PutMetricEmailAlarm(string alarmDescription, string
alarmName, ComparisonOperator comparison,
       string metricName, string metricNamespace, double threshold, List<string>
alarmActions = null!)
   {
       try
       {
           var putEmailAlarmResponse = await
_amazonCloudWatch.PutMetricAlarmAsync(
               new PutMetricAlarmRequest()
               {
                   AlarmActions = alarmActions,
                   AlarmDescription = alarmDescription,
                   AlarmName = alarmName,
                   ComparisonOperator = comparison,
                   Threshold = threshold,
                   Namespace = metricNamespace,
                   MetricName = metricName,
                   EvaluationPeriods = 1,
                   Period = 10,
                   Statistic = new Statistic("Maximum"),
                   DatapointsToAlarm = 1,
                   TreatMissingData = "ignore"
               });
           return putEmailAlarmResponse.HttpStatusCode == HttpStatusCode.OK;
       }
       catch (LimitExceededException lex)
       {
           _logger.LogError(lex, $"Unable to add alarm {alarmName}. Alarm quota
has already been reached.");
       }

       return false;
   }

   /// <summary>
   /// Add specific email actions to a list of action strings for a CloudWatch
alarm.
   /// </summary>
   /// <param name="accountId">The AccountId for the alarm.</param>
   /// <param name="region">The region for the alarm.</param>
```

```
    /// <param name="emailTopicName">An Amazon Simple Notification Service (SNS)
 topic for the alarm email.</param>
    /// <param name="alarmActions">Optional list of existing alarm actions to
 append to.</param>
    /// <returns>A list of string actions for an alarm.</returns>
    public List<string> AddEmailAlarmAction(string accountId, string region,
        string emailTopicName, List<string>? alarmActions = null)
    {
        alarmActions ??= new List<string>();
        var snsAlarmAction = $"arn:aws:sns:{region}:{accountId}:
{emailTopicName}";
        alarmActions.Add(snsAlarmAction);
        return alarmActions;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考PutMetricAlarm中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

包括必需的檔案。

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

建立警示以監視指標。

```
        Aws::CloudWatch::CloudWatchClient cw;
        Aws::CloudWatch::Model::PutMetricAlarmRequest request;
        request.SetAlarmName(alarm_name);
```

```
        request.SetComparisonOperator(
            Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
        request.SetEvaluationPeriods(1);
        request.SetMetricName("CPUUtilization");
        request.SetNamespace("AWS/EC2");
        request.SetPeriod(60);
        request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
        request.SetThreshold(70.0);
        request.SetActionsEnabled(false);
        request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
        request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);

        Aws::CloudWatch::Model::Dimension dimension;
        dimension.SetName("InstanceId");
        dimension.SetValue(instanceId);

        request.AddDimensions(dimension);

        auto outcome = cw.PutMetricAlarm(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to create CloudWatch alarm:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully created CloudWatch alarm " << alarm_name
                << std::endl;
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考[PutMetricAlarm](#)中的。

CLI

AWS CLI

在 CPU 使用率超過 70% 時傳送 Amazon Simple Notification Service 電子郵件訊息

以下範例使用 `put-metric-alarm` 命令，在 CPU 使用率超過 70% 時傳送 Amazon Simple Notification Service 電子郵件訊息：

```
aws cloudwatch put-metric-alarm --alarm-name cpu-mon --alarm-description "Alarm
 when CPU exceeds 70 percent" --metric-name CPUUtilization --namespace AWS/
EC2 --statistic Average --period 300 --threshold 70 --comparison-operator
 GreaterThanThreshold  --dimensions "Name=InstanceId,Value=i-12345678" --
evaluation-periods 2 --alarm-actions arn:aws:sns:us-east-1:111122223333:MyTopic
 --unit Percent
```

如果成功，此命令會回到提示字元。如果已存在具有相同名稱的警示，將會被新警示所覆寫。

指定多個維度

下列範例說明如何指定多個維度。每個維度都指定為「名稱/值」對，名稱和值之間用逗號分隔。多個維度由空格分隔：

```
aws cloudwatch put-metric-alarm --alarm-name "Default_Test_Alarm3" --alarm-
description "The default example alarm" --namespace "CW EXAMPLE METRICS" --
metric-name Default_Test --statistic Average --period 60 --evaluation-periods 3
 --threshold 50 --comparison-operator GreaterThanOrEqualToThreshold --dimensions
 Name=key1,Value=value1 Name=key2,Value=value2
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutMetricAlarm中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    public static String createAlarm(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
```

```
            String customMetricName =
rootNode.findValue("customMetricName").asText();
            String alarmName = rootNode.findValue("exampleAlarmName").asText();
            String emailTopic = rootNode.findValue("emailTopic").asText();
            String accountId = rootNode.findValue("accountId").asText();
            String region = rootNode.findValue("region").asText();

            // Create a List for alarm actions.
            List<String> alarmActions = new ArrayList<>();
            alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
            PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
                    .alarmActions(alarmActions)
                    .alarmDescription("Example metric alarm")
                    .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
                    .threshold(100.00)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .evaluationPeriods(1)
                    .period(10)
                    .statistic("Maximum")
                    .datapointsToAlarm(1)
                    .treatMissingData("ignore")
                    .build();

            cw.putMetricAlarm(alarmRequest);
            System.out.println(alarmName + " was successfully created!");
            return alarmName;

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考PutMetricAlarm中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

匯入 SDK 和用戶端模組，然後呼叫 API。

```
import { PutMetricAlarmCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  // This alarm triggers when CPUUtilization exceeds 70% for one minute.
  const command = new PutMetricAlarmCommand({
    AlarmName: process.env.CLOUDWATCH_ALARM_NAME, // Set the value of
 CLOUDWATCH_ALARM_NAME to the name of an existing alarm.
    ComparisonOperator: "GreaterThanThreshold",
    EvaluationPeriods: 1,
    MetricName: "CPUUtilization",
    Namespace: "AWS/EC2",
    Period: 60,
    Statistic: "Average",
    Threshold: 70.0,
    ActionsEnabled: false,
    AlarmDescription: "Alarm when server CPU exceeds 70%",
    Dimensions: [
      {
        Name: "InstanceId",
        Value: process.env.EC2_INSTANCE_ID, // Set the value of EC_INSTANCE_ID to
 the Id of an existing Amazon EC2 instance.
      },
    ],
    Unit: "Percent",
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
```

```
  }
};

export default run();
```

在單獨的模組中建立用戶端並將其匯出。

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考PutMetricAlarm中的。

適用於 JavaScript (v2) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

var params = {
  AlarmName: "Web_Server_CPU_Utilization",
  ComparisonOperator: "GreaterThanThreshold",
  EvaluationPeriods: 1,
  MetricName: "CPUUtilization",
  Namespace: "AWS/EC2",
  Period: 60,
  Statistic: "Average",
  Threshold: 70.0,
  ActionsEnabled: false,
```

```
  AlarmDescription: "Alarm when server CPU exceeds 70%",
  Dimensions: [
    {
      Name: "InstanceId",
      Value: "INSTANCE_ID",
    },
  ],
  Unit: "Percent",
};

cw.putMetricAlarm(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考PutMetricAlarm中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun putMetricAlarm(alarmNameVal: String, instanceIdVal: String) {

    val dimensionOb = Dimension {
        name = "InstanceId"
        value = instanceIdVal
    }

    val request = PutMetricAlarmRequest {
        alarmName = alarmNameVal
```

```
        comparisonOperator = ComparisonOperator.GreaterThanThreshold
        evaluationPeriods = 1
        metricName = "CPUUtilization"
        namespace = "AWS/EC2"
        period = 60
        statistic = Statistic.fromValue("Average")
        threshold = 70.0
        actionsEnabled = false
        alarmDescription = "An Alarm created by the Kotlin SDK when server CPU
 utilization exceeds 70%"
        unit = StandardUnit.fromValue("Seconds")
        dimensions = listOf(dimensionOb)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(request)
        println("Successfully created an alarm with name $alarmNameVal")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK PutMetricAlarm中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource
```

```
   def create_metric_alarm(
       self,
       metric_namespace,
       metric_name,
       alarm_name,
       stat_type,
       period,
       eval_periods,
       threshold,
       comparison_op,
   ):
       """
       Creates an alarm that watches a metric.

       :param metric_namespace: The namespace of the metric.
       :param metric_name: The name of the metric.
       :param alarm_name: The name of the alarm.
       :param stat_type: The type of statistic the alarm watches.
       :param period: The period in which metric data are grouped to calculate
                      statistics.
       :param eval_periods: The number of periods that the metric must be over
the
                            alarm threshold before the alarm is set into an
alarmed
                            state.
       :param threshold: The threshold value to compare against the metric
statistic.
       :param comparison_op: The comparison operation used to compare the
threshold
                             against the metric.
       :return: The newly created alarm.
       """
       try:
           metric = self.cloudwatch_resource.Metric(metric_namespace,
metric_name)
           alarm = metric.put_alarm(
               AlarmName=alarm_name,
               Statistic=stat_type,
               Period=period,
               EvaluationPeriods=eval_periods,
               Threshold=threshold,
               ComparisonOperator=comparison_op,
           )
```

```
            logger.info(
                "Added alarm %s to track metric %s.%s.",
                alarm_name,
                metric_namespace,
                metric_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't add alarm %s to metric %s.%s",
                alarm_name,
                metric_namespace,
                metric_name,
            )
            raise
        else:
            return alarm
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件PutMetricAlarm中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
# Creates or updates an alarm in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm.
# @param alarm_description [String] A description about the alarm.
# @param metric_name [String] The name of the metric associated with the alarm.
# @param alarm_actions [Array] A list of Strings representing the
#   Amazon Resource Names (ARNs) to execute when the alarm transitions to the
#   ALARM state.
# @param namespace [String] The namespace for the metric to alarm on.
```

```
# @param statistic [String] The statistic for the metric.
# @param dimensions [Array] A list of dimensions for the metric, specified as
#   Aws::CloudWatch::Types::Dimension.
# @param period [Integer] The number of seconds before re-evaluating the metric.
# @param unit [String] The unit of measure for the statistic.
# @param evaluation_periods [Integer] The number of periods over which data is
#   compared to the specified threshold.
# @param theshold [Float] The value against which the specified statistic is
 compared.
# @param comparison_operator [String] The arithmetic operation to use when
#   comparing the specified statistic and threshold.
# @return [Boolean] true if the alarm was created or updated; otherwise, false.
# @example
#   exit 1 unless alarm_created_or_updated?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket',
#     'Objects exist in this bucket for more than 1 day.',
#     'NumberOfObjects',
#     ['arn:aws:sns:us-east-1:111111111111:Default_CloudWatch_Alarms_Topic'],
#     'AWS/S3',
#     'Average',
#     [
#       {
#         name: 'BucketName',
#         value: 'doc-example-bucket'
#       },
#       {
#         name: 'StorageType',
#         value: 'AllStorageTypes'
#       }
#     ],
#     86_400,
#     'Count',
#     1,
#     1,
#     'GreaterThanThreshold'
#   )
def alarm_created_or_updated?(
  cloudwatch_client,
  alarm_name,
  alarm_description,
  metric_name,
  alarm_actions,
  namespace,
```

```
  statistic,
  dimensions,
  period,
  unit,
  evaluation_periods,
  threshold,
  comparison_operator
)
  cloudwatch_client.put_metric_alarm(
    alarm_name: alarm_name,
    alarm_description: alarm_description,
    metric_name: metric_name,
    alarm_actions: alarm_actions,
    namespace: namespace,
    statistic: statistic,
    dimensions: dimensions,
    period: period,
    unit: unit,
    evaluation_periods: evaluation_periods,
    threshold: threshold,
    comparison_operator: comparison_operator
  )
  return true
rescue StandardError => e
  puts "Error creating alarm: #{e.message}"
  return false
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考PutMetricAlarm中的。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    TRY.
```

```
        lo_cwt->putmetricalarm(
          iv_alarmname                = iv_alarm_name
          iv_comparisonoperator       = iv_comparison_operator
          iv_evaluationperiods        = iv_evaluation_periods
          iv_metricname               = iv_metric_name
          iv_namespace                = iv_namespace
          iv_statistic                = iv_statistic
          iv_threshold                = iv_threshold
          iv_actionsenabled           = iv_actions_enabled
          iv_alarmdescription         = iv_alarm_description
          iv_unit                     = iv_unit
          iv_period                   = iv_period
          it_dimensions               = it_dimensions
        ).
        MESSAGE 'Alarm created.' TYPE 'I'.
      CATCH /aws1/cx_cwtlimitexceededfault.
        MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
    ENDTRY.
```

- 如需 API 詳細資訊，請參閱 AWS SDK PutMetricAlarm中的 SAP ABAP API 參考資料。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**PutMetricData**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutMetricData。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用指標、儀表板和警示
- 管理指標和警示

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Add some metric data using a call to a wrapper class.
    /// </summary>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <returns></returns>
    private static async Task<List<MetricDatum>> PutRandomMetricData(string
customMetricName,
        string customMetricNamespace)
    {
        List<MetricDatum> customData = new List<MetricDatum>();
        Random rnd = new Random();

        // Add 10 random values up to 100, starting with a timestamp 15 minutes
in the past.
        var utcNowMinus15 = DateTime.UtcNow.AddMinutes(-15);
        for (int i = 0; i < 10; i++)
        {
            var metricValue = rnd.Next(0, 100);
            customData.Add(
                new MetricDatum
                {
                    MetricName = customMetricName,
                    Value = metricValue,
                    TimestampUtc = utcNowMinus15.AddMinutes(i)
                }
            );
        }

        await _cloudWatchWrapper.PutMetricData(customMetricNamespace,
customData);
```

```
        return customData;
    }

    /// <summary>
    /// Wrapper to add metric data to a CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricData">A data object for the metric data.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricData(string metricNamespace,
        List<MetricDatum> metricData)
    {
        var putDataResponse = await _amazonCloudWatch.PutMetricDataAsync(
            new PutMetricDataRequest()
            {
                MetricData = metricData,
                Namespace = metricNamespace,
            });

        return putDataResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考PutMetricData中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

包括必需的檔案。

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricDataRequest.h>
#include <iostream>
```

將資料放入指標。

```
        Aws::CloudWatch::CloudWatchClient cw;

        Aws::CloudWatch::Model::Dimension dimension;
        dimension.SetName("UNIQUE_PAGES");
        dimension.SetValue("URLS");

        Aws::CloudWatch::Model::MetricDatum datum;
        datum.SetMetricName("PAGES_VISITED");
        datum.SetUnit(Aws::CloudWatch::Model::StandardUnit::None);
        datum.SetValue(data_point);
        datum.AddDimensions(dimension);

        Aws::CloudWatch::Model::PutMetricDataRequest request;
        request.SetNamespace("SITE/TRAFFIC");
        request.AddMetricData(datum);

        auto outcome = cw.PutMetricData(request);
        if (!outcome.IsSuccess())
        {
            std::cout << "Failed to put sample metric data:" <<
                outcome.GetError().GetMessage() << std::endl;
        }
        else
        {
            std::cout << "Successfully put sample metric data" << std::endl;
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考PutMetricData中的。

CLI

AWS CLI

若要將自訂指標發佈到 Amazon CloudWatch

下列範例使用`put-metric-data`命令將自訂指標發佈到 Amazon CloudWatch：

```
aws cloudwatch put-metric-data --namespace "Usage Metrics" --metric-data file://
metric.json
```

指標本身的值會儲存在 JSON 檔案 `metric.json` 中。

以下是該檔案的內容：

```
[
  {
    "MetricName": "New Posts",
    "Timestamp": "Wednesday, June 12, 2013 8:28:20 PM",
    "Value": 0.50,
    "Unit": "Count"
  }
]
```

如需詳細資訊，請參閱 Amazon CloudWatch 開發人員指南中的發佈自訂指標。

指定多個維度

下列範例說明如何指定多個維度。每個維度均指定為 Name=Value 對。使用逗號分隔多個維度：

```
aws cloudwatch put-metric-data --metric-name Buffers --namespace
 MyNameSpace --unit Bytes --value 231434333 --dimensions
 InstanceID=1-23456789,InstanceType=m1.small
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[PutMetricData](#)中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
   public static void addMetricDataForAlarm(CloudWatchClient cw, String
fileName) {
       try {
           // Read values from the JSON file.
           JsonParser parser = new JsonFactory().createParser(new
File(fileName));
           com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
           String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
           String customMetricName =
rootNode.findValue("customMetricName").asText();

           // Set an Instant object.
           String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
           Instant instant = Instant.parse(time);

           MetricDatum datum = MetricDatum.builder()
                   .metricName(customMetricName)
                   .unit(StandardUnit.NONE)
                   .value(1001.00)
                   .timestamp(instant)
                   .build();

           MetricDatum datum2 = MetricDatum.builder()
                   .metricName(customMetricName)
                   .unit(StandardUnit.NONE)
                   .value(1002.00)
                   .timestamp(instant)
                   .build();

           List<MetricDatum> metricDataList = new ArrayList<>();
           metricDataList.add(datum);
           metricDataList.add(datum2);

           PutMetricDataRequest request = PutMetricDataRequest.builder()
                   .namespace(customMetricNamespace)
                   .metricData(metricDataList)
                   .build();

           cw.putMetricData(request);
```

```
            System.out.println("Added metric values for for metric " +
 customMetricName);

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考PutMetricData中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

匯入 SDK 和用戶端模組，然後呼叫 API。

```
import { PutMetricDataCommand } from "@aws-sdk/client-cloudwatch";
import { client } from "../libs/client.js";

const run = async () => {
  // See https://docs.aws.amazon.com/AmazonCloudWatch/latest/APIReference/
API_PutMetricData.html#API_PutMetricData_RequestParameters
  // and https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
publishingMetrics.html
  // for more information about the parameters in this command.
  const command = new PutMetricDataCommand({
    MetricData: [
      {
        MetricName: "PAGES_VISITED",
        Dimensions: [
          {
            Name: "UNIQUE_PAGES",
            Value: "URLS",
          },
```

```
        ],
        Unit: "None",
        Value: 1.0,
      },
    ],
    Namespace: "SITE/TRAFFIC",
  });

  try {
    return await client.send(command);
  } catch (err) {
    console.error(err);
  }
};

export default run();
```

在單獨的模組中建立用戶端並將其匯出。

```
import { CloudWatchClient } from "@aws-sdk/client-cloudwatch";

export const client = new CloudWatchClient({});
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考PutMetricData中的。

適用於 JavaScript (v2) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在 AWS 程式碼範例儲存庫中設定和執行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });
```

```
// Create CloudWatch service object
var cw = new AWS.CloudWatch({ apiVersion: "2010-08-01" });

// Create parameters JSON for putMetricData
var params = {
  MetricData: [
    {
      MetricName: "PAGES_VISITED",
      Dimensions: [
        {
          Name: "UNIQUE_PAGES",
          Value: "URLS",
        },
      ],
      Unit: "None",
      Value: 1.0,
    },
  ],
  Namespace: "SITE/TRAFFIC",
};

cw.putMetricData(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", JSON.stringify(data));
  }
});
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考PutMetricData中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum = MetricDatum {
        metricName = customMetricName
        unit = StandardUnit.None
        value = 1001.00
        timestamp = aws.smithy.kotlin.runtime.time.Instant(instant)
    }

    val datum2 = MetricDatum {
        metricName = customMetricName
        unit = StandardUnit.None
        value = 1002.00
        timestamp = aws.smithy.kotlin.runtime.time.Instant(instant)
    }

    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)

    val request = PutMetricDataRequest {
        namespace = customMetricNamespace
        metricData = metricDataList
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK PutMetricData中的 Kotlin API 參考。

PowerShell

適用的工具 PowerShell

範例 1：建立新 MetricDatum 物件，並將其寫入 Amazon Web Services CloudWatch 指標。

```
### Create a MetricDatum .NET object
$Metric = New-Object -TypeName Amazon.CloudWatch.Model.MetricDatum
$Metric.Timestamp = [DateTime]::UtcNow
$Metric.MetricName = 'CPU'
$Metric.Value = 50

### Write the metric data to the CloudWatch service
Write-CWMetricData -Namespace instance1 -MetricData $Metric
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程PutMetricData式參考中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data(self, namespace, name, value, unit):
        """
        Sends a single data value to CloudWatch for a metric. This metric is
 given
```

```
        a timestamp of the current UTC time.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param value: The value of the metric.
        :param unit: The unit of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{"MetricName": name, "Value": value, "Unit": unit}],
            )
            logger.info("Put data for metric %s.%s", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data for metric %s.%s", namespace,
 name)
            raise
```

將一組資料放入指 CloudWatch 標中。

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
        """
        Sends a set of data to CloudWatch for a metric. All of the data in the
 set
        have the same timestamp and unit.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param timestamp: The UTC timestamp for the metric.
        :param unit: The unit of the metric.
```

```
        :param data_set: The set of data to send. This set is a dictionary that
                         contains a list of values and a list of corresponding
counts.
                         The value and count lists must be the same length.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[
                    {
                        "MetricName": name,
                        "Timestamp": timestamp,
                        "Values": data_set["values"],
                        "Counts": data_set["counts"],
                        "Unit": unit,
                    }
                ],
            )
            logger.info("Put data set for metric %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data set for metric %s.%s.",
namespace, name)
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件PutMetricData中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
require "aws-sdk-cloudwatch"

# Adds a datapoint to a metric in Amazon CloudWatch.
```

```
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric to add the
#   datapoint to.
# @param metric_name [String] The name of the metric to add the datapoint to.
# @param dimension_name [String] The name of the dimension to add the
#   datapoint to.
# @param dimension_value [String] The value of the dimension to add the
#   datapoint to.
# @param metric_value [Float] The value of the datapoint.
# @param metric_unit [String] The unit of measurement for the datapoint.
# @return [Boolean]
# @example
#   exit 1 unless datapoint_added_to_metric?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC',
#     'UniqueVisitors',
#     'SiteName',
#     'example.com',
#     5_885.0,
#     'Count'
#   )
def datapoint_added_to_metric?(
  cloudwatch_client,
  metric_namespace,
  metric_name,
  dimension_name,
  dimension_value,
  metric_value,
  metric_unit
)
  cloudwatch_client.put_metric_data(
    namespace: metric_namespace,
    metric_data: [
      {
        metric_name: metric_name,
        dimensions: [
          {
            name: dimension_name,
            value: dimension_value
          }
        ],
        value: metric_value,
```

```
        unit: metric_unit
      }
    ]
  )
  puts "Added data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}'."
  return true
rescue StandardError => e
  puts "Error adding data about '#{metric_name}' to namespace " \
    "'#{metric_namespace}': #{e.message}"
  return false
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考PutMetricData中的。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

CloudWatch 使用 AWS SDK 的案例

下列程式碼範例說明如何在 AWS SDK 中 CloudWatch實作常見案例。這些案例會示範如何透過在其中呼叫多個函式來完成特定工作 CloudWatch。每個案例都包含一個連結 GitHub，您可以在其中找到如何設定和執行程式碼的指示。

範例

- 使用 AWS SDK 開始使用 CloudWatch 警示
- 使用 AWS SDK 開始使用 CloudWatch 指標、儀表板和警示
- 使用 AWS SDK 管理 CloudWatch 指標和警示

使用 AWS SDK 開始使用 CloudWatch 警示

以下程式碼範例顯示做法：

- 建立警示。
- 停用警示動作。
- 描述警示。
- 刪除警示。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
DATA lt_alarmnames TYPE /aws1/cl_cwtalarmnames_w=>tt_alarmnames.
DATA lo_alarmname TYPE REF TO /aws1/cl_cwtalarmnames_w.

"Create an alarm"
TRY.
    lo_cwt->putmetricalarm(
      iv_alarmname                = iv_alarm_name
      iv_comparisonoperator       = iv_comparison_operator
      iv_evaluationperiods        = iv_evaluation_periods
      iv_metricname               = iv_metric_name
      iv_namespace                = iv_namespace
      iv_statistic                = iv_statistic
      iv_threshold                = iv_threshold
      iv_actionsenabled           = iv_actions_enabled
      iv_alarmdescription         = iv_alarm_description
      iv_unit                     = iv_unit
      iv_period                   = iv_period
      it_dimensions               = it_dimensions
    ).
    MESSAGE 'Alarm created' TYPE 'I'.
  CATCH /aws1/cx_cwtlimitexceededfault.
    MESSAGE 'The request processing has exceeded the limit' TYPE 'E'.
ENDTRY.

"Create an ABAP internal table for the created alarm."
CREATE OBJECT lo_alarmname EXPORTING iv_value = iv_alarm_name.
INSERT lo_alarmname INTO TABLE lt_alarmnames.

"Disable alarm actions."
TRY.
    lo_cwt->disablealarmactions(
```

```
          it_alarmnames                = lt_alarmnames
        ).
        MESSAGE 'Alarm actions disabled' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_disablealarm_exception).
        DATA(lv_disablealarm_error) = |"{ lo_disablealarm_exception-
>av_err_code }" - { lo_disablealarm_exception->av_err_msg }|.
        MESSAGE lv_disablealarm_error TYPE 'E'.
    ENDTRY.

    "Describe alarm using the same ABAP internal table."
    TRY.
        oo_result = lo_cwt->describealarms(                       " oo_result is
 returned for testing purpose "
          it_alarmnames                = lt_alarmnames
        ).
        MESSAGE 'Alarms retrieved' TYPE 'I'.
      CATCH /aws1/cx_rt_service_generic INTO DATA(lo_describealarms_exception).
        DATA(lv_describealarms_error) = |"{ lo_describealarms_exception-
>av_err_code }" - { lo_describealarms_exception->av_err_msg }|.
        MESSAGE lv_describealarms_error TYPE 'E'.
    ENDTRY.

    "Delete alarm."
    TRY.
        lo_cwt->deletealarms(
          it_alarmnames = lt_alarmnames
        ).
        MESSAGE 'Alarms deleted' TYPE 'I'.
      CATCH /aws1/cx_cwtresourcenotfound .
        MESSAGE 'Resource being access is not found.' TYPE 'E'.
    ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP 的AWS SDK API 參考》中的下列主題。
  - DeleteAlarms
  - DescribeAlarms
  - DisableAlarmActions
  - PutMetricAlarm

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDK 開始使用 CloudWatch 指標、儀表板和警示

下列程式碼範例示範如何：

- 列出 CloudWatch 命名空間和測量結果。
- 取得指標和預估帳單的統計資料。
- 建立並更新儀表板。
- 建立資料並將其新增至指標。
- 建立並觸發警示，然後檢視警示歷史記錄。
- 新增異常偵測器。
- 取得指標映像，然後清除資源。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

在命令提示中執行互動式案例。

```
public class CloudWatchScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    To enable billing metrics and statistics for this example, make sure billing
 alerts are enabled for your account:
    https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics

    This .NET example performs the following tasks:
        1. List and select a CloudWatch namespace.
        2. List and select a CloudWatch metric.
        3. Get statistics for a CloudWatch metric.
```

```
        4. Get estimated billing statistics for the last week.
        5. Create a new CloudWatch dashboard with two metrics.
        6. List current CloudWatch dashboards.
        7. Create a CloudWatch custom metric and add metric data.
        8. Add the custom metric to the dashboard.
        9. Create a CloudWatch alarm for the custom metric.
       10. Describe current CloudWatch alarms.
       11. Get recent data for the custom metric.
       12. Add data to the custom metric to trigger the alarm.
       13. Wait for an alarm state.
       14. Get history for the CloudWatch alarm.
       15. Add an anomaly detector.
       16. Describe current anomaly detectors.
       17. Get and display a metric image.
       18. Clean up resources.
    */

    private static ILogger logger = null!;
    private static CloudWatchWrapper _cloudWatchWrapper = null!;
    private static IConfiguration _configuration = null!;
    private static readonly List<string> _statTypes = new List<string>
{ "SampleCount", "Average", "Sum", "Minimum", "Maximum" };
    private static SingleMetricAnomalyDetector? anomalyDetector = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonCloudWatch>()
            .AddTransient<CloudWatchWrapper>()
        )
        .Build();

        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
```

```
                true) // Optionally, load local settings.
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<CloudWatchScenario>();

        _cloudWatchWrapper =
host.Services.GetRequiredService<CloudWatchWrapper>();

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon CloudWatch example scenario.");
        Console.WriteLine(new string('-', 80));

        try
        {
            var selectedNamespace = await SelectNamespace();
            var selectedMetric = await SelectMetric(selectedNamespace);
            await GetAndDisplayMetricStatistics(selectedNamespace,
selectedMetric);
            await GetAndDisplayEstimatedBilling();
            await CreateDashboardWithMetrics();
            await ListDashboards();
            await CreateNewCustomMetric();
            await AddMetricToDashboard();
            await CreateMetricAlarm();
            await DescribeAlarms();
            await GetCustomMetricData();
            await AddMetricDataForAlarm();
            await CheckForMetricAlarm();
            await GetAlarmHistory();
            anomalyDetector = await AddAnomalyDetector();
            await DescribeAnomalyDetectors();
            await GetAndOpenMetricImage();
            await CleanupResources();
        }
        catch (Exception ex)
        {
            logger.LogError(ex, "There was a problem executing the scenario.");
            await CleanupResources();
        }

    }

    /// <summary>
```

```
    /// Select a namespace.
    /// </summary>
    /// <returns>The selected namespace.</returns>
    private static async Task<string> SelectNamespace()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"1. Select a CloudWatch Namespace from a list of
Namespaces.");
        var metrics = await _cloudWatchWrapper.ListMetrics();
        // Get a distinct list of namespaces.
        var namespaces = metrics.Select(m => m.Namespace).Distinct().ToList();
        for (int i = 0; i < namespaces.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {namespaces[i]}");
        }

        var namespaceChoiceNumber = 0;
        while (namespaceChoiceNumber < 1 || namespaceChoiceNumber >
namespaces.Count)
        {
            Console.WriteLine(
                "Select a namespace by entering a number from the preceding
list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out namespaceChoiceNumber);
        }

        var selectedNamespace = namespaces[namespaceChoiceNumber - 1];

        Console.WriteLine(new string('-', 80));

        return selectedNamespace;
    }

    /// <summary>
    /// Select a metric from a namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <returns>The metric name.</returns>
    private static async Task<Metric> SelectMetric(string metricNamespace)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"2. Select a CloudWatch metric from a namespace.");
```

```
        var namespaceMetrics = await
_cloudWatchWrapper.ListMetrics(metricNamespace);

        for (int i = 0; i < namespaceMetrics.Count && i < 15; i++)
        {
            var dimensionsWithValues = namespaceMetrics[i].Dimensions
                .Where(d => !string.Equals("None", d.Value));
            Console.WriteLine($"\t{i + 1}. {namespaceMetrics[i].MetricName} " +
                              $"{string.Join(", :", dimensionsWithValues.Select(d
=> d.Value))}");
        }

        var metricChoiceNumber = 0;
        while (metricChoiceNumber < 1 || metricChoiceNumber >
namespaceMetrics.Count)
        {
            Console.WriteLine(
                "Select a metric by entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out metricChoiceNumber);
        }

        var selectedMetric = namespaceMetrics[metricChoiceNumber - 1];

        Console.WriteLine(new string('-', 80));

        return selectedMetric;
    }

    /// <summary>
    /// Get and display metric statistics for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task GetAndDisplayMetricStatistics(string
metricNamespace, Metric metric)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"3. Get CloudWatch metric statistics for the last
day.");

        for (int i = 0; i < _statTypes.Count; i++)
        {
```

```
            Console.WriteLine($"\t{i + 1}. {_statTypes[i]}");
        }

        var statisticChoiceNumber = 0;
        while (statisticChoiceNumber < 1 || statisticChoiceNumber >
_statTypes.Count)
        {
            Console.WriteLine(
                "Select a metric statistic by entering a number from the
preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out statisticChoiceNumber);
        }

        var selectedStatistic = _statTypes[statisticChoiceNumber - 1];
        var statisticsList = new List<string> { selectedStatistic };

        var metricStatistics = await
_cloudWatchWrapper.GetMetricStatistics(metricNamespace, metric.MetricName,
statisticsList, metric.Dimensions, 1, 60);

        if (!metricStatistics.Any())
        {
            Console.WriteLine($"No {selectedStatistic} statistics found for
{metric} in namespace {metricNamespace}.");
        }

        metricStatistics = metricStatistics.OrderBy(s => s.Timestamp).ToList();
        for (int i = 0; i < metricStatistics.Count && i < 10; i++)
        {
            var metricStat = metricStatistics[i];
            var statValue =
metricStat.GetType().GetProperty(selectedStatistic)!.GetValue(metricStat, null);
            Console.WriteLine($"\t{i + 1}. Timestamp
{metricStatistics[i].Timestamp:G} {selectedStatistic}: {statValue}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get and display estimated billing statistics.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
```

```
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task GetAndDisplayEstimatedBilling()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"4. Get CloudWatch estimated billing for the last
week.");

        var billingStatistics = await SetupBillingStatistics();

        for (int i = 0; i < billingStatistics.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. Timestamp
{billingStatistics[i].Timestamp:G} : {billingStatistics[i].Maximum}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get billing statistics using a call to a wrapper class.
    /// </summary>
    /// <returns>A collection of billing statistics.</returns>
    private static async Task<List<Datapoint>> SetupBillingStatistics()
    {
        // Make a request for EstimatedCharges with a period of one day for the
past seven days.
        var billingStatistics = await _cloudWatchWrapper.GetMetricStatistics(
            "AWS/Billing",
            "EstimatedCharges",
            new List<string>() { "Maximum" },
            new List<Dimension>() { new Dimension { Name = "Currency", Value =
"USD" } },
            7,
            86400);

        billingStatistics = billingStatistics.OrderBy(n => n.Timestamp).ToList();

        return billingStatistics;
    }

    /// <summary>
    /// Create a dashboard with metrics.
    /// </summary>
```

```
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task CreateDashboardWithMetrics()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"5. Create a new CloudWatch dashboard with metrics.");
        var dashboardName = _configuration["dashboardName"];
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);
        var newDashboardString = JsonSerializer.Serialize(
            newDashboard,
            new JsonSerializerOptions
            {
                DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull
            });
        var validationMessages =
            await _cloudWatchWrapper.PutDashboard(dashboardName,
newDashboardString);

        Console.WriteLine(validationMessages.Any() ? $"\tValidation messages:" :
null);
        for (int i = 0; i < validationMessages.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {validationMessages[i].Message}");
        }
        Console.WriteLine($"\tDashboard {dashboardName} was created.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// List dashboards.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task ListDashboards()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"6. List the CloudWatch dashboards in the current
account.");

        var dashboards = await _cloudWatchWrapper.ListDashboards();

        for (int i = 0; i < dashboards.Count; i++)
        {
```

```
            Console.WriteLine($"\t{i + 1}. {dashboards[i].DashboardName}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Create and add data for a new custom metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateNewCustomMetric()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"7. Create and add data for a new custom metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var customData = await PutRandomMetricData(customMetricName,
customMetricNamespace);

        var valuesString = string.Join(',', customData.Select(d => d.Value));
        Console.WriteLine($"\tAdded metric values for for metric
{customMetricName}: \n\t{valuesString}");

        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// Add some metric data using a call to a wrapper class.
    /// </summary>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <returns></returns>
    private static async Task<List<MetricDatum>> PutRandomMetricData(string
customMetricName,
        string customMetricNamespace)
    {
        List<MetricDatum> customData = new List<MetricDatum>();
        Random rnd = new Random();

        // Add 10 random values up to 100, starting with a timestamp 15 minutes
in the past.
```

```
        var utcNowMinus15 = DateTime.UtcNow.AddMinutes(-15);
        for (int i = 0; i < 10; i++)
        {
            var metricValue = rnd.Next(0, 100);
            customData.Add(
                new MetricDatum
                {
                    MetricName = customMetricName,
                    Value = metricValue,
                    TimestampUtc = utcNowMinus15.AddMinutes(i)
                }
            );
        }

        await _cloudWatchWrapper.PutMetricData(customMetricNamespace,
customData);
        return customData;
    }

    /// <summary>
    /// Add the custom metric to the dashboard.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task AddMetricToDashboard()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"8. Add the new custom metric to the dashboard.");

        var dashboardName = _configuration["dashboardName"];

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var validationMessages = await SetupDashboard(customMetricNamespace,
customMetricName, dashboardName);

        Console.WriteLine(validationMessages.Any() ? $"\tValidation messages:" :
null);
        for (int i = 0; i < validationMessages.Count; i++)
        {
            Console.WriteLine($"\t{i + 1}. {validationMessages[i].Message}");
        }
        Console.WriteLine($"\tDashboard {dashboardName} updated with metric
{customMetricName}.");
```

```
        Console.WriteLine(new string('-', 80));
    }


    /// <summary>
    /// Set up a dashboard using a call to the wrapper class.
    /// </summary>
    /// <param name="customMetricNamespace">The metric namespace.</param>
    /// <param name="customMetricName">The metric name.</param>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A list of validation messages.</returns>
    private static async Task<List<DashboardValidationMessage>> SetupDashboard(
        string customMetricNamespace, string customMetricName, string
dashboardName)
    {
        // Get the dashboard model from configuration.
        var newDashboard = new DashboardModel();
        _configuration.GetSection("dashboardExampleBody").Bind(newDashboard);

        // Add a new metric to the dashboard.
        newDashboard.Widgets.Add(new Widget
        {
            Height = 8,
            Width = 8,
            Y = 8,
            X = 0,
            Type = "metric",
            Properties = new Properties
            {
                Metrics = new List<List<object>>
                    { new() { customMetricNamespace, customMetricName } },
                View = "timeSeries",
                Region = "us-east-1",
                Stat = "Sum",
                Period = 86400,
                YAxis = new YAxis { Left = new Left { Min = 0, Max = 100 } },
                Title = "Custom Metric Widget",
                LiveData = true,
                Sparkline = true,
                Trend = true,
                Stacked = false,
                SetPeriodToTimeRange = false
            }
        });
```

```
        var newDashboardString = JsonSerializer.Serialize(newDashboard,
            new JsonSerializerOptions
            { DefaultIgnoreCondition = JsonIgnoreCondition.WhenWritingNull });
        var validationMessages =
            await _cloudWatchWrapper.PutDashboard(dashboardName,
newDashboardString);

        return validationMessages;
    }

    /// <summary>
    /// Create a CloudWatch alarm for the new metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CreateMetricAlarm()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"9. Create a CloudWatch alarm for the new metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var alarmName = _configuration["exampleAlarmName"];
        var accountId = _configuration["accountId"];
        var region = _configuration["region"];
        var emailTopic = _configuration["emailTopic"];
        var alarmActions = new List<string>();

        if (GetYesNoResponse(
                $"\tAdd an email action for topic {emailTopic} to alarm
{alarmName}? (y/n)"))
        {
            _cloudWatchWrapper.AddEmailAlarmAction(accountId, region, emailTopic,
alarmActions);
        }

        await _cloudWatchWrapper.PutMetricEmailAlarm(
            "Example metric alarm",
            alarmName,
            ComparisonOperator.GreaterThanOrEqualToThreshold,
            customMetricName,
            customMetricNamespace,
            100,
```

```
            alarmActions);

        Console.WriteLine($"\tAlarm {alarmName} added for metric
{customMetricName}.");
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Describe Alarms.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task DescribeAlarms()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"10. Describe CloudWatch alarms in the current
account.");

        var alarms = await _cloudWatchWrapper.DescribeAlarms();
        alarms = alarms.OrderByDescending(a => a.StateUpdatedTimestamp).ToList();

        for (int i = 0; i < alarms.Count && i < 10; i++)
        {
            var alarm = alarms[i];
            Console.WriteLine($"\t{i + 1}. {alarm.AlarmName}");
            Console.WriteLine($"\tState: {alarm.StateValue} for
{alarm.MetricName} {alarm.ComparisonOperator} {alarm.Threshold}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get the recent data for the metric.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetCustomMetricData()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"11. Get current data for new custom metric.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var accountId = _configuration["accountId"];
```

```
        var query = new List<MetricDataQuery>
        {
            new MetricDataQuery
            {
                AccountId = accountId,
                Id = "m1",
                Label = "Custom Metric Data",
                MetricStat = new MetricStat
                {
                    Metric = new Metric
                    {
                        MetricName = customMetricName,
                        Namespace = customMetricNamespace,
                    },
                    Period = 1,
                    Stat = "Maximum"
                }
            }
        };

        var metricData = await _cloudWatchWrapper.GetMetricData(
            20,
            true,
            DateTime.UtcNow.AddMinutes(1),
            20,
            query);

        for (int i = 0; i < metricData.Count; i++)
        {
            for (int j = 0; j < metricData[i].Values.Count; j++)
            {
                Console.WriteLine(
                    $"\tTimestamp {metricData[i].Timestamps[j]:G} Value:
{metricData[i].Values[j]}");
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add metric data to trigger an alarm.
    /// </summary>
    /// <returns>Async task.</returns>
```

```
    private static async Task AddMetricDataForAlarm()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"12. Add metric data to the custom metric to trigger
an alarm.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var nowUtc = DateTime.UtcNow;
        List<MetricDatum> customData = new List<MetricDatum>
        {
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc.AddMinutes(-2)
            },
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc.AddMinutes(-1)
            },
            new MetricDatum
            {
                MetricName = customMetricName,
                Value = 101,
                TimestampUtc = nowUtc
            }
        };
        var valuesString = string.Join(',', customData.Select(d => d.Value));
        Console.WriteLine($"\tAdded metric values for for metric
{customMetricName}: \n\t{valuesString}");
        await _cloudWatchWrapper.PutMetricData(customMetricNamespace,
customData);

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Check for a metric alarm using the DescribeAlarmsForMetric action.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CheckForMetricAlarm()
```

```
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"13. Checking for an alarm state.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];
        var hasAlarm = false;
        var retries = 10;
        while (!hasAlarm && retries > 0)
        {
            var alarms = await
_cloudWatchWrapper.DescribeAlarmsForMetric(customMetricNamespace,
customMetricName);
            hasAlarm = alarms.Any(a => a.StateValue == StateValue.ALARM);
            retries--;
            Thread.Sleep(20000);
        }

        Console.WriteLine(hasAlarm
            ? $"\tAlarm state found for {customMetricName}."
            : $"\tNo Alarm state found for {customMetricName} after 10
retries.");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get history for an alarm.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetAlarmHistory()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"14. Get alarm history.");

        var exampleAlarmName = _configuration["exampleAlarmName"];

        var alarmHistory = await
_cloudWatchWrapper.DescribeAlarmHistory(exampleAlarmName, 2);

        for (int i = 0; i < alarmHistory.Count; i++)
        {
            var history = alarmHistory[i];
```

```
            Console.WriteLine($"\t{i + 1}. {history.HistorySummary}, time
{history.Timestamp:g}");
        }
        if (!alarmHistory.Any())
        {
            Console.WriteLine($"\tNo alarm history data found for
{exampleAlarmName}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Add an anomaly detector.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<SingleMetricAnomalyDetector> AddAnomalyDetector()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"15. Add an anomaly detector.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var detector = new SingleMetricAnomalyDetector
        {
            MetricName = customMetricName,
            Namespace = customMetricNamespace,
            Stat = "Maximum"
        };
        await _cloudWatchWrapper.PutAnomalyDetector(detector);
        Console.WriteLine($"\tAdded anomaly detector for metric
{customMetricName}.");

        Console.WriteLine(new string('-', 80));
        return detector;
    }

    /// <summary>
    /// Describe anomaly detectors.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task DescribeAnomalyDetectors()
    {
```

```
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"16. Describe anomaly detectors in the current
account.");

        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var detectors = await
_cloudWatchWrapper.DescribeAnomalyDetectors(customMetricNamespace,
customMetricName);

        for (int i = 0; i < detectors.Count; i++)
        {
            var detector = detectors[i];
            Console.WriteLine($"\t{i + 1}.
{detector.SingleMetricAnomalyDetector.MetricName}, state
{detector.StateValue}");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Fetch and open a metrics image for a CloudWatch metric and namespace.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task GetAndOpenMetricImage()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("17. Get a metric image from CloudWatch.");

        Console.WriteLine($"\tGetting Image data for custom metric.");
        var customMetricNamespace = _configuration["customMetricNamespace"];
        var customMetricName = _configuration["customMetricName"];

        var memoryStream = await
_cloudWatchWrapper.GetTimeSeriesMetricImage(customMetricNamespace,
customMetricName, "Maximum", 10);
        var file = _cloudWatchWrapper.SaveMetricImage(memoryStream,
"MetricImages");

        ProcessStartInfo info = new ProcessStartInfo();

        Console.WriteLine($"\tFile saved as {Path.GetFileName(file)}.");
```

```
        Console.WriteLine($"\tPress enter to open the image.");
        Console.ReadLine();
        info.FileName = Path.Combine("ms-photos://", file);
        info.UseShellExecute = true;
        info.CreateNoWindow = true;
        info.Verb = string.Empty;

        Process.Start(info);

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up created resources.
    /// </summary>
    /// <param name="metricNamespace">The namespace for metrics.</param>
    /// <param name="metric">The CloudWatch metric.</param>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"18. Clean up resources.");

        var dashboardName = _configuration["dashboardName"];
        if (GetYesNoResponse($"\tDelete dashboard {dashboardName}? (y/n)"))
        {
            Console.WriteLine($"\tDeleting dashboard.");
            var dashboardList = new List<string> { dashboardName };
            await _cloudWatchWrapper.DeleteDashboards(dashboardList);
        }

        var alarmName = _configuration["exampleAlarmName"];
        if (GetYesNoResponse($"\tDelete alarm {alarmName}? (y/n)"))
        {
            Console.WriteLine($"\tCleaning up alarms.");
            var alarms = new List<string> { alarmName };
            await _cloudWatchWrapper.DeleteAlarms(alarms);
        }

        if (GetYesNoResponse($"\tDelete anomaly detector? (y/n)") &&
anomalyDetector != null)
        {
            Console.WriteLine($"\tCleaning up anomaly detector.");
```

```
            await _cloudWatchWrapper.DeleteAnomalyDetector(
                anomalyDetector);
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
}
```

案例用於動 CloudWatch 作的包裝方法。

```
/// <summary>
/// Wrapper class for Amazon CloudWatch methods.
/// </summary>
public class CloudWatchWrapper
{
    private readonly IAmazonCloudWatch _amazonCloudWatch;
    private readonly ILogger<CloudWatchWrapper> _logger;

    /// <summary>
    /// Constructor for the CloudWatch wrapper.
    /// </summary>
    /// <param name="amazonCloudWatch">The injected CloudWatch client.</param>
    /// <param name="logger">The injected logger for the wrapper.</param>
    public CloudWatchWrapper(IAmazonCloudWatch amazonCloudWatch,
 ILogger<CloudWatchWrapper> logger)
```

```
    {
        _logger = logger;
        _amazonCloudWatch = amazonCloudWatch;
    }

    /// <summary>
    /// List metrics available, optionally within a namespace.
    /// </summary>
    /// <param name="metricNamespace">Optional CloudWatch namespace to use when
listing metrics.</param>
    /// <param name="filter">Optional dimension filter.</param>
    /// <param name="metricName">Optional metric name filter.</param>
    /// <returns>The list of metrics.</returns>
    public async Task<List<Metric>> ListMetrics(string? metricNamespace = null,
DimensionFilter? filter = null, string? metricName = null)
    {
        var results = new List<Metric>();
        var paginateMetrics = _amazonCloudWatch.Paginators.ListMetrics(
            new ListMetricsRequest
            {
                Namespace = metricNamespace,
                Dimensions = filter != null ? new List<DimensionFilter>
{ filter } : null,
                MetricName = metricName
            });
        // Get the entire list using the paginator.
        await foreach (var metric in paginateMetrics.Metrics)
        {
            results.Add(metric);
        }

        return results;
    }

    /// <summary>
    /// Wrapper to get statistics for a specific CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <param name="statistics">The list of statistics to include.</param>
    /// <param name="dimensions">The list of dimensions to include.</param>
    /// <param name="days">The number of days in the past to include.</param>
    /// <param name="period">The period for the data.</param>
    /// <returns>A list of DataPoint objects for the statistics.</returns>
```

```
    public async Task<List<Datapoint>> GetMetricStatistics(string
 metricNamespace,
        string metricName, List<string> statistics, List<Dimension> dimensions,
 int days, int period)
    {
        var metricStatistics = await _amazonCloudWatch.GetMetricStatisticsAsync(
            new GetMetricStatisticsRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName,
                Dimensions = dimensions,
                Statistics = statistics,
                StartTimeUtc = DateTime.UtcNow.AddDays(-days),
                EndTimeUtc = DateTime.UtcNow,
                Period = period
            });

        return metricStatistics.Datapoints;
    }

    /// <summary>
    /// Wrapper to create or add to a dashboard with metrics.
    /// </summary>
    /// <param name="dashboardName">The name for the dashboard.</param>
    /// <param name="dashboardBody">The metric data in JSON for the dashboard.</
param>
    /// <returns>A list of validation messages for the dashboard.</returns>
    public async Task<List<DashboardValidationMessage>> PutDashboard(string
 dashboardName,
        string dashboardBody)
    {
        // Updating a dashboard replaces all contents.
        // Best practice is to include a text widget indicating this dashboard
 was created programmatically.
        var dashboardResponse = await _amazonCloudWatch.PutDashboardAsync(
            new PutDashboardRequest()
            {
                DashboardName = dashboardName,
                DashboardBody = dashboardBody
            });

        return dashboardResponse.DashboardValidationMessages;
    }
```

```
    /// <summary>
    /// Get information on a dashboard.
    /// </summary>
    /// <param name="dashboardName">The name of the dashboard.</param>
    /// <returns>A JSON object with dashboard information.</returns>
    public async Task<string> GetDashboard(string dashboardName)
    {
        var dashboardResponse = await _amazonCloudWatch.GetDashboardAsync(
            new GetDashboardRequest()
            {
                DashboardName = dashboardName
            });

        return dashboardResponse.DashboardBody;
    }


    /// <summary>
    /// Get a list of dashboards.
    /// </summary>
    /// <returns>A list of DashboardEntry objects.</returns>
    public async Task<List<DashboardEntry>> ListDashboards()
    {
        var results = new List<DashboardEntry>();
        var paginateDashboards = _amazonCloudWatch.Paginators.ListDashboards(
            new ListDashboardsRequest());
        // Get the entire list using the paginator.
        await foreach (var data in paginateDashboards.DashboardEntries)
        {
            results.Add(data);
        }

        return results;
    }

    /// <summary>
    /// Wrapper to add metric data to a CloudWatch metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricData">A data object for the metric data.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricData(string metricNamespace,
        List<MetricDatum> metricData)
```

```
    {
        var putDataResponse = await _amazonCloudWatch.PutMetricDataAsync(
            new PutMetricDataRequest()
            {
                MetricData = metricData,
                Namespace = metricNamespace,
            });

        return putDataResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Get an image for a metric graphed over time.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metric">The name of the metric.</param>
    /// <param name="stat">The name of the stat to chart.</param>
    /// <param name="period">The period to use for the chart.</param>
    /// <returns>A memory stream for the chart image.</returns>
    public async Task<MemoryStream> GetTimeSeriesMetricImage(string
metricNamespace, string metric, string stat, int period)
    {
        var metricImageWidget = new
        {
            title = "Example Metric Graph",
            view = "timeSeries",
            stacked = false,
            period = period,
            width = 1400,
            height = 600,
            metrics = new List<List<object>>
                { new() { metricNamespace, metric, new { stat } } }
        };

        var metricImageWidgetString =
JsonSerializer.Serialize(metricImageWidget);
        var imageResponse = await _amazonCloudWatch.GetMetricWidgetImageAsync(
            new GetMetricWidgetImageRequest()
            {
                MetricWidget = metricImageWidgetString
            });

        return imageResponse.MetricWidgetImage;
    }
```

```
    /// <summary>
    /// Save a metric image to a file.
    /// </summary>
    /// <param name="memoryStream">The MemoryStream for the metric image.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The path to the file.</returns>
    public string SaveMetricImage(MemoryStream memoryStream, string metricName)
    {
        var metricFileName = $"{metricName}_{DateTime.Now.Ticks}.png";
        using var sr = new StreamReader(memoryStream);
        // Writes the memory stream to a file.
        File.WriteAllBytes(metricFileName, memoryStream.ToArray());
        var filePath = Path.Join(AppDomain.CurrentDomain.BaseDirectory,
            metricFileName);
        return filePath;
    }

    /// <summary>
    /// Get data for CloudWatch metrics.
    /// </summary>
    /// <param name="minutesOfData">The number of minutes of data to include.</
param>
    /// <param name="useDescendingTime">True to return the data descending by
 time.</param>
    /// <param name="endDateUtc">The end date for the data, in UTC.</param>
    /// <param name="maxDataPoints">The maximum data points to include.</param>
    /// <param name="dataQueries">Optional data queries to include.</param>
    /// <returns>A list of the requested metric data.</returns>
    public async Task<List<MetricDataResult>> GetMetricData(int minutesOfData,
 bool useDescendingTime, DateTime? endDateUtc = null,
        int maxDataPoints = 0, List<MetricDataQuery>? dataQueries = null)
    {
        var metricData = new List<MetricDataResult>();
        // If no end time is provided, use the current time for the end time.
        endDateUtc ??= DateTime.UtcNow;
        var timeZoneOffset =
 TimeZoneInfo.Local.GetUtcOffset(endDateUtc.Value.ToLocalTime());
        var startTimeUtc = endDateUtc.Value.AddMinutes(-minutesOfData);
        // The timezone string should be in the format +0000, so use the timezone
 offset to format it correctly.
        var timeZoneString = $"{timeZoneOffset.Hours:D2}
{timeZoneOffset.Minutes:D2}";
        var paginatedMetricData = _amazonCloudWatch.Paginators.GetMetricData(
```

```
            new GetMetricDataRequest()
            {
                StartTimeUtc = startTimeUtc,
                EndTimeUtc = endDateUtc.Value,
                LabelOptions = new LabelOptions { Timezone = timeZoneString },
                ScanBy = useDescendingTime ? ScanBy.TimestampDescending :
ScanBy.TimestampAscending,
                MaxDatapoints = maxDataPoints,
                MetricDataQueries = dataQueries,
            });

        await foreach (var data in paginatedMetricData.MetricDataResults)
        {
            metricData.Add(data);
        }
        return metricData;
    }

    /// <summary>
    /// Add a metric alarm to send an email when the metric passes a threshold.
    /// </summary>
    /// <param name="alarmDescription">A description of the alarm.</param>
    /// <param name="alarmName">The name for the alarm.</param>
    /// <param name="comparison">The type of comparison to use.</param>
    /// <param name="metricName">The name of the metric for the alarm.</param>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="threshold">The threshold value for the alarm.</param>
    /// <param name="alarmActions">Optional actions to execute when in an alarm
state.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutMetricEmailAlarm(string alarmDescription, string
alarmName, ComparisonOperator comparison,
        string metricName, string metricNamespace, double threshold, List<string>
alarmActions = null!)
    {
        try
        {
            var putEmailAlarmResponse = await
_amazonCloudWatch.PutMetricAlarmAsync(
                new PutMetricAlarmRequest()
                {
                    AlarmActions = alarmActions,
                    AlarmDescription = alarmDescription,
                    AlarmName = alarmName,
```

```
                    ComparisonOperator = comparison,
                    Threshold = threshold,
                    Namespace = metricNamespace,
                    MetricName = metricName,
                    EvaluationPeriods = 1,
                    Period = 10,
                    Statistic = new Statistic("Maximum"),
                    DatapointsToAlarm = 1,
                    TreatMissingData = "ignore"
                });
            return putEmailAlarmResponse.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (LimitExceededException lex)
        {
            _logger.LogError(lex, $"Unable to add alarm {alarmName}. Alarm quota
 has already been reached.");
        }

        return false;
    }

    /// <summary>
    /// Add specific email actions to a list of action strings for a CloudWatch
 alarm.
    /// </summary>
    /// <param name="accountId">The AccountId for the alarm.</param>
    /// <param name="region">The region for the alarm.</param>
    /// <param name="emailTopicName">An Amazon Simple Notification Service (SNS)
 topic for the alarm email.</param>
    /// <param name="alarmActions">Optional list of existing alarm actions to
 append to.</param>
    /// <returns>A list of string actions for an alarm.</returns>
    public List<string> AddEmailAlarmAction(string accountId, string region,
        string emailTopicName, List<string>? alarmActions = null)
    {
        alarmActions ??= new List<string>();
        var snsAlarmAction = $"arn:aws:sns:{region}:{accountId}:
{emailTopicName}";
        alarmActions.Add(snsAlarmAction);
        return alarmActions;
    }

    /// <summary>
    /// Describe the current alarms, optionally filtered by state.
```

```
    /// </summary>
    /// <param name="stateValue">Optional filter for alarm state.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarms(StateValue? stateValue =
null)
    {
        List<MetricAlarm> alarms = new List<MetricAlarm>();
        var paginatedDescribeAlarms =
_amazonCloudWatch.Paginators.DescribeAlarms(
            new DescribeAlarmsRequest()
            {
                StateValue = stateValue
            });

        await foreach (var data in paginatedDescribeAlarms.MetricAlarms)
        {
            alarms.Add(data);
        }
        return alarms;
    }

    /// <summary>
    /// Describe the current alarms for a specific metric.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The name of the metric.</param>
    /// <returns>The list of alarm data.</returns>
    public async Task<List<MetricAlarm>> DescribeAlarmsForMetric(string
metricNamespace, string metricName)
    {
        var alarmsResult = await _amazonCloudWatch.DescribeAlarmsForMetricAsync(
            new DescribeAlarmsForMetricRequest()
            {
                Namespace = metricNamespace,
                MetricName = metricName
            });

        return alarmsResult.MetricAlarms;
    }

    /// <summary>
    /// Describe the history of an alarm for a number of days in the past.
    /// </summary>
    /// <param name="alarmName">The name of the alarm.</param>
```

```
    /// <param name="historyDays">The number of days in the past.</param>
    /// <returns>The list of alarm history data.</returns>
    public async Task<List<AlarmHistoryItem>> DescribeAlarmHistory(string
alarmName, int historyDays)
    {
        List<AlarmHistoryItem> alarmHistory = new List<AlarmHistoryItem>();
        var paginatedAlarmHistory =
_amazonCloudWatch.Paginators.DescribeAlarmHistory(
            new DescribeAlarmHistoryRequest()
            {
                AlarmName = alarmName,
                EndDateUtc = DateTime.UtcNow,
                HistoryItemType = HistoryItemType.StateUpdate,
                StartDateUtc = DateTime.UtcNow.AddDays(-historyDays)
            });

        await foreach (var data in paginatedAlarmHistory.AlarmHistoryItems)
        {
            alarmHistory.Add(data);
        }
        return alarmHistory;
    }

    /// <summary>
    /// Delete a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteAlarms(List<string> alarmNames)
    {
        var deleteAlarmsResult = await _amazonCloudWatch.DeleteAlarmsAsync(
            new DeleteAlarmsRequest()
            {
                AlarmNames = alarmNames
            });

        return deleteAlarmsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Disable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
```

```
    public async Task<bool> DisableAlarmActions(List<string> alarmNames)
    {
        var disableAlarmActionsResult = await
_amazonCloudWatch.DisableAlarmActionsAsync(
            new DisableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return disableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Enable the actions for a list of alarms from CloudWatch.
    /// </summary>
    /// <param name="alarmNames">A list of names of alarms.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableAlarmActions(List<string> alarmNames)
    {
        var enableAlarmActionsResult = await
_amazonCloudWatch.EnableAlarmActionsAsync(
            new EnableAlarmActionsRequest()
            {
                AlarmNames = alarmNames
            });

        return enableAlarmActionsResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Add an anomaly detector for a single metric.
    /// </summary>
    /// <param name="anomalyDetector">A single metric anomaly detector.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> PutAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var putAlarmDetectorResult = await
_amazonCloudWatch.PutAnomalyDetectorAsync(
            new PutAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });
```

```
        return putAlarmDetectorResult.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Describe anomaly detectors for a metric and namespace.
    /// </summary>
    /// <param name="metricNamespace">The namespace of the metric.</param>
    /// <param name="metricName">The metric of the anomaly detectors.</param>
    /// <returns>The list of detectors.</returns>
    public async Task<List<AnomalyDetector>> DescribeAnomalyDetectors(string
metricNamespace, string metricName)
    {
        List<AnomalyDetector> detectors = new List<AnomalyDetector>();
        var paginatedDescribeAnomalyDetectors =
_amazonCloudWatch.Paginators.DescribeAnomalyDetectors(
            new DescribeAnomalyDetectorsRequest()
            {
                MetricName = metricName,
                Namespace = metricNamespace
            });

        await foreach (var data in
paginatedDescribeAnomalyDetectors.AnomalyDetectors)
        {
            detectors.Add(data);
        }

        return detectors;
    }

    /// <summary>
    /// Delete a single metric anomaly detector.
    /// </summary>
    /// <param name="anomalyDetector">The anomaly detector to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteAnomalyDetector(SingleMetricAnomalyDetector
anomalyDetector)
    {
        var deleteAnomalyDetectorResponse = await
_amazonCloudWatch.DeleteAnomalyDetectorAsync(
            new DeleteAnomalyDetectorRequest()
            {
                SingleMetricAnomalyDetector = anomalyDetector
            });
```

```
        return deleteAnomalyDetectorResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete a list of CloudWatch dashboards.
    /// </summary>
    /// <param name="dashboardNames">List of dashboard names to delete.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteDashboards(List<string> dashboardNames)
    {
        var deleteDashboardsResponse = await
 _amazonCloudWatch.DeleteDashboardsAsync(
            new DeleteDashboardsRequest()
            {
                DashboardNames = dashboardNames
            });

        return deleteDashboardsResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for .NET API 參考》中的下列主題。
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage
  - ListMetrics
  - PutAnomalyDetector
  - PutDashboard

- PutMetricAlarm
- PutMetricData

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.AlarmHistoryItem;
import software.amazon.awssdk.services.cloudwatch.model.AlarmType;
import software.amazon.awssdk.services.cloudwatch.model.AnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import
 software.amazon.awssdk.services.cloudwatch.model.DashboardValidationMessage;
import software.amazon.awssdk.services.cloudwatch.model.Datapoint;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.DeleteAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.DeleteDashboardsRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricResponse;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
```

```
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageRequest;
import
 software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageResponse;
import software.amazon.awssdk.services.cloudwatch.model.HistoryItemType;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataQuery;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataResult;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.MetricStat;
import
 software.amazon.awssdk.services.cloudwatch.model.PutAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardResponse;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.ScanBy;
import
 software.amazon.awssdk.services.cloudwatch.model.SingleMetricAnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
import
 software.amazon.awssdk.services.cloudwatch.paginators.ListDashboardsIterable;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
```

```
import java.nio.file.Files;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To enable billing metrics and statistics for this example, make sure billing
 * alerts are enabled for your account:
 * https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics
 *
 * This Java code example performs the following tasks:
 *
 * 1. List available namespaces from Amazon CloudWatch.
 * 2. List available metrics within the selected Namespace.
 * 3. Get statistics for the selected metric over the last day.
 * 4. Get CloudWatch estimated billing for the last week.
 * 5. Create a new CloudWatch dashboard with metrics.
 * 6. List dashboards using a paginator.
 * 7. Create a new custom metric by adding data for it.
 * 8. Add the custom metric to the dashboard.
 * 9. Create an alarm for the custom metric.
 * 10. Describe current alarms.
 * 11. Get current data for the new custom metric.
 * 12. Push data into the custom metric to trigger the alarm.
 * 13. Check the alarm state using the action DescribeAlarmsForMetric.
 * 14. Get alarm history for the new alarm.
 * 15. Add an anomaly detector for the custom metric.
 * 16. Describe current anomaly detectors.
 * 17. Get a metric image for the custom metric.
```

```
 * 18. Clean up the Amazon CloudWatch resources.
 */
public class CloudWatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException {
        final String usage = """

                Usage:
                  <myDate> <costDateWeek> <dashboardName> <dashboardJson>
 <dashboardAdd> <settings> <metricImage> \s

                Where:
                  myDate - The start date to use to get metric statistics. (For
 example, 2023-01-11T18:35:24.00Z.)\s
                  costDateWeek - The start date to use to get AWS/Billinget
 statistics. (For example, 2023-01-11T18:35:24.00Z.)\s
                  dashboardName - The name of the dashboard to create.\s
                  dashboardJson - The location of a JSON file to use to create a
 dashboard. (See Readme file.)\s
                  dashboardAdd - The location of a JSON file to use to update a
 dashboard. (See Readme file.)\s
                  settings - The location of a JSON file from which various
 values are read. (See Readme file.)\s
                  metricImage - The location of a BMP file that is used to create
 a graph.\s
                """;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        Region region = Region.US_EAST_1;
        String myDate = args[0];
        String costDateWeek = args[1];
        String dashboardName = args[2];
        String dashboardJson = args[3];
        String dashboardAdd = args[4];
        String settings = args[5];
        String metricImage = args[6];

        Double dataPoint = Double.parseDouble("10.0");
```

```
        Scanner sc = new Scanner(System.in);
        CloudWatchClient cw = CloudWatchClient.builder()
                .region(region)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon CloudWatch example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "1. List at least five available unique namespaces from Amazon
CloudWatch. Select one from the list.");
        ArrayList<String> list = listNameSpaces(cw);
        for (int z = 0; z < 5; z++) {
            int index = z + 1;
            System.out.println("    " + index + ". " + list.get(z));
        }

        String selectedNamespace = "";
        String selectedMetrics = "";
        int num = Integer.parseInt(sc.nextLine());
        if (1 <= num && num <= 5) {
            selectedNamespace = list.get(num - 1);
        } else {
            System.out.println("You did not select a valid option.");
            System.exit(1);
        }
        System.out.println("You selected " + selectedNamespace);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. List available metrics within the selected
namespace and select one from the list.");
        ArrayList<String> metList = listMets(cw, selectedNamespace);
        for (int z = 0; z < 5; z++) {
            int index = z + 1;
            System.out.println("    " + index + ". " + metList.get(z));
        }
        num = Integer.parseInt(sc.nextLine());
        if (1 <= num && num <= 5) {
            selectedMetrics = metList.get(num - 1);
        } else {
```

```
            System.out.println("You did not select a valid option.");
            System.exit(1);
        }
        System.out.println("You selected " + selectedMetrics);
        Dimension myDimension = getSpecificMet(cw, selectedNamespace);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get statistics for the selected metric over the
last day.");
        String metricOption = "";
        ArrayList<String> statTypes = new ArrayList<>();
        statTypes.add("SampleCount");
        statTypes.add("Average");
        statTypes.add("Sum");
        statTypes.add("Minimum");
        statTypes.add("Maximum");

        for (int t = 0; t < 5; t++) {
            System.out.println("    " + (t + 1) + ". " + statTypes.get(t));
        }
        System.out.println("Select a metric statistic by entering a number from
the preceding list:");
        num = Integer.parseInt(sc.nextLine());
        if (1 <= num && num <= 5) {
            metricOption = statTypes.get(num - 1);
        } else {
            System.out.println("You did not select a valid option.");
            System.exit(1);
        }
        System.out.println("You selected " + metricOption);
        getAndDisplayMetricStatistics(cw, selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get CloudWatch estimated billing for the last
week.");
        getMetricStatistics(cw, costDateWeek);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create a new CloudWatch dashboard with metrics.");
        createDashboardWithMetrics(cw, dashboardName, dashboardJson);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. List dashboards using a paginator.");
        listDashboards(cw);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create a new custom metric by adding data to
it.");
        createNewCustomMetric(cw, dataPoint);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Add an additional metric to the dashboard.");
        addMetricToDashboard(cw, dashboardAdd, dashboardName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Create an alarm for the custom metric.");
        String alarmName = createAlarm(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Describe ten current alarms.");
        describeAlarms(cw);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Get current data for new custom metric.");
        getCustomMetricData(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Push data into the custom metric to trigger the
alarm.");
        addMetricDataForAlarm(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Check the alarm state using the action
DescribeAlarmsForMetric.");
        checkForMetricAlarm(cw, settings);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("14. Get alarm history for the new alarm.");
        getAlarmHistory(cw, settings, myDate);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Add an anomaly detector for the custom metric.");
        addAnomalyDetector(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Describe current anomaly detectors.");
        describeAnomalyDetectors(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Get a metric image for the custom metric.");
        getAndOpenMetricImage(cw, metricImage);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up the Amazon CloudWatch resources.");
        deleteDashboard(cw, dashboardName);
        deleteCWAlarm(cw, alarmName);
        deleteAnomalyDetector(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon CloudWatch example scenario is
complete.");
        System.out.println(DASHES);
        cw.close();
    }

    public static void deleteAnomalyDetector(CloudWatchClient cw, String
fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
```

```
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

            DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

            cw.deleteAnomalyDetector(request);
            System.out.println("Successfully deleted the Anomaly Detector.");

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
        try {
            DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
                    .alarmNames(alarmName)
                    .build();

            cw.deleteAlarms(request);
            System.out.println("Successfully deleted alarm " + alarmName);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
   public static void deleteDashboard(CloudWatchClient cw, String dashboardName)
{
       try {
           DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
                   .dashboardNames(dashboardName)
                   .build();
           cw.deleteDashboards(dashboardsRequest);
           System.out.println(dashboardName + " was successfully deleted.");

       } catch (CloudWatchException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   public static void getAndOpenMetricImage(CloudWatchClient cw, String
fileName) {
       System.out.println("Getting Image data for custom metric.");
       try {
           String myJSON = "{\n" +
                   "  \"title\": \"Example Metric Graph\",\n" +
                   "  \"view\": \"timeSeries\",\n" +
                   "  \"stacked \": false,\n" +
                   "  \"period\": 10,\n" +
                   "  \"width\": 1400,\n" +
                   "  \"height\": 600,\n" +
                   "  \"metrics\": [\n" +
                   "    [\n" +
                   "      \"AWS/Billing\",\n" +
                   "      \"EstimatedCharges\",\n" +
                   "      \"Currency\",\n" +
                   "      \"USD\"\n" +
                   "    ]\n" +
                   "  ]\n" +
                   "}";

           GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
                   .metricWidget(myJSON)
                   .build();

           GetMetricWidgetImageResponse response =
cw.getMetricWidgetImage(imageRequest);
```

```
            SdkBytes sdkBytes = response.metricWidgetImage();
            byte[] bytes = sdkBytes.asByteArray();
            File outputFile = new File(fileName);
            try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
                outputStream.write(bytes);
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();
            DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                    .maxResults(10)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
            List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
            for (AnomalyDetector detector : anomalyDetectorList) {
                System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
                System.out.println("State: " + detector.stateValue());
            }

        } catch (CloudWatchException | IOException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

            PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

            cw.putAnomalyDetector(anomalyDetectorRequest);
            System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getAlarmHistory(CloudWatchClient cw, String fileName,
String date) {
        try {
            // Read values from the JSON file.
```

```
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String alarmName = rootNode.findValue("exampleAlarmName").asText();

            Instant start = Instant.parse(date);
            Instant endDate = Instant.now();
            DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                    .startDate(start)
                    .endDate(endDate)
                    .alarmName(alarmName)
                    .historyItemType(HistoryItemType.ACTION)
                    .build();

            DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
            List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
            if (historyItems.isEmpty()) {
                System.out.println("No alarm history data found for " + alarmName
+ ".");
            } else {
                for (AlarmHistoryItem item : historyItems) {
                    System.out.println("History summary: " +
item.historySummary());
                    System.out.println("Time stamp: " + item.timestamp());
                }
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void checkForMetricAlarm(CloudWatchClient cw, String fileName)
{
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
```

```
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();
            boolean hasAlarm = false;
            int retries = 10;

            DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            while (!hasAlarm && retries > 0) {
                DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
                hasAlarm = response.hasMetricAlarms();
                retries--;
                Thread.sleep(20000);
                System.out.println(".");
            }
            if (!hasAlarm)
                System.out.println("No Alarm state found for " + customMetricName
+ " after 10 retries.");
            else
                System.out.println("Alarm state found for " + customMetricName +
".");

        } catch (CloudWatchException | IOException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void addMetricDataForAlarm(CloudWatchClient cw, String
fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
```

```
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            // Set an Instant object.
            String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
            Instant instant = Instant.parse(time);

            MetricDatum datum = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1001.00)
                    .timestamp(instant)
                    .build();

            MetricDatum datum2 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1002.00)
                    .timestamp(instant)
                    .build();

            List<MetricDatum> metricDataList = new ArrayList<>();
            metricDataList.add(datum);
            metricDataList.add(datum2);

            PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

            cw.putMetricData(request);
            System.out.println("Added metric values for for metric " +
customMetricName);

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getCustomMetricData(CloudWatchClient cw, String fileName)
{
        try {
```

```
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            // Set the date.
            Instant nowDate = Instant.now();

            long hours = 1;
            long minutes = 30;
            Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
                    ChronoUnit.MINUTES);

            Metric met = Metric.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

            MetricStat metStat = MetricStat.builder()
                    .stat("Maximum")
                    .period(1)
                    .metric(met)
                    .build();

            MetricDataQuery dataQUery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();

            List<MetricDataQuery> dq = new ArrayList<>();
            dq.add(dataQUery);

            GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(date2)
                    .metricDataQueries(dq)
```

```
                    .build();

            GetMetricDataResponse response = cw.getMetricData(getMetReq);
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                System.out.println("The label is " + item.label());
                System.out.println("The status code is " +
item.statusCode().toString());
            }

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void describeAlarms(CloudWatchClient cw) {
        try {
            List<AlarmType> typeList = new ArrayList<>();
            typeList.add(AlarmType.METRIC_ALARM);

            DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
                    .alarmTypes(typeList)
                    .maxRecords(10)
                    .build();

            DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
            List<MetricAlarm> alarmList = response.metricAlarms();
            for (MetricAlarm alarm : alarmList) {
                System.out.println("Alarm name: " + alarm.alarmName());
                System.out.println("Alarm description: " +
alarm.alarmDescription());
            }
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createAlarm(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new
File(fileName));
```

```
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();
            String alarmName = rootNode.findValue("exampleAlarmName").asText();
            String emailTopic = rootNode.findValue("emailTopic").asText();
            String accountId = rootNode.findValue("accountId").asText();
            String region = rootNode.findValue("region").asText();

            // Create a List for alarm actions.
            List<String> alarmActions = new ArrayList<>();
            alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
            PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
                    .alarmActions(alarmActions)
                    .alarmDescription("Example metric alarm")
                    .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
                    .threshold(100.00)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .evaluationPeriods(1)
                    .period(10)
                    .statistic("Maximum")
                    .datapointsToAlarm(1)
                    .treatMissingData("ignore")
                    .build();

            cw.putMetricAlarm(alarmRequest);
            System.out.println(alarmName + " was successfully created!");
            return alarmName;

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static void addMetricToDashboard(CloudWatchClient cw, String fileName,
String dashboardName) {
```

```
        try {
            PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
                    .dashboardName(dashboardName)
                    .dashboardBody(readFileAsString(fileName))
                    .build();

            cw.putDashboard(dashboardRequest);
            System.out.println(dashboardName + " was successfully updated.");

        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void createNewCustomMetric(CloudWatchClient cw, Double
dataPoint) {
        try {
            Dimension dimension = Dimension.builder()
                    .name("UNIQUE_PAGES")
                    .value("URLS")
                    .build();

            // Set an Instant object.
            String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
            Instant instant = Instant.parse(time);

            MetricDatum datum = MetricDatum.builder()
                    .metricName("PAGES_VISITED")
                    .unit(StandardUnit.NONE)
                    .value(dataPoint)
                    .timestamp(instant)
                    .dimensions(dimension)
                    .build();

            PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace("SITE/TRAFFIC")
                    .metricData(datum)
                    .build();

            cw.putMetricData(request);
            System.out.println("Added metric values for for metric
PAGES_VISITED");
```

```
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listDashboards(CloudWatchClient cw) {
        try {
            ListDashboardsIterable listRes = cw.listDashboardsPaginator();
            listRes.stream()
                    .flatMap(r -> r.dashboardEntries().stream())
                    .forEach(entry -> {
                        System.out.println("Dashboard name is: " +
entry.dashboardName());
                        System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
                    });

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
        try {
            PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
                    .dashboardName(dashboardName)
                    .dashboardBody(readFileAsString(fileName))
                    .build();

            PutDashboardResponse response = cw.putDashboard(dashboardRequest);
            System.out.println(dashboardName + " was successfully created.");
            List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
            if (messages.isEmpty()) {
                System.out.println("There are no messages in the new Dashboard");
            } else {
                for (DashboardValidationMessage message : messages) {
                    System.out.println("Message is: " + message.message());
                }
            }
```

```
        } catch (CloudWatchException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String readFileAsString(String file) throws IOException {
        return new String(Files.readAllBytes(Paths.get(file)));
    }

    public static void getMetricStatistics(CloudWatchClient cw, String
costDateWeek) {
        try {
            Instant start = Instant.parse(costDateWeek);
            Instant endDate = Instant.now();
            Dimension dimension = Dimension.builder()
                    .name("Currency")
                    .value("USD")
                    .build();

            List<Dimension> dimensionList = new ArrayList<>();
            dimensionList.add(dimension);
            GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
                    .metricName("EstimatedCharges")
                    .namespace("AWS/Billing")
                    .dimensions(dimensionList)
                    .statistics(Statistic.MAXIMUM)
                    .startTime(start)
                    .endTime(endDate)
                    .period(86400)
                    .build();

            GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
            List<Datapoint> data = response.datapoints();
            if (!data.isEmpty()) {
                for (Datapoint datapoint : data) {
                    System.out
                            .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
                }
            } else {
```

```
                System.out.println("The returned data list is empty");
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
            String metricOption, String date, Dimension myDimension) {
        try {
            Instant start = Instant.parse(date);
            Instant endDate = Instant.now();

            GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
                    .endTime(endDate)
                    .startTime(start)
                    .dimensions(myDimension)
                    .metricName(metVal)
                    .namespace(nameSpace)
                    .period(86400)
                    .statistics(Statistic.fromValue(metricOption))
                    .build();

            GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
            List<Datapoint> data = response.datapoints();
            if (!data.isEmpty()) {
                for (Datapoint datapoint : data) {
                    System.out
                            .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
                }
            } else {
                System.out.println("The returned data list is empty");
            }

        } catch (CloudWatchException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
    }

    public static Dimension getSpecificMet(CloudWatchClient cw, String namespace)
{
        try {
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

            ListMetricsResponse response = cw.listMetrics(request);
            List<Metric> myList = response.metrics();
            Metric metric = myList.get(0);
            return metric.dimensions().get(0);

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static ArrayList<String> listMets(CloudWatchClient cw, String
namespace) {
        try {
            ArrayList<String> metList = new ArrayList<>();
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                    .flatMap(r -> r.metrics().stream())
                    .forEach(metrics -> metList.add(metrics.metricName()));

            return metList;

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static ArrayList<String> listNameSpaces(CloudWatchClient cw) {
```

```
        try {
            ArrayList<String> nameSpaceList = new ArrayList<>();
            ListMetricsRequest request = ListMetricsRequest.builder()
                    .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                    .flatMap(r -> r.metrics().stream())
                    .forEach(metrics -> {
                        String data = metrics.namespace();
                        if (!nameSpaceList.contains(data)) {
                            nameSpaceList.add(data);
                        }
                    });

            return nameSpaceList;
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的下列主題。
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage
  - ListMetrics
  - PutAnomalyDetector

- PutDashboard
- PutMetricAlarm
- PutMetricData

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 To enable billing metrics and statistics for this example, make sure billing
 alerts are enabled for your account:
 https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics

 This Kotlin code example performs the following tasks:

 1. List available namespaces from Amazon CloudWatch. Select a namespace from the
 list.
 2. List available metrics within the selected namespace.
 3. Get statistics for the selected metric over the last day.
 4. Get CloudWatch estimated billing for the last week.
 5. Create a new CloudWatch dashboard with metrics.
 6. List dashboards using a paginator.
 7. Create a new custom metric by adding data for it.
 8. Add the custom metric to the dashboard.
 9. Create an alarm for the custom metric.
 10. Describe current alarms.
 11. Get current data for the new custom metric.
 12. Push data into the custom metric to trigger the alarm.
```

```
 13. Check the alarm state using the action DescribeAlarmsForMetric.
 14. Get alarm history for the new alarm.
 15. Add an anomaly detector for the custom metric.
 16. Describe current anomaly detectors.
 17. Get a metric image for the custom metric.
 18. Clean up the Amazon CloudWatch resources.
 */

val DASHES: String? = String(CharArray(80)).replace("\u0000", "-")
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <myDate> <costDateWeek> <dashboardName> <dashboardJson>
 <dashboardAdd> <settings> <metricImage>

        Where:
            myDate - The start date to use to get metric statistics. (For
 example, 2023-01-11T18:35:24.00Z.)
            costDateWeek - The start date to use to get AWS Billing and Cost
 Management statistics. (For example, 2023-01-11T18:35:24.00Z.)
            dashboardName - The name of the dashboard to create.
            dashboardJson - The location of a JSON file to use to create a
 dashboard. (See Readme file.)
            dashboardAdd - The location of a JSON file to use to update a
 dashboard. (See Readme file.)
            settings - The location of a JSON file from which various values are
 read. (See Readme file.)
            metricImage - The location of a BMP file that is used to create a
 graph.
    """

    if (args.size != 7) {
        println(usage)
        System.exit(1)
    }

    val myDate = args[0]
    val costDateWeek = args[1]
    val dashboardName = args[2]
    val dashboardJson = args[3]
    val dashboardAdd = args[4]
    val settings = args[5]
    var metricImage = args[6]
    val dataPoint = "10.0".toDouble()
```

```
    val inOb = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the Amazon CloudWatch example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. List at least five available unique namespaces from Amazon
CloudWatch. Select a CloudWatch namespace from the list.")
    val list: ArrayList<String> = listNameSpaces()
    for (z in 0..4) {
        println("    ${z + 1}. ${list[z]}")
    }

    var selectedNamespace: String
    var selectedMetrics = ""
    var num = inOb.nextLine().toInt()
    println("You selected $num")

    if (1 <= num && num <= 5) {
        selectedNamespace = list[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $selectedNamespace")
    println(DASHES)

    println(DASHES)
    println("2. List available metrics within the selected namespace and select
one from the list.")
    val metList = listMets(selectedNamespace)
    for (z in 0..4) {
        println("    ${ z + 1}. ${metList?.get(z)}")
    }
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        selectedMetrics = metList!![num - 1]
    } else {
        println("You did not select a valid option.")
        System.exit(1)
    }
    println("You selected $selectedMetrics")
    val myDimension = getSpecificMet(selectedNamespace)
```

```
    if (myDimension == null) {
        println("Error - Dimension is null")
        exitProcess(1)
    }
    println(DASHES)

    println(DASHES)
    println("3. Get statistics for the selected metric over the last day.")
    val metricOption: String
    val statTypes = ArrayList<String>()
    statTypes.add("SampleCount")
    statTypes.add("Average")
    statTypes.add("Sum")
    statTypes.add("Minimum")
    statTypes.add("Maximum")

    for (t in 0..4) {
        println("    ${t + 1}. ${statTypes[t]}")
    }
    println("Select a metric statistic by entering a number from the preceding
list:")
    num = inOb.nextLine().toInt()
    if (1 <= num && num <= 5) {
        metricOption = statTypes[num - 1]
    } else {
        println("You did not select a valid option.")
        exitProcess(1)
    }
    println("You selected $metricOption")
    getAndDisplayMetricStatistics(selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension)
    println(DASHES)

    println(DASHES)
    println("4. Get CloudWatch estimated billing for the last week.")
    getMetricStatistics(costDateWeek)
    println(DASHES)

    println(DASHES)
    println("5. Create a new CloudWatch dashboard with metrics.")
    createDashboardWithMetrics(dashboardName, dashboardJson)
    println(DASHES)

    println(DASHES)
```

```
    println("6. List dashboards using a paginator.")
    listDashboards()
    println(DASHES)

    println(DASHES)
    println("7. Create a new custom metric by adding data to it.")
    createNewCustomMetric(dataPoint)
    println(DASHES)

    println(DASHES)
    println("8. Add an additional metric to the dashboard.")
    addMetricToDashboard(dashboardAdd, dashboardName)
    println(DASHES)

    println(DASHES)
    println("9. Create an alarm for the custom metric.")
    val alarmName: String = createAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("10. Describe 10 current alarms.")
    describeAlarms()
    println(DASHES)

    println(DASHES)
    println("11. Get current data for the new custom metric.")
    getCustomMetricData(settings)
    println(DASHES)

    println(DASHES)
    println("12. Push data into the custom metric to trigger the alarm.")
    addMetricDataForAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("13. Check the alarm state using the action
DescribeAlarmsForMetric.")
    checkForMetricAlarm(settings)
    println(DASHES)

    println(DASHES)
    println("14. Get alarm history for the new alarm.")
    getAlarmHistory(settings, myDate)
    println(DASHES)
```

```
    println(DASHES)
    println("15. Add an anomaly detector for the custom metric.")
    addAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("16. Describe current anomaly detectors.")
    describeAnomalyDetectors(settings)
    println(DASHES)

    println(DASHES)
    println("17. Get a metric image for the custom metric.")
    getAndOpenMetricImage(metricImage)
    println(DASHES)

    println(DASHES)
    println("18. Clean up the Amazon CloudWatch resources.")
    deleteDashboard(dashboardName)
    deleteAlarm(alarmName)
    deleteAnomalyDetector(settings)
    println(DASHES)

    println(DASHES)
    println("The Amazon CloudWatch example scenario is complete.")
    println(DASHES)
}

suspend fun deleteAnomalyDetector(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal = SingleMetricAnomalyDetector {
        metricName = customMetricName
        namespace = customMetricNamespace
        stat = "Maximum"
    }

    val request = DeleteAnomalyDetectorRequest {
        singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
```

```
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAnomalyDetector(request)
        println("Successfully deleted the Anomaly Detector.")
    }
}

suspend fun deleteAlarm(alarmNameVal: String) {
    val request = DeleteAlarmsRequest {
        alarmNames = listOf(alarmNameVal)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteAlarms(request)
        println("Successfully deleted alarm $alarmNameVal")
    }
}

suspend fun deleteDashboard(dashboardName: String) {
    val dashboardsRequest = DeleteDashboardsRequest {
        dashboardNames = listOf(dashboardName)
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.deleteDashboards(dashboardsRequest)
        println("$dashboardName was successfully deleted.")
    }
}

suspend fun getAndOpenMetricImage(fileName: String) {
    println("Getting Image data for custom metric.")
    val myJSON = """{
        "title": "Example Metric Graph",
        "view": "timeSeries",
        "stacked ": false,
        "period": 10,
        "width": 1400,
        "height": 600,
        "metrics": [
            [
            "AWS/Billing",
            "EstimatedCharges",
            "Currency",
            "USD"
```

```
            ]
        ]
        }"""

    val imageRequest = GetMetricWidgetImageRequest {
        metricWidget = myJSON
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricWidgetImage(imageRequest)
        val bytes = response.metricWidgetImage
        if (bytes != null) {
            File(fileName).writeBytes(bytes)
        }
    }
    println("You have successfully written data to $fileName")
}

suspend fun describeAnomalyDetectors(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val detectorsRequest = DescribeAnomalyDetectorsRequest {
        maxResults = 10
        metricName = customMetricName
        namespace = customMetricNamespace
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAnomalyDetectors(detectorsRequest)
        response.anomalyDetectors?.forEach { detector ->
            println("Metric name:
 ${detector.singleMetricAnomalyDetector?.metricName}")
            println("State: ${detector.stateValue}")
        }
    }
}

suspend fun addAnomalyDetector(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
```

```
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    val singleMetricAnomalyDetectorVal = SingleMetricAnomalyDetector {
        metricName = customMetricName
        namespace = customMetricNamespace
        stat = "Maximum"
    }

    val anomalyDetectorRequest = PutAnomalyDetectorRequest {
        singleMetricAnomalyDetector = singleMetricAnomalyDetectorVal
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putAnomalyDetector(anomalyDetectorRequest)
        println("Added anomaly detector for metric $customMetricName.")
    }
}

suspend fun getAlarmHistory(fileName: String, date: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val start = Instant.parse(date)
    val endDateVal = Instant.now()

    val historyRequest = DescribeAlarmHistoryRequest {
        startDate = aws.smithy.kotlin.runtime.time.Instant(start)
        endDate = aws.smithy.kotlin.runtime.time.Instant(endDateVal)
        alarmName = alarmNameVal
        historyItemType = HistoryItemType.Action
    }

    CloudWatchClient { credentialsProvider = EnvironmentCredentialsProvider();
 region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarmHistory(historyRequest)
        val historyItems = response.alarmHistoryItems
        if (historyItems != null) {
            if (historyItems.isEmpty()) {
                println("No alarm history data found for $alarmNameVal.")
            } else {
```

```
                for (item in historyItems) {
                    println("History summary ${item.historySummary}")
                    println("Time stamp: ${item.timestamp}")
                }
            }
        }
    }
}

suspend fun checkForMetricAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    var hasAlarm = false
    var retries = 10

    val metricRequest = DescribeAlarmsForMetricRequest {
        metricName = customMetricName
        namespace = customMetricNamespace
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        while (!hasAlarm && retries > 0) {
            val response = cwClient.describeAlarmsForMetric(metricRequest)
            if (response.metricAlarms?.count()!! > 0) {
                hasAlarm = true
            }
            retries--
            delay(20000)
            println(".")
        }
        if (!hasAlarm) println("No Alarm state found for $customMetricName after
 10 retries.") else println("Alarm state found for $customMetricName.")
    }
}

suspend fun addMetricDataForAlarm(fileName: String?) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
```

```
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum = MetricDatum {
        metricName = customMetricName
        unit = StandardUnit.None
        value = 1001.00
        timestamp = aws.smithy.kotlin.runtime.time.Instant(instant)
    }

    val datum2 = MetricDatum {
        metricName = customMetricName
        unit = StandardUnit.None
        value = 1002.00
        timestamp = aws.smithy.kotlin.runtime.time.Instant(instant)
    }

    val metricDataList = ArrayList<MetricDatum>()
    metricDataList.add(datum)
    metricDataList.add(datum2)

    val request = PutMetricDataRequest {
        namespace = customMetricNamespace
        metricData = metricDataList
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric $customMetricName")
    }
}

suspend fun getCustomMetricData(fileName: String) {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()

    // Set the date.
```

```
    val nowDate = Instant.now()
    val hours: Long = 1
    val minutes: Long = 30
    val date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(
        minutes,
        ChronoUnit.MINUTES
    )

    val met = Metric {
        metricName = customMetricName
        namespace = customMetricNamespace
    }

    val metStat = MetricStat {
        stat = "Maximum"
        period = 1
        metric = met
    }

    val dataQUery = MetricDataQuery {
        metricStat = metStat
        id = "foo2"
        returnData = true
    }

    val dq = ArrayList<MetricDataQuery>()
    dq.add(dataQUery)
    val getMetReq = GetMetricDataRequest {
        maxDatapoints = 10
        scanBy = ScanBy.TimestampDescending
        startTime = aws.smithy.kotlin.runtime.time.Instant(nowDate)
        endTime = aws.smithy.kotlin.runtime.time.Instant(date2)
        metricDataQueries = dq
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricData(getMetReq)
        response.metricDataResults?.forEach { item ->
            println("The label is ${item.label}")
            println("The status code is ${item.statusCode}")
        }
    }
}
```

```
suspend fun describeAlarms() {
    val typeList = ArrayList<AlarmType>()
    typeList.add(AlarmType.MetricAlarm)
    val alarmsRequest = DescribeAlarmsRequest {
        alarmTypes = typeList
        maxRecords = 10
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.describeAlarms(alarmsRequest)
        response.metricAlarms?.forEach { alarm ->
            println("Alarm name: ${alarm.alarmName}")
            println("Alarm description: ${alarm.alarmDescription}")
        }
    }
}

suspend fun createAlarm(fileName: String): String {
    // Read values from the JSON file.
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode: JsonNode = ObjectMapper().readTree(parser)
    val customMetricNamespace =
 rootNode.findValue("customMetricNamespace").asText()
    val customMetricName = rootNode.findValue("customMetricName").asText()
    val alarmNameVal = rootNode.findValue("exampleAlarmName").asText()
    val emailTopic = rootNode.findValue("emailTopic").asText()
    val accountId = rootNode.findValue("accountId").asText()
    val region2 = rootNode.findValue("region").asText()

    // Create a List for alarm actions.
    val alarmActionObs: MutableList<String> = ArrayList()
    alarmActionObs.add("arn:aws:sns:$region2:$accountId:$emailTopic")
    val alarmRequest = PutMetricAlarmRequest {
        alarmActions = alarmActionObs
        alarmDescription = "Example metric alarm"
        alarmName = alarmNameVal
        comparisonOperator = ComparisonOperator.GreaterThanOrEqualToThreshold
        threshold = 100.00
        metricName = customMetricName
        namespace = customMetricNamespace
        evaluationPeriods = 1
        period = 10
        statistic = Statistic.Maximum
        datapointsToAlarm = 1
```

```
        treatMissingData = "ignore"
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricAlarm(alarmRequest)
        println("$alarmNameVal was successfully created!")
        return alarmNameVal
    }
}

suspend fun addMetricToDashboard(fileNameVal: String, dashboardNameVal: String) {
    val dashboardRequest = PutDashboardRequest {
        dashboardName = dashboardNameVal
        dashboardBody = readFileAsString(fileNameVal)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully updated.")
    }
}

suspend fun createNewCustomMetric(dataPoint: Double) {
    val dimension = Dimension {
        name = "UNIQUE_PAGES"
        value = "URLS"
    }

    // Set an Instant object.
    val time =
 ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT)
    val instant = Instant.parse(time)
    val datum = MetricDatum {
        metricName = "PAGES_VISITED"
        unit = StandardUnit.None
        value = dataPoint
        timestamp = aws.smithy.kotlin.runtime.time.Instant(instant)
        dimensions = listOf(dimension)
    }

    val request = PutMetricDataRequest {
        namespace = "SITE/TRAFFIC"
        metricData = listOf(datum)
    }
```

```
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.putMetricData(request)
        println("Added metric values for for metric PAGES_VISITED")
    }
}

suspend fun listDashboards() {
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        cwClient.listDashboardsPaginated({})
            .transform { it.dashboardEntries?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name is ${obj.dashboardName}")
                println("Dashboard ARN is ${obj.dashboardArn}")
            }
    }
}

suspend fun createDashboardWithMetrics(dashboardNameVal: String, fileNameVal:
 String) {
    val dashboardRequest = PutDashboardRequest {
        dashboardName = dashboardNameVal
        dashboardBody = readFileAsString(fileNameVal)
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.putDashboard(dashboardRequest)
        println("$dashboardNameVal was successfully created.")
        val messages = response.dashboardValidationMessages
        if (messages != null) {
            if (messages.isEmpty()) {
                println("There are no messages in the new Dashboard")
            } else {
                for (message in messages) {
                    println("Message is: ${message.message}")
                }
            }
        }
    }
}

fun readFileAsString(file: String): String {
    return String(Files.readAllBytes(Paths.get(file)))
}
```

```
suspend fun getMetricStatistics(costDateWeek: String?) {
    val start = Instant.parse(costDateWeek)
    val endDate = Instant.now()
    val dimension = Dimension {
        name = "Currency"
        value = "USD"
    }

    val dimensionList: MutableList<Dimension> = ArrayList()
    dimensionList.add(dimension)

    val statisticsRequest = GetMetricStatisticsRequest {
        metricName = "EstimatedCharges"
        namespace = "AWS/Billing"
        dimensions = dimensionList
        statistics = listOf(Statistic.Maximum)
        startTime = aws.smithy.kotlin.runtime.time.Instant(start)
        endTime = aws.smithy.kotlin.runtime.time.Instant(endDate)
        period = 86400
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data: List<Datapoint>? = response.datapoints
        if (data != null) {
            if (!data.isEmpty()) {
                for (datapoint in data) {
                    println("Timestamp:  ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}

suspend fun getAndDisplayMetricStatistics(nameSpaceVal: String, metVal: String,
 metricOption: String, date: String, myDimension: Dimension) {
    val start = Instant.parse(date)
    val endDate = Instant.now()
    val statisticsRequest = GetMetricStatisticsRequest {
        endTime = aws.smithy.kotlin.runtime.time.Instant(endDate)
        startTime = aws.smithy.kotlin.runtime.time.Instant(start)
```

```
        dimensions = listOf(myDimension)
        metricName = metVal
        namespace = nameSpaceVal
        period = 86400
        statistics = listOf(Statistic.fromValue(metricOption))
    }

    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.getMetricStatistics(statisticsRequest)
        val data = response.datapoints
        if (data != null) {
            if (data.isNotEmpty()) {
                for (datapoint in data) {
                    println("Timestamp: ${datapoint.timestamp} Maximum value:
 ${datapoint.maximum}")
                }
            } else {
                println("The returned data list is empty")
            }
        }
    }
}

suspend fun listMets(namespaceVal: String?): ArrayList<String>? {
    val metList = ArrayList<String>()
    val request = ListMetricsRequest {
        namespace = namespaceVal
    }
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val reponse = cwClient.listMetrics(request)
        reponse.metrics?.forEach { metrics ->
            val data = metrics.metricName
            if (!metList.contains(data)) {
                metList.add(data!!)
            }
        }
    }
    return metList
}

suspend fun getSpecificMet(namespaceVal: String?): Dimension? {
    val request = ListMetricsRequest {
        namespace = namespaceVal
    }
```

```
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(request)
        val myList = response.metrics
        if (myList != null) {
            return myList[0].dimensions?.get(0)
        }
    }
    return null
}

suspend fun listNameSpaces(): ArrayList<String> {
    val nameSpaceList = ArrayList<String>()
    CloudWatchClient { region = "us-east-1" }.use { cwClient ->
        val response = cwClient.listMetrics(ListMetricsRequest {})
        response.metrics?.forEach { metrics ->
            val data = metrics.namespace
            if (!nameSpaceList.contains(data)) {
                nameSpaceList.add(data!!)
            }
        }
    }
    return nameSpaceList
}
```

- 如需 API 詳細資訊，請參閱《AWS 適用於 Kotlin 的 SDK API 參考》中的下列主題。
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage
  - ListMetrics
  - PutAnomalyDetector

- PutDashboard
- PutMetricAlarm
- PutMetricData

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDK 管理 CloudWatch 指標和警示

以下程式碼範例顯示做法：

- 建立警示以觀看指 CloudWatch 標。
- 將資料放入指標並觸發警示。
- 從警示中取得資料。
- 刪除警示。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

創建一個包裝 CloudWatch 操作的類。

```
from datetime import datetime, timedelta
import logging
from pprint import pprint
import random
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
class CloudWatchWrapper:
    """Encapsulates Amazon CloudWatch functions."""

    def __init__(self, cloudwatch_resource):
        """
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.cloudwatch_resource = cloudwatch_resource


    def put_metric_data_set(self, namespace, name, timestamp, unit, data_set):
        """
        Sends a set of data to CloudWatch for a metric. All of the data in the
 set
        have the same timestamp and unit.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param timestamp: The UTC timestamp for the metric.
        :param unit: The unit of the metric.
        :param data_set: The set of data to send. This set is a dictionary that
                         contains a list of values and a list of corresponding
 counts.
                         The value and count lists must be the same length.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[
                    {
                        "MetricName": name,
                        "Timestamp": timestamp,
                        "Values": data_set["values"],
                        "Counts": data_set["counts"],
                        "Unit": unit,
                    }
                ],
            )
            logger.info("Put data set for metric %s.%s.", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data set for metric %s.%s.",
 namespace, name)
```

```
            raise


    def create_metric_alarm(
        self,
        metric_namespace,
        metric_name,
        alarm_name,
        stat_type,
        period,
        eval_periods,
        threshold,
        comparison_op,
    ):
        """
        Creates an alarm that watches a metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :param alarm_name: The name of the alarm.
        :param stat_type: The type of statistic the alarm watches.
        :param period: The period in which metric data are grouped to calculate
                       statistics.
        :param eval_periods: The number of periods that the metric must be over
the
                             alarm threshold before the alarm is set into an
alarmed
                             state.
        :param threshold: The threshold value to compare against the metric
statistic.
        :param comparison_op: The comparison operation used to compare the
threshold
                              against the metric.
        :return: The newly created alarm.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace,
metric_name)
            alarm = metric.put_alarm(
                AlarmName=alarm_name,
                Statistic=stat_type,
                Period=period,
                EvaluationPeriods=eval_periods,
                Threshold=threshold,
```

```
                ComparisonOperator=comparison_op,
            )
            logger.info(
                "Added alarm %s to track metric %s.%s.",
                alarm_name,
                metric_namespace,
                metric_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't add alarm %s to metric %s.%s",
                alarm_name,
                metric_namespace,
                metric_name,
            )
            raise
        else:
            return alarm


    def put_metric_data(self, namespace, name, value, unit):
        """
        Sends a single data value to CloudWatch for a metric. This metric is
given
        a timestamp of the current UTC time.

        :param namespace: The namespace of the metric.
        :param name: The name of the metric.
        :param value: The value of the metric.
        :param unit: The unit of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(namespace, name)
            metric.put_data(
                Namespace=namespace,
                MetricData=[{"MetricName": name, "Value": value, "Unit": unit}],
            )
            logger.info("Put data for metric %s.%s", namespace, name)
        except ClientError:
            logger.exception("Couldn't put data for metric %s.%s", namespace,
name)
            raise
```

```
   def get_metric_statistics(self, namespace, name, start, end, period,
stat_types):
       """
       Gets statistics for a metric within a specified time span. Metrics are
grouped
       into the specified period.

       :param namespace: The namespace of the metric.
       :param name: The name of the metric.
       :param start: The UTC start time of the time span to retrieve.
       :param end: The UTC end time of the time span to retrieve.
       :param period: The period, in seconds, in which to group metrics. The
period
                      must match the granularity of the metric, which depends on
                      the metric's age. For example, metrics that are older than
                      three hours have a one-minute granularity, so the period
must
                      be at least 60 and must be a multiple of 60.
       :param stat_types: The type of statistics to retrieve, such as average
value
                          or maximum value.
       :return: The retrieved statistics for the metric.
       """
       try:
           metric = self.cloudwatch_resource.Metric(namespace, name)
           stats = metric.get_statistics(
               StartTime=start, EndTime=end, Period=period,
Statistics=stat_types
           )
           logger.info(
               "Got %s statistics for %s.", len(stats["Datapoints"]),
stats["Label"]
           )
       except ClientError:
           logger.exception("Couldn't get statistics for %s.%s.", namespace,
name)
           raise
       else:
           return stats


   def get_metric_alarms(self, metric_namespace, metric_name):
       """
       Gets the alarms that are currently watching the specified metric.
```

```
        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        :returns: An iterator that yields the alarms.
        """
        metric = self.cloudwatch_resource.Metric(metric_namespace, metric_name)
        alarm_iter = metric.alarms.all()
        logger.info("Got alarms for metric %s.%s.", metric_namespace,
metric_name)
        return alarm_iter


    def delete_metric_alarms(self, metric_namespace, metric_name):
        """
        Deletes all of the alarms that are currently watching the specified
metric.

        :param metric_namespace: The namespace of the metric.
        :param metric_name: The name of the metric.
        """
        try:
            metric = self.cloudwatch_resource.Metric(metric_namespace,
metric_name)
            metric.alarms.delete()
            logger.info(
                "Deleted alarms for metric %s.%s.", metric_namespace, metric_name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete alarms for metric %s.%s.",
                metric_namespace,
                metric_name,
            )
            raise
```

使用包裝函式類別將資料放入指標中、觸發觀察指標的警示，以及從警示取得資料。

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon CloudWatch metrics and alarms demo!")
```

```
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    cw_wrapper = CloudWatchWrapper(boto3.resource("cloudwatch"))

    minutes = 20
    metric_namespace = "doc-example-metric"
    metric_name = "page_views"
    start = datetime.utcnow() - timedelta(minutes=minutes)
    print(
        f"Putting data into metric {metric_namespace}.{metric_name} spanning the
"
        f"last {minutes} minutes."
    )
    for offset in range(0, minutes):
        stamp = start + timedelta(minutes=offset)
        cw_wrapper.put_metric_data_set(
            metric_namespace,
            metric_name,
            stamp,
            "Count",
            {
                "values": [
                    random.randint(bound, bound * 2)
                    for bound in range(offset + 1, offset + 11)
                ],
                "counts": [random.randint(1, offset + 1) for _ in range(10)],
            },
        )

    alarm_name = "high_page_views"
    period = 60
    eval_periods = 2
    print(f"Creating alarm {alarm_name} for metric {metric_name}.")
    alarm = cw_wrapper.create_metric_alarm(
        metric_namespace,
        metric_name,
        alarm_name,
        "Maximum",
        period,
        eval_periods,
        100,
        "GreaterThanThreshold",
```

```
    )
    print(f"Alarm ARN is {alarm.alarm_arn}.")
    print(f"Current alarm state is: {alarm.state_value}.")

    print(
        f"Sending data to trigger the alarm. This requires data over the
threshold "
        f"for {eval_periods} periods of {period} seconds each."
    )
    while alarm.state_value == "INSUFFICIENT_DATA":
        print("Sending data for the metric.")
        cw_wrapper.put_metric_data(
            metric_namespace, metric_name, random.randint(100, 200), "Count"
        )
        alarm.load()
        print(f"Current alarm state is: {alarm.state_value}.")
        if alarm.state_value == "INSUFFICIENT_DATA":
            print(f"Waiting for {period} seconds...")
            time.sleep(period)
        else:
            print("Wait for a minute for eventual consistency of metric data.")
            time.sleep(period)
            if alarm.state_value == "OK":
                alarm.load()
                print(f"Current alarm state is: {alarm.state_value}.")

    print(
        f"Getting data for metric {metric_namespace}.{metric_name} during
timespan "
        f"of {start} to {datetime.utcnow()} (times are UTC)."
    )
    stats = cw_wrapper.get_metric_statistics(
        metric_namespace,
        metric_name,
        start,
        datetime.utcnow(),
        60,
        ["Average", "Minimum", "Maximum"],
    )
    print(
        f"Got {len(stats['Datapoints'])} data points for metric "
        f"{metric_namespace}.{metric_name}."
    )
    pprint(sorted(stats["Datapoints"], key=lambda x: x["Timestamp"]))
```

```
    print(f"Getting alarms for metric {metric_name}.")
    alarms = cw_wrapper.get_metric_alarms(metric_namespace, metric_name)
    for alarm in alarms:
        print(f"Alarm {alarm.name} is currently in state {alarm.state_value}.")

    print(f"Deleting alarms for metric {metric_name}.")
    cw_wrapper.delete_metric_alarms(metric_namespace, metric_name)

    print("Thanks for watching!")
    print("-" * 88)
```

- 如需 API 的詳細資訊，請參閱《適用於 Python (Boto3) 的AWS SDK API 參考資料》中的下列主題。
  - DeleteAlarms
  - DescribeAlarmsForMetric
  - DisableAlarmActions
  - EnableAlarmActions
  - GetMetricStatistics
  - ListMetrics
  - PutMetricAlarm
  - PutMetricData

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 SDK 的跨服務 CloudWatch 範 AWS 例

下列範例應用程式使用 AWS SDK CloudWatch 與其他 AWS 服務應用程式結合使用。每個範例都包含一個連結 GitHub，您可以在其中找到如何設定和執行應用程式的指示。

範例

- 使用開發套件監控 Amazon DynamoDB 的效能 AWS

使用開發套件監控 Amazon DynamoDB 的效能 AWS

下列程式碼範例顯示如何設定應用程式使用 DynamoDB 來監視效能。

Java

適用於 Java 2.x 的 SDK

此範例顯示如何設定 Java 應用程式以監視 DynamoDB 的效能。應用程式會將測量結果資料傳送至您 CloudWatch 可以監視效能的位置。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- CloudWatch
- DynamoDB

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭 CloudWatch 配 AWS SDK 使用。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

Amazon 的安全性 CloudWatch

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。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端的安全性 — AWS 負責保護在 AWS 雲端中執行 AWS 服務的基礎架構。 AWS 還為您提供可以安全使用的服務。若要深入瞭解適用於的規範遵循計劃 CloudWatch，請參閱合規計劃的AWS 服務範圍範圍)。
- 雲端中的安全性 — 您的責任取決於您使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規

本文件可協助您了解如何在使用 Amazon 時應用共同的責任模型 CloudWatch。它說明如何設定 Amazon CloudWatch 以符合您的安全和合規目標。您還將學習如何使用其他 AWS 服務來幫助您監控和保護您的 CloudWatch 資源。

目錄

- Amazon 的數據保護 CloudWatch
- Amazon 的身份和訪問管理 CloudWatch
- Amazon 的合規驗證 CloudWatch
- Amazon 的韌性 CloudWatch
- Amazon 基礎設施安全 CloudWatch
- AWS Security Hub
- 使用 CloudWatch 和 CloudWatch Synthetics 與介面 VPC 端點
- Synthetics Canary 的安全考量

Amazon 的數據保護 CloudWatch

AWS 共同責任模型適用於 Amazon 中的資料保護 CloudWatch。如此模型中所述， AWS 負責保護執行所有 AWS 雲端. 您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

基於資料保護目的，我們建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 登入資料並設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源進行通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用設定 API 和使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案以及其中的所有默認安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在透過命令列介面或 API 存取時需要經 AWS 過 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用主控台、API CloudWatch 或 AWS SDK 時 AWS 服務 使用或其他使用時。 AWS CLI您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

傳輸中加密

CloudWatch 使用傳輸中的數據 end-to-end 加密。

Amazon 的身份和訪問管理 CloudWatch

AWS Identity and Access Management (IAM) 可協助管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM 管理員控制哪些人可以通過身份驗證 (登入) 和授權 (具有權限) 來使用 CloudWatch 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
- Amazon 如何與 IAM 合 CloudWatch 作
- Amazon 的基於身份的政策示例 CloudWatch
- 疑難排解 Amazon CloudWatch 身分和存取
- CloudWatch 儀表板權限更新

- AWS 的管理 (預先定義) 策略 CloudWatch
- 客戶受管政策範例
- CloudWatch AWS 受管理策略的更新
- 使用條件鍵限制對 CloudWatch 命名空間的訪問
- 使用條件索引鍵限制 Contributor Insights 使用者存取日誌群組
- 使用條件索引鍵限制警示動作
- 使用 CloudWatch 的服務連結角色
- 針 CloudWatch對 RUM 使用服務連結角色
- 將服務連結角色用於 CloudWatch 應用程式深入
- AWS Amazon CloudWatch 應用程式洞察的受管政策
- Amazon CloudWatch 許可參考

物件

您使用 AWS Identity and Access Management (IAM) 的方式會有所不同，具體取決於您在進行的工作 CloudWatch。

服務使用者 — 如果您使用 CloudWatch 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 CloudWatch 功能來完成工作時，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。如果無法存取中的圖徵 CloudWatch，請參閱疑難排解 Amazon CloudWatch 身分和存取。

服務管理員 — 如果您負責公司的 CloudWatch 資源，您可能擁有完整的存取權 CloudWatch。決定您的服務使用者應該存取哪些 CloudWatch 功能和資源是您的工作。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。若要進一步瞭解貴公司如何搭配使用 IAM CloudWatch，請參閱Amazon 如何與 IAM 合 CloudWatch 作。

IAM 管理員 — 如果您是 IAM 管理員，您可能想要瞭解如何撰寫政策來管理存取權限的詳細資訊 CloudWatch。若要檢視可在 IAM 中使用的 CloudWatch 基於身分的政策範例，請參閱。Amazon 的基於身份的政策示例 CloudWatch

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。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。如需更多資訊，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任務。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。

聯合身分

最佳作法是要求人類使用者 (包括需要系統管理員存取權的使用者) 使用與身分識別提供者的同盟，才能使用臨時認證 AWS 服務 來存取。

聯合身分識別是來自企業使用者目錄的使用者、Web 身分識別提供者、Identi ty Center 目錄，或使用透過身分識別來源提供的認證進行存取 AWS 服務 的任何使用者。 AWS Directory Service同盟身分存取時 AWS 帳戶，他們會假設角色，而角色則提供臨時認證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步到自己身分識別來源中的一組使用者和群組，以便在所有應用程式 AWS 帳戶 和應用程式中使用。如需 IAM Identity Center 的相關資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

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。如需更多資訊，請參閱 IAM 使用者指南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。您無法以群組身分簽署。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程序變得更為容易。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。如需進一步了解，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更多相關資訊，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。如需有關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提供者建立角色。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。
- 跨帳戶存取權 – 您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的委託人) 存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。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權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。如需更多資訊，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 執行個體內儲存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暫時憑證。如需更多資訊，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更多相關資訊，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

身分型政策是可以附加到身分 (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。若要了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 – 許可範圍是一種進階功能，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。如需許可範圍的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 實體許可範圍。

- 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。如需組織和 SCP 的更多相關資訊，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。如需更多資訊，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。

Amazon 如何與 IAM 合 CloudWatch 作

在您使用 IAM 管理存取權限之前 CloudWatch，請先了解哪些 IAM 功能可搭配使用 CloudWatch。

您可以與 Amazon 搭配使用的 IAM 功能 CloudWatch

IAM 功能	CloudWatch 支持
身分型政策	是
資源型政策	否
政策動作	是
政策資源	是
政策條件索引鍵 (服務特定)	是
ACL	否
ABAC(政策中的標籤)	部分

IAM 功能	CloudWatch 支持
臨時憑證	是
主體許可	是
服務角色	是
服務連結角色	否

若要深入瞭解如何以 CloudWatch 及其他 AWS 服務如何使用大多數 IAM 功能，請參閱 IAM 使用者指南中的搭配 IAM 使用的AWS 服務。

以身分識別為基礎的原則 CloudWatch

支援身分型政策	是

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。

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。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

以身分識別為基礎的原則範例 CloudWatch

若要檢視以 CloudWatch 身分為基礎的原則範例，請參閱。Amazon 的基於身份的政策示例 CloudWatch

以資源為基礎的政策 CloudWatch

支援以資源基礎的政策	否

資源型政策是附加到資源的 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

若要啟用跨帳戶存取，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，作為資源型政策的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主體和資源位於不同時 AWS 帳戶，受信任帳戶中的 IAM 管理員也必須授與主體實體 (使用者或角色) 權限，才能存取資源。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 角色與資源型政策有何差異。

的政策動作 CloudWatch

支援政策動作	是

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。也有一些作業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予執行相關聯動作的許可。

若要查看 CloudWatch 動作清單，請參閱服務授權參考 CloudWatch中 Amazon 定義的動作。

中的策略動作在動作之前 CloudWatch 使用下列前置詞：

```
cloudwatch
```

若要在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "cloudwatch:action1",
      "cloudwatch:action2"
         ]
```

若要檢視以 CloudWatch 身分為基礎的原則範例，請參閱。Amazon 的基於身份的政策示例 CloudWatch

的政策資源 CloudWatch

支援政策資源	是

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 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

要查看 CloudWatch 資源類型及其 ARN 的列表，請參閱服務授權參考 CloudWatch中由 Amazon 定義的資源。若要了解可以使用哪些動作指定每個資源的 ARN，請參閱 Amazon CloudWatch 定義的動作。

若要檢視以 CloudWatch 身分為基礎的原則範例，請參閱。Amazon 的基於身份的政策示例 CloudWatch

的政策條件索引鍵 CloudWatch

支援服務特定政策條件金鑰	是

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。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。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。

若要查看 CloudWatch 條件金鑰清單，請參閱服務授權參考 CloudWatch中的 Amazon 條件金鑰。若要了解您可以使用條件金鑰的動作和資源，請參閱 Amazon 定義的動作 CloudWatch。

若要檢視以 CloudWatch 身分為基礎的原則範例，請參閱。Amazon 的基於身份的政策示例 CloudWatch

ACL 在 CloudWatch

支援 ACL	否

存取控制清單 (ACL) 可控制哪些主體 (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。

阿巴克與 CloudWatch

支援 ABAC (政策中的標籤)	部分

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在中 AWS，這些屬性稱為標籤。您可以將標籤附加到 IAM 實體 (使用者或角色) 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

若要根據標籤控制存取，請使用 aws: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。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的什麼是 ABAC?。如要查看含有設定 ABAC 步驟的教學課程，請參閱《IAM 使用者指南》中的使用屬性型存取控制 (ABAC)。

使用臨時登入資料 CloudWatch

支援臨時憑證	是

當您使用臨時憑據登錄時，某些 AWS 服務 不起作用。如需其他資訊，包括哪些 AWS 服務 與臨時登入資料搭配AWS 服務 使用，請參閱 IAM 使用者指南中的 IAM。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立暫時認證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱《IAM 使用者指南》中的切換至角色 (主控台)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

的跨服務主體權限 CloudWatch

支援轉寄存取工作階段 (FAS)	是

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》。

CloudWatch 的服務角色

支援服務角色	是

服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。

> ⚠ Warning
>
> 變更服務角色的權限可能會中斷 CloudWatch功能。只有在 CloudWatch 提供指引時才編輯服務角色。

Amazon 的基於身份的政策示例 CloudWatch

依預設，使用者和角色沒有建立或修改 CloudWatch資源的權限。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 來執行工作。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。

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》中的建立 IAM 政策。

有關由定義的動作和資源類型的詳細資訊 CloudWatch，包括每種資源類型的 ARN 格式，請參閱服務授權參考 CloudWatch中適用於 Amazon 的動作、資源和條件金鑰。

主題

- 政策最佳實務
- 使用 CloudWatch 主控台

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 CloudWatch 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。如需更多資訊，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。
- 套用最低許可許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的權限。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。如需使用 IAM 套用許可的更多相關資訊，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。如需更多資訊，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。如需更多資訊，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。若要在呼叫 API 作業時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，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 CloudWatch 主控台

若要存取 Amazon CloudWatch 主控台，您必須擁有最少一組許可。這些權限必須允許您列出和檢視有關 AWS 帳戶. CloudWatch 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。

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，只需允許存取符合他們嘗試執行之 API 操作的動作就可以了。

若要確保使用者和角色仍可使用 CloudWatch 主控台，請同時將 CloudWatch *ConsoleAccess*或受*ReadOnly* AWS 管理的原則附加至實體。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。

CloudWatch控制台所需的權限

下面列出了使用 CloudWatch 控制台所需的完整權限集。這些權限提供對 CloudWatch控制台的完整寫入和讀取存取權限。

- 應用程式自動調度：DescribeScalingPolicies
- 自動調度資源：DescribeAutoScalingGroups
- 自動調度資源：DescribePolicies
- 雲步道：DescribeTrails
- 雲觀察:DeleteAlarms

- 雲觀察:DescribeAlarmHistory
- 雲觀察:DescribeAlarms
- 雲觀察:GetMetricData
- 雲觀察:GetMetricStatistics
- 雲觀察:ListMetrics
- 雲觀察:PutMetricAlarm
- 雲觀察:PutMetricData
- ec2：DescribeInstances
- ec2：DescribeTags
- ec2：DescribeVolumes
- 是:DescribeElasticsearchDomain
- 是:ListDomainNames
- 事件：DeleteRule
- 事件：DescribeRule
- 事件：DisableRule
- 事件：EnableRule
- 事件：ListRules
- 事件：PutRule
- IAM：AttachRolePolicy
- IAM：CreateRole
- IAM：GetPolicy
- IAM：GetPolicyVersion
- IAM：GetRole
- IAM：ListAttachedRolePolicies
- IAM：ListRoles
- 室壁運動:DescribeStream
- 室壁運動:ListStreams
- 拉姆達：AddPermission
- 拉姆達：CreateFunction

- 拉姆達：GetFunctionConfiguration
- 拉姆達：ListAliases
- 拉姆達：ListFunctions
- 拉姆達：ListVersionsByFunction
- 拉姆達：RemovePermission
- 日誌：CancelExportTask
- 日誌：CreateExportTask
- 日誌：CreateLogGroup
- 日誌：CreateLogStream
- 日誌：DeleteLogGroup
- 日誌：DeleteLogStream
- 日誌：DeleteMetricFilter
- 日誌：DeleteRetentionPolicy
- 日誌：DeleteSubscriptionFilter
- 日誌：DescribeExportTasks
- 日誌：DescribeLogGroups
- 日誌：DescribeLogStreams
- 日誌：DescribeMetricFilters
- 日誌：DescribeQueries
- 日誌：DescribeSubscriptionFilters
- 日誌：FilterLogEvents
- 日誌：GetLogGroupFields
- 日誌：GetLogRecord
- 日誌：GetLogEvents
- 日誌：GetQueryResults
- 日誌：PutMetricFilter
- 日誌：PutRetentionPolicy
- 日誌：PutSubscriptionFilter
- 日誌：StartQuery

- 日誌：StopQuery
- 日誌：TestMetricFilter
- S3：CreateBucket
- S3：ListBucket
- SNS: CreateTopic
- SNS: GetTopicAttributes
- SNS: ListSubscriptions
- SNS: ListTopics
- SNS: SetTopicAttributes
- sns:Subscribe
- sns:Unsubscribe
- 平方：GetQueueAttributes
- 平方：GetQueueUrl
- 平方：ListQueues
- 平方：SetQueueAttributes
- SWF：CreateAction
- SWF：DescribeAction
- SWF：ListActionTemplates
- SWF：RegisterAction
- SWF：RegisterDomain
- SWF：UpdateAction

此外，要查看 X-Ray 追蹤地圖，您需要 `AWSXrayReadOnlyAccess`

疑難排解 Amazon CloudWatch 身分和存取

使用下列資訊可協助您診斷和修正使用和 IAM 時可能會遇到的 CloudWatch 常見問題。

主題

- 我沒有執行操作的授權 CloudWatch
- 我沒有授權執行 iam：PassRole

- 我想允許我以外的人訪 AWS 帳戶 問我的 CloudWatch 資源

我沒有執行操作的授權 CloudWatch

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

下列範例錯誤會在mateojackson IAM 使用者嘗試使用主控台檢視一個虛構 *my-example-widget* 資源的詳細資訊，但卻無虛構 cloudwatch:*GetWidget* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 cloudwatch:GetWidget on resource: my-example-widget
```

在此情況下，必須更新 mateojackson 使用者的政策，允許使用 cloudwatch:*GetWidget* 動作存取 *my-example-widget* 資源。

如果您需要協助，請聯絡您的 AWS 系統管理員。您的管理員提供您的簽署憑證。

我沒有授權執行 iam：PassRole

如果您收到未獲授權執行iam:PassRole動作的錯誤訊息，則必須更新您的原則以允許您將角色傳遞給 CloudWatch。

有些 AWS 服務 允許您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為的 IAM 使用者marymajor嘗試使用主控台執行中的動作時，會發生下列範例錯誤 CloudWatch。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。

如果您需要協助，請聯絡您的 AWS 系統管理員。您的管理員提供您的簽署憑證。

我想允許我以外的人訪 AWS 帳戶 問我的 CloudWatch 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物件取得該角色。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的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要瞭解是否 CloudWatch 支援這些功能，請參閱Amazon 如何與 IAM 合 CloudWatch 作。
- 若要了解如何提供對您所擁有資源 AWS 帳戶 的存取權，請參閱《IAM 使用者指南》中您擁有的另一 AWS 帳戶 個 IAM 使用者提供存取權限。
- 若要了解如何將資源存取權提供給第三方 AWS 帳戶，請參閱 IAM 使用者指南中的提供第三方 AWS 帳戶 擁有的存取權。
- 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。

CloudWatch 儀表板權限更新

自 2018 年 5 月 1 日起， AWS 變更了存取 CloudWatch 儀表板所需的權限。 CloudWatch 控制台中的儀表板存取現在需要 2017 年引入的權限才能支援儀表板 API 操作：

- 雲觀察:GetDashboard
- 雲觀察:ListDashboards
- 雲觀察:PutDashboard
- 雲觀察:DeleteDashboards

若要存取 CloudWatch 儀表板，您需要下列其中一項：

- 政AdministratorAccess策。
- 政CloudWatchFullAccess策。
- 自訂政策，其中包含一或多個特定許可：
  - `cloudwatch:GetDashboard` 和 `cloudwatch:ListDashboards` 能夠檢視儀表板
  - `cloudwatch:PutDashboard` 能夠建立或修改儀表板
  - `cloudwatch:DeleteDashboards` 能夠刪除儀表板

如需有關使用政策變更 IAM 使用者許可的詳細資訊，請參閱「變更 IAM 使用者的許可」。

如需有關 CloudWatch 權限的詳細資訊，請參閱Amazon CloudWatch 許可參考。

如需儀表板 API 操作的詳細資訊，請參閱 Amazon CloudWatch API 參考PutDashboard中的。

AWS 的管理 (預先定義) 策略 CloudWatch

AWS 透過提供由建立和管理的獨立 IAM 政策來解決許多常見使用案例 AWS。這些 AWS 受管理的政策會為常見使用案例授與必要的權限，因此您可以避免調查需要哪些權限。如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

下列 AWS 受管理的策略 (您可以附加至帳戶中的使用者) 是特定的 CloudWatch。

主題

- CloudWatchFullAccessV2
- CloudWatchFullAccess
- CloudWatchReadOnlyAccess
- CloudWatchActions存取
- CloudWatchAutomaticDashboardsAccess
- CloudWatchAgentServerPolicy
- CloudWatchAgentAdminPolicy
- AWS CloudWatch 跨帳戶可觀察性的託管（預先定義）策略
- AWS CloudWatch Synthetics 的管理 (預先定義) 政策
- AWS Amazon CloudWatch RUM 的受管（預先定義）政策
- AWS CloudWatch 明顯的託管（預定義）策略
- AWSAWS 系統管理員事件管理員的管理原則

CloudWatchFullAccessV2

AWS 最近新增了 CloudWatchFullAccessV2 受管 IAM 政策。此政策授予對 CloudWatch 動作和資源的完整存取權，並且更適當地設定授與其他服務 (例如 Amazon SNS 和) 的許可範圍 Amazon EC2 Auto Scaling。我們建議您開始使用此政策，而不是使用CloudWatchFullAccess。 AWS 計劃在不久的CloudWatchFullAccessfuture 棄用。

它包含`application-signals:`許可權，以便用戶可以從「應用程序信號」下的 CloudWatch 控制台訪問所有功能。它包含一些`autoscaling:Describe`權限，因此具有此原則的使用者可以看到與 CloudWatch警示相關聯的 Auto Scaling 動作。它包括一些`sns`許可，以便具有此政策的使用者可以擷取建立 Amazon SNS 主題並將其與 CloudWatch 警示建立關聯。它包含 IAM 許

可，因此具有此政策的使用者可以檢視與相關聯的服務連結角色的相 CloudWatch關資訊。它包含oam:ListSinks和oam:ListAttachedLinks權限，以便具有此策略的使用者可以使用主控台以 CloudWatch 跨帳戶觀察性檢視從來源帳戶共用的資料。

它包括 rumsynthetics，和xray權限，以便用戶可以完全訪問 CloudWatch Synthetics AWS X-Ray，和 CloudWatch RUM，所有這些都在 CloudWatch 服務下。

CloudWatchFullAccessV2 的內容如下：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchFullAccessPermissions",
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:DescribeScalingPolicies",
                "application-signals:*",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribePolicies",
                "cloudwatch:*",
                "logs:*",
                "sns:CreateTopic",
                "sns:ListSubscriptions",
                "sns:ListSubscriptionsByTopic",
                "sns:ListTopics",
                "sns:Subscribe",
                "iam:GetPolicy",
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "oam:ListSinks",
                "rum:*",
                "synthetics:*",
                "xray:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchApplicationSignalsServiceLinkedRolePermissions",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/application-
signals.cloudwatch.amazonaws.com/AWSServiceRoleForCloudWatchApplicationSignals",
```

```
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "application-
signals.cloudwatch.amazonaws.com"
                }
            }
        },
        {
            "Sid": "EventsServicePermissions",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/events.amazonaws.com/
AWSServiceRoleForCloudWatchEvents*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "events.amazonaws.com"
                }
            }
        },
         {
            "Sid": "OAMReadPermissions",
            "Effect": "Allow",
            "Action": [
                "oam:ListAttachedLinks"
            ],
            "Resource": "arn:aws:oam:*:*:sink/*"
        }
    ]
}
```

CloudWatchFullAccess

該CloudWatchFullAccess策略正在棄用的路徑上。我們建議您停止使用它，並改用CloudWatchFullAccessV2。

的內容CloudWatchFullAccess如下：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "autoscaling:Describe*",
                "cloudwatch:*",
                "logs:*",
                "sns:*",
                "iam:GetPolicy",
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "oam:ListSinks"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/events.amazonaws.com/
AWSServiceRoleForCloudWatchEvents*",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "events.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "oam:ListAttachedLinks"
            ],
            "Resource": "arn:aws:oam:*:*:sink/*"
        }
    ]
}
```

CloudWatchReadOnlyAccess

CloudWatchReadOnlyAccess原則會授與的唯讀存取權 CloudWatch。

此原則包含一些logs:權限，因此具有此原則的使用者可以使用主控台來檢視 CloudWatch 記錄資訊和 CloudWatch 記錄檔見解查詢。其中包括autoscaling:Describe*，以便具有此原則的使用者可以看到與 CloudWatch 警示相關聯的 Auto Scaling 動作。它包括application-signals:權限，以便用戶可以使用應用程序信號來監視其服務的健康狀態。包括 application-autoscaling:DescribeScalingPolicies，讓具有此政策的使用者可以存取 Application Auto Scaling 政策的相關資訊。它包括sns:Get*和sns:List*，以便具

有此政策的使用者可以擷取有關接收CloudWatch 警示通知之 Amazon SNS 主題的資訊。它包含oam:ListSinks和oam:ListAttachedLinks權限，因此具有此策略的使用者可以使用主控台，以 CloudWatch 跨帳戶觀察性檢視從來源帳戶共用的資料。它包括iam:GetRole權限，以便用戶可以檢查是否已設置 CloudWatch 應用程序信號。

它包括 rumsynthetics，和xray權限，以便用戶可以對 CloudWatch Synthetics， AWS X-Ray和 CloudWatch RUM 具有唯讀訪問權限，所有這些都在 CloudWatch 服務下。

以下是政CloudWatchReadOnlyAccess策的內容。

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CloudWatchReadOnlyAccessPermissions",
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:DescribeScalingPolicies",
                "application-signals:BatchGet*",
                "application-signals:Get*",
                "application-signals:List*",
                "autoscaling:Describe*",
                "cloudwatch:BatchGet*",
                "cloudwatch:Describe*",
                "cloudwatch:GenerateQuery",
                "cloudwatch:Get*",
                "cloudwatch:List*",
                "logs:Get*",
                "logs:List*",
                "logs:StartQuery",
                "logs:StopQuery",
                "logs:Describe*",
                "logs:TestMetricFilter",
                "logs:FilterLogEvents",
                "logs:StartLiveTail",
                "logs:StopLiveTail",
                "oam:ListSinks",
                "sns:Get*",
                "sns:List*",
                "rum:BatchGet*",
                "rum:Get*",
                "rum:List*",
                "synthetics:Describe*",
```

```
                "synthetics:Get*",
                "synthetics:List*",
                "xray:BatchGet*",
                "xray:Get*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "OAMReadPermissions",
            "Effect": "Allow",
            "Action": [
                "oam:ListAttachedLinks"
            ],
            "Resource": "arn:aws:oam:*:*:sink/*"
        },
        {
            "Sid": "CloudWatchReadOnlyGetRolePermissions",
            "Effect": "Allow",
            "Action": "iam:GetRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/application-
signals.cloudwatch.amazonaws.com/AWSServiceRoleForCloudWatchApplicationSignals"
        }
    ]
}
```

CloudWatchActions存取

除了 Amazon EC2 中繼資料之外，CloudWatchActionsEC2Access 政策還授予 CloudWatch 警示和指標的唯讀存取權。其還會授予存取以停止、終止和重新啟動適用於 EC2 執行個體的 API 動作。

以下是 CloudWatchActionsEC2Access 政策的內容。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:Describe*",
        "ec2:Describe*",
        "ec2:RebootInstances",
        "ec2:StopInstances",
        "ec2:TerminateInstances"
```

```
      ],
      "Resource": "*"
    }
  ]
}
```

CloudWatchAutomaticDashboardsAccess

CloudWatch-CrossAccountAccess CrossAccountSharingRole IAM 角色會使用 CloudWatch- 受管政策。此角色和政策可讓跨帳戶儀表板的使用者檢視共用儀表板的每個帳戶中的自動儀表板。

以下是內容 CloudWatchAutomaticDashboardsAccess：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "autoscaling:DescribeAutoScalingGroups",
        "cloudfront:GetDistribution",
        "cloudfront:ListDistributions",
        "dynamodb:DescribeTable",
        "dynamodb:ListTables",
        "ec2:DescribeInstances",
        "ec2:DescribeVolumes",
        "ecs:DescribeClusters",
        "ecs:DescribeContainerInstances",
        "ecs:ListClusters",
        "ecs:ListContainerInstances",
        "ecs:ListServices",
        "elasticache:DescribeCacheClusters",
        "elasticbeanstalk:DescribeEnvironments",
        "elasticfilesystem:DescribeFileSystems",
        "elasticloadbalancing:DescribeLoadBalancers",
        "kinesis:DescribeStream",
        "kinesis:ListStreams",
        "lambda:GetFunction",
        "lambda:ListFunctions",
        "rds:DescribeDBClusters",
        "rds:DescribeDBInstances",
        "resource-groups:ListGroupResources",
        "resource-groups:ListGroups",
        "route53:GetHealthCheck",
```

```
        "route53:ListHealthChecks",
        "s3:ListAllMyBuckets",
        "s3:ListBucket",
        "sns:ListTopics",
        "sqs:GetQueueAttributes",
        "sqs:GetQueueUrl",
        "sqs:ListQueues",
        "synthetics:DescribeCanariesLastRun",
        "tag:GetResources"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": [
        "apigateway:GET"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:apigateway:*::/restapis*"
      ]
    }
  ]
```

CloudWatchAgentServerPolicy

該CloudWatchAgentServerPolicy政策可用於連接到 Amazon EC2 執行個體的 IAM 角色，以允許 CloudWatch 代理程式從執行個體讀取資訊並將其寫入 CloudWatch。其內容如下。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CWACloudWatchServerPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "ec2:DescribeVolumes",
                "ec2:DescribeTags",
                "logs:PutLogEvents",
                "logs:PutRetentionPolicy",
                "logs:DescribeLogStreams",
                "logs:DescribeLogGroups",
```

```
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets",
                "xray:GetSamplingStatisticSummaries"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CWASSMServerPermissions",
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameter"
            ],
            "Resource": "arn:aws:ssm:*:*:parameter/AmazonCloudWatch-*"
        }
    ]
}
```

CloudWatchAgentAdminPolicy

此CloudWatchAgentAdminPolicy政策可用於連接至 Amazon EC2 執行個體的 IAM 角色。此原則可讓 CloudWatch 代理程式從執行個體讀取資訊並將其寫入 CloudWatch，以及將資訊寫入參數存放區。其內容如下。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CWACloudWatchPermissions",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData",
                "ec2:DescribeTags",
                "logs:PutLogEvents",
                "logs:PutRetentionPolicy",
                "logs:DescribeLogStreams",
                "logs:DescribeLogGroups",
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "xray:PutTraceSegments",
```

```
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets",
                "xray:GetSamplingStatisticSummaries"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CWASSMPermissions",
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameter",
                "ssm:PutParameter"
            ],
            "Resource": "arn:aws:ssm:*:*:parameter/AmazonCloudWatch-*"
        }
    ]
}
```

> **Note**
>
> 您可以登入 IAM 主控台並在該處搜尋特定政策，來檢閱這些許可政策。

您也可以建立自己的自訂 IAM 政策，以允許 CloudWatch 動作和資源的許可。您可以將這些自訂政策連接至需要這些許可的 IAM 使用者或群組。

AWS CloudWatch 跨帳戶可觀察性的託管（預先定義）策略

本節中的政策授予與 CloudWatch 跨帳戶觀察性相關的權限。如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。

CloudWatchCrossAccountSharingConfiguration

此CloudWatchCrossAccountSharingConfiguration原則會授予建立、管理和檢視可觀察性存取管理員連結的存取權，以便在帳號之間共用 CloudWatch 資源。如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。其內容如下：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "cloudwatch: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/*"
            ]
        }
    ]
}
```

OAM FullAccess

OAM FullAccess 政策授予建立、管理和檢視可觀察性存取管理員接收器和連結的存取權，這些連結用於 CloudWatch跨帳戶觀察性。

OAM FullAccess 政策本身不允許您跨連結共用可觀察性資料。若要建立共用 CloudWatch 量度的連結，您還需要CloudWatchFullAccess或CloudWatchCrossAccountSharingConfiguration。若要建立共用 CloudWatch 記錄檔記錄群組的連結，您還需要CloudWatchLogsFullAccess或CloudWatchLogsCrossAccountSharingConfiguration。若要建立共用 X-Ray 軌跡的連結，您還需要AWSXRayFullAccess或AWSXRayCrossAccountSharingConfiguration。

如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。其內容如下：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "oam:*"
            ],
            "Resource": "*"
        }
    ]
}
```

OAM ReadOnlyAccess

OAM ReadOnlyAccess 原則會授與可觀察性存取管理員資源的唯讀存取權，這些資源用於 CloudWatch 跨帳戶觀察性。如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。其內容如下：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "oam:Get*",
                "oam:List*"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS CloudWatch Synthetics 的管理 (預先定義) 政策

您可以將CloudWatchSyntheticsFullAccess和CloudWatchSyntheticsReadOnlyAccess AWS 受管理的策略指派給將管理或使用 CloudWatch Synthetics 的使用者。下列其他政策也是相關的：

- AmazonS3 ReadOnlyAccess 和 CloudWatchReadOnlyAccess— 這些對於能夠讀取控制台中的所有 Synthetics 數據是必不可少的。 CloudWatch
- AWSLambdaReadOnlyAccess— 為了能夠查看金絲雀使用的源代碼。

- CloudWatchSyntheticsFullAccess使您能夠創建 Canary，此外，要創建和刪除具有為其創建新的 IAM 角色的 Canary，還需要以下內嵌政策聲明：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:CreatePolicy",
                "iam:DeletePolicy",
                "iam:AttachRolePolicy",
                "iam:DetachRolePolicy",
            ],
            "Resource": [
                "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*",
                "arn:aws:iam::*:policy/service-role/CloudWatchSyntheticsPolicy*"
            ]
        }
    ]
}
```

> **Important**
>
> 向使用者授予 iam:CreateRole、iam:DeleteRole、iam:CreatePolicy、iam:DeletePolicy、iam:Atta 和 iam:DetachRolePolicy 許可，該使用者便擁有完整管理存取權，可建立、連接和刪除具有與 arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole* 和 arn:aws:iam::*:policy/service-role/CloudWatchSyntheticsPolicy* 相符之 ARN 的角色和政策。例如，擁有這些許可的使用者可以建立具有所有資源完整許可的政策，並將該政策連接至符合該 ARN 模式的任何角色。對於您授與這些許可的對象，請務必謹慎。

如需連接政策和授與許可給使用者的資訊，請參閱變更 IAM 使用者的許可和嵌入使用者或角色的內嵌政策。

CloudWatchSyntheticsFullAccess

以下是政CloudWatchSyntheticsFullAccess策的內容。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "synthetics:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutEncryptionConfiguration"
            ],
            "Resource": [
                "arn:aws:s3:::cw-syn-results-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles",
                "s3:ListAllMyBuckets",
                "xray:GetTraceSummaries",
                "xray:BatchGetTraces",
                "apigateway:GET"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::cw-syn-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::aws-synthetics-library-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lambda.amazonaws.com",
                        "synthetics.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:ListAttachedRolePolicies"
            ],
            "Resource": [
                "arn:aws:iam::*:role/service-role/CloudWatchSyntheticsRole*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
```

```
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DeleteAlarms"
            ],
            "Resource": [
                "arn:aws:cloudwatch:*:*:alarm:Synthetics-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": [
                "arn:aws:cloudwatch:*:*:alarm:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:CreateFunction",
                "lambda:AddPermission",
                "lambda:PublishVersion",
                "lambda:UpdateFunctionCode",
                "lambda:UpdateFunctionConfiguration",
                "lambda:GetFunctionConfiguration",
                "lambda:DeleteFunction"
            ],
            "Resource": [
                "arn:aws:lambda:*:*:function:cwsyn-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:GetLayerVersion",
                "lambda:PublishLayerVersion",
                "lambda:DeleteLayerVersion"
```

```
            ],
            "Resource": [
                "arn:aws:lambda:*:*:layer:cwsyn-*",
                "arn:aws:lambda:*:*:layer:Synthetics:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:ListTopics"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:CreateTopic",
                "sns:Subscribe",
                "sns:ListSubscriptionsByTopic"
            ],
            "Resource": [
                "arn:*:sns:*:*:Synthetics-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:ListAliases"
            ],
            "Resource": "*"
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:*:*:key/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:*:*:key/*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                        "s3.*.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

CloudWatchSyntheticsReadOnlyAccess

以下是政CloudWatchSyntheticsReadOnlyAccess策的內容。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "synthetics:Describe*",
                "synthetics:Get*",
                "synthetics:List*",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS Amazon CloudWatch RUM 的受管（預先定義）政策

您可以將 AmazonCloudWatchR AmazonCloudWatchUM FullAccess 和 RUM ReadOnlyAccess AWS 管理的政策指派給將管理或使用 CloudWatch RUM 的使用者。

AmazonCloudWatch朗姆酒 FullAccess

以下是 R AmazonCloudWatchUM FullAccess 政策的內容。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rum:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/rum.amazonaws.com/
AWSServiceRoleForRealUserMonitoring"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/RUM-Monitor*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "cognito-identity.amazonaws.com"
                    ]
                }
```

```
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DescribeAlarms"
            ],
            "Resource": "arn:aws:cloudwatch:*:*:alarm:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cognito-identity:CreateIdentityPool",
                "cognito-identity:ListIdentityPools",
                "cognito-identity:DescribeIdentityPool",
                "cognito-identity:GetIdentityPoolRoles",
                "cognito-identity:SetIdentityPoolRoles"
            ],
            "Resource": "arn:aws:cognito-identity:*:*:identitypool/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:DeleteLogGroup",
                "logs:PutRetentionPolicy",
                "logs:CreateLogStream"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:*RUMService*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
```

```
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "logs:DescribeResourcePolicies"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": "arn:aws:logs:*:*:log-group::log-stream:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "synthetics:describeCanaries",
                "synthetics:describeCanariesLastRun"
            ],
            "Resource": "arn:aws:synthetics:*:*:canary:*"
        }
    ]
}
```

AmazonCloudWatch朗姆酒 ReadOnlyAccess

以下是 R AmazonCloudWatchUM ReadOnlyAccess 政策的內容。

```
{

 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "rum:GetAppMonitor",
    "rum:GetAppMonitorData",
    "rum:ListAppMonitors",
    "rum:ListRumMetricsDestinations",
    "rum:BatchGetRumMetricDefinitions"
   ],
   "Resource": "*"
  }
```

```
 ]
}
```

AmazonCloudWatch朗姆酒 ServiceRolePolicy

您無法將 AmazonCloudWatchRUM 附加ServiceRolePolicy到 IAM 實體。此原則附加至服務連結角色，可讓 CloudWatch RUM 將監視資料發佈至其他相關 AWS 服務。如需此服務連結角色的詳細資訊，請參閱 針 CloudWatch對 RUM 使用服務連結角色。

AmazonCloudWatch朗姆酒的完整內容ServiceRolePolicy如下。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "xray:PutTraceSegments"
   ],
   "Resource": [
    "*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": "cloudwatch:PutMetricData",
   "Resource": "*",
   "Condition": {
    "StringLike": {
     "cloudwatch:namespace": [
      "RUM/CustomMetrics/*",
      "AWS/RUM"
     ]
    }
   }
  }
 ]
}
```

AWS CloudWatch 明顯的託管（預定義）策略

您可以將CloudWatchEvidentlyFullAccess和CloudWatchEvidentlyReadOnlyAccess AWS 受管理的政策指派給將管理或使用「 CloudWatch明顯」的使用者。

CloudWatchEvidentlyFullAccess

以下是該CloudWatchEvidentlyFullAccess政策的內容。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "evidently:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/service-role/CloudWatchRUMEvidentlyRole-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:DescribeAlarmHistory",
```

```
                "cloudwatch:DescribeAlarmsForMetric",
                "cloudwatch:ListTagsForResource"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DescribeAlarms",
                "cloudwatch:TagResource",
                "cloudwatch:UnTagResource"
            ],
            "Resource": [
                "arn:aws:cloudwatch:*:*:alarm:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudtrail:LookupEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricAlarm"
            ],
            "Resource": [
                "arn:aws:cloudwatch:*:*:alarm:Evidently-Alarm-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:ListTopics"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "sns:CreateTopic",
                "sns:Subscribe",
                "sns:ListSubscriptionsByTopic"
            ],
            "Resource": [
                "arn:*:sns:*:*:Evidently-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

CloudWatchEvidentlyReadOnlyAccess

以下是該CloudWatchEvidentlyReadOnlyAccess政策的內容。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "evidently:GetExperiment",
                "evidently:GetFeature",
                "evidently:GetLaunch",
                "evidently:GetProject",
                "evidently:GetSegment",
                "evidently:ListExperiments",
                "evidently:ListFeatures",
                "evidently:ListLaunches",
                "evidently:ListProjects",
                "evidently:ListSegments",
                "evidently:ListSegmentReferencs"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

AWSAWS 系統管理員事件管理員的管理原則

此AWSCloudWatchAlarms_ActionSSMIncidentsServiceRolePolicy原則附加至服務連結角色，可讓 CloudWatch 您代表您在 AWS Systems Manager 事件管理員中啟動事件。如需詳細資訊，請參閱 CloudWatch 警示系統管理員事件管理員動作的服務連結角色權限。

下列政策具有以下許可：

- 短信事件：StartIncident

客戶受管政策範例

在本節中，您可以找到授與各種CloudWatch 動作權限的範例使用者策略。當您使用 CloudWatch API、 AWS SDK 或. AWS CLI

範例

- 範例 1：允許使用者完整存取 CloudWatch
- 範例 2：允許唯讀存取 CloudWatch
- 範例 3：停止或終止 Amazon EC2 執行個體

範例 1：允許使用者完整存取 CloudWatch

若要授與使用者的完整存取權 CloudWatch，您可以使用授與受CloudWatchFullAccess管政策，而不是建立客戶管理的政策。中列出CloudWatchFullAccess的內容CloudWatchFullAccess。

範例 2：允許唯讀存取 CloudWatch

下列政策允許使用者以唯讀方式存取 CloudWatch 和檢視 Amazon EC2 自動擴展動作、 CloudWatch 指標、 CloudWatch 日誌資料和警報相關的 Amazon SNS 資料。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "autoscaling:Describe*",
```

```
        "cloudwatch:Describe*",
        "cloudwatch:Get*",
        "cloudwatch:List*",
        "logs:Get*",
        "logs:Describe*",
        "logs:StartQuery",
        "logs:StopQuery",
        "logs:TestMetricFilter",
        "logs:FilterLogEvents",
        "logs:StartLiveTail",
        "logs:StopLiveTail",
        "sns:Get*",
        "sns:List*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

範例 3：停止或終止 Amazon EC2 執行個體

下列政策允許停止或終止 EC2 執行個體的 CloudWatch 警示動作。在下面的範例中， GetMetricData ListMetrics、和 DescribeAlarms動作是選擇性的。建議您包含這些動作，以確保您正確地停止或終止執行個體。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudwatch:PutMetricAlarm",
        "cloudwatch:GetMetricData",
        "cloudwatch:ListMetrics",
        "cloudwatch:DescribeAlarms"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
```

```
        "ec2:DescribeInstanceStatus",
        "ec2:DescribeInstances",
        "ec2:StopInstances",
        "ec2:TerminateInstances"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

CloudWatch AWS 受管理策略的更新

檢視 CloudWatch 自此服務開始追蹤這些變更以來的 AWS 受管理策略更新詳細資料。如需有關此頁面變更的自動警示，請訂閱「 CloudWatch 文件歷史記錄」頁面上的 RSS 摘要。

變更	描述	日期
CloudWatchFullAccessV2 — 現有策略的更新	CloudWatch 更新了名為 CloudWatchFullAccessV2 的策略。 `CloudWatchFullAccessPermissions` 政策的範圍已更新為新增，以`application-signals:*` 便使用者可以使用 CloudWatch應用程式信號來檢視、調查和診斷其服務健康狀況的問題。	2024年5月20日
CloudWatchReadOnlyAccess – 更新現有政策	CloudWatch 更新了名為的策略CloudWatchReadOnlyAccess。 `CloudWatchReadOnlyAccessPermissions` 原	2024年5月20日

變更	描述	日期
	則的範圍已更新為新增application-signals:BatchGet* 、application-signals:List* ，以application-signals:Get* 便使用者可以使用CloudWatch應用程式信號來檢視、調查及診斷其服務健康狀況的問題。的範圍CloudWatchReadOnlyGetRolePermissions 已更新以新增iam:GetRole 動作，以便使用者可以檢查是否已設定CloudWatch 應用程式訊號。	
CloudWatchApplicationSignalsServiceRolePolicy – 更新現有政策	CloudWatch 更新了名為的策略CloudWatchApplicationSignalsServiceRolePolicy。 logs:StartQuery 和logs:GetQueryResults 權限的範圍已變更，以新增arn:aws:logs:*:*:log-group:/aws/appsignals/*:* 和arn:aws:logs:*:*:log-group:/aws/application-signals/data:* ARN，以便在更多架構上啟用應用程式訊號。	2024年4月18日

變更	描述	日期
CloudWatchApplicationSignalsServiceRolePolicy – 更新現有政策	CloudWatch 變更中的權限範圍CloudWatchApplicationSignalsServiceRolePolicy。 `cloudwatch:GetMetricData` 權限的範圍已變更為 `*`，以便應用程式信號可以從連結帳戶中的來源擷取指標。	2024年4月08 日
CloudWatchAgentServerPolicy – 更新現有政策	CloudWatch 將權限添加到 CloudWatchAgentServerPolicy . 已新增`xray:PutTraceSegments` `xray:PutTelemetryRecords` 、`xray:GetSamplingRules` 、`xray:GetSamplingStatisticSummaries` 和`logs:PutRetentionPolicy` 權限`xray:GetSamplingTargets` ，以便 CloudWatch 代理程式可以發佈 X-Ray 追蹤並修改記錄群組保留期間。	2024年2月12日

變更	描述	日期
CloudWatchAgentAdminPolicy – 更新現有政策	CloudWatch 將權限添加到 CloudWatchAgentAdminPolicy. 已新增`xray:PutTraceSegments` `xray:PutTelemetryRecords` 、`xray:GetSamplingRules` 、`xray:GetSamplingStatisticSummaries` 和`logs:PutRetentionPolicy` 權限`xray:GetSamplingTargets` ，以便 CloudWatch 代理程式可以發佈 X-Ray 追蹤並修改記錄群組保留期間。	2024年2月12日

變更	描述	日期
CloudWatchFullAccessV2 — 現有策略的更新	CloudWatch 向 CloudWatchFullAccessV2 添加了權限。 已新增 CloudWatch Synthetics、X-Ray 和 RUM 動作的現有權限，以及 CloudWatch 應用程式訊號的新權限，讓具有此原則的使用者可以管理 CloudWatch 應用程式訊號。 已新增建立「CloudWatch 應用程式信號」服務連結角色的權限，以允許「CloudWatch 應用程式訊號」探索記錄檔、度量、追蹤和標籤中的遙測資料。	2023 年 12 月 5 日
CloudWatchReadOnlyAccess – 更新現有政策	CloudWatch 將權限添加到 CloudWatchReadOnlyAccess. 已新增 CloudWatch Synthetics、X-Ray 和 RUM 動作的現有唯讀權限，以及 CloudWatch 應用程式訊號的新唯讀權限，以便具有此原則的使用者可分類和消除應用程式 Sigals 所報告的服務健康狀態問題。 CloudWatch 已新增`cloudwatch:GenerateQuery` 權限，以便具有此原則的使用者可以從自然語言提示中產生 CloudWatch 指標見解查詢字串。	2023 年 12 月 5 日

變更	描述	日期
CloudWatchApplicationSignalsServiceRolePolicy – 新政策	CloudWatch 增加了一個新的策略CloudWatchApplicationSignalsServiceRolePolicy。 即將CloudWatchApplicationSignalsServiceRolePolicy授予即將到來的功能權限，以收集 CloudWatch 日誌資料、X-Ray 追蹤資料、 CloudWatch 指標資料和標記資料。	2023 年 11 月 9 日
AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsServiceRolePolicy – 新政策	CloudWatch 增加了一個新的策略AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsServiceRolePolicy。 AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsServiceRolePolicy授與代表您從資料庫擷取 Performance Insights 指標的權限。 CloudWatch	2023 年 9 月 20 日
CloudWatchReadOnlyAccess – 更新現有政策	CloudWatch 已將權限新增至 CloudWatchReadOnlyAccess. 新增了 `application-autoscaling:DescribeScalingPolicies` 許可，讓具有此政策的使用者可以存取 Application Auto Scaling 政策的相關資訊。	2023 年 9 月 14 日

變更	描述	日期
CloudWatchFullAccessV2 — 新政策	CloudWatch 添加了一個新的策略 CloudWatchFullAccessV2。 CloudWatchFullAccessV2 可授與 CloudWatch 動作和資源的完整存取權，同時更好地限定授與其他服務 (例如 Amazon SNS 和) 的許可範圍。 Amazon EC2 Auto Scaling如需詳細資訊，請參閱 CloudWatchFullAccessV2。	2023 年 8 月 1 日
AWSServiceRoleForInternetMonitor – 更新現有政策	Amazon 網 CloudWatch 際網路監視器新增了新的許可來監控 Network Load Balancer 需有 `elasticloadbalancing:DescribeLoadBalancers` 和 `ec2:DescribeNetworkInterfaces` 許可，網路監視器才能分析 NLB 資源的流程日誌，監控客戶的 Network Load Balancer 流量。 如需詳細資訊，請參閱 使用 Amazon CloudWatch 網絡監控。	2023 年 7 月 15 日

變更	描述	日期
CloudWatchReadOnlyAccess – 更新現有政策	CloudWatch 將權限添加到 CloudWatchReadOnlyAccess. `logs:StartLiveTail` 和`logs:StopLiveTail` 權限已新增，讓具有此原則的使用者可以使用主控台來啟動和停止 CloudWatch 記錄即時尾端工作階段。如需詳細資訊，請參閱使用 Live Tail 以近乎即時的方式檢視日誌。	2023 年 6 月 6 日
CloudWatchCrossAccountSharingConfiguration – 新政策	CloudWatch 新增政策可讓您管理共 CloudWatch 用指標的 CloudWatch 跨帳戶觀察性連結。 如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。	2022 年 11 月 27 日
OAM FullAccess — 新政策	CloudWatch 新增政策可讓您完全管理 CloudWatch跨帳戶可觀察性連結和接收器。 如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。	2022 年 11 月 27 日

變更	描述	日期
OAM ReadOnlyAccess — 新政策	CloudWatch 新增政策可讓您檢視有關CloudWatch 跨帳戶可觀察性連結和接收器的資訊。 如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。	2022 年 11 月 27 日
CloudWatchFullAccess – 更新現有政策	CloudWatch 將權限添加到 CloudWatchFullAccess. 已新增`oam:ListSinks` 和`oam:ListAttachedLinks` 權限，以便具有此原則的使用者可以使用主控台，以 CloudWatch 跨帳戶觀察性檢視從來源帳戶共用的資料。	2022 年 11 月 27 日
CloudWatchReadOnlyAccess – 更新現有政策	CloudWatch 將權限添加到 CloudWatchReadOnlyAccess. 已新增`oam:ListSinks` 和`oam:ListAttachedLinks` 權限，以便具有此原則的使用者可以使用主控台，以 CloudWatch 跨帳戶觀察性檢視從來源帳戶共用的資料。	2022 年 11 月 27 日

變更	描述	日期
AmazonCloudWatchRUMServiceRolePolicy — 現有政策的更新	CloudWatch 朗姆酒更新了 R AmazonCloudWatchUM 中的條件鍵ServiceRolePolicy。 `"Condition": { "StringEquals": { "cloudwatch:namespace": "AWS/RUM" } }`條件鍵已更改為以下內容，以便 CloudWatch RUM 可以將自定義指標發送到自定義指標命名空間。 `"Condition": { "StringLike": { "cloudwatch:namespace": [ "RUM/CustomMetrics/*", "AWS/RUM" ] } }`	2023 年 2 月 2 日

變更	描述	日期
AmazonCloudWatch朗姆酒ReadOnlyAccess-更新的政策	CloudWatch 添加權限的 AmazonCloudWatchRUM ReadOnlyAccess 策略。 添加了`rum:ListRumMetricsDestinations` 和`rum:BatchGetRumMetricsDefinitions` 權限，以便 CloudWatch RUM 可以將擴展指標發送到 CloudWatch 並顯而易見。	2022 年 10 月 27 日
AmazonCloudWatchRUMServiceRolePolicy — 現有政策的更新	CloudWatch RUM 向 AmazonCloudWatch RUM 添加了權限ServiceRolePolicy。 添加了`cloudwatch:PutMetricData` 權限，以便 CloudWatch RUM 可以將擴展指標發送到 CloudWatch.	2022 年 10 月 26 日
CloudWatchEvidentlyReadOnlyAccess – 更新現有政策	CloudWatch 明顯地添加了權限。CloudWatchEvidentlyReadOnlyAccess 已新增 `evidently:GetSegment` 、`evidently:ListSegments` 和`evidently:ListSegmentReferences` 許可，便於具有此政策的使用者可以看到已建立的 Evidently 受眾客群。	2022 年 8 月 12 日

變更	描述	日期
CloudWatchSyntheticsFullAccess – 更新現有政策	CloudWatch Synthetics 添加了權限。CloudWatchSyntheticsFullAccess 添加了`lambda:DeleteFunction` 和`lambda:DeleteLayerVersion` 權限，以便 CloudWatch Synthetics 可以在刪除初期測試時刪除相關資源。已新增 `iam:ListAttachedRolePolicies` ，以便客戶可以檢視連接到 Canary IAM 角色的政策。	2022 年 5 月 6 日
AmazonCloudWatch朗姆酒FullAccess-新政策	CloudWatch 新增新原則以啟用 CloudWatch RUM 的完整管理。 CloudWatch RUM 允許您對 Web 應用程序執行真實的用戶監視。如需詳細資訊，請參閱使用 CloudWatch 朗姆酒。	2021 年 11 月 29 日
AmazonCloudWatch朗姆酒ReadOnlyAccess-新政策	CloudWatch 已新增新原則以啟用 CloudWatch RUM 的唯讀存取權。 CloudWatch RUM 允許您對 Web 應用程序執行真實的用戶監視。如需詳細資訊，請參閱使用 CloudWatch 朗姆酒。	2021 年 11 月 29 日

變更	描述	日期
CloudWatchEvidentlyFullAccess – 新政策	CloudWatch 添加了一個新的政策，以實現 CloudWatch顯而易見的全面管理。 CloudWatch 顯然，您可以對 Web 應用程序執行 A/B 實驗，並逐步推出它們。如需詳細資訊，請參閱 CloudWatch 明顯地執行發射和 A/B 實驗。	2021 年 11 月 29 日
CloudWatchEvidentlyReadOnlyAccess – 新政策	CloudWatch 添加了一個新的策略，以啟用 CloudWatch Editional 的只讀訪問權限。 CloudWatch 顯然，您可以對 Web 應用程序執行 A/B 實驗，並逐步推出它們。如需詳細資訊，請參閱 CloudWatch 明顯地執行發射和 A/B 實驗。	2021 年 11 月 29 日
AWSServiceRoleForCloudWatchRUM— 新的受管理策略	CloudWatch 為新的服務連結角色新增政策，以允許 CloudWatch RUM 將監視資料發佈到其他相關 AWS 服務。	2021 年 11 月 29 日

變更	描述	日期
CloudWatchSyntheticsFullAccess – 更新現有政策	CloudWatch Synthetics 添加了權限 CloudWatchSyntheticsFullAccess，並且還改變了一個權限的範圍。 已新增`kms:ListAliases` 權限，讓使用者可以列出可用來加密初期測試成品的可用金 AWS KMS 鑰。已新增 `kms:DescribeKey` 許可，以便使用者可以看到金鑰的詳細資訊，該金鑰可用於加密 Canary 成品。已新增 `kms:Decrypt` 許可，讓使用者能夠解密 Canary 成品。此解密功能僅限於在 Simple Storage Service (Amazon S3) 儲存貯體內的資源上使用。 `s3:GetBucketLocation` 許可的 `Resource` 範圍已從 `*` 變更為 `arn:aws:s3:::*` 。	2021 年 9 月 29 日
CloudWatchSyntheticsFullAccess – 更新現有政策	CloudWatch Synthetics 添加了一個權限。CloudWatchSyntheticsFullAccess 新增 `lambda:UpdateFunctionCode` 許可，以便具有此政策的使用者可以變更 Canary 的執行時間版本。	2021 年 7 月 20 日

變更	描述	日期
AWSCloudWatchAlarms_ActionSSMIncidentsServiceRolePolicy— 新的受管理策略	CloudWatch 新增了新的受管 IAM 政策，以 CloudWatch 允許在事 AWS Systems Manager 件管理器中建立事件。	2021 年 5 月 10 日
CloudWatchAutomaticDashboardsAccess – 更新現有政策	CloudWatch 已新增CloudWatchAutomaticDashboardsAccess受管理原則的權限。此政策已新增`synthetics:DescribeCanaries LastRun` 權限，讓跨帳戶儀表板使用者能夠查看有關 CloudWatch Synthetics 料初期測試執行的詳細資料。	2021 年 4 月 20 日
CloudWatch 開始追蹤變更	CloudWatch 開始追蹤其 AWS 受管理策略的變更。	2021 年 4 月 14 日

使用條件鍵限制對 CloudWatch 命名空間的訪問

使用 IAM 條件金鑰限制使用者只能在您指定的 CloudWatch命名空間中發佈指標。

僅允許在一個命名空間中發佈

下列政策限制使用者只能在名為 `MyCustomNamespace` 的命名空間中發佈指標。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Resource": "*",
        "Action": "cloudwatch:PutMetricData",
        "Condition": {
            "StringEquals": {
                "cloudwatch:namespace": "MyCustomNamespace"
            }
```

```
        }
    }
}
```

從命名空間排除發佈

下列政策允許使用者在 CustomNamespace2 除外的任何命名空間中發佈指標。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": "cloudwatch:PutMetricData"
        },
        {
            "Effect": "Deny",
            "Resource": "*",
            "Action": "cloudwatch:PutMetricData",
            "Condition": {
                "StringEquals": {
                    "cloudwatch:namespace": "CustomNamespace2"
                }
            }
        }
    ]
}
```

使用條件索引鍵限制 Contributor Insights 使用者存取日誌群組

若要在 Contributor Insights 中建立規則並查看其結果，使用者必須擁有 `cloudwatch:PutInsightRule` 許可。根據預設，具有此權限的使用者可以建立 Contributor Insights 規則，以評估 CloudWatch 記錄中的任何記錄群組，然後查看結果。結果可能包含這些日誌群組的參與者資料。

您可以建立具有條件索引鍵的 IAM 政策，以授予使用者為某些記錄群組撰寫 Contributor Insights 規則的許可，同時防止他們為其他日誌群組撰寫規則和查看此資料。

如需 IAM 政策中的 `Condition` 元素的詳細資訊，請參閱 IAM JSON 政策元素：Condition。

僅允許存取特定日誌群組的寫入規則和檢視結果

下列政策允許使用者存取寫入規則，並檢視名為 AllowedLogGroup 的日誌群組和名稱開頭為 AllowedWildCard 的所有日誌群組的結果。它不會授予寫入規則或檢視任何其他日誌群組的規則結果的存取權。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCertainLogGroups",
            "Effect": "Allow",
            "Action": "cloudwatch:PutInsightRule",
            "Resource": "arn:aws:cloudwatch:*:*:insight-rule/*",
            "Condition": {
                "ForAllValues:StringEqualsIgnoreCase": {
                    "cloudwatch:requestInsightRuleLogGroups": [
                        "AllowedLogGroup",
                        "AllowedWildcard*"
                    ]
                }
            }
        }
    ]
}
```

拒絕特定日誌群組的寫入規則，但允許所有其他日誌群組的寫入規則

下列政策會明確拒絕使用者存取寫入規則，並檢視名為 ExplicitlyDeniedLogGroup 的日誌群組，但允許為所有其他日誌群組的寫入規則和檢視規則結果。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowInsightRulesOnLogGroupsByDefault",
            "Effect": "Allow",
            "Action": "cloudwatch:PutInsightRule",
            "Resource": "arn:aws:cloudwatch:*:*:insight-rule/*"

        },
        {
            "Sid": "ExplicitDenySomeLogGroups",
            "Effect": "Deny",
            "Action": "cloudwatch:PutInsightRule",
```

```
            "Resource": "arn:aws:cloudwatch:*:*:insight-rule/*",
            "Condition": {
                "ForAllValues:StringEqualsIgnoreCase": {
                    "cloudwatch:requestInsightRuleLogGroups": [
                        "/test/alpine/ExplicitlyDeniedLogGroup"
                    ]
                }
            }
        }
    ]
}
```

使用條件索引鍵限制警示動作

當 CloudWatch 警示狀態變更時，它們可以執行不同的動作，例如停止和終止 EC2 執行個體，以及執行 Systems Manager 動作。當警示變更為任何狀態時，即可啟動這些動作，包括「ALARM」、「OK」或「INSUFFICIENT_DATA」。

使用 `cloudwatch:AlarmActions` 條件索引鍵，允許使用者建立警示，其中這些警示只能在警示狀態變更時執行您指定的動作。例如，您可以允許使用者建立只能執行非 EC2 動作之動作的警示。

允許使用者建立只能傳送 Amazon SNS 通知或執行 Systems Manager 動作的警示

下列政策限制使用者建立只能傳送 Amazon SNS 通知和執行 Systems Manager 動作的警示。使用者無法建立執行 EC2 動作的警示。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateAlarmsThatCanPerformOnlySNSandSSMActions",
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricAlarm",
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringLike": {
                    "cloudwatch:AlarmActions": [
                        "arn:aws:sns:*",
                        "arn:aws:ssm:*"
                    ]
                }
            }
        }
```

```
    ]
}
```

使用 CloudWatch 的服務連結角色

Amazon CloudWatch 使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結到 CloudWatch的唯一 IAM 角色類型。服務連結角色由預先定義，CloudWatch 並包含服務代表您呼叫其他服 AWS 務所需的所有權限。

中的一個服務連結角色 CloudWatch 可設定可終止、停止或重新啟動 Amazon EC2 執行個體的 CloudWatch 警示，而不需要您手動新增必要的許可。另一個服務連結角色可讓監控帳戶從您指定的其他帳戶存取 CloudWatch資料，以建立跨帳戶跨區域儀表板。

CloudWatch 定義這些服務連結角色的權限，除非另有定義，否則只 CloudWatch 能擔任該角色。定義的許可包括信任政策和許可政策，並且該許可政策不能附加到任何其他 IAM 實體。

您必須先刪除角色的相關資源，才能刪除角色。這項限制可保護您的 CloudWatch 資源，因為您無法不小心移除存取資源的權限。

如需關於支援服務連結角色的其他服務的資訊，請參閱可搭配 IAM 運作的AWS 服務，並尋找 Service-Linked Role (服務連結角色) 欄顯示為 Yes (是) 的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

用於 CloudWatch 警示 EC2 動作的服務連結角色許可

CloudWatch 使用名為的服務連結角色 AWSServiceRoleForCloudWatchEvents— CloudWatch 使用此服務連結角色執行 Amazon EC2 警示動作。

服 AWSServiceRoleForCloudWatchEvents 務連結角色會信任 E CloudWatch vents 服務擔任該角色。CloudWatch 事件會在警示呼叫時叫用終止、停止或重新啟動執行個體動作。

AWSServiceRoleForCloudWatchEvents 服務連結角色許可政策允許 CloudWatch 事件在 Amazon EC2 執行個體上完成下列動作：

- `ec2:StopInstances`
- `ec2:TerminateInstances`
- `ec2:RecoverInstances`
- `ec2:DescribeInstanceRecoveryAttribute`
- `ec2:DescribeInstances`
- `ec2:DescribeInstanceStatus`

AWSServiceRoleForCloudWatchCrossAccount服務連結角色權限原則 CloudWatch 允許完成下列動作：

- sts:AssumeRole

CloudWatch 應用程式訊號的服務連結角色權限

CloudWatch 應用程式信號使用名為的服務連結角色 AWSServiceRoleForCloudWatchApplicationSignals— CloudWatch 使用此服務連結角色來收集記 CloudWatch 錄資料、X-Ray 追蹤資料、 CloudWatch 指標資料，以及標記您已啟用應用程式訊號的應用 CloudWatch 程式中的資料。

AWSServiceRoleForCloudWatchApplicationSignals服務連結角色會信任 CloudWatch 應用程式信號擔任該角色。Application Signals 會從您的帳戶中收集日誌、追蹤、指標和標記資料。

AWSServiceRoleForCloudWatchApplicationSignals已附加 IAM 政策，並命名為此政策CloudWatchApplicationSignalsServiceRolePolicy。本政策授予 CloudWatch 應用程式信號從其他相關 AWS 服務收集監控和標記資料的權限。它包含允許 Application Signals 完成下列動作的許可：

- xray:GetServiceGraph
- logs:StartQuery
- logs:GetQueryResults
- cloudwatch:GetMetricData
- cloudwatch:ListMetrics
- tag:GetResources

的完整內容CloudWatchApplicationSignalsServiceRolePolicy如下：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "XRayPermission",
            "Effect": "Allow",
            "Action": [
                "xray:GetServiceGraph"
            ],
            "Resource": [
                "*"
```

```
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
            "Sid": "CWLogsPermission",
            "Effect": "Allow",
            "Action": [
                "logs:StartQuery",
                "logs:GetQueryResults"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws/appsignals/*:*",
                "arn:aws:logs:*:*:log-group:/aws/application-signals/data:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
            "Sid": "CWListMetricsPermission",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:ListMetrics"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        },
        {
            "Sid": "CWGetMetricDataPermission",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData"
```

```
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "TagsPermission",
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "${aws:PrincipalAccount}"
                }
            }
        }
    ]
}
```

CloudWatch 警示 Systems Manager OpsCenter 動作的服務連結角色權限

CloudWatch 使用名為的服務連結角色 AWSServiceRoleForCloudWatchAlarms_ActionSSM— 當 CloudWatch 警示進入 ALARM 狀態時， CloudWatch會使用此服務連結角色執行 Systems Manager OpsCenter 動作。

服 AWSServiceRoleForCloudWatchAlarms_ActionSSM 務連結角色會信任CloudWatch 服務擔任該角色。 CloudWatch 警示會在警示呼叫時叫用「Systems Manager OpsCenter 」動作。

AWSServiceRoleForCloudWatchAlarms_ActionSSM服務連結角色權限原則可讓 Systems Manager 完成下列動作：

- `ssm:CreateOpsItem`

CloudWatch 警示系統管理員事件管理員動作的服務連結角色權限

CloudWatch 使用名為的服務連結角色 AWSServiceRoleForCloudWatchAlarms_ActionSSMIncidents— 當 CloudWatch 警示進入 ALARM 狀態時， CloudWatch 會使用此服務連結角色啟動事件管理員事件。

服AWSServiceRoleForCloudWatchAlarms_ActionSSMIncidents務連結角色會信任 CloudWatch 服務擔任該角色。 CloudWatch 警示會在警示呼叫時叫用「Systems Manager 管理員事件管理員」動作。

AWSServiceRoleForCloudWatchAlarms_ActionSSMIncidents服務連結角色權限原則可讓 Systems Manager 完成下列動作：

- `ssm-incidents:StartIncident`

跨帳戶 CloudWatch 跨區域的服務連結角色權限

CloudWatch 使用名為的服務連結角色 AWSServiceRoleForCloudWatchCrossAccount— CloudWatch 使用此角色存取您指定之其他 AWS 帳戶中的 CloudWatch 資料。SLR 僅提供假定角色權限，以允許 CloudWatch 服務承擔共用帳戶中的角色。它是提供資料存取的共享角色。

AWSServiceRoleForCloudWatchCrossAccount服務連結角色權限原則 CloudWatch 允許完成下列動作：

- `sts:AssumeRole`

服AWSServiceRoleForCloudWatchCrossAccount務連結角色會信任 CloudWatch 服務擔任該角色。

CloudWatch 資料庫 Performance Insights 的服務連結角色權限

CloudWatch 使用名為AWSServiceRoleForCloudWatchMetrics_DbPerfInsights的服務連結角色。— CloudWatch 使用此角色擷取 Performance Insights 指標，以建立警示和快照。

AWSServiceRoleForCloudWatchMetrics_DbPerfInsights服務連結角色已附加 `AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsServiceRolePolicy` IAM 政策。該政策的內容如下：

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "pi:GetResourceMetrics"
   ],
   "Resource": "*",
   "Condition": {
```

```
    "StringEquals": {
     "aws:ResourceAccount": "${aws:PrincipalAccount}"
    }
   }
  }
 ]
}
```

服AWSServiceRoleForCloudWatchMetrics_DbPerfInsights務連結角色會信任 CloudWatch 服務擔任該角色。

建立服務連結角色 CloudWatch

您不需要手動建立任何這些服務連結角色。當您第一次在 IAM CLI 或 IAM API 中 AWS Management Console建立警示時，CloudWatch 會 AWSServiceRoleForCloudWatchEvents AWSServiceRoleForCloudWatchAlarms_ActionSSM為您建立警示。

第一次啟用服務和拓撲探索時，應用程式信號會AWSServiceRoleForCloudWatchApplicationSignals為您建立。

當您第一次啟用帳戶成為跨帳戶跨區域功能的監控帳戶時， CloudWatch 會AWSServiceRoleForCloudWatchCrossAccount為您建立。

當您第一次建立使用DB_PERF_INSIGHTS公制數學函數的鬧鐘時， CloudWatch 會AWSServiceRoleForCloudWatchMetrics_DbPerfInsights為您建立。

如需詳細資訊，請參閱 IAM 使用者指南中的建立服務連結角色。

編輯下列項目的服務連結角色 CloudWatch

CloudWatch 不允許您編輯AWSServiceRoleForCloudWatchEvents、AWSServiceRoleForCloudWatchAlarms_ActionSSMAWSServic或AWSServiceRoleForCloudWatchMetrics_DbPerfInsights角色。在您建立這些角色之後，您便無法變更其名稱，因為各種實體可能會參考這些角色。然而，您可使用 IAM 來編輯這些角色描述。

編輯服務連結角色說明 (IAM 主控台)

您可以使用 IAM 主控台來編輯服務連結角色的說明。

編輯服務連結角色的說明 (主控台)

1. 在 IAM 主控台的導覽窗格中，選擇 Roles (角色)。

2. 選擇要修改之角色的名稱。
3. 在 Role description (角色說明) 的最右邊，選擇 Edit (編輯)。
4. 在方塊中鍵入新的說明，然後選擇 Save (儲存)。

編輯服務連結角色描述 (AWS CLI)

您可以使用中的 IAM 命令 AWS Command Line Interface 來編輯服務連結角色的說明。

變更服務連結角色的說明 (AWS CLI)

1. (選用) 若要檢視角色的目前說明，請使用下列命令：

   ```
   $ aws iam get-role --role-name role-name
   ```

   透過 AWS CLI 命令，使用角色名稱 (而非 ARN) 來參照角色。例如，如果角色具有下列 ARN：`arn:aws:iam::123456789012:role/myrole`，請將角色參照為 **myrole**。
2. 若要更新服務連結角色的說明，請使用下列命令：

   ```
   $ aws iam update-role-description --role-name role-name --description description
   ```

編輯服務連結角色說明 (IAM API)

您可以使用 IAM API 來編輯服務連結角色的說明。

變更服務連結角色的說明 (API)

1. (選用) 若要檢視角色的目前說明，請使用下列命令：

   GetRole
2. 若要更新角色的說明，請使用下列命令：

   UpdateRoleDescription

刪除下列項目的服務連結角色 CloudWatch

如果您不再有會自動停止、終止或重新啟動 EC2 執行個體的警示，建議您刪除該 AWSServiceRoleForCloudWatchEvents角色。

如果您不再有執行「Systems Manager OpsCenter 」動作的警示，建議您刪除 AWSServiceRoleForCloudWatchAlarms_ActionSSM 角色。

如果您刪除所有使用DB_PERF_INSIGHTS量度數學函數的警示，建議您刪除AWSServiceRoleForCloudWatchMetrics_DbPerfInsights服務連結的角色。

如此一來，您就沒有未主動監控或維護的未使用實體。然而，務必清除您的服務連結角色，之後才能將其刪除。

清除服務連結角色

您必須先確認服務連結角色沒有作用中的工作階段，並移除該角色使用的資源，之後才能使用 IAM 將其刪除。

檢查服務連結角色是否於 IAM 主控台有作用中的工作階段

1. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇角色。選擇 AWSServiceRoleForCloudWatchEvents 角色的名稱 (不是核取方塊)。
3. 在 Summary (摘要) 頁面上，針對所選角色選擇 Access Advisor (存取 Advisor)，然後檢閱服務連結角色的近期活動。

> **Note**
>
> 如果您不確定 CloudWatch 是否正在使用 AWSServiceRoleForCloudWatchEvents角色，請嘗試刪除角色。如果服務正在使用該角色，則刪除會失敗，而您可以檢視正在使用該角色的 區域。如果服務正在使用該角色，您必須先等到工作階段結束，才能刪除該角色。您無法撤銷服務連結角色的工作階段。

刪除服務連結角色 (IAM 主控台)

您可以使用 IAM 主控台刪除服務連結角色。

刪除服務連結角色 (主控台)

1. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇角色。選擇您要刪除的角色名稱旁的核取方塊，而非名稱或資料列本身。
3. 對於 Role actions (角色動作)，選擇 Delete role (刪除角色)。

4. 在確認對話方塊中，檢閱服務上次存取資料，以顯示每個所選取角色上次存取 AWS 服務的時間。這可協助您確認角色目前是否作用中。若要繼續，請選擇 Yes, Delete (是，刪除)。
5. 查看 IAM 主控台通知，監視服務連結角色刪除的進度。因為 IAM 服務連結角色刪除不同步，所以在您提交角色進行刪除之後，刪除任務可能會成功或失敗。如果任務失敗，從通知中選擇 View details (檢視詳細資訊) 或 View Resources (檢視資源)，以了解刪除失敗原因。如果刪除因角色使用服務中資源而失敗，則失敗原因會包含資源清單。

刪除服務連結角色 (AWS CLI)

您可以使用的 IAM 命令 AWS Command Line Interface 來刪除服務連結角色。

刪除服務連結角色 (AWS CLI)

1. 因為無法刪除正在使用或具有相關聯資源的服務連結角色，所以您必須提交刪除要求。如果不符合這些條件，則可以拒絕該請求。您必須從回應中擷取 `deletion-task-id`，以檢查刪除任務的狀態。鍵入下列命令，以提交服務連結角色刪除要求：

   ```
   $ aws iam delete-service-linked-role --role-name service-linked-role-name
   ```

2. 鍵入下列命令，以檢查刪除任務的狀態：

   ```
   $ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
   ```

   刪除任務的狀態可以是 `NOT_STARTED`、`IN_PROGRESS`、`SUCCEEDED` 或 `FAILED`。如果刪除失敗，則呼叫會傳回失敗原因，以進行疑難排解。

刪除服務連結角色 (IAM API)

您可以使用 IAM API 刪除服務連結角色。

刪除服務連結角色 (API)

1. 若要提交服務連結角色的刪除要求，請致電DeleteServiceLinkedRole。在要求中，指定您要刪除的角色名稱。

   因為無法刪除正在使用或具有相關聯資源的服務連結角色，所以您必須提交刪除要求。如果不符合這些條件，則可以拒絕該請求。您必須從回應中擷取 `DeletionTaskId`，以檢查刪除任務的狀態。

2. 要檢查刪除狀態，請致電GetServiceLinkedRoleDeletionStatus。在請求中，指定 `DeletionTaskId`。

   刪除任務的狀態可以是 `NOT_STARTED`、`IN_PROGRESS`、`SUCCEEDED` 或 `FAILED`。如果刪除失敗，則呼叫會傳回失敗原因，以進行疑難排解。

CloudWatch AWS 服務連結角色的更新

檢視 CloudWatch 自此服務開始追蹤這些變更以來的 AWS 受管理策略更新詳細資料。如需有關此頁面變更的自動警示，請訂閱「 CloudWatch 文件歷史記錄」頁面上的 RSS 摘要。

變更	描述	日期
AWSServiceRoleForCloudWatchApplicationSignals- 更新服務鏈接角色策略的權限	CloudWatch 將更多記錄群組新增至此角色所授`logs:StartQuery` 與的`logs:GetQueryResults` 權限範圍。	2024年4月24日
AWSServiceRoleForCloudWatchApplicationSignals — 新的服務連結角色	CloudWatch 新增這個新的服務連結角色，可讓CloudWatch應用程式信號收集 CloudWatch 記錄資料、X-Ray 追蹤資料、CloudWatch 指標資料，以及標記您已針對應用程式啟用「應用程式訊號」的應用 CloudWatch 程式資料。	2023 年 11 月 9 日
AWSServiceRoleForCloudWatchMetrics_DbPerfInsights— 新的服務連結角色	CloudWatch 新增這個新的服務連結角色，以允許擷 CloudWatch 取 Performance Insights 指標，以進行警報和快照。IAM 政策已附加至此角色，而且該政策授予代表您擷	2023 年 9 月 13 日

變更	描述	日期
	CloudWatch 取 Performance Insights 指標的權限。	
AWSServiceRoleForCloudWatchAlarms_ActionSSMIncidents— 新的服務連結角色	CloudWatch 新增服務連結角色，以允許 CloudWatch 在事 AWS Systems Manager 件管理員中建立事件。	2021 年 4 月 26 日
CloudWatch 開始追蹤變更	CloudWatch 開始追蹤其服務連結角色的變更。	2021 年 4 月 26 日

針 CloudWatch對 RUM 使用服務連結角色

CloudWatch RUM 使用 AWS Identity and Access Management (IAM) 服務連結的角色。服務連結角色是直接連結至 RUM 的一種特殊 IAM 角色類型。服務連結角色由 RUM 預先定義，包含服務代表您呼叫其他 AWS 服務所需的所有權限。

RUM 會定義此服務連結角色的許可，除非另外定義，否則只有 RUM 才能擔任此角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

您必須先刪除與角色相關的資源，才能刪除角色。此限制可保護您的 RUM 資源，避免您不小心移除資源的存取許可。

如需關於支援服務連結角色的其他服務資訊，請參閱《可搭配 IAM 運作的AWS 服務》，尋找服務連結角色欄中顯示為是的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

RUM 的服務連結角色許可：

RUM 使用名為的服務連結角色 AWSServiceRoleForCloudWatchRUM— 此角色可讓 RUM 將追 AWS X-Ray 蹤資料傳送至您的帳戶，以供您啟用 X-Ray 追蹤的應用程式監視器。

服AWSServiceRoleForCloudWatchRUM務連結角色會信任 X-Ray 服務擔任該角色。X-Ray 會將追蹤資料傳送到您的帳戶。

AWSServiceRoleForCloudWatchRUM服務連結角色具有附加名為 AmazonCloudWatchRUM ServiceRolePolicy 的 IAM 政策。此政策授予 CloudWatch RUM 將監視資料發佈至其他相關 AWS 服務的權限。它包含允許 RUM 完成下列動作的許可：

- xray:PutTraceSegments
- cloudwatch:PutMetricData

AmazonCloudWatch朗姆酒的完整內容ServiceRolePolicy如下。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "xray:PutTraceSegments"
   ],
   "Resource": [
    "*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": "cloudwatch:PutMetricData",
   "Resource": "*",
   "Condition": {
    "StringLike": {
     "cloudwatch:namespace": [
      "RUM/CustomMetrics/*",
      "AWS/RUM"
     ]
    }
   }
  }
 ]
}
```

為 RUM 建立服務連結角色

您不需要手動建立 CloudWatch RUM 的服務連結角色。當您第一次建立已啟用 X-Ray 追蹤的應用程式監視器，或更新應用程式監視器以使用 X-Ray 追蹤時，RUM 會AWSServiceRoleForCloudWatchRUM為您建立。

如需詳細資訊，請參閱 IAM 使用者指南中的建立服務連結角色。

為 RUM 編輯服務連結角色

CloudWatch RUM 不允許您編輯AWSServiceRoleForCloudWatchRUM角色。在您建立這些角色之後，您便無法變更其名稱，因為各種實體可能會參考這些角色。然而，您可使用 IAM 來編輯這些角色描述。

編輯服務連結角色說明 (IAM 主控台)

您可以使用 IAM 主控台來編輯服務連結角色的說明。

編輯服務連結角色的說明 (主控台)

1. 在 IAM 主控台的導覽窗格中，選擇 Roles (角色)。
2. 選擇要修改之角色的名稱。
3. 在 Role description (角色說明) 的最右邊，選擇 Edit (編輯)。
4. 在方塊中鍵入新的說明，然後選擇 Save (儲存)。

編輯服務連結角色描述 (AWS CLI)

您可以使用中的 IAM 命令 AWS Command Line Interface 來編輯服務連結角色的說明。

變更服務連結角色的說明 (AWS CLI)

1. (選用) 若要檢視角色的目前說明，請使用下列命令：

   ```
   $ aws iam get-role --role-name role-name
   ```

   透過 AWS CLI 命令，使用角色名稱 (而非 ARN) 來參照角色。例如，如果角色具有下列 ARN：`arn:aws:iam::123456789012:role/myrole`，請將角色參照為 **`myrole`**。
2. 若要更新服務連結角色的說明，請使用下列命令：

   ```
   $ aws iam update-role-description --role-name role-name --description description
   ```

編輯服務連結角色說明 (IAM API)

您可以使用 IAM API 來編輯服務連結角色的說明。

變更服務連結角色的說明 (API)

1. (選用) 若要檢視角色的目前說明，請使用下列命令：

   GetRole
2. 若要更新角色的說明，請使用下列命令：

   UpdateRoleDescription

為 RUM 刪除服務連結角色

如果您不再啟用 X-Ray 的應用程式監視器，建議您刪除該AWSServiceRoleForCloudWatchRUM角色。

如此一來，您就沒有未主動監控或維護的未使用實體。然而，務必清除您的服務連結角色，之後才能將其刪除。

清除服務連結角色

您必須先確認服務連結角色沒有作用中的工作階段，並移除該角色使用的資源，之後才能使用 IAM 將其刪除。

檢查服務連結角色是否於 IAM 主控台有作用中的工作階段

1. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇角色。選擇AWSServiceRoleForCloudWatchRUM角色的名稱 (不是核取方塊)。
3. 在 Summary (摘要) 頁面上，針對所選角色選擇 Access Advisor (存取 Advisor)，然後檢閱服務連結角色的近期活動。

> **Note**
>
> 如果您不確定 RUM 是否正在使用此AWSServiceRoleForCloudWatchRUM角色，請嘗試刪除角色。如果服務正在使用該角色，則刪除會失敗，而您可以檢視正在使用該角色的 區域。如果服務正在使用該角色，您必須先等到工作階段結束，才能刪除該角色。您無法撤銷服務連結角色的工作階段。

刪除服務連結角色 (IAM 主控台)

您可以使用 IAM 主控台刪除服務連結角色。

刪除服務連結角色 (主控台)

1. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇角色。選擇您要刪除的角色名稱旁的核取方塊，而非名稱或資料列本身。
3. 對於 Role actions (角色動作)，選擇 Delete role (刪除角色)。
4. 在確認對話方塊中，檢閱服務上次存取資料，以顯示每個所選取角色上次存取 AWS 服務的時間。這可協助您確認角色目前是否作用中。若要繼續，請選擇 Yes, Delete (是，刪除)。
5. 查看 IAM 主控台通知，監視服務連結角色刪除的進度。因為 IAM 服務連結角色刪除不同步，所以在您提交角色進行刪除之後，刪除任務可能會成功或失敗。如果任務失敗，從通知中選擇 View details (檢視詳細資訊) 或 View Resources (檢視資源)，以了解刪除失敗原因。如果刪除因角色使用服務中資源而失敗，則失敗原因會包含資源清單。

刪除服務連結角色 (AWS CLI)

您可以使用的 IAM 命令 AWS Command Line Interface 來刪除服務連結角色。

刪除服務連結角色 (AWS CLI)

1. 因為無法刪除正在使用或具有相關聯資源的服務連結角色，所以您必須提交刪除要求。如果不符合這些條件，則可以拒絕該請求。您必須從回應中擷取 `deletion-task-id`，以檢查刪除任務的狀態。鍵入下列命令，以提交服務連結角色刪除要求：

   ```
   $ aws iam delete-service-linked-role --role-name service-linked-role-name
   ```

2. 鍵入下列命令，以檢查刪除任務的狀態：

   ```
   $ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
   ```

   刪除任務的狀態可以是 `NOT_STARTED`、`IN_PROGRESS`、`SUCCEEDED` 或 `FAILED`。如果刪除失敗，則呼叫會傳回失敗原因，以進行疑難排解。

刪除服務連結角色 (IAM API)

您可以使用 IAM API 刪除服務連結角色。

刪除服務連結角色 (API)

1. 若要提交服務連結角色的刪除要求，請致電DeleteServiceLinkedRole。在要求中，指定您要刪除的角色名稱。

   因為無法刪除正在使用或具有相關聯資源的服務連結角色，所以您必須提交刪除要求。如果不符合這些條件，則可以拒絕該請求。您必須從回應中擷取 `DeletionTaskId`，以檢查刪除任務的狀態。
2. 要檢查刪除狀態，請致電GetServiceLinkedRoleDeletionStatus。在請求中，指定 `DeletionTaskId`。

   刪除任務的狀態可以是 `NOT_STARTED`、`IN_PROGRESS`、`SUCCEEDED` 或 `FAILED`。如果刪除失敗，則呼叫會傳回失敗原因，以進行疑難排解。

將服務連結角色用於 CloudWatch 應用程式深入

CloudWatch 應用程式深入解析使用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 CloudWatch 應用程式深入解析的唯一 IAM 角色類型。服務連結角色由 Ap CloudWatch plication Insights 預先定義，並包含服務代表您呼叫其他服 AWS 務所需的所有權限。

服務連結角色可讓您輕鬆設定 CloudWatch 應用程式深入解析，因為您不需要手動新增必要的權限。CloudWatch 應用程式深入解析會定義其服務連結角色的權限，除非另有定義，否則只有 CloudWatch 應用程式深入解析可以擔任其角色。定義的許可包括信任政策和許可政策，並且該許可政策不能連接到任何其他 IAM 實體。

如需關於支援服務連結角色的其他服務的資訊，請參閱可搭配 IAM 運作的AWS 服務，並尋找 Service-Linked Role (服務連結角色) 欄顯示為 Yes (是) 的服務。選擇 Yes (是) 連結，檢視該服務的服務連結角色文件。

CloudWatch 應用程式見解的服務連結角色權限

CloudWatch 應用程式洞察會使用名為AWSServiceRoleForApplicationInsights的服務連結角色。Application Insights 使用此角色執行作業，例如分析客戶的資源群組、建立 CloudFormation 堆疊以在指標上建立警示，以及在 EC2 執行個體上設定 CloudWatch 代理程式。此服務連結角色連接了一個名為 `CloudwatchApplicationInsightsServiceLinkedRolePolicy` 的 IAM 政策。如需更新此政策，請參閱「AWS 受管政策的 Application Insights 更新」。

角色權限原則可讓 CloudWatch 應用程式深入解析在資源上完成下列動作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:GetMetricData",
        "cloudwatch:ListMetrics",
        "cloudwatch:PutMetricAlarm",
        "cloudwatch:DeleteAlarms",
        "cloudwatch:PutAnomalyDetector",
        "cloudwatch:DeleteAnomalyDetector",
        "cloudwatch:DescribeAnomalyDetectors"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:FilterLogEvents",
        "logs:GetLogEvents",
        "logs:DescribeLogStreams",
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:DescribeRule"
      ],
      "Resource": [
        "*"
      ]
    },
    {
```

```
      "Effect": "Allow",
      "Action": [
        "cloudFormation:CreateStack",
        "cloudFormation:UpdateStack",
        "cloudFormation:DeleteStack",
        "cloudFormation:DescribeStackResources"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:*:stack/ApplicationInsights-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudFormation:DescribeStacks",
        "cloudFormation:ListStackResources",
        "cloudFormation:ListStacks"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "tag:GetResources"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "resource-groups:ListGroupResources",
        "resource-groups:GetGroupQuery",
        "resource-groups:GetGroup"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
```

```
      "Action": [
        "resource-groups:CreateGroup",
        "resource-groups:DeleteGroup"
      ],
      "Resource": [
        "arn:aws:resource-groups:*:*:group/ApplicationInsights-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticloadbalancing:DescribeLoadBalancers",
        "elasticloadbalancing:DescribeTargetGroups",
        "elasticloadbalancing:DescribeTargetHealth"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:DescribeAutoScalingGroups"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:PutParameter",
        "ssm:DeleteParameter",
        "ssm:AddTagsToResource",
        "ssm:RemoveTagsFromResource",
        "ssm:GetParameters"
      ],
      "Resource": "arn:aws:ssm:*:*:parameter/AmazonCloudWatch-ApplicationInsights-*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:CreateAssociation",
        "ssm:UpdateAssociation",
```

```
        "ssm:DeleteAssociation",
        "ssm:DescribeAssociation"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:instance/*",
        "arn:aws:ssm:*:*:association/*",
        "arn:aws:ssm:*:*:managed-instance/*",
        "arn:aws:ssm:*:*:document/AWSEC2-
ApplicationInsightsCloudwatchAgentInstallAndConfigure",
        "arn:aws:ssm:*:*:document/AWS-ConfigureAWSPackage",
        "arn:aws:ssm:*:*:document/AmazonCloudWatch-ManageAgent"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetOpsItem",
        "ssm:CreateOpsItem",
        "ssm:DescribeOpsItems",
        "ssm:UpdateOpsItem",
        "ssm:DescribeInstanceInformation"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:AddTagsToResource"
      ],
      "Resource": "arn:aws:ssm:*:*:opsitem/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:ListCommandInvocations",
        "ssm:GetCommandInvocation"
      ],
      "Resource": [
        "*"
      ]
    },
    {
```

```
      "Effect": "Allow",
      "Action": "ssm:SendCommand",
      "Resource": [
        "arn:aws:ec2:*:*:instance/*",
        "arn:aws:ssm:*:*:document/AWSEC2-CheckPerformanceCounterSets",
        "arn:aws:ssm:*:*:document/AWS-ConfigureAWSPackage",
        "arn:aws:ssm:*:*:document/AWSEC2-DetectWorkload",
        "arn:aws:ssm:*:*:document/AmazonCloudWatch-ManageAgent"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeVolumes",
        "ec2:DescribeVolumeStatus",
        "ec2:DescribeVpcs",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeNatGateways"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "rds:DescribeDBInstances",
        "rds:DescribeDBClusters"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "lambda:ListFunctions",
        "lambda:GetFunctionConfiguration",
        "lambda:ListEventSourceMappings"
      ],
      "Resource": [
        "*"
      ]
```

```
    },
    {
      "Effect": "Allow",
      "Action": [
        "events:PutRule",
        "events:PutTargets",
        "events:RemoveTargets",
        "events:DeleteRule"
      ],
      "Resource": [
        "arn:aws:events:*:*:rule/AmazonCloudWatch-ApplicationInsights-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "xray:GetServiceGraph",
        "xray:GetTraceSummaries",
        "xray:GetTimeSeriesServiceStatistics",
        "xray:GetTraceGraph"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:ListTables",
        "dynamodb:DescribeTable",
        "dynamodb:DescribeContributorInsights",
        "dynamodb:DescribeTimeToLive"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "application-autoscaling:DescribeScalableTargets"
      ],
      "Resource": [
        "*"
```

```
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets",
        "s3:GetMetricsConfiguration",
        "s3:GetReplicationConfiguration"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "states:ListStateMachines",
        "states:DescribeExecution",
        "states:DescribeStateMachine",
        "states:GetExecutionHistory"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GET"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ecs:DescribeClusters",
        "ecs:DescribeContainerInstances",
        "ecs:DescribeServices",
        "ecs:DescribeTaskDefinition",
        "ecs:DescribeTasks",
        "ecs:DescribeTaskSets",
        "ecs:ListClusters",
```

```
        "ecs:ListContainerInstances",
        "ecs:ListServices",
        "ecs:ListTasks"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ecs:UpdateClusterSettings"
      ],
      "Resource": [
        "arn:aws:ecs:*:*:cluster/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "eks:DescribeCluster",
        "eks:DescribeFargateProfile",
        "eks:DescribeNodegroup",
        "eks:ListClusters",
        "eks:ListFargateProfiles",
        "eks:ListNodegroups",
        "fsx:DescribeFileSystems",
        "fsx:DescribeVolumes"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:GetSubscriptionAttributes",
        "sns:GetTopicAttributes",
        "sns:GetSMSAttributes",
        "sns:ListSubscriptionsByTopic",
        "sns:ListTopics"
      ],
      "Resource": [
        "*"
```

```
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ListQueues"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:DeleteSubscriptionFilter"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutSubscriptionFilter"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:*",
        "arn:aws:logs:*:*:destination:AmazonCloudWatch-ApplicationInsights-
LogIngestionDestination*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticfilesystem:DescribeFileSystems"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "route53:GetHostedZone",
        "route53:GetHealthCheck",
        "route53:ListHostedZones",
```

```
        "route53:ListHealthChecks",
        "route53:ListQueryLoggingConfigs"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "route53resolver:ListFirewallRuleGroupAssociations",
        "route53resolver:GetFirewallRuleGroup",
        "route53resolver:ListFirewallRuleGroups",
        "route53resolver:ListResolverEndpoints",
        "route53resolver:GetResolverQueryLogConfig",
        "route53resolver:ListResolverQueryLogConfigs",
        "route53resolver:ListResolverQueryLogConfigAssociations",
        "route53resolver:GetResolverEndpoint",
        "route53resolver:GetFirewallRuleGroupAssociation"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

建立 CloudWatch 應用程式深入解析的服務連結角色

您不需要手動建立一個服務連結角色。當您在中建立新的應用程式見解應用程式時 AWS Management Console， CloudWatch 應用程式深入解析會為您建立服務連結角色。

如果您刪除此服務連結角色後，又想再次建立，您可以使用相同程序在帳戶中重新建立角色。當您建立新的應用程式見解應用程式時， CloudWatch 應用程式深入解析會再次為您建立服務連結角色。

編輯 CloudWatch 應用程式深入解析的服務連結角色

CloudWatch 應用程式深入解析不允許您編輯 AWSServiceRoleForApplicationInsights 服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需更多資訊，請參閱 IAM 使用者指南中的編輯服務連結角色。

刪除 CloudWatch 應用程式深入解析的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，就不會有未主動監控或維護的未使用實體。不過，您必須先刪除 Application Insights 中的所有應用程式，才可以手動刪除該角色。

> **Note**
>
> 當您嘗試刪除資源時，如果 CloudWatch 應用程式深入解析服務正在使用該角色，則刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

若要刪除使用的 CloudWatch 應用程式見解資源 AWSServiceRoleForApplicationInsights

- 刪除所有應用程式見解 CloudWatch 應用程式。如需詳細資訊，請參閱應用程式深入解析使用者指南中的「刪除 CloudWatch 應用程式」。

使用 IAM 手動刪除服務連結角色

使用 IAM 主控台或 AWS API 刪除 AWSServiceRoleForApplicationInsights 服務連結角色。 AWS CLI 如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

CloudWatch 應用程式洞察服務連結角色的支援區域

CloudWatch 應用程式深入解析支援在所有提供服務的 AWS 區域中使用服務連結角色。如需詳細資訊，請參閱CloudWatch 應用程式洞察區域和端點。

AWS Amazon CloudWatch 應用程式洞察的受管政策

受 AWS 管理的策略是由建立和管理的獨立策略 AWS。 AWS 受管理的策略旨在為許多常見使用案例提供權限，以便您可以開始將權限指派給使用者、群組和角色。

請記住， AWS 受管理的政策可能不會為您的特定使用案例授與最低權限權限，因為這些權限可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的客戶管理政策，以便進一步減少許可。

您無法變更受 AWS 管理策略中定義的權限。如果 AWS 更新 AWS 受管理原則中定義的權限，則此更新會影響附加原則的所有主體識別 (使用者、群組和角色)。 AWS 當新的啟動或新 AWS 服務 的 API 操作可供現有服務使用時，最有可能更新 AWS 受管理策略。

如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

AWS 受管理的策略： CloudWatchApplicationInsightsFullAccess

您可將 CloudWatchApplicationInsightsFullAccess 政策連接到 IAM 身分。

此政策授予允許完整存取 Application Insights 功能的管理許可。

許可詳細資訊

此政策包含以下許可。

- applicationinsights – 允許完整存取 Application Insights 功能。
- iam— 允許應用程式深入解析建立服務連結的角色 AWSServiceRoleForApplicationInsights。這是必要的，以便 Application Insights 可以執行諸如分析客戶的資源群組、建立 CloudFormation 堆疊以在指標上建立警示，以及在 EC2 執行個體上設定 CloudWatch代理程式等作業。如需詳細資訊，請參閱 將服務連結角色用於 CloudWatch 應用程式深入。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "applicationinsights:*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeVolumes",
        "rds:DescribeDBInstances",
        "rds:DescribeDBClusters",
```

```
        "sqs:ListQueues",
        "elasticloadbalancing:DescribeLoadBalancers",
        "elasticloadbalancing:DescribeTargetGroups",
        "elasticloadbalancing:DescribeTargetHealth",
        "autoscaling:DescribeAutoScalingGroups",
        "lambda:ListFunctions",
        "dynamodb:ListTables",
        "s3:ListAllMyBuckets",
        "sns:ListTopics",
        "states:ListStateMachines",
        "apigateway:GET",
        "ecs:ListClusters",
        "ecs:DescribeTaskDefinition",
        "ecs:ListServices",
        "ecs:ListTasks",
        "eks:ListClusters",
        "eks:ListNodegroups",
        "fsx:DescribeFileSystems",
        "logs:DescribeLogGroups",
        "elasticfilesystem:DescribeFileSystems"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/aws-service-role/application-insights.amazonaws.com/
AWSServiceRoleForApplicationInsights"
      ],
      "Condition": {
        "StringEquals": {
          "iam:AWSServiceName": "application-insights.amazonaws.com"
        }
      }
    }
  ]
}
```

AWS 受管理的策略： CloudWatchApplicationInsightsReadOnlyAccess

您可將 CloudWatchApplicationInsightsReadOnlyAccess 政策連接到 IAM 身分。

此政策授予允許唯讀存取 Application Insights 所有功能的管理許可。

許可詳細資訊

此政策包含以下許可。

- applicationinsights – 允許唯讀存取 Application Insights 功能。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "applicationinsights:Describe*",
                "applicationinsights:List*"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS 受管理的策略： CloudwatchApplicationInsightsServiceLinkedRolePolicy

您無法附加 CloudwatchApplicationInsightsServiceLinkedRolePolicy 到 IAM 實體。此政策連結至服務連結角色，可讓 Application Insights 監控客戶資源。如需詳細資訊，請參閱 將服務連結角色用於 CloudWatch 應用程式深入。

AWS 受管政策的 Application Insights 更新

自此服務開始追蹤這些變更後，檢視應用程式見解 AWS 受管原則更新的詳細資料。如需有關此頁面變更的自動提醒，請訂閱 Application Insights Document history (Application Insights 文件歷程記錄) 頁面上的 RSS 摘要。

變更	描述	日期
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 更新現有政策	應用程式洞察新增了新的權限來列出 CloudFormation 堆疊。 Amazon CloudWatch 應用程式深入解析需要這些許可，才能分析和監控 CloudFormation 堆疊中巢狀的 AWS 資源。	2023 年 4 月 24 日
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 更新現有政策	Application Insights 已新增取得 Amazon VPC 和 Route 53 資源清單的新許可。 Amazon CloudWatch 應用程式洞察需要這些許可，才能自動設定最佳實務網路監控 Amazon CloudWatch。	2023 年 1 月 23 日
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 更新現有政策	Application Insights 新增了取得 SSM 命令叫用結果的新許可。 Amazon CloudWatch 應用程式洞察需要這些許可，才能自動偵測和監控在 Amazon EC2 執行個體上執行的工作負載。	2022 年 12 月 19 日
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 更新現有政策	Application Insights 已新增描述 Amazon VPC 和 Route 53 資源的新許可。 Amazon CloudWatch 應用程式洞察需要這些許可才能讀取客戶的 Amazon VPC 和 Route 53 資源組態，以及協助客戶自動設定最佳實務網路監控。Amazon CloudWatch	2022 年 12 月 19 日

變更	描述	日期
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 更新現有政策	Application Insights 新增了描述 EFS 資源的新許可。 Amazon CloudWatch 應用程式洞察需要這些許可才能讀取 Amazon EFS 客戶資源組態，以及協助客戶自動設定 EFS 監控的最佳實務 CloudWatch。	2022 年 10 月 3 日
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 更新現有政策	Application Insights 新增了描述 EFS 檔案系統的新許可。 Amazon CloudWatch 應用程式洞察需要這些許可，才能透過查詢帳戶中所有支援的資源來建立以帳戶為基礎的應用程式。	2022 年 10 月 3 日
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 更新現有政策	Application Insights 新增了擷取 FSx 資源相關資訊的新許可。 Amazon CloudWatch 應用程式洞察需要這些許可，才能擷取有關基礎 FSx 磁碟區的足夠資訊來監控工作負載。	2022 年 9 月 12 日
AWS 受管理的策略：CloudWatchApplicationInsightsFullAccess – 更新現有政策	Application Insights 新增了說明日誌群組的新許可。 Amazon CloudWatch 應用程式深入解析需要此許可，以確保在建立新應用程式時在帳戶中具有監控日誌群組的正確許可。	2022 年 1 月 24 日

變更	描述	日期
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 更新現有政策	應用程式深入解析新增權限來建立和刪除 CloudWatch 記錄訂閱篩選器。 Amazon CloudWatch 應用程式洞察需要這些許可才能建立訂閱篩選器，以便在已設定的應用程式中對資源進行日誌監控。	2022 年 1 月 24 日
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 更新現有政策	Application Insights 已新增許可，描述 Elastic Load Balancer 的目標群組和目標運作狀態。 Amazon CloudWatch 應用程式洞察需要這些許可，才能透過查詢帳戶中所有支援的資源來建立以帳戶為基礎的應用程式。	2021 年 11 月 4 日
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 更新現有政策	Application Insights 已新增在 Amazon EC2 執行個體上執行 `AmazonCloudWatch-ManageAgent` SSM 文件的許可。 Amazon CloudWatch 應用程式深入解析需要此許可，才能清除由應用程式洞察建立的 CloudWatch 代理程式組態檔案。	2021 年 9 月 30 日

變更	描述	日期
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 更新現有政策	Application Insights 已新增許可，以支援帳戶型應用程式監控，上載並監控您帳戶中所有支援的資源。 Amazon CloudWatch 應用程式洞察需要這些許可，才能查詢、標記資源和建立這些資源的群組。 Application Insights 已新增許可，以支援 SNS 主題的監控。 Amazon CloudWatch 應用程式深入解析需要這些許可，才能從 SNS 資源收集中繼資料，以設定 SNS 主題的監控。	2021 年 9 月 15 日
AWS 受管理的策略：CloudWatchApplicationInsightsFullAccess – 更新現有政策	Application Insights 已新增描述及列出支援資源的新許可。 Amazon CloudWatch 應用程式洞察需要這些許可，才能透過查詢帳戶中所有支援的資源來建立以帳戶為基礎的應用程式。	2021 年 9 月 15 日
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 更新現有政策	Application Insights 新增了描述 FSX 資源的新許可。 Amazon CloudWatch 應用程式洞察需要這些許可才能讀取客戶 FSx 資源組態，並協助客戶自動設定最佳實務 FSx 監控。CloudWatch	2021 年 8 月 31 日

變更	描述	日期
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 更新現有政策	Application Insights 新增了描述及列出 ECS 和 EKS 資源的新許可。 Amazon CloudWatch 應用程式深入解析需要此權限才能讀取客戶容器資源組態，並協助客戶自動設定最佳實務容器監控 CloudWatch。	2021 年 5 月 18 日
CloudwatchApplicationInsightsServiceLinkedRolePolicy – 更新現有政策	「應用程式深入解析」新增了新的權限， OpsCenter 以允許 OpsItems使用資源類型的資源上的`ssm:AddTagsToResource` 動作進行標記。`opsitem` 此權限是必需的 OpsCenter 。Amazon CloudWatch 應用程式洞察會建立， OpsItems 讓客戶能夠使用 AWS SSM OpsCenter 解決問題。	2021 年 4 月 13 日
Application Insights 開始追蹤變更	應用程式見解開始追蹤其 AWS 受管理原則的變更。	2021 年 4 月 13 日

Amazon CloudWatch 許可參考

下表列出每個 CloudWatch API 作業，以及您可授與執行動作之權限的對應動作。您在政策的 `Action` 欄位中指定動作，然後在政策的 `Resource` 欄位將萬用字元 (*) 指定為資源值。

您可以在 CloudWatch 原則中使用 AWS寬條件金鑰來表示條件。如需全金鑰的 AWS完整清單，請參閱 IAM 使用者指南中的AWS 全域和 IAM 條件內容金鑰。

> **Note**
>
> 若要指定動作，請使用 `cloudwatch:` 前綴，後面接著 API 操作名稱。例如：`cloudwatch:GetMetricData`、`cloudwatch:ListMetrics` 或 `cloudwatch:*` (意指所有 CloudWatch 動作)。

主題

- CloudWatch API 操作和動作所需的權限
- CloudWatch 參與者見解 API 作業和動作所需的權限
- CloudWatch 事件 API 作業和動作的必要權限
- CloudWatch 記錄 API 操作和動作所需的權限
- Amazon EC2 API 操作與動作所需的許可
- Amazon EC2 Auto Scaling API 操作與動作所需的許可

CloudWatch API 操作和動作所需的權限

CloudWatch API 作業	所需許可 (API 動作)
DeleteAlarms	`cloudwatch:DeleteAlarms` 需要刪除警示。
DeleteDashboards	`cloudwatch:DeleteDashboards` 需要刪除儀表板。
DeleteMetricStream	`cloudwatch:DeleteMetricStream` 需要刪除指標串流。
DescribeAlarmHistory	`cloudwatch:DescribeAlarmHistory` 需要查看警示歷史記錄。若要擷取複合警示的相關資訊，您的 `cloudwatch:Describ`

CloudWatch API 作業	所需許可 (API 動作)
	eAlarmHistory 許可必須具有 * 範圍。您無法傳回複合警示的相關資訊，如果您的 cloudwatch:DescribeAlarmHistory 許可的範圍較窄。
DescribeAlarms	cloudwatch:DescribeAlarms 需要擷取警示相關資訊。 若要擷取複合警示的相關資訊，您的 cloudwatch:DescribeAlarms 許可必須具有 * 範圍。您無法傳回複合警示的相關資訊，如果您的 cloudwatch:DescribeAlarms 許可的範圍較窄。
DescribeAlarmsForMetric	cloudwatch:DescribeAlarmsFo rMetric 需要查看指標的警示。
DisableAlarmActions	cloudwatch:DisableAlarmActions 需要停用警示動作。
EnableAlarmActions	cloudwatch:EnableAlarmActions 需要啟用警示動作。
GetDashboard	cloudwatch:GetDashboard 需要顯示現有儀表板的相關資料。

CloudWatch API 作業	所需許可 (API 動作)
GetMetricData	`cloudwatch:GetMetricData` 需要在 CloudWatch 主控台中繪製度量資料圖形、擷取大量指標資料，以及對該資料執行度量數學運算。
GetMetricStatistics	`cloudwatch:GetMetricStatistics` 檢視 CloudWatch 主控台其他部分和儀表板 Widget 中的圖形所需。
GetMetricStream	`cloudwatch:GetMetricStream` 需要檢視指標串流相關資訊。
GetMetricWidgetImage	`cloudwatch:GetMetricWidgetImage` 需要擷取一或多個 CloudWatch度量的快照圖形作為點陣圖影像。
ListDashboards	`cloudwatch:ListDashboards` 需要檢視您帳戶中的 CloudWatch 儀表板清單。
ListMetrics	`cloudwatch:ListMetrics` 必須在 CloudWatch主控台和 CLI 中檢視或搜尋測量結果名稱。需要在儀表板 widget 上選取指標。
ListMetricStreams	`cloudwatch:ListMetricStreams` 需要檢視或搜尋帳戶中的指標串流清單。

CloudWatch API 作業	所需許可 (API 動作)
PutCompositeAlarm	cloudwatch:PutCompositeAlarm 需要建立複合警示。 若要建立複合警示，您的 cloudwatch:PutCompositeAlarm 許可必須具有 * 範圍。您無法傳回複合警示的相關資訊，如果您的 cloudwatch:PutCompositeAlarm 許可的範圍較窄。
PutDashboard	cloudwatch:PutDashboard 需要建立儀表板或更新現有的儀表板。
PutMetricAlarm	cloudwatch:PutMetricAlarm 需要建立或更新警示。
PutMetricData	cloudwatch:PutMetricData 需要建立指標。
PutMetricStream	cloudwatch:PutMetricStream 需要建立指標串流。
SetAlarmState	cloudwatch:SetAlarmState 需要手動設定警示的狀態。
StartMetricStreams	cloudwatch:StartMetricStreams 需要在指標串流中開始指標流程。

CloudWatch API 作業	所需許可 (API 動作)
StopMetricStreams	`cloudwatch:StopMetricStreams` 需要在指標串流中暫時停止指標流程。
TagResource	`cloudwatch:TagResource` 需要新增或更新 CloudWatch 資源 (例如警示和貢獻者見解規則) 的標籤。
UntagResource	`cloudwatch:UntagResource` 需要從 CloudWatch 資源中移除標籤。

CloudWatch 參與者見解 API 作業和動作所需的權限

Important

當您授與使用者`cloudwatch:PutInsightRule`權限時，依預設，該使用者可以建立用來評估記錄檔中任何 CloudWatch 記錄群組的規則。您可以新增 IAM 政策條件，以限制這些許可，進而讓使用者包含和排除特定日誌群組。如需詳細資訊，請參閱 使用條件索引鍵限制 Contributor Insights 使用者存取日誌群組。

CloudWatch 貢獻者洞察 API 操作	所需許可 (API 動作)
DeleteInsightRules	`cloudwatch:DeleteInsightRules` 需要刪除 Contributor Insights 規則。
DescribeInsightRules	`cloudwatch:DescribeInsightRules` 需要檢視您帳戶中的 Contributor Insights 規則。

CloudWatch 貢獻者洞察 API 操作	所需許可 (API 動作)
EnableInsightRules	`cloudwatch:EnableInsightRules` 需要啟用 Contributor Insights 規則。
GetInsightRuleReport	`cloudwatch:GetInsightRuleReport` 需要擷取由 Contributor Insights 規則收集的時間序列資料和其他統計資料。
PutInsightRule	`cloudwatch:PutInsightRule` 需要建立 Contributor Insights 規則。請參閱此表格開頭部分的 Important (重要) 注意事項。

CloudWatch 事件 API 作業和動作的必要權限

CloudWatch 事件 API 作業	所需許可 (API 動作)
DeleteRule	`events:DeleteRule` 刪除規則時需要。
DescribeRule	`events:DescribeRule` 列出規則的詳細資訊時需要。
DisableRule	`events:DisableRule` 停用規則時需要。
EnableRule	`events:EnableRule` 啟用規則時需要。

CloudWatch 事件 API 作業	所需許可 (API 動作)
ListRuleNamesByTarget	`events:ListRuleNamesByTarget` 列出與規則相關聯的目標時需要。
ListRules	`events:ListRules` 列出您帳戶中的所有規則時需要。
ListTargetsByRule	`events:ListTargetsByRule` 列出與規則相關聯的所有目標時需要。
PutEvents	`events:PutEvents` 新增可符合規則的自訂事件時需要。
PutRule	`events:PutRule` 建立或更新規則時需要。
PutTargets	`events:PutTargets` 將目標新增至規則時需要。
RemoveTargets	`events:RemoveTargets` 從規則中移除目標時需要。
TestEventPattern	`events:TestEventPattern` 針對特定事件測試事件模式時需要。

CloudWatch 記錄 API 操作和動作所需的權限

CloudWatch 記錄 API 作業	所需許可 (API 動作)
CancelExportTask	`logs:CancelExportTask` 取消待處理或執行匯出任務時為必要。
CreateExportTask	`logs:CreateExportTask` 從日誌群組將資料匯出至 Simple Storage Service (Amazon S3) 儲存貯體時為必要。
CreateLogGroup	`logs:CreateLogGroup` 建立新日誌群組時為必要。
CreateLogStream	`logs:CreateLogStream` 在日誌群組中建立新日誌串流時為必要。
DeleteDestination	`logs:DeleteDestination` 刪除日誌目的地及停用其任何訂閱篩選條件時為必要。
DeleteLogGroup	`logs:DeleteLogGroup` 刪除日誌群組及任何相關的存檔日誌事件時為必要。
DeleteLogStream	`logs:DeleteLogStream` 刪除日誌串流及任何相關的存檔日誌事件時為必要。

CloudWatch 記錄 API 作業	所需許可 (API 動作)
DeleteMetricFilter	`logs:DeleteMetricFilter` 刪除與日誌群組相關聯的指標篩選條件時為必要。
DeleteQueryDefinition	`logs:DeleteQueryDefinition` 刪除 CloudWatch 記錄深入解析中儲存的查詢定義所需。
DeleteResourcePolicy	`logs:DeleteResourcePolicy` 刪除 CloudWatch 記錄檔資源策略所需。
DeleteRetentionPolicy	`logs:DeleteRetentionPolicy` 刪除日誌群組的保留政策時為必要。
DeleteSubscriptionFilter	`logs:DeleteSubscriptionFilter` 刪除與日誌群組相關聯的訂閱篩選條件時為必要。
DescribeDestinations	`logs:DescribeDestinations` 檢視與帳戶相關的所有目的地時為必要。
DescribeExportTasks	`logs:DescribeExportTasks` 檢視與帳戶相關的所有匯出任務時為必要。
DescribeLogGroups	`logs:DescribeLogGroups` 檢視與帳戶相關的所有日誌群組時為必要。

CloudWatch 記錄 API 作業	所需許可 (API 動作)
DescribeLogStreams	`logs:DescribeLogStreams` 檢視與日誌群組相關的所有日誌串流時為必要。
DescribeMetricFilters	`logs:DescribeMetricFilters` 檢視與日誌群組相關的所有指標時為必要。
DescribeQueryDefinitions	`logs:DescribeQueryDefinitions` 需要在 CloudWatch 日誌見解中查看已儲存的查詢定義清單。
DescribeQueries	`logs:DescribeQueries` 需要查看已排程、執行或最近執行的 CloudWatch 記錄見解查詢清單。
DescribeResourcePolicies	`logs:DescribeResourcePolicies` 檢視記 CloudWatch 錄檔資源策略清單所需。
DescribeSubscriptionFilters	`logs:DescribeSubscriptionFilters` 檢視與日誌群組相關聯的所有訂閱篩選條件時為必要。
FilterLogEvents	`logs:FilterLogEvents` 依據日誌群組篩選條件模式排序日誌事件時為必要。

CloudWatch 記錄 API 作業	所需許可 (API 動作)
GetLogEvents	`logs:GetLogEvents` 從日誌串流擷取日誌事件時為必要。
GetLogGroupFields	`logs:GetLogGroupFields` 擷取日誌群組內日誌事件中包含的欄位清單時為必要。
GetLogRecord	`logs:GetLogRecord` 從單一日誌事件擷取詳細資訊時為必要。
GetQueryResults	`logs:GetQueryResults` 擷取 CloudWatch 日誌見解查詢的結果所需。
ListTagsLogGroup	`logs:ListTagsLogGroup` 列出與日誌群組相關的標籤時為必要。
PutDestination	`logs:PutDestination` 需要建立或更新目的地日誌串流 (例如 Kinesis 串流) 時為必要。
PutDestinationPolicy	`logs:PutDestinationPolicy` 建立或更新與現有日誌目的地相關的存取政策時為必要。

CloudWatch 記錄 API 作業	所需許可 (API 動作)
PutLogEvents	`logs:PutLogEvents` 將日誌事件批次上傳至日誌串流時為必要。
PutMetricFilter	`logs:PutMetricFilter` 建立或更新指標篩選條件並將其與日誌群組建立關聯時為必要。
PutQueryDefinition	`logs:PutQueryDefinition` 必須在 CloudWatch 記錄檔見解中儲存查詢。
PutResourcePolicy	`logs:PutResourcePolicy` 建立記 CloudWatch 錄檔資源策略所需。
PutRetentionPolicy	`logs:PutRetentionPolicy` 設定將日誌事件保持 (保留) 在日誌群組中的天數時為必要。
PutSubscriptionFilter	`logs:PutSubscriptionFilter` 建立或更新訂閱篩選條件並將其與日誌群組建立關聯時為必要。
StartQuery	`logs:StartQuery` 需要啟動 CloudWatch 日誌見解查詢。

CloudWatch 記錄 API 作業	所需許可 (API 動作)
StopQuery	`logs:StopQuery` 需要停止正在進行的 CloudWatch 日誌見解查詢。
TagLogGroup	`logs:TagLogGroup` 新增或更新日誌群組標籤時為必要。
TestMetricFilter	`logs:TestMetricFilter` 針對日誌事件訊息的取樣來測試篩選條件模式時為必要。

Amazon EC2 API 操作與動作所需的許可

Amazon EC2 API 操作	所需許可 (API 動作)
DescribeInstanceStatus	`ec2:DescribeInstanceStatus` 需要檢視 EC2 執行個體狀態的詳細資訊。
DescribeInstances	`ec2:DescribeInstances` 需要檢視 EC2 執行個體的詳細資訊。
RebootInstances	`ec2:RebootInstances` 需要重新啟動 EC2 執行個體。
StopInstances	`ec2:StopInstances` 需要停止 EC2 執行個體。

Amazon EC2 API 操作	所需許可 (API 動作)
TerminateInstances	`ec2:TerminateInstances` 需要終止 EC2 執行個體。

Amazon EC2 Auto Scaling API 操作與動作所需的許可

Amazon EC2 Auto Scaling API 操作	所需許可 (API 動作)
擴展	`autoscaling:Scaling` 需要擴展 Auto Scaling 群組。
觸發條件	`autoscaling:Trigger` 需要觸發 Auto Scaling 動作。

Amazon 的合規驗證 CloudWatch

第三方稽核員會在多個合規計劃中評估 Amazon CloudWatch 的安全性和合 AWS 規性。這些計劃包括 SOC、PCI、FedRAMP、HIPAA 等等。

如需特定規範計劃範圍內的 AWS 服務清單，請參閱合規計劃AWS 服務範圍方案)。如需一般資訊，請參閱 AWS 合規計劃 。

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。

使用 Amazon 時的合規責任取決 CloudWatch 於資料的敏感度、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：

- 安全與合規快速入門指南：這些部署指南討論架構考量，並提供在 AWS上部署以安全及合規為重心之基準環境的步驟。
- 建構 HIPAA 安全性與合規性白皮書 — 本白皮書說明公司如何使用建立符合 HIPAA 標準的應用 AWS 程式。

- 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。
- 使用AWS Config 開發人員指南中的規則評估資源 — AWS Config；評估您的資源配置如何符合內部實踐，業界準則和法規。
- AWS Security Hub— 此 AWS 服務提供安全狀態的全面檢視，協助您檢查您 AWS 是否符合安全性產業標準和最佳做法。

Amazon 的韌性 CloudWatch

AWS 全球基礎架構是圍繞區 AWS 域和可用區域建立的。區域提供多個分開且隔離的實際可用區域，並以低延遲、高輸送量和高度備援網路連線相互連結。透過可用區域，您可以設計與操作的應用程式和資料庫，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、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需區域和可用區域的相關 AWS 資訊，請參閱AWS 全域基礎結構。

Amazon 基礎設施安全 CloudWatch

作為受管服務，Amazon CloudWatch 受到 AWS 全球網路安全的保護。有關 AWS 安全服務以及如何 AWS 保護基礎架構的詳細資訊，請參閱AWS 雲端安全 若要使用基礎架構安全性的最佳做法來設計您的 AWS 環境，請參閱安全性支柱架構良 AWS 好的架構中的基礎結構保護。

您可以使用 AWS 已發佈的 API 呼叫透 CloudWatch 過網路存取。使用者端必須支援下列專案：

- 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，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

網路隔離

Virtual Private Cloud (VPC) 是 Amazon Web Services 雲端中您自己的邏輯隔離區域中的虛擬網路。子網是您的 VPC 中的 IP 地址範圍。您可以在 VPC 的子網路中部署各種 AWS 資源。例如，您可以在子網路中部署 Amazon EC2 執行個體執行個體、EMR 叢集和 DynamoDB 資料表。如需詳細資訊，請參閱 Amazon VPC 使用者指南。

若要在不透 CloudWatch 過公用網際網路的情況下與 VPC 中的資源進行通訊，請使用 AWS PrivateLink。如需詳細資訊，請參閱 使用 CloudWatch 和 CloudWatch Synthetics 與介面 VPC 端點。

私有子網路是沒有通往公有網際網路預設路由的子網路。在私有子網路中部署 AWS 資源並不會阻止 Amazon CloudWatch 從資源收集內建指標。

如果您需要從私有子網路中的 AWS 資源發佈自訂指標，可以使用 Proxy 伺服器執行此操作。代理服務器將這些 HTTPS 請求轉發到的公共 API 端點。 CloudWatch

AWS Security Hub

使用 Security Hub 監視與安全性最佳 CloudWatch 作法相關的使用 AWS 情況。Security Hub 會透過安全控制來評估資源組態和安全標準，協助您遵守各種合規架構。如需有關使用 Security Hub 評估 CloudWatch 資源的詳細資訊，請參閱 AWS 安全中心使用者指南中的 Amazon CloudWatch 控制。

使用 CloudWatch 和 CloudWatch Synthetics 與介面 VPC 端點

如果您使用 Amazon Virtual Private Cloud (Amazon VPC) 託管資 AWS 源，則可以在 VPC 和 CloudWatch Synthetics 之間建立私有連接。 CloudWatch您可以使用這些連接來啟用 CloudWatch 和 CloudWatch Synthetics 與您的 VPC 上的資源進行通信，而無需通過公共互聯網。

Amazon VPC 是一項 AWS 服務，可用於在您定義的虛擬網路中啟動 AWS 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 CloudWatch 或 CloudWatch Synthetics，您需要定義一個接口 VPC 端點以將 VPC 連接到服務。 AWS 端點提供可靠、可擴充的連 CloudWatch 線能 CloudWatch Synthetics 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 私人雲端端點的支援是一種使用具有私有 IP 位址的 elastic network interface AWS PrivateLink，在 AWS 服務之間進行私人通訊的 AWS 技術。如需詳細資訊，請參閱新增 AWS 服 AWS PrivateLink 務部落格文章。

下列步驟適用於 Amazon VPC 的使用者。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的入門。

CloudWatch VPC 端端

CloudWatch 目前支援下列 AWS 區域中的 VPC 端點：

- 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)
- 亞太區域 (香港)
- Asia Pacific (Mumbai)
- 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)
- Europe (Paris)
- 中東 (阿拉伯聯合大公國)
- 南美洲 (聖保羅)
- AWS GovCloud (美國東部)
- AWS GovCloud (美國西部)

為下列項目建立 VPC 端點 CloudWatch

若要開始使 CloudWatch 用您的 VPC，請建立的介面 VPC 端點。 CloudWatch要選擇的服務名稱為 `com.amazonaws.region.monitoring`。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的建立界面端點。

您不需要變更的設定 CloudWatch。 CloudWatch 使用公用 AWS 端點或私有介面 VPC 端點 (以使用中為準) 來呼叫其他服務。例如，如果您為其建立介面 VPC 端點 CloudWatch，且已經有指標 CloudWatch 從 VPC 上的資源流向，則這些指標依預設會開始流經介面 VPC 端點。

控制對 CloudWatch 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。

以下是的端點策略範例 CloudWatch。此原則允許 CloudWatch 透過 VPC 連線的使用者將指標資料傳送至， CloudWatch 並防止他們執行其他 CloudWatch 動作。

```
{
  "Statement": [
    {
      "Sid": "PutOnly",
      "Principal": "*",
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

若要編輯下列項目的 VPC 端點原則 CloudWatch

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. 如果尚未為其建立端點 CloudWatch，請選擇「建立端點」。選取 com.amazonaws.**#**.monitoring，然後選擇 Create endpoint (建立端點)。
4. 選取 com.amazonaws.**##**.monitoring 端點，然後選擇 Policy (政策) 索引標籤。
5. 選擇 Edit Policy (編輯政策)，然後進行變更。

CloudWatch Synthetics VPC 端點

CloudWatch Synthetics 目前支援以下區域的 VPC 端點： AWS

- 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)

- 亞太區域 (香港)
- Asia Pacific (Mumbai)
- 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)
- Europe (Paris)
- 南美洲 (聖保羅)

為 CloudWatch Synthetics 建立 VPC 端點

若要開始在 VPC 上使用 CloudWatch Synthetics，請為 Synthetics 建立介面 VPC 端點。 CloudWatch 要選擇的服務名稱為 `com.amazonaws.region.synthetics`。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的建立界面端點。

您不需要變更「 CloudWatch Synthetics」的設定。 CloudWatch Synthetics 使用公有端點或私有介面 VPC 端點 (以使用中為準) 與其他 AWS 服務進行通訊。例如，如果您為 CloudWatch 合成材料建立了一個介面虛擬私人雲端端點，並且已經有適用於 Amazon S3 的介面端點，則 CloudWatch Synthetics 預設會開始透過介面 VPC 端點與 Amazon S3 進行通訊。

控制對 CloudWatch Synthetics 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，我們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。

端點政策會影響由 VPC 私下管理的 canary。在私有子網路上執行的 canary 不需要它們。

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。

以下是 S CloudWatch ynthetics 的端點策略示例。此原則可讓使用者透過 VPC 連線至 CloudWatch Synthetics 料，以檢視金絲雀及其執行的相關資訊，但不能建立、修改或刪除金絲雀。

```
{
    "Statement": [
        {
            "Action": [
                "synthetics:DescribeCanaries",
                "synthetics:GetCanaryRuns"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

編輯 Synthetics 的 VPC 端點原則 CloudWatch

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「 CloudWatch Synthetics」的端點，請選擇「建立端點」。選取 com.amazonaws.**##**.synthetics 然後選擇 Create endpoint (建立端點)。
4. 選取 com.amazonaws.**##**.synthetics 端點然後選擇 Policy (政策) 索引標籤。
5. 選擇 Edit Policy (編輯政策)，然後進行變更。

Synthetics Canary 的安全考量

下列章節說明在 Synthetics 中建立並執行 Canary 時，您應該考量的安全問題。

使用安全連線

由於 Canary 程式碼和 Canary 測試執行的結果可能包含敏感資訊，因此請勿讓 Canary 透過未加密的連線連至端點。請一律使用加密連線，例如以 `https://` 開頭的連線。

Canary 命名考量

金絲雀的 Amazon 資源名稱 (ARN) 包含在使用者代理程式標頭中，作為從木偶驅動的 Chromium 瀏覽器進行的輸出呼叫的一部分，該瀏覽器包含在 Synthetics 包裝函式庫中。 CloudWatch 這有助於識別 CloudWatch Synthetics 金絲雀流量，並將其關聯回正在撥打電話的金絲雀。

Canary ARN 包括 Canary 名稱。請選擇不會透露專屬資訊的 Canary 名稱。

此外，請務必僅將 Canary 指向您控制的網站和端點。

Canary 程式碼中的秘密和敏感資訊

如果您使用 zip 文件將初期測試代碼直接傳遞到 Canary 中，則可以在 AWS CloudTrail 日誌中看到腳本的內容。

如果您在 Canary 指令碼中有敏感資訊或秘密 (例如存取金鑰或資料庫憑證)，我們強烈建議您將指令碼存放為 Amazon S3 中的版本控制物件，並將 Amazon S3 位置傳遞至 Canary，而不是透過 zip 檔案傳遞 Canary 程式碼。

如果您確實使用 zip 檔案來傳遞 Canary 指令碼，我們強烈建議您不要在 Canary 原始碼中包含秘密或敏感資訊。如需有關如何使用 AWS Secrets Manager 來協助保護秘密安全的詳細資訊，請參閱什麼是 AWS Secrets Manager？ 。

許可考量

我們建議您限制對 CloudWatch Synthetics 所建立或使用的資源的存取。在 Canary 存放測試執行結果和其他成品 (例如日誌和螢幕擷取畫面) 的 Amazon S3 儲存貯體上，使用嚴密的許可。

同樣地，對存放 Canary 來源碼的位置保持使用嚴格的許可，以免使用者意外 (或惡意) 刪除用於 Canary 的 Lambda 圖層或 Lambda 函數。

為了協助確保執行您想要的 Canary 程式碼，您可以在存放 Canary 程式碼的 Amazon S3 儲存貯體上使用物件版本控制。然後，當您指定此程式碼做為 Canary 來執行時，您可以將物件 `versionId` 做為路徑的一部分，如下例所示。

```
https://bucket.s3.amazonaws.com/path/object.zip?versionId=version-id
https://s3.amazonaws.com/bucket/path/object.zip?versionId=version-id
https://bucket.s3-region.amazonaws.com/path/object.zip?versionId=version-id
```

堆疊追蹤和異常情況訊息

默認情況下，無論腳本是自定義還是來自藍圖， CloudWatch Synthetics 金絲雀都會捕獲由初期測試腳本拋出的任何異常。 CloudWatch Synthetics 將異常消息和堆棧跟踪記錄到三個位置：

- 當您描述測試運行時，返回 CloudWatch Synthetics 服務以加快調試
- 根據您建立 Lambda 函數的組態，將 CloudWatch 記錄檔納入記錄中
- 記錄至 Synthetics 日誌檔案，這是純文字檔案，此檔案會上傳至您為 Canary `resultsLocation` 設定之值所指定的 Amazon S3 位置

如果您想發送和存儲較少的信息，則可以在異常返回到 CloudWatch Synthetics 包裝器庫之前捕獲異常。

您也可以在錯誤中包含請求 URL。 CloudWatch Synthetics 掃描腳本拋出的錯誤中的任何 URL，並根據配置從中編輯受限制的 URL 參數。restrictedUrlParameters如果您在指令碼中記錄錯誤訊息，則可以使用 getSanitizedError訊息 以在記錄之前修訂 URL。

以較小範圍限制您的 IAM 角色

建議您不要將 Canary 設定為前往潛在的惡意 URL 或端點。將 Canary 指向不受信任或未知的網站或端點，可能會將您的 Lambda 函數程式碼暴露在惡意使用者的指令碼中。假設惡意網站可以突破 Chromium，則如果您使用網際網路瀏覽器連線至該網站，該網站可能會以類似方式存取您的 Lambda 程式碼。

使用具有較小範圍許可的 IAM 執行角色，來執行您的 Lambda 函數。如此一來，如果您的 Lambda 函數受到惡意指令碼入侵，則作為 Canary AWS 帳戶執行時可能採取的動作會受到限制。

當您使用 CloudWatch 主控台建立初期測試時，會使用降低範圍的 IAM 執行角色建立該主控台。

敏感資料修訂

CloudWatch Synthetics 捕獲 URL，狀態代碼，故障原因（如果有的話）以及請求和響應的標題和主體。這樣一來，canary 使用者即能了解、監控和偵測 canary。

以下各節中描述的組態可以在 canary 執行的任何時候進行設定。您也可以選擇將不同的組態套用至不同的綜合步驟。

請求 URL

根據預設， CloudWatch Synthetics 會記錄請求 URL、狀態碼，以及初期測試記錄中每個 URL 的狀態原因。請求 URL 也可以出現在 canary 執行報告、HAR 檔案等中。您的請求 URL 可能包含敏感的查詢參數，例如存取字符或密碼。您可以編輯敏感信息，防止 CloudWatch Synthetics 記錄。

若要編輯敏感資訊，請設定組態屬性restrictedUrlParameters。如需詳細資訊，請參閱 SyntheticsConfiguration 類。這會導致 CloudWatch Synthetics 根據記錄restrictedUrlParameters之前編輯 URL 參數，包括路徑和查詢參數值。如果您在指令碼中記錄 URL，則可以使用 getSanitizedUrl（網址，步驟配置 = 空） 以在記錄之前修訂 URL。如需詳細資訊，請參閱 SyntheticsLogHelper 類。

標頭

默認情況下， CloudWatch Synthetics 不會記錄請求/響應標頭。對於 UIcanary，這是使用執行時間版本 `syn-nodejs-puppeteer-3.2` 及更高版本的 canary 的預設行為。

如果您的標頭不包含敏感資訊，您可以將和includeResponseHeaders屬性設定為，在 HAR 檔案includeRequestHeaders和 HTTP 報表中啟用標頭`true`。您可以啟用所有標頭，但選擇限制敏感標頭金鑰的值。例如，您可以選擇僅修訂 canary 產生的成品中的 `Authorization`標頭。

請求與回應內文

默認情況下， CloudWatch Synthetics 不會在初期測試日誌或報告中記錄請求/響應主體。此資訊對於 API canary 尤為實用。Synthetics 會擷取所有 HTTP 請求，並可顯示標頭、請求和回應內文。如需詳細資訊，請參閱 executeHttpStep（stepName，請求選項，[回調]，[步驟配置]）。您可以將includeRequestBody和includeResponseBody屬性設定為，以選擇啟用要求/回應主體。`true`

記錄 Amazon CloudWatch API 呼叫 AWS CloudTrail

Amazon CloudWatch 和 CloudWatch Synthetics 整合在一起 AWS CloudTrail，該服務可提供使用者、角色或 AWS 服務採取的操作記錄。 CloudTrail 擷取您帳戶或代表您 AWS 帳戶進行的 API 呼叫。擷取的呼叫包括來自主控台的呼叫，以及對 API 操作的程式碼呼叫。

如果您建立追蹤，您可以啟用 CloudTrail事件持續傳遞至 S3 儲存貯體，包括 CloudWatch. 如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail 主控台中最近的事件。使用收集的資訊 CloudTrail，您可以判斷提出要求 CloudWatch、提出要求的 IP 位址、提出要求的人員、提出要求的時間以及其他詳細資訊。

若要進一步了解 CloudTrail，包括如何設定和啟用它，請參閱AWS CloudTrail 使用者指南。

每一筆事件或日誌專案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：

- 要求是使用根使用者登入資料還是 AWS Identity and Access Management (IAM) 使用者登入資料提出。
- 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。
- 請求是否由其他 AWS 服務提出。

如需詳細資訊，請參閱CloudTrail 使 userIdentity 元素。

如需 AWS 帳戶中持續記錄事件 (包括合成事件 CloudWatch 和 CloudWatch Synthetics)，請建立追蹤。追蹤可 CloudTrail 將日誌檔傳遞至 S3 儲存貯體。根據預設，當您在主控台中建立追蹤時，追蹤會套用至所有 AWS 區域。追蹤記錄來自 AWS 分割區中所有區域的事件，並將日誌檔傳送到您指定的 S3 儲存貯體。您可以設定其他 AWS 服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定的 Amazon SNS 通知 CloudTrail
- 從多個區域接收 CloudTrail 記錄檔並從多個帳戶接收 CloudTrail記錄檔

> **Note**
>
> 如需有關已登入之 CloudWatch 記錄 API 呼叫的詳細資訊 CloudTrail，請參閱中的 CloudWatch 記錄檔資訊 CloudTrail。

主題

- CloudWatch 中的資訊 CloudTrail
- CloudWatch 互聯網監控 CloudTrail
- CloudWatch Synthetics 資訊 CloudTrail

CloudWatch 中的資訊 CloudTrail

CloudWatch 支援將下列動作記錄為記 CloudTrail 錄檔中的事件：

- DeleteAlarms
- DeleteAnomalyDetector
- DeleteDashboards
- DescribeAlarmHistory
- DescribeAlarms
- DescribeAlarmsForMetric
- DescribeAnomalyDetectors
- DisableAlarmActions
- EnableAlarmActions
- GetDashboard
- ListDashboards
- PutAnomalyDetector
- PutDashboard
- PutMetricAlarm
- SetAlarmState

範例： CloudWatch 記錄檔項目

下列範例顯示示範PutMetricAlarm動作的 CloudTrail 記錄項目。


```
{
    "Records": [{
        "eventVersion": "1.01",
        "userIdentity": {
            "type": "Root",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::123456789012:root",
            "accountId": "123456789012",
            "accessKeyId": "EXAMPLE_KEY_ID"
        },
        "eventTime": "2014-03-23T21:50:34Z",
        "eventSource": "monitoring.amazonaws.com",
        "eventName": "PutMetricAlarm",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "aws-sdk-ruby2/2.0.0.rc4 ruby/1.9.3 x86_64-linux Seahorse/0.1.0",
        "requestParameters": {
            "threshold": 50.0,
            "period": 60,
            "metricName": "CloudTrail Test",
            "evaluationPeriods": 3,
            "comparisonOperator": "GreaterThanThreshold",
            "namespace": "AWS/CloudWatch",
            "alarmName": "CloudTrail Test Alarm",
            "statistic": "Sum"
        },
        "responseElements": null,
        "requestID": "29184022-b2d5-11e3-a63d-9b463e6d0ff0",
        "eventID": "b096d5b7-dcf2-4399-998b-5a53eca76a27"
    },
    ..additional entries
  ]
  }
```


下列記錄檔項目顯示使用者呼叫「 CloudWatch 事件」PutRule 動作。


```
{
         "eventVersion":"1.03",
         "userIdentity":{
```

```
            "type":"Root",
            "principalId":"123456789012",
            "arn":"arn:aws:iam::123456789012:root",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "sessionContext":{
               "attributes":{
                  "mfaAuthenticated":"false",
                  "creationDate":"2015-11-17T23:56:15Z"
               }
            }
         },
         "eventTime":"2015-11-18T00:11:28Z",
         "eventSource":"events.amazonaws.com",
         "eventName":"PutRule",
         "awsRegion":"us-east-1",
         "sourceIPAddress":"AWS Internal",
         "userAgent":"AWS CloudWatch Console",
         "requestParameters":{
            "description":"",
            "name":"cttest2",
            "state":"ENABLED",
            "eventPattern":"{\"source\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance
 State-change Notification\"]}",
            "scheduleExpression":""
         },
         "responseElements":{
            "ruleArn":"arn:aws:events:us-east-1:123456789012:rule/cttest2"
         },
         "requestID":"e9caf887-8d88-11e5-a331-3332aa445952",
         "eventID":"49d14f36-6450-44a5-a501-b0fdcdfaeb98",
         "eventType":"AwsApiCall",
         "apiVersion":"2015-10-07",
         "recipientAccountId":"123456789012"
}
```

下列記錄檔項目顯示使用者呼叫「 CloudWatch 記錄檔」CreateExportTask 動作。

```
{
        "eventVersion": "1.03",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
```

```
            "arn": "arn:aws:iam::123456789012:user/someuser",
            "accountId": "123456789012",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "someuser"
        },
        "eventTime": "2016-02-08T06:35:14Z",
        "eventSource": "logs.amazonaws.com",
        "eventName": "CreateExportTask",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "aws-sdk-ruby2/2.0.0.rc4 ruby/1.9.3 x86_64-linux Seahorse/0.1.0",
        "requestParameters": {
            "destination": "yourdestination",
            "logGroupName": "yourloggroup",
            "to": 123456789012,
            "from": 0,
            "taskName": "yourtask"
        },
        "responseElements": {
            "taskId": "15e5e534-9548-44ab-a221-64d9d2b27b9b"
        },
        "requestID": "1cd74c1c-ce2e-12e6-99a9-8dbb26bd06c9",
        "eventID": "fd072859-bd7c-4865-9e76-8e364e89307c",
        "eventType": "AwsApiCall",
        "apiVersion": "20140328",
        "recipientAccountId": "123456789012"
}
```


CloudWatch 互聯網監控 CloudTrail

CloudWatch 網際網路監視器支援記錄下列動作為記 CloudTrail 錄檔中的事件。

- CreateMonitor
- DeleteMonitor
- GetHealthEvent
- GetMonitor
- GetQueryResults
- GetQueryStatus
- ListHealthEvents
- ListMonitors

- ListTagsForResource
- StartQuery
- StopQuery
- UpdateMonitor

範例： CloudWatch 網際網路監視器記錄檔項目

下列範例顯示示範`ListMonitors`動作的 CloudTrail 網際網路監視器記錄項目。


```
{
        "eventVersion": "1.08",
        "userIdentity": {
            "type": "AssumedRole",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::000000000000:assumed-role/role_name",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "sessionContext": {
                "sessionIssuer": {
                "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::000000000000:role/Admin",
                "accountId":"123456789012",
                "userName": "SAMPLE_NAME"
                },
                "webIdFederationData": {},
                "attributes": {
                    "creationDate": "2022-10-11T17:25:41Z",
                    "mfaAuthenticated": "false"
                }
            }
        },
        "eventTime": "2022-10-11T17:30:18Z",
        "eventSource": "internetmonitor.amazonaws.com",
        "eventName": "ListMonitors",
        "awsRegion": "us-east-2",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)",
        "requestParameters": null,
        "responseElements": null,
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
```

```
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "readOnly": true,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "eventCategory": "Management"
    }
```

下列範例顯示示範CreateMonitor動作的 CloudTrail 網際網路監視器記錄項目。

```
{
        "eventVersion": "1.08",
        "userIdentity": {
            "type": "AssumedRole",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::000000000000:assumed-role/role_name",
            "accountId":"123456789012",
            "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
            "sessionContext": {
                "sessionIssuer": {
                "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::000000000000:role/Admin",
                "accountId":"123456789012",
                "userName": "SAMPLE_NAME"
                },
                "webIdFederationData": {},
                "attributes": {
                    "creationDate": "2022-10-11T17:25:41Z",
                    "mfaAuthenticated": "false"
                }
            }
        },
        "eventTime": "2022-10-11T17:30:08Z",
        "eventSource": "internetmonitor.amazonaws.com",
        "eventName": "CreateMonitor",
        "awsRegion": "us-east-2",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)",
        "requestParameters": {
            "MonitorName": "TestMonitor",
            "Resources": ["arn:aws:ec2:us-east-2:444455556666:vpc/vpc-febc0b95"],
            "ClientToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333"
```

```
        },
        "responseElements": {
            "Arn": "arn:aws:internetmonitor:us-east-2:444455556666:monitor/ct-
onboarding-test",
            "Status": "PENDING"
        },
        "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "eventCategory": "Management"
    }
```

CloudWatch Synthetics 資訊 CloudTrail

CloudWatch Synthetics 支持將以下操作記錄為 CloudTrail 日誌文件中的事件：

- CreateCanary
- DeleteCanary
- DescribeCanaries
- DescribeCanariesLastRun
- DescribeRuntimeVersions
- GetCanary
- GetCanaryRuns
- ListTagsForResource
- StartCanary
- StopCanary
- TagResource
- UntagResource
- UpdateCanary

範例： CloudWatch Synthetics 件記錄檔項目

下列範例顯示示範`DescribeCanaries`動作的 CloudTrail Synthetics 記錄項目。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::111222333444:role/Administrator",
                "accountId":"123456789012",
                "userName": "SAMPLE_NAME"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-04-08T21:43:24Z"
            }
        }
    },
    "eventTime": "2020-04-08T23:06:47Z",
    "eventSource": "synthetics.amazonaws.com",
    "eventName": "DescribeCanaries",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.590
 Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.212-b03
 java/1.8.0_212 vendor/Oracle_Corporation",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "201ed5f3-15db-4f87-94a4-123456789",
    "eventID": "73ddbd81-3dd0-4ada-b246-123456789",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```


下列範例顯示示範`UpdateCanary`動作的 CloudTrail Synthetics 記錄項目。


```
{
```

```
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
               "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::111222333444:role/Administrator",
       "accountId":"123456789012",
                "userName": "SAMPLE_NAME"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-04-08T21:43:24Z"
            }
        }
    },
    "eventTime": "2020-04-08T23:06:47Z",
    "eventSource": "synthetics.amazonaws.com",
    "eventName": "UpdateCanary",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.590
Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.212-b03
java/1.8.0_212 vendor/Oracle_Corporation",
    "requestParameters": {
        "Schedule": {
            "Expression": "rate(1 minute)"
        },
        "name": "sample_canary_name",
        "Code": {
            "Handler": "myOwnScript.handler",
            "ZipFile": "SAMPLE_ZIP_FILE"
        }
    },
    "responseElements": null,
    "requestID": "fe4759b0-0849-4e0e-be71-1234567890",
    "eventID": "9dc60c83-c3c8-4fa5-bd02-1234567890",
    "readOnly": false,
```

```
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

下列範例顯示示範GetCanaryRuns動作的 CloudTrail Synthetics 記錄項目。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:assumed-role/role_name",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::111222333444:role/Administrator",
       "accountId":"123456789012",
                "userName": "SAMPLE_NAME"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-04-08T21:43:24Z"
            }
        }
    },
    "eventTime": "2020-04-08T23:06:30Z",
    "eventSource": "synthetics.amazonaws.com",
    "eventName": "GetCanaryRuns",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.590
 Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.212-b03
 java/1.8.0_212 vendor/Oracle_Corporation",
    "requestParameters": {
        "Filter": "TIME_RANGE",
        "name": "sample_canary_name",
        "FilterValues": [
            "2020-04-08T23:00:00.000Z",
            "2020-04-08T23:10:00.000Z"
```

```
        ]
    },
    "responseElements": null,
    "requestID": "2f56318c-cfbd-4b60-9d93-1234567890",
    "eventID": "52723fd9-4a54-478c-ac55-1234567890",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

標記您的 Amazon CloudWatch 資源

標籤是您或 AWS 指派給 AWS 資源的自訂屬性標籤。每個標籤有兩個部分：

- 標籤鍵 (例如，CostCenter、Environment 或 Project)。標籤鍵會區分大小寫。
- 選用欄位，稱為標籤值 (例如 111122223333 或 Production)。忽略標籤值基本上等同於使用空字串。與標籤鍵相同，標籤值會區分大小寫。

標籤可協助您執行以下操作：

- 識別和組織您的 AWS 資源。許多 AWS 服務都支援標記，因此您可以將相同標籤指派給來自不同服務的資源，以指出資源是相關的。例如，您可以將相同的標籤指派給指派給 EC2 執行個體的 CloudWatch 規則。

以下各節提供有關的標籤的詳細資訊 CloudWatch。

支援的資源 CloudWatch

CloudWatch 支持標記的以下資源：

- 警報 — 您可以使用標籤資源 AWS CLI 命令和 API 來標記警報。TagResource您也可以使用 CloudWatch 主控台中的 [警報詳細資料] 頁面來檢視和管理鬧鐘標籤。
- 加那利群島-您可以使用控制台標記加那利群島。 CloudWatch 如需詳細資訊，請參閱 建立 Canary。
- 貢獻者見解規則 — 您可以在建立貢獻者見解規則時，使用put-insight-rule AWS CLI 命令和 API 標記參與者見解規則PutInsightRule。您可以使用標籤資源 AWS CLI 命令和 API 將標籤新增至現有規則。TagResource
- 量度串流 — 您可以在使用put-metric-stream AWS CLI 指令和 PutMetricStreamAPI 建立測量結果串流時標記測量結果串流。您可以使用標籤資源 AWS CLI 命令和 API，將標記新增至現有的指標串流。TagResource

如需新增和管理標籤的詳細資訊，請參閱管理標籤。

管理標籤

標籤是由資源上的 `Key` 和 `Value` 屬性組成。您可以使用 CloudWatch 主控台 AWS CLI、或 CloudWatch API 來新增、編輯或刪除這些屬性的值。如需使用標籤的資訊，請參閱下列內容：

- TagResourceUntagResource、和ListTagsForResource在 Amazon CloudWatch API 參考
- 在 Amazon CLI 參考中標記資源、無標記資list-tags-for-resource源 CloudWatch
- 《Resource Groups 使用者指南》中的使用標籤編輯器

標籤命名和使用慣例

下列基本命名和使用慣例適用於搭配CloudWatch 資源使用標籤：

- 每個資源的上限為 50 個標籤。
- 對於每一個資源，每個標籤金鑰必須是唯一的，且每個標籤金鑰只能有一個值。
- 最大標籤索引鍵長度為 128 個 UTF-8 形式的 Unicode 字元。
- 最大標籤值長度為 256 個 UTF-8 形式的 Unicode 字元。
- 允許的字元包括可用 UTF-8 表示的英文字母、數字、空格，還有以下特殊字元：. : + = @ _ / - (連字號)。
- 標籤鍵與值皆區分大小寫。做為最佳實務，請決定大寫標籤的策略，並一致地在所有資源類型中實作該策略。例如，決定要使用 `Costcenter`、`costcenter` 還是 `CostCenter`，並針對所有標籤使用相同的慣例。避免針對相似的標籤使用不一致的大小寫處理。
- 標籤禁止使用`aws:`前綴，因為它保留供 AWS 使用。您不可編輯或刪除具此字首的標籤金鑰或值。具此字首的標籤，不算在受資源限制的標籤計數內。

Grafana 整合

您可以使用 Grafana 6.5.0 及更新版本，透過 CloudWatch 主控台以內容方式前進，並使用萬用字元查詢動態指標清單。這可協助您監控 AWS 資源的指標，例如 Amazon Elastic Compute Cloud 執行個體或容器。在 Auto Scaling 事件過程中建立新執行個體時，它們會自動顯示在圖表中。您不必追蹤新的執行個體 ID。預先建置的儀表板有助於簡化監控 Amazon EC2、Amazon 彈性區塊存放區和 AWS Lambda 資源的入門體驗。

您可以使用 Grafana 7.0 版及更新版本，對記錄檔中 CloudWatch 的 CloudWatch 記錄群組執行記錄見解查詢。您可以將查詢結果視覺化，成為長條圖、折線圖和堆疊圖形，並以表格格式呈現。如需 CloudWatch 日誌深入解析的相關資訊，請參閱使用日誌深入分析分析 CloudWatch 記錄資料

如需有關如何開始使用的詳細資訊，請參閱 Grafana 實驗室文件 AWS CloudWatch 中的在 Grafana 中使用。

跨帳戶跨 CloudWatch 區域主控台

為了獲得指標、日誌和跟踪的最豐富的跨帳戶觀察性和發現體驗，我們建議您使用 CloudWatch 跨帳戶觀察能力。如需詳細資訊，請參閱 CloudWatch 跨帳戶可觀察性。

CloudWatch 還提供跨帳戶的跨區域 CloudWatch 儀表板。此功能可讓您跨帳戶檢視儀表板、警示、指標和自動儀表板。它無法跨帳戶檢視日誌或追蹤。

如果您也使用 CloudWatch 跨帳戶可觀察性，則此跨帳戶 CloudWatch 儀表板的一個使用案例是讓您的一個跨帳戶可觀察性來源 CloudWatch 帳戶看到另一個來源帳戶的指標。

本節的其餘部分會說明跨帳戶、跨區域儀表板。您可以使用它來建立儀表板，將來自多個 AWS 帳戶和多個 AWS 區域的 CloudWatch 資料彙總到單一儀表板中。您也可以在一個帳戶中建立警示，以監視位於不同帳戶中的指標。

許多組織都將其 AWS 資源部署在多個帳戶中，以提供計費和安全性限制。在此情況下，我們建議您將一或多個帳戶指定為監控帳戶，並在這些帳戶中建立跨帳戶儀表板。

跨帳戶功能與整合 AWS Organizations，可協助您有效率地建置跨帳戶儀表板。

跨區域功能

現已自動內建跨區域功能。您不需要執行任何額外的步驟，就能在同一個圖形或同一個儀表板上顯示單一帳戶中來自不同區域的指標。警示不支援跨區域功能，因此您無法在一個觀察不同區域指標的區域中建立警示。

主題

- 啟用跨帳戶功能 CloudWatch
- (選擇性) 整合 AWS Organizations
- CloudWatch 跨帳戶設定疑難排解
- 使用跨帳戶後停用和清除

啟用跨帳戶功能 CloudWatch

若要在 CloudWatch 主機中設定跨帳戶功能，請使用主 CloudWatch 控台來設定共用帳戶和監控帳戶。

設定共用帳戶

您必須在每個帳戶中啟用共用，讓資料可供監控帳戶使用。

這會將您在步驟 5 中選擇的唯讀許可授予所有在您共用的帳戶中檢視跨帳戶儀表板的使用者，如果使用者在您共用的帳戶中擁有對應的許可。

讓您的帳戶能夠與其他帳戶共用 CloudWatch 資料

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇設定。
3. 針對 [共用您的 CloudWatch 資料] 選擇 [設定]。
4. 在 Sharing (共用) 中，選擇 Specific accounts (特定帳戶)，然後輸入您要共用資料之帳戶 ID。

   您在此處指定的任何帳戶都可以檢視您帳戶的 CloudWatch 資料。僅指定您知道且信任的帳戶 ID。
5. 在 Permissions (權限)，請指定如何使用下列其中一個選項以共用您的資料：

   - 提供 CloudWatch 指標、儀表板和警示的唯讀存取權。此選項可讓監控帳戶建立跨帳戶儀表板，其中包含包含來自您帳戶 CloudWatch 資料的 Widget。
   - 包括 CloudWatch 自動儀表板。如果您選取此選項，監控帳戶中的使用者也可以檢視此帳戶自動儀表板中的資訊。如需詳細資訊，請參閱 開始使用 Amazon CloudWatch。
   - 包含 X-Ray 追蹤地圖的 X-Ray 唯讀存取權。如果您選取此選項，監控帳戶中的使用者也可以檢視此帳戶中的 X-Ray 追蹤地圖和 X-Ray 追蹤資訊。如需詳細資訊，請參閱使用 X-Ray 追蹤地圖。
   - 完整唯讀存取您帳戶中的所有內容。此選項可讓您用於共用的帳戶建立跨帳戶儀表板，其中包含包含來自您帳戶 CloudWatch 資料的 Widget。它也可以讓這些帳戶更深入地查看您的帳戶，並在其他 AWS 服務的主控台中檢視您帳戶的資料。
6. 選擇 [啟動 CloudFormation 範本]。

   在確認畫面中輸入 **`Confirm`**，並選擇 Launch template (啟動範本)。
7. 選取 I acknowledge... (我知道...) 核取方塊，然後選擇 Create stack (建立堆疊)。

與整個組織共用

完成前述程序會建立一個 IAM 角色，讓您的帳戶能夠與一個帳戶共用資料。您可以建立或編輯與組織中所有帳戶共用資料的 IAM 角色。請在您知道並信任組織中的所有帳戶時，才執行這項操作。

這會將先前程序的步驟 5 中所示政策中列出的唯讀許可授予所有在您共用的帳戶中檢視跨帳戶儀表板的使用者，如果使用者在您共用的帳戶中擁有對應的許可。

與組織中的所有 CloudWatch 帳戶共用您的帳戶資料

1. 如果您尚未完成，請完成上述程序，以便與一個 AWS 帳戶共用您的資料。
2. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
3. 在導覽窗格中，選擇角色。
4. 在角色清單中，選擇 CloudWatch-CrossAccountSharingRole。
5. 選擇 Trust relationships (信任關係)、Edit trust relationship (編輯信任關係)。

   您會看到以下政策：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

6. 將該政策變更為下列內容，以您組織的 ID 取代 *org-id*。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalOrgID": "org-id"
        }
      }
```

```
    }
  ]
}
```

7. 選擇 Update Trust Policy (更新信任政策)。

設定監控帳戶

如果要查看跨帳戶 CloudWatch 數據，請啟用每個監控帳戶。

當您完成下列程序時， CloudWatch 會建立服務連結角色，該角色會在監控帳戶中 CloudWatch 使用該角色來存取從其他帳戶共用的資料。此服務連結角色稱AWSServiceRoleForCloudWatchCrossAccount為。如需詳細資訊，請參閱使用 CloudWatch 的服務連結角色。

讓您的帳戶能夠檢視跨帳戶資 CloudWatch 料

1. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇 Settings (設定)，然後在 Cross-account cross-region (跨帳戶跨區域) 區段中，選擇 Configure (設定)。
3. 在 [檢視跨帳戶跨區域] 區段下，選擇 [啟用]，然後選取 [在主控台中顯示選取器] 核取方塊，以便在繪製指標或建立警示時，帳戶選擇器顯示在 CloudWatch 主控台中。
4. 在 View cross-account cross-region (檢視跨帳戶跨區域) 下，選擇下列其中一個選項：

   - Account Id Input (帳戶 ID 輸入)。此選項會在您每次檢視跨帳戶資料，並想切換帳戶時，提示您手動輸入帳戶 ID。
   - AWS 組織帳戶選取器。此選項會導致帳戶出現 (此帳戶是當您完成跨帳戶與 Organizations 的整合時所指定的帳戶)。下次使用主控台時， CloudWatch會顯示這些帳戶的下拉式清單，供您在檢視跨帳戶資料時選擇。

     若要這麼做，您必須先使用組織管理帳戶，才能 CloudWatch 查看組織中的帳戶清單。如需詳細資訊，請參閱 (選擇性) 整合 AWS Organizations。
   - Custom account selector (自訂帳戶選擇器)。此選項會提示您輸入帳戶 ID 清單。下次使用主控台時， CloudWatch 會顯示這些帳戶的下拉式清單，供您在檢視跨帳戶資料時選擇。

     您也可以為這些帳戶個別輸入標籤，以協助您在選擇要檢視的帳戶時識別這些帳戶。

     使用者在此處進行的帳戶選擇器設定只會保留供該使用者使用，不適用於監控帳戶中的所有其他使用者。

5. 選擇 啟用 。

完成此設定之後，您可以建立跨帳戶儀表板。如需詳細資訊，請參閱 跨帳戶跨區域儀表板。

(選擇性) 整合 AWS Organizations

如果要將跨帳戶功能與整合 AWS Organizations，則必須列出組織中所有可供監視帳戶使用的帳戶清單。

若要啟用跨帳戶 CloudWatch 功能，以存取組織中所有帳戶的清單

1. 登入您組織的管理帳戶。
2. 開啟主 CloudWatch 控台，網址為 https://console.aws.amazon.com/cloudwatch/。
3. 在導覽窗格中，選擇 Settings (設定)，然後選擇 Configure (配置)。
4. 在 Grant permission to view the list of accounts in the organization (授與檢視組織中帳戶清單權限) 中，請選擇要提示輸入帳戶 ID 清單的 Specific accounts (特定帳戶)。組織中的帳戶清單只會與您在此處指定的帳戶共用。
5. 選擇 Share organization account list (共用組織帳戶清單)。
6. 選擇 [啟動 CloudFormation 範本]。

   在確認畫面中輸入 **`Confirm`**，並選擇 Launch template (啟動範本)。

CloudWatch 跨帳戶設定疑難排解

本節包含中 CloudWatch跨帳戶、主控台部署的疑難排解秘訣。

我收到顯示跨帳戶資料的存取被拒錯誤

請檢查以下內容：

- 您的監控帳戶應具有名為的角色AWSServiceRoleForCloudWatchCrossAccount。如果沒有，則需要建立該角色。如需詳細資訊，請參閱 Set Up a Monitoring Account。
- 每個共享帳戶都應該有一個名為 CloudWatch- 的角色CrossAccountSharingRole。如果沒有，則需要建立該角色。如需詳細資訊，請參閱 Set Up A Sharing Account。
- 該共用角色必須信任監控帳戶。

確認您的角色已正確設定 CloudWatch 跨帳戶主控台

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇角色。
3. 在角色清單中，確定所需的角色存在。在共享帳戶中，尋找 CloudWatch-CrossAccountSharingRole。在監控帳戶中，尋找AWSServiceRoleForCloudWatchCrossAccount。
4. 如果您在共用帳戶中且CloudWatchCrossAccountSharingRole已存在，請選擇 CloudWatch-CrossAccountSharingRole。
5. 選擇 Trust relationships (信任關係)、Edit trust relationship (編輯信任關係)。
6. 確認政策會列出監控帳戶的帳戶 ID，或包含監控帳戶之組織的組織 ID。

我在主控台中看不到帳戶的下拉式清單

首先，請檢查您是否已建立正確的 IAM 角色，如前述故障診斷一節中所述。如果設定正確，請確定您已啟用此帳戶以檢視跨帳戶資料，如 Enable Your Account to View Cross-Account Data 中所述。

使用跨帳戶後停用和清除

若要停用的跨帳戶功能 CloudWatch，請依照下列步驟執行。

步驟 1：移除跨帳戶堆疊或角色

最好的方法是移除用來啟用跨帳戶功能的 AWS CloudFormation 堆疊。

- 在每個共用帳戶中，移除 CloudWatch-CrossAccountSharingRole 堆疊。
- 如果您曾經針 AWS Organizations 對組織中的所有帳戶啟用跨帳戶功能，請移除組織管理帳戶中的 CloudWatch-CrossAccountListAccountsRole 堆疊。

如果您未使用 AWS CloudFormation 堆疊來啟用跨帳戶功能，請執行下列動作：

- 在每個共用帳戶中，刪除 CloudWatch-CrossAccountSharingRole IAM 角色。
- 如果您曾經針 AWS Organizations 對組織中的所有帳戶啟用跨帳戶功能，請刪除組織管理帳戶中的 CloudWatchCrossAccountSharing--ListAccountsRole IAM 角色。

步驟 2：移除服務連結角色

在監控帳戶中，刪除AWSServiceRoleForCloudWatchCrossAccount服務連結的 IAM 角色。

CloudWatch 服務配額

CloudWatch 具有下列指標、警示、API 要求和警示電子郵件通知的配額。

> **Note**
>
> 對於某些 AWS 服務 CloudWatch，您可以使用使 CloudWatch 用量指標在 CloudWatch 圖形和儀表板上以視覺化方式呈現您目前的服務使用情況。您可以使用 CloudWatch 量度數學函數，在圖形上顯示這些資源的服務配額。您也可以設定警示，在您的用量接近服務配額時發出警示。如需詳細資訊，請參閱 視覺化您的服務配額和設定警示。

資源	預設配額
警示動作	5/警示。此配額無法變更。
警示評估期間	最大值為一天 (86,400 秒)，透過將警示週期乘以使用的評估週期數計算得出。此配額無法變更。
警示	每個客戶每月 10 次免費。額外警示會產生費用。 每個帳戶的警示總數沒有任何限制。 根據指標數學表達式的警示最多可有 10 個指標。 每個區域 200 個指標洞察警報。您可以要求增加配額。
異常偵測模型	每個帳戶每個區域 500 次。
API 請求	每個客戶每月 1,000,000 次免費。
Canary	每個帳戶每個區域 200 次。 您可以要求增加配額。
Contributor Insights API 請求	下列 API 的配額為每個區域每秒 20 筆交易 (TPS)。 • DescribeInsightRules 配額無法變更。

資源	預設配額
	• GetInsightRuleReport 您可以要求增加配額。 下列 API 的配額為每個區域 5 TPS。此配額無法變更。 • DeleteInsightRules • PutInsightRule 下列 API 的配額為每個區域 1 TPS。此配額無法變更。 • DisableInsightRules • EnableInsightRules
Contributor Insights 規則	每個帳戶每個區域 100 條規則。 您可以要求增加配額。
自訂指標	沒有配額。
儀表板	每個儀表板多達 500 個小工具。每個儀表板 Widget 多達 500 個指標。每個儀表板跨所有 Widget 最多 2500 個指標。 這些配額包括擷取以在指標數學函數中使用的所有指標，即使是沒有在圖形上顯示的指標也包含在其中。 無法變更這些配額。
DescribeAlarms	每個區域每秒 9 次交易 (TPS)。您每秒可以提出而不受限制的操作請求數量上限。 您可以要求增加配額。

資源	預設配額
DeleteAlarms請求 DescribeAlarmHistory請求 DisableAlarmActions請求 EnableAlarmActions請求 SetAlarmState請求	以上每個操作每個區域 3 TPS。您每秒可以提出而不受限制的操作請求數量上限。 無法變更這些配額。
DescribeAlarmsForMetric請求	每個區域 9 TPS。您每秒可以提出而不受限制的操作請求數量上限。 此配額無法變更。
DeleteDashboards請求 GetDashboard請求 ListDashboards請求 PutDashboard請求	以上每個操作每個區域 10 TPS。您每秒可以提出而不受限制的操作請求數量上限。 無法變更這些配額。
PutAnomalyDetector DescribeAnomalyDetectors	每個區域 10 TPS。您每秒可以提出而不受限制的操作請求數量上限。
DeleteAnomalyDetector	每個區域 5 TPS。您每秒可以提出而不受限制的操作請求數量上限。
維度	每個指標 30 個。此配額無法變更。

資源	預設配額
GetMetricData	每個區域 10 TPS，適用於包含 Metrics Insights 查詢的操作。對於不包含 Metrics Insights 查詢的操作，配額為每個區域 50 TPS。這是您每秒可以提出而不受限制的操作請求數量上限。您可以要求增加配額。 對於包含 Metrics Insights 查詢的 GetMetricData 操作，最近 3 小時內的配額為每秒 4,300,000 個資料點 (DPS)。這是根據查詢 (其中可包含不超過 10,000 個指標) 所掃描的資料點總數計算而來。 每秒 180,000 個資料點 (DPS)，如果從目前時間算起，在 API 請求中使用的 StartTime 少於或等於三小時。396,000 DPS，如果從目前時間算起，StartTime 超過三小時。這是您使用一或多個 API 呼叫且不受限制的每秒可要求的資料點數量上限。此配額無法變更。 DPS 是根據預估資料點 (而不是實際資料點) 所計算。資料點預估是使用請求的時間範圍、期間和保留期間進行計算。也就是說，如果請求指標中的實際資料點稀疏或空白，在預估資料點超過配額的情況下，調節仍會發生。DPS 配額是依每個區域規定的。
GetMetricData	單個 GetMetricData 呼叫可以包括以下內容： • 多達 500 個 MetricDataQuery 結構。 • 多達 100 個 SERVICE_QUOTA() 函數。 • 多達 100 個 SEARCH() 函數。 • 多達 5 個 LAMBDA() 函數。 這些配額無法變更。
GetMetricStatistics	每個區域 400 TPS。您每秒可以提出而不受限制的操作請求數量上限。 您可以要求增加配額。

資源	預設配額
GetMetricWidgetImage	每個影像最多 500 個指標。此配額無法變更。 每個區域 20 TPS。您每秒可以提出而不受限制的操作請求數量上限。 您可以要求增加配額。
ListMetrics	每個區域 25 TPS。您每秒可以提出而不受限制的操作請求數量上限。 您可以要求增加配額。
指標資料值	指標資料點的數值必須在 -2^360 到 2^360 的範圍內。不支援特殊值 (例如 NaN、+Infinity、-Infinity)。此配額無法變更。
MetricDatum項目	PutMetricData請求 MetricDatum物件可以包含單一值或代表許多值的StatisticSet物件。此配額無法變更。
指標	每個客戶每月 10 次免費。
Metrics Insights 查詢	單一查詢可處理不超過 10,000 個指標。這表示如果 SELECT (選取)、FROM (從) 和 WHERE (哪裡) 子句符合 10,000 個以上的指標時，查詢只會處理所找到這些指標中的前 10,000 個。 單一查詢可以傳回不超過 500 個時間序列。 您只能查詢最近三個小時的資料。
可觀察性存取管理員 (OAM) API 要求率。	每個區域提供 1 TPS，適用於 PutSinkPolicy. 每個區域為彼此的 CloudWatch OAM API 提供 10 個 TPS。 這些配額會反映您每秒可以提出而不受限制的操作請求數量上限。 這些配額無法變更。

資源	預設配額
OAM 來源帳戶連結	每個來源帳戶最多可連結至 5 個監視帳戶 此配額無法變更。
OAM 接收器	每個帳戶每個區域 1 個接收器 此配額無法變更。
PutCompositeAlarm請求	每個區域 3 TPS。您每秒可以提出而不受限制的操作請求數量上限。 您可以要求增加配額。
PutMetricAlarm請求	每個區域 3 TPS。您每秒可以提出而不受限制的操作請求數量上限。 您可以要求增加配額。
PutMetricData請求	1 MB (適用於 HTTP POST 請求)。PutMetricData每秒可處理 500 個交易 (TPS)，這是您每秒可以在不受限制的情況下發出的最大操作請求數量。 PutMetricData 每個請求可以處理 1,000 個指標。 您可以要求增加配額。
Amazon SNS 電子郵件通知	每個客戶每月 1,000 次免費。
Synthetics 群組	每個帳戶 20 個。 此配額無法變更。
TagResource	每個區域 20 TPS。您每秒可以提出而不受限制的操作請求數量上限。 此配額無法變更。

資源	預設配額
UntagResource	每個區域 20 TPS。您每秒可以提出而不受限制的操作請求數量上限。 此配額無法變更。

文件歷史紀錄

下表說明從 2018 年 6 月開始，每個版本的 Amazon CloudWatch 使用者指南中的重要變更。如需有關此文件更新的通知，您可以訂閱 RSS 訂閱源。

變更	描述	日期
CloudWatch 應用程式訊號服務對應支援金絲雀、RUM 用戶端和 AWS 服務相依性群組。	應用程式 Signals 預覽版本已在服務對應中新增預設分組，適用於金絲雀、RUM 用戶端和相同類型的 AWS 服務相依性。此變更可減少服務對應預設檢視中的圖示數目，以便於檢視和瀏覽。	2024年5月21 日
CloudWatchReadOnlyAccess IAM 政策已更新	CloudWatch 變更中的權限範圍CloudWatchReadOnlyAccess。原則的範圍新增了、和`application-signals:List*` 動作 `application-signals:BatchGet*` `application-signals:Get*` ，以便使用者可以使用 CloudWatch 應用程式信號來檢視、調查及診斷其服務健全狀況的問題。 CloudWatch還添加了一個`iam:GetRole` 動作，以便用戶可以檢查是否設置了應用程序信號。	2024年5月17日
CloudWatchFullAccessV2 IAM 政策已更新	CloudWatch 更改了 CloudWatchFullAccessV2 中權限的範圍。原則的範圍新增了，`application-signals:*` 讓使用者可以使用 CloudWatch 應用程式信號	2024年5月17日

	來檢視、調查及診斷其服務健康狀態的問題。	
Lambda 洞見支援 AWS GovCloud (美國東部) 和 AWS GovCloud (美國西部)	CloudWatch Lambda 洞見已新增對 AWS GovCloud (美國東部) 和 AWS GovCloud (美國西部) 區域的支援。	2024年4月29 日
CloudWatch 跨帳戶可觀察性支援資源篩選	您現在可以建立篩選器，以指定在帳戶之間建立連結時，從來源帳戶共用哪些度量命名空間和記錄群組至監視帳戶。	2024年4月26日
CloudWatch 應用信號更新	應用程式訊號預覽版本新增了三個功能。應用程序信號現在支持 Python 應用。它為 Amazon EKS 架構上的應用程式提供了更簡單的啟用程序。它包含新的設定，您可以用來管理所收集量度的基數。	2024年4月26日
CloudWatch 具有增強 Amazon EKS 可觀察性的容器洞見可以收集 E AWS lastic Fabric Adapter (EFA) 指標	您現在可以使用具有增強 Amazon EKS 可觀察性的 CloudWatch 容器洞見，從 Amazon EKS 叢集收集 AWS 彈性結構適配器 (EFA) 指標。	2024年4月23 日

更新的 IAM 政策	CloudWatch 更新了CloudWatchApplicationSignalsServiceRolePolicy策略。此原則中的logs:StartQuery 和logs:GetQueryResults 權限範圍已變更，以便在更多建築物上新增arn:aws:logs:*:*:log-group:/aws/appsignals/*:* 及啟"arn:aws:logs:*:*:log-group:/aws/application-signals/data:*" 用應用程式訊號。此原則會附加至AWSServiceRoleForCloudWatchApplicationSignals服務連結角色。	2024年4月18日
互聯網監控提供了一個全球互聯網天氣圖，以認證 AWS 客	Amazon CloudWatch 網際網路監控器現在會顯示全球網際網路天氣地圖，可在主控台中供所有經過驗證的 AWS 客戶使用。若要檢視地圖，請在 Amazon 主 CloudWatch 控台中導覽至網際網路監控器。	2024年4月16日
CloudWatch 具有增強 Amazon EKS 可觀察性的容器洞見可以收集 AWS 神經元指標	您現在可以使用具有增強的 Amazon EKS 可觀察性的 CloudWatch 容器洞見，從 Amazon EKS 叢集收集 AWS 神經元指標。	2024年4月16日

CloudWatch 應用程式訊號新增服務概觀索引標籤和更多指標，以協助診斷	新的服務概述索引標籤會顯示您的服務概觀，包括作業數量、相依性、合成和用戶端頁面。此索引標籤會顯示整個服務的關鍵指標，以及常用的作業和相依性。您現在也可以檢視與問題 (包括故障、錯誤和延遲問題) 相關的 X-Ray 追蹤。	2024年4月16日
CloudWatch 具有增強 Amazon EKS 可觀察性的容器洞見，增加了對 Windows 的支援	您現在可以使用具有增強 Amazon EKS 可觀察性的 CloudWatch 容器洞見，從 Amazon EKS 叢集上的 Windows 工作者節點收集指標。	2024年4月10日
CloudWatchApplicationSignalsServiceRolePolicyIAM 政策已更新	CloudWatch 變更中的權限範圍CloudWatchApplicationSignalsServiceRolePolicy。`cloudwatch:GetMetricData` 權限的範圍已變更為，以*便應用程式信號可以從連結帳戶中的來源擷取指標。	2024年4月8日
Amazon CloudWatch 互聯網監控器現在支持跨帳戶觀察	您現在可以使用 Internet Monitor 跨帳戶觀察能力來監視跨單一跨越多個應用 AWS 帳戶 程式。 AWS 區域	2024年3月29 日

CloudWatchAgentServerPolicy 和CloudWatchAgentAdminPolicy政策已更新	CloudWatch 已新增CloudWatchAgentServerPolicy 和CloudWatchAgentAdminPolicy 原則的權限，以允許 CloudWatch 代理程式發佈 X-Ray 追蹤並修改記錄群組保留期間。在這兩個原則中xray:PutTraceSegments xray:PutTelemetryRecords ，都新增了xray:GetSamplingRules xray:GetSamplingTargets 、、xray:GetSamplingStatisticSummaries 和logs:PutRetentionPolicy 權限	2024年2月12日
適用於 CloudWatch 網路監視器的新服務連結角色和 IAM 政策	CloudWatch 新增服務連結角色，稱為 AWSServiceRoleForNetworkMonitor. CloudWatch 新增此新服務連結角色，可讓您建立監視器，以擷取來源子網路和目標 IP 位址之間的網路指標。新的 CloudWatchNetworkMonitorServiceRolePolicyIAM 政策會附加至此角色，而且該政策授予代表您擷 CloudWatch 取網路指標的權限。	2023 年 12 月 22 日

CloudWatch 發布 Amazon CloudWatch 網絡監控	CloudWatch 發布了一項新功能，Amazon CloudWatch 網絡監控器。這是一項新的作用中網路監控服務，可識別網路或您自己的公司 AWS 網路中是否存在網路問題。	2023 年 12 月 22 日
CloudWatchReadOnlyAccess政策已更新	CloudWatch 新增了 CloudWatch Synthetics、X-Ray 和 R CloudWatch UM 的現有唯讀權限，以及 CloudWatch 應用程式信號的新唯讀權限，以CloudWatchReadOnlyAccess便具有此原則的使用者可以分類和診斷 CloudWatch 應用程式信號所報告的服務健康狀態問題。已新增`cloudwatch:GenerateQuery` 權限，以便具有此原則的使用者可以從自然語言提示中產生 CloudWatch 指標見解查詢字串。	2023 年 12 月 5 日
CloudWatchFullAccessV2 政策已更新	CloudWatch 為 CloudWatch Synthetics、X-Ray 和 R CloudWatch UM 新增了 CloudWatchFullAccessV2 的現有權限，並新增了 CloudWatch 應用程式訊號的新權限，以便具有此原則的使用者可以完整管理應用程式信號，以分類和診斷服務健康狀態的問題。	2023 年 12 月 5 日

新服務連結角色和新的 IAM 政策	CloudWatch 新增服務連結角色，稱為 AWSServiceRoleForCloudWatchApplicationSignals. CloudWatch 新增這個新的服務連結角色，可讓 CloudWatch 應用程式信號收集 CloudWatch 記錄資料、X-Ray 追蹤資料、 CloudWatch 指標資料，以及標記您已針對應用程式啟用「應用程式訊號」的應用 CloudWatch程式資料。新的 CloudWatchApplicationSignalsServiceRolePolicy IAM 政策已附加至此角色，而且該政策授予 CloudWatch 應用程式信號的權限，以便從其他相關 AWS 服務收集監控和標記資料。	2023 年 11 月 30 日
CloudWatch 啟動應用程式訊號的預覽版	CloudWatch 應用程式訊號處於預覽狀態。使用應用程式信號來檢測您的應用程式，以 AWS 便您可以監視目前的應用程式健康狀態、建立服務等級目標 (SLO)，以及根據業務目標追蹤長期應用程式效能。如需詳細資訊，請參閱 Application Signals。	2023 年 11 月 30 日
CloudWatch 添加了對查詢其他數據源的支持	您可以使用 CloudWatch 從其他資料來源查詢、視覺化和建立指標的警示。如需詳細資訊，請參閱查詢其他資料來源的指標。	2023 年 11 月 26 日

CloudWatch 指標見解支援自然語言查詢產生	CloudWatch 指標見解支援自然語言查詢，以產生和更新查詢。如需詳細資訊，請參閱使用自然語言產生和更新 CloudWatch 指標見解查詢。	2023 年 11 月 26 日
CloudWatch 發布具有增強 Amazon EKS 可觀察性的容器洞察	CloudWatch 發行了新版的容器見解。此版本支援 Amazon EKS 叢集的增強型可觀測性，並且可從執行於 Amazon EKS 的叢集中收集更詳細的指標。安裝之後，它會自動收集 Amazon EKS 叢集的詳細基礎設施遙測和容器日誌。然後，您可使用經策管且立即可用的儀表板，來深入了解應用程式和基礎設施遙測。	2023 年 11 月 6 日
CloudWatch 指標串流新增快速夥伴設定	CloudWatch 指標串流現在提供快速的合作夥伴設定選項，您可以使用此選項快速設定給某些第三方提供者的指標串流。	2023 年 10 月 17 日
CloudWatch 發布警報建議	CloudWatch Synthetics 現在可針對其他 AWS 服務的指標提供警示建議。這些建議可協助您識別應設定警示的指標，以遵循監控這些服務的最佳實務。	2023 年 10 月 16 日
CloudWatch Synthetics 版本運行時 -6.0 syn-nodejs-puppeteer	CloudWatch Synthetics 釋放運行時`syn-nodejs-puppeteer-6.0` 。	2023 年 9 月 26 日
為跨帳戶 CloudWatch 應用程式新增 Amazon 應用程式洞察支援	您現在可以跨帳戶界限共 CloudWatch 用應用程式見解應用程式。	2023 年 9 月 26 日

新服務連結角色和新的 IAM 政策	CloudWatch 新增服務連結角色，稱為 AWSServiceRoleForCloudWatchMetrics_DbPerfInsights. CloudWatch 新增這個新的服務連結角色，允許擷 CloudWatch 取 Performance Insights 指標，以進行警報、異常偵測和快照。新的 AWSServiceRoleForCloudWatchMetrics_DbPerfInsightsServiceRolePolicyIAM 政策會附加至此角色，而且該政策授予代表您擷 CloudWatch 取 Performance Insights 指標的權限。	2023 年 9 月 20 日
新增指標數學函數	CloudWatch 新增了新的指標數學函數DB_PERF_INSIGHTS ，您可以使用該函數從 AWS 資料庫服務擷取 Performance Insights 指標，以進行警報、異常偵測和快照。	2023 年 9 月 20 日
CloudWatchReadOnlyAccess政策已更新	CloudWatch 已新增application-autoscaling:DescribeScalingPolicies 權限，CloudWatchReadOnlyAccess讓具有此原則的使用者可以存取有關 Application Auto Scaling 原則的資訊。	2023 年 9 月 14 日
CloudWatch 代理添加了對 AL2023 的支持	此代 CloudWatch 理程式支援 AL2023。	2023 年 8 月 8 日

新的受管 IAM 政策，CloudWatchFullAccessV2	CloudWatch 添加了一個新的策略 CloudWatchFullAccessV2。此政策授予對 CloudWatch 動作和資源的完整存取權，同時更好地限定授與其他服務 (例如 Amazon SNS 和) 的許可範圍。 Amazon EC2 Auto Scaling	2023 年 8 月 1 日
已更新 Amazon CloudWatch 網際網路監視器的服務連結角色 — 更新至現有政策	將 `elasticloadbalancing:DescribeLoadBalancers` 和 `ec2:DescribeNetworkInterfaces` 許可新增至網路監視器的服務連結角色，支援監控特定 Network Load Balancer 資源的流量。	2023 年 7 月 25 日
增加了對 Amazon CloudWatch 互聯網監控網絡負載均衡資源的支持	新支援使用特定 Network Load Balancer 資源，在網路監視器中建立監視器，為您的應用程式提供更精細的可觀測性。	2023 年 7 月 25 日
儀表板變數功能	CloudWatch 已發行的儀表板變數，您可以使用這些變數建立彈性儀表板，根據您在儀表板中設定一個輸入欄位的方式，快速顯示不同的內容。例如，您可以建立一個儀表板，以便在不同的 Lambda 函數或 Amazon EC2 執行個體 ID 之間快速切換，或切換至不同 AWS 區域的執行個體 ID。如需詳細資訊，請參閱使用儀表板變數建立彈性儀表板。	2023 年 6 月 28 日

網路監視器現在支援自訂運作狀態事件的閾值	網路監視器新增了在全域效能分數或可用性分數觸發運作狀態事件時自訂閾值的功能。如需詳細資訊，請參閱追蹤 Amazon CloudWatch 網際網路監控中的即時效能和可用性。	2023 年 6 月 26 日
網路監視器現在支援所有商業區域	互聯網監控增加了七個新的 AWS 區域 ，現在支持所有的商業區域。	2023 年 6 月 19 日
新的 Lambda Insights 延伸版本	CloudWatch 為 X86-64 平台和 ARM64 平台添加了 Lambda 見解擴展的 1.0.229.0 版本。如需詳細資訊，請參閱 Lambda Insights 延伸的可用版本。	2023 年 6 月 12 日
CloudWatchReadOnlyAccess政策已更新	CloudWatch 將權限添加到 CloudWatchReadOnlyAccess. `logs:StartLiveTail` 和`logs:StopLiveTail` 權限已新增，讓具有此原則的使用者可以使用主控台來啟動和停止 CloudWatch 記錄即時尾端工作階段。如需詳細資訊，請參閱使用 Live Tail 以近乎即時的方式檢視日誌。	2023 年 6 月 6 日

CloudWatch RUM 增加了對自訂指標的支援	您可以使用 CloudWatch RUM 應用程序監視器來創建自定義指標並將其發送到 CloudWatch 並 CloudWatch 顯而易見。此功能包含 R AmazonCloudWatchUM ServiceRolePolicy 受管 IAM 政策的更新。在該原則中，條件索引鍵已變更，以便 CloudWatch RUM 可以將自訂指標傳送至自訂指標命名空間。	2023 年 2 月 9 日
新的和更新的管理政策 CloudWatch	為了支援 CloudWatch 跨帳戶可觀察性，`CloudWatchFullAccess` 和`CloudWatchReadOnlyAccess` 政策已更新，且已新增下列新的受管理政策：`CloudWatchCrossAccountSharingConfiguration` `OAMFullAccess` 、和。`OAMReadOnlyAccess` 如需詳細資訊，請參閱AWS 受管理策略的更CloudWatch 新。	2023 年 2 月 7 日
CloudWatch 應用程式深入解析服務連結角色原則更新 — 更新至現有原則。	CloudWatch 應用程式深入解析更新現有的 AWS 服務連結角色原則	2022 年 12 月 19 日
Amazon CloudWatch 應用程式洞見從容器洞察主控台支援容器化應用程式和微服務。	您可以在容器洞見儀表板上顯示針對 Amazon ECS 和 Amazon EKS 偵測到的 CloudWatch 應用程式洞見問題。	2021 年 11 月 17 日

監控 SAP HANA 資料庫的 Amazon CloudWatch 應用程式洞察。	您可以使用 Application Insights 來監控 SAP HANA 資料庫。	2021 年 11 月 15 日
Amazon CloudWatch 應用程式洞察支援，可監控帳戶中的所有資源。	您可以上架並監控帳戶中的所有資源。	2021 年 9 月 15 日
Amazon FSx 的 Amazon CloudWatch 應用程式洞察支援。	您可以監控從 Amazon FSx 擷取的指標。	2021 年 8 月 31 日
不再支援 SDK Metrics。	CloudWatch 不再支援 SDK 量度。	2021 年 8 月 25 日
CloudWatch 用於設定容器監控的 Amazon 應用程式洞察支援。	您可以使用 Amazon CloudWatch 應用程式洞察，使用最佳實務監控容器。	2021 年 5 月 18 日
指標串流已全面推出	您可以使用指標串流，持續將 CloudWatch 指標串流至您選擇的目的地。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的指標串流。	2021 年 3 月 31 日
針對 Amazon RDS 和 Amazon EC2 上甲骨文數據庫進行亞馬遜 CloudWatch 應用程序洞察	您可以使 CloudWatch 用 Amazon 應用程式洞察來監控從 Oracle 擷取的指標和日誌。	2021 年 1 月 16 日
Lambda Insights 已全面推出	CloudWatch Lambda 洞察是一種監控和故障排除解決方案，適用於在上 AWS Lambda執行的無伺服 如需詳細資訊，請參閱 Amazon 使用 CloudWatch 者指南中的使用 Lambda 洞察。	2020 年 12 月 3 日

針對 Prometheus JMX 匯出器指標進行 Amazon CloudWatch 應用程式洞察監控。	您可以使用 Amazon 應用程式洞察來監控從 Prometheus JMX 匯出器擷取的指標。CloudWatch	2020 年 11 月 20 日
CloudWatch Synthetics 發布新的運行時版本	CloudWatch Synthetics 發布了一個新的運行時版本。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的 Canary 執行階段版本。	2020 年 9 月 11 日
在 Amazon RDS 和 Amazon Amazon EC2 上監控 SQL 的亞馬遜 CloudWatch 應用程序洞察。	您可以監控使用在 Amazon RDS 或 Amazon EC2 上執行的 PostgreSQL 建置的應用程式。	2020 年 9 月 11 日
CloudWatch 支援儀表板共用	您現在可以與組織和 AWS 帳戶以外的人員共用 CloudWatch 儀表板。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的共用 CloudWatch 儀表板。	2020 年 9 月 10 日
透過應用程式深入解析，在後端使用 SQL Server 設定 .NET 應用 CloudWatch 程式的監視器	您可以使用文件教學課程，協助您在後端使用 SQL Server 與 CloudWatch 應用程式深入解析來設定 .NET 應用程式的監視器。	2020 年 8 月 19 日
AWS CloudFormation 支援 Amazon 應用程式洞察 CloudWatch 應用程式。	您可以直接從 AWS CloudFormation 範本將「CloudWatch 應用程式見解」監視 (包括關鍵指標和遙測) 新增至應用程式、資料庫和 Web 伺服器。	2020 年 7 月 30 日

針對 MySQL 資料庫叢集的 Aurora 監控 Amazon CloudWatch 應用程式洞見。	您可以使 CloudWatch 用 Amazon 應用程式洞察來監控 MySQL 資料庫叢集 (RDS Aurora) 的 Aurora。	2020 年 7 月 2 日
CloudWatch 貢獻者洞察一般可用性	CloudWatch 貢獻者見解現已正式推出。它可讓您分析日誌資料，以及建立顯示參與者資料的時間序列。您可以查看與前 N 個參與者有關的指標、唯一參與者的總數及其用量。如需詳細資訊，請參閱 Amazon 使用者指南中的使用參與者洞察分析高基數資料。 CloudWatch	2020 年 4 月 2 日
CloudWatch Synthetics 公開預覽	CloudWatch Synthetics 現已公開預覽。它可以讓您建立 Canary 來監控端點和 API。如需詳細資訊，請參閱 Amazon 使用 CloudWatch 者指南中的使用金絲雀。	2019 年 11 月 25 日
CloudWatch 貢獻者洞察公開預覽	CloudWatch 貢獻者深入解析現已開放公開預覽。它可讓您分析日誌資料，以及建立顯示參與者資料的時間序列。您可以查看與前 N 個參與者有關的指標、唯一參與者的總數及其用量。如需詳細資訊，請參閱 Amazon 使用者指南中的使用參與者洞察分析高基數資料。 CloudWatch	2019 年 11 月 25 日

CloudWatch 啟動 ServiceLens 功能	ServiceLens 透過將追蹤、指標、記錄和警示整合到單一位置，藉此發揮服務和應用程式的可觀察性。 ServiceLens CloudWatch 與整合 AWS X-Ray 以提供應用程式的 end-to-end 檢視。	2019 年 11 月 21 日
用 CloudWatch 於主動管理您的 AWS 服務配額	您可以用 CloudWatch 來主動管理 AWS 服務配額。 CloudWatch 使用量度可讓您了解帳戶對資源和 API 操作的使用情況。如需詳細資訊，請參閱 Amazon 使用者指南中的 Service Quotas 整合和使 CloudWatch 用指標。	2019 年 11 月 19 日
CloudWatch 當警示變更狀態時傳送事件	CloudWatch 現在， EventBridge 當任何 CloudWatch 警報狀態改變時，向 Amazon 發送事件。如需詳細資訊，請參閱警示事件和 Amazon 使 CloudWatch 用者指南 EventBridge中的。	2019 年 10 月 8 日
Container Insights	CloudWatch 容器深入解析現已正式推出。它可讓您從容器化應用程式和微型服務收集、彙總和總結指標和日誌。如需詳細資訊，請參閱 Amazon 使用 CloudWatch 者指南中的使用容器深入解析。	2019 年 8 月 30 日

Amazon EKS 和 Kubernetes 上 Container Insights 預覽指標的更新	Amazon EKS 和 Kubernetes 上 Container Insights 的公開預覽已更新。 InstanceId 現在已包含為叢集 EC2 執行個體的維度。這可讓在這些指標上建立的警示觸發下列 EC2 動作：停止、終止、重新開機和復原。此外，Pod 和服務指標現在會由 Kubernetes 命名空間報告，簡化透過命名空間監控及警示指標。	2019 年 8 月 19 日
AWS Systems Manager OpsCenter 整合的更新	CloudWatch 應用程式洞察如何與 Systems Manager 整合的更新 OpsCenter。	2019 年 8 月 7 日
CloudWatch 使用量度	CloudWatch 使用量度可協助您追蹤資 CloudWatch 源的使用情況，並保持在服務限制範圍內。如需詳細資訊，請參閱 https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch-Usage-Metrics.html。	2019 年 8 月 6 日
CloudWatch 容器洞察公開預覽	CloudWatch 容器深入解析現已開放公開預覽。它可讓您從容器化應用程式和微型服務收集、彙總和總結指標和日誌。如需詳細資訊，請參閱 Amazon 使用 CloudWatch 者指南中的使用容器深入解析。	2019 年 7 月 9 日

CloudWatch 異常偵測公開預覽	CloudWatch 異常偵測現已開放公開預覽。 CloudWatch 將機器學習演算法套用至量度的過去資料，以建立量度預期值的模型。您可以使用此模型來進行視覺化和設定警示。如需詳細資訊，請參閱 Amazon 使用 CloudWatch 者指南中的使用 CloudWatch 異常偵測。	2019 年 7 月 9 日
CloudWatch .NET 和 SQL 伺服器的應用程式洞察	CloudWatch 適用於 .NET 和 SQL 伺服器的應用程式洞察有助於觀察 .NET 和 SQL 伺服器應用程式。它可協助您設定應用程式資源的最佳監控，以持續分析資料的應用程式問題跡象。	2019 年 6 月 21 日
CloudWatch 代理部分重組	已重新撰寫 CloudWatch 代理程式文件以提高清晰度，特別是對於使用命令列安裝和設定代理程式的客戶而言。如需詳細資訊，請參閱 Amazon 使用者指南中的使用 CloudWatch 代理程式從 Amazon EC2 執行個體和現場部署伺服器收集 CloudWatch 指標和日誌。	2019 年 3 月 28 日
已將 SEARCH 函數新增至指標數學運算式	您現在可以在指標數學表達式中使用 SEARCH 函數。這可讓您建立儀表板，在建立符合搜尋查詢的新資源時自動更新。如需詳細資訊，請參閱 Amazon 使用 CloudWatch 者指南中的在圖形中使用搜尋運算式。	2019 年 3 月 21 日

AWS 企業 Support 的 SDK 指標	SDK 指標可協助您評估 AWS 服務的運作狀態，並診斷因達到帳戶使用限制或服務中斷而造成的延遲。如需詳細資訊，請參閱 Amazon 使用 CloudWatch 者指南中的使用 AWS SDK 指標監控應用程式。	2018 年 12 月 11 日
數學運算式的警示	CloudWatch 支援根據度量數學運算式建立警示。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的數學運算式警示。	2018 年 11 月 20 日
新 CloudWatch 主控台首頁	Amazon 在 CloudWatch 主控台中建立了新的首頁，該首頁會自動顯示您正在使用的所有 AWS 服務的關鍵指標和警示。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南 CloudWatch中的 Amazon 入門。	2018 年 11 月 19 日
AWS CloudFormation CloudWatch 代理程式的範本	Amazon 已上傳可用於安裝和更新 CloudWatch代理程式的 AWS CloudFormation 範本。如需詳細資訊，請參閱 Amazon 使用 CloudWatch 者指南中的使用 AWS CloudFormation在新執行個體上安裝 CloudWatch 代理程式。	2018 年 11 月 9 日

CloudWatch 代理程式的增強功能	CloudWatch 代理程式已更新，可與 StatsD 和集合通訊協定一起使用。其中對跨帳戶的支援也獲得改善。有關詳情，請參閱 Amazon CloudWatch 使用者指南中的使用 StatsD 擷取自訂指標、使用 collectd 擷取自訂指標以及將指標和日誌傳送到其他 AWS 帳戶。	2018 年 9 月 28 日
支援 Amazon VPC 端點	您現在可以在 VPC 和 CloudWatch. 之間建立私人連線。如需詳細資訊，請參閱 Amazon 使 CloudWatch用 CloudWatch 者指南中的與介面 VPC 端點搭配使用。	2018 年 6 月 28 日

下表說明 2018 年 6 月之前對 Amazon CloudWatch 使用者指南進行的重要變更。

變更	描述	發行日期
指標數學	您現在可以對 CloudWatch 量度執行數學運算式，產生新的時間序列，您可以新增至儀表板上的圖形。如需詳細資訊，請參閱 使用指標數學。	2018 年 4 月 4 日
「M / N」警示	您現在可以將警示設定為根據在任何警示評估間隔「M / N」資料點來進行觸發。如需詳細資訊，請參閱 評估警示。	2017 年 12 月 8 日
CloudWatch 代理	新的統一 CloudWatch 代理程式已發佈。您可以使用整合多平台代理程式，從 Amazon EC2 執行個體和現場部署伺服器收集自訂的系統指標和日誌檔案。新的代理程式支援 Windows 和 Linux，並可自訂收集的指標，包括子資源指標 (例如，每個 CPU 核心)。如需詳細資訊，請參閱 使用 CloudWatch 代理程式收集指標、記錄和追蹤。	2017 年 9 月 7 日

變更	描述	發行日期
NAT 閘道指標	新增了 Amazon VPC NAT 閘道的指標。	2017 年 9 月 7 日
高解析度指標	您現在可以選擇性將自訂指標設定為高解析度指標，最短可達一秒間隔的精細度。如需詳細資訊，請參閱 高解析度指標。	2017 年 7 月 26 日
儀表板 API	您現在可以使用 API 和 AWS CLI建立、修改和刪除儀表板。如需詳細資訊，請參閱 建立 CloudWatch 儀表板。	2017 年 7 月 6 日
AWS Direct Connect 度量	新增的量度 AWS Direct Connect。	2017 年 6 月 29 日
Amazon VPC VPN 指標	新增了 Amazon VPC VPN 的指標。	2017 年 5 月 15 日
AppStream 2.0 個指標	添加了 AppStream 2.0 的指標。	2017 年 3 月 8 日
CloudWatch 控制台顏色選擇	您現在可以選擇每個指標在儀表板 widget 上的顏色。如需詳細資訊，請參閱 在 CloudWatch 儀表板上編輯圖形 。	2017 年 2 月 27 日
儀表板上的警示	現在可以將警示新增至儀表板。如需詳細資訊，請參閱 從 CloudWatch 儀表板新增或移除警報小工具 。	2017 年 2 月 15 日
新增了 Amazon Polly 的指標	新增了 Amazon Polly 的指標。	2016 年 12 月 1 日
新增了 Amazon Managed Service for Apache Flink 的指標	新增了 Amazon Managed Service for Apache Flink 的指標。	2016 年 12 月 1 日

變更	描述	發行日期
新增支援百分位數統計資料	您可以指定任何百分位數，最多使用兩位小數 (例如，p95.45)。如需詳細資訊，請參閱 百分位數。	2016 年 11 月 17 日
新增了 Amazon Simple Email Service 的指標	新增了 Amazon Simple Email Service 的指標。	2016 年 11 月 2 日
更新指標保留	Amazon CloudWatch 現在會保留指標資料 15 個月，而不是 14 天。	2016 年 11 月 1 日
更新指標主控台介面	主 CloudWatch 控台已更新，改善了現有功能和新功能。	2016 年 11 月 1 日
新增了 Amazon Elastic Transcoder 的指標	新增了 Amazon Elastic Transcoder 的指標。	2016 年 9 月 20 日
新增了 Amazon API Gateway 的指標	新增了 Amazon API Gateway 的指標。	2016 年 9 月 9 日
新增的量度 AWS Key Management Service	新增的量度 AWS Key Management Service。	2016 年 9 月 9 日
新增了由 Elastic Load Balancing 支援的全新 Application Load Balancer 的指標	新增了 Application Load Balancers 的指標。	2016 年 8 月 11 日

變更	描述	發行日期
為 Amazon EC2 添加了新 NetworkPacketsIn 的 NetworkPacketsOut 指標	為 Amazon EC2 添加了新 NetworkPacketsIn 的 NetworkPacketsOut 指標。	2016 年 3 月 23 日
新增了 Amazon EC2 Spot 機群的新指標	新增了 Amazon EC2 Spot 機群的新指標。	2016 年 3 月 21 日
添加了新的 CloudWatch 日誌指標	添加了新的 CloudWatch 日誌指標。	2016 年 3 月 10 日
添加了 Amazon OpenSearch 服務以及 AWS WAF 指標和維度	添加了 Amazon OpenSearch 服務以及 AWS WAF 指標和維度。	2015 年 10 月 14 日
新增對 CloudWatch 儀表板的支援	儀表板是 CloudWatch 主控台中可自訂的首頁，您可以使用它們在單一檢視中監視資源，即使是分散在不同區域的資源也是如此。如需詳細資訊，請參閱 使用 Amazon CloudWatch 儀表。	2015 年 10 月 8 日
新增 AWS Lambda 量度和維度	新增 AWS Lambda 量度和維度。	2015 年 9 月 4 日
新增了 Amazon Elastic Container Service 指標與維度	新增了 Amazon Elastic Container Service 指標與維度。	2015 年 8 月 17 日

變更	描述	發行日期
新增了 Amazon Simple Storage Service 指標與維度	新增了 Amazon Simple Storage Service 指標與維度。	2015 年 7 月 26 日
新功能：重啟警示動作	新增重新啟動警示動作和用於警示動作的新 IAM 角色。如需詳細資訊，請參閱 建立警示以停止、終止、重新啟動或復原 EC2 執行個體。	2015 年 7 月 23 日
添加了 Amazon WorkSpaces 指標和維度	添加了 Amazon WorkSpaces 指標和維度。	2015 年 4 月 30 日
新增了 Amazon Machine Learning 指標與維度	新增了 Amazon Machine Learning 指標與維度。	2015 年 4 月 9 日
新功能：Amazon EC2 執行個體復原警示動作	更新警示動作，包含新的 EC2 執行個體復原動作。如需詳細資訊，請參閱 建立警示以停止、終止、重新啟動或復原 EC2 執行個體。	2015 年 3 月 12 日
添加了 Amazon CloudFront 和 Amazon CloudSearch 指標和維度	添加了 Amazon CloudFront 和 Amazon CloudSearch 指標和維度。	2015 年 3 月 6 日
新增了 Amazon Simple Workflow Service 指標與維度	新增了 Amazon Simple Workflow Service 指標與維度。	2014 年 5 月 9 日

變更	描述	發行日期
更新指南添加支持 AWS CloudTrail	添加了一個新主題來解釋如 AWS CloudTrail 何使用在 Amazon 中記錄活動 CloudWatch。如需詳細資訊，請參閱 記錄 Amazon CloudWatch API 呼叫 AWS CloudTrail。	2014 年 4 月 30 日
更新了使用新的指南 AWS Command Line Interface （AWS CLI）	AWS CLI 是跨服務 CLI，具有簡化的安裝、統一的組態和一致的命令列語法。在 Linux /Unix、視窗和 Mac 上都支援 AWS CLI。本指南中的 CLI 範例已更新為使用新的 AWS CLI。 如需有關如何安裝和設定新 AWS CLI 的資訊，請參閱《AWS Command Line Interface 使用者指南》中的使用 AWS CLI 介面進行設定。	2014 年 2 月 21 日
添加了 Amazon Redshift 以及 AWS OpsWorks 指標和維度	添加了 Amazon Redshift 以及 AWS OpsWorks 指標和維度。	2013 年 7 月 16 日
新增了 Amazon Route 53 指標與維度	新增了 Amazon Route 53 指標與維度。	2013 年 6 月 26 日
新功能：Amazon CloudWatch 警報操作	新增區段以記錄 Amazon CloudWatch 警示動作，您可以使用這些動作停止或終止 Amazon 彈性運算雲端執行個體。如需詳細資訊，請參閱 建立警示以停止、終止、重新啟動或復原 EC2 執行個體。	2013 年 1 月 8 日
更新 EBS 指標	更新 EBS 指標以包含佈建 IOPS 磁碟區的兩個新指標。	2012 年 11 月 20 日
新的帳單提醒	您現在可以使用 Amazon 指 CloudWatch 標監控 AWS 費用，並建立警示，以在超過指定閾值時通知您。如需詳細資訊，請參閱 建立帳單警示以監控您的估計 AWS 費用。	2012 年 5 月 10 日

變更	描述	發行日期
新指標	您現在可以存取六個新 Elastic Load Balancing 指標，為各種 HTTP 回應代碼提供計數。	2011 年 10 月 19 日
新功能	您現在可以從 Amazon EMR 存取指標。	2011 年 6 月 30 日
新功能	您現在可以從 Amazon Simple Notification Service 和 Amazon Simple Queue Service 存取指標。	2011 年 7 月 14 日
新功能	新增有關使用 `PutMetricData` API 發佈自訂指標的資訊。如需詳細資訊，請參閱 發佈 自訂指標。	2011 年 5 月 10 日
更新指標保留	Amazon CloudWatch 現在保留了警報的歷史兩週而不是六週。透過這項改變，警示的保留期可匹配指標資料的保留期。	2011 年 4 月 7 日
新功能	新增了可在指標超過閾值時，傳送 Amazon Simple Notification Service 或 Auto Scaling 通知的功能。如需詳細資訊，請參閱 警示。	2010 年 12 月 2 日
新功能	現在有許多 CloudWatch 動作包括 MaxRecords 和 NextToken 參數，可讓您控制要顯示的結果頁面。	2010 年 12 月 2 日
新功能	此服務現在與 AWS Identity and Access Management (IAM) 整合。	2010 年 12 月 2 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。