使用者指南

AWS 大型主機現代化



Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS 大型主機現代化: 使用者指南

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

什麼是 AWS 大型主機現代化 1
AWS Mainframe Modernization 的功能 1
模式 2
如何開始使用 AWS 大型主機現代化 3
相關服務 3
存取 AWS 大型主機現代化 4
您是第一次使用 AWS 大型主機現代化嗎？ 4
AWS 大型主機現代化定價 4
設定 AWS 大型主機現代化 5
註冊 AWS 帳戶 5
建立具有管理存取權的使用者 5
概念 7
應用程式 7
應用程式定義 7
批次任務 8
組態 8
資料集 9
環境 9
大型主機現代化 9
遷移旅程 9
掛載點 9
自動化重構 9
平台轉換 9
資源 9
執行期引擎 10
現代化方法 11
評估階段 11
調動階段 11
遷移和現代化階段 12
操作和最佳化階段 12
開始使用 13
教學課程：為 AWS 藍年期設定受管執行時間 13
先決條件 14
步驟 1：上傳示範應用程式 14

步驟 2：建立應用程式定義 14
步驟 3：建立執行時間環境 15
步驟 4：建立應用程式 19
步驟 5：部署應用程式 21
步驟 6：啟動應用程式 23
步驟 7：存取應用程式 23
步驟 8：測試應用程式 24
清除資源 25
教學課程：設定 Rocket 軟體的受管執行期 25
先決條件 26
步驟 1：建立和載入 Amazon S3 儲存貯體 27
步驟 2：建立和設定資料庫 28
步驟 3：建立和設定 AWS KMS key 30
步驟 4：建立和設定 AWS Secrets Manager 資料庫秘密 31
步驟 5：將 sslMode 新增至秘密 32
步驟 6：建立執行時間環境 32
步驟 7：建立應用程式 37
步驟 8：部署應用程式 43
步驟 9：匯入資料集 45
步驟 10：啟動應用程式 51
步驟 11：連線至 CardDemo CICS 應用程式 52
清除資源 59
後續步驟 60
元件生命週期 61
元件生命週期概觀 61
版本升級 62
AWS Mainframe Modernization Refactor with AWS Blu Age 版本概觀 62
受管應用程式 64
為遷移的應用程式建立 AWS 資源 65
所需的許可 65
Amazon S3 儲存貯體 65
資料庫 66
AWS Key Management Service 金鑰 66
AWS Secrets Manager 秘密 67
建立應用程式 67
建立應用程式 67

部署應用程式 68
部署應用程式 69
更新應用程式 70
更新應用程式 70
刪除應用程式 70
刪除應用程式 71
提交應用程式的批次任務 71
提交批次任務 72
重新啟動批次任務 72
取消應用程式的批次任務 73
取消批次任務 74
匯入應用程式的資料集 74
匯入資料集 74
匯出應用程式的資料集 75
匯出資料集 75
管理應用程式的交易 76
管理應用程式的交易 76
設定 Rocket Software 受管應用程式 77
Rocket 軟體支援的第三方整合 77
設定 AWS Blu Age 受管應用程式 79
AWS Blu Age 受管應用程式的結構 79
設定受管應用程式的公用程式存取權 81
設定受管應用程式的其他屬性 91
應用程式定義參考 111
一般標頭區段 112
定義區段概觀 113
AWS Blu Age 應用程式定義範例 113
AWS Blu Age 定義詳細資訊 114
Rocket 軟體應用程式定義 119
Rocket 軟體定義詳細資訊 121
資料集定義參考 131
一般屬性 132
VSAM 的資料集請求格式範例 133
GDG 基礎的資料集請求格式範例 135
PS 或 GDG 世代的範例資料集請求格式 136
PO 的資料集請求格式範例 137

受管執行期環境 139
建立執行時間環境 139
建立執行時間環境 140
更新執行時間環境 142
更新執行時間環境 143
Maintenance window (維護時段) 143
停止執行期環境 144
停止執行期環境 144
重新啟動執行時間環境 145
重新啟動執行時間環境 145
刪除執行時間環境 145
刪除執行時間環境 145
應用程式測試 147
什麼是應用程式測試 147
您是第一次使用應用程式測試嗎？ 148
應用程式測試的優點 148
與 整合 AWS CloudFormation 148
應用程式測試的運作方式 149
相關服務 3
存取應用程式測試 150
應用程式測試的定價 150
應用程式測試概念 151
測試案例 151
測試套件 152
測試環境組態 152
上傳 152
重新播放 152
Compare 153
資料庫比較 153
資料集比較 153
比較狀態 153
對等規則 154
最終狀態資料集比較 154
狀態進展資料庫比較 154
功能對等 (FE) 154
線上 3270 螢幕比較 155

重播資料 155
參考資料 155
上傳、重播和比較 155
差異 156
對等 156
來源應用程式 156
目標應用程式 156
應用程式測試先決條件 156
應用程式測試中的主控台工作流程 157
在應用程式測試中建立測試案例 157
在應用程式測試中建立測試套件 159
在應用程式測試中建立測試環境組態 161
教學課程：在應用程式測試中設定 CardDemo 應用程式 163
先決條件 163
步驟 1：準備設定 CardDemo 163
步驟 2：建立所有必要的資源 164
步驟 3：部署並啟動應用程式 164
步驟 4：匯入初始資料 165
步驟 5：連線至 CardDemo 應用程式 165
教學課程：使用 CardDemo 重播和比較 AWS 藍時期 167
步驟 1：取得 AWS 藍時 Amazon EC2 Amazon Machine Image (AMI) 167
步驟 2：使用 AWS Blu Age AMI 啟動 Amazon EC2 執行個體 167
步驟 3：將 CardDemo 相依檔案上傳至 S3 169
步驟 4：載入資料庫並初始化 CardDemo 應用程式 169
步驟 5：啟動 AWS Blu Age 執行期 CloudFormation 171
步驟 6：測試 AWS Blu Age Amazon EC2 執行個體 174
步驟 7：驗證先前步驟是否正確完成 175
步驟 8：建立測試案例 175
步驟 9：建立測試套件 176
步驟 10：建立測試環境組態 176
步驟 11：在測試套件中上傳輸入資料 176
步驟 12：重播並比較 177
應用程式測試中支援的資料集程式碼頁面 177
應用程式測試中的資料保護 187
AWS 大型主機現代化應用程式測試收集的資料 188
AWS 大型主機現代化應用程式測試的靜態資料加密 189

建立客戶受管金鑰 190
為 AWS 大型主機現代化應用程式測試指定客戶受管金鑰 191
AWS Mainframe Modernization Application Testing 加密內容 191
監控加密金鑰 192
傳輸中加密 192
應用程式測試如何與 IAM 搭配使用 192
身分型政策 193
資源型政策 193
政策動作 194
政策資源 195
政策條件索引鍵 196
ACL 197
ABAC 197
臨時憑證 197
轉送存取工作階段 198
服務角色 198
服務連結角色 198
AWS 藍光年齡重構 200
AWS 藍卡年齡版本 201
AWS Blu Age 版本控制 201
AWS Blu Age 執行時間選項 202
AWS Blu Age 版本備註 205
AWS Blu Age 安全性漏洞 298
升級 AWS 藍髮年齡 299
AWS 藍卡存留期生命週期 301
AWS Blu Age 執行期概念 302
高階架構 302
現代化應用程式結構 306
了解資料簡化程式 340
AWS 藍本年齡藍調 347
公用程式 Web 應用程式中的可用程式 367
Blusam 管理主控台 369
AWS Blu Age 執行期組態 405
應用程式組態基本概念 405
應用程式優先順序 407
資料庫的 JNDI 407

AWS Blu Age 執行期秘密 408
其他檔案 (groovy、sql 等） 419
其他 Web 應用程式 420
啟用屬性 420
可用的 Redis 快取屬性 478
設定 Gapwalk 應用程式的安全性 492
AWS Blu Age 執行時間 APIs 508
建置 URLs端點 508
Gapwalk 應用程式的端點 509
Blusam 應用程式主控台 REST 端點 527
管理 JICS 應用程式主控台 549
資料結構 568
設定 AWS Blu Age 執行期 （非受管） 577
AWS Blu Age 執行期先決條件 577
加入 AWS 藍卡年齡執行期 578
基礎設施設定要求 583
AWS Blu Age 執行期成品 589
在 Amazon EC2 上部署 AWS Blu Age 執行期 591
在 Amazon ECS 和 Amazon EKS 上部署 AWS Blu Age 執行期 602
測試 PlanetsDemo 應用程式 609
使用 Blu Age Developer IDE 修改原始程式碼 613
教學課程：為 AWS 藍年開發人員 IDE 設定 AppStream 2.0 613
教學課程：在 AppStream 2.0 上使用 AWS Blu Age Developer 618
AWS 藍年常見問答集 634
一般 634
AWS Blu Age 執行時間 636
資料 642
轉換 643
部署 644
火箭軟體複寫 647
設定 Rocket 軟體 （在 Amazon EC2 上） 647
Rocket Software （在 Amazon EC2 上） 先決條件 648
建立 Amazon S3 的 Amazon VPC 端點 648
請求帳戶的允許清單更新 650
建立 AWS Identity and Access Management 角色 651
授予 License Manager 所需的許可 658

訂閱 Amazon Machine Image 659
啟動 Rocket 軟體執行個體 663
沒有網際網路存取的子網路或 VPC 668
設定 AppStream 2.0 自動化 675
在工作階段開始時設定自動化 675
在工作階段結束時設定自動化 676
在企業開發人員中將資料集檢視為資料表 676
先決條件 677
步驟 1：設定連至 Rocket Software 資料存放區的 ODBC 連線 (Amazon RDS 資料庫） 677
步驟 2：建立 MFDBFH.cfg 檔案 679
步驟 3：為您的複製手冊配置建立結構 (STR) 檔案 680
步驟 4：使用 結構 (STR) 檔案建立資料庫檢視 682
步驟 5：將 Rocket Software （ 先前稱為 Micro Focus) 資料集檢視為資料表和資料欄 683
在企業開發人員中使用資料檔案工具編輯資料集 684
先決條件 684
啟動 Rocket Software （ 先前稱為 Micro Focus) 資料檔案工具 684
編輯存放在 MFDBFH 資料庫中的 VSAM 資料集 685
編輯存放在 MFDBFH 資料庫中的非 VSAM 資料集 689
編輯存放在檔案系統中的 VSAM 和非 VSAM 資料集 (EFS/FSx) 691
火箭軟體教學課程 691
教學課程：設定 BankDemo 範例應用程式的建置 691
教學課程：使用 Rocket Enterprise Developer 設定CI/CD管道 701
教學課程：設定適用於 Enterprise Analyzer 和 Enterprise Developer 的 AppStream 2.0 724
教學課程：搭配 Rocket Enterprise Developer 使用 範本 733
教學課程：設定 Enterprise Analyzer 743
教學課程：設定企業開發人員 753
批次公用程式 758
二進位位置 758
M2SFTP 批次公用程式 759
M2WAIT 批次公用程式 765
TXT2PDF 批次公用程式 767
M2DFUTIL 批次公用程式 773
M2RUNCMD 批次公用程式 779
檔案傳輸 783
什麼是檔案傳輸 783
AWS Mainframe Modernization File Transfer 的優點 783

AWS Mainframe Modernization File Transfer 的運作方式 784
安裝檔案傳輸代理程式 785
步驟 1：為 M2-agent程式建立 zFS 資料集 785
步驟 2：將資料集格式化為 zFS 785
步驟 3：掛載 檔案系統 786
步驟 4：驗證掛載 786
步驟 5：輸入 OMVS 786
步驟 6：設定代理程式安裝目錄環境變數 786
步驟 7：設定工作目錄環境變數 787
步驟 8：建立工作目錄 787
步驟 9：複製代理程式 tar 檔案並複製工作目錄 787
步驟 10：完成代理程式安裝 787
設定檔案傳輸代理程式 788
步驟 1：設定許可和已啟動的任務控制 (STC) 789
步驟 2：建立 Amazon S3 儲存貯體 790
步驟 3：建立用於加密 AWS KMS 的客戶受管金鑰 790
步驟 4：建立大型主機登入資料的 AWS Secrets Manager 秘密 791
步驟 5：建立 IAM 政策 792
步驟 6：建立具有長期存取憑證的 IAM 使用者 793
步驟 7：為要擔任的代理程式建立 IAM 角色 794
步驟 8：客服人員組態 795
建立資料傳輸端點 798
建立資料傳輸端點 798
建立傳輸任務 799
建立傳輸任務 800
檢視傳輸任務 803
教學課程：檔案傳輸入門 803
概觀 803
步驟 1：將代理程式二進位檔 tar 套件從 轉移到 AWS 大型主機邏輯分割區 804
步驟 2：在來源大型主機上設定檔案傳輸代理程式 804
步驟 3：建立資料傳輸端點 804
步驟 4：建立轉移任務 804
步驟 5：檢視傳輸任務進度 804
支援的來源和目標程式碼頁面 804
大型主機資料集類型 804
支援的程式碼頁面 805

AWS Transform 適用於大型主機 807
主要優點 807
大型主機應用程式主控台演練的轉換 807
使用 Precisely 進行資料複寫 808
先決條件 808
訂閱 Amazon Machine Image 808
使用 Precisely 啟動 AWS 大型主機現代化資料複寫 809
建立 IAM 政策 810
建立 IAM 角色 810
將 IAM 角色連接至 Amazon EC2 執行個體 811
使用 mLogica 進行組裝器轉換 812
什麼是使用 mLogica 進行組裝器轉換 812
程式碼轉換編譯器 813
程式碼轉換架構 813
自動化方法 814
安全 814
其他資源 814
了解程式碼轉換帳單 814
程式碼轉換帳單和範圍 814
程式碼轉換概念 816
巨集處理 817
程式碼頁面 (EBCDIC 與 ASCII) 817
CodeBuild 817
了解元件和程序 817
AWS Mainframe Modernization 容器 817
S3 專案儲存貯體 818
日誌檔案位置 818
程序概觀 819
教學課程：將程式碼從 Assembler 轉換為 COBOL 819
先決條件 820
步驟 1：與 共用建置資產 AWS 帳戶 820
步驟 2：建立 Amazon S3 儲存貯體 821
步驟 3：建立 IAM 政策 821
步驟 4：建立 IAM 角色 823
步驟 5：將 IAM 政策連接至 IAM 角色 824
步驟 6：建立 CodeBuild 專案 824

步驟 7：定義專案並上傳原始碼 830
步驟 8：執行分析並了解報告 831
步驟 9：執行程式碼轉換 832
步驟 10：驗證程式碼轉換 835
步驟 11：下載轉換後的程式碼 836
清除資源 837
Charon 整合 838
Charon-SSP 簡介 838
支援的訪客作業系統 839
Charon-SSP 雲端執行個體先決條件 840
執行個體先決條件 841
為 Charon 建立和設定 AWS 雲端執行個體 （新 GUI) 842
一般先決條件 842
使用 AWS Management Console 啟動新的執行個體 843
使用 NTT DATA 進行複寫 848
先決條件 848
訂閱 Amazon Machine Image 848
使用 NTT DATA 執行個體啟動 AWS 大型主機現代化平台 848
NTT 資料入門 849
教學課程：在 NTT DATA 上部署 CardDemo 應用程式 850
部署流程圖 851
先決條件 851
步驟 1：準備環境 852
步驟 2：建立 TPE 區域 853
步驟 3：建立 BPE 節點和子系統 853
步驟 4：編譯和部署 CardDemo 應用程式 862
步驟 5：匯入 BPE 和 TPE 目錄 863
步驟 6：啟動並連接 TPE 與 BPE 864
步驟 7：執行 CardDemo 應用程式 864
故障診斷 870
安全 872
資料保護 873
AWS Mainframe Modernization 收集的資料 873
AWS Mainframe Modernization 服務靜態資料加密 874
AWS Mainframe Modernization 如何在 中使用授予 AWS KMS 876
建立客戶受管金鑰 878

為 AWS 大型主機現代化指定客戶受管金鑰 879
AWS Mainframe Modernization 加密內容 880
監控加密金鑰 881
進一步了解 896
傳輸中加密 896
身分和存取權管理 897
目標對象 897
使用身分驗證 898
使用政策管理存取權 900
AWS Mainframe Modernization 如何與 IAM 搭配使用 902
身分型政策範例 916
故障診斷 919
使用服務連結角色 920
法規遵循驗證 923
恢復能力 924
基礎架構安全 924
AWS PrivateLink 924
考量事項 925
建立介面端點 925
建立端點政策 925
監控 927
使用 CloudWatch 進行監控 927
執行時間環境指標 927
應用程式指標 928
維度 933
使用 CloudTrail 記錄 API 呼叫 933
AWS CloudTrail 中的大型主機現代化資訊 933
了解 AWS Mainframe Modernization 日誌檔案項目 934
在 M2 中進行故障診斷 936
故障診斷錯誤：等待資料集名稱解除鎖定時逾時 936
常見原因 936
Resolution 937
強制鎖定釋放 937
設定 Blusam 自動修復機制 938
Blusam 鎖定管理員 938
故障診斷錯誤：無法存取應用程式 URL 939

常見原因 939
Resolution 939
故障診斷： AWS Blu Insights 不會從主控台開啟 940
常見原因 940
Resolution 940
故障診斷錯誤：環境運作狀態不佳 941
常見原因 941
Resolution 941
針對 Rocket Software 的授權問題進行故障診斷 942
確認 Amazon EC2 執行個體具有 IAM 授權角色 942
使用連線能力分析器 943
執行 license-daemon 943
作業系統修補後，Linux 上的 Enterprise Server 或 Enterprise Build Tools 的授權問題 944
文件歷史紀錄 945
..... cml

什麼是 AWS 大型主機現代化？

AWS Mainframe Modernization 可協助您將大型主機應用程式現代化為 AWS 受管執行期環境。它提供工具和資源來協助您規劃和實作遷移和現代化。您可以分析現有的大型主機應用程式，使用 COBOL 或 開發或更新它們PL/I，並實作自動化管道以持續整合和持續交付(CI/CD)應用程式。您可以根據用戶端的需求，選擇自動重構和轉譯模式。如果您是協助用戶端遷移其大型主機工作負載的顧問，您可以使用 AWS 大型主機現代化工具進行遷移和現代化旅程的所有階段，從初始規劃到遷移後雲端操作。

您可以使用 AWS Mainframe Modernization 來協助您有效建立和管理大型主機應用程式在 上的 AWS 執行期環境，以及管理和監控現代化應用程式。

主題

- AWS Mainframe Modernization 的功能
- 模式
- 如何開始使用 AWS 大型主機現代化
- 相關服務
- 存取 AWS 大型主機現代化
- 您是第一次使用 AWS 大型主機現代化嗎？
- AWS 大型主機現代化定價

> **Note**
>
> 您是否已針對 AWS 大型主機現代化專案與大型主機遷移能力合作夥伴或 AWS 專業服務互動？ 如果沒有，強烈建議您讓專案的專家參與。
>
> - AWS 大型主機現代化能力合作夥伴
> - AWS Professional Services

AWS Mainframe Modernization 的功能和使用案例支援進化的現代化方法，透過提高敏捷性和許多稍後最佳化和創新的機會，提供短期成功。如需詳細資訊，請參閱現代化方法。

AWS Mainframe Modernization 的功能

AWS Mainframe Modernization 功能支援下列使用案例：

- 評估： AWS 大型主機現代化評估功能可協助您評估、調整範圍和規劃遷移和現代化專案。
- 重構：採用 AWS Blu Age 技術，您可以使用重構來轉換舊版應用程式程式設計語言、建立巨集服務或微服務，以及現代化使用者介面 (UIs) 和應用程式軟體堆疊。

  AWS Blu Insights 現在可透過單一登入 AWS Management Console 從 取得。您不再需要管理個別的 AWS Blu Insights 登入資料。您可以直接從 存取 AWS AWS Blu Age Codebase 和 Transformation Center 功能 AWS Management Console。
- Replatform：採用 Micro Focus Enterprise 解決方案，您可以移植應用程式，其中大部分應用程式原始程式碼都會重新編譯，而不會變更。
- 開發人員 IDE： AWS Mainframe Modernization 提供隨需整合的開發環境 (IDE)，因此開發人員可以透過智慧編輯和偵錯、即時程式碼編譯和單元測試，更快撰寫程式碼。
- 受管執行時間： AWS Mainframe Modernization 受管執行時間環境會持續監控叢集，以使用自我修復運算和自動化擴展來保持企業工作負載的執行。
- 持續整合和交付(CI/CD)： AWS 大型主機現代化CI/CD功能可協助應用程式開發團隊更頻繁、更可靠地交付程式碼變更，進而加速遷移速度、提高品質，並有助於縮短time-to-market以推出新的業務功能。
- 與其他 AWS 服務的整合： AWS 大型主機現代化支援 AWS PrivateLink和 AWS CloudFormation，AWS Key Management Service 用於可重複的部署和更高的安全性和合規性。
- 擴大可用性： AWS Mainframe Modernization 現已在美國東部 （俄亥俄）、美國西部 （加利佛尼亞北部）、亞太區域 （孟買）、亞太區域 （首爾）、亞太區域 （新加坡）、亞太區域 （東京）、歐洲 （倫敦） 和歐洲 （巴黎） 提供。

如需詳細資訊，請參閱AWS 大型主機現代化功能。

模式

自動化重構模式採用 AWS Blu Age，專注於透過將完整的舊版應用程式堆疊及其資料層轉換為現代 Java 型應用程式，同時保持功能等效性來加速現代化。在此自動化轉換期間，它會建立多層應用程式，其中包含以角度為基礎的前端、啟用 API 的 Java 後端，以及存取現代資料存放區的資料層。重構程序提供與舊版堆疊相同的功能，以增加專案自動化，進而加快實現業務利益的速度、品質並降低成本。如需詳細資訊，請參閱AWS 大型主機現代化自動化重構。

採用 Rocket Software 前身為 Micro Focus) Enterprise 套件的 Replatforming 模式專注於保留應用程式語言、程式碼和成品，以將對應用程式資產和團隊的影響降至最低。它可協助客戶維持應用程式知識和技能。雖然應用程式變更有限，但此模式也有助於基礎設施和程序的現代化。基礎設施會變更為現代

雲端型受管服務，而程序則會變更為遵循應用程式開發和 IT 操作的最佳實務。如需詳細資訊，請參閱 AWS Mainframe Modernization Replatform。

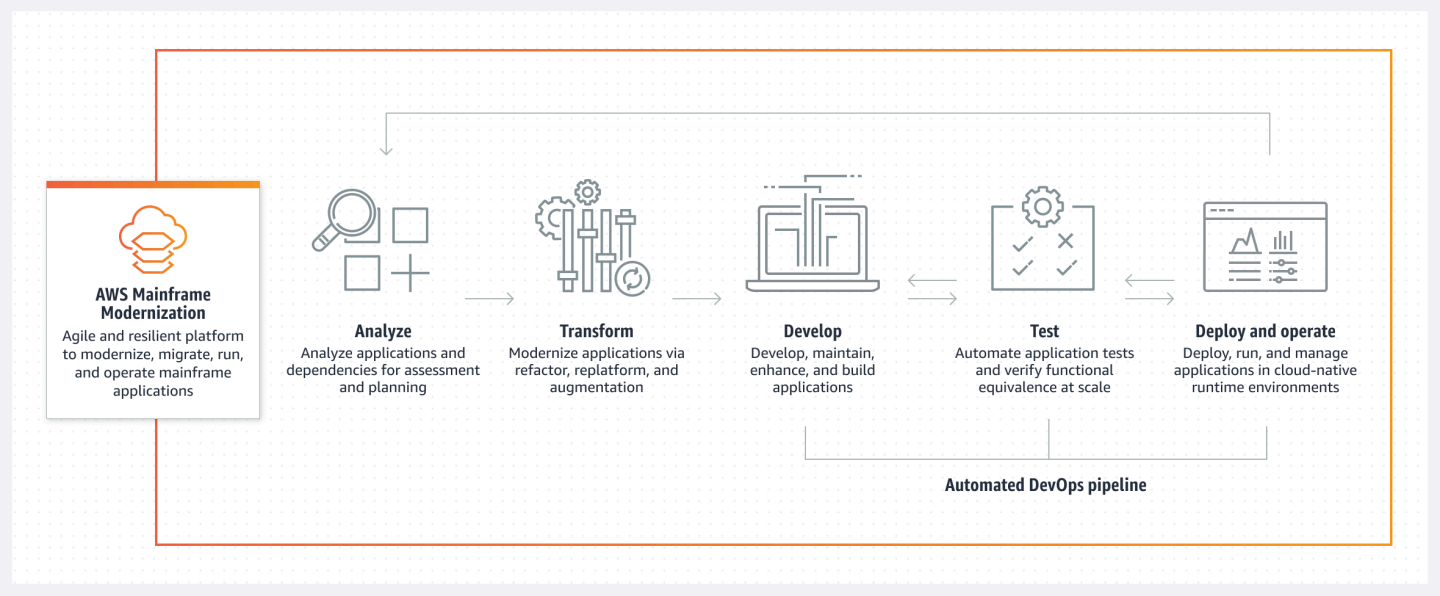
如何開始使用 AWS 大型主機現代化

試試看！我們提供教學課程和範例應用程式，協助您了解 AWS Mainframe Modernization 提供的內容。選擇 教學課程：為 AWS 藍年期設定受管執行時間或 ，教學課程：設定 Rocket Software 的受管執行期（先前稱為 Micro Focus)以取得完整的step-by-step教學課程。

如果您有興趣自動重構，請參閱 AWS BluInsights 的 Blu Age 工具。您也可以設定 AppStream 2.0 來存取 AWS Blu Age Developer IDE，或 Rocket Enterprise Analyzer（先前稱為 Micro Focus Enterprise Analyzer) 和 Rocket Enterprise Developer（先前稱為 Micro Focus Enterprise Developer) 工具。

教學課程和範例應用程式只會讓您了解 AWS Mainframe Modernization 提供的內容。當您準備好啟動現代化專案時，請參閱 現代化方法 以取得現代化專案階段和任務的詳細資訊。

下圖顯示 AWS Mainframe Modernization 服務的工作流程，用於分析、轉換、開發、測試和部署和操作大型主機應用程式。



相關服務

除了用於自動重構的 Blu Insights 之外，您還可以搭配 AWS Mainframe Modernization 使用下列 AWS 服務。

- 用於託管遷移資料庫的 Amazon RDS
- Amazon S3 用於儲存應用程式二進位檔和定義檔案
- 用於儲存應用程式資料的 Amazon FSx 或 Amazon EFS
- Amazon AppStream 可存取 Rocket Enterprise Analyzer 和 Rocket Enterprise Developer 工具
- AWS CloudFormation 適用於自動化 DevOps 管道，您可以用來CI/CD設定遷移應用程式
- AWS Migration Hub
- AWS DMS 用於遷移資料庫

存取 AWS 大型主機現代化

目前，您可以透過 主控台存取 AWS Mainframe Modernization，網址為 https：//https://console.aws.amazon.com/m2/。如需可使用 AWS 大型主機現代化的區域清單，請參閱《》中的AWS 大型主機現代化端點和配額Amazon Web Services 一般參考。

您是第一次使用 AWS 大型主機現代化嗎？

如果您是首次使用 AWS Mainframe Modernization，建議您先閱讀以下章節：

- AWS 大型主機現代化入門
- 設定 AWS 大型主機現代化

AWS 大型主機現代化定價

AWS Mainframe Modernization 會針對支援受管執行期環境之執行個體的使用收費。此外， AWS 大型主機現代化提供一些工具，無需額外費用。您必須負責支付 AWS 與 AWS 大型主機現代化相關的其他服務所產生的費用。 AWS 將在任何定價變更生效前 30 天通知 ，以使用 AWS 大型主機現代化。如需詳細資訊，請參閱 Mainframe Modernization with AWS。

使用 AWS Blu Insights，您需要支付轉換中心的用量。如需詳細資訊，請參閱AWS 大型主機現代化定價。

設定 AWS 大型主機現代化

在開始使用 AWS Mainframe Modernization 之前，您或您的管理員需要註冊 AWS 帳戶，使用管理設定建立使用者，並保護您的 IAM 使用者。

主題

- 註冊 AWS 帳戶
- 建立具有管理存取權的使用者

註冊 AWS 帳戶

如果您沒有 AWS 帳戶，請完成下列步驟來建立一個。

註冊 AWS 帳戶

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

   註冊程序的一部分包括接聽電話或文字訊息，並在電話鍵盤上輸入驗證碼。

   當您註冊 時 AWS 帳戶，AWS 帳戶根使用者會建立 。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。作為安全最佳實務，請將管理存取權指派給使用者，並且僅使用根使用者來執行需要根使用者存取權的任務。

AWS 會在註冊程序完成後傳送確認電子郵件給您。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。

建立具有管理存取權的使用者

註冊 後 AWS 帳戶，請保護 AWS 帳戶根使用者、啟用 AWS IAM Identity Center和建立管理使用者，以免將根使用者用於日常任務。

保護您的 AWS 帳戶根使用者

1. 選擇根使用者並輸入 AWS 帳戶 您的電子郵件地址，以帳戶擁有者AWS Management Console身分登入 。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)。

   如需說明，請參閱《IAM 使用者指南》中的為您的 AWS 帳戶 根使用者 （主控台） 啟用虛擬 MFA 裝置。

建立具有管理存取權的使用者

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 Identity Center 中，將管理存取權授予使用者。

   如需使用 IAM Identity Center 目錄 做為身分來源的教學課程，請參閱AWS IAM Identity Center 《 使用者指南》中的使用預設值設定使用者存取 IAM Identity Center 目錄。

以具有管理存取權的使用者身分登入

- 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 Identity Center 使用者登入的說明，請參閱AWS 登入 《 使用者指南》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 Identity Center 中，建立一個許可集來遵循套用最低權限的最佳實務。

   如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的建立許可集。

2. 將使用者指派至群組，然後對該群組指派單一登入存取權。

   如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的新增群組。

AWS 大型主機現代化概念

AWS Mainframe Modernization 提供工具和資源，協助您遷移、現代化和執行大型主機工作負載 AWS。您可以使用此頁面來了解 AWS 大型主機現代化中的各種概念，包括應用程式、現代化、環境、重建、重構和執行期引擎。

主題

- 應用程式
- 應用程式定義
- 批次任務
- 組態
- 資料集
- 環境
- 大型主機現代化
- 遷移旅程
- 掛載點
- 自動化重構
- 平台轉換
- 資源
- 執行期引擎

應用程式

Mainframe Modernization 中的執行 AWS 中大型主機工作負載。一組批次任務、互動式交易 (CICS 或 IMS) 或其他元件組成 應用程式。您可以定義範圍。您必須定義並指定工作負載所需的任何元件或資源，例如 CICS 交易或批次任務。

應用程式定義

在 AWS Mainframe Modernization 中執行的應用程式（大型主機工作負載）所需的元件和資源的定義或規格。將定義與應用程式本身分開非常重要，因為可以針對由不同執行時間環境表示的多個階段（生產前、生產前）重複使用相同的定義。

批次任務

已設定為執行而不需要使用者互動的排程程式。在 AWS 大型主機現代化中，您需要將批次任務 JCL 檔案和批次任務二進位檔同時存放在 Amazon S3 儲存貯體中，並在應用程式定義檔案中提供兩者的位置。當您執行批次任務時， AWS 大型主機現代化會報告下列狀態值：

正在提交

批次任務正在提交中。

保留

批次任務已保留。

分派

批次任務正在分派中。

執行中

批次任務目前正在執行中。

取消

正在取消批次任務。

已取消

批次任務已取消。

Succeeded

批次任務已成功執行。

失敗

批次任務失敗。

使用警告成功

批次工作已成功執行，並回報次要錯誤。在 GetBatchJobExecution 回應中傳回的任務條件碼指出錯誤的原因。

組態

環境或應用程式的特性。環境組態包含引擎類型、引擎版本、可用性模式、選用的檔案系統組態等。

應用程式組態可以是靜態或動態。靜態組態只會在您透過部署新版本更新應用程式時變更。動態組態通常是操作活動，例如開啟或關閉追蹤，會在您更新時立即變更。

資料集

包含供應用程式使用之資料的檔案。

環境

為託管一或多個應用程式而建立的 AWS 運算資源、執行期引擎和組態詳細資訊的命名組合。

大型主機現代化

將應用程式從舊版大型主機環境遷移至 的程序 AWS。

遷移旅程

遷移和現代化舊版應用程式的end-to-end程序，通常由下列階段組成：評估、調動、遷移和現代化，以及操作和最佳化。

掛載點

檔案系統中的目錄，可讓您存取存放在該系統中的檔案。

自動化重構

現代化舊版應用程式成品的程序，以便在現代雲端環境中執行。它可以包含程式碼和資料轉換。如需詳細資訊，請參閱AWS 大型主機現代化自動化重構。

平台轉換

將應用程式和應用程式成品從一個運算平台移至不同運算平台的程序。如需詳細資訊，請參閱 AWS Mainframe Modernization Replatform。

資源

電腦系統內的實體或虛擬元件。

執行期引擎

促進應用程式執行的軟體。

現代化方法

遷移很複雜，變數很多。 AWS Mainframe Modernization 提供一種創新方法，透過提高敏捷性，以及許多稍後最佳化和創新的機會，來提供一些短期成功。此外， AWS Mainframe Modernization 有助於簡化旅程，但仍遵守客戶公司和業務的詳細資訊。 AWS Mainframe Modernization 支援的兩個主要方法是自動重構或重建。要選擇哪些項目取決於用戶端的情況。

自動化重構使用 AWS Blu Age 工具，自動將程式碼、資料和相依性轉換為現代語言、資料存放區和架構，同時保證具有相同業務職能的功能等效性。

Replatforming 使用 Rocket Software （ 先前稱為 Micro Focus) 工具，將大型主機工作負載轉換為靈活的服務 AWS。

您可以分階段思考現代化之旅。第一階段包含三個階段：評估、調動和遷移和現代化。下一個階段包含操作和最佳化階段，您可以在其中找出更多創新的機會。

主題

- 評估階段
- 調動階段
- 遷移和現代化階段
- 操作和最佳化階段

評估階段

在最高層級，評估階段會查看您是否已準備好遷移。您定義商業案例，然後使用 提供的研討會和沉浸日 （ 示範和實驗室） 來教育您的團隊 AWS。研討會和臨場日處理不同的主題。這些任務是在 AWS Mainframe Modernization 之外執行。

調動階段

在行動化階段，您會從啟動開始專案，然後執行探索程序，從大型主機應用程式擷取資料並將其擷取至遷移工具。您可以識別要遷移的應用程式，並選取要試行的一些應用程式。您可以完善您的商業案例、撰寫遷移計畫，並決定如何處理安全性與合規性、帳戶控管，以及營運模型。您與團隊中適當的人員一起設定雲端卓越中心。您執行試行方案並記錄所學到的內容。您可以精簡遷移計畫和商業案例。其中許多任務是在 AWS 大型主機現代化之外執行。

遷移和現代化階段

遷移和現代化階段適用於每個應用程式，包含數個任務，包括指派人員、執行深入探索、找出正確的應用程式架構 AWS、設定應用程式執行時間環境、修改或重構程式碼、與其他系統整合，當然還有測試。在階段結束時，您會將經過修改或重構的應用程式部署到生產環境，並切換到 AWS 上的新系統。大多數或所有這些任務都是在 AWS Mainframe Modernization、其他服務 AWS 或 AWS Mainframe Modernization 提供存取權的工具中執行。

如果您想要使用自動重構，請參閱 Blu Insights。 AWS Blu Insights 現在可透過單一登入 AWS Management Console 從 取得。您不再需要管理單獨的 AWS Blu Insights 登入資料。您可以從 直接存取 AWS AWS Blu Age Codebase 和 Transformation Center 功能 AWS Management Console。

若要將資料從大型主機遷移至 AWS，我們建議使用 AWS SCT 和 AWS Database Migration Service。如需詳細資訊，請參閱《 Schema Conversion Tool使用者指南》中的什麼是 AWS AWS Schema Conversion Tool？和《 AWS Database Migration Service 使用者指南》中的什麼是 AWS Database Migration Service？。

操作和最佳化階段

在操作和最佳化階段，您會專注於監控已部署的應用程式、管理資源，以及確保安全性和合規性是最新的。您也可以評估最佳化遷移工作負載的機會。

AWS 大型主機現代化入門

您可以遵循向您介紹服務和每個執行期引擎的教學課程，開始使用 AWS Mainframe Modernization。

主題

- 教學課程：為 AWS 藍年期設定受管執行時間
- 教學課程：設定 Rocket Software 的受管執行期（先前稱為 Micro Focus)

若要繼續學習，請參閱下列教學課程。

- 教學課程：為 BankDemo 範例應用程式設定 Rocket Software（先前稱為 Micro Focus) 組建
- 教學課程：設定CI/CD管道以與 Rocket Enterprise Developer（先前稱為 Micro Focus Enterprise Developer) 搭配使用

教學課程：為 AWS 藍年期設定受管執行時間

您可以使用本教學課程中指定的示範應用程式，將 AWS Blu Age 現代化應用程式部署至 AWS Mainframe Modernization 執行期環境。

主題

- 先決條件
- 步驟 1：上傳示範應用程式
- 步驟 2：建立應用程式定義
- 步驟 3：建立執行時間環境
- 步驟 4：建立應用程式
- 步驟 5：部署應用程式
- 步驟 6：啟動應用程式
- 步驟 7：存取應用程式
- 步驟 8：測試應用程式
- 清除資源

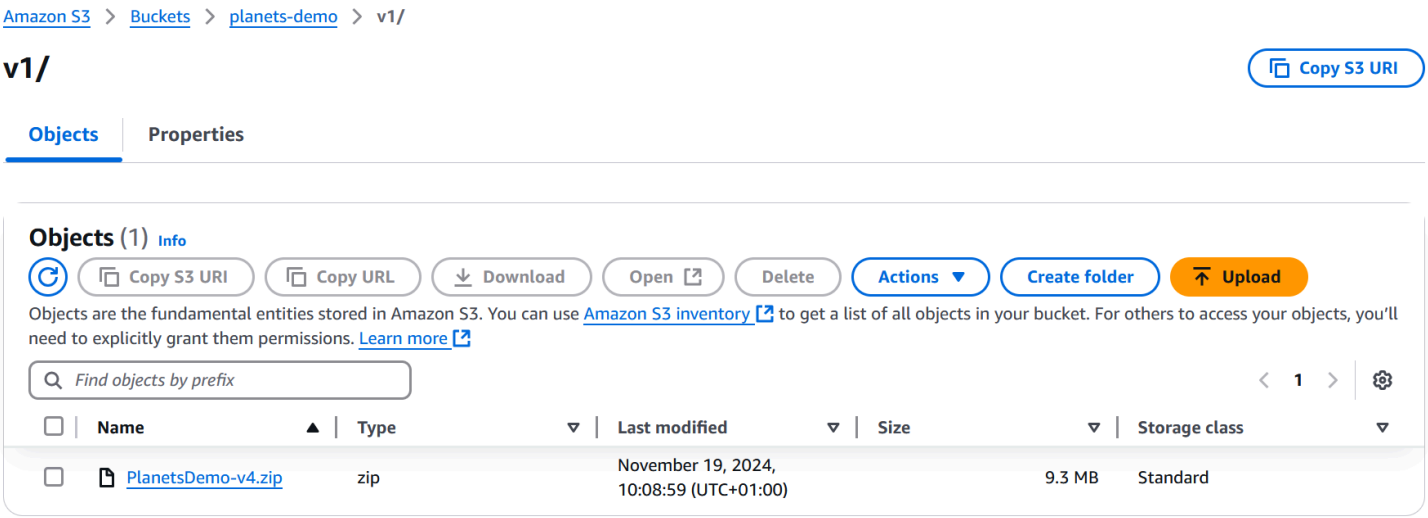
先決條件

若要完成本教學課程，請下載示範應用程式封存 PlanetsDemo-v4.zip。

執行中的示範應用程式需要現代瀏覽器才能存取。無論您是從桌面還是從 Amazon Elastic Compute Cloud 執行個體執行此瀏覽器，例如在 VPC 中， 都會決定您的安全設定。

步驟 1：上傳示範應用程式

將示範應用程式上傳至 Amazon S3 儲存貯體。請確定此儲存貯體與您部署應用程式 AWS 區域 的位置相同。下列範例顯示名為 planets-demo 的儲存貯體，其中包含名為 v1 的金鑰字首或資料夾，以及名為 的封存`planetsdemo-v4.zip`。



> **Note**
>
> 儲存貯體中的 資料夾是必要的。

步驟 2：建立應用程式定義

若要將應用程式部署至受管執行期，您需要 AWS Mainframe Modernization 應用程式定義。此定義是描述應用程式位置和設定的 JSON 檔案。下列範例是示範應用程式的應用程式定義：

```
{
    "template-version": "2.0",
    "source-locations": [{
```

```
        "source-id": "s3-source",
        "source-type": "s3",
        "properties": {
            "s3-bucket": "planets-demo",
            "s3-key-prefix": "v1"
        }
    }],
    "definition": {
        "listeners": [{
            "port": 8196,
            "type": "http"
        }],
        "ba-application": {
            "app-location": "${s3-source}/PlanetsDemo-v4.zip"
        }
    }
}
```

將s3-bucket項目變更為範例應用程式 zip 檔案的名稱 （例如 planets-demo)，並將app-location項目變更為您存放範例應用程式 zip 檔案的 S3 路徑 （例如 ${s3-source}/PlanetsDemo-v4.zip)。

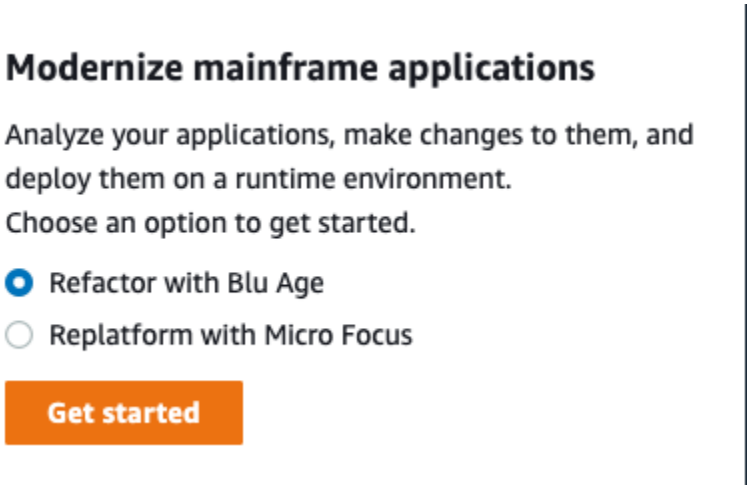
> **Note**
>
> 請務必在本機上建立應用程式定義檔案做為文字檔案。

如需應用程式定義的詳細資訊，請參閱 AWS Blu Age 應用程式定義範例。

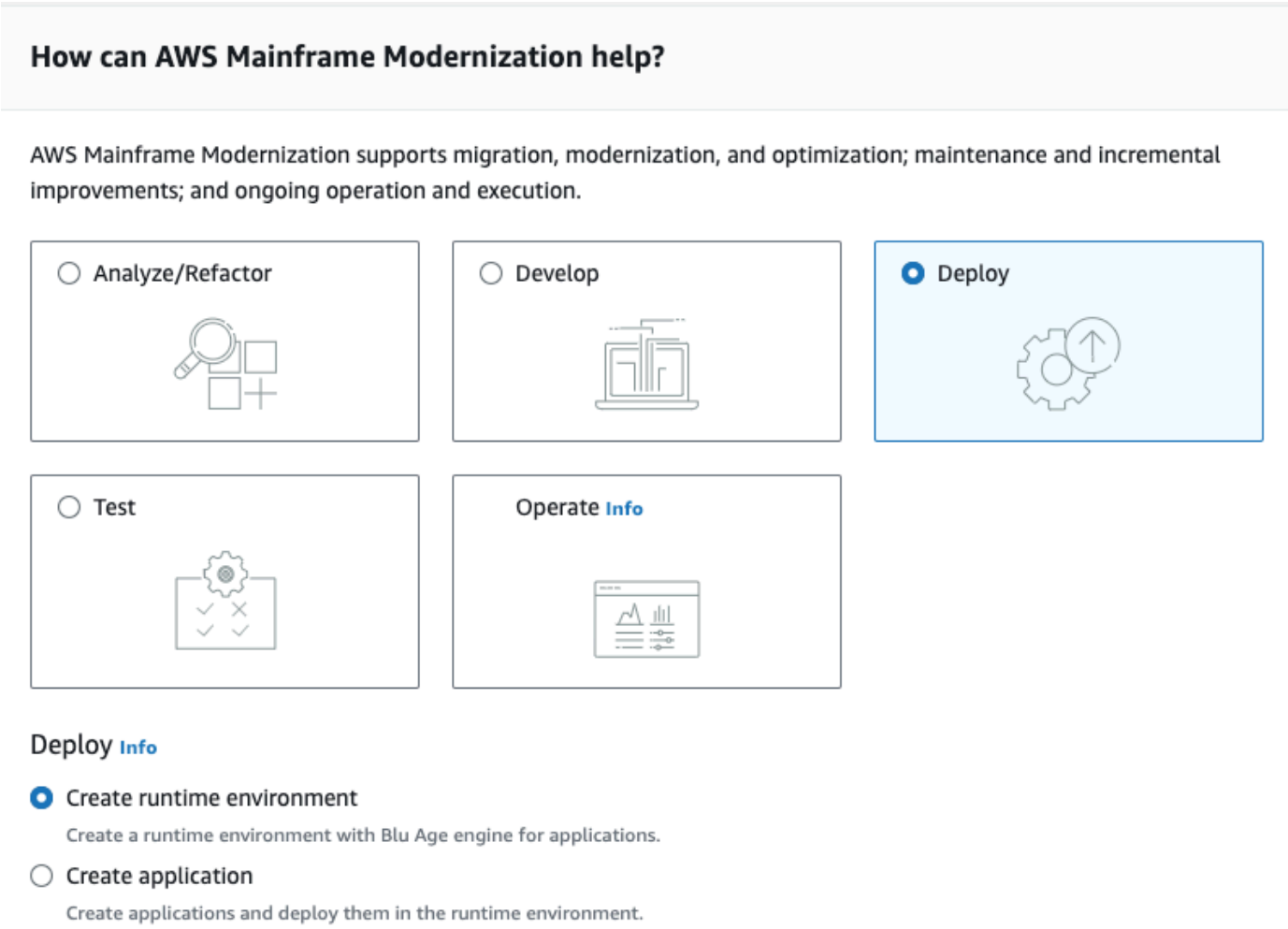
步驟 3：建立執行時間環境

若要建立 AWS Mainframe Modernization 執行時間環境，請執行下列步驟：

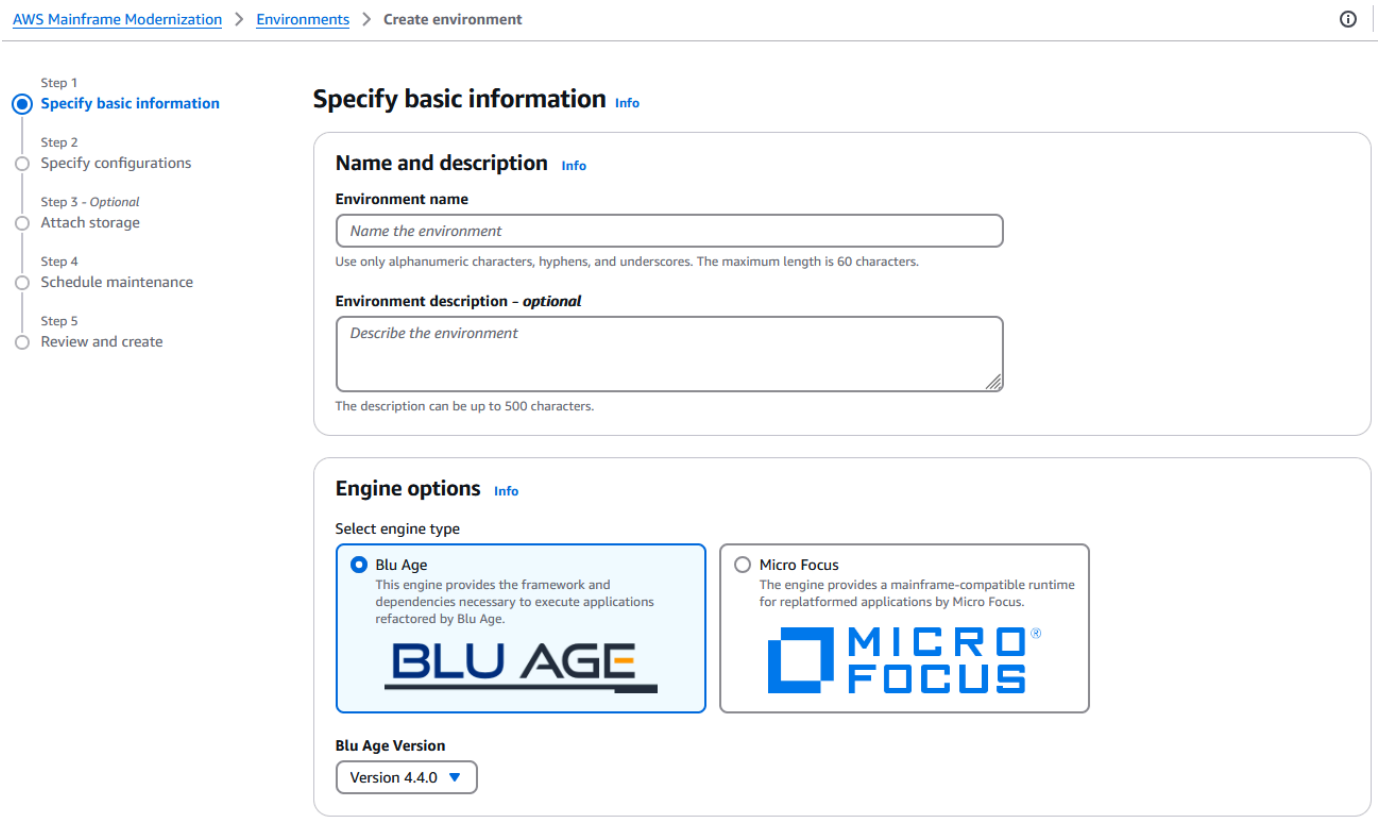
1. 開啟 AWS Mainframe Modernization 主控台。
2. 在 AWS 區域 選擇器中，選擇您要建立環境的區域。這 AWS 區域 必須符合您在 中建立 S3 儲存貯體的區域步驟 1：上傳示範應用程式。
3. 在現代化大型主機應用程式下，選擇重構搭配模糊年齡，然後選擇開始使用。



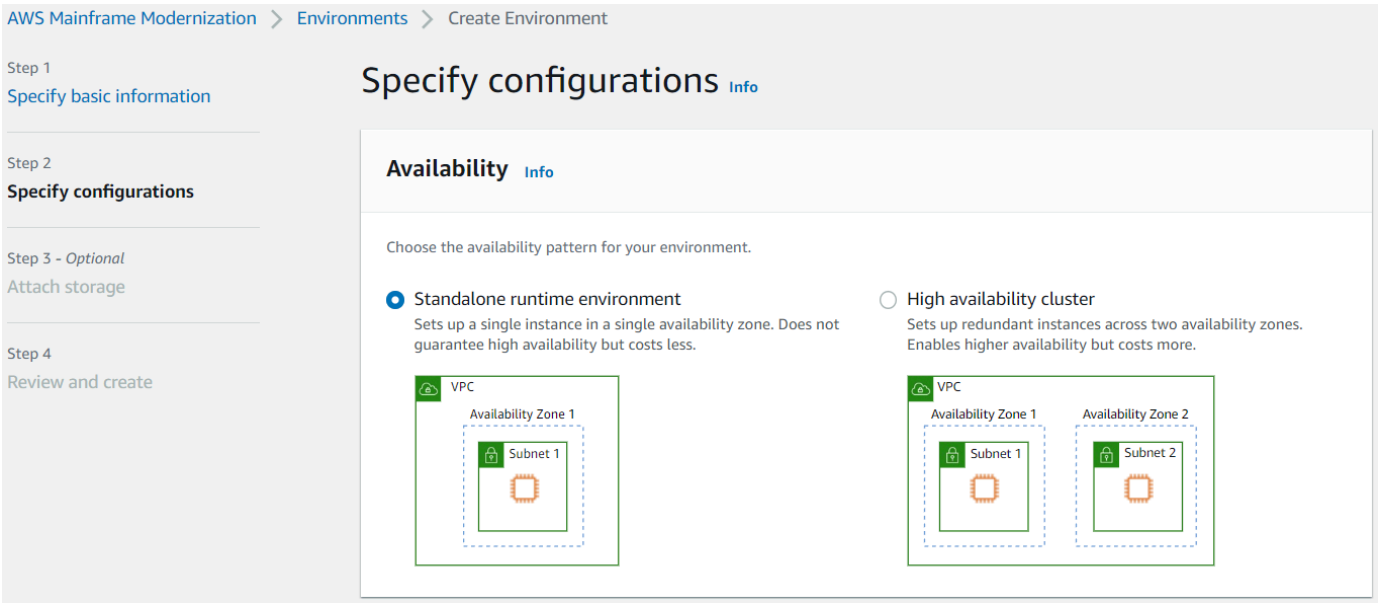
4. 在 AWS Mainframe Modernization 說明下，選擇部署和建立執行時間環境。



5. 在左側導覽中，選擇環境，然後選擇建立環境。在指定基本資訊頁面上，輸入您環境的名稱和描述，然後確定已選取 AWS Blu Age 引擎。或者，您可以將標籤新增至建立的資源。然後選擇下一步。



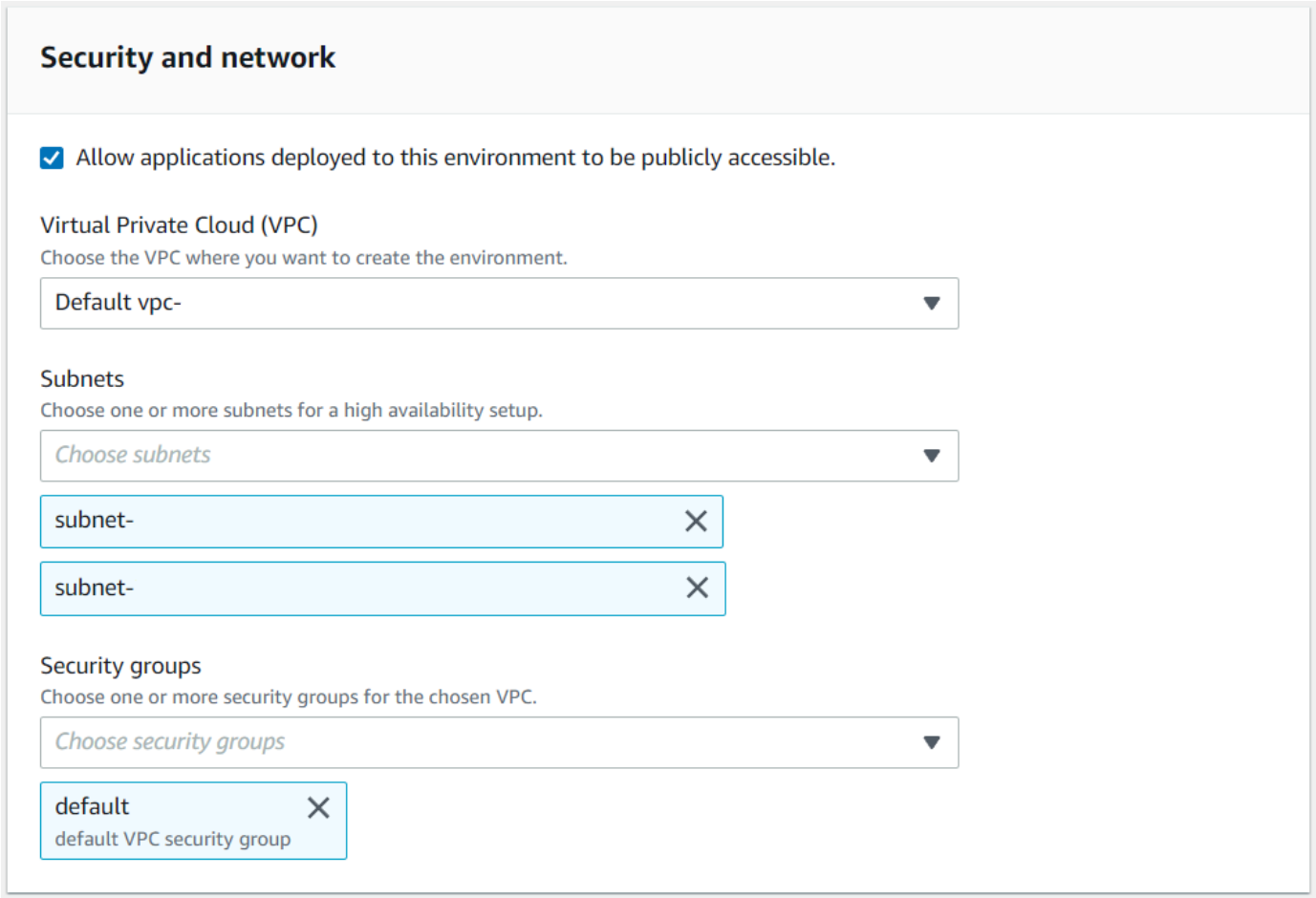
6. 在指定組態頁面上，選擇獨立執行期環境。



7. 在安全與網路下，進行下列變更：

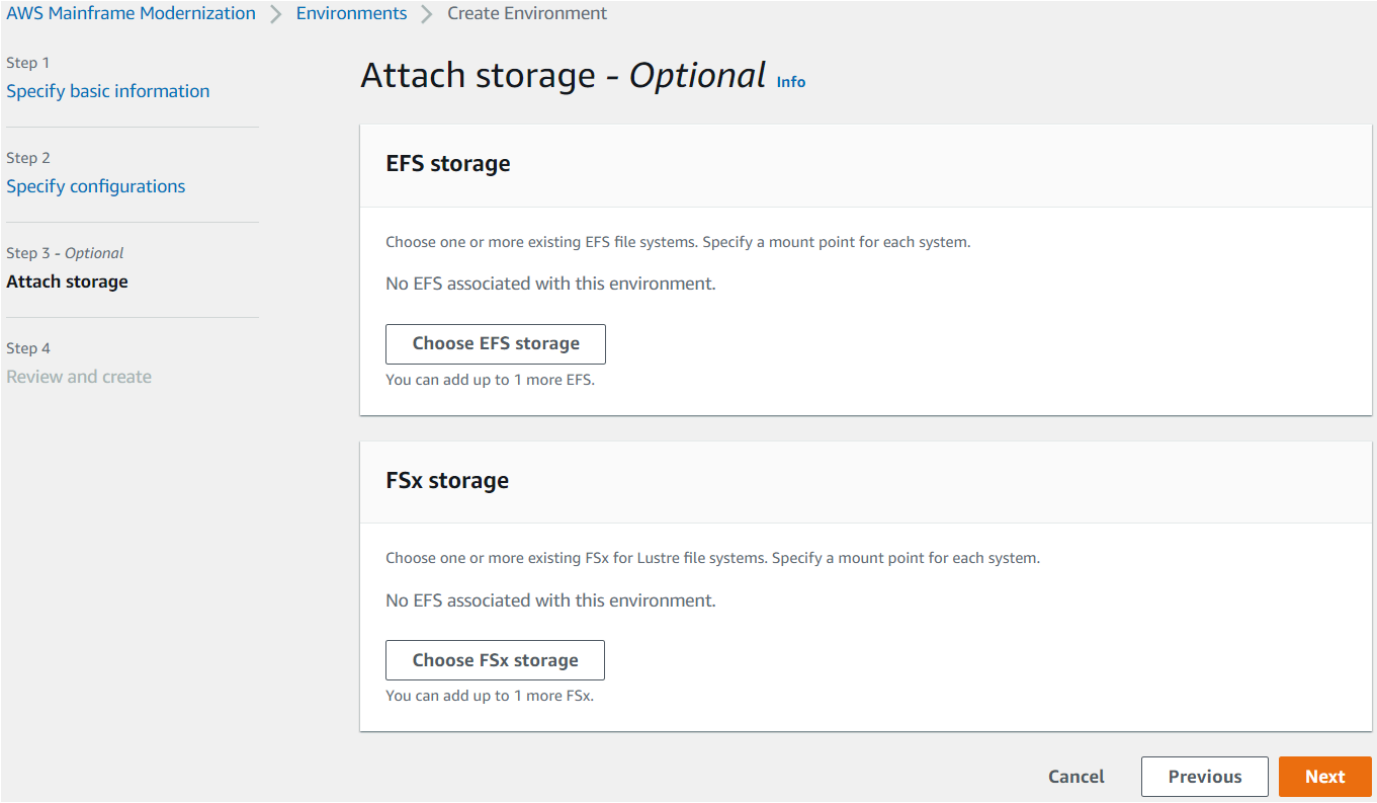
   - 選擇允許部署到此環境的應用程式可公開存取。此選項會將公有 IP 地址指派給應用程式，以便您可以從桌面存取。
   - 選擇 VPC。您可以使用預設值。

- 選擇兩個子網路。請確定子網路允許指派公有 IP 地址。
- 選擇安全群組。您可以使用預設值。請確定您選擇的安全群組允許從瀏覽器 IP 地址存取您在應用程式定義的 `listener` 屬性中指定的連接埠。如需詳細資訊，請參閱步驟 2：建立應用程式定義。



如果您想要從您選擇的 VPC 外部存取應用程式，請確定該 VPC 的傳入規則已正確設定。如需詳細資訊，請參閱故障診斷錯誤：無法存取應用程式 URL。

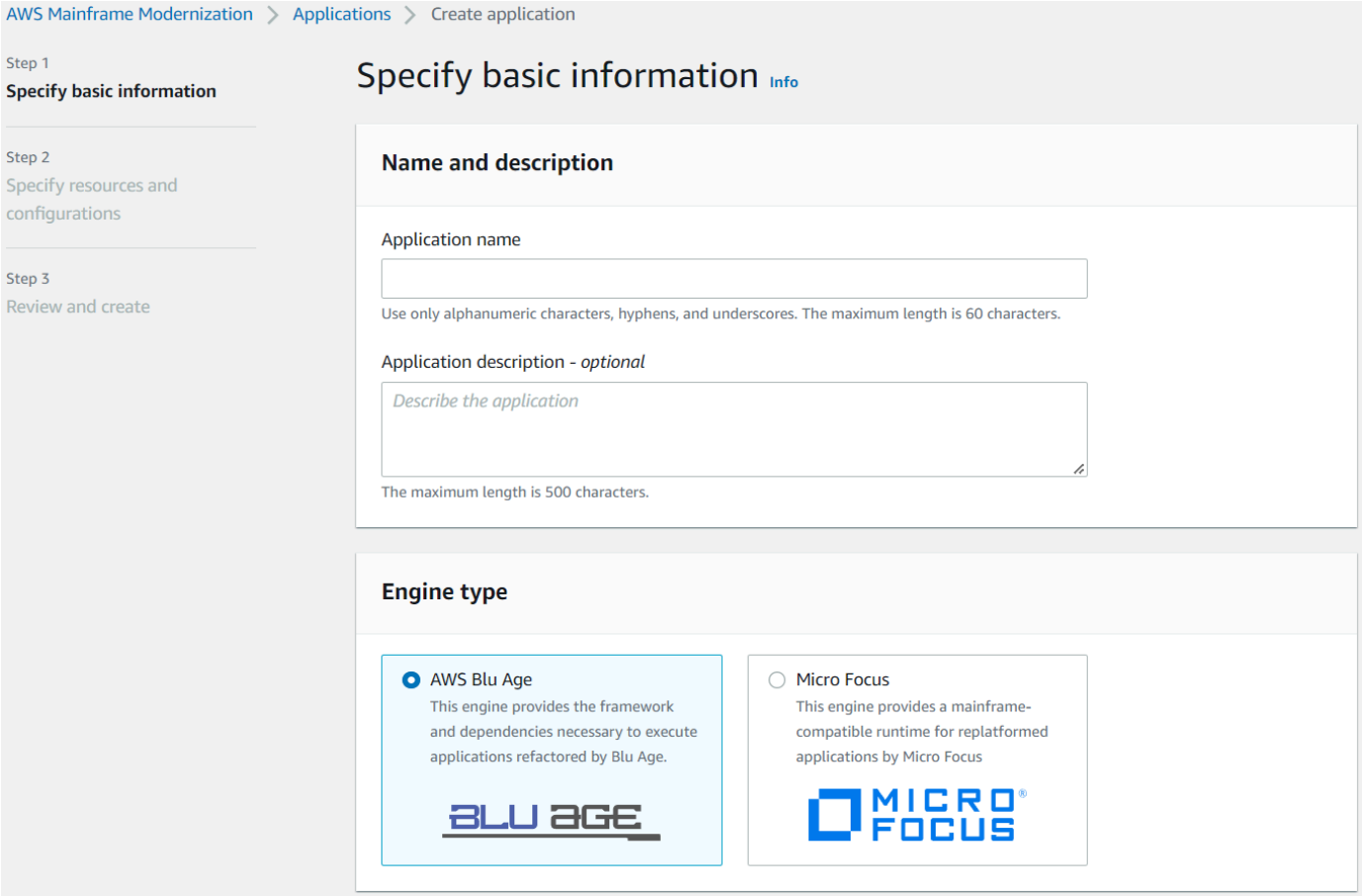
8. 選擇下一步。
9. 在連接儲存 - 選用中，保留預設選擇，然後選擇下一步。



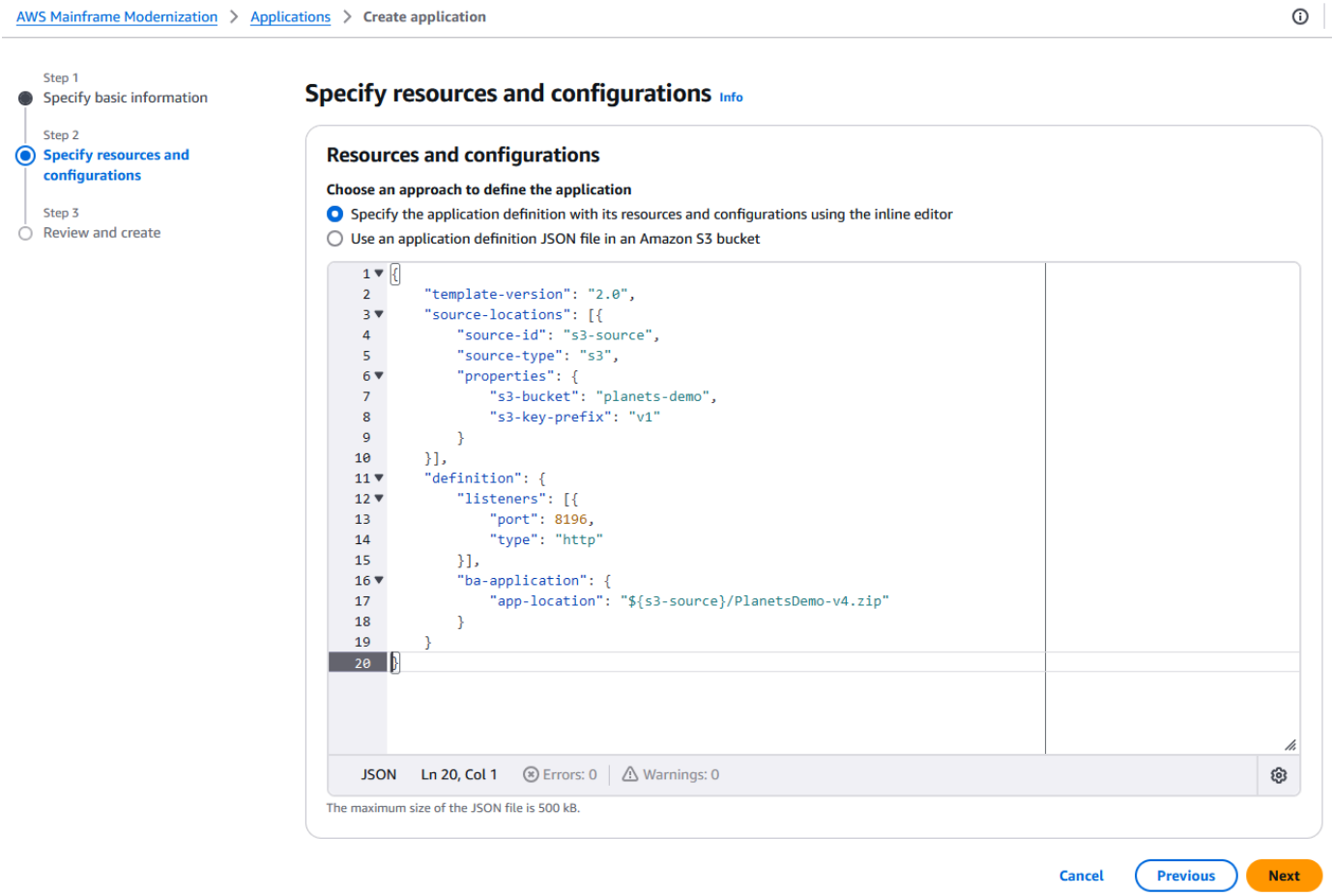
10. 在排程維護中，選擇無偏好設定，然後選擇下一步。
11. 在檢閱和建立中，檢閱資訊，然後選擇建立環境。

步驟 4：建立應用程式

1. 導覽至 中的 AWS Mainframe Modernization AWS Management Console。
2. 在導覽窗格中，選擇 Applications (應用程式)，然後選擇 Create application (建立應用程式)。在指定基本資訊頁面上，輸入應用程式的名稱和描述，並確認已選取 AWS Blu Age 引擎。然後選擇下一步。



3. 在指定資源和組態頁面上，複製並貼上您在 中建立的更新應用程式定義 JSONthe section called "步驟 2：建立應用程式定義"。



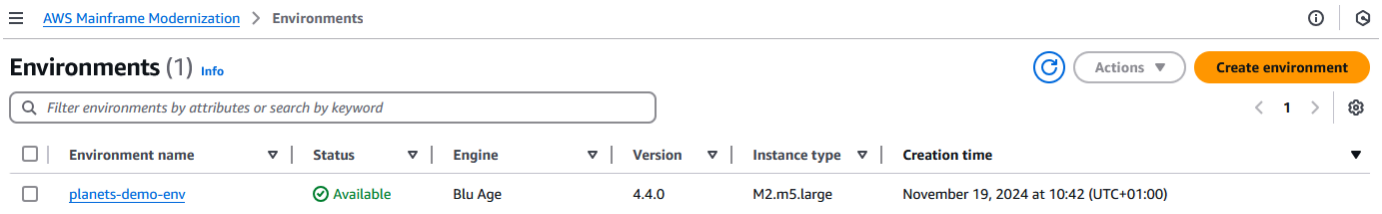
4. 在檢閱和建立中，檢閱您的選擇，然後選擇建立應用程式。

> **Note**
>
> 如果您的應用程式建立失敗，請檢查您輸入的 S3 路徑，因為它區分大小寫。

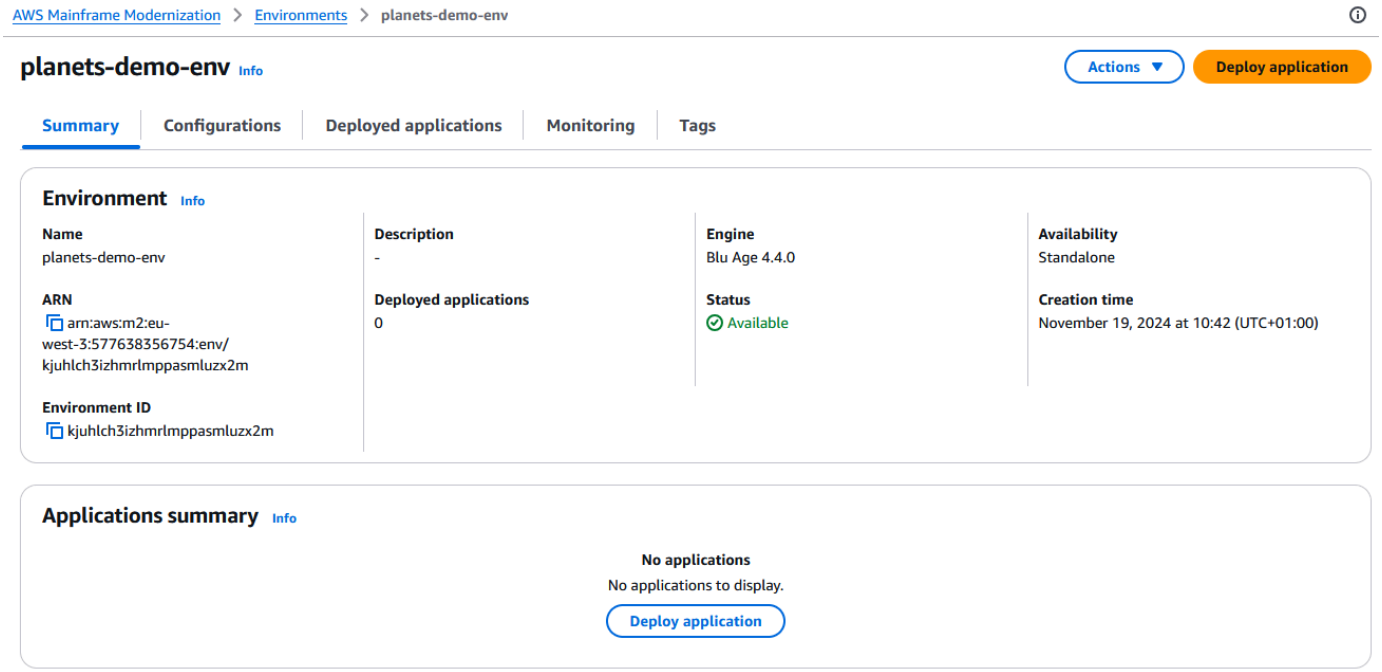
步驟 5：部署應用程式

成功建立 AWS Mainframe Modernization 執行期環境和應用程式，且兩者都處於可用狀態後，您可以將應用程式部署到執行期環境。若要執行此動作，請執行下列步驟。

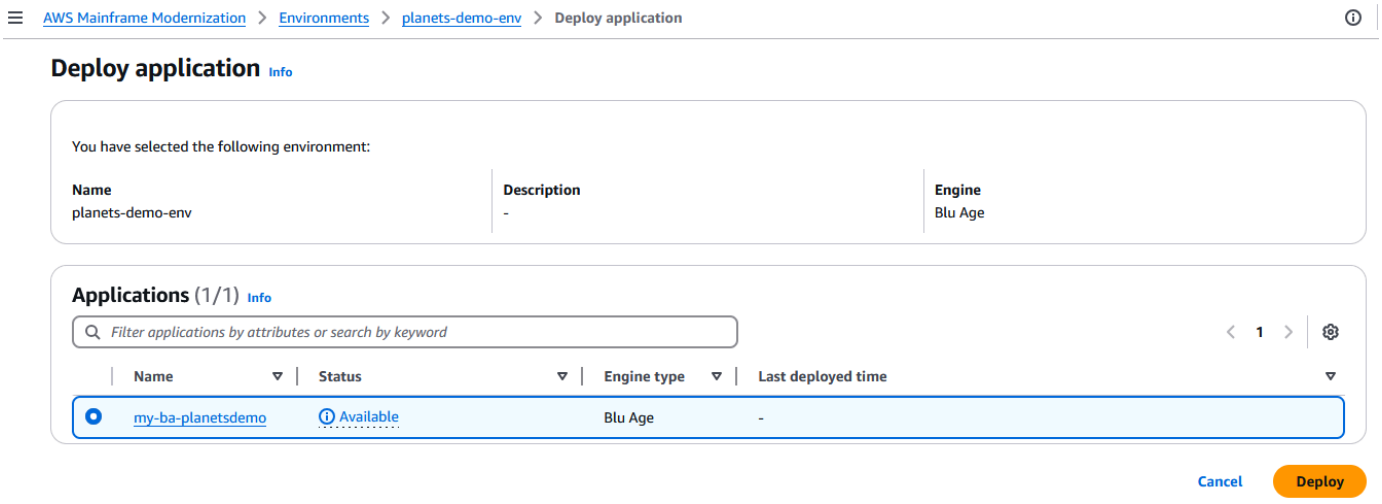
1. 在 AWS 管理主控台中導覽至 AWS Mainframe Modernization。在導覽窗格中，選擇 Environments (環境)。環境清單頁面隨即顯示。



2. 選擇先前建立的執行時間環境。環境詳細資訊頁面隨即顯示。
3. 選擇部署應用程式。



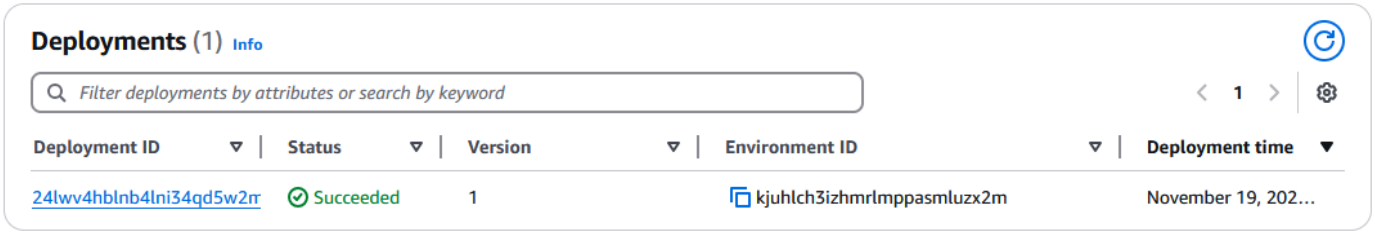
4. 選擇先前建立的應用程式，然後選擇您要部署應用程式的版本。然後選擇 Deploy (部署)。



5. 等待應用程式完成部署。您將看到橫幅，其中包含應用程式已成功部署的訊息。

步驟 6：啟動應用程式

1. 在 中導覽至 AWS Mainframe Modernization， AWS Management Console 然後選擇應用程式。
2. 選擇您的應用程式，然後前往 部署。應用程式的狀態應該成功。



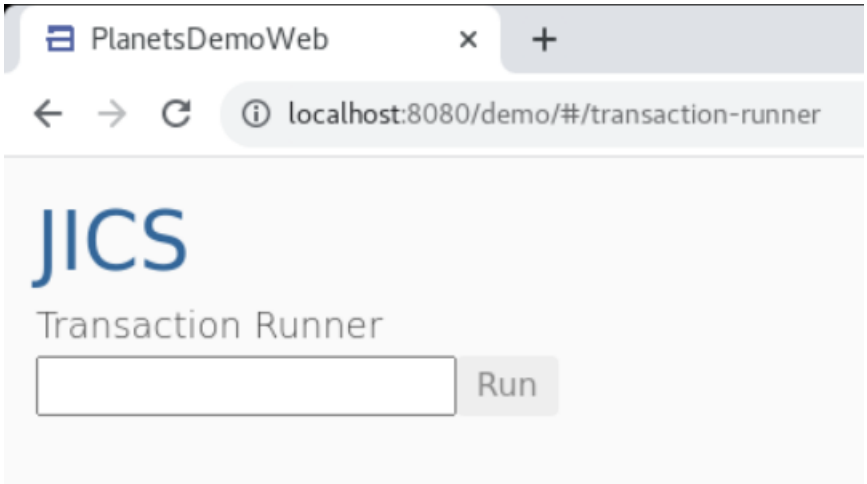
3. 選擇動作，然後選擇啟動應用程式。

步驟 7：存取應用程式

1. 等到應用程式處於執行中狀態。您將看到橫幅，其中包含應用程式已成功啟動的訊息。
2. 複製應用程式 DNS 主機名稱。您可以在應用程式的應用程式資訊區段中找到此主機名稱。
3. 在瀏覽器中，導覽至 `http://{hostname}:{portname}/PlanetsDemo-web-1.0.0/`，其中：

   - `hostname` 是先前複製的 DNS 主機名稱。
   - `portname` 是您在 中建立的應用程式定義中定義的 Tomcat 連接埠步驟 2：建立應用程式定義。

   JICS 畫面隨即出現。



如果您無法存取應用程式，請參閱 故障診斷錯誤：無法存取應用程式 URL。

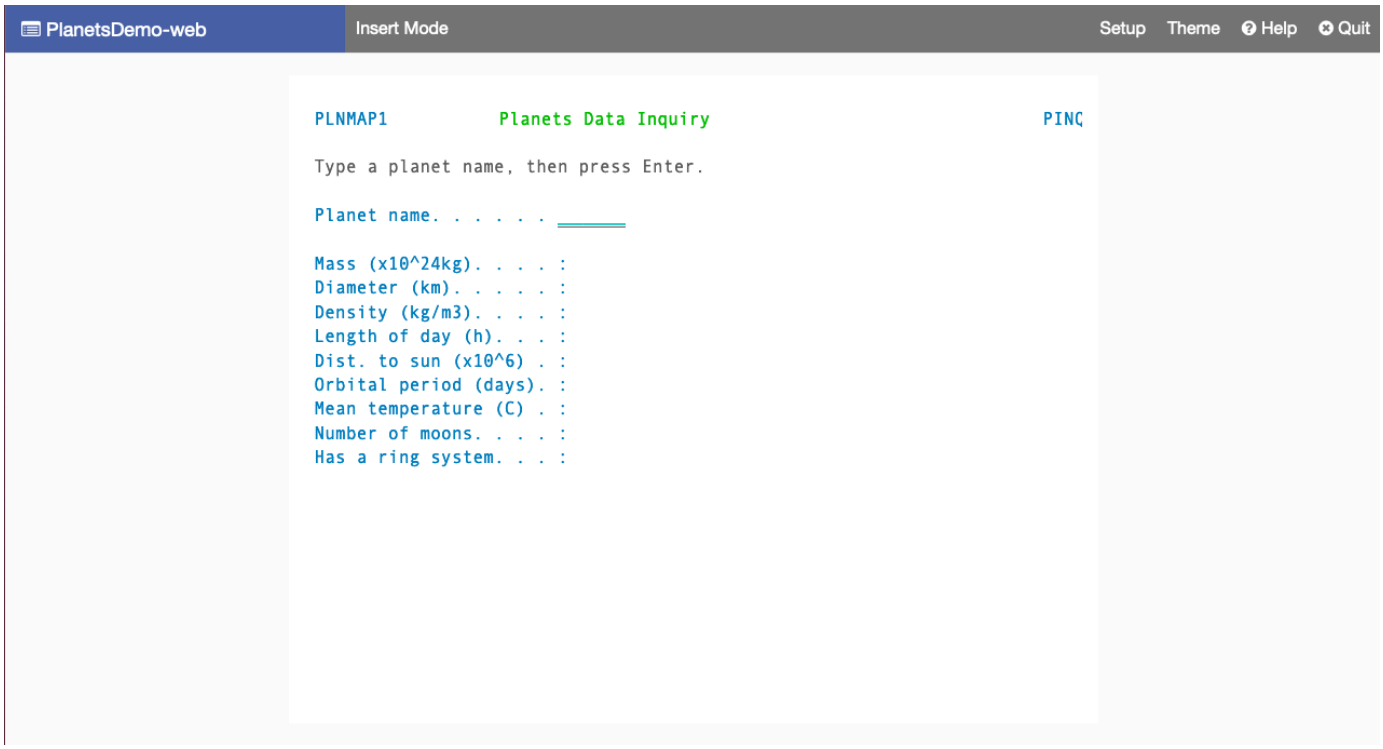
> **Note**
>
> 如果應用程式無法存取，且安全群組上的傳入規則已在連接埠 8196 上選取「我的 IP」，請指定規則以允許連接埠 8196 上來自 LB i/p 的流量。

步驟 8：測試應用程式

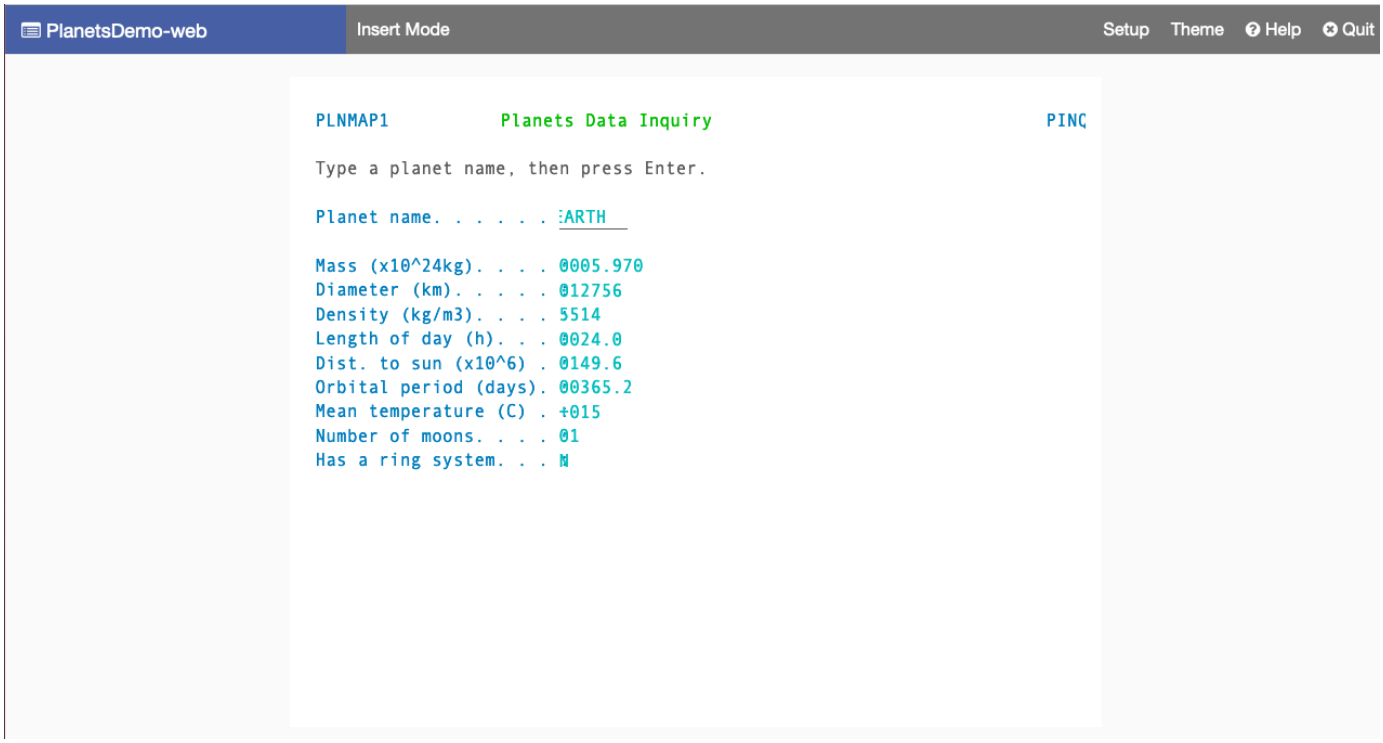
在此步驟中，您會在已遷移的應用程式中執行交易。

1. 在 JICS 畫面上，輸入 PINQ輸入欄位，然後選擇執行 （或按下 Enter) 以啟動應用程式交易。

   示範應用程式畫面應會出現。



2. 在對應的欄位中輸入行星名稱，然後按 Enter 鍵。



您應該會看到有關行星的詳細資訊。

清除資源

如果您不再需要為本教學課程建立的資源，請刪除這些資源以避免額外費用。若要這樣做，請完成下列步驟：

- 如果 AWS Mainframe Modernization 應用程式仍在執行，請將其停止。
- 刪除 應用程式。如需詳細資訊，請參閱刪除 AWS Mainframe Modernization 應用程式。
- 刪除執行時間環境。如需詳細資訊，請參閱刪除 AWS 大型主機現代化執行時間環境。

教學課程：設定 Rocket Software 的受管執行期 （ 先前稱為 Micro Focus)

您可以使用 Rocket Software 執行期引擎，在 AWS Mainframe Modernization 受管執行期環境中部署和執行應用程式。本教學課程說明如何使用 Rocket Software 執行期引擎，在 AWS Mainframe Modernization 受管執行期環境中部署和執行 CardDemo 範例應用程式。CardDemo 範例應用程式是一種簡化的信用卡應用程式，專為大型主機現代化使用案例進行測試和展示 AWS 和合作技術而開發。

在教學課程中，您會在其他 中建立資源 AWS 服務。其中包括 Amazon Simple Storage Service、Amazon Relational Database Service AWS Key Management Service和 AWS Secrets Manager。

主題

- 先決條件
- 步驟 1：建立和載入 Amazon S3 儲存貯體
- 步驟 2：建立和設定資料庫
- 步驟 3：建立和設定 AWS KMS key
- 步驟 4：建立和設定 AWS Secrets Manager 資料庫秘密
- 步驟 5：將 sslMode 新增至秘密
- 步驟 6：建立執行時間環境
- 步驟 7：建立應用程式
- 步驟 8：部署應用程式
- 步驟 9：匯入資料集
- 步驟 10：啟動應用程式
- 步驟 11：連線至 CardDemo CICS 應用程式
- 清除資源
- 後續步驟

先決條件

- 請確定您能夠存取 3270 模擬器來使用 CICS 連線。免費和試用 3270 模擬器可從第三方網站取得。或者，您可以啟動 AWS Mainframe Modernization AppStream 2.0 Rocket Software 執行個體，並使用 Rumba 3270 模擬器 （不可免費使用）。

  如需 AppStream 2.0 的相關資訊，請參閱 the section called “教學課程：設定適用於 Enterprise Analyzer 和 Enterprise Developer 的 AppStream 2.0”。

  > **Note**
  >
  > 建立堆疊時，請選擇企業開發人員 (ED) 選項，而非企業分析器 (EA)。

- 下載 CardDemo 範例應用程式，並將下載的檔案解壓縮至任何本機目錄。此目錄將包含名為 的子目錄`CardDemo_runtime`。
- 識別您帳戶中的 VPC，您可以在其中定義在本教學課程中建立的資源。VPC 至少需要兩個可用區域中的子網路。如需 Amazon VPC 的詳細資訊，請參閱 Amazon VPC 的運作方式。

步驟 1：建立和載入 Amazon S3 儲存貯體

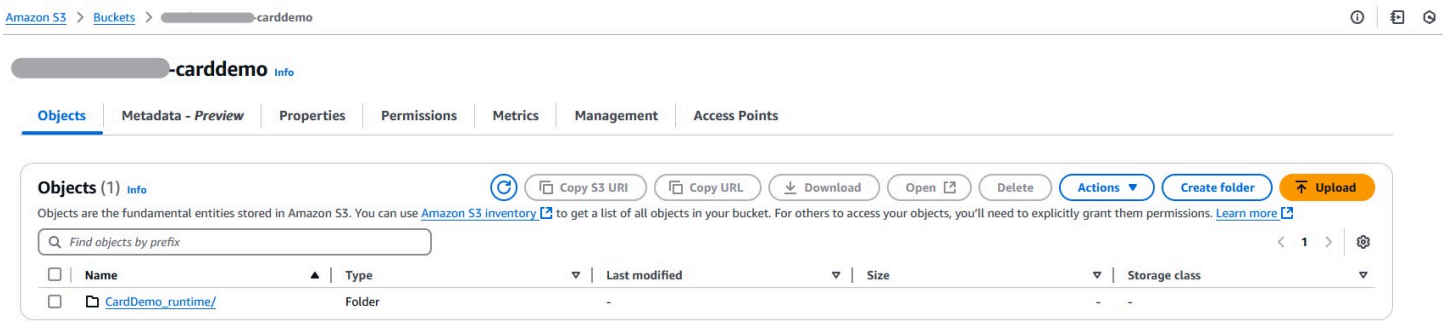
在此步驟中，您會建立 Amazon S3 儲存貯體，並將 CardDemo 檔案上傳至此儲存貯體。在本教學課程稍後，您會使用這些檔案在 AWS Mainframe Modernization Rocket Software Managed Runtime 環境中部署和執行 CardDemo 範例應用程式。

> **Note**
>
> 您不需要建立新的 S3 儲存貯體，但您選擇的儲存貯體必須與本教學中使用的其他資源位於相同的 區域。

建立 Amazon S3 儲存貯體

1. 開啟 Amazon S3 主控台，然後選擇建立儲存貯體。
2. 在一般組態中，選擇您要建置 AWS Mainframe Modernization Rocket Software Managed Runtime 的 AWS 區域。
3. 輸入儲存貯體名稱，例如 `yourname-aws-region-carddemo`。保留預設設定，然後選擇建立儲存貯體。或者，您也可以從現有的 Amazon S3 儲存貯體複製設定，然後選擇建立儲存貯體。
4. 選擇您剛建立的儲存貯體，然後選擇上傳。
5. 在上傳區段中，選擇新增資料夾，然後從本機電腦瀏覽至`CardDemo_runtime`目錄。
6. 選擇上傳以開始上傳程序。上傳時間會根據您的連線速度而有所不同。
7. 上傳完成時，請確認所有檔案都已成功上傳，然後選擇關閉。

您的 Amazon S3 儲存貯體現在包含 `CardDemo_runtime` 資料夾。



如需 S3 儲存貯體的資訊，請參閱建立、設定和使用 Amazon S3 儲存貯體。

步驟 2：建立和設定資料庫

在此步驟中，您會在 Amazon Relational Database Service (Amazon RDS) 中建立 PostgreSQL 資料庫。 Amazon Relational Database Service 在教學課程中，此資料庫包含 CardDemo 範例應用程式用於客戶有關信用卡交易任務的資料集。

在 Amazon RDS 中建立資料庫

1. 開啟 Amazon RDS 主控台。
2. 選擇您要在其中建立資料庫執行個體的 AWS 區域。
3. 在導覽窗格中，選擇 Databases (資料庫)。
4. 選擇建立資料庫，然後選擇標準建立。
5. 針對引擎類型，選擇 PostgreSQL。
6. 選擇 15 或更高版本的引擎。

> **Note**
>
> 儲存引擎版本，因為您稍後在本教學課程中需要它。

7. 在 範本 區段中，選擇 免費方案。
8. 將資料庫執行個體識別符變更為有意義的項目，例如 `MicroFocus-Tutorial`。
9. 避免在 中管理主登入資料 AWS Secrets Manager。請改為輸入主密碼並確認。

> **Note**
>
> 儲存您用於資料庫的使用者名稱和密碼。您將在本教學課程的後續步驟中安全地存放它們。

10. 在連線下，選擇您要建立 AWS Mainframe Modernization 受管執行期環境的 VPC。
11. 選擇建立資料庫。

在 Amazon RDS 中建立自訂參數群組

1. 在 Amazon RDS 主控台導覽窗格中，選擇參數群組，然後選擇建立參數群組。
2. 在建立參數群組視窗中，針對參數群組系列，選取符合您資料庫版本的 Postgres 選項。

> **Note**
>
> 有些 Postgres 版本需要 類型。視需要選取資料庫參數群組。輸入參數群組的群組名稱和描述。

3. 選擇建立。

設定自訂參數群組

1. 選擇新建立的參數群組。
2. 選擇動作，然後選擇編輯。
3. 篩選 `max_prepared_transactions` 並將參數值變更為 100。
4. 選擇 Save Changes (儲存變更)。

將自訂參數群組與資料庫建立關聯

1. 在 Amazon RDS 主控台導覽窗格中，選擇資料庫，然後選擇您要修改的資料庫執行個體。
2. 選擇 Modify (修改)。Modify DB instance (修改資料庫執行個體) 頁面隨即出現。

> **Note**
>
> 在資料庫完成建立和備份之前，修改選項無法使用，這可能需要幾分鐘的時間。

3. 在修改資料庫執行個體頁面上，導覽至其他組態，並將資料庫參數群組變更為參數群組。如果您的參數群組在清單中不可用，請檢查其是否使用正確的資料庫版本建立。
4. 選擇繼續，然後檢查修改摘要。
5. 選擇立即套用以立即套用變更。

6. 選擇修改資料庫執行個體以儲存您的變更。

如需參數群組的詳細資訊，請參閱使用參數群組。

> **Note**
>
> 您也可以搭配 AWS Mainframe Modernization 使用 Amazon Aurora PostgreSQL 資料庫，但沒有免費方案選項。如需詳細資訊，請參閱使用 Amazon Aurora postgreSQL。

步驟 3：建立和設定 AWS KMS key

若要安全地存放 Amazon RDS 執行個體的登入資料，請先建立 AWS KMS key。

建立 AWS KMS key

1. 開啟 Key Management Service 主控台。
2. 選擇 Create Key (建立金鑰)。
3. 保留金鑰類型的對稱預設值，以及金鑰用量的加密和解密。
4. 選擇下一步。
5. 為金鑰提供別名，例如 `MicroFocus-Tutorial-RDS-Key`和選用的描述。
6. 選擇下一步。
7. 勾選使用者或角色旁的方塊，以指派金鑰管理員。
8. 選擇下一步。
9. 勾選使用者或角色旁的方塊，以指派金鑰使用許可。
10. 選擇下一步。
11. 在檢閱畫面上，編輯金鑰政策，然後在現有的「陳述式」陣列中輸入以下內容：

```
{
     "Sid" : "Allow access for Mainframe Modernization Service",
     "Effect" : "Allow",
        "Principal" : {
           "Service" : "m2.amazonaws.com"
                 },
      "Action" : "kms:Decrypt",
      "Resource" : "*"
},
```

此政策會授予使用此特定金鑰政策的 AWS Mainframe Modernization 解密許可。

12. 選擇完成以建立金鑰。

如需詳細資訊，請參閱《 AWS Key Management Service 開發人員指南》中的建立金鑰。

步驟 4：建立和設定 AWS Secrets Manager 資料庫秘密

現在使用 AWS Secrets Manager 和 安全地存放資料庫登入資料 AWS KMS key。

建立和設定 AWS Secrets Manager 資料庫秘密

1. 開啟 Secrets Manager 主控台。
2. 在導覽窗格中，選擇 Secrets (祕密)。
3. 在秘密中，選擇儲存新的秘密。
4. 將秘密類型設定為 Amazon RDS 資料庫的登入資料。
5. 輸入您在建立資料庫時指定的登入資料。
6. 在加密金鑰下，選取您在步驟 3 中建立的金鑰。
7. 在資料庫區段中，選取您為此教學課程建立的資料庫，然後選擇下一步。
8. 在秘密名稱下，輸入名稱，例如 `MicroFocus-Tutorial-RDS-Secret`和選用描述。
9. 在資源許可區段中，選擇編輯許可，並以下列政策取代內容：

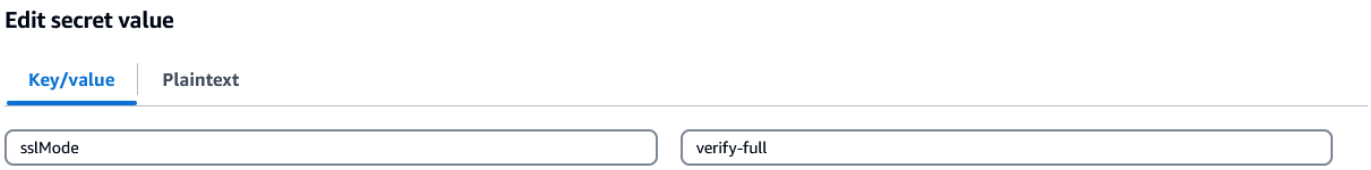
```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect" : "Allow",
            "Principal" : {
                "Service" : "m2.amazonaws.com"
            },
            "Action" : "secretsmanager:GetSecretValue",
            "Resource" : "*"
        }
    ]
}
```

10. 選擇儲存。
11. 針對後續畫面選擇下一步，然後選擇儲存。

步驟 5：將 sslMode 新增至秘密

將 sslMode 新增至秘密

1. 重新整理秘密清單以查看新的秘密。
2. 選擇步驟 4 中新建立的秘密，並記下 ，Secret ARN因為您稍後在教學課程中需要它。
3. 在秘密的概觀索引標籤中，選擇擷取秘密值。
4. 選擇編輯，然後選擇新增列。
5. 新增值sslMode為 的 金鑰verify-full：

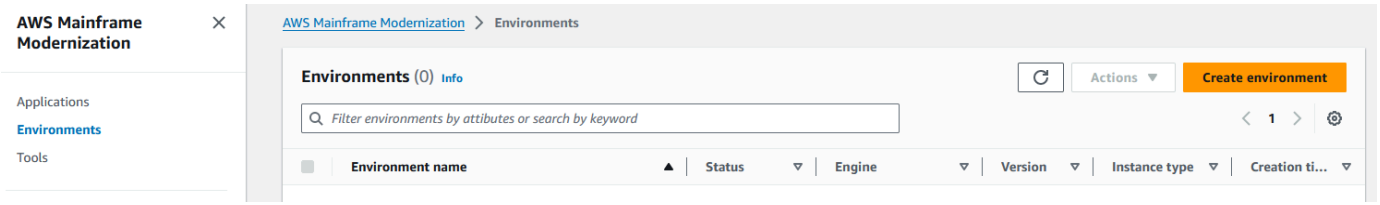


6. 選擇儲存。

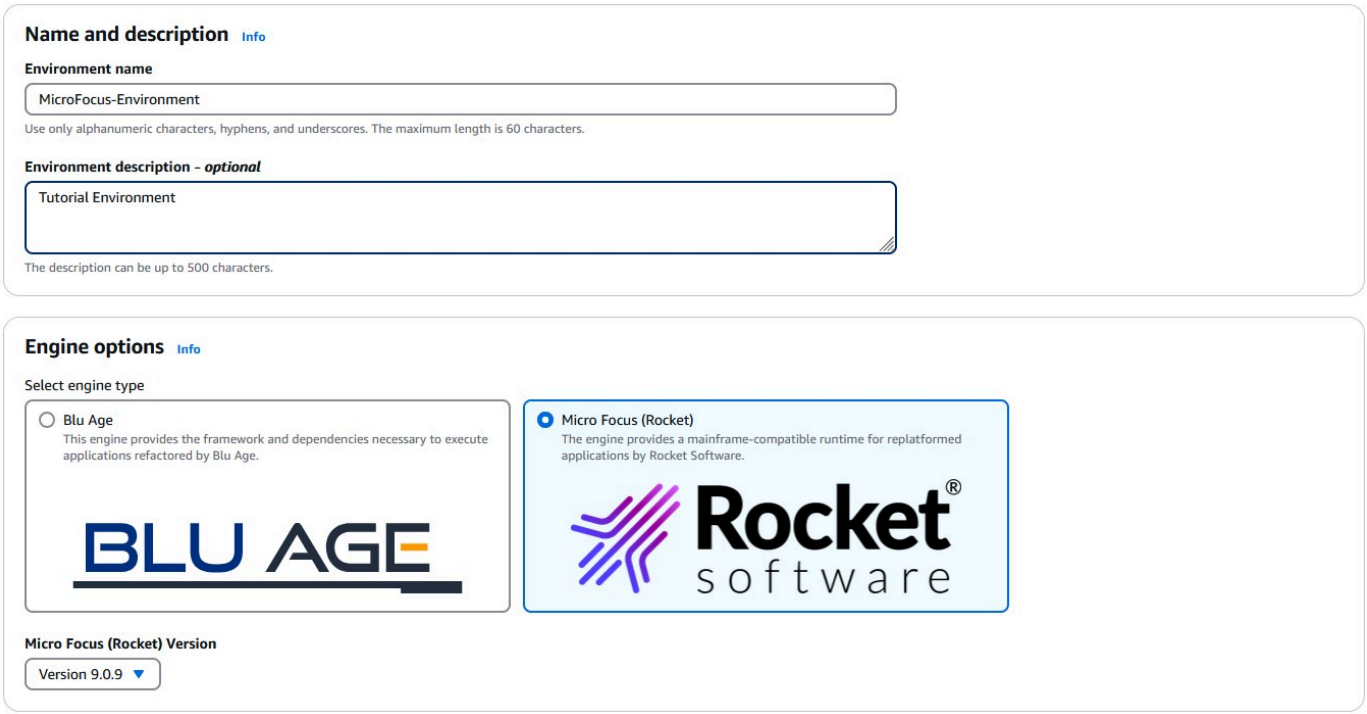
步驟 6：建立執行時間環境

建立執行期環境

1. 開啟 AWS Mainframe Modernization 主控台。
2. 在導覽窗格中，選擇 Environments (環境)。然後選擇建立環境。



3. 在指定基本資訊下，
   a. 輸入 MicroFocus-Environment 做為環境名稱。
   b. 在引擎選項下，確定已選取 Micro Focus (Rocket)。
   c. 選擇最新的 Micro Focus (Rocket) 版本。
   d. 選擇下一步。



4. 設定環境

   a. 在可用性下，選擇高可用性叢集。

   b. 在資源下，選擇執行個體類型的 M2.c5.large 或 M2.m5.large，以及您想要的執行個體數量。指定最多兩個執行個體。

   c. 在安全和網路下，選擇允許部署到此環境的應用程式可公開存取，並選擇至少兩個公有子網路。

   d. 選擇下一步。

Specify configurations Info

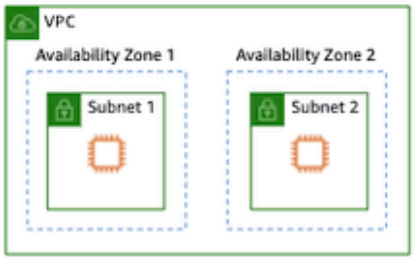
Availability Info

Choose the availability pattern for your environment.

- Standalone runtime environment
  Sets up a single instance in a single availability zone. Does not guarantee high availability but costs less.



- High availability cluster
  Sets up redundant instances across two availability zones. Enables higher availability but costs more.



Resources

Instance type
Choose the instance type for your high availability cluster.

M2.m5.large

Desired capacity
Specify the desired number of instances.

2

Security and network

☑ Allow applications deployed to this environment to be publicly accessible.

Virtual Private Cloud (VPC)
Choose the VPC where you want to create the environment.

Default vpc-15

Subnets
Choose one or more subnets for a high availability setup.

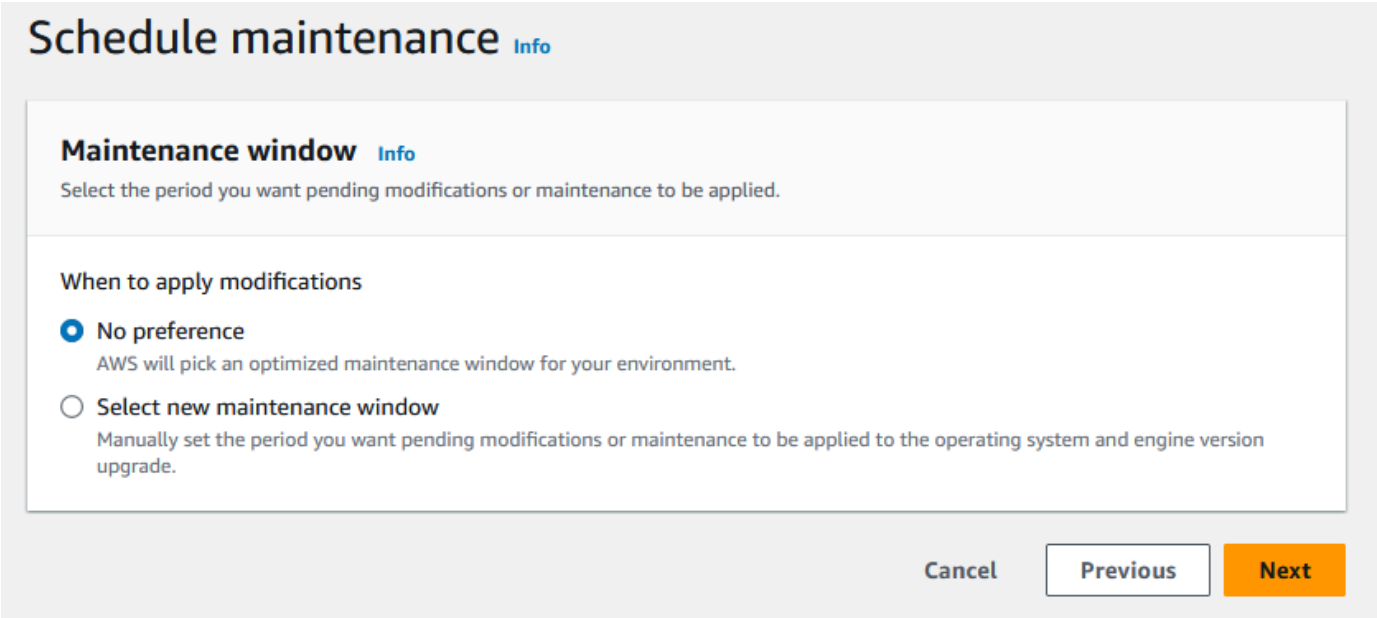
Choose subnets

subnet-56f1a | us-west-2a ×

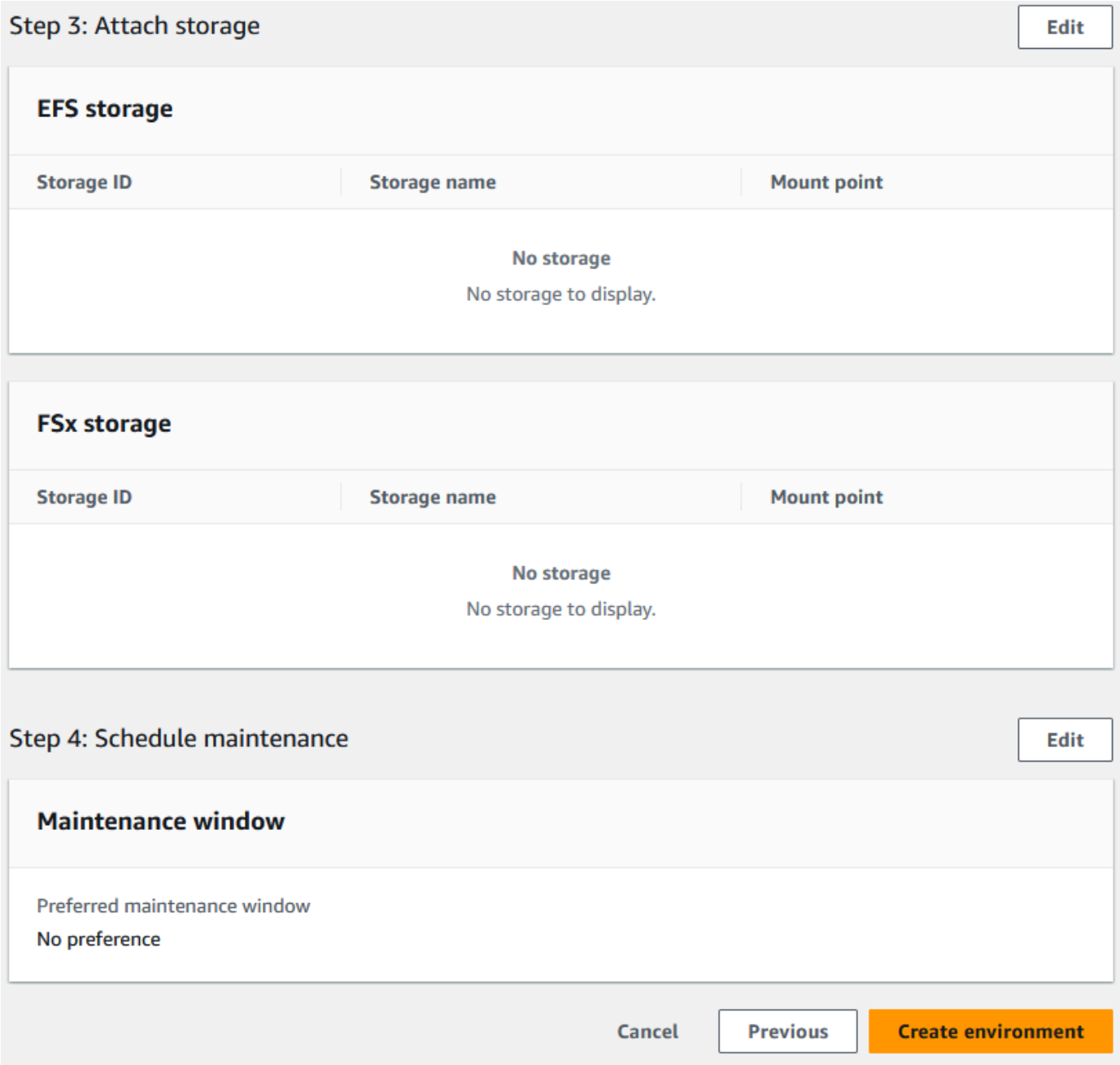
subnet-6685 | us-west-2b ×

Security groups
Choose one or more security groups for the chosen VPC.

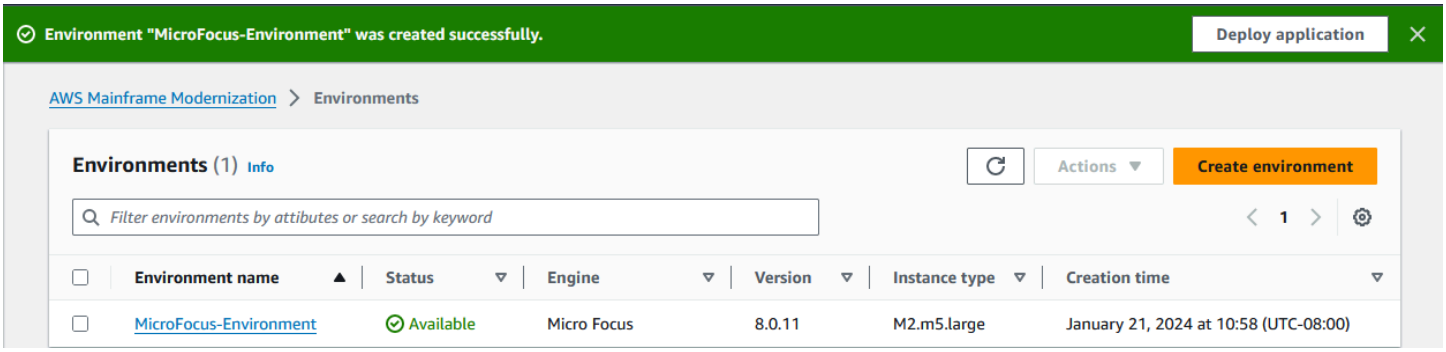
5. 在連接儲存頁面上，選擇下一步。
6. 在排程維護頁面上，選擇無偏好設定，然後選擇下一步。



7. 在檢閱和建立頁面上，檢閱您為執行時間環境提供的所有組態，然後選擇建立環境。



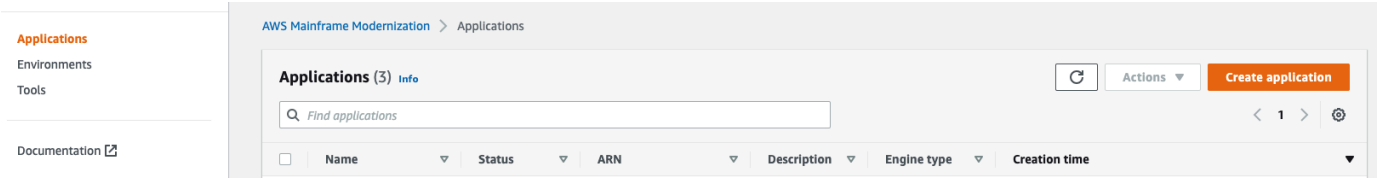
當您建立環境後，會出現一個橫幅，指出 `Environment` *`name`* `was created successfully`，且狀態欄位會變更為可用。環境建立程序需要幾分鐘的時間，但您可以在執行時繼續後續步驟。



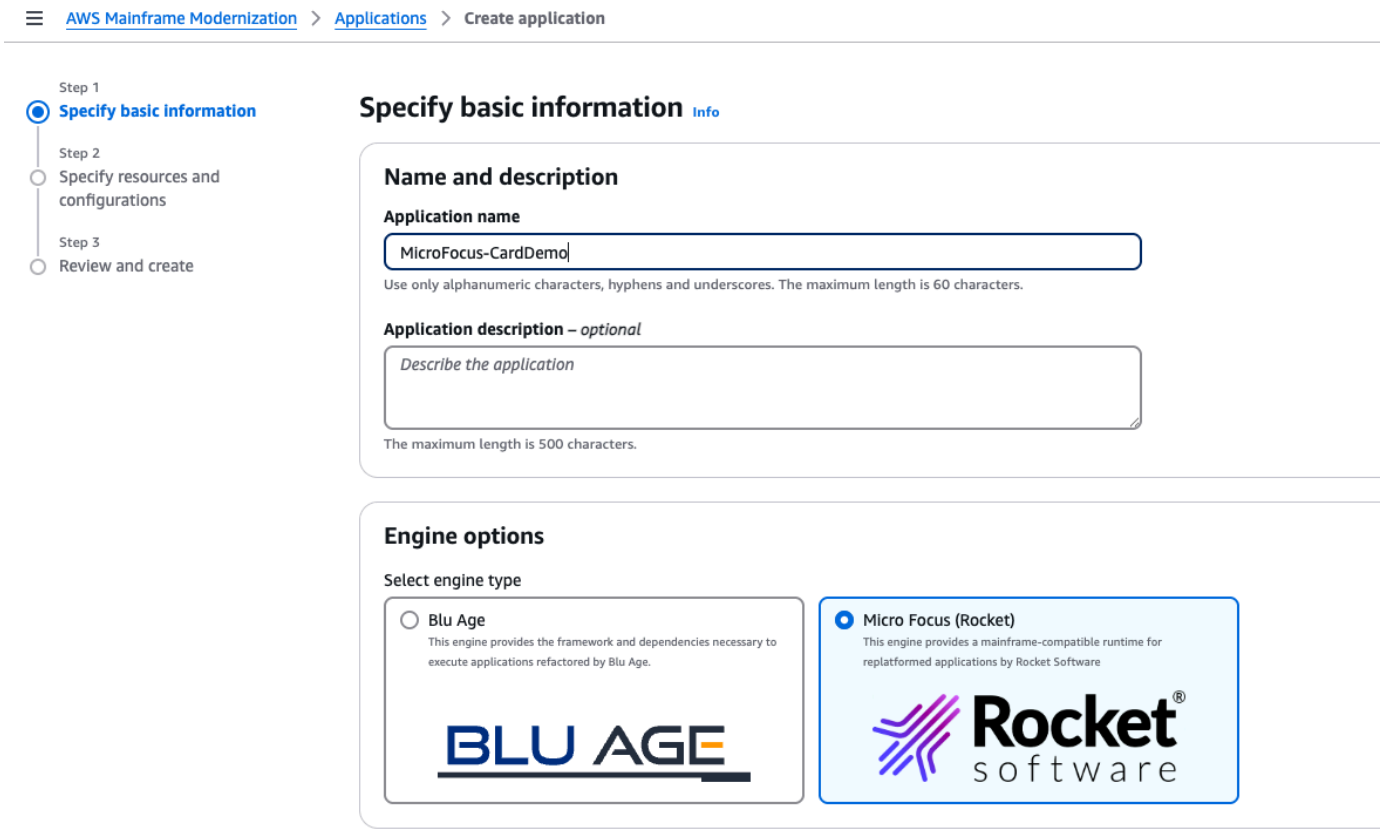
步驟 7：建立應用程式

建立應用程式

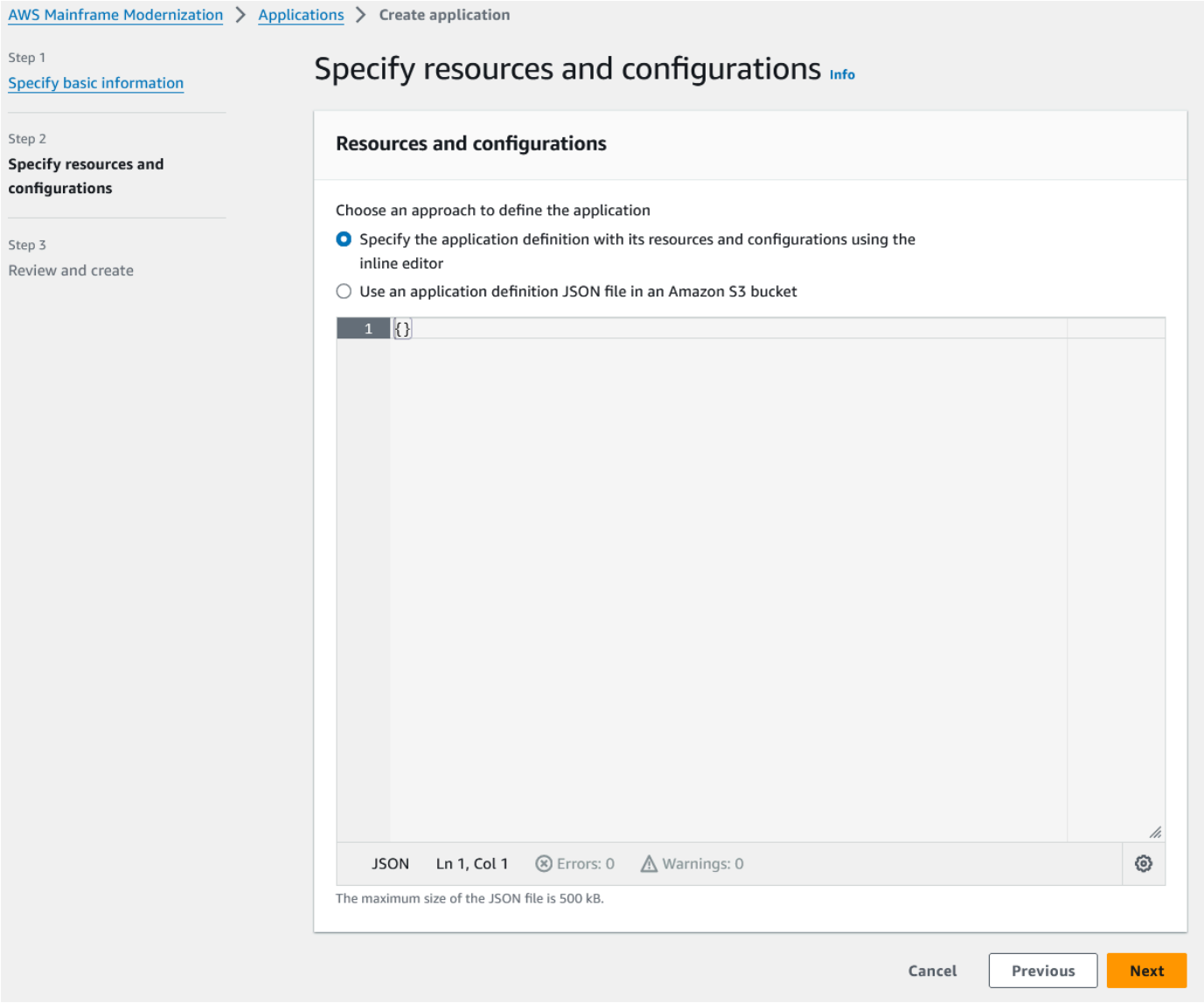
1. 在導覽窗格中，選擇 Applications (應用程式)。然後選擇建立應用程式。



2. 在建立應用程式頁面的指定基本資訊下，`MicroFocus-CardDemo`輸入應用程式名稱和引擎類型下的 ，確定已選取 Micro Focus (Rocket)。然後選擇下一步。



3. 在指定資源和組態下，選擇使用內嵌編輯器，以其資源和組態指定應用程式定義的選項。



在編輯器中輸入下列應用程式定義：

```
{
  "template-version": "2.0",
  "source-locations": [
    {
      "source-id": "s3-source",
      "source-type": "s3",
      "properties": {
        "s3-bucket": "yourname-aws-region-carddemo",
        "s3-key-prefix": "CardDemo_runtime"
      }
    }
```

```
  ],
  "definition": {
    "listeners": [
      {
        "port": 6000,
        "type": "tn3270"
      }
    ],
    "dataset-location": {
      "db-locations": [
        {
          "name": "Database1",
          "secret-manager-arn":
 "arn:aws:secretsmanager:Region:123456789012:secret:MicroFocus-Tutorial-RDS-Secret-
xxxxxx"
        }
      ]
    },
    "batch-settings": {
      "initiators": [
        {
          "classes": [
               "A",
               "B"
            ],
          "description": "initiator_AB...."
        },
        {
          "classes": [
                "C",
                "D"
             ],
          "description": "initiator_CD...."
        }
      ],
      "jcl-file-location": "${s3-source}/catalog/jcl"
    },
    "cics-settings": {
      "binary-file-location": "${s3-source}/loadlib",
      "csd-file-location": "${s3-source}/rdef",
      "system-initialization-table": "CARDSIT"
    },
    "xa-resources": [
      {
```

```
        "name": "XASQL",
        "secret-manager-arn":
 "arn:aws:secretsmanager:Region:123456789012:secret:MicroFocus-Tutorial-RDS-Secret-
xxxxxx",
        "module": "${s3-source}/xa/ESPGSQLXA64.so"
      }
    ]
  }
}
```

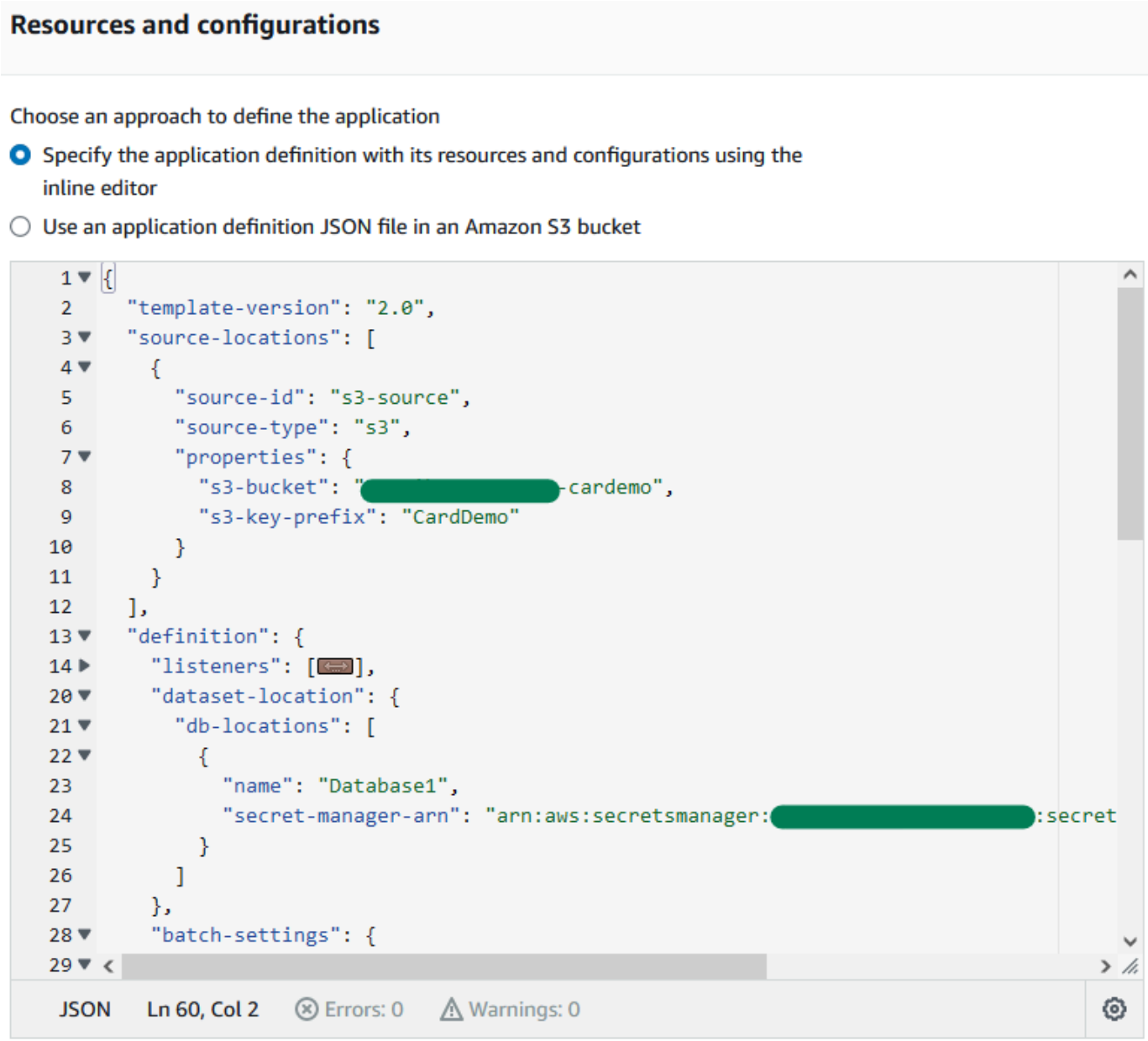
> **Note**
>
> 此檔案可能會有所變更。

4. 在來源位置的屬性物件中編輯應用程式 JSON，如下所示：

   a. 將 的值取代`s3_bucket`為您在步驟 1 中建立的 Amazon S3 儲存貯體名稱。

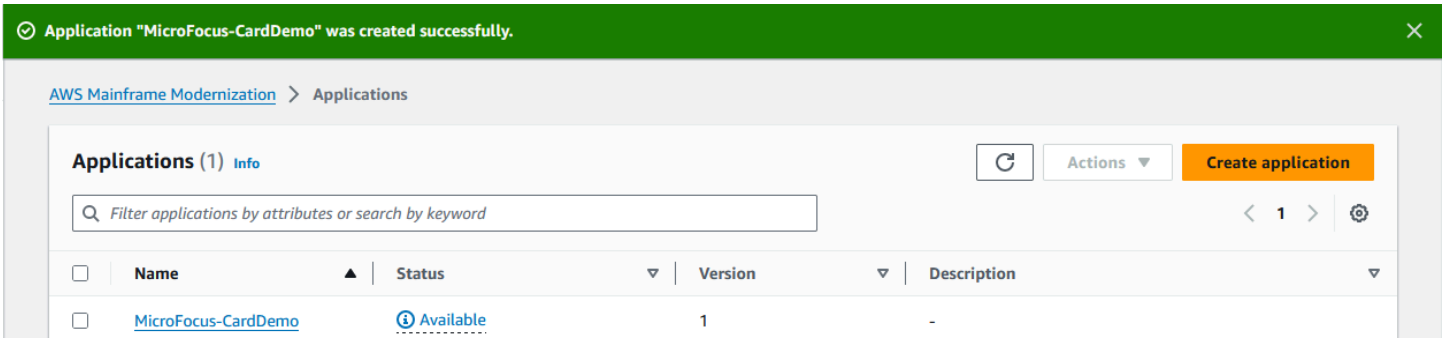
   b. 將 的值取代`s3-key-prefix`為您上傳 CardDemo 範例檔案的資料夾 （ 金鑰字首 ）。如果您直接將`CardDemo`目錄上傳到 Amazon S3 儲存貯體，則`s3-key-prefix`不需要變更 。

   c. 將這兩個`secret-manager-arn`值取代為您在步驟 4 中建立的資料庫秘密的 ARN。



如需應用程式定義的詳細資訊，請參閱 Rocket Software （先前稱為 Micro Focus) 應用程式定義。

5. 選擇 Next (下一步) 繼續。
6. 在檢閱和建立頁面上，檢閱您提供的資訊，然後選擇建立應用程式。

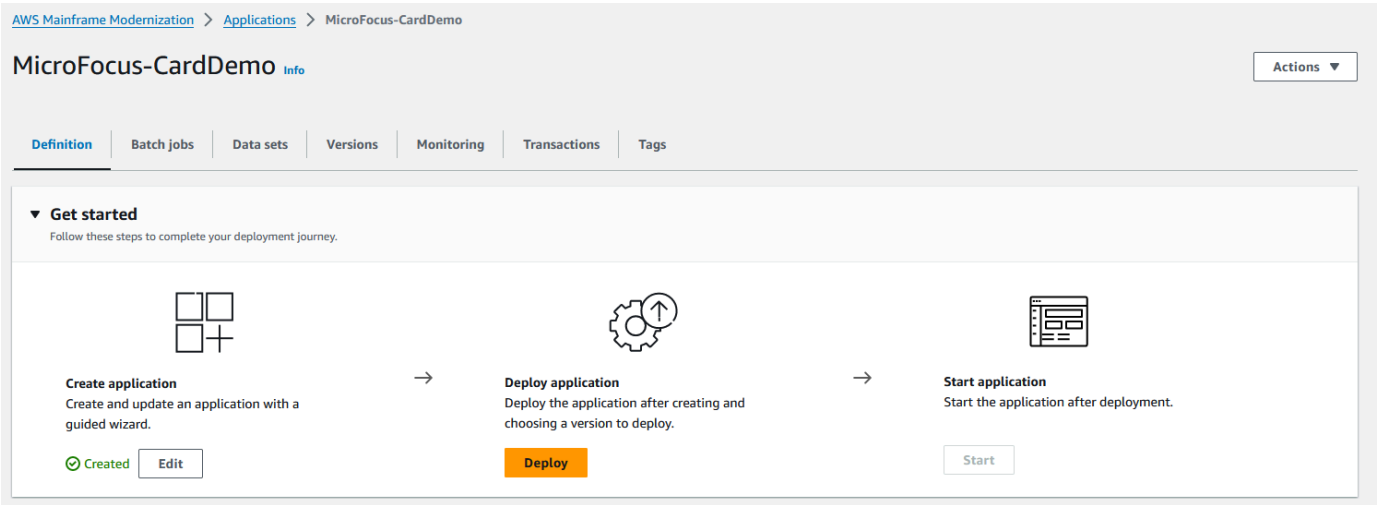


建立應用程式後，會出現一個橫幅，指出 Application *name* was created successfully。狀態欄位會變更為可用。

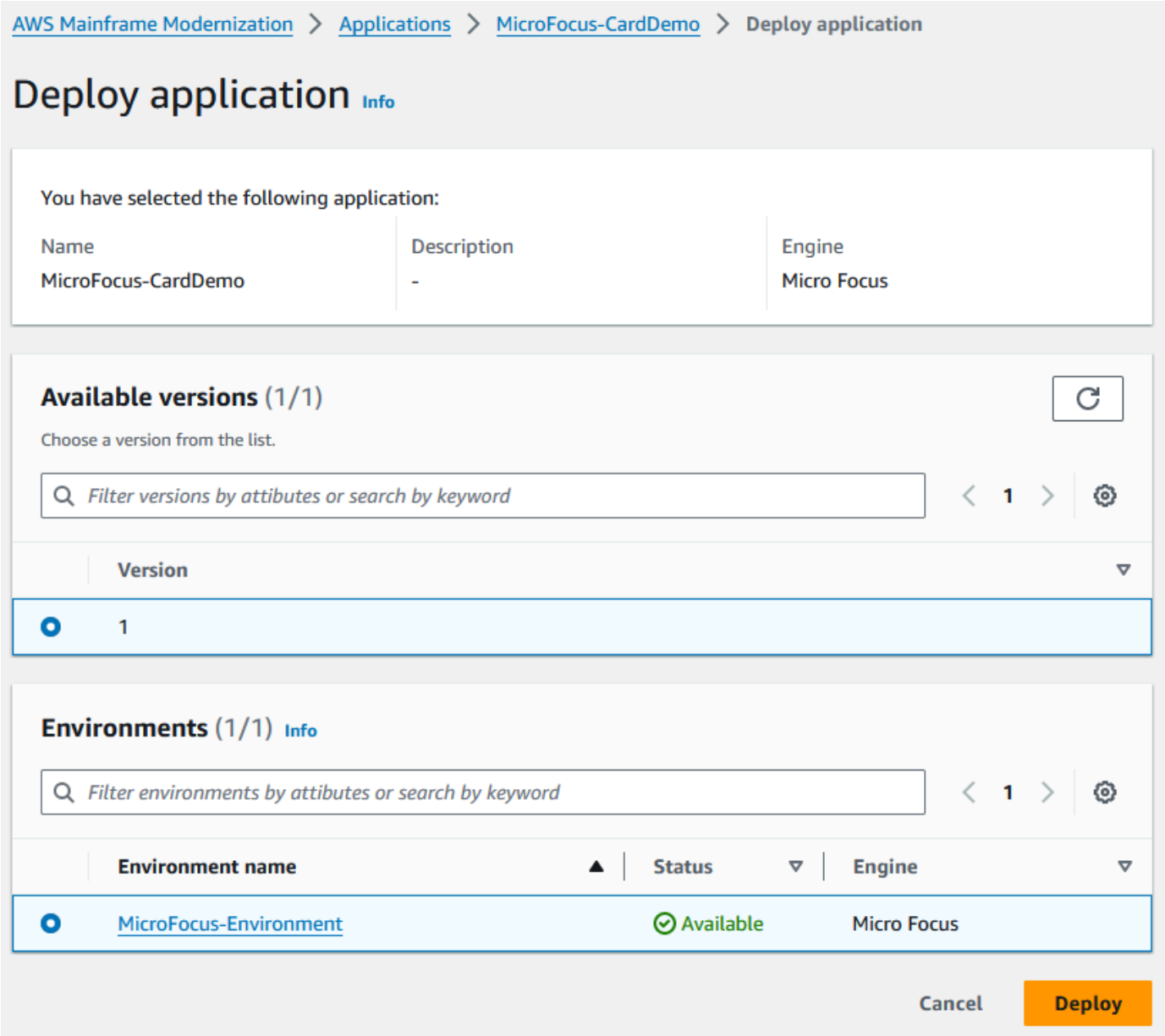
步驟 8：部署應用程式

部署應用程式

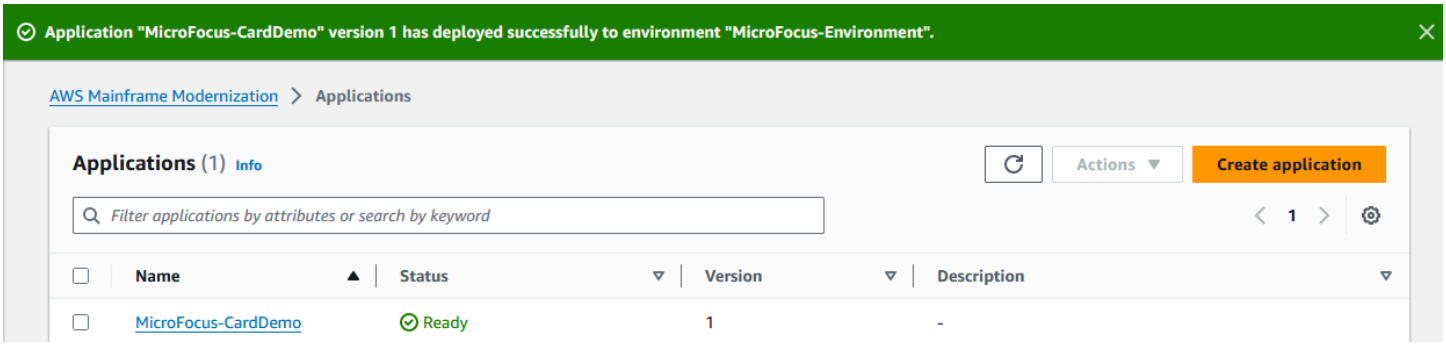
1. 在導覽窗格中，選擇應用程式，然後選擇 MicroFocus-CardDemo。
2. 在部署應用程式下，選擇部署。



3. 選擇最新版本的應用程式和您先前建立的環境，然後選擇部署。



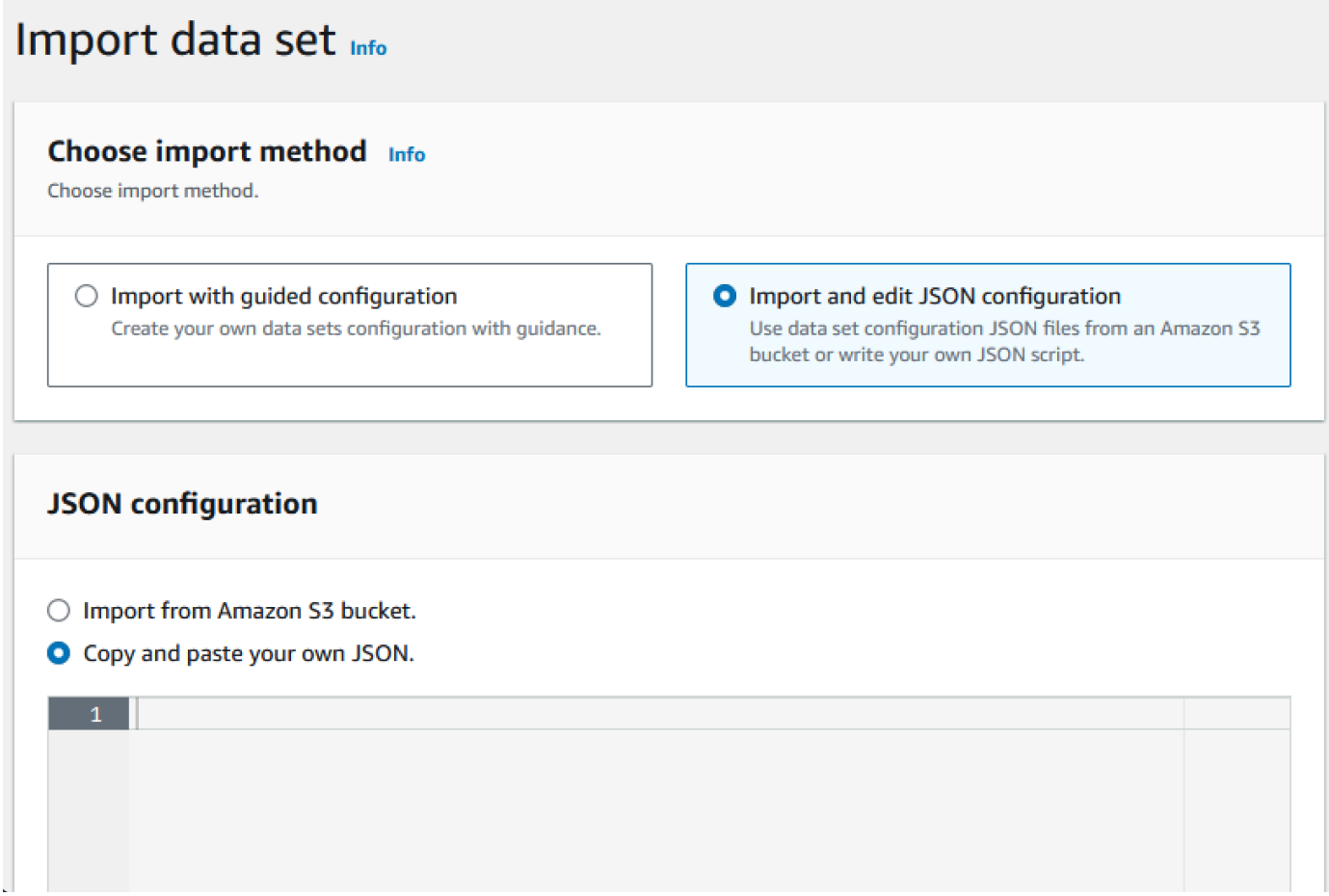
當 CardDemo 應用程式成功部署時，狀態會變更為就緒。



步驟 9：匯入資料集

匯入資料集

1. 在導覽窗格中，選擇應用程式，然後選擇應用程式。
2. 選擇資料集索引標籤。然後選擇 Import (匯入)。
3. 選擇匯入和編輯 JSON 組態，然後選擇複製並貼上您自己的 JSON 選項。



4. 複製並貼上下列 JSON，但尚未選擇「提交」。此 JSON 包含示範應用程式所需的所有資料集，但需要您的 Amazon S3 儲存貯體詳細資訊。

```
{
    "dataSets": [
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.ACCTDATA.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
```

```
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 11,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 300,
                    "max": 300
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo_runtime/catalog/data/
AWS.M2.CARDDEMO.ACCTDATA.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CARDDATA.VSAM.AIX.PATH",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 11,
                            "offset": 16
                        }
                    }
                },
                "recordLength": {
                    "min": 150,
                    "max": 150
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo_runtime/catalog/data/
AWS.M2.CARDDEMO.CARDDATA.VSAM.KSDS.DAT"
            }
        },
```

```
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CARDDATA.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 16,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 150,
                    "max": 150
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo_runtime/catalog/data/
AWS.M2.CARDDEMO.CARDDATA.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CARDXREF.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 16,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 50,
                    "max": 50
                }
```

```
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo_runtime/catalog/data/
AWS.M2.CARDDEMO.CARDXREF.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CUSTDATA.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 9,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 500,
                    "max": 500
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo_runtime/catalog/data/
AWS.M2.CARDDEMO.CUSTDATA.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.CARDXREF.VSAM.AIX.PATH",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 11,
                            "offset": 25
                        }
```

```
                    }
                },
                "recordLength": {
                    "min": 50,
                    "max": 50
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo_runtime/catalog/data/
AWS.M2.CARDDEMO.CARDXREF.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 16,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 350,
                    "max": 350
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo_runtime/catalog/data/
AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT"
            }
        },
        {
            "dataSet": {
                "storageType": "Database",
                "datasetName": "AWS.M2.CARDDEMO.USRSEC.VSAM.KSDS",
                "relativePath": "DATA",
                "datasetOrg": {
                    "vsam": {
```

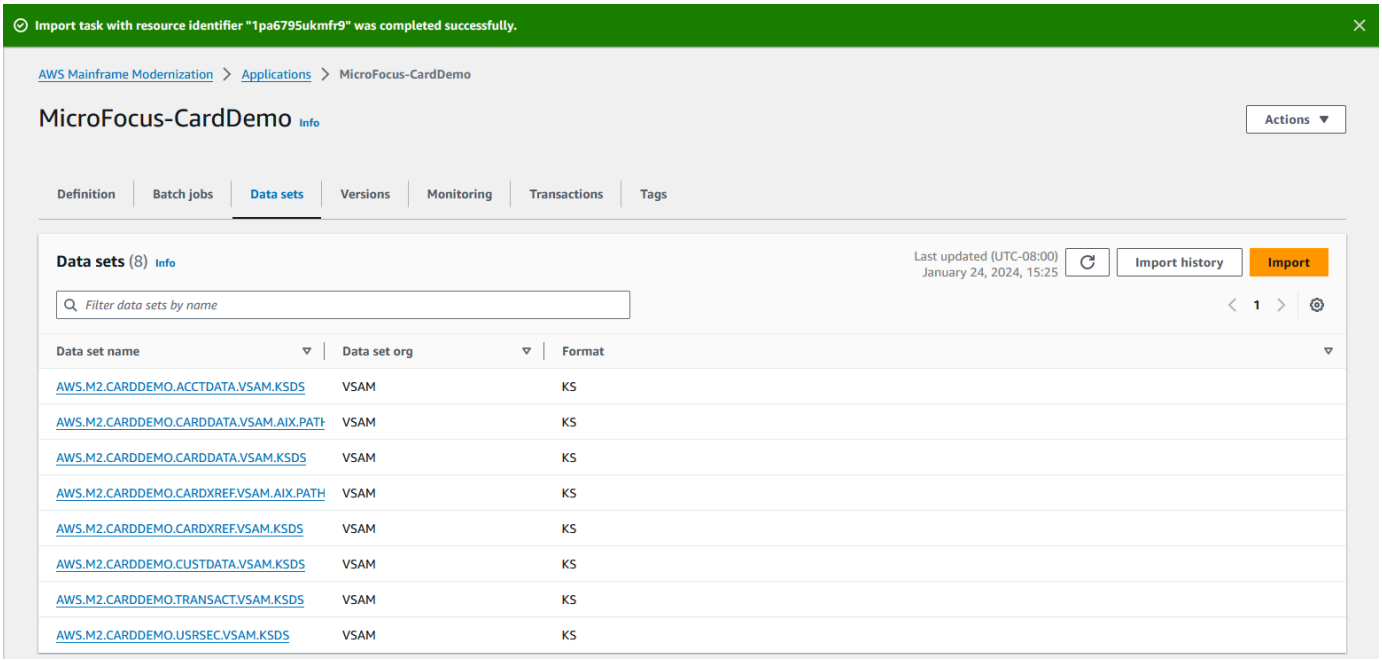
```
                        "format": "KS",
                        "encoding": "A",
                        "primaryKey": {
                            "length": 8,
                            "offset": 0
                        }
                    }
                },
                "recordLength": {
                    "min": 80,
                    "max": 80
                }
            },
            "externalLocation": {
                "s3Location": "s3://<s3-bucket-name>/CardDemo_runtime/catalog/data/
AWS.M2.CARDDEMO.USRSEC.VSAM.KSDS.DAT"
            }
        }
    ]
}
```

5. 以包含 CardDemo 資料夾的 Amazon S3 儲存貯體名稱取代每次出現 `<s3-bucket-name>`（有八個），例如 `your-name-aws-region-carddemo`。

> **Note**
>
> 若要複製 Amazon S3 中資料夾的 Amazon S3 URI，請選取 資料夾，然後選擇複製 Amazon S3 URI。

6. 選擇提交。

匯入完成時，會出現橫幅並顯示下列訊息：會顯示匯入資料集的`Import task with resource identifier name was completed successfully.`清單。

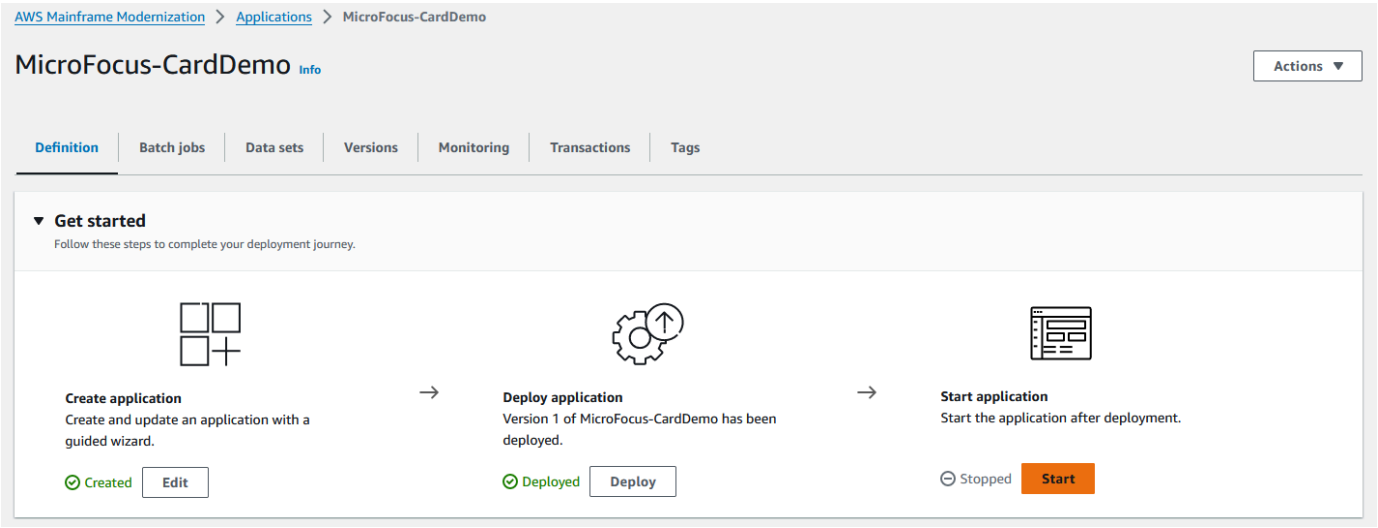


您也可以在資料集標籤上選擇匯入歷史記錄，以檢視所有資料集匯入的狀態。

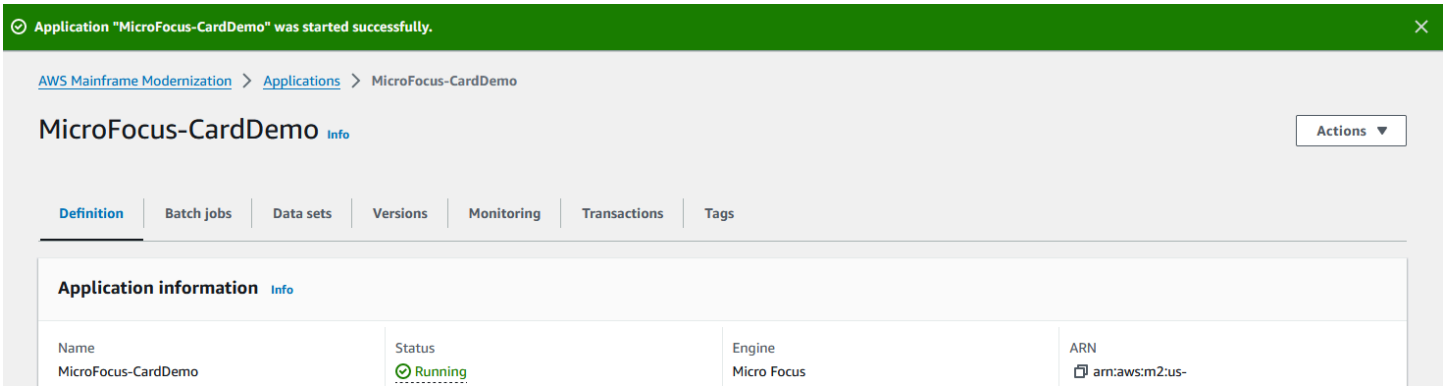
步驟 10：啟動應用程式

啟動應用程式

1. 在導覽窗格中，選擇應用程式，然後選擇應用程式。
2. 選擇啟動應用程式。



當 CardDemo 應用程式開始成功執行時，會出現橫幅並顯示下列訊息：`Application name was started successfully`。Status 欄位會變更為執行中。



步驟 11：連線至 CardDemo CICS 應用程式

連線之前，請確定您為應用程式指定的 VPC 和安全群組與您套用的網路界面相同的 VPC 和安全群組。

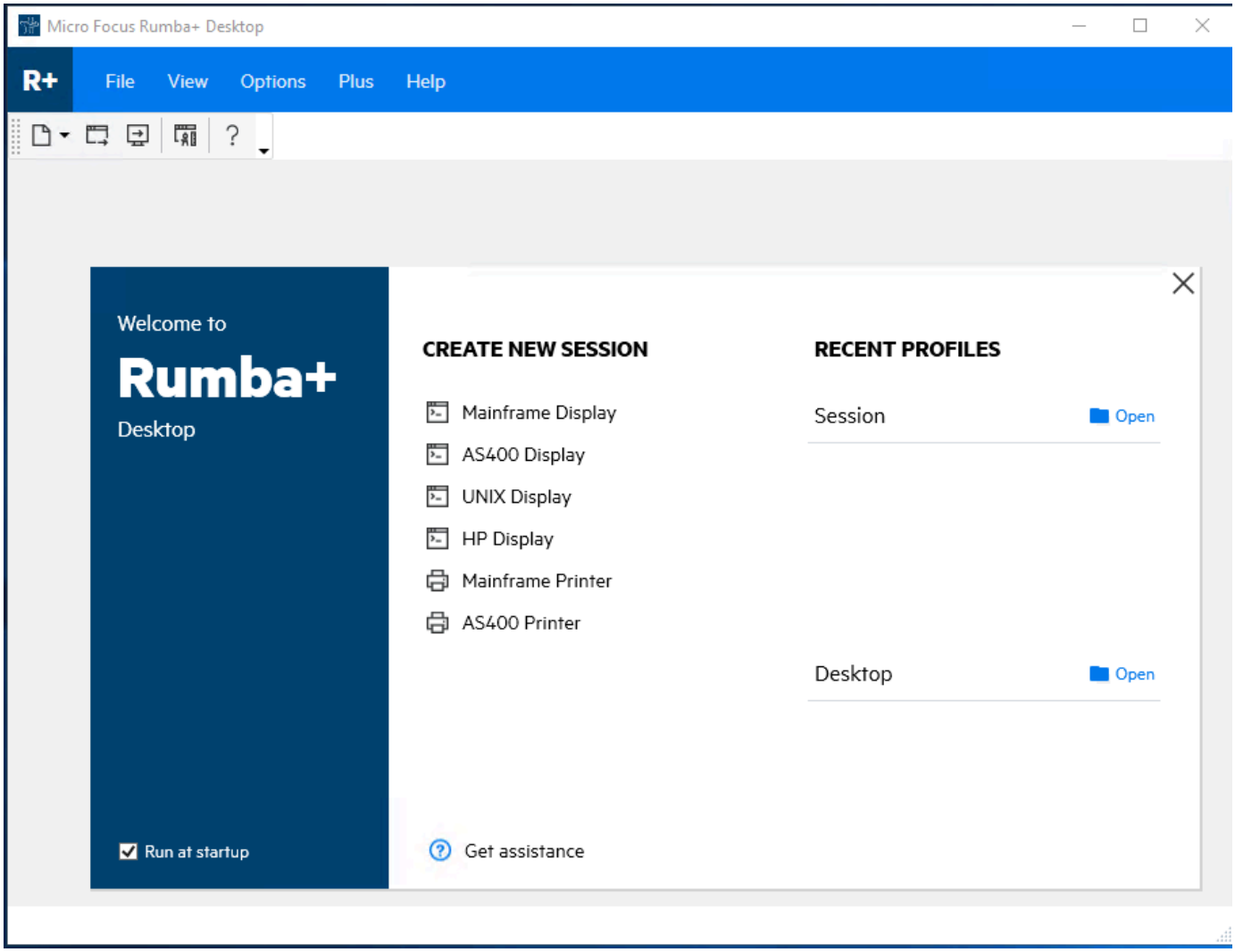
若要設定 TN3270 連線，您也需要 DNS 主機名稱和應用程式的連接埠。

使用終端機模擬器設定應用程式並將其連接至大型主機

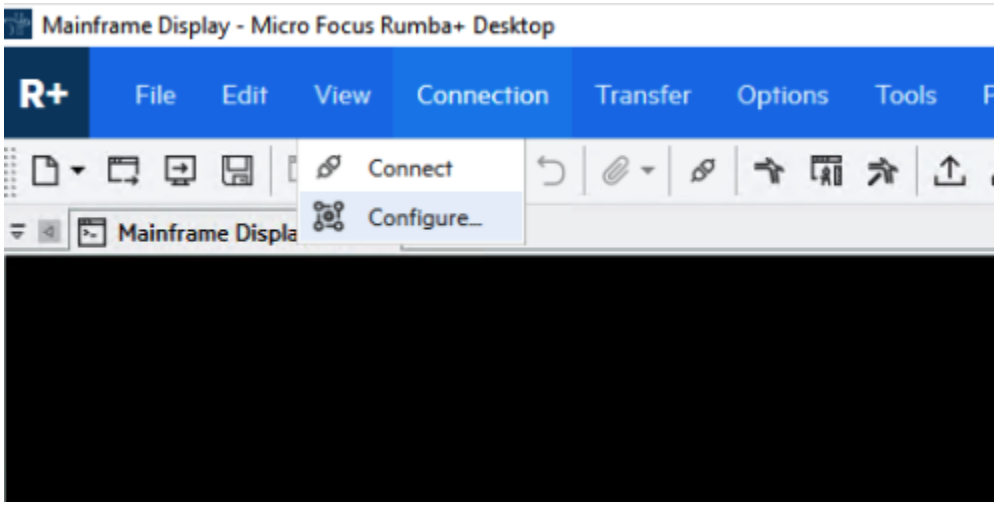
1. 開啟 AWS Mainframe Modernization 主控台，然後選擇應用程式，然後選擇 `MicroFocus-CardDemo`。
2. 選擇複製圖示以複製 DNS 主機名稱。也請務必記下連接埠號碼。
3. 啟動終端機模擬器。本教學課程使用 Micro Focus Rumba+。

> **Note**
>
> 組態步驟因模擬器而異。

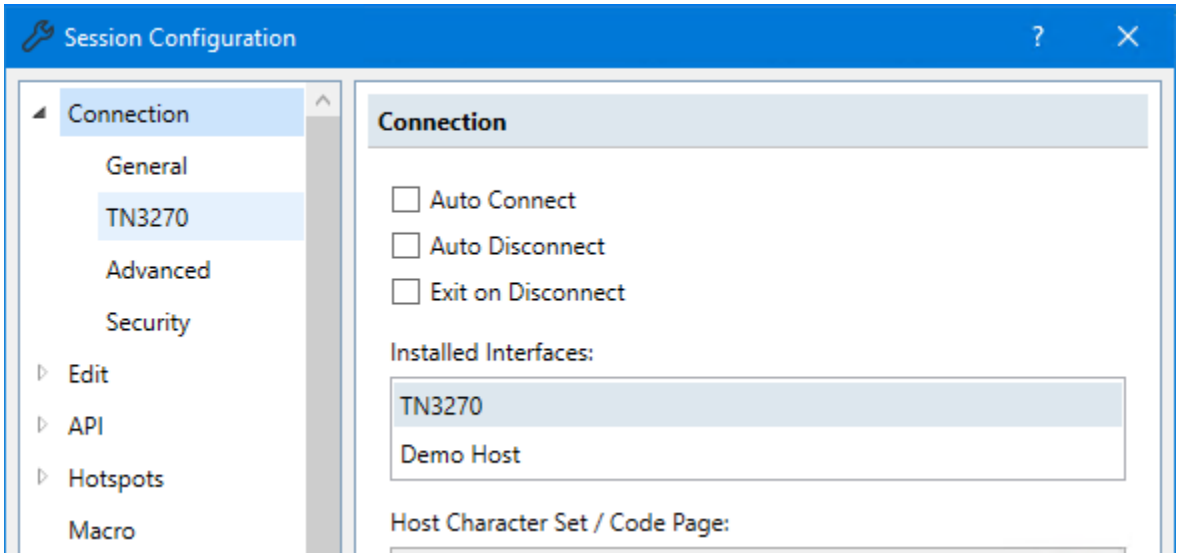
4. 選擇大型主機顯示。



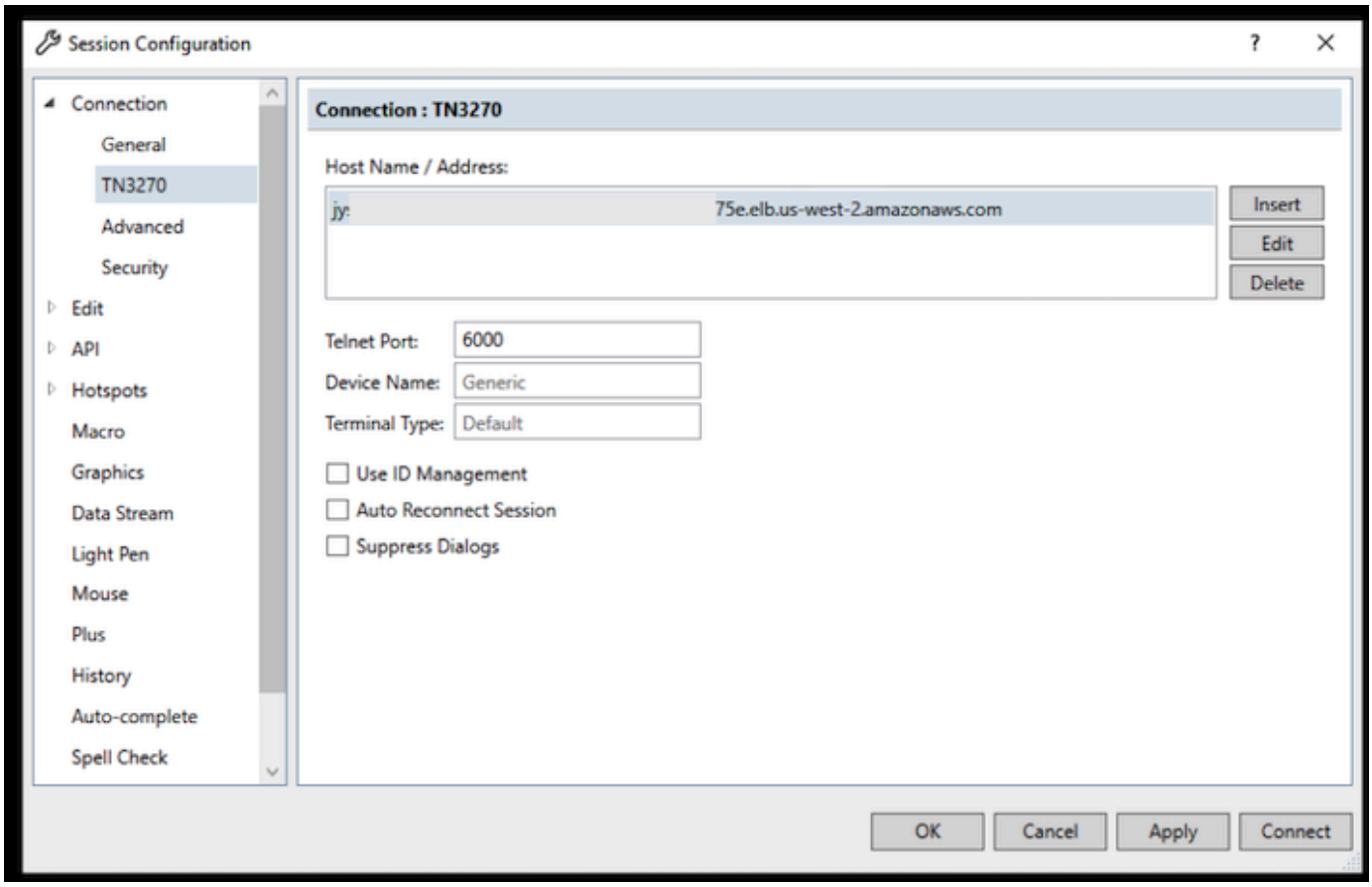
5. 選擇連線，然後選擇設定。



6. 在已安裝的界面下，選擇 TN3270，然後在連線功能表下TN3270再次選擇 。



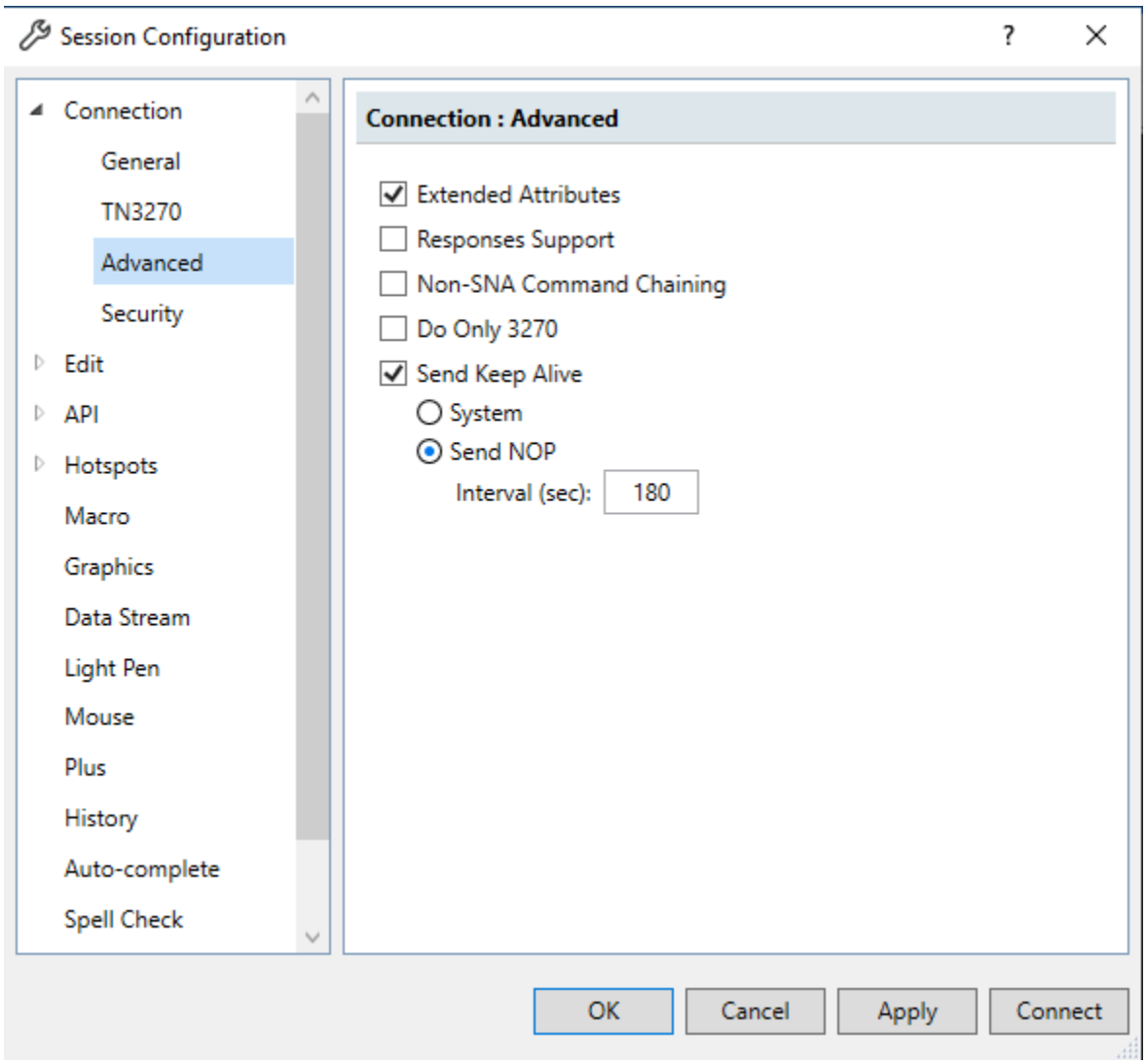
7. 選擇插入，然後貼上DNS Hostname應用程式的 。6000 為 Telnet 連接埠指定 。



> **Note**
>
> 如果您在瀏覽器中使用 AWS AppStream 2.0，並難以貼上值，請參閱對 AppStream 2.0 使用者問題進行故障診斷。

8. 在連線下，選擇進階，然後選擇傳送保持運作狀態和傳送 NOP，然後輸入 180 作為間隔。

> **Note**
>
> 將 TN3270 終端機上的保持運作設定設定為至少 180 秒，有助於確保 Network Load Balancer 不會捨棄您的連線。

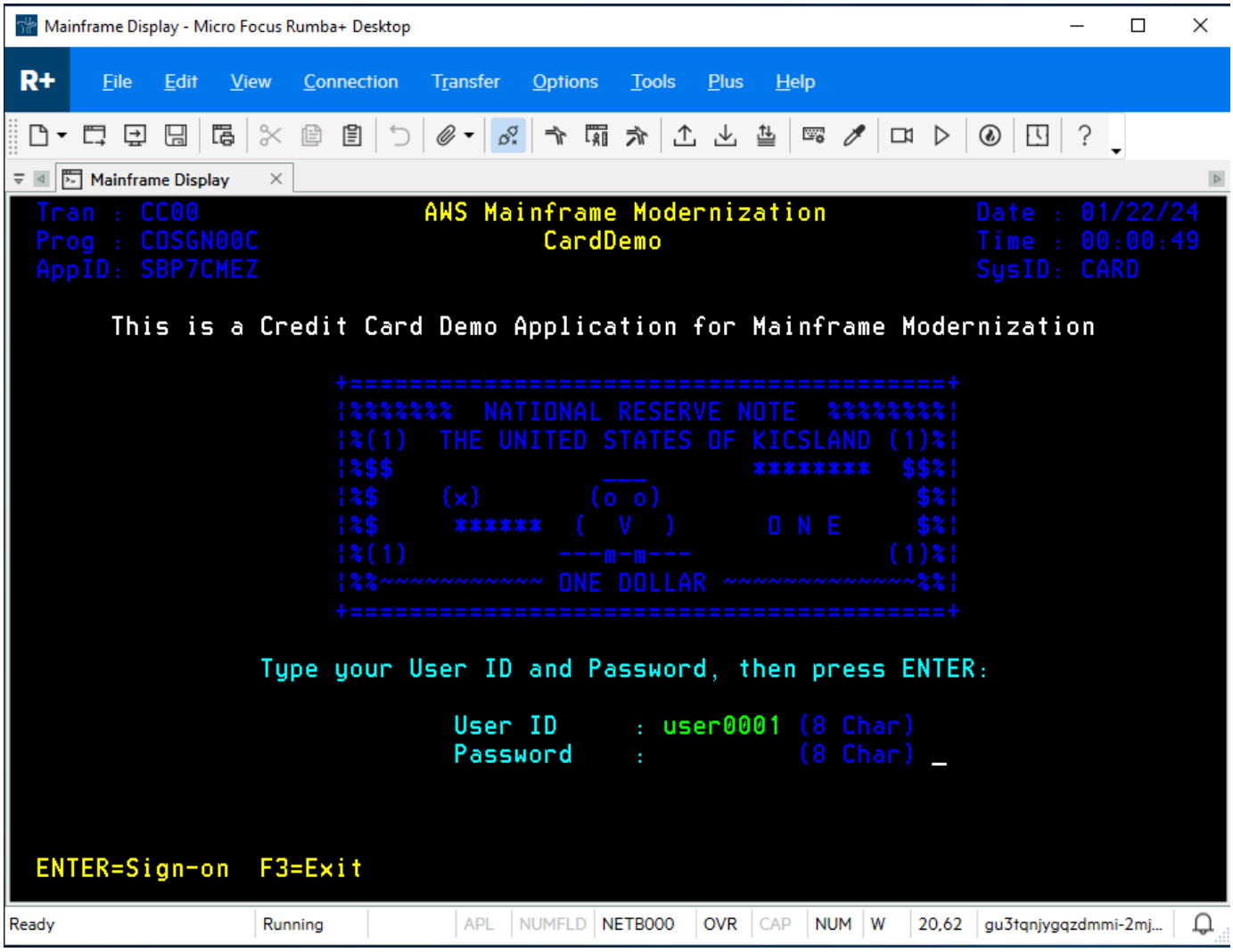


9. 選擇連線。

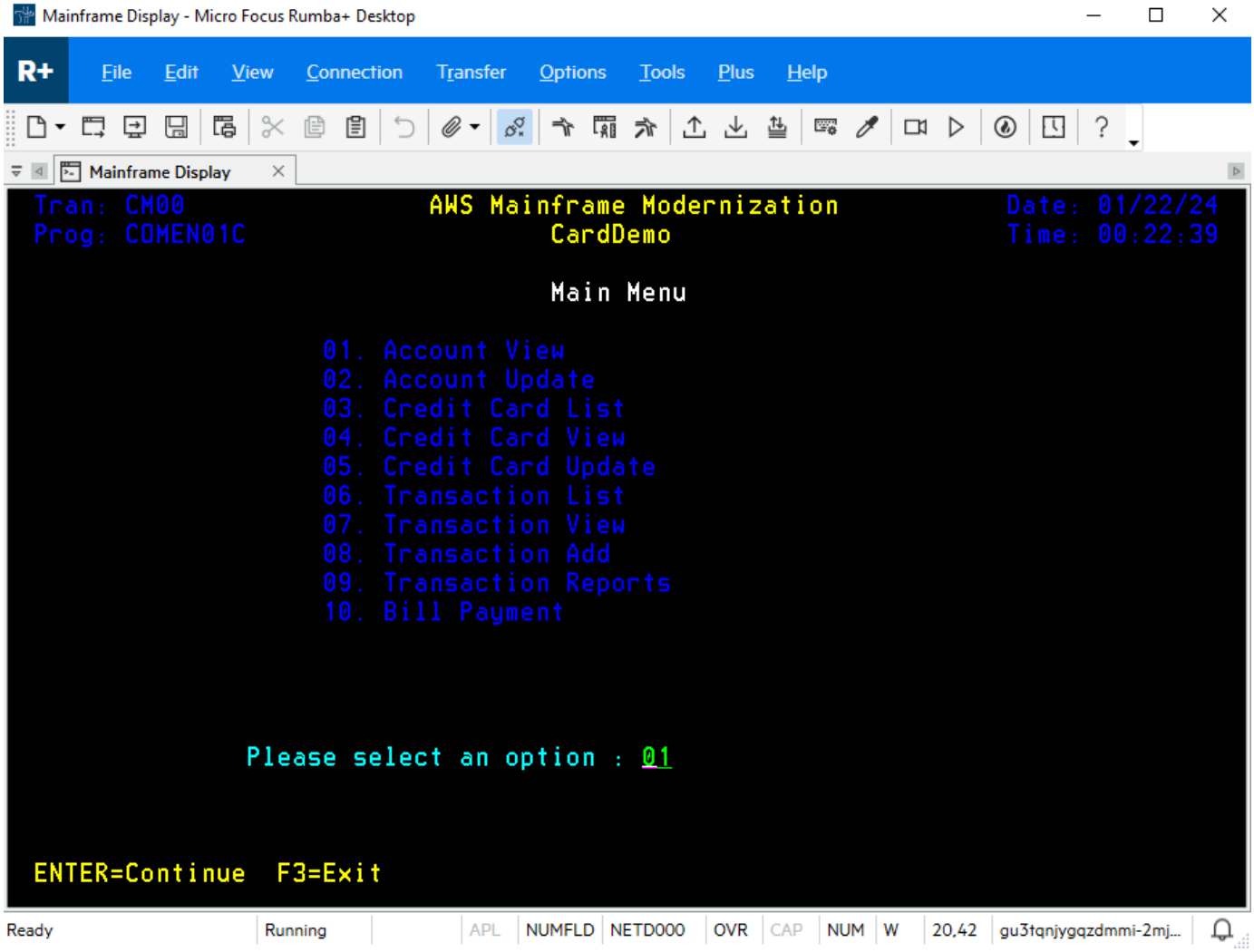
> **Note**
>
> 如果連線失敗：
>
> - 如果您使用的是 AppStream 2.0，請確認為應用程式環境指定的 VPC 和安全群組與 AppStream 2.0 機群相同。
> - 使用 VPC Reachability Analyzer 來分析連線。您可以透過 主控台存取 Reachability Analyzer。
> - 做為診斷步驟，請嘗試新增或變更應用程式的安全群組傳入規則，以允許來自任何地方的連接埠 6000 流量（即 CIDR 區塊 0.0.0.0/0)。如果您成功連線，則表示安全群組封鎖流量。將安全群組來源變更為更具體的內容。如需安全群組的詳細資訊，請參閱安全群組基本概念。

10. 輸入 `USER0001` 表示使用者名稱，輸入 `password` 表示密碼。

> **Note**
>
> 在 Rumba 中，Clear 的預設值為 ctrl-shift-z，重設的預設值為 ctrl-r。

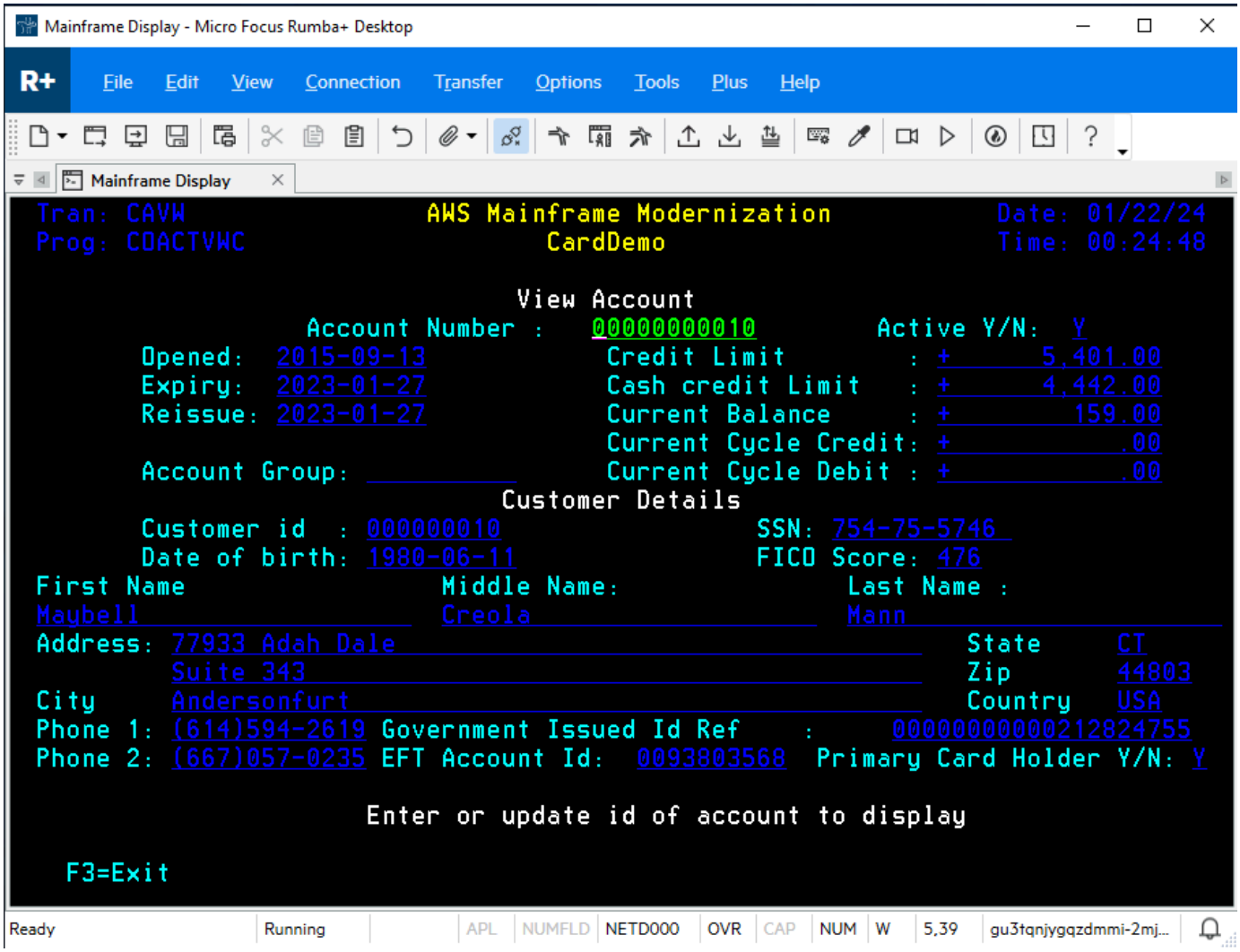


11. 成功登入後，您可以瀏覽 CardDemo 應用程式。
12. 在帳戶檢視中輸入 `01` 。



13. 輸入 00000000010 以取得帳號，然後按鍵盤上的 Enter 鍵。

> **Note**
>
> 其他有效的帳戶為 0000000011和 00000000020。



14. 按 F3 退出選單，然後按 F3 退出交易。

清除資源

如果您不再需要為本教學課程建立的資源，請將其刪除以避免額外費用。若要這樣做，請完成下列步驟：

- 如有必要，請停止應用程式。
- 刪除 應用程式。如需詳細資訊，請參閱刪除 AWS Mainframe Modernization 應用程式。
- 刪除執行時間環境。如需詳細資訊，請參閱刪除 AWS 大型主機現代化執行時間環境。
- 刪除您為此教學課程建立的 Amazon S3 儲存貯體。如需詳細資訊，請參閱《Amazon S3 使用者指南》中的刪除儲存貯體。

- 刪除您為此教學課程建立的 AWS Secrets Manager 秘密。如需詳細資訊，請參閱刪除秘密。
- 刪除您為此教學課程建立的 KMS 金鑰。如需詳細資訊，請參閱刪除 AWS KMS 金鑰。
- 刪除您為此教學課程建立的 Amazon RDS 資料庫。如需詳細資訊，請參閱《Amazon RDS 使用者指南》中的刪除 EC2 執行個體和資料庫執行個體。
- 如果您已新增連接埠 6000 的安全群組規則，請刪除規則。

後續步驟

若要了解如何為現代化應用程式設定開發環境，請參閱教學課程：設定 AppStream 2.0 以搭配 Rocket Enterprise Analyzer 和 Rocket Enterprise Developer 使用。

AWS Mainframe Modernization 元件生命週期

AWS Mainframe Modernization 的每個元件都會經歷版本升級和開發生命週期。您可以使用此頁面做為概觀，以了解這些元件、其版本升級計劃，以及 AWS Mainframe Modernization 如何傳達這些元件或其版本的發行或棄用。

元件生命週期概觀

AWS Mainframe Modernization lifecycle 描述了在整個生命週期中發佈和支援 AWS Mainframe Modernization 服務元件的方法和時間表。提供可預測且一致的生命週期可協助您規劃、測試和部署較新的版本。

所有 AWS提供的 AWS Mainframe Modernization 元件都受益於 提供的產品支援， 支援 從發行到每個元件的發行行事曆資料表淘汰為止。您可以在 Compare 支援 Plans 中進一步了解 支援 範圍和活動。在積極的現代化專案期間，我們通常會鼓勵專業服務交付團隊根據工作陳述式先提供客戶支援。

AWS Mainframe Modernization 發行了一些元件，其版本源自於供應商，這些供應商本身 AWS 、特定 AWS 合作夥伴或社群。對於每個 AWS 大型主機現代化元件，版本都有主要版本編號和次要版本編號。每個元件都有自己的主要和次要版本編號。

對於版本控制的元件，我們具有以下意圖：

- 定期或根據客戶需求發行 AWS Mainframe Modernization 元件較新版本。 如果需要元件的較新版本，但尚未在 AWS Mainframe Modernization 服務中提供，您可以透過 支援 產品功能請求 (PFR) 提出明確請求。
- 若要讓 AWS Mainframe Modernization 元件特定版本的終止支援和淘汰日期與元件供應商的終止支援日期一致。
- 在元件的主要版本淘汰前大約一年通知客戶。

雖然我們努力符合這些準則，但在某些情況下，我們可能會以較短的通知時間範圍更快淘汰特定版本。例如，我們可能會以較短的通知時間範圍，立即淘汰具有安全問題的版本。當次要版本在較新的次要版本中解決了重大錯誤或安全問題時，我們也可能會提早淘汰次要版本。萬一發生這種情況，我們會通知客戶，並傳達計劃和淘汰時間表。根據情況，特定情況可能會規定不同的時間表。

> **Note**
>
> 元件的重大更新可能隨時可用。例如，基於安全考量或為生產環境提供修正，可能會立即提供新版本。對於透過 提出的請求 支援，支援計劃會指定程序、嚴重性和回應時間。

當元件版本淘汰時， AWS Mainframe Modernization 不會將這些版本分發給客戶以進行新的部署。因此， 也不支援這些版本 支援。執行現有元件部署超過其版本淘汰日期的客戶，應了解這麼做的風險。AWS 不負責提供已淘汰元件版本的安全性更新、技術支援或熱修復。此外，我們不會自動移除存取或刪除您環境的資源。我們強烈建議您每 3 個月檢查新版本，並將所有 AWS Mainframe Modernization 元件升級至最新的支援版本。

版本升級

AWS Mainframe Modernization 提供每個支援元件的較新版本，讓您隨時掌握 up-to-date 最新的維護更新和功能。較新的版本可以包含錯誤修正、安全性增強功能，以及元件的其他改善。我們建議您定期升級，以受益於安全性修正、錯誤修正和功能增強。當 AWS Mainframe Modernization 發行新版本時，您可以選擇升級現有部署的方式和時間。升級有兩種：主要版本升級和次要版本升級。一般而言，主要引擎版本升級可能帶來與現有應用程式不相容的變更。在這種情況下，主要版本升級可能需要重大的應用程式變更。相反地，次要版本升級包括大部分與現有應用程式回溯相容的變更。次要版本升級可能只需要稍微變更到不需要變更。

您應該在執行元件的版本升級之前執行非迴歸測試。最佳實務是使用 DevOps 測試和部署管道。DevOps 測試管道可以在現代化專案期間建置，而且應該維護以在執行元件升級和應用程式程式碼變更時自動化應用程式測試。您也可以在升級期間使用藍/綠部署或 Canary 部署。您可以在 AWS Well-Architected 可靠性支柱進一步了解這類部署和變更管理。

AWS Mainframe Modernization Refactor with AWS Blu Age 版本概觀

使用 AWS Blu Age 執行時間時，版本會遵循`Major.Minor.Patch`模式。例如，對於 AWS Blu Age 執行時間版本 `4.1.0`，主要版本為 4，次要版本為 1，修補程式版本為 0。

我們希望在執行時間或其相依性有影響變更時，發佈新的 AWS Blu Age 執行期主要版本。除非出現一些常見漏洞與暴露 (CVEs)，否則至少支援 AWS 12 個月。支援涵蓋了執行時間功能中的錯誤，如我們的文件所述。在執行時間相依性 (Spring、Java、Tomcat 等） 中發生嚴重和高 CVEs 的情況下，除非

另有明確說明，否則高 CVEs 的主要版本支援持續時間會縮短為 6 個月，而關鍵 CVEs 則是從新執行時間版本發行日期起 3 個月，以修正 CVE。

我們打算每月發行新的 AWS Blu Age 次要版本。客戶應定期升級版本，以取得最新的安全性修正、錯誤修正和功能增強功能。尚未在生產環境中運作的作用中專案，必須在可用時立即採用最新的執行時間版本。

針對引發問題的特定主要版本，在最新的次要版本中提供新的修正。如果您需要新的修正，您需要升級至新的次要版本，才能套用這些修正。

支援版本的修補版本僅用於解決先前支援的次要版本中不存在的關鍵執行期瑕疵。

Alpha 發行前版本是短期版本，可在交付專案期間快速迭代。修正 alpha 發行前版本中識別的問題會在較新的次要版本中提供，因為 Alpha 發行前版本不會交付修補程式。

您可以在 中找到每個執行時間版本的發行日期和詳細資訊the section called “AWS Blu Age 版本備註”。

安全性掃描由 Amazon Inspector 執行。

了解 中的受管應用程式 AWS Mainframe Modernization

如果您是新手，可 AWS Mainframe Modernization 查看下列主題以開始使用：

- 什麼是 AWS 大型主機現代化？
- 設定 AWS 大型主機現代化
- 教學課程：為 AWS 藍年期設定受管執行時間
- 教學課程：設定 Rocket Software 的受管執行期（先前稱為 Micro Focus)

中的應用程式 AWS Mainframe Modernization 包含遷移的大型主機工作負載。應用程式類似於大型主機上的工作負載，並與執行期環境相關聯。您可以將批次檔案和資料集新增至應用程式，並在應用程式執行時監控應用程式。您可以為遷移的每個工作負載建立 AWS Mainframe Modernization 應用程式。建立 AWS Mainframe Modernization 應用程式時，您可以指定應用程式在建立時執行的引擎。如果您使用的是自動重構模式，請選擇 AWS 藍年，如果您使用的是複寫模式，請選擇火箭軟體（先前稱為 Micro Focus)。

主題

- 為遷移的應用程式建立 AWS 資源
- 建立 AWS Mainframe Modernization 應用程式
- 部署 AWS Mainframe Modernization 應用程式
- 更新 AWS Mainframe Modernization 應用程式
- 刪除 AWS Mainframe Modernization 應用程式
- 提交 AWS Mainframe Modernization 應用程式的批次任務
- 取消 AWS Mainframe Modernization 應用程式的批次任務
- 匯入 AWS Mainframe Modernization 應用程式的資料集
- 匯出應用程式的資料集 AWS Mainframe Modernization
- 管理應用程式的交易 AWS Mainframe Modernization
- 設定 Rocket Software（先前稱為 Micro Focus) 受管應用程式
- 設定 AWS Blu Age 受管應用程式
- AWS Mainframe Modernization 應用程式定義參考
- AWS Mainframe Modernization 資料集定義參考

為遷移的應用程式建立 AWS 資源

若要在 中執行遷移的應用程式 AWS，您必須使用其他 建立一些 AWS 資源 AWS 服務。您必須建立的資源包括下列項目：

- 存放應用程式碼、組態、資料檔案和其他必要成品的 S3 儲存貯體。
- Amazon RDS 或 Amazon Aurora 資料庫，用於存放應用程式所需的資料。
- AWS KMS key，這是 建立和存放秘密的必要 AWS Secrets Manager 項目。
- 用來保存資料庫登入資料的 Secrets Manager 秘密。

> **Note**
>
> 每個遷移的應用程式都需要自己的一組這些資源。這是最小集合。您的應用程式可能還需要其他資源，例如 Amazon Cognito 秘密或 MQ 佇列。

所需的許可

請確定您有下列許可：

- `s3:CreateBucket`, `s3:PutObject`
- `rds:CreateDBInstance`
- `kms:CreateKey`
- `secretsmanager:CreateSecret`

Amazon S3 儲存貯體

重構和修改的應用程式都需要您設定的 Amazon S3 儲存貯體，如下所示：

```
bucket-name/root-folder-name/application-name
```

bucket-name

Amazon S3 命名限制範圍內的任何名稱。建議您將 AWS 區域 名稱納入儲存貯體名稱的一部分。請確定您在計劃部署遷移應用程式的相同區域中建立儲存貯體。

root-folder-name

滿足應用程式定義中限制所需的名稱，您建立此定義做為 AWS Mainframe Modernization 應用程式的一部分。您可以使用 `root-folder-name`來區分應用程式的不同版本，例如 V1 和 V2。

application-name

遷移應用程式的名稱，例如 PlanetsDemo 或 BankDemo。

資料庫

重構和修改的應用程式都可能需要資料庫。您必須根據每個執行時間引擎的特定需求來建立、設定和管理資料庫。 AWS Mainframe Modernization 支援在此資料庫上傳輸中的加密。如果您在資料庫上啟用 SSL，請確定您在資料庫秘密`sslMode`中指定 以及資料庫的連線詳細資訊。如需詳細資訊，請參閱AWS Secrets Manager 秘密。

如果您使用 AWS Blu Age 重構模式，且需要 BluSam 資料庫，則 AWS Blu Age 執行期引擎會預期 Amazon Aurora PostgreSQL 資料庫，您必須建立、設定和管理該資料庫。BluSam 資料庫是選用的。只有當您的應用程式需要時，才建立此資料庫。若要建立資料庫，請遵循《Amazon Aurora 使用者指南》中的建立 Amazon Aurora 資料庫叢集中的步驟。

如果您使用的是 Rocket Software 轉換模式，您可以建立 Amazon RDS 或 Amazon Aurora PostgreSQL 資料庫。若要建立資料庫，請遵循《Amazon RDS 使用者指南》中的建立 Amazon RDS 資料庫執行個體或《Amazon Aurora 使用者指南》中的建立 Amazon Aurora 資料庫叢集中的步驟。

對於這兩種執行期引擎，您必須 AWS Secrets Manager 使用 將資料庫登入資料存放在 中 AWS KMS key ，以加密它們。

AWS Key Management Service 金鑰

您必須安全地存放應用程式資料庫的登入資料 AWS Secrets Manager。若要在 Secrets Manager 中建立秘密，您必須建立 AWS KMS key。若要建立 KMS 金鑰，請遵循《 AWS Key Management Service 開發人員指南》中的建立金鑰中的步驟。

建立金鑰後，您必須更新金鑰政策以授予 AWS Mainframe Modernization 解密許可。新增下列政策陳述式：

```
{
   "Effect" : "Allow",
   "Principal" : {
```

```
   "Service" : "m2.amazonaws.com"
   },
   "Action" : "kms:Decrypt",
   "Resource" : "*"
   }
```

AWS Secrets Manager 秘密

您必須安全地存放應用程式資料庫的登入資料 AWS Secrets Manager。若要建立秘密，請遵循AWS Secrets Manager 《 使用者指南》中的建立資料庫秘密中的步驟。

AWS Mainframe Modernization 支援在此資料庫上傳輸中的加密。如果您在資料庫上啟用 SSL，請確定您在資料庫秘密`sslMode`中指定 以及資料庫的連線詳細資訊。您可以為 指定下列其中一個值`sslMode`：`verify-full`、 `verify-ca`或 `disable`。

在金鑰建立過程中，選擇資源許可 - 選用，然後選擇編輯許可。在政策編輯器中，新增資源型政策，例如下列，以擷取加密欄位的內容。

```
{
   "Effect" : "Allow",
   "Principal" : {
   "Service" : "m2.amazonaws.com"
   },
   "Action" : "secretsmanager:GetSecretValue",
   "Resource" : "*"
   }
```

建立 AWS Mainframe Modernization 應用程式

使用 AWS Mainframe Modernization 主控台來建立 AWS Mainframe Modernization 應用程式。建立應用程式可讓您使用遷移的大型主機工作負載執行任務。

這些說明假設您已完成設定 AWS 大型主機現代化中的步驟。

建立應用程式

建立應用程式

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。

2. 在 AWS 區域 選擇器中，選擇您要建立應用程式的 區域。
3. 在 Applications (應用程式) 頁面上，選擇 Create application (建立應用程式)。
4. 在指定基本資訊頁面上的名稱和描述區段中，輸入應用程式的名稱。
5. （選用）在應用程式描述欄位中，輸入應用程式的描述。此描述可協助您和其他使用者識別應用程式的目的。
6. 在引擎類型區段中，選擇自動重構的模糊年齡，或選擇 Micro Focus (Rocket) 進行複寫。
7. 在 KMS 金鑰區段中，如果您想要使用客戶受管 AWS KMS 金鑰，請選擇自訂加密設定。如需詳細資訊，請參閱AWS Mainframe Modernization 服務靜態資料加密。

> **Note**
>
> 根據預設， 會使用 AWS Mainframe Modernization 擁有和管理的 AWS KMS 金鑰 AWS Mainframe Modernization 來加密您的資料。不過，您可以選擇使用客戶受管 AWS KMS 金鑰。

8. （選用）依名稱或 Amazon Resource Name (ARN) 選擇 AWS KMS 金鑰，或選擇建立 AWS KMS 金鑰以前往 AWS KMS 主控台並建立新的 AWS KMS 金鑰。
9. （選用）在標籤區段中，選擇新增標籤，將一或多個應用程式標籤新增至您的應用程式。應用程式標籤是自訂屬性標籤，可協助您組織和管理 AWS 資源）。
10. 選擇下一步。
11. 在資源和組態區段中，使用內嵌編輯器輸入應用程式定義。或者，選擇在 Amazon S3 儲存貯體中使用應用程式定義 JSON 檔案，並提供您要使用的應用程式定義位置。如需詳細資訊，請參閱 AWS Blu Age 應用程式定義範例 或 Rocket Software（先前稱為 Micro Focus) 應用程式定義 。
12. 選擇下一步。
13. 在檢閱和建立頁面上，檢閱您輸入的資訊，然後選擇建立應用程式。

部署 AWS Mainframe Modernization 應用程式

使用 AWS Mainframe Modernization 主控台來部署 AWS Mainframe Modernization 應用程式。您需要在執行時間環境中部署應用程式，才能執行任務。

這些說明假設您已完成設定 AWS 大型主機現代化中的步驟。

部署應用程式

若要執行 AWS Mainframe Modernization 應用程式，您必須先將其部署到執行時間環境。應用程式可以有多個版本。應用程式的每個版本都有自己的應用程式定義。若要部署應用程式，您必須指定要部署的版本。

您一次只能部署一個版本的指定應用程式。如果您部署應用程式版本，則決定改為部署不同的版本，則必須先停止執行中的應用程式。

部署應用程式

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 在 AWS 區域 選擇器中，選擇您要建立應用程式的 區域。
3. 在應用程式頁面上，選擇您要部署的應用程式。
4. 選擇部署應用程式。
5. 在可用版本區段中，選擇您要部署的版本。
6. 在環境區段中，選擇您希望應用程式執行的執行時間環境。
7. 選擇部署。

部署不同版本的已部署應用程式

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 在 AWS 區域 選擇器中，選擇您要建立應用程式的 區域。
3. 在應用程式頁面上，選擇您要部署的應用程式。
4. 從動作功能表中，選擇停止應用程式。
5. 應用程式停止後，選擇部署應用程式。
6. 在可用版本區段中，選擇您要部署的版本。在環境區段中，已預先選取應用程式已在其中部署的環境。
7. 選擇部署。

更新 AWS Mainframe Modernization 應用程式

使用 AWS Mainframe Modernization 主控台更新 AWS Mainframe Modernization 應用程式。更新應用程式會建立新的應用程式版本。

這些說明假設您已完成設定 AWS 大型主機現代化中的步驟。

更新應用程式

AWS Mainframe Modernization 應用程式可以有多個版本，每個版本都有自己的應用程式定義。若要更新應用程式，請提供新的應用程式定義。這會建立新的應用程式版本。

更新應用程式

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 在 AWS 區域 選擇器中，選擇您要更新的應用程式建立所在的區域。
3. 在應用程式頁面上，選擇您要更新的應用程式。
4. 在應用程式詳細資訊頁面的目前定義區段中，選擇編輯以更新目前的應用程式定義。
5. 在更新應用程式頁面上，使用內嵌編輯器來更新目前的應用程式定義。

   或者，選擇在 Amazon S3 儲存貯體中使用應用程式定義 JSON 檔案，並提供您要使用的應用程式定義位置。如需詳細資訊，請參閱 AWS Blu Age 應用程式定義範例 或 Rocket Software （ 先前稱為 Micro Focus) 應用程式定義 。
6. 完成更新應用程式定義後，請選擇更新。

> **Note**
>
> 更新應用程式後，您必須再次部署應用程式。如需詳細資訊，請參閱部署 AWS Mainframe Modernization 應用程式。

刪除 AWS Mainframe Modernization 應用程式

您可以使用 AWS Mainframe Modernization 主控台從環境刪除 AWS Mainframe Modernization 應用程式。

這些說明假設您已完成設定 AWS 大型主機現代化中的步驟。

刪除應用程式

如果您需要刪除 AWS Mainframe Modernization 應用程式，且應用程式正在執行中，請務必先將其停止。您可以在應用程式頁面上查看應用程式狀態。

如欲刪除應用程式

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https://console.aws.amazon.com/m2/：//。
2. 在 AWS 區域 選擇器中，選擇您要從環境中刪除的應用程式建立所在的區域。
3. 在應用程式頁面上，選擇您要從環境中刪除的應用程式，然後選擇動作。
4. （選用）如果應用程式的狀態為 Running，請選擇停止應用程式。
5. 選擇從環境刪除。

刪除程序會立即開始。

提交 AWS Mainframe Modernization 應用程式的批次任務

在 AWS Mainframe Modernization 中，您可以為您的應用程式提交批次任務。您可以提交或取消批次任務，並檢閱批次任務執行的詳細資訊。每次您提交批次任務時， AWS Mainframe Modernization 都會建立個別的批次任務執行。您可以監控此任務執行。您可以依名稱搜尋批次任務，並將 JCL 或指令碼檔案提供給批次任務。

> **Important**
>
> 如果您取消批次任務，這不會刪除任務。它會取消批次任務的特定執行。批次任務記錄仍然可供您在批次任務執行的詳細資訊中檢視。

如果您的批次任務需要存取一或多個資料集，請使用 AWS Mainframe Modernization 主控台匯入資料集。如需詳細資訊，請參閱匯入 AWS Mainframe Modernization 應用程式的資料集。

這些指示假設您已完成 設定 AWS 大型主機現代化和 中的步驟建立 AWS Mainframe Modernization 應用程式。

主題

- 提交批次任務
- 重新啟動批次任務

提交批次任務

提交批次任務

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 在 AWS 區域 選擇器中，選擇您要為其提交批次任務的應用程式建立所在的區域。
3. 在應用程式頁面上，選擇要提交批次任務的應用程式。

> **Note**
>
> 您必須先成功部署應用程式，才能將批次任務提交至應用程式。

4. 在應用程式詳細資訊頁面上，選擇批次任務。
5. 選擇 Submit job (提交任務)。
6. 在選取指令碼區段中，選擇指令碼。您可以依名稱搜尋所需的指令碼。
7. 選擇 Submit job (提交任務)。

重新啟動批次任務

重新啟動批次任務

> **Important**
>
> 批次工作重新啟動可在下列引擎版本上使用：
>
> - Micro Focus (Rocket) 環境引擎版本 8.0.6 或更新版本。您也需要將 EFS 或 FSx 檔案系統連接到您的環境。
> - AWS Blu Age 環境引擎 4.3.0 版或更新版本。如果 EFS 或 FSx 檔案系統是 HA 環境，您也需要連接它。

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https://console.aws.amazon.com/m2/：//。

2. 在 AWS 區域 選擇器中，選擇應用程式和批次任務建立所在的區域。
3. 在應用程式頁面上，選擇您要重新啟動批次任務的應用程式。
4. 在應用程式詳細資訊頁面上，選擇批次任務。
5. 從產生的清單中選取要重新啟動的批次任務。導覽至動作功能表，然後選擇重新啟動任務。
6. 指定您要如何重新啟動批次任務。您可以針對 Micro Focus (Rocket) 環境引擎和 AWS Blu Age 環境引擎執行下列動作：

   - 對於 Micro Focus (Rocket) 環境引擎，您可以選擇從頭開始重新啟動，或使用步驟或程序重新啟動。
     - 從開始選項重新啟動可讓您從頭開始重新啟動批次任務的所有步驟。
     - 使用步驟或程序選項重新啟動 可讓您選擇要重新啟動的特定步驟或程序步驟，也可以選擇您要結束的步驟或程序步驟。

   > **Note**
   >
   > 結束步驟或 procstep 必須大於或等於開始步驟或 procstep 號碼。

   - 對於 AWS Blu Age 環境引擎，您可以從先前失敗的 JCL/PROC 步驟重新啟動批次任務的最新執行，或透過略過先前成功的步驟來執行延遲重新啟動。
     - 您可以選擇要重新啟動的特定步驟名稱。
     - 或者，您可以使用略過步驟略過選取的步驟，並從工作流程中的下一個步驟重新啟動。
7. 選擇 Submit job (提交任務)。

取消 AWS Mainframe Modernization 應用程式的批次任務

在 中 AWS Mainframe Modernization ，您可以取消應用程式的批次任務。您可以檢閱批次任務執行的詳細資訊。每次您提交批次任務時， AWS Mainframe Modernization 都會建立個別的批次任務執行。您可以監控此任務執行。您可以依名稱搜尋批次任務，並將 JCL 或指令碼檔案提供給批次任務。

> **Important**
>
> 如果您取消批次任務，這不會刪除任務。它會取消批次任務的特定執行。批次任務記錄仍然可供您在批次任務執行的詳細資訊中檢視。

取消批次任務

當您取消批次任務時，它不會刪除批次任務，而是該批次任務的任務執行。您仍然可以檢視批次任務的詳細資訊。

取消批次任務

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https://console.aws.amazon.com/m2/：//。
2. 在 AWS 區域 選取器中，選擇批次任務的應用程式所在的區域。
3. 從批次任務清單中，尋找並選取您要取消的批次任務。
4. 選擇動作，然後選擇取消任務。
5. 選擇取消批次任務。

這將取消您已排程執行的任何批次任務。

匯入 AWS Mainframe Modernization 應用程式的資料集

透過 AWS Mainframe Modernization ，您可以匯入資料集以搭配您的應用程式使用。您可以指定要匯入 Amazon S3 儲存貯體中存放的 JSON 檔案中的資料集，也可以分別指定資料集組態值。匯入資料集之後，您可以檢閱匯入任務的詳細資訊，以確認您想要的資料集已匯入。應用程式的所有目錄化資料集都會在 主控台中一起列出。

使用 AWS Mainframe Modernization 主控台匯入 AWS Mainframe Modernization 應用程式的資料集。

這些指示假設您已完成 設定 AWS 大型主機現代化和 中的步驟建立 AWS Mainframe Modernization 應用程式。

匯入資料集

匯入資料集

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 在 AWS 區域 選擇器中，選擇您要為其匯入資料集的應用程式建立所在的區域。
3. 在應用程式頁面上，選擇要匯入資料集的應用程式。
4. 在應用程式詳細資訊頁面上，選擇資料集。

5. 選擇匯入。
6. 執行以下任意一項：
   - 選擇在 Amazon S3 儲存貯體中使用資料集組態 JSON 檔案，並提供資料集組態的位置。
   - 選擇使用引導式組態分別指定資料集組態值。如需特定定義詳細資訊the section called “資料集定義參考”，請參閱 。

     輸入每個資料集組態值的名稱、資料集組織 (VSAM、GDG、PO、PS)、位置和外部 Amazon S3 位置，以及參數設定。在引導式組態中，您也可以選擇產生 JSON 以從輸入中檢閱 JSON 組態。
7. 選擇提交。

匯出應用程式的資料集 AWS Mainframe Modernization

透過 AWS Mainframe Modernization ，您可以匯出資料集以搭配您的應用程式使用。您可以指定要匯出到存放在 Amazon S3 儲存貯體中 JSON 檔案中的資料集，也可以分別指定資料集組態值。匯出資料集之後，您可以檢閱匯出任務的詳細資訊，以確認您想要的資料集已匯出。

使用 AWS Mainframe Modernization 主控台匯出 AWS Mainframe Modernization 應用程式的資料集。

這些指示假設您已完成 設定 AWS 大型主機現代化和 中的步驟建立 AWS Mainframe Modernization 應用程式。

匯出資料集

匯出資料集

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 在 AWS 區域 選擇器中，選擇您要為其匯入資料集的應用程式建立所在的區域。
3. 在應用程式頁面上，選擇要匯出資料集的應用程式。
4. 在應用程式詳細資訊頁面上，選擇資料集。
5. 選擇 Export (匯出)。
6. 執行以下任意一項：
   - 選擇在 Amazon S3 儲存貯體中使用資料集組態 JSON 檔案，並提供資料集組態的位置。
   - 選擇使用引導式組態分別指定資料集組態值。如需詳細資訊，請參閱the section called “資料集定義參考”。

輸入每個資料集組態值的資料集名稱、外部 Amazon S3 位置和參數設定。在引導式組態中，您也可以選擇產生 JSON 以從輸入中檢閱 JSON 組態。

7. 選擇提交。

管理應用程式的交易 AWS Mainframe Modernization

透過 ，AWS Mainframe Modernization 您可以依請求執行應用程式，就像提交請求以使用相同檔案和程式執行相同應用程式的其他許多使用者一樣。單一交易由一或多個應用程式組成，這些應用程式會執行所需的處理。

這些指示假設您已完成 設定 AWS 大型主機現代化和 中的步驟建立 AWS Mainframe Modernization 應用程式。

管理應用程式的交易

管理應用程式的交易

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 在 AWS 區域 選擇器中，選擇您要執行的應用程式建立所在的區域。
3. 在應用程式頁面上，選擇您要管理交易的應用程式。
4. 在交易索引標籤的交易資源下，從下拉式清單中選擇您希望資源顯示的方式。您可以根據交易資源、群組、清單或 SITs顯示資源。

   - 交易資源可讓您根據檔案定義、交易定義、程式定義或暫時性資料佇列定義來選擇資源類型。

     > **Note**
     >
     > AWS Mainframe Modernization 服務支援其他資源類型來管理應用程式的交易，並且可以在 主控台中存取。

   - 群組是交易資源的集合。您可以選擇要與交易資源建立關聯的群組。
   - 清單是群組的排序集合。您可以在清單檢視中查看所有交易資源和群組。啟動清單會決定初始化伺服器時要載入的資源。
     - 使用 AWS Blu Age Refactor Engine，您可以指定要在啟動時包含的清單。清單數量沒有限制。

- 使用 Rocket Software replatform 引擎，您可以在一個 SIT 中指定最多四個清單。
- SIT （系統初始化表） 會顯示所有可用的交易組態。您可以根據屬性 （名稱、描述和啟動清單） 尋找 SITs。您也可以選擇要與所選 SIT 建立關聯的清單。

> **Note**
>
> SITs僅適用於 Rocket Software Replatform 引擎。

5. 選擇交易資源以顯示所有資源資訊。您也可以檢視與交易資源相關聯的所有屬性。

設定 Rocket Software （先前稱為 Micro Focus) 受管應用程式

您可以使用 Rocket Software 執行期引擎來設定應用程式，以自訂其他屬性，包括整合。

主題

- Rocket 軟體支援的第三方整合
  - 印表機

Rocket 軟體支援的第三方整合

若要使用第三方整合，您的 AWS Mainframe Modernization 受管環境必須使用支援此類型組態的 Rocket Software 引擎版本。支援具有後綴 R 的引擎版本 （例如 9.0.9.R 版）。這表示，引擎版本 9.0.9.R 包含對當事方整合的用戶端安裝支援，但 9.0.9 不支援。

印表機

印表機資源是透過 Rocket Software 應用程式定義進行設定，如 the section called “印表機 - 選用”一節中所述。

印表機定義可以定義印表機的自訂或服務提供的結束模組。一些可能的結束模組組態範例如下：

1. 載入服務提供的二進位檔的範例。

```
...
{
    "name": "p1",
    "classes": [
        "AB"
    ],
```

```
    "description": "Using service managed LRS Queue exit module",
    "exit-module": {
        "name": "lrsprte6"
    }
},
...
```

2. 從 S3 提供二進位檔的範例。

```
"exit-module": {
        "name": "s3Exit",
        "module": "${s3-source}/3pa/s3Exit.so"
}
```

3. 從 EFS 提供二進位檔的範例。

> **Note**
>
> 若要使用 EFS 掛載，必須在環境建立期間連接它，以及一些要設定的其他值，例如 `program-path`。

```
...
"batch-settings": {
    "jes-printers": [
        {
            "name": "p3",
            "classes": [
                "EF"
            ],
            "description": "Using binary from customer provided exit module on EFS
 Mount",
            "exit-module": {
                "name": "efsExit"
            }
        }
    ],
    "program-path": "$EFS_MOUNT/path/to/directory/containing/binaries/"
},
"runtime-settings": {
    "environment-variables": {
        "EFS_MOUNT": "/m2/mount/efs"
```

```
    }
}
...
```

LRS 佇列 - 選用

若要使用 LRS 佇列，您必須使用支援第三方成品 （即以 結尾的引擎.R) 的 Rocket 軟體引擎。除了設定具有指向 的結束模組lrsprte6作為結束模組項目名稱的印表機之外，LRS 佇列還需要額外的環境變數，如 Rocket Software 應用程式定義中預先存在的「執行時間設定」區塊所定義。

LRSQ_ADDRESS

（必要）指定 LRSQ 列印結束模組的 LRS 伺服器地址。

LRS 印表機 - 設定 LRS 印表機需要 the section called “印表機 - 選用”區段中指定的 jes 印表機定義。

此外，必須在應用程式定義中將 LRSQ_ADDRESS 指定為 runtime-settings 欄位的一部分。

```
"runtime-settings": {
    "environment-variables": {
        "LRSQ_ADDRESS": "<lrsq-address>"
    }
}
```

設定 AWS Blu Age 受管應用程式

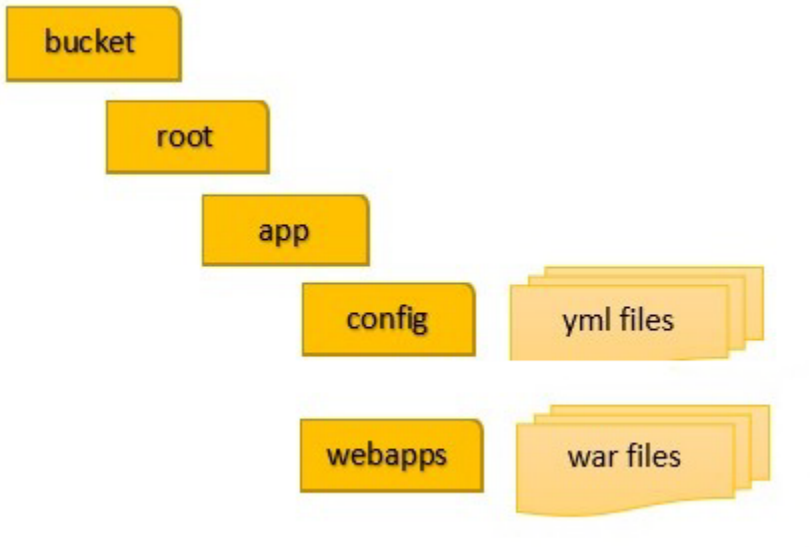
您可以設定應用程式，以包含對舊版公用程式的存取。您也可以自訂其他屬性。為了了解您可以設定的內容和位置，請參閱 the section called “AWS Blu Age 受管應用程式的結構”一節以了解 AWS Blu Age 現代化應用程式的整體結構。

主題

- AWS Blu Age 受管應用程式的結構
- 設定受管應用程式的公用程式存取權
- 使用 AWS Blu Age 引擎新增受管應用程式的組態屬性

AWS Blu Age 受管應用程式的結構

如果您使用 AWS Blu Age 重構模式， AWS Blu Age 執行期引擎預期 S3 儲存貯體中 application-name 資料夾內的結構如下：



config

包含專案的 YAML 檔案。這些是應用程式特有的 YAML 檔案，通常命名如下`application-planetsdemo.yaml`，而不是 AWS 大型主機現代化為您自動提供和設定`application-main.yaml`的檔案。

Web 應用程式

包含您應用程式`war`的檔案。這些檔案是現代化程序的輸出。

應用程式也可以有下列選用資料夾：

jics/sql

包含指令碼`initJics.sql`，可為您的應用程式初始化 JICS 資料庫。

指令碼

包含應用程式指令碼，您也可以直接在`war`檔案內提供。

sql

包含應用程式 SQL 檔案，您也可以直接在`war`檔案內提供。

lnk

包含應用程式 LNK 檔案，您也可以直接在`war`檔案內提供。

額外

包含可為現代化應用程式提供額外的功能的 jar。

管理應用程式的 Java 選項

若要管理應用程式的一些 Java 選項，請將名為 `tomcat.properties`的屬性檔案新增至 `application-name` 資料夾。此檔案可以有三個屬性： `xms`，指定 Java 記憶體耗用下限， `xmx`，指定 Java 記憶體耗用上限，以及 `dnscachettl`，其管理 dns 解析度的快取持續時間。以下是有效`tomcat.properties`檔案的內容範例。

```
xms=512M
xmx=1G
dnscachettl=5
```

您為前兩個屬性指定的值可以是下列任何單位：

- 位元組：請勿指定單位。
- Kilobytes：將 K 附加至 值。
- MB：將 M 附加至 值。
- Gigabytes：將 G 附加至 值。

第三個屬性的值代表以秒為單位的快取持續時間，值可以是 -1 （永遠快取），也可以是 0 （永遠不會快取） 到 999。在受管應用程式部署的內容中，預設值為 -1。

設定受管應用程式的公用程式存取權

當您重構具有 AWS Blu Age 的大型主機應用程式時，如果您的應用程式依賴於 IDCAMS、INFUTILB、SORT 等各種舊版平台公用程式，您可能需要提供支援。 AWS Blu Age Refactoring 提供此存取權，搭配現代化應用程式部署。此 Web 應用程式需要`application-utility-pgm.yml`您必須提供的組態檔案 。如果您未提供此組態檔案，Web 應用程式將無法與您的應用程式一起部署，也無法使用。

主題

- 組態屬性

本主題說明您可以在application-utility-pgm.yml組態檔案中指定的所有可能屬性及其預設值。主題說明必要和選用屬性。下列範例是完整的組態檔案。它會依我們建議的順序列出屬性。您可以使用此範例做為您自己的組態檔案的起點。

```
# If the datasource support mode is not static-xa, spring JTA transactions
 autoconfiguration must be disabled
 spring.jta.enabled: false
 logging.config: 'classpath:logback-utility.xml'

 # Encoding
 encoding: cp1047

 # Encoding to be used by INFUTILB and DSNUTILB to generate and read SYSPUNCH files
 sysPunchEncoding: cp1047

 # Utility database access
 spring.aws.client.datasources.primary.secret: `arn:aws:secretsmanager:us-
west-2:111122223333:secret:business-FfmXLG`

 treatLargeNumberAsInteger: false

 # Zoned mode : valid values = EBCDIC_STRICT, EBCDIC_MODIFIED, AS400
 zonedMode: EBCDIC_STRICT

 jcl.type: mvs

 # Unload properties
 # For date/time: if use database configuration is enabled, formats are ignored
 # For nbi; use hexadecimal syntaxe to specify the byte value
 unload:
  sqlCodePointShift: 384
  nbi:
    whenNull: "6F"
    whenNotNull: "00"
  useDatabaseConfiguration: false
  format:
    date: MM/dd/yyyy
    time: HH.mm.ss
    timestamp: yyyy-MM-dd-HH.mm.ss.SSSSSS
  chunkSize:500
  fetchSize: 500
  varCharIsNull: false
  columnFiller: space
```

```
# Load properties
# Batch size for DSNUTILB Load Task
load:
 sqlCodePointShift: 384
 batchSize: 500
 format:
   localDate: dd.MM.yyyy|dd/MM/yyyy|yyyy-MM-dd
   dbDate: yyyy-MM-dd
   localTime: 'HH:mm:ss|HH.mm.ss'
   dbTime: 'HH:mm:ss'

table-mappings:
 TABLE_1_NAME : LEGACY_TABLE_1_NAME
 TABLE_2_NAME : LEGACY_TABLE_2_NAME
```

組態屬性

您可以在組態檔案中指定下列屬性。

spring.jta.enabled

（選用）控制是否啟用 JTA 支援。對於公用程式，我們建議您將此值設定為 `false`。

```
spring.jta.enabled : false
```

logging.config

（必要）指定專用記錄器組態檔案的路徑。我們建議您使用 名稱`logback-utility.xml`，並提供此檔案做為現代化應用程式的一部分。組織這些檔案的常見方法是將所有記錄器組態檔案放在相同位置，通常位於 子資料夾中，`/config/logback`其中 `/config` 是包含 YAML 組態檔案的資料夾。如需詳細資訊，請參閱 日誌傳回文件中的第 3 章：日誌傳回組態。

```
logging.config : classpath:logback-utility.xml
```

編碼

（必要）指定公用程式使用的字元集。在大多數情況下，當您從 z/OS 平台遷移時，此字元集是 EBCDIC 變體，並且應該符合針對現代化應用程式設定的字元集。如果未設定 ，則預設為 `ASCII`。

```
encoding : cp1047
```

sysPunchEncoding

（選用）指定 INFUTILB 和 DSNUTILB 用來產生和讀取 SYSPUNCH 檔案的字元集。如果您使用舊版平台中的 SYSPUNCH 檔案，則此值應為 EBCDIC 變體。如果未設定 ，則預設為 ASCII。

```
sysPunchEncoding : cp1047
```

資料來源組態

有些資料庫相關的公用程式，例如 LOAD 和 UNLOAD，需要透過資料來源存取目標資料庫。如同 AWS 大型主機現代化內的其他資料來源定義，此存取需要您使用 AWS Secrets Manager。指向 Secrets Manager 中適當秘密的屬性如下：

主要資料來源

這是主要商業應用程式資料庫。

spring.aws.client.datasources.primary.secret

（選用）在 Secrets Manager 中指定包含資料來源屬性的秘密。

```
spring.aws.client.datasources.primary.secret: datasource-secret-ARN
```

spring.aws.client.datasources.primary.dbname

（選用）如果資料庫名稱未直接在資料庫秘密中以 dbname 屬性提供，則指定目標資料庫名稱。

```
spring.aws.client.datasources.primary.dbname: target-database-name
```

spring.aws.client.datasources.primary.type

（選用）指定要使用的連線集區實作的完整名稱。預設值為 com.zaxxer.hikari.HikariDataSource。

```
spring.aws.client.datasources.primary.type: target-datasource-type
```

如果主要資料來源的類型為 com.zaxxer.hikari.HikariDataSource，您可以指定其他屬性，如下所示：

spring.datasource.primary.【property_name】

（選用）您可以使用此格式來指定額外的屬性，以設定主要資料來源連線集區實作。

以下是類型 的主要資料來源範例com.zaxxer.hikari.HikariDataSource。

```
spring:
   datasource:
       primary:
         autoCommit: XXXX
         maximumPoolSize: XXXX
         keepaliveTime: XXXX
         minimumIdle: XXXX
         idleTimeout: XXXX
         connectionTimeout: XXXX
         maxLifetime: XXXX
```

其他公用程式資料來源

除了主要資料來源之外，您還可以提供其他公用程式資料來源。

spring.aws.client.utility.pgm.datasources.names

（選用）指定公用程式資料來源名稱的清單。

```
spring.aws.client.utility.pgm.datasources.names: dsname1, dsname2, dsname3
```

spring.aws.client.utility.pgm.datasources.【dsname】.secret

（選用）指定 SSM 中託管資料來源屬性的秘密 ARN。在 中指定的名稱清單中提供【dsname】spring.aws.client.utility.pgm.datasources.names。

```
spring.aws.client.utility.pgm.datasources.dsname1.secret: datasource-secret-ARN
```

spring.aws.client.utility.pgm.datasources.【dsname】.dbname

（選用）如果資料庫名稱不是使用 dbname 屬性直接在資料庫秘密中提供，則指定目標資料庫名稱。在 中指定的名稱清單中提供【dsname】spring.aws.client.utility.pgm.datasources.names。

```
spring.aws.client.utility.pgm.datasources.dsname1.dbname: target-database-name
```

spring.aws.client.utility.pgm.datasources.【dsname】.type

（選用）指定要使用的連線集區實作的完整名稱。預設值為 com.zaxxer.hikari.HikariDataSource。在 中指定的名稱清單中提供 【dsname】spring.aws.client.utility.pgm.datasources.names。

```
spring.aws.client.utility.pgm.datasources.dsname1.type: target-datasource-type
```

如果公用程式資料來源類型為 com.zaxxer.hikari.HikariDataSource，您可以提供其他屬性，如下所示：

spring.datasource.【dsname】.【property_name】

（選用）指定額外屬性的集合，以設定公用程式資料來源連線集區實作。在 中指定的名稱清單中提供 【dsname】spring.aws.client.utility.pgm.datasources.names。以下列格式指定屬性： property_name : value

以下是類型 的其他公用程式資料來源範例com.zaxxer.hikari.HikariDataSource：

```
spring:
   datasource:
       dsname1:
         connectionTimeout: XXXX
         maxLifetime: XXXX
       dsname2:
         connectionTimeout: XXXX
         maxLifetime: XXXX
       dsname3:
         connectionTimeout: XXXX
         maxLifetime: XXXX
```

treatLargeNumberAsInteger

（選用）與 Oracle 資料庫引擎詳細資訊和 DSNTEP2/DSNTEP4 公用程式使用量相關。如果您將此旗標設定為 true，則來自 Oracle 資料庫 (NUMBER (38，0)) 的大型數字會視為整數。預設：false

```
treatLargeNumberAsInteger : false
```

zonedMode

（選用）設定分區模式以編碼或解碼分區資料類型。此設定會影響符號數字的表示方式。有效值如下：

- EBCDIC_STRICT：預設。使用嚴格定義來處理標誌。根據字元集是 EBCDIC 還是 ASCII，符號位數表示會使用下列字元：
  - 對應至位元組 (Cn+Dn) 的 EBCDIC 字元，代表正負位數範圍 (+0 到 +9、 -0到 -9)。字元會顯示為 {、A 到 }、 IJ到 R
  - 對應至位元組 (3n+7n) 的 ASCII 字元，代表正負位數範圍 (+0 到 +9、 -0到 -9)。字元顯示為 0 到 9，p顯示為 y
- EBCDIC_MODIFIED：使用修改後的符號處理定義。對於 EBDIC 和 ASCII，相同的字元清單代表符號數字，也就是+0+9映射到 { + A 到 I 和-0-9映射到 } + J到 R。 \
- AS400：用於來自 iSeries (AS400) 平台的現代化舊版資產。

```
zonedMode:EBCDIC_STRICT
```

jcl.type

（選用）指出現代化 JCL 指令碼的舊版類型。如果叫用 JCL 的類型為 ，IDCAMS 公用程式會使用此設定來量身打造傳回碼vse。有效值如下：

- mvs (預設值)
- vse

```
jcl.type : mvs
```

資料庫卸載公用程式相關屬性

使用這些屬性來設定公用程式，將資料庫資料表卸載至資料集。下列所有屬性皆為選用。

此範例顯示所有可能的卸載屬性。

```
# Unload properties
 # For date/time: if use database configuration is enabled, formats are ignored
 # For nbi; use hexadecimal syntaxe to specify the byte value
 unload:
 sqlCodePointShift: 0
 nbi:
```

```
whenNull: "6F"
whenNotNull: "00"
useDatabaseConfiguration: false
format:
date: MM/dd/yyyy
time: HH.mm.ss
timestamp: yyyy-MM-dd-HH.mm.ss.SSSSSS
chunkSize: 0
fetchSize: 0
varCharIsNull: false
columnFiller: space
```

sqlCodePointShift

（選用）指定整數值，代表資料上使用的 SQL 程式碼點轉移。預設值為 0。這表示不會進行程式碼點轉移。將此設定與用於現代化應用程式的 SQL 程式碼點轉移參數對齊。使用程式碼點轉移時，此參數最常見的值為 384。

```
unload.sqlCodePointShift: 0
```

nbi

（選用）指定 null 指標位元組。這是新增至資料值右側的十六進位值（做為字串）。兩個可能的值如下：

- whenNull：當資料值為 null 時新增十六進位值。預設值為 6`。有時候FF會改用高值。

  ```
  unload.nbi.whenNull: "6F"
  ```

- whenNotNull：當資料值不是 null 但資料欄是 null 時，新增十六進位值。預設值為 00（低值）。

  ```
  unload.nbi.whenNotNull: "00"
  ```

useDatabaseConfiguration

（選用）指定日期和時間格式屬性。這用於處理 UNLOAD 查詢中的日期/時間物件。預設值為 false。

- 如果設定為 true， 會使用主組態檔案 () 中的 pgmDateFormatpgmTimeFormat、 和 pgmTimestampFormat 屬性application-main.yml。
- 如果設定為 false， 會使用下列日期和時間格式屬性：
  - unload.format.date：指定日期格式模式。預設值為 MM/dd/yyyy。

- unload.format.time：指定時間格式模式。預設值為 HH.mm.ss。
- unload.format.timestamp：指定時間戳記格式模式。預設值為 yyyy-MM-dd-HH.mm.ss.SSSSSS。

chunkSize

（選用）指定用於建立 SYSREC 資料集的資料區塊大小。這些資料集是具有平行操作的資料集卸載操作的目標。預設值為 0（無區塊）。

```
unload.chunkSize:0
```

fetchSize

（選用）指定資料擷取大小。值是使用資料區塊策略時，一次要擷取的記錄數目。預設：0。

```
unload.fetchSize:0
```

varCharIsNull

（選用）指定如何處理含空白內容的非 null varchar 資料欄。預設值為 false。

如果您將此值設定為 true，資料欄內容會被視為空字串以用於卸載，而不是單一空格字串。僅true針對 Oracle 資料庫引擎案例將此旗標設為 。

```
unload.varCharIsNull: false
```

columnFiller

（選用）指定用於填補 varchar 資料欄中卸載資料欄的值。可能的值為空格或低值。預設為空格。

```
unload.columnFiller: space
```

資料庫載入相關屬性

使用這些屬性來設定公用程式，將資料集記錄載入目標資料庫，例如 DSNUTILB。下列所有屬性皆為選用。

此範例顯示所有可能的負載屬性。

```
# Load properties
```

```
# Batch size for DSNUTILB Load Task
load:
sqlCodePointShift: 384
batchSize: 500
format:
localDate: dd.MM.yyyy|dd/MM/yyyy|yyyy-MM-dd
dbDate: yyyy-MM-dd
localTime: HH:mm:ss|HH.mm.ss
dbTime: HH:mm:ss

table-mappings:
TABLE_1_NAME : LEGACY_TABLE_1_NAME
TABLE_2_NAME : LEGACY_TABLE_2_NAME
```

sqlCodePointShift

（選用）指定整數值，代表用於資料上的 SQL 程式碼點轉移。預設為 0，這表示應用程式不會轉移程式碼點。將此設定與用於現代化應用程式的 SQL 程式碼點轉移參數對齊。當您使用程式碼點轉移時，此參數最常見的值為 384。

```
load.sqlCodePointShift : 384
```

batchSize

（選用）指定整數值，代表在將實際批次陳述式傳送至資料庫之前要處理的記錄數。預設為 0。

```
load.batchSize: 500
```

格式

（選用）指定資料庫載入操作期間用於日期/時間轉換的日期和時間格式模式。

- `load.format.localDate`：本機日期格式模式。預設值為 `dd.MM.yyyy|dd/MM/yyyy|yyyy-MM-dd`。
- `load.format.dbDate`：資料庫日期格式模式。預設值為 `yyyy-MM-dd`。
- `load.format.localTime`：本機時間格式模式。預設值為 `HH:mm:ss|HH.mm.ss`。
- `load.format.dbTime`：資料庫時間格式模式。預設值為 `HH:mm:ss`。

資料表映射

（選用）指定舊版和現代資料表名稱之間客戶提供的映射集合。DSNUTILB 公用程式會使用這些映射。

以下列格式指定值：MODERN_TABLE_NAME：LEGACY_TABLE_NAME

請見此處範例：

```
table-mappings:
 TABLE_1_NAME : LEGACY_TABLE_1_NAME
 TABLE_2_NAME : LEGACY_TABLE_2_NAME
 ...
 TABLE_*N*_NAME : LEGACY_TABLE_*N*_NAME
```

> **Note**
>
> 當公用程式應用程式啟動時，它會明確記錄所有提供的映射。

使用 AWS Blu Age 引擎新增受管應用程式的組態屬性

您可以在重構應用程式的 `config` 資料夾中新增檔案，讓您存取 AWS Blu Age 執行期引擎的新功能。您必須將此檔案命名為 `user-properties.yml`。此檔案不會取代應用程式定義，而是將其延伸。本主題說明您可以在 `user-properties.yml` 檔案中包含的屬性。

> **Note**
>
> 您無法變更某些參數，因為它們是由 AWS Mainframe Modernization 或應用程式定義控制。您應用程式的應用程式定義中定義的所有參數，其優先順序都高於您在 中指定的參數`user-properties.yml`。

如需重構應用程式結構的詳細資訊，請參閱 AWS Blu Age 受管應用程式的結構。

下圖顯示在 AWS Blu Age 範例應用程式 PlanetsDemo 的結構中，在哪裡找到`user-properties.yml`檔案。

```
PlanetsDemo-v1/
   ## config/
   #  ## application-PlanetsDemo.yml
   #  ## user-properties.yml
   ## jics/
   ## webapps/
```

組態屬性參考

這是可用屬性的清單。所有參數都是選用的。

主題

- Gapwalk 應用程式屬性
- Gapwalk 批次指令碼屬性
- Gapwalk Blugen 屬性
- Gapwalk CL 命令屬性
- Gapwalk CL 執行器屬性
- Gapwalk JHDB 屬性
- Gapwalk JICS 屬性
- Gapwalk 執行期屬性
- Gapwalk 公用程式屬性
- 其他屬性

Gapwalk 應用程式屬性

bluesam.fileLoading.commitInterval

選用。BluSAM 遞交間隔。

類型：數字

預設：100000

card.encoding

選用。卡片編碼：與 搭配使用`useControlMVariable`。

類型：字串

預設：CP1145

checkinputfilesize

選用。指定是否釋出檢查，如果檔案大小是記錄大小的倍數。

類型：布林值

預設：false

database.cursor.overflow.allowed

選用。指定是否允許游標溢位。將 設定為 `true`以在游標上執行下一個呼叫，無論其位置為何。設定為 `false`，檢查游標是否位於最後一個位置，再對游標執行下一個呼叫。只有在游標為可掃描（敏感或不敏感）時才啟用

類型：布林值

預設：true

dataSimplifier.onInvalidNumericData

選用。解碼無效數值資料時如何反應。允許的值為 `reject`、`toleratespaces`、`toleratespaceslowvalues`、`toleratemost`。

類型：字串

預設：拒絕

defaultKeepExistingFiles

選用。指定是否設定資料集預設的先前值。

類型：布林值

預設：false

disposition.checkexistence

選用。指定是否針對具有 DISP SHR 或 OLD 的資料集，釋出檔案存在的檢查。

類型：布林值

預設：false

externalSort.threshold

選用。排序閾值：何時切換到外部（合併）排序。

類型：字串

預設：null

`externalSort.threshold: 12MB`

blockSizeDefault

選用。用於 BDW 位元組的預設區塊大小。

類型：數字

預設：32760

```
blockSizeDefault: 32760
```

forceHR

選用。指定是否在主控台或檔案輸出上使用人類可讀 SYSPRINT。

類型：布林值

預設：false

forcedDate

選用。強制資料庫中的特定日期和時間。僅在開發和測試期間使用 。

預設：null

```
forcedDate: 2022-08-26T12:59:58.123456+01:57
```

frozenDate

選用。凍結資料庫中的日期和時間。僅在開發和測試期間使用 。

預設：false

```
frozenDate: false
```

ims.messages.extendedSize

選用。指定是否設定 extendedSize on ims 訊息。

類型：布林值

預設：false

lockTimeout

選用。當 無法在指定的時間範圍內取得鎖定時，交易的逾時以毫秒為單位。

類型：數字

預設：500

mapTransfo.prefixes

選用。轉換 controlM 變數時要使用的字首清單。每個以逗號分隔。

類型：字串

預設：&，@，%%

query.useConcatCondition

選用。指定金鑰條件是否由金鑰串連建置。

類型：布林值

預設：false

rollbackOnRTE

選用。指定是否要在執行時間例外狀況時轉返隱含執行單位交易。

類型：布林值

預設：false

sctThreadLimit

選用。觸發指令碼的執行緒限制。

類型：數字

預設：5

sqlCodePointShift

選用。sql 程式碼點轉移。將舊版 rdbms 資料遷移至現代 rdbms 時，我們可能遇到的控制角色的程式碼點轉移。例如，您可以指定 384來比對 Unicode 字元 \u0180。

類型：數字

預設：0

sqlIntegerOverflowAllowed

選用。指定是否允許 SQL 整數溢位，表示是否允許在主機變數中放置較大的值。

類型：布林值

預設：false

stepFailWhenAbend

選用。指定在步驟失敗或完成執行時是否引發無效。

類型：布林值

預設：true

stopExecutionWhenProgNotFound

選用。指定在找不到程式時是否停止執行。如果設定為 `true`，則如果找不到程式， 會中斷執行。

類型：布林值

預設：true

uppercaseUserInput

選用。指定使用者輸入是否為大寫。

類型：布林值

預設：true

useControlMVariable

選用。指定是否使用 control-M 規格取代變數。

類型：布林值

預設：false

jcl.checkpoint.expireTimeout

選用。指定在持久性提供者或記憶體內登錄檔中保留 JCL 檢查點的持續時間。

類型：數字

預設：-1

jcl.checkpoint.expireTimeoutUnit

選用。指定 `jcl.checkpoint.expireTimeout` 屬性的持續時間單位。支援的列舉常值：`java.util.concurrent.TimeUnit`。

類型：字串

預設：SECONDS

Gapwalk 批次指令碼屬性

編碼

選用。Batchscript 專案中使用的編碼 （ 而非 groovy)。預期有效的編碼 CP1047、IBM930、ASCII、UTF-8...

類型：字串

預設：ASCII

Gapwalk Blugen 屬性

manager.trancode

選用。對話方塊管理員轉錄碼映射。可讓您將 JICS 交易代碼映射至對話方塊管理員。預期的格式為 `trancode1:dialogManager1;trancode2:dialogManager2;`。

類型：字串

預設：null

```
managers.trancode: OR12:MYDIALOG1
```

Gapwalk CL 命令屬性

command-off

選用。要關閉的命令清單，以逗號分隔。允許的值為 PGM_BASIC、RCVMSG、SNDRCVF、CHGVAR、QCLRDTAQ、RTVJOBA、ADDLFM、ADDPFM、RCVF、OVRD 當您想要停用或覆寫現有程式時很有用。 PGM_BASIC 是專為偵錯目的設計的特定 AWS Blu Age Runtime 程式。

類型：字串

預設：null

spring.datasource.primary.jndi-name

選用。主要 Java 命名和目錄界面 (JNDI) 資料來源。

類型：字串

預設：jdbc/primary

zonedMode

選用。編碼或解碼區域資料類型的模式。允許的值為 EBCDIC_STRICT / EBCDIC_MODIFIED / AS400。

類型：字串

預設：EBCDIC_STRICT

Gapwalk CL 執行器屬性

cl.configuration.context.encoding

選用。CL 檔案的編碼。預期有效的編碼 CP1047、IBM930、ASCII、UTF-8...

類型：字串

預設：CP297

cl.zonedMode

選用。編碼或解碼控制語言 (CL) 命令的模式。允許的值為 EBCDIC_STRICT / EBCDIC_MODIFIED / AS400。

類型：字串

預設：EBCDIC_STRICT

Gapwalk JHDB 屬性

ims.programs

選用。要使用的 IMS 程式清單。以分號 (;) 分隔每個參數，並以逗號 () 分隔每個交易,。例如：ims.programs:
PCP008,PCT008;PCP054,PCT054;PCP066,PCT066;PCP068,PCT068;

類型：字串

預設：null

jhdb.checkpointPath

選用。如果 jhdb.checkpointPersistence 不是none，則此參數可讓您設定檢查點持久性路徑 (checkpoint.dat 檔案儲存位置），則登錄檔中包含的所有檢查點資料都會序列化，並備份在提供的資料夾中的檔案 (checkpoint.dat) 中。請注意，只有檢查點資料 (scriptId、stepId、資料庫位置和檢查點區域） 會考慮此備份。

類型：字串

預設：file：./setup/

jhdb.checkpointPersistence

選用。檢查點持久性模式。允許的值為 none / add / end。add 使用 在建立新的檢查點並新增至登錄檔時保留檢查點。使用 end 在伺服器關閉時保留檢查點。任何其他值都會停用持久性。請注意，每次將新的檢查點新增至登錄檔時，所有現有的檢查點都會序列化，並清除檔案。它不會附加到 檔案中的現有資料。因此，根據檢查點的數量，它可能會對效能產生一些影響。

類型：字串

預設：none

jhdb.configuration.context.encoding

選用。JHDB (Java 階層式資料庫） 編碼。預期有效的編碼字串
CP1047、IBM930、ASCII、UTF-8...

類型：字串

預設：CP297

jhdb.identificationCardData

選用。用來將一些「運算子識別卡資料」硬式編碼至 CARD 參數指定的 MID 欄位。

類型：字串

預設: ""

jhdb.lterm

選用。在 IMS 模擬的情況下，可讓您強制使用常見的邏輯終端機 ID。如果未設定，則會使用 sessionId。

類型：字串

預設：null

jhdb.metadata.extrapath

組態參數，指定 psb 和 dbds 資料夾的額外執行時間特定根資料夾。

類型：字串

預設：file：./setup/

> **Note**
>
> 目前，針對部署限制，您必須在應用程式的組態目錄中或組態目錄的子目錄中複製 dbds 和 psbs 目錄：例如 config/setup
>
> ```
> config
> |- setup
>    |- dbds
>    |- psbs
> ```
>
> 在 application-jhdb.yml 中設定 和 `jhdb.metadata.extrapath: file: ./config/setup/`

jhdb.navigation.cachenexts

選用。RDBMS 階層導覽中使用的快取持續時間 （以毫秒為單位）。

類型：數字

預設：5000

jhdb.query.limitJoinUsage

選用。指定是否在 RDBMS 圖形上使用限制聯結用量參數。

類型：布林值

預設：true

jhdb.use-db-prefix

選用。指定是否要在 RDBMS 的階層導覽中啟用資料庫字首。

類型：布林值

預設：true

Gapwalk JICS 屬性

jics.data.dataJsonInitLocation

選用。分析器從剖析 CSD 準備的 json 檔案位置，用於初始化 jics 資料庫，

類型：字串

預設: ""

jics.db.dataScriptLocation

選用。由 Analyzer 從大型主機剖析 CSD 匯出所準備的 initJics.sql 指令碼位置。

類型：字串

預設: ""

jics.db.dataTestQueryLocation

選用。包含單一 sql 查詢的 sql 指令碼位置，預期會傳回物件計數（例如：計算 jics 程式資料表中的記錄數）。如果計數等於 0，則會使用`jics.db.dataScriptLocation`指令碼載入資料庫，否則會略過資料庫載入。

類型：字串

預設: ""

jics.db.ddlScriptLocation

選用。Jics ddl 指令碼位置。可讓您使用 .sql 指令碼啟動 jics 資料庫結構描述。

類型：字串

預設: ""

```
jics.db.ddlScriptLocation: ./jics/sql/jics.sql
```

jics.db.schemaTestQueryLocation

選用。應包含唯一查詢的 sql 檔案位置，該查詢會傳回 jics 結構描述中的物件數目（如果有的話）。

類型：字串

預設: ""

jics.runUnitLauncherPool.enable

選用。指定是否要在 JICS 中啟用執行單位啟動器集區。

類型：布林值

預設：false

jics.runUnitLauncherPool.size

選用。JICS 中的執行單位啟動器集區大小。

類型：數字

預設：20

jics.runUnitLauncherPool.validationInterval

選用：JICS 中執行單位啟動器集區的驗證間隔，以毫秒為單位。

類型：數字

預設：1000

jics.queues.sqs.region

選用。 AWS 區域 適用於 JICS 的 Amazon SQS。建議將已部署應用程式的相同區域設定為效能，但不是強制性的。

類型：字串

預設：eu-west-1

jics.xa.agent.timeout

選用。定義負責管理分散式交易的 xa 代理程式完成其操作的最長持續時間。

類型：數字

預設：null

mq.queues.sqs.region

選用。Amazon SQS MQ AWS 區域 服務的 。

類型：字串

預設：eu-west-3

taskExecutor.allowCoreThreadTimeOut

選用。指定是否允許核心執行緒在 JCIS 中逾時。這可讓動態成長和縮減，即使與非零佇列結合（因為最大集區大小只會在佇列已滿時成長）。

類型：布林值

預設：false

taskExecutor.corePoolSize

選用。當終端機中的交易透過 groovy 指令碼啟動時，會建立新的執行緒。使用此參數來設定核心集區大小。

類型：數字

預設：5

taskExecutor.maxPoolSize

選用。當終端機中的交易透過 groovy 指令碼啟動時，就會建立新的廣告。使用此參數來設定最大集區大小（平行執行緒的數量上限）。

類型：數字

預設：10

taskExecutor.queueCapacity

選用。當終端機中的交易透過 groovy 指令碼啟動時，就會建立新的廣告。使用此參數來設定佇列大小。(= `taskExecutor.maxPoolSize` 達到 時待處理交易的最大數量）

類型：數字

預設：50

Gapwalk 執行期屬性

cacheMetadata

選用。指定是否快取資料庫中繼資料。

類型：布林值

預設：true

check-groovy-file

選用。指定是否要在註冊之前檢查 groovy 檔案內容。

類型：布林值

預設：true

databaseStatistics

選用。指定是否允許 SQL 建置器收集和顯示統計資料資訊。

類型：布林值

預設：false

dateTimeFormat

選用。dateTimeFormat 說明如何將資料庫日期時間戳記類型溢出到資料簡化器實體。允許的值為 ISO / EUR / USA / LOCAL

類型：字串

預設：ISO

dbDateFormat

選用。資料庫目標日期格式。

類型：字串

預設：yyyy-MM-dd

dbTimeFormat

選用。資料庫目標時間格式。

類型：字串

預設：HH：mm：ss

dbTimestampFormat

選用。資料庫目標時間戳記格式。

類型：字串

預設：yyyy-MM-dd HH：mm：ss.SSSSSS

fetchSize

選用。游標的 fetchSize 值。依載入/卸載 utils 使用區塊擷取資料時使用 。

類型：數字

預設：10

forceDisableSQLTrimStringType

選用。指定是否停用所有 sql 字串參數的修剪。

類型：布林值

預設：false

localDateFormat

選用。本機日期格式的清單。使用 分隔每個格式 | 。

類型：字串

localTimeFormat

選用。本機時間格式的清單。使用 | 分隔每個格式。

類型：字串

localTimestampFormat

選用。本機時間戳記格式的清單。使用 分隔每個格式 | 。

類型：字串

預設：

pgmDateFormat

選用。程式中使用的日期時間格式。

類型：字串

預設：yyyy-MM-dd

pgmTimeFormat

選用。用於 pgm （程式） 執行的時間格式。

類型：字串

預設值：https：//HH.mm.ss

pgmTimestampFormat

選用。時間戳記格式。

類型：字串

預設： yyyy-MM-dd-HH.mm.ss.SSSSSS

Gapwalk 公用程式屬性

jcl.type

Optional. `.jcl` 檔案類型。允許的值為 `jcl` / `vse`。如果非 vse jcl 的檔案為空，IDCAMS 公用程式 Print/REPRO 命令會傳回 4。

類型：字串

預設：mvs

listcat.variablelengthpreprocessor.enabled

選用。指定是否啟用 LISTCAT 命令的可變長度預處理器。

類型：布林值

預設：false

listcat.variablelengthpreprocessor.type

選用。如果您啟用 ，Listcat 檔案中包含的物件類型`listcat.variablelengthpreprocessor.enabled`。允許的值為 `rdw` / `bdw`。

類型：字串

預設：rdw

load.batchSize

選用。負載公用程式批次大小。

類型：數字

預設：0

load.format.dbDate

選用。要使用的負載公用程式資料庫格式。

類型：字串

預設：yyyy-MM-dd

load.format.dbTime

選用。要使用的負載公用程式資料庫時間。

類型：字串

預設：HH：mm：ss

load.format.localDate

選用。要使用的負載公用程式本機日期格式。

類型：字串

預設： dd.MM.yyyy|dd/MM/yyyy|yyyy-MM-dd

load.format.localTime

選用。要使用的負載公用程式本機時間格式。

類型：字串

預設值：HH：mm：ss|HH.mm.ss

load.sqlCodePointShift

選用。負載公用程式的 SQL 程式碼指標轉移。執行轉移字元程序。當您的目標資料庫來自 DB2 為 Postgresql 時為必要。

類型：數字

預設：0

sysPunchEncoding

選用。syspunch 編碼字元集。支援的值為 Cp1047 / ASCII。

類型：字串

預設：ASCII

treatLargeNumberAsInteger

選用。指定是否將大型數字視為 Integer。根據BigDecimal預設，它們會被視為 。

類型：布林值

預設：false

unload.chunkSize

選用。用於卸載公用程式的區塊大小。

類型：數字

預設：0

unload.columnFiller

選用。卸載公用程式資料欄填充。

類型：字串

預設：空格

unload.fetchSize

選用。可讓您在卸載公用程式中處理游標時調整擷取大小。

類型：數字

預設：0

unload.format.date

選用。如果unload.useDatabaseConfiguration已啟用 ，則在卸載公用程式中使用的日期格式。

類型：字串

預設：MM/dd/yyyy

unload.format.time

選用。如果unload.useDatabaseConfiguration已啟用 ，則在卸載公用程式中使用的時間格式。

類型：字串

預設值：https：//HH.mm.ss

unload.format.timestamp

選用。如果unload.useDatabaseConfiguration已啟用 ，則在卸載公用程式中使用的時間戳記格式。

類型：字串

預設： yyyy-MM-dd-HH.mm.ss.SSSSSS

unload.nbi.whenNotNull

選用。當資料庫的值不是 null 時，要新增的 Null 位元組指標 (nbi) 值。

類型：十六進位

預設：00

unload.nbi.whenNull

選用。當資料庫的值為 Null 時，要新增的 Null Byte Indicator (nbi) 值。

類型：十六進位

預設：6F

unload.nbi.writeNullIndicator

選用。指定是否要在卸載輸出檔案中寫入 null 指標。

類型：布林值

預設：false

unload.sqlCodePointShift

選用。卸載公用程式的 SQL 程式碼指標。執行轉移字元程序。當您的目標資料庫來自 DB2 為 Postgresql 時為必要。

類型：數字

預設：0

unload.useDatabaseConfiguration

選用。指定是否要在卸載公用程式中使用 application-main.yml 的日期或時間組態。

類型：布林值

預設：false

unload.varCharIsNull

選用。在 INFTILB 程式中使用此參數，如果設定為 `true`，則所有具有空白（空格）值的不可為 null 欄位都會傳回空字串。

類型：布林值

預設：false

其他屬性

qtemp.cleanup.threshold.hours

選用。指定`qtemp.dblog`啟用 的時間。db 分割區生命週期（以小時為單位）。

類型：數字

預設：0

qtemp.dblog

選用。是否啟用 QTEMP 資料庫記錄。

類型：布林值

預設：false

qtemp.uuid.length

選用。QTEMP 唯一 ID 長度。

類型：數字

預設：9

quartz.scheduler.stand-by-if-error

選用。指定如果任務排程器處於待命模式，是否觸發任務執行。如果為 true，則啟用任務執行時不會觸發。

類型：布林值

預設：false

warmUpCache

選用。指定是否要在伺服器啟動時將所有 datacom 資料表資料載入暖機快取。

類型：布林值

預設：false

AWS Mainframe Modernization 應用程式定義參考

在 中 AWS Mainframe Modernization，您會在應用程式定義 JSON 檔案中設定已遷移的大型主機應用程式，這是您選擇的執行期引擎特有的。應用程式定義同時包含一般資訊和引擎特定資訊。本主題說明 AWS Blu Age 和 Rocket Software （ 先前稱為 Micro Focus) 應用程式定義，並識別所有必要和選用的元素。

內容

- 一般標頭區段
- 定義區段概觀
- AWS Blu Age 應用程式定義範例
- AWS Blu Age 定義詳細資訊
  - Listener(s) - 必要
  - AWS Blu Age 應用程式 - 必要
  - BluSAM - 選用
  - AWS Blu Age 訊息佇列 - 選用
  - AWS Blu Age 應用程式儲存 EFS 組態 - 選用
- Rocket Software （ 先前稱為 Micro Focus) 應用程式定義
- Rocket 軟體定義詳細資訊
  - Listener(s) - 必要

- 資料集位置 - 必要
- Amazon Cognito 身分驗證和授權處理常式 - 選用
- LDAP 和 Active Directory 處理常式 - 選用
- 批次設定 - 必要
- CICS 設定 - 必要
- 印表機 - 選用
- XA 資源 - 選用
- 執行時間設定 - 選用

一般標頭區段

每個應用程式定義都從範本版本和來源位置的一般資訊開始。應用程式定義的目前版本為 2.0。

使用以下結構來指定範本版本和來源位置。

```
"template-version": "2.0",
    "source-locations": [
        {
            "source-id": "s3-source",
            "source-type": "s3",
            "properties": {
                "s3-bucket": "mainframe-deployment-bucket",
                "s3-key-prefix": "v1"
            }
        }
    ]
```

> **Note**
>
> 如果您想要將 S3 ARN 輸入為 s3-bucket ，您可以使用下列語法：
>
> ```
> "template-version": "2.0",
>   "source-locations": [
>       {
>           "source-id": "s3-source",
>           "source-type": "s3",
>           "properties": {
>               "s3-bucket": "arn:aws:s3:::mainframe-deployment-bucket",
> ```

```
            "s3-key-prefix": "v1"
        }
    }
]
```

範本版本

（必要）指定應用程式定義檔案的版本。請不要變更此值。目前，唯一允許的值為 2.0。以字串指定 `template-version`。

來源位置

指定應用程式在執行時間所需的檔案和其他資源的位置。

source-id

指定位置的名稱。此名稱用於視需要參考應用程式定義 JSON 中的來源位置。

source-type

指定來源的類型。目前，唯一允許的值是 s3。

屬性

提供來源位置的詳細資訊。每個屬性皆以字串指定。

- `s3-bucket` - 必要。指定存放檔案的 Amazon S3 儲存貯體名稱。
- `s3-key-prefix` - 必要。指定存放檔案之 Amazon S3 儲存貯體中的資料夾名稱。

定義區段概觀

指定應用程式需要執行的服務、設定、資料和其他典型資源的資源定義。當您更新應用程式定義時，會比較應用程式定義 JSON 檔案先前和目前版本的 `source-locations` 和 `definition`清單，以 AWS Mainframe Modernization 偵測變更。

定義區段是引擎特定的，可能會有所變更。下列各節顯示兩個引擎的範例引擎特定應用程式定義。

AWS Blu Age 應用程式定義範例

```
{
 "template-version": "2.0",
 "source-locations": [
        {
```

```
            "source-id": "s3-source",
            "source-type": "s3",
            "properties": {
                "s3-bucket": "mainframe-deployment-bucket-aaa",
                "s3-key-prefix": "v1"
            }
        }
    ],
    "definition" : {
        "listeners": [{
            "port": 8194,
            "type": "http"
        }],
        "ba-application": {
            "app-location": "${s3-source}/murachs-v6/"
        },
        "blusam": {
            "db": {
                "nb-threads": 8,
                "batch-size": 10000,
                "name": "blusam",
                "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:blusam-FfmXLG"
            },
            "redis": {
                "hostname": "blusam.c3geul.ng.0001.usw2.cache.amazonaws.com",
                "port": 6379,
                "useSsl": true,
                "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:bluesamredis-nioefm"
            }
        }
    }
}
```

AWS Blu Age 定義詳細資訊

Listener(s) - 必要

指定您將用來透過建立的 Elastic Load Balancing AWS Mainframe Modernization存取應用程式的連接埠。使用下列結構：

```
"listeners": [{
```

```
            "port": 8194,
            "type": "http"
}],
```

port

（必要）您可以使用任何可用的連接埠，除了 0 到 1023 的已知連接埠。建議使用 8192 到 8199 的範圍。請確定沒有其他接聽程式或應用程式在此連接埠上運作。

type

（必要）目前僅支援 `http` 。

AWS Blu Age 應用程式 - 必要

使用以下結構指定引擎挑選應用程式映像檔案的位置。

```
"ba-application": {
            "app-location": "${s3-source}/murachs-v6/",
            "files-directory": "/m2/mount/myfolder",
            "enable-jics": <true|false>,
            "enable-batch-restart": <true|false>,
            "shared-app-location": "${s3-source}/shared/"
},
```

應用程式位置

Amazon S3 中存放應用程式映像檔案的特定位置。

file-directory

（選用）批次的輸入/輸出檔案位置。必須是環境層級 Amazon EFS 或 Amazon FSx 掛載點設定的子資料夾。子資料夾必須由適當的使用者擁有，以供在 內執行的 Blu Age 應用程式使用 AWS Mainframe Modernization。若要達成此目的，將磁碟機連接至 Linux Amazon EC2 執行個體時，`3001`必須建立 ID 為 的群組`101`和 ID 為 的使用者，且所需的資料夾必須由此使用者擁有。例如，這種方式可讓 Blu Age AWS Mainframe Modernization Managed 使用 *`testclient`* 資料夾。

```
groupadd -g 101 mygroup
useradd -M -g mygroup -p mypassword -u 3001 myuser
mkdir testclient
chown myuser:mygroup testclient
```

enable-jics

（選用）指定是否啟用 JICS。預設為 true。將此設定為 false 可防止產生 JICS 資料庫。

enable-batch-restart

（選用）指定是否要為批次任務啟用重新啟動功能。預設為 false。如需批次重新啟動組態的詳細資訊，請參閱使用 AWS Blu Age 引擎之受管應用程式的組態屬性`jcl.checkpoint`中字首為 的 Blu Age 引擎屬性。 AWS

shared-app-location

（選用）在 Amazon S3 中存放共用應用程式元素的進一步位置。它可以包含與應用程式位置相同的應用程式結構類型。

BluSAM - 選用

使用以下結構指定 BluSAM 資料庫和 Redis 快取。

```
"blusam": {
            "db": {
                "nb-threads": 8,
                "batch-size": 10000,
                "name": "blusam",
                "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:blusam-FfmXLG"
            },
            "redis": {
                "hostname": "blusam.c3geul.ng.0001.usw2.cache.amazonaws.com",
                "port": 6379,
                "useSsl": true,
                "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret:bluesamredis-nioefm"
            }
}
```

db

指定與應用程式搭配使用之資料庫的屬性。資料庫必須是 Aurora PostgreSQL 資料庫。您可以指定下列屬性：

- `nb-threads` -（選用）指定有多少專用執行緒用於 BluSAM 引擎所依賴的寫入落後機制。預設值為 8。

- batch-size - （選用）指定寫入落後機制用來啟動批次儲存操作的閾值。閾值代表將啟動批次儲存操作的修改後記錄數目，以確保修改後的記錄持續存在。觸發條件本身是以批次大小和經過一秒的組合為基礎，以先達到者為準。預設值為 10000。
- name - （選用）指定資料庫的名稱。
- secret-manager-arn - 指定包含資料庫憑證之秘密的 Amazon Resource Name (ARN)。如需詳細資訊，請參閱步驟 4：建立和設定 AWS Secrets Manager 資料庫秘密。

Redis

指定 Redis 快取的屬性，供應用程式用來將臨時資料存放在中央位置以改善效能。我們建議您同時加密和密碼保護 Redis 快取。

- hostname - 指定 Redis 快取的位置。
- port - 指定連接埠，通常是 6379，其中 Redis 快取會傳送和接收通訊。
- useSsl - 指定是否加密 Redis 快取。如果快取未加密，請將 useSsl設定為 false。
- secret-manager-arn - 指定包含 Redis 快取密碼之秘密的 Amazon Resource Name (ARN)。如果 Redis 快取未受密碼保護，請勿指定 secret-manager-arn。如需詳細資訊，請參閱步驟 4：建立和設定 AWS Secrets Manager 資料庫秘密。

AWS Blu Age 訊息佇列 - 選用

指定 AWS Blu Age 應用程式的 JMS-MQ 連線詳細資訊。

```
"message-queues": [
  {
      "product-type": "JMS-MQ",
      "queue-manager": "QMgr1",
      "channel": "mqChannel1",
      "hostname": "mqserver-host1",
      "port": 1414,
      "user-id": "app-user1",
      "ssl-cipher": "*TLS12ORHIGHER",
      "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:sample/mq/test-279PTa"
  },
  {
      "product-type": "JMS-MQ",
      "queue-manager": "QMgr2",
      "channel": "mqChannel2",
      "hostname": "mqserver-host2",
      "port": 1412,
```

```
      "user-id": "app-user2",
      "ssl-cipher": "*TLS12ORHIGHER",
      "secret-manager-arn": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:sample/mq/test-279PTa"
  }
]
```

product-type

（必要）指定產品類型。目前，這只能是「JMS-MQ」用於 AWS 藍時應用程式。

queue-manager

（必要）指定佇列管理員的名稱。

通道

（必要）指定伺服器連線頻道的名稱。

hostname

（必要）指定訊息佇列伺服器的主機名稱。

port

（必要）指定伺服器接聽的接聽程式連接埠號碼。

user-id

（選用）指定允許在指定頻道上執行訊息佇列操作的使用者帳戶 ID。

ssl-cipher

（選用）指定連線的 SSL 密碼規格。

secret-manager-arn

（選用）指定提供指定使用者密碼的 Secrets Manager 的 Amazon Resource Name (ARN)。

AWS Blu Age 應用程式儲存 EFS 組態 - 選用

使用以下結構指定應用程式儲存體 EFS 存取點詳細資訊。

```
"ba-application": {
            "file-permission-mask": "UMASK002"
},
"efs-configs": [
           {
```

```
            "file-system-id": "fs-01376dfsvfvrsvsr",
            "mount-point": "/m2/mount/efs-ap2",
            "access-point-id": fsap-0eaesefvrefrewgv8"
           }
]
```

file-system-id

（必要）存取點套用的 EFS 檔案系統 ID。模式："fs-(【0-9a-f】{8，40}){1，128}$"

掛載點

（必要）應用程式層級檔案系統的掛載點。這必須與環境層級儲存體掛載點不同。

access-point-id

（必要）Amazon EFS 指派的存取點 ID。模式："^fsap-(【0-9a-f】{8，40}){1，128}$"

file-permission-mask

（選用）定義應用程式程序所建立檔案的檔案建立遮罩。例如，當值設為 時UMASK006，所有檔案都會有 660 的許可。這表示只有檔案擁有者和檔案群組具有讀取和寫入存取權，而其他使用者則沒有任何許可。

> **Note**
>
> 只有在使用應用程式層級 EFS 儲存體時，才會考慮為此欄位設定的值。

> **Note**
>
> 提供 efs 組態時，必須在應用程式定義區段中指定 files-directory。它必須是在應用程式層級設定的 Amazon EFS 掛載點的子資料夾。

Rocket Software（先前稱為 Micro Focus) 應用程式定義

下列範例定義區段適用於 Rocket Software 執行期引擎，並同時包含必要和選用的元素。

```
{
 "template-version": "2.0",
 "source-locations": [
        {
```

```
            "source-id": "s3-source",
            "source-type": "s3",
            "properties": {
                "s3-bucket": "mainframe-deployment-bucket-aaa",
                "s3-key-prefix": "v1"
            }
        }
    ],
    "definition" : {
        "listeners": [{
            "port": 5101,
            "type": "tn3270"
        }],
        "dataset-location": {
            "db-locations": [{
                "name": "Database1",
                "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456"
            }]
        },
        "cognito-auth-handler": {
            "user-pool-id": "cognito-idp.us-west-2.amazonaws.com/us-west-2_rvYFnQIxL",
            "client-id": "58k05jb8grukjjsudm5hhn1v87",
            "identity-pool-id": "us-west-2:64464b12-0bfb-4dea-ab35-5c22c6c245f6"
        },
        "ldap-ad-auth-handler": {
            "ldap-ad-connection-secrets": [LIST OF AD-SECRETS]
        },
        "batch-settings": {
            "initiators": [{
                "classes": ["A", "B"],
                "description": "initiator...."
            }],
            "jcl-file-location": "${s3-source}/batch/jcl",
            "program-path": "/m2/mount/libs/loadlib:$EFS_MOUNT/emergency/loadlib",
            "system-procedure-libraries":"SYS1.PROCLIB;SYS2.PROCLIB",
            "aliases": [
               {"alias": "FDSSORT", "program": "SORT"},
               {"alias": "MFADRDSU", "program": "ADRDSSU"}
            ]
        },
        "cics-settings": {
            "binary-file-location": "${s3-source}/cics/binaries",
            "csd-file-location": "${s3-source}/cics/def",
            "system-initialization-table": "BNKCICV"
```

```
        },
        "jes-printers": [
           {
             "name": "printerName",
              "classes": [
                 "A",
                 "B"
             ],
             "description": "printer desc....",
             "exit-module": {
                 "name": "lrsprte6"
                }
             }
         ],
        "xa-resources" : [{
            "name": "XASQL",
            "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456",
            "xa-connection-type": "postgres",
            "module": "${s3-source}/xa/ESPGSQLXA64.so"
        }],
        "runtime-settings": {
          "base-configuration-location": "${s3-source}/exported.json",
          "environment-variables": {
            "ES_JES_RESTART": "N",
            "EFS_MOUNT": "/m2/mount/efs",
            "LRSQ_ADDRESS": "<lrsq-address>"
          }
        }
    }
}
```

Rocket 軟體定義詳細資訊

Rocket Software 應用程式定義檔案定義區段中的內容會有所不同，取決於遷移大型主機應用程式在執行時間所需的資源。

Listener(s) - 必要

使用下列結構指定接聽程式：

```
"listeners": [{
    "port": 5101,
    "type": "tn3270"
```

```
}],
```

port

對於 tn3270，預設值為 5101。對於其他類型的服務接聽程式，連接埠會有所不同。您可以使用任何可用的連接埠，除了 0 到 1023 的已知連接埠。每個接聽程式都應有一個獨特的連接埠。接聽程式不應共用連接埠。如需詳細資訊，請參閱 Micro Focus Enterprise Server 文件中的接聽程式控制。

type

指定服務接聽程式的類型。如需詳細資訊，請參閱 Micro Focus Enterprise Server 文件中的接聽程式。

資料集位置 - 必要

使用下列結構指定資料集位置。

```
"dataset-location": {
            "db-locations": [{
                "name": "Database1",
                "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456"
            }],
 }
```

db-locations

指定遷移應用程式建立的資料集位置。目前， 僅 AWS Mainframe Modernization 支援來自單一 VSAM 資料庫的資料集。

- `name` - 指定資料庫執行個體的名稱，其中包含遷移應用程式建立的資料集。
- `secret-manager-arn` - 指定包含資料庫憑證之秘密的 Amazon Resource Name (ARN)。

Amazon Cognito 身分驗證和授權處理常式 - 選用

AWS Mainframe Modernization 使用 Amazon Cognito 對遷移的應用程式進行身分驗證和授權。使用以下結構指定 Amazon Cognito 身分驗證處理常式。

```
"cognito-auth-handler": {
```

```
            "user-pool-id": "cognito-idp.Region.amazonaws.com/Region_rvYFnQIxL",
            "client-id": "58k05jb8grukjjsudm5hhn1v87",
            "identity-pool-id": "Region:64464b12-0bfb-4dea-ab35-5c22c6c245f6"
}
```

user-pool-id

指定 Amazon Cognito 使用者集區， AWS Mainframe Modernization 用於驗證已遷移應用程式的使用者。使用者集 AWS 區域 區的 應與 AWS Mainframe Modernization 應用程式的 AWS 區域 相符。

client-id

指定已驗證使用者可存取的遷移應用程式。

identity-pool-id

指定 Amazon Cognito 身分集區，其中已驗證的使用者會交換使用者集區字符，以取得允許使用者存取的登入資料 AWS Mainframe Modernization。身分集 AWS 區域 區的 應與 AWS Mainframe Modernization 應用程式的 AWS 區域 相符。

LDAP 和 Active Directory 處理常式 - 選用

您可以將應用程式與 Active Directory (AD) 或任何類型的 LDAP 伺服器整合，讓應用程式的使用者能夠使用其 LDAP/AD 登入資料進行授權和身分驗證。

將您的應用程式與 AD 整合

1. 遵循 Micro Focus Enterprise Server 文件中設定適用於企業伺服器安全性的 Active Directory 中所述的步驟。
2. 使用 AD/LDAP 詳細資訊為您要與應用程式搭配使用的每個 AD/LDAP 伺服器建立 AWS Secrets Manager 秘密。如需如何建立秘密的資訊，請參閱 AWS Secrets Manager 《使用者指南》中的建立 AWS Secrets Manager 秘密。針對秘密類型，選擇其他類型的秘密，並包含下列鍵值對。

```
{
    "connectionPath"       : "<HOST-ADDRESS>:<PORT>",
    "authorizedId"         : "<USER-FULL-DN>",
    "password"             : "<PASSWORD>",
    "baseDn"               : "<BASE-FULL-DN>",
    "userClassDn"          : "<USER-TYPE>",
    "userContainerDn"      : "<USER-CONTAINER-DN>",
```

```
    "groupContainerDn"     : "<GROUP-CONTAINER-DN>",
    "resourceContainerDn"  : "<RESOURCE-CONTAINER-DN>"
}
```

⚠ 安全建議

- 對於 connectionPath， AWS Mainframe Modernization 支援 LDAP 和 LDAP over SSL (LDAPS) 通訊協定。我們建議您使用 LDAPS，因為它更安全，並防止登入資料出現在網路傳輸中。
- 對於 authorizedId和 password，我們建議您指定使用者的登入資料，其許可不超過應用程式執行所需的最嚴格唯讀和驗證許可。
- 建議您定期輪換 AD/LDAP 登入資料。
- 請勿使用使用者名稱awsuser或 建立 AD 使用者mfuser。這兩個使用者名稱會保留供 AWS 使用。

以下是範例。

```
{
    "connectionPath" : "ldaps://msad4.m2.example.people.aws.dev:636",
    "authorizedId" :
 "CN=LDAPUser,OU=Users,OU=msad4,DC=msad4,DC=m2,DC=example,DC=people,DC=aws,DC=dev",
    "password" : "ADPassword",
    "userContainerDn" : "CN=Enterprise Server Users,CN=Micro Focus,CN=Program
 Data,OU=msad4,DC=msad4,DC=m2,DC=example,DC=people,DC=aws,DC=dev",
    "groupContainerDn" : "CN=Enterprise Server Groups,CN=Micro Focus,CN=Program
 Data,OU=msad4,DC=msad4,DC=m2,DC=example,DC=people,DC=aws,DC=dev",
    "resourceContainerDn" : "CN=Enterprise Server Resources,CN=Micro
 Focus,CN=Program Data,OU=msad4,DC=msad4,DC=m2,DC=example,DC=people,DC=aws,DC=dev"
}
```

使用客戶管理的 KMS 金鑰建立秘密。您必須授予 AWS Mainframe Modernization 秘密的 GetSecretValue和 DescribeSecret 許可，以及 KMS 金鑰的 Decrypt和 DescribeKey許可。如需詳細資訊，請參閱 AWS Secrets Manager 《 使用者指南》中的 KMS 金鑰的許可。

3. 將以下內容新增至您的應用程式定義。

```
"ldap-ad-auth-handler": {
    "ldap-ad-connection-secrets": [LIST OF AD/LDAP SECRETS]
```

```
}
```

以下是範例。

```
"ldap-ad-auth-handler": {
    "ldap-ad-connection-secrets": ["arn:aws:secrets:1234:us-east-1:secret:123456"]
}
```

如果應用程式已與 LDAP 整合並已啟動，您必須提供登入資料，才能執行支援授權清單中提及的至少一個應用程式相關操作。

LDAP/AD 身分驗證處理常式適用於 Micro Focus (Rocket) 8.0.11 和更新版本。

> **Note**
>
> 目前，LDAP 管理員必須在其 LDAP 目錄中的「OPERCMDS」企業伺服器資源中的`casstart`公用程式上提供「更改」許可。這需要對所有所需的預設使用者（例如，CICSUSER - 如果應用程式與 CICS 相關）完成，才能成功啟動應用程式。

提供 LDAP 使用者登入資料以進行身分驗證和授權

1. AWS Secrets Manager 使用下列索引鍵和值建立 ：

```
{
  "username" : "<USERNAME>",
  "password" : "<PASSWORD>"
}
```

> **Important**
>
> 您應該有權在正在使用的 Secrets Manager `GetSecretValue` 上執行`DescribeSecrets`和 。此外，建立 KMS 金鑰和 必要許可的關聯 AWS Secrets Manager，如選擇 AWS KMS key所述。

2. 選擇 Secrets Manager 參數。

AWS console

從 AWS 主控台執行操作時，將有一個選項可選擇需要傳遞的 Secrets Manager。

AWS CLI (or SDK)

從 AWS CLI （或 SDK) 執行操作時，API 參數`auth-secrets-manager-arn`應與 Secrets Manager ARN 一起傳遞。

以下是目前支援授權的應用程式操作清單：

- `StartBatchJob`
- `CancelBatchJobExecution`
- `ListBatchJobRestartPoints`

批次設定 - 必要

使用下列結構指定作為應用程式一部分執行的批次任務所需的詳細資訊。

```
"batch-settings": {
            "initiators": [{
                "classes": ["A", "B"],
                "description": "initiator...."
            }],
            "jcl-file-location": "${s3-source}/batch/jcl",
            "program-path": "/m2/mount/libs/loadlib:$EFS_MOUNT/emergency/loadlib",
            "system-procedure-libraries":"SYS1.PROCLIB;SYS2.PROCLIB",
            "aliases": [
               {"alias": "FDSSORT", "program": "SORT"},
               {"alias": "MFADRDSU", "program": "ADRDSSU"}
    ]
}
```

啟動器

指定批次啟動器，該啟動器會在遷移的應用程式成功啟動時啟動，並繼續執行，直到應用程式停止為止。您可以為每個啟動器定義一或多個類別。您也可以定義多個啟動器。例如：

```
"batch-settings": {
            "initiators": [
```

```
                {
                  "classes": ["A", "B"],
                  "description": "initiator...."
                },
                {
                  "classes": ["C", "D"],
                  "description": "initiator...."
                }

        ],
}
```

如需詳細資訊，請參閱 Micro Focus Enterprise Server 文件中的如何定義批次啟動器或印表機 SEP。

- `classes` - 指定啟動器可執行的任務類別。您最多可以使用 36 個字元。您可以使用下列字元：A-Z 或 0-9。
- `description` - 描述啟動器的用途。

jcl-file-location

指定遷移應用程式執行之批次任務所需的 JCL （任務控制語言） 檔案位置。

program-path

指定當 JCL 中的程式不在預設位置時，執行批次任務所需的路徑。不同的路徑名稱會以冒號 (:) 分隔。

> **Note**
>
> 程式路徑只能是 EFS 路徑。

system-procedure-libraries

指定將搜尋 JCL 程序的預設分割資料集。不過，在 JCL 中或透過 JCLLIB 陳述式找不到此程序。這些資料集必須編製目錄，而且必須使用目錄名稱。而項目會以分號 (;) 分隔。

別名

定義 JCL 中使用的公用程式和程式名稱映射到公用程式的實作名稱。以及第三方批次公用程式 AWS （例如 M2SFTP, M2WAIT、Syncsort 等） 可以選擇性地具有別名，以消除變更 JCL 的需求。例如：

- 適用於 SORT 的 FDSSORT 別名 FDSSORT 和適用於 ICETOOL 的 FDSICET 別名
- ADRDSSU 的 ADRDSSU 別名 MFADRDSU
- 同步 SORT 的別名 DMXMFSRT

CICS 設定 - 必要

使用下列結構，指定做為應用程式一部分執行之 CICS 交易所需的詳細資訊。

```
"cics-settings": {
            "binary-file-location": "${s3-source}/cics/binaries",
            "csd-file-location": "${s3-source}/cics/def",
            "system-initialization-table": "BNKCICV"
}
```

binary-file-location

指定 CICS 交易程式檔案的位置。

csd-file-location

指定此應用程式的 CICS 資源定義 (CSD) 檔案位置。如需詳細資訊，請參閱 Micro Focus Enterprise Server 文件中的 CICS 資源定義。

system-initialization-table

指定遷移應用程式使用的系統初始化資料表 (SIT)。SIT 資料表的名稱最多可達 8 個字元。您可以使用 A-Z、0-9、$、@ 和 #。如需詳細資訊，請參閱 Micro Focus Enterprise Server 文件中的 CICS 資源定義。

印表機 - 選用

使用以下結構指定 jes 印表機。

```
"jes-printers": [
    {
        "name": "printerName",
        "classes": [
            "A",
            "B"
        ],
        "description": "printer desc....",
```

```
        "exit-module": {
            "name": "lrsprte6",
            "module" : "program"
        }
    }
],
```

> **Note**
>
> 指定應用程式最多可設定 25 個印表機。

name

（必要）指定要與此印表機資源建立關聯的名稱。每個印表機的名稱必須是唯一的，並且可以使用 128 個英數字元的限制。

類別

（必要）指定適用於此印表機資源的輸出類別。最多可使用 36 個英數字元。

description

（選用）印表機的其他描述性文字。

exit-module

（選用）指定列印結束的自訂模組。沒有預設值，如果未指定，則不會使用結束模組。您可以使用受管列印結束模組或提供自己的列印結束模組。受管列印結束模組是使用 LRS 佇列`lrsprte6`的預留名稱定義，或使用模組參數提供您自己的名稱來指定位置和名稱。

的結構`exit-module`有兩個元件：

- `name`-（必要），如果使用 `exit-module` 。結束模組項目的名稱。結束模組項目名稱的限制最多為 8 個字元。
- `module` -（選用）列印結束模組二進位檔的 S3 位置。

您可以在 the section called “印表機”區段中查看更多定義結束模組的範例。

XA 資源 - 選用

使用以下結構指定應用程式所需的 XA 資源所需的詳細資訊。

```
"xa-resources" : [{
            "name": "XASQL",
            "secret-manager-arn": "arn:aws:secrets:1234:us-east-1:secret:123456",
            "xa-connection-type": "postgres",
            "module": "${s3-source}/xa/ESPGSQLXA64.so"
}]
```

> **Note**
>
> XA 資源定義已更新為包含選用`xa-connection-type`欄位。如果未提供，則會假設連線類型為「postgres」。

name

（必要）指定 XA 資源的名稱。

secret-manager-arn

（必要）指定秘密的 Amazon Resource Name (ARN)，其中包含用於連線至資料庫的登入資料。

xa-connection-type

（選用）指定 XA 資源連線類型。

模組

（必要）指定 RM 切換模組可執行檔的位置。如需詳細資訊，請參閱 Micro Focus Enterprise Server 文件中的規劃和設計 XARs。

執行時間設定 - 選用

指定執行時間設定所需的詳細資訊，以使用下列結構管理允許的環境變數。

```
"runtime-settings": {
          "base-configuration-location": "${s3-source}/exported.json",
          "environment-variables": {
            "ES_JES_RESTART": "N",
            "EFS_MOUNT": "/m2/mount/efs"
    }
}
```

base-configuration-location

（選用）指定 Micro Focus 伺服器組態的大量匯入位置。此檔案應該是有效的 JSON，且與上述定義的應用程式成品位置位於相同的 S3 位置。若要從現有應用程式匯出組態，請參閱 Rocket 軟體文件中的從 ESCWA 匯出區域一節。

環境變數

指定套用至此應用程式執行時間的 Micro Focus 支援環境變數。

- `ES_JES_RESTART` 是 Rocket Software 環境變數，可啟用 JCL 重新啟動處理。或者，您也可以使用 `ES_ALLOC_OVERRIDE`做為 Rocket Software 環境變數。
- `EFS_MOUNT` 是自訂環境變數，您的應用程式可能會用來識別環境的 EFS 掛載位置。

您可以在 Rocket Enterprise Server for UNIX 指南中存取所有 Rocket 軟體環境變數。

AWS Mainframe Modernization 資料集定義參考

如果您的應用程式需要多個資料集來處理，請在 AWS Mainframe Modernization 主控台中逐一輸入它們是無效的。反之，我們建議您建立 JSON 檔案來指定每個資料集。雖然許多參數很常見，但不同的資料集類型在 JSON 中的指定方式不同。本文件說明匯入不同類型的資料集所需的 JSON 詳細資訊。

> **Note**
>
> 匯入任何資料集之前，您必須將資料集從大型主機傳輸到 AWS。資料集的格式必須可載入至選取的執行期引擎。在許多情況下，這可以是循序檔案，但對於 Rocket Software（先前稱為 Micro Focus) VSAM，它需要採用其專屬格式。`DFCONV` 公用程式是轉換 檔案的建議方法。在資料集定義 JSON 檔案中指定儲存貯體和資料夾的名稱。
>
> 如需 Rocket Software 執行期引擎的詳細資訊，請參閱 Rocket Software 文件中的 DFCONV 批次檔案轉換。
>
> 如需 AWS Blu Age 的詳細資訊，請參閱 the section called “AWS Blu Age 執行期組態”。

主題

- 一般屬性
- VSAM 的資料集請求格式範例

- GDG 基礎的資料集請求格式範例
- PS 或 GDG 世代的範例資料集請求格式
- PO 的資料集請求格式範例

一般屬性

所有資料集都有數個常見參數。這些參數涵蓋下列區域：

- 資料集的相關資訊 (datasetName、datasetOrg、recordLength、encoding)
- 您匯入來源位置的相關資訊，也就是資料集的來源位置。這不是大型主機上的位置。這是上傳資料集的 Amazon S3 位置路徑 (externalLocation)。
- 您要匯入的位置相關資訊，也就是資料集的目標位置。此位置可以是資料庫或檔案系統，視您的執行時間引擎而定。 (storageType 和 relativePath)。
- 資料集類型的相關資訊 （特定資料集類型、格式、編碼等）。

每個資料集定義都有相同的 JSON 結構。下列範例 JSON 會顯示所有這些常見參數。

```
{
    "dataSet": {
        "storageType": "Database",
        "datasetName": "MFI01V.MFIDEMO.BNKACC",
        "relativePath": "DATA",
        "datasetOrg": {
            "type": {
                type-specific properties
                    ...
            },
        },
    },
}
```

下列屬性適用於所有資料集。

storageType

必要。適用於目標位置。指定資料集是存放在資料庫還是檔案系統中。可能的值為 Database 或 FileSystem。

- AWS Blu Age 執行期引擎：不支援檔案系統。您必須使用資料庫。

- Rocket Software 執行期引擎：支援資料庫和檔案系統。您可以將 Amazon Relational Database Service 或 Amazon Aurora 用於資料庫，以及將 Amazon Elastic File System 或 Amazon FSx for Lustre 用於檔案系統。

datasetName

（必要）指定資料集在大型主機上顯示的完整名稱。

relativePath

（必要）適用於目標位置。指定資料庫或檔案系統中資料集的相對位置。

datasetOrg

（必要）指定資料集的類型。可能值為 vsam、gdg、ps、po 或 unknown。

- AWS Blu Age 執行期引擎：僅支援 VSAM 類型資料集。
- Rocket 軟體執行期引擎：支援 VSAM、GDG、PS、PO 或未知類型資料集。

> **Note**
>
> 如果您的應用程式需要非 COBOL 資料檔案但為 PDF 或其他二進位檔案的檔案，您可以指定這些檔案，如下所示：
>
> ```
> "datasetOrg": {
>             "type": PS {
>                 "format": U
>             },
> ```

VSAM 的資料集請求格式範例

- AWS Blu Age 執行期引擎：支援。
- Rocket Software 執行期引擎：支援。

如果您要匯入 VSAM 資料集，請將 指定vsam為 datasetOrg。您的 JSON 應類似下列範例：

```
{
    "storageType": "Database",
    "datasetName": "AWS.M2.VSAM.KSDS",
    "relativePath": "DATA",
```

```
    "datasetOrg": {
        "vsam": {

            "encoding": "A",
            "format": "KS",
            "primaryKey": {
                "length": 11,
                "offset": 0
            }
        }
    },
    "recordLength": {
        "min": 300,
        "max": 300
    }
},
"externalLocation": {
    "s3Location": "s3://$M2_DATA_STORE/catalog/data/AWS.M2.VSAM.KSDS.DAT"
}
```

VSAM 資料集支援下列屬性。

編碼

（必要）指定資料集的字元集編碼。可能的值為 ASCII (A)、EBCDIC (E) 和 Unknown (?)。

格式

（必要）指定 VSAM 資料集類型和記錄格式。

- AWS Blu Age 執行期引擎：可能的值為 ESDS (ES) 和 KSDS ()KS。記錄格式可以是固定的，也可以是變數。
- Rocket Software 執行期引擎：可能的值為 ESDS (ES)、KSDS (KS) 和 RRDS ()RR。VSAM 定義包含記錄格式，因此您不需要另外指定。

primaryKey

（必要）僅適用於 VSAM KSDS 資料集。指定主索引鍵。包含主索引鍵名稱、索引鍵位移和索引鍵長度。name 是選用的； offset length是必要項目。

recordLength

（必要）指定記錄的長度。對於固定長度的記錄格式，這些值必須相符。

- AWS Blu Age 執行期引擎：對於 VSAM ESDS 和 KSDS， min是選用的，max並且是必要的。

- Rocket Software 執行期引擎： `min`和 `max`為必要項目。

externalLocation

（必要）指定來源位置：也就是上傳資料集的 Amazon S3 儲存貯體。

Blu Age 引擎特定屬性

AWS Blu Age 執行期引擎支援 VSAM 資料集的壓縮。下列範例顯示如何在 JSON 中指定此屬性。

```
{
    common properties
        ...
        "datasetOrg": {
            "vsam": {
                common properties
                ...
                "compressed": boolean,
                common properties
                ...
            }
        }
}
```

指定壓縮屬性，如下所示：

compression

（選用）指定此資料集的索引是否儲存為壓縮值。如果您有大型資料集（通常 > 100 Mb)，請考慮將此旗標設定為 `true`。

GDG 基礎的資料集請求格式範例

- AWS Blu Age 執行期引擎：不支援。
- Rocket Software 執行期引擎：支援。

如果您要匯入 GDG 基礎資料集，請將 指定`gdg`為 `datasetOrg`。您的 JSON 應類似下列範例：

```
{
    "storageType": "Database",
```

```
    "datasetName": "AWS.M2.GDG",
    "relativePath": "DATA",
    "datasetOrg": {
        "gdg": {
            "limit": "3",
            "rollDisposition": "Scratch and No Empty"
        }
    }
}
```

GDG 基礎資料集支援下列屬性。

limit

（必要）指定作用中世代或偏差的數量。對於 GDG 基礎叢集，上限為 255。

rollDisposition

（選用）指定達到或超過上限時如何處理產生資料集。可能值為 `No Scratch and No Empty`、`Scratch and No Empty`、`Scratch and Empty`、 或 `No Scratch and Empty`。預設值為 `Scratch and No Empty`。

PS 或 GDG 世代的範例資料集請求格式

- AWS Blu Age 執行期引擎：不支援。
- Rocket Software 執行期引擎：支援。

如果您要匯入 PS 或 GDG 產生資料集，請將 指定`ps`為 `datasetOrg`。您的 JSON 應類似下列範例：

```
{
    "storageType": "Database",
    "datasetName": "AWS.M2.PS.FB",
    "relativePath": "DATA",
    "datasetOrg": {
        "ps": {
            "format": "FB",
            "encoding": "A"
        }
    },
    "recordLength": {
        "min": 300,
```

```
        "max": 300
    }
},
"externalLocation": {
    "s3Location": "s3://$M2_DATA_STORE/catalog/data/AWS.M2.PS.LSEQ"
}
}
```

PS 或 GDG 世代資料集支援下列屬性。

格式

（必要）指定資料集記錄的格式。可能的值為 F、FA、FB、FBAFBM、FBS、FM、FSLSEQ、U、V、、VAVB、、VBAVBM、VBS、VM、 和 VS。

編碼

（必要）指定資料集的字元集編碼。可能的值為 ASCII (A)、EBCDIC (E) 和 Unknown (?)

recordLength

（必要）指定記錄的長度。您必須同時指定記錄的最小 (min) 和最大 (max) 長度。對於固定長度的記錄格式，這些值必須相符。

externalLocation

（必要）指定來源位置：也就是上傳資料集的 Amazon S3 儲存貯體。

PO 的資料集請求格式範例

如果您要匯入 PO 資料集，請將 指定po為 `datasetOrg`。您的 JSON 應類似下列範例：

```
{
    "storageType": "Database",
    "datasetName": "AWS.M2.PO.PROC",
    "relativePath": "DATA",
    "datasetOrg": {
        "po": {
            "format": "LSEQ",
            "encoding": "A",
            "memberFileExtensions": ["PRC"]
        }
    },
    "recordLength": {
```

```
        "min": 80,
        "max": 80
    }
},
"externalLocation": {
    "s3Location": "s3://$M2_DATA_STORE/source/proc/"
}
}
```

PO 資料集支援下列屬性。

格式

（必要）指定資料集記錄的格式。可能的值為 F、FA、FB、FBA、FBM、FBSFM、FSLSEQ、U、V、VA、VB、、VM、、、 VBA VBM VBS和 VS。

編碼

（必要）指定資料集的字元集編碼。可能的值為 ASCII (A)、EBCDIC (E) 和 Unknown (?)。

memberFileExtensions

（必要）指定包含一或多個檔案名稱副檔名的陣列，可讓您指定要包含哪些檔案做為 PDS 成員。

recordLength

（選用）指定記錄的長度。記錄的最小 (min) 和最大 (max) 長度都是選用的。對於固定長度的記錄格式，這些值必須相符。

externalLocation

（必要）指定來源位置：也就是上傳資料集的 Amazon S3 儲存貯體。

> **Note**
>
> Rocket Software 執行期引擎目前的實作會將 PDS 項目新增為動態資料集。

AWS Mainframe Modernization 中的受管執行期環境

如果您是 AWS Mainframe Modernization 的新手，請參閱下列主題以開始使用：

- 什麼是 AWS 大型主機現代化？
- 設定 AWS 大型主機現代化
- AWS 大型主機現代化入門
- 教學課程：為 AWS 藍年期設定受管執行時間
- 教學課程：設定 Rocket Software 的受管執行期（先前稱為 Micro Focus)

AWS Mainframe Modernization 中的執行時間環境是 AWS 運算資源、執行時間引擎和您指定組態詳細資訊的命名組合。執行時間環境託管一或多個應用程式。 AWS Mainframe Modernization 中的應用程式包含遷移的大型主機工作負載。您可以為您建立的環境選擇執行期引擎。如果您使用的是自動重構模式，請選擇 AWS 「模糊年齡」；如果您使用的是複寫模式，請選擇「火箭軟體」（先前稱為 Micro Focus)。您也可以選擇適合您應用程式的運算資源量，並選擇性地將儲存體連接至執行期環境。 AWS Mainframe Modernization 會為您啟用 Amazon CloudWatch 指標和記錄，以便您可以監控執行期環境。

主題

- 建立 AWS 大型主機現代化執行時間環境
- 更新 AWS Mainframe Modernization 執行時間環境
- 停止 AWS 大型主機現代化執行期環境
- 重新啟動 AWS Mainframe Modernization 執行時間環境
- 刪除 AWS 大型主機現代化執行時間環境

建立 AWS 大型主機現代化執行時間環境

使用 AWS Mainframe Modernization 主控台建立 AWS Mainframe Modernization 環境。

這些指示假設您已完成 中的步驟設定 AWS 大型主機現代化。

建立執行時間環境

建立執行期環境

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 在 AWS 區域 選擇器中，選擇您要建立環境的區域。
3. 在環境頁面上，選擇建立環境。
4. 在指定基本資訊頁面上，提供下列資訊：
   a. 在名稱和描述區段中，輸入環境的名稱。
   b. (選用)。在環境描述欄位中，輸入環境的描述。此描述可協助您和其他使用者識別執行時間環境的目的。
   c. 在引擎選項區段中，選擇自動重構的模糊年齡，或選擇 Micro Focus (Rocket) 進行複寫。
   d. 為您選取的引擎選擇版本。
   e. (選用)。在標籤區段中，選擇新增標籤，將一或多個環境標籤新增至您的環境。環境標籤是自訂屬性標籤，可協助您組織和管理 AWS 資源。
   f. 選擇下一步。
5. 在指定組態頁面上，提供下列資訊：
   a. 在可用性區段中，選擇獨立執行期環境或高可用性叢集。

      可用性模式會決定您的應用程式在執行時的可用性。獨立可以用於開發目的。高可用性適用於必須隨時可用的應用程式。
   b. 在資源中，選擇執行個體類型和所需容量。

      這些資源是 AWS Mainframe Modernization 受管的 Amazon EC2 執行個體，會託管您的執行時間環境。獨立執行期環境為執行個體類型提供兩種選擇，並僅允許一個執行個體。高可用性執行期環境為執行個體類型提供兩種選擇，並允許最多兩個執行個體。

      如需詳細資訊，請參閱 Amazon EC2 執行個體類型，並聯絡 AWS 大型主機專家以取得指引。
6. 在安全與網路區段中，執行下列動作：
   a. 如果您希望應用程式可公開存取，請選擇允許部署到此環境的應用程式可公開存取。

b. 選擇網路類型。如果您選擇 IPv4， AWS 大型主機現代化環境應用程式只會提供 IPv4 請求。在雙堆疊模式中，應用程式將同時提供 IPv4 和 IPv6 請求。如果您選擇雙堆疊模式，請確定至少有 1 個 VPC 具有IPv6-enabled的子網路。

c. 選擇 Virtual Private Cloud (VPC)。

d. 如果您使用的是高可用性模式，請選擇兩個或多個子網路。如果您搭配 AWS Blu Age 引擎使用獨立模式，請選擇兩個或多個子網路。如果您將獨立模式與 Rocket 軟體引擎搭配使用，您可以指定一個子網路。

e. 為您選取的 VPC 選擇安全群組。

> **Note**
>
> AWS Mainframe Modernization 會為您建立 Network Load Balancer，以將連線分散至執行期環境。請確定您的安全群組傳入和傳出規則允許從 IP 地址存取您在應用程式定義的 `the section called “Listener(s) - ##”` 屬性中指定的連接埠。如需詳細資訊，請參閱 Network Load Balancer 使用者指南中的更新 Network Load Balancer 的安全群組。

f. 在 KMS 金鑰欄位中，如果您想要使用客戶受管，請選擇自訂加密設定 AWS KMS key。如需詳細資訊，請參閱AWS Mainframe Modernization 服務靜態資料加密。

> **Note**
>
> 根據預設， AWS Mainframe Modernization 會使用 AWS KMS key AWS Mainframe Modernization 擁有並為您管理的 來加密您的資料。不過，您可以選擇使用客戶受管的 AWS KMS key。

g. （選用）AWS KMS key 依名稱或 Amazon Resource Name (ARN) 選擇 。或者，選擇建立 AWS KMS key以前往 AWS KMS 主控台並建立新的 AWS KMS key。

h. 選擇下一步。

7. （選用）在連接儲存頁面上，選擇一或多個 Amazon EFS 或 Amazon FSx 檔案系統。

   掛載至 AWS Mainframe Modernization 環境的檔案系統必須由適當的使用者擁有，以供在 AWS Mainframe Modernization 主控台中執行的應用程式使用。

   若要設定這些使用者設定，您可以將磁碟機連接至 Linux Amazon EC2 執行個體。然後建立 ID 為的群組`101`和 ID 為 的使用者`3001`。此外，請確定應用程式將使用的所需資料資料夾（必須由此使用者擁有）。

例如， AWS 在 Mainframe Modernization Managed 中執行 AWS 的主要主機現代化應用程式可以使用 `myFiles` 資料夾。

```
groupadd -g 101 mygroup
useradd -M -g mygroup -p mypassword -u 3001 myuser
mkdir myFiles
chown myuser:mygroup myFiles
```

> **Note**
>
> 若要啟用檔案系統的存取，應設定下列安全群組規則，以在 EFS 和 M2 環境執行個體之間建立網路連線：
>
> - M2 環境安全群組 – 包含允許透過 NFS 2049 連接埠進行流量的傳出規則。
> - 檔案系統掛載目標安全群組 – 包含允許來自執行個體安全群組 （如上所列） 之 NFS 2049 連接埠流量的傳入規則，以及允許透過 NFS 2049 連接埠流量的傳出規則。

8. 選擇下一步。
9. 在維護時段區段中，選擇您要將待定變更套用至環境的時間。
   - 如果您選擇無偏好設定， AWS Mainframe Modernization 會為您選擇最佳化的維護時段。
   - 若要指定特定的維護時段，請選擇選取新的維護時段。然後選擇一週中的一天、開始時間，以及維護時段的持續時間。

   如需維護時段的詳細資訊，請參閱 AWS Mainframe Modernization 維護時段。

   選擇下一步。
10. 在檢閱和建立頁面上，檢閱您輸入的資訊，然後選擇建立環境。

更新 AWS Mainframe Modernization 執行時間環境

使用 AWS Mainframe Modernization 主控台更新 AWS Mainframe Modernization 執行時間環境。您可以更新執行時間引擎的次要版本，或託管執行時間環境的執行個體類型。您可以選擇是否要立即或在偏好的維護時段期間套用更新。

這些說明假設您已完成設定 AWS 大型主機現代化中的步驟。

更新執行時間環境

更新執行時間環境

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 在 AWS 區域 選擇器中，選擇您要更新的環境建立所在的區域。
3. 在環境頁面上，選擇您要更新的環境。
4. 在環境的詳細資訊頁面上，選擇動作，然後選擇編輯環境。
5. 進行下列任何變更：
   - 在引擎選項區段中，選擇您想要的引擎版本。
   - 在資源區段中，選擇您想要的執行個體類型。
   - 在維護時段區段中，選擇您想要的日期、時間和持續時間。

   > **Note**
   >
   > 您可以選擇在維護時段套用的唯一變更是引擎版本變更。您必須立即套用所有其他變更。

6. 選擇下一步。
7. 在套用這些變更的時機中，選擇立即或在下一個維護時段期間。然後選擇更新環境。

如果您選擇立即，當環境完成更新時，您會看到一則訊息。

AWS Mainframe Modernization 維護時段

每個執行時間環境都有每週兩個小時的維護時段。在此期間會套用任何系統變更。維護時段是您控制何時修改，以及軟體和安全性修補發生的機會。如果維護事件排定在指定週內，則會在兩小時的維護時段內開始。大多數維護事件也會在兩小時的維護時段內完成，但較大的維護事件可能需要幾個小時才能完成。

兩小時維護時段是從每個區域的 8 小時時段隨機選取。如果您在建立執行時間環境時未指定維護時段， AWS 大型主機現代化會在一週中隨機選取一天指派 2 小時的維護時段。

AWS Mainframe Modernization 會在套用維護時，使用您環境執行個體中的一些資源。您可能會在維護期間觀察到對效能或應用程式的某些中斷的影響最小。

停止 AWS 大型主機現代化執行期環境

使用 AWS Mainframe Modernization 主控台停止 AWS Mainframe Modernization 執行時間環境。當您停止環境時，會保留目前的應用程式部署，而且在環境重新啟動之前，不會向您收取該環境的費用。

這些說明假設您已完成設定 AWS 大型主機現代化中的步驟。

停止執行期環境

如果您需要停止 AWS Mainframe Modernization 執行時間環境，請遵循與更新環境區段類似的步驟。

使用 AWS Mainframe Modernization 主控台停止 AWS Mainframe Modernization 執行時間環境。當您停止環境時，會保留目前的應用程式部署，而且在環境重新啟動之前，不會向您收取該環境的費用。

> **Note**
>
> 您必須在停止環境之前停止所有應用程式。

停止執行時間環境

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 在 AWS 區域 選擇器中，選擇您要停止的環境建立所在的區域。
3. 在環境頁面上，選擇您要停止的環境。
4. 在環境的詳細資訊頁面上，選擇動作，然後選擇編輯環境。
5. 在編輯環境頁面上，尋找資源區段，並將所需的容量更新為零。

   > **Note**
   >
   > 若要停止環境，您只能選擇立即停止。

6. 選擇下一步。
7. 在何時套用這些變更中，選擇立即。然後選擇更新環境。

環境容量更新時，您會看到訊息。

重新啟動 AWS Mainframe Modernization 執行時間環境

使用 AWS Mainframe Modernization 主控台重新啟動 AWS Mainframe Modernization 執行時間環境。當您重新啟動執行期環境時，環境的帳單將會繼續。

重新啟動執行時間環境

若要重新啟動 AWS Mainframe Modernization 執行時間環境，請遵循與停止環境區段類似的步驟。

重新啟動執行時間環境

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 在 AWS 區域 選擇器中，選擇您要重新啟動的環境建立所在的區域。
3. 在環境頁面上，選擇您要重新啟動的環境。
4. 在環境的詳細資訊頁面上，選擇動作，然後選擇編輯環境。

> **Note**
>
> 獨立環境所需的容量只能更新為 1。若要重新啟動執行時間環境，您只能選擇立即重新啟動。

5. 在編輯環境頁面上，尋找資源區段，並將所需的容量從零更新為所需的容量。
6. 選擇下一步。
7. 在何時套用這些變更中，選擇立即。然後選擇更新環境。

當環境容量更新且環境重新啟動時，您會看到訊息。

刪除 AWS 大型主機現代化執行時間環境

使用 AWS Mainframe Modernization 主控台刪除 AWS Mainframe Modernization 執行時間環境。

這些說明假設您已完成設定 AWS 大型主機現代化中的步驟。

刪除執行時間環境

如果您需要刪除 AWS Mainframe Modernization 執行時間環境，請務必先從環境刪除任何已部署的應用程式。您無法刪除應用程式部署所在的執行期環境。

刪除環境

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 在 AWS 區域 選擇器中，選擇您要刪除的環境建立所在的區域。
3. 在環境頁面上，選擇您要刪除的環境，然後選擇動作和刪除環境。
4. 在刪除環境視窗中，輸入 `delete`以確認您要刪除執行時間環境，然後選擇刪除。

AWS 大型主機現代化中的應用程式測試

AWS Mainframe Modernization Application Testing 為您的遷移專案提供自動化的功能等效性測試。AWS Mainframe Modernization Application Testing 利用雲端的彈性來加速遷移專案。您可以視需要在任意數量的平行環境中執行獨立測試套件，以減少測試時間表。應用程式測試的主要優點包括測試加速性和敏捷性、高測試可重複性、內建可擴展性和彈性、大規模自動化、成本效益，以及與 的無縫整合， AWS CloudFormation 以建立目標測試環境。

主題

- 什麼是 AWS 大型主機現代化應用程式測試？
- AWS 大型主機現代化應用程式測試概念
- AWS 大型主機現代化應用程式測試先決條件
- 應用程式測試主控台工作流程
- 教學課程：在 AWS 大型主機現代化應用程式測試中設定 CardDemo 範例應用程式
- 教學課程：使用部署在 Amazon EC2 上的 CardDemo for AWS Blu Age，在 AWS 大型主機現代化應用程式測試中重播和比較 Amazon EC2
- AWS Mainframe Modernization Application Testing 支援的資料集程式碼頁面
- AWS 大型主機現代化應用程式測試中的資料保護
- AWS Mainframe Modernization 應用程式測試如何與 IAM 搭配使用

什麼是 AWS 大型主機現代化應用程式測試？

測試會大幅影響現代化專案。 AWS 應用程式測試是 AWS Mainframe Modernization 的一項功能，可為您的遷移應用程式提供自動化功能對等測試。功能對等測試可協助您驗證 上的應用程式 AWS 雲端是否等同於大型主機上的應用程式。 AWS 應用程式測試會自動比較大型主機與 之間的資料集、資料庫記錄和線上 3270 螢幕的變更 AWS。此外，Application Testing 允許重複測試，因此您可以在更新目標架構、解決問題以及向完全遷移的應用程式進展時多次執行測試案例。遷移後，您可以繼續使用應用程式測試進行迴歸測試，以確保對執行期引擎或其他元件的更新不會導致迴歸。應用程式測試具有成本效益：目標測試環境是使用使用者提供的 CloudFormation 範本建立的，利用Infrastructure-as-Code(IaC) 概念。應用程式測試使用雲端的彈性加速遷移專案。您可以視需要在任意數量的平行環境中執行獨立測試套件，以減少測試時間表。

主題

- 您是第一次使用應用程式測試嗎？

- 應用程式測試的優點
- 與 整合 AWS CloudFormation
- 應用程式測試的運作方式
- 相關服務
- 存取應用程式測試
- 應用程式測試的定價

您是第一次使用應用程式測試嗎？

如果您是第一次使用應用程式測試，建議您先閱讀以下章節：

- 應用程式測試概念
- 教學課程：在應用程式測試中設定 CardDemo 應用程式
- the section called “教學課程：使用 CardDemo 重播和比較 AWS 藍時期”

應用程式測試的優點

應用程式測試提供多種優勢，可協助您進行遷移程序：

- 測試加速、敏捷性和靈活性。
- 「在大型主機上錄製一次，在 AWS 中重播多次」測試概念。
- 透過使用者提供的 CloudFormation 範本建立目標環境的 IaC。
- 高度的測試可重複性。
- 專為雲端打造，考量可擴展性和彈性。
- 使用高度自動化進行大規模測試。
- 成本效益。

與 整合 AWS CloudFormation

應用程式測試使用基礎設施做為程式碼 AWS CloudFormation。此設計選擇可簡化和改善您的測試體驗。 AWS CloudFormation 可讓您自主和獨立性，以定義更符合您需求的基礎設施。您可以獨立選取或定義許多參數（執行個體大小、RDS 執行個體、最佳安全群組）。您可以新增資源，例如您在測試條件下應用程式正常運作所需的 Amazon SQS 佇列。

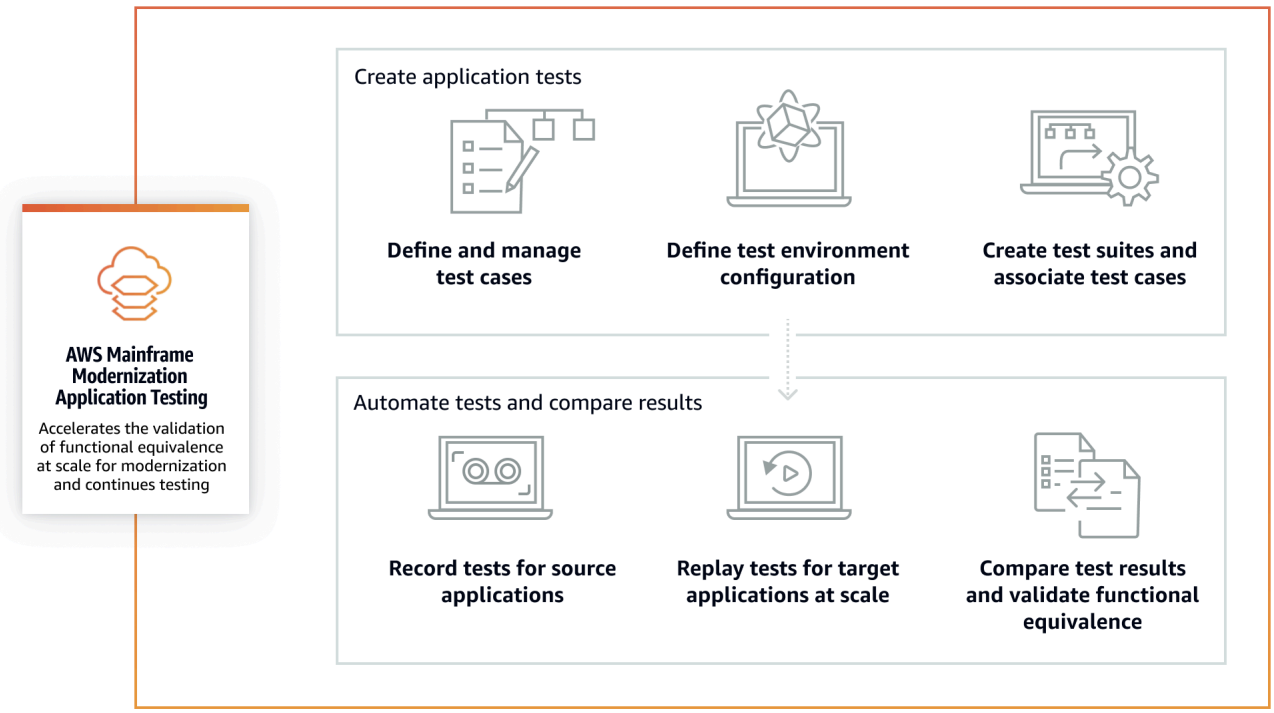
在提供下載的 AWS CloudFormation 範本中，您會注意到一些常見功能：

- Application Testing 會建立完全隔離的堆疊，包括 AWS 大型主機現代化執行期環境和應用程式，以及其自己的網路和安全定義。此隔離堆疊提供彈性，因為相同 中的其他演員 AWS 帳戶 無法干擾測試活動。它也可以避免系統運算子修改預設 VPC 或安全群組的情況，這可能會導致測試活動失敗。
- 安全群組也可讓您控制對測試所用資源的外部存取。例如，資料庫可能包含機密資料。
- 完全隔離可防止共用 VPC 的其他演員窺探流量。
- 它可增強效能。例如，範本建立的 AWS Mainframe Modernization 應用程式與其 Amazon RDS 資料庫之間的通訊會在個別網路（私有 VPC) 上進行，以避免其他執行者降低流量。

我們建議您也在建立的 AWS CloudFormation 範本中實作這些功能。

應用程式測試的運作方式

下圖概述 Application Testing 的運作方式。



- 您可以使用 AWS 檔案傳輸或您偏好的大型主機資料傳輸工具，將輸入資料從來源傳輸到 。
- 您在來源和目標上執行相同的商業邏輯。
- 應用程式測試會自動比較來源和目標的輸出資料（資料集、關聯式資料庫變更、線上 3270 畫面和使用者互動）。在大型主機上執行測試案例後，您可以擷取輸出資料並將其傳輸到 AWS，然後在

目標上重新執行測試案例。應用程式測試會自動比較 上測試執行的輸出資料 AWS 與來源的輸出資料。您可以快速查看哪些記錄相同、同等、不同或遺失。此外，您可以定義同等規則，以便將不相同但具有相同業務意義的記錄理解為同等。

您在應用程式測試中遵循的工作流程包含下列步驟：

1. 建立測試案例：測試案例是測試動作的最小單位。建立測試案例時，您也可以識別要比較的資料類型，最適合代表來源和目標之間的功能相等性。
2. 定義測試環境組態：透過指定 AWS CloudFormation 範本和其他屬性來指定您的環境組態。
3. 建立測試套件：測試套件是測試案例的集合。
4. 在來源上上傳資料集，並在目標上重播：擷取大型主機上的輸入和輸出資料集，並將其上傳至其中 AWS。然後重新播放測試案例 AWS。
5. 比較來源和目標資料集：Application Testing 會自動比較來源和目標的輸出資料集，讓您一目了然地看到什麼是正確的，什麼是錯誤的。

測試案例的最終動作和整個程序的目標都是識別來源和目標測試執行之間的差異。應用程式測試會比較來源版本和目標版本，以在測試執行期間擷取所有互動通道的資料。它也會比較相關資料的最終狀態（如測試案例所定義）。

相關服務

應用程式測試是 AWS 大型主機現代化的一項功能。它還使用基礎設施作為程式碼與 AWS CloudFormation ，以確保測試可重複性、自動化和成本效益。如需詳細資訊，請參閱：

- AWS 大型主機現代化
- AWS CloudFormation

存取應用程式測試

您可以在 AWS 主要主機現代化主控台存取應用程式測試主控台https://console.aws.amazon.com/apptest/，方法是在左側導覽窗格中選擇應用程式測試。

應用程式測試的定價

您可以在 AWS Mainframe Modernization 定價中找到應用程式測試的定價。

AWS 大型主機現代化應用程式測試概念

AWS Application Testing 會使用其他測試服務或軟體套件可能使用的術語，其意義略有不同。下列各節說明 AWS Mainframe Modernization Application Testing 如何使用此術語。

主題

- 測試案例
- 測試套件
- 測試環境組態
- 上傳
- 重新播放
- Compare
- 資料庫比較
- 資料集比較
- 比較狀態
- 對等規則
- 最終狀態資料集比較
- 狀態進展資料庫比較
- 功能對等 (FE)
- 線上 3270 螢幕比較
- 重播資料
- 參考資料
- 上傳、重播和比較
- 差異
- 對等
- 來源應用程式
- 目標應用程式

測試案例

測試案例是測試工作流程中最原子的個別動作單位。通常，測試案例用於代表修改資料的獨立商業邏輯單位。每個測試案例都會進行比較。測試案例會新增至測試套件。測試案例包含測試案例修改的資料成

品（資料集、資料庫）中繼資料，以及測試案例執行期間觸發的業務功能中繼資料：批次任務、3270 個互動式對話方塊等。例如，資料集的名稱和程式碼頁面。

輸入資料 → 測試案例 → 輸出資料

測試案例可以是線上或批次類型：

- 線上 3270 螢幕測試案例是使用者執行互動式螢幕對話方塊 (3270) 以讀取、修改或產生新業務資料（資料庫和/或資料集記錄）的測試案例。
- 批次測試案例是需要提交批次以讀取、處理和修改或產生新業務資料（資料集和/或資料庫記錄）的測試案例。

測試套件

測試套件具有一系列測試案例，這些測試案例會依序執行。重播會在測試套件層級完成。重新播放測試套件時，測試套件中的所有測試案例都會在目標測試環境中執行。如果比較參考和重播測試成品後有差異，則差異會顯示在測試案例層級。

例如，Test Suite A：

測試案例 1、測試案例 2、測試案例 3 等。

測試環境組態

測試環境組態可讓您使用 CloudFormation 設定初始資料集和組態參數（或資源），以讓測試執行可重複執行。

上傳

上傳是在測試套件層級完成。在上傳期間，您必須提供 Amazon S3 位置，其中包含要與之比較的來源大型主機中的成品、資料集和關聯式資料庫 CDC 日誌。這些會被視為來源大型主機的參考資料。在重播期間，產生的重播資料將與上傳的參考資料進行比較，以確保應用程式對等性。

重新播放

重播會在測試套件層級完成。在重播期間， AWS 大型主機現代化應用程式測試會使用 CloudFormation 指令碼來建立目標測試環境並執行應用程式。系統會擷取在重播期間修改的資料集和資料庫記錄，並與大型主機的參考資料進行比較。一般而言，您會在大型主機上上傳一次，然後重新播放多次，直到達到功能等效為止。

Compare

重播成功完成後會自動進行比較。在比較期間，您在上傳階段上傳和擷取的參考資料會與重播階段產生的重播資料進行比較。比較會分別在資料集、資料庫記錄和線上畫面的個別測試案例層級進行。

資料庫比較

應用程式測試在比較來源和目標應用程式之間的資料庫記錄變更時，會採用狀態進度比對功能。狀態進度比對會比較每個個別執行 INSERT、UPDATE 和 DELETE 陳述式的差異，與程序結束時比較資料表列不同。狀態進度比對比替代方案更有效率，只需比較變更的資料和偵測交易流程中的自我校正錯誤，即可提供更快且更準確的比較。透過使用 CDC （變更的資料擷取） 技術，Application Testing 可以偵測個別關聯式資料庫變更，並在來源和目標之間進行比較。

關聯式資料庫變更是由測試的應用程式程式碼使用 DML （資料修改語言） 陳述式在來源和目標上產生，例如 SQL INSERT、UPDATE 或 DELETE，但也會在應用程式使用預存程序時間接產生，或在某些資料表上設定資料庫觸發時，或使用 CASCADE DELETE 來保證參考完整性時，自動觸發額外的刪除。

資料集比較

應用程式測試會自動比較來源 （錄製） 和目標重播） 系統上產生的參考和重播資料集。

若要比較資料集：

1. 從來源和目標上的相同輸入資料 （資料集、資料庫） 開始。
2. 在來源系統 （大型主機） 上執行您的測試案例。
3. 擷取產生的資料集，並將其上傳至 Amazon S3 儲存貯體。您可以使用 CDC 日誌、畫面和資料集，將輸入資料集從來源傳輸到 AWS 。
4. 指定上傳測試案例時，大型主機資料集上傳的 Amazon S3 儲存貯體位置。

重播完成後，Application Testing 會自動比較輸出參考和目標資料集，顯示記錄是否相同、同等、不同或遺失。例如，相對於工作負載執行時間 （日 + 1、當月結束等） 的日期欄位會自動視為同等欄位。此外，您可以選擇定義等效規則，讓不相同的記錄仍然具有相同的業務意義，並標記為等效。

比較狀態

應用程式測試使用以下比較狀態：IDENTICAL、EQUIVALENT 和 DIFFERENT。

相同

來源和目標資料完全相同。

等式

來源和目標資料包含被視為對等的錯誤差異，例如日期或時間戳記，當它們相對於工作負載執行的時刻不會影響功能對等。您可以定義等效規則，以識別這些差異是什麼。當所有重播的測試套件與其參考測試套件顯示 IDENTICAL 或 EQUIVALENT 狀態時，您的測試套件不會顯示任何差異。

不同

來源和目標資料包含差異，例如資料集中的不同記錄數，或相同記錄中的不同值。

對等規則

一組規則，用於識別可視為同等結果的錯誤差異。離線功能相等性測試 (OFET) 必然會導致來源和目標系統之間的某些結果差異。例如，更新時間戳記因設計而異。對等規則說明如何針對這些差異進行調整，並避免比較時的誤報。例如，如果日期是特定資料欄中的執行時間 + 2 天，則對等規則會加以描述，並接受目標系統上的執行時間 + 2 天，而不是在參考上傳中嚴格等於相同資料欄的值。

最終狀態資料集比較

已建立或修改之資料集的結束狀態，包括從其初始狀態對資料集進行的所有變更或更新。對於資料集，Application Testing 會在測試案例執行結束時查看這些資料集中的記錄，並比較結果。

狀態進展資料庫比較

將對資料庫記錄所做的變更做為個別 DML （刪除、更新、插入） 陳述式的序列進行比較。應用程式測試會將來源資料庫的個別變更 （插入、更新或刪除資料表的資料列） 與目標資料庫進行比較，並識別每個個別變更的差異。例如，個別 INSERT 陳述式可用於在資料表中插入來源資料庫與目標資料庫具有不同值的資料列。

功能對等 (FE)

如果兩個系統在所有可觀測的操作上產生相同的結果，則考慮到相同的輸入資料，則兩個系統在功能上被視為相等。例如，如果相同的輸入資料產生相同的輸出資料 （透過畫面、資料集變更或資料庫變更），則兩個應用程式在功能上視為相等。

線上 3270 螢幕比較

在 中的 AWS Blu Age 執行時間下執行目標系統時，比較大型主機 3270 畫面的輸出與現代化應用程式 Web 畫面的輸出 AWS 雲端。當目標系統在 中的 Rocket Software （ 先前稱為 Micro Focus) 執行時間下執行時，它會比較大型主機 3270 畫面的輸出與重新託管應用程式的 3270 畫面 AWS 雲端。

重播資料

重播資料用於描述在目標測試環境中重播測試套件所產生的資料。例如，當測試套件在 AWS Mainframe Modernization 服務應用程式上執行時，會產生重播資料。然後，將重播資料與從來源擷取的參考資料進行比較。每次在目標環境中重播工作負載時，都會產生新一代的重播資料。

參考資料

參考資料用於描述在來源大型主機上擷取的資料。這是將比較重播 （ 目標 ） 產生資料的參考。通常，對於建立參考資料的大型主機上的每個記錄，將會有許多重播。這是因為使用者通常會在大型主機上擷取應用程式的正確狀態，並在目標現代化應用程式上重播測試案例，以驗證等效性。如果發現錯誤，則會修正錯誤，並再次重播測試案例。通常，多次重播、修正錯誤，以及再次重播以驗證發生的情況。這稱為擷取一次，重新執行多次測試範例。

上傳、重播和比較

應用程式測試以三個步驟運作：

- 上傳： 擷取針對測試案例的每個測試案例在大型主機上建立的參考資料。這些可以包含 3270 個線上畫面、資料集和資料庫記錄。
  - 對於線上 3270 畫面，您必須使用 Blu Insights 終端機模擬器來擷取來源工作負載。如需詳細資訊，請參閱 Blu Insights 文件。
  - 對於資料集，您將需要使用常用工具擷取大型主機上每個測試案例產生的資料集，例如 FTP 或 AWS 大型主機現代化的資料集傳輸服務部分。
  - 對於資料庫變更，您可以使用 AWS Mainframe Modernization Data Replication with Precisely 文件來擷取和產生包含變更的 CDC 日誌。
- 重播：測試套件會在目標環境中重播。測試套件執行中指定的所有測試案例。個別測試案例所建立的指定資料類型，例如資料集、關聯式資料庫變更或 3270 個畫面，將以自動化方式擷取。這些資料稱為重播資料，將與上傳階段擷取的參考資料進行比較。

> **Note**
>
> 關聯式資料庫變更將需要初始條件 CloudFormation 範本中的 DMS 特定組態選項。

- 比較：比較來源測試參考資料和目標重播資料，結果會以相同、不同、同等或遺失的資料顯示給您。

差異

表示已透過資料比較偵測到參考和重播資料集之間的差異。例如，線上 3270 畫面中顯示與來源大型主機和目標現代化應用程式之間業務邏輯觀點不同值的欄位，將被視為差異。另一個範例是在來源和目標應用程式之間不同資料集中的上傳。

對等

對等記錄是參考和重播資料集之間的記錄，但不應被視為與業務邏輯觀點不同。例如，包含產生資料集之時間戳記的記錄（工作負載執行時間）。使用可自訂的等效規則，您可以指示 Application Testing 將這類誤報差異視為等效，即使其在參考和重播資料之間顯示不同的值。

來源應用程式

要比較的來源大型主機應用程式。

目標應用程式

完成測試的新應用程式或修改的應用程式，並將與來源應用程式進行比較，以偵測任何瑕疵，並實現來源和目標應用程式之間的功能相等性。目標應用程式通常會在 AWS 雲端中執行。

AWS 大型主機現代化應用程式測試先決條件

AWS Mainframe Modernization 中的 AWS Mainframe Modernization Application Testing 功能可讓您針對遷移專案執行自動化功能對等測試。若要準備在 AWS Mainframe Modernization 主控台中使用應用程式測試，請執行下列動作：

1. 定義測試案例：為目標應用程式定義您想要以特定順序執行和重播的基本測試單位。如需如何建立測試案例的詳細資訊，請參閱 the section called “在應用程式測試中建立測試案例”。
2. 準備 CloudFormation 範本和輸入資料：建立將用於佈建目標測試環境的 CloudFormation 範本。此範本的變數將用於在 AWS Mainframe Modernization 應用程式中新增輸入資料和輸出變數名稱。如需詳細資訊，請參閱AWS CloudFormation 《使用者指南》中的使用 AWS CloudFormation 範本。

3. 確保大型主機存取和資料擷取：確認您可存取來源大型主機。這也會確保您能夠擷取和上傳在大型主機上執行的應用程式所產生的來源資料。

應用程式測試主控台工作流程

AWS Mainframe Modernization Application Testing 主控台可協助您建立測試案例、測試套件和測試環境組態。

主題

- 在 AWS 大型主機現代化應用程式測試中建立測試案例
- 在 AWS 大型主機現代化應用程式測試中建立測試套件
- 在 AWS 大型主機現代化應用程式測試中建立測試環境組態

在 AWS 大型主機現代化應用程式測試中建立測試案例

測試案例是一個原子單位，代表工作流程中的特定動作。如需各種概念的其他資訊，請參閱 ???。

> **Important**
>
> 您必須先建立至少一個測試環境組態，才能執行測試案例。若要建立您的第一個環境組態，請參閱 the section called “在應用程式測試中建立測試環境組態”。

主題

- 建立批次測試案例
- 建立線上 3270 螢幕測試案例

建立批次測試案例

批次測試案例可讓您提交批次，以讀取、處理和修改或產生新的業務資料 （資料庫和/或資料集記錄）。

建立批次測試案例

1. 在 開啟 AWS 大型主機現代化應用程式測試主控台https://console.aws.amazon.com/apptest/。
2. 在 AWS 區域 選擇器中，選擇可使用應用程式測試的區域。

> **Note**
>
> 應用程式測試目前僅適用於美國東部（維吉尼亞北部）、亞太區域（雪梨）、歐洲（法蘭克福）和南美洲（聖保羅）區域。

3. 在左側導覽窗格中，選取測試案例。
4. 在定義測試案例中，輸入您的測試案例名稱和選用描述。選擇測試案例類型下的批次。
5. 選擇下一步。
6. （選用）在指定批次 JCL 參數頁面上，新增 JCL（任務控制語言）名稱和任務參數（名稱和值）。
7. 選擇下一步。
8. 在要擷取的資料來源頁面上，您可以選擇關聯式資料庫變更、資料集或兩者。
   - 當您希望測試案例修改資料庫記錄時，請選擇關聯式資料庫變更。
   - 當您希望測試案例修改資料集時，請選擇資料集。在輸出資料集下，新增輸出資料集的名稱。

   > **Note**
   >
   > 您可以新增多個資料集。
9. 選擇下一步。
10. 在檢閱和建立頁面上，檢閱所有資訊，然後選擇建立測試案例。

建立線上 3270 螢幕測試案例

線上 3270 螢幕測試案例可讓您執行互動式螢幕對話方塊 (3270)，以讀取、修改或產生新的商業資料（資料庫和/或資料集記錄）。

建立線上 3270 螢幕測試案例

1. 在 開啟 AWS 大型主機現代化應用程式測試主控台https://console.aws.amazon.com/apptest/。
2. 在 AWS 區域 選擇器中，選擇可使用應用程式測試的區域。

> **Note**
>
> 應用程式測試目前僅適用於美國東部（維吉尼亞北部）、亞太區域（雪梨）、歐洲（法蘭克福）和南美洲（聖保羅）區域。

3. 在左側導覽窗格中，選取測試案例。
4. 在定義測試案例中，輸入您的測試案例名稱和選用描述。選擇測試案例類型下的線上 3270 畫面。
5. 選擇下一步。

> **Note**
>
> 線上 3270 畫面不需要您指定 JCL 參數。

6. 選擇下一步。
7. 在要擷取的資料來源頁面上，預設選項為線上 3270 畫面。此外，您可以選擇關聯式資料庫變更和資料集。
   - 當您希望測試案例修改資料庫記錄時，請選擇關聯式資料庫變更。
   - 當您希望測試案例修改資料集時，請選擇資料集。在輸出資料集下，新增輸出資料集的名稱。

> **Note**
>
> 您可以新增多個資料集。

8. 選擇下一步。
9. 在檢閱和建立頁面上，檢閱所有資訊，然後選擇建立測試案例。

在 AWS 大型主機現代化應用程式測試中建立測試套件

測試套件是一系列按順序執行的測試案例。測試套件對於重播測試案例很重要。

> **Important**
>
> 建立測試套件之前，您至少需要有一個測試案例。您可以使用 建立您的第一個測試案例the section called “在應用程式測試中建立測試案例”。

如需各種概念的其他資訊，請參閱 the section called “應用程式測試概念”。

主題

- 建立測試套件
- 上傳參考資料
- 重播和比較

建立測試套件

測試套件可讓您執行不同的測試案例，並在稍後重播和比較這些案例。

建立測試套件

1. 在 開啟 AWS 大型主機現代化應用程式測試主控台https://console.aws.amazon.com/apptest/。
2. 在 AWS 區域 選擇器中，選擇可使用應用程式測試的區域。

   > **Note**
   >
   > 應用程式測試目前僅適用於美國東部（維吉尼亞北部）、亞太區域（雪梨）、歐洲（法蘭克福）和南美洲（聖保羅）區域。

3. 在左側導覽窗格中，選取測試案例。
4. 選擇建立測試套件。
5. 在建立測試套件區段中，從測試案例庫尋找測試案例，然後選擇新增選取的測試案例。

   > **Note**
   >
   > 您可以在一個測試套件中新增最多 20 個測試案例。

6. 在測試套件窗格中，輸入您的測試套件名稱和選用描述。此外，從受管執行時間或非受管執行時間中選取 ，這會定義測試套件如何設定和取消設定 AWS Mainframe Modernization 應用程式。或者，新增 AWS 大型主機現代化匯入資料集 JSON S3 URI。
7. 在新增的測試案例區段中，依您要上傳和重播的順序堆疊您的測試案例。
8. 選擇 Create test suite (建立測試套件)。

上傳參考資料

將大型主機參考資料上傳至 AWS 應用程式測試。您只需要第一次儲存上傳的參考資料。測試服務可以重複使用來源上傳的結果，並將其與目標上重播的結果連續進行比較。

上傳參考資料

1. 從測試套件區段中，選擇要上傳參考資料的測試套件。
2. 選擇上傳。
3. 在上傳參考資料頁面上，選取要重播的測試案例。完成資料擷取日期、資料庫變更日誌 S3 位置、資料集 S3 位置的欄位，然後選擇上傳。

重播和比較

重播和比較程序會將您的測試案例與目標測試環境建立關聯，並執行應用程式。您需要先上傳資料，才能執行重播程序。

重播和比較

1. 從測試套件區段中，選擇要重播的測試套件。
2. 選擇重播並比較。
3. 在重播和比較概觀頁面上，選取您的測試環境組態並檢閱資訊。編輯 函數可讓您編輯任何測試環境組態欄位。您也可以找到 AWS CloudFormation 參數。
4. 在要重播的測試案例區段下，選擇測試案例，並依照您要重播的順序放置它們。
5. 選擇重播並比較。

在 AWS 大型主機現代化應用程式測試中建立測試環境組態

測試環境組態可讓您使用 設定初始資料集和組態參數 （或資源） AWS CloudFormation ，讓測試執行可重複執行。

如需各種概念的其他資訊，請參閱 the section called “應用程式測試概念”。

建立測試環境組態

設定您的測試環境，以在應用程式測試中重播和比較測試案例。

設定測試環境組態

1. 在 開啟 AWS 大型主機現代化應用程式測試主控台https://console.aws.amazon.com/apptest/。
2. 在 AWS 區域 選擇器中，選擇可使用應用程式測試的區域。

> **Note**
>
> 應用程式測試目前僅適用於美國東部（維吉尼亞北部）、亞太區域（雪梨）、歐洲（法蘭克福）和南美洲（聖保羅）區域。

3. 在左側導覽窗格中，選取測試環境組態 。
4. 選擇建立測試環境組態。
5. 在建立測試環境組態窗格中，輸入名稱和描述。同時新增包含用於應用程式測試的 CloudFormation 範本的 Amazon S3 儲存貯體。此外，您可以新增將在 CloudFormation 堆疊建立期間使用的 CloudFormation 輸入參數。
6. 指定將受此測試組態影響的 AWS Mainframe Modernization 應用程式。新增 AWS Mainframe Modernization 應用程式 ID、執行時間引擎 (AWS Blu Age 非受管或 Rocket Software（先前稱為 Micro Focus) 受管）的輸出變數名稱。

> **Note**
>
> AWS Mainframe Modernization 應用程式 ID 的輸出變數名稱應與 CloudFormation 範本的輸出變數名稱相符，以便建立堆疊。

> **Important**
>
> AWS Blu Age 非受管執行期也會要求您指定 VPC 端點服務 ID 的輸出變數名稱、接聽程式連接埠的輸出變數名稱，以及 WebApp 名稱的輸出變數名稱。這些名稱應與 CloudFormation 範本中的輸出變數名稱相符。

7. （選用）可以為 Database Migration Service (DMS) 任務 Amazon Resource Name (ARN) 定義輸出變數名稱等其他屬性，用於擷取關聯式資料庫變更。另一個屬性是來源資料庫 DDL S3 URI。

> **Important**
>
> 輸出變數名稱應與 CloudFormation 範本中的變數名稱相符。

8. （選用）自訂您的 Key Management Service (KMS) 金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的管理客戶受管金鑰的存取。
9. 選擇建立測試環境組態。

教學課程：在 AWS 大型主機現代化應用程式測試中設定 CardDemo 範例應用程式

在本教學課程中，您會建立 AWS CloudFormation 堆疊，協助您設定 CardDemo 範例應用程式，以便使用 Micro Focus on AWS Mainframe Modernization 受管服務進行複寫，以及包括 AWS Mainframe Modernization Application Testing 在內的功能。本教學課程說明可用來建立堆疊的範例 AWS CloudFormation 範本。我們也提供必要應用程式成品的壓縮檔案。範例範本會佈建資料庫、執行時間環境、應用程式和完全隔離的網路環境。

此範本會建立數個 AWS 資源。如果您從此範本建立堆疊，則會向您收取這些費用。

先決條件

- 下載並解壓縮 IC3-card-demo-zip和 datasets_Mainframe_ebcdic.zip。這些檔案包含 CardDemo 範例和範例資料集，可用於 AWS 應用程式測試。
- 建立 Amazon S3 儲存貯體以保留 CardDemo 檔案和其他成品。例如 `my-carddemo-bucket`。

步驟 1：準備設定 CardDemo

上傳 CardDemo 範例檔案，並編輯將建立 CardDemo 應用程式的 AWS CloudFormation 範本。

1. 將先前解壓縮的 `datasets_Mainframe_ebcdic`和 `IC3-card-demo` 資料夾上傳到儲存貯體。
2. 從儲存貯體下載`aws-m2-math-mf-carddemo.yaml` AWS CloudFormation 範本。它位於 `IC3-card-demo` 資料夾。
3. 編輯`aws-m2-math-mf-carddemo.yaml` AWS CloudFormation 範本，如下所示：
    - 將 `BucketName` 參數變更為您先前定義的儲存貯體名稱，例如 `my-carddemo-bucket`。
    - 將 `ImportJsonPath`變更為 `mf-carddemo-datasets-import.json` 檔案儲存貯體中的位置。例如，`s3://my-carddemo-bucket/IC3-card-demo/mf-carddemo-datasets-import.json`更新此值可確保輸出`M2ImportJson`具有正確的值。
    - （選用）調整 `EngineVersion`和 `InstanceType` 參數以符合您的標準。

> **Note**
>
> 請勿修改 `M2EnvironmentId`和 `M2ApplicationId`輸出。應用程式測試使用這些值來尋找將與其互動的資源。

步驟 2：建立所有必要的資源

執行自訂 AWS CloudFormation 範本，以建立成功完成本教學課程所需的所有資源。此範本會設定 CardDemo 應用程式，讓您可以在測試時使用它。

1. 登入 AWS CloudFormation 主控台，然後選擇建立堆疊，然後選擇使用新資源 （標準）。
2. 在先決條件 - 準備範本中，選擇範本已就緒。
3. 在指定範本中，選擇上傳範本檔案，然後選擇選擇檔案。
4. 導覽至您下載的位置`aws-m2-math-mf-carddemo.yaml`並選擇該檔案，然後選擇下一步。
5. 在指定堆疊詳細資訊中，提供堆疊的名稱，讓您可以輕鬆地在清單中找到堆疊，然後選擇下一步。
6. 在設定堆疊選項中，保留預設值，然後選擇下一步。
7. 在檢閱中，檢查 AWS CloudFormation 為您建立的內容，然後選擇提交。

AWS CloudFormation 建立堆疊大約需要 10-15 分鐘。

> **Note**
>
> 範本設定為將唯一的尾碼附加至其建立的資源名稱。這表示您可以平行建立此堆疊範本的多個執行個體，這是 Application Testing 的關鍵功能，可讓您同時執行多個測試套件。

步驟 3：部署並啟動應用程式

部署為您 AWS CloudFormation 建立的 CardDemo 應用程式，並確認其正在執行。

1. 開啟 AWS Mainframe Modernization 主控台，然後從左側導覽中選擇應用程式。
2. 選擇 CardDemo 應用程式，它命名為類似 。 `aws-m2-math-mf-carddemo-abc1d2e3`
3. 選擇動作，然後選擇部署應用程式。

4. 在環境中，選擇與應用程式對應的執行時間環境。名稱結尾將附加相同的唯一識別符。例如 `aws-m2-math-mf-carddemo-abc1d2e3`。
5. 選擇部署。等待應用程式成功部署並處於就緒狀態。
6. 選擇應用程式，然後選擇動作和啟動應用程式。等到應用程式處於執行中狀態。
7. 在應用程式詳細資訊頁面中，複製您需要的連接埠和 DNS 主機名稱，以連線至執行中的應用程式。

步驟 4：匯入初始資料

若要使用 CardDemo 範例應用程式，您必須匯入一組初始資料。完成下列步驟。

1. 下載 `mf-carddemo-datasets-import.json` 檔案。
2. 在您偏好的文字編輯器中編輯 檔案。
3. 找到 `s3Location` 參數並更新 值，以指向您建立的 Amazon S3 儲存貯體。
4. 針對所有出現的 進行相同的變更`s3Location`，然後儲存檔案。
5. 登入 Amazon S3 主控台並導覽至您先前建立的儲存貯體。
6. 上傳自訂`mf-carddemo-datasets-import.json`檔案。
7. 開啟 AWS Mainframe Modernization 主控台，然後從左側導覽中選擇應用程式。
8. 選擇 CardDemo 應用程式。
9. 選擇資料集，然後選擇匯入。
10. 導覽至 Amazon S3 中您上傳自訂 JSON 檔案的位置，然後選擇提交。

此任務會匯入 23 個資料集。若要監控匯入任務的結果，請檢查 主控台。成功匯入所有資料集時，請連線至應用程式。

> **Note**
>
> 當您在應用程式測試中使用此範本時，輸出`M2ImportJson`會自動處理匯入程序。

步驟 5：連線至 CardDemo 應用程式

使用您選擇的 3270 模擬器連線至 CardDemo 範例應用程式。

- 當應用程式執行時，請使用 3270 模擬器連線到應用程式，並視需要指定 DNS 主機名稱和連接埠名稱。

  例如，如果您使用的是開放原始碼 c3270 模擬器，您的命令如下所示：

  ```
  c3270 -port port-number DNS-hostname
  ```

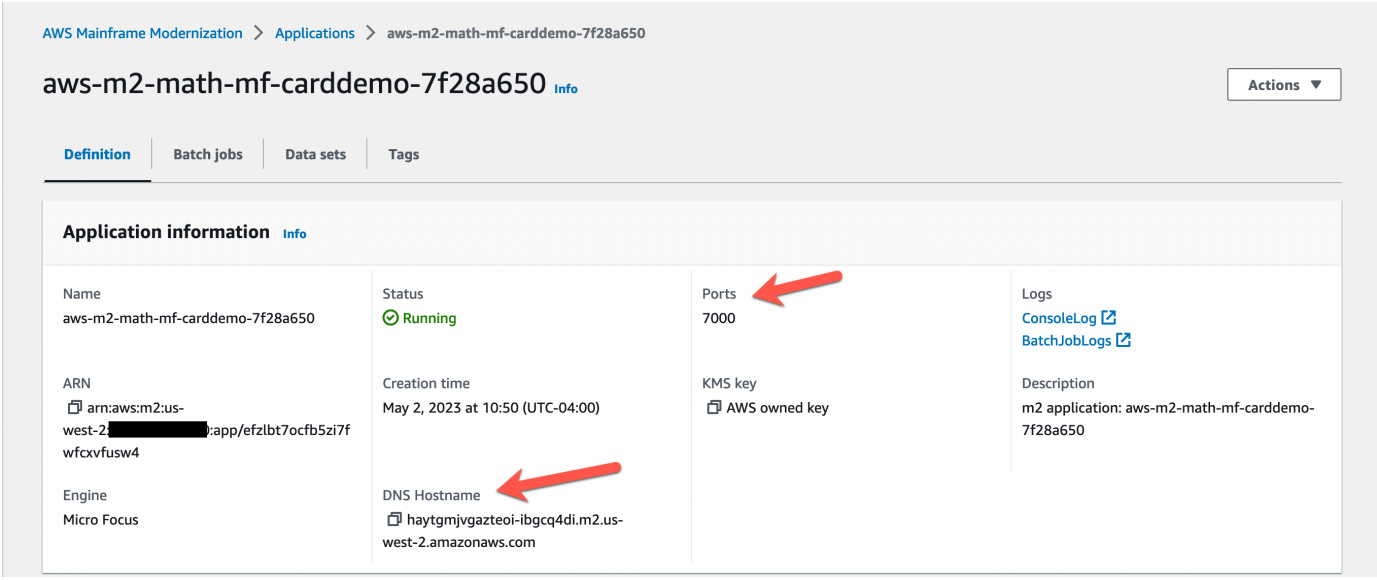
  port

  應用程式詳細資訊頁面上指定的連接埠。例如，6000。

  Hostname (主機名稱)

  應用程式詳細資訊頁面上指定的 DNS 主機名稱。

  下圖顯示在何處尋找連接埠和 DSN 主機名稱。



教學課程：使用部署在 Amazon EC2 上的 CardDemo for AWS Blu Age，在 AWS 大型主機現代化應用程式測試中重播和比較 Amazon EC2

在本教學課程中，您將完成必要步驟，以重新執行測試工作負載，並將其與在 Amazon EC2 上部署的在 AWS Blu Age 上執行的 CardDemo 應用程式進行比較。

步驟 1：取得 AWS 藍時 Amazon EC2 Amazon Machine Image (AMI)

請遵循 AWS Blu Age Runtime （在 Amazon EC2 上）設定教學中的指示，了解存取 Amazon EC2 AMI 上的 AWS Blu Age 所需的入門步驟。

步驟 2：使用 AWS Blu Age AMI 啟動 Amazon EC2 執行個體

1. 設定您的 AWS 登入資料。
2. 從 Amazon S3 儲存貯體識別 3.5.0 Amazon S3 EC2 AMI 二進位檔案 （僅限 CLI/AWS Blu Age 版本）的位置： Amazon EC2

```
aws s3 ls s3://aws-bluage-runtime-artifacts-xxxxxxx-eu-west-1/
aws s3 ls s3://aws-bluage-runtime-artifacts-xxxxxxx-eu-west-1/3.5.0/AMI/
```

> **Note**
>
> 應用程式測試功能僅適用於 prod 中的 4 個區域 (us-east-1、sa-east-1、eu-central-1 和 ap-southeast-2)。

3. 使用下列命令還原您帳戶中的 AMI：

```
aws ec2 create-restore-image-task --object-key 3.5.0/AMI/ami-0182ffe3b9d63925b.bin
 --bucket aws-bluage-runtime-artifacts-xxxxxxx-eu-west-1 --region eu-west-1 --name
 "AWS BLUAGE RUNTIME AMI"
```

> **Note**
>
> 取代 AMI bin 檔案名稱和您要建立 AMI 的區域。

4. 建立 Amazon EC2 執行個體後，您可以在 Amazon EC2 映像目錄中找到從 Amazon S3 儲存貯體還原 AMI 的正確 AMI ID。

> **Note**
>
> 在本教學課程中，AMI ID 為 ami-0d0fafcc636fd1e6d，您必須將此不同組態檔案中的 ID 變更為提供給您的組態檔案。

1. 如果 aws ec2 create-restore-image-task 失敗，請使用下列命令檢查您的 Python 和 CLI 版本：

```
aws --version
```

> **Note**
>
> Python 版本必須 >= 3，而 CLI 版本必須 >= 2。

2. 如果這些版本已淘汰，則必須更新 CLI。若要更新 CLI：
   a. 請遵循安裝或更新最新版本的 AWS CLI 中的指示。
   b. 使用下列命令移除 CLI v1：

```
sudo yum remove awscli
```

   c. 並使用下列命令安裝 CLI v2：

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
 "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

   d. 最後，使用以下命令檢查 Python 和 CLI 的版本：

```
aws --version
```

3. 然後，您可以重做 aws ec2 create-restore-image-task。

步驟 3：將 CardDemo 相依檔案上傳至 S3

複製資料夾資料庫、檔案系統和使用者資料的內容。下載並解壓縮 CardDemo 應用程式。這三個資料夾必須複製到本文件中名為 amzn-s3-demo-bucket 的其中一個儲存貯體。

步驟 4：載入資料庫並初始化 CardDemo 應用程式

建立臨時 Amazon EC2 執行個體，您將用作計算資源，以產生 CardDemo 應用程式所需的資料庫快照。此 EC2 執行個體不會執行 CardDemo 應用程式本身，而是產生稍後將使用的資料庫快照。

首先編輯提供的 CloudFormation 範本，名為 'load-and-create-ba-snapshots.yml'。這是 CloudFormation 範本，用於建立用於產生資料庫快照的 Amazon EC2 執行個體。

1. 產生並提供將用於 EC2 執行個體的 EC2 金鑰對。如需詳細資訊，請參閱建立金鑰對。

   範例：

   ```
   Ec2KeyPair:
       Description: 'ec2 key pair'
       Default: 'm2-tests-us-west-2'
       Type: String
   ```

2. 指定資料夾的 Amazon S3 路徑，其中您已放置上一個步驟的資料庫資料夾：

   ```
   S3DBScriptsPath:
       Description: 'S3 DB scripts folder path'
       Type: String
       Default: 's3://amzn-s3-demo-bucket/databases'
   ```

3. 指定資料夾的 Amazon S3 路徑，其中您已放置上一個步驟的檔案系統資料夾：

   ```
   S3ApplicationFilesPath:
       Description: 'S3 application files folder path'
       Type: String
       Default: 's3://amzn-s3-demo-bucket/file-system'
   ```

4. 指定資料夾的 Amazon S3 路徑，其中您已放置上一個步驟的 userdata 資料夾：

   ```
   S3UserDataPath:
       Description: 'S3 userdata folder path'
       Type: String
   ```

```
    Default: 's3://amzn-s3-demo-bucket/userdata'
```

5. 另請指定 Amazon S3 路徑，您將在其中儲存結果檔案，以便在下一個步驟中使用。

```
S3SaveProducedFilesPath:
    Description: 'S3 path folder to save produced files'
    Type: String
    Default: 's3://amzn-s3-demo-bucket/post-produced-files'
```

6. 使用下列範本，以本教學課程稍早取得的正確 AMI ID 來變更 AMI ID：

```
  BaaAmiId:
    Description: 'ami id (AL2) for ba anywhere'
    Default: 'ami-0bd41245734fd20d9'
    Type: String
```

- 您可以選擇性地變更透過 CloudFormation 執行載入資料庫所建立的三個快照的名稱。這些項目會在 CloudFormation 堆疊建立時於其中顯示，並在本教學課程中稍後使用。請記得記下用於資料庫快照的名稱。

```
SnapshotPrimary:
    Description: 'Snapshot Name DB BA Primary'
    Type: String
    Default: 'snapshot-primary'

  SnapshotBluesam:
    Description: 'Snapshot Name DB BA Bluesam'
    Type: String
    Default: 'snapshot-bluesam'

  SnapshotJics:
    Description: 'Snapshot Name DB BA Jics'
    Type: String
    Default: 'snapshot-jics'
```

> **Note**
>
> 在本文件中，我們假設快照的名稱保持不變。

7. 使用建立堆疊按鈕和精靈，透過 CLI 或 AWS 主控台執行 CloudFormation。在程序結束時，您應該會在 RDS 主控台中看到三個快照，其中包含您選擇的名稱，後面接著唯一 ID。在下一個步驟中，您將需要這些名稱。

> **Note**
>
> RDS 會將字首新增至 AWS CloudFormation 範本中定義的快照名稱。請務必先從 RDS 取得完整快照名稱，再繼續進行下一個步驟。

CLI 命令範例-

```
aws cloudformation create-stack --stack-name load-and-create-ba-snapshots --
template-url https://your-apptest-bucket.s3.us-west-2.amazonaws.com/load-and-
create-ba-snapshots.yml --capabilities CAPABILITY_NAMED_IAM
```

您也可以在為 Amazon S3 S3SaveProducedFilesPath 路徑中檢查資料集是否已正確建立。

步驟 5：啟動 AWS Blu Age 執行期 CloudFormation

使用 CloudFormation 搭配 CardDemo AWS Blu Age 應用程式執行 Amazon EC2 執行個體。您必須在啟動 CFN 期間編輯 YAML 檔案或修改主控台中的值，`m2-with-ba-using-snapshots-https-authentication.yml`以取代 CloudFormation 中名為 的部分變數。

1. 使用下列命令，修改 `AllowedVpcEndpointPrincipals` 以指定哪個帳戶將到達 VPC 端點以存取 AWS Blu Age 執行時間：

```
AllowedVpcEndpointPrincipals:
    Description: 'comma-separated list of IAM users, IAM roles, or AWS accounts'
    Default: 'apptest.amazonaws.com'
    Type: String
```

2. 將 SnapshotPrimaryDb、SnapshotBlusamDb 和 SnapshotJicsDb 變數的值變更為快照的名稱。另外，在上一個步驟中建立快照名稱之後，從 RDS 取得快照名稱。

```
SnapshotPrimary:
    Description: 'Snapshot DB cluster for DB Primary'
    Type: String
    Default: 'snapshot-primary87d067b0'
```

```
  SnapshotBluesam:
    Description: 'Snapshot DB cluster for DB Bluesam'
    Type: String
    Default: 'snapshot-bluesam87d067b0'

  SnapshotJics:
    Description: 'Snapshot DB cluster for DB Jics'
    Type: String
    Default: 'snapshot-jics87d067b0'
```

> **Note**
>
> RDS 會將自己的字首新增至快照名稱。

3. 使用此命令為 Amazon EC2 EC2 金鑰對：

```
Ec2KeyPair:
    Description: 'ec2 key pair'
    Default: 'm2-tests-us-west-2'
    Type: String
```

4. 提供您在 AMI 註冊程序期間為變數 BaaAmiId 取得的 AMI ID，使用：

```
BaaAmiId:
    Description: 'ami id (AL2) for ba anywhere'
    Default: 'ami-0d0fafcc636fd1e6d'
    Type: String
```

5. 使用下列命令，提供您在上一個步驟中用來儲存產生的檔案的 Amazon S3 資料夾路徑：

```
S3ApplicationFilesPath:
    Description: 'bucket name'
    Type: String
    Default: 's3://amzn-s3-demo-bucket/post-produced-files'
```

6. 最後，提供 s3-userdata-folder-path 的資料夾路徑：

```
S3UserDataPath:
    Description: 'S3 userdata folder path'
    Type: String
```

```
    Default: 's3://amzn-s3-demo-bucket/userdata'
```

- （選用）您可以為 tomcat 啟用 HTTPS 模式和基本 HTTP 身分驗證。雖然預設設定也可以運作。

> **Note**
>
> 根據預設，HTTPS 模式會在參數 BacHttpsMode 中停用，並設定為模式 HTTP：

例如：

```
BacHttpsMode:
    Description: 'http or https for Blue Age Runtime connection mode '
    Default: 'http'
    Type: String
    AllowedValues: [http, https]
```

- （選用）若要啟用 HTTPS 模式，您必須將值變更為 HTTPS，並透過變更變數 ACMCertArn 的值來提供 ACM 憑證 ARN：

```
ACMCertArn:
    Type: String
    Description: 'ACM certificate ARN'
    Default: 'your arn certificate'
```

- （選用）預設會停用基本身分驗證，並將 參數 WithBacBasicAuthentication 設為 false。您可以將 值設定為 true 來啟用它。

```
WithBacBasicAuthentication:
    Description: 'false or true for Blue Age Runtime Basic Authentication '
    Default: false
    Type: String
    AllowedValues: [true, false]
```

7. 完成組態後，您可以使用編輯的 CloudFormation 範本建立堆疊。

步驟 6：測試 AWS Blu Age Amazon EC2 執行個體

手動執行 CloudFormation 範本，為 CardDemo 應用程式建立 AWS Blu Age Amazon EC2 執行個體，以確保其啟動時不會發生錯誤。這是為了驗證 CloudFormation 範本和所有先決條件是否有效，然後再搭配應用程式測試功能使用 CloudFormation 範本。然後，您可以使用應用程式測試，在重播和比較期間自動建立目標 AWS Blu Age Amazon EC2 執行個體。

1. 執行 CloudFormation 建立堆疊命令來建立 AWS Blu Age Amazon EC2 執行個體，提供您在上一個步驟中編輯的 m2-with-ba-using-snapshots-https-authentication.yml CloudFormation 範本：

```
aws cloudformation create-stack --stack-name load-and-create-ba-snapshots —-
template-url https://apptest-ba-demo.s3.us-west-2.amazonaws.com/m2-with-ba-using-
snapshots-https-authentication.yml --capabilities CAPABILITY_NAMED_IAM --region us-
west-2
```

> **Note**
>
> 請記得指定還原 AWS Blu Age AMI 的正確區域。

2. 在主控台中尋找執行中的 Amazon EC2 執行個體，以確保一切正常運作。使用 Session Manager 連線到它。
3. 連線至 Amazon EC2 執行個體後，請使用下列命令：

```
sudo su
cd /m2-anywhere/tomcat.gapwalk/velocity/logs
cat catalina.log
```

4. 請確定日誌中沒有例外狀況或錯誤。
5. 接著，使用此命令來檢查應用程式是否正在回應：

```
curl http://localhost:8080/gapwalk-application/
```

您將看到「Jics 應用程式正在執行」訊息。

步驟 7：驗證先前步驟是否正確完成

在接下來的幾個步驟中，我們將使用 AWS 大型主機現代化應用程式測試來重播和比較 CardDemo 應用程式建立的資料集。這些步驟依賴於成功完成本教學課程中的所有先前步驟。在繼續之前驗證下列項目：

1. 您已成功透過 AWS CloudFormation 範本在 Amazon EC2 執行個體上建立 AWS Blu Age。
2. Amazon EC2 上 AWS Blu Age 上的 Tomcat 服務已啟動並執行，沒有例外。

當您取得使用 CardDemo 應用程式執行的 EC2 執行個體時，請在應用程式測試主控台上完成下列步驟，以執行重播並比較批次資料集。

步驟 8：建立測試案例

在此步驟中，您會建立測試案例，用來比較卡片示範應用程式中建立的資料集。

1. 建立新的測試案例。提供名稱和描述。
2. 指定 `CREASTMT.JCL`做為 JCL 名稱。
3. 將下列資料集新增至測試案例定義：

名稱	CCSID	RecordFormat	RecordLength
AWS.M2.CARDDEMO.STATEMNT.PS	「037」	FB	80
AWS.M2.CARDDEMO.STATEMNT.HTML	"037"	FB	100

> **Note**
>
> 您的 JCL 名稱和資料集詳細資訊必須相符。

步驟 9：建立測試套件

1. 建立新的測試套件，並提供其名稱和描述。
2. 將您在上一個步驟中建立的測試案例新增至您的測試套件。
3. 測試套件建立後，在大型主機上擷取測試案例，並將大型主機參考資料上傳至 AWS 應用程式測試。
4. 選擇 Create test suite (建立測試套件)。

步驟 10：建立測試環境組態

1. 建立新的測試環境組態，並提供名稱和描述。
2. 新增 CloudFormation 範本。您也可以從 CloudFormation 範本新增輸入參數名稱和值。
3. 選擇 AWS Mainframe Modernization 服務非受管 AWS Blu Age 做為您的執行時間。
4. 新增 AWS Mainframe Modernization 應用程式 ID 名稱的輸出變數名稱、VPC 端點服務 ID 的輸出變數名稱、接聽程式連接埠的輸出變數名稱，以及 WebApp 名稱的輸出變數名稱。

   > **Note**
   >
   > 這些欄位的名稱應與 CloudFormation 範本中的輸出變數名稱相符，該名稱將在堆疊建立期間從 AWS Mainframe Modernization 傳回。

5. （選用）選擇 DMS（資料庫遷移服務）任務 ARN 和來源資料庫 DDL（資料庫定義語言）S3 URI 位置的輸出變數名稱。
6. （選用）自訂您的 Key Management Service (KMS) 金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的管理客戶受管金鑰的存取。
7. 選擇建立測試環境組態。

步驟 11：在測試套件中上傳輸入資料

在此步驟中，您會在來源上執行測試案例。若要執行此作業：

1. 下載並執行源自 CardDemo 應用程式大型主機執行的資料集。
2. 將解壓縮的資料夾上傳至您的 Amazon S3 儲存貯體。此 Amazon S3 儲存貯體必須與其他應用程式測試資源位於相同的區域。

> **Note**
>
> 應該有兩個檔案的名稱符合先前測試案例中傳遞的資料集名稱。

3. 在測試套件概觀頁面上，選擇上傳按鈕。
4. 在上傳參考資料頁面上，指定您上傳從來源大型主機取得之資料集的 Amazon S3 位置。
5. 選擇上傳以開始上傳程序。

> **Note**
>
> 等待錄製完成，再執行重播和比較。

步驟 12：重播並比較

在 Amazon EC2 環境的目標 AWS AWS Blu Age 中執行測試套件和測試案例。應用程式測試會擷取重播產生的資料集，並將其與大型主機上記錄的參考資料集進行比較。

1. 選擇重播並比較。建立 CloudFormation 堆疊並執行比較大約需要三分鐘。

完成所有項目後，您應該會有比較結果，其中有一些差異是為了本示範而刻意建立的。

AWS Mainframe Modernization Application Testing 支援的資料集程式碼頁面

使用下表來判斷 AWS Application Testing 是否支援資料的編碼字元集識別符 (CCSID)。如果您的資料使用不支援的 CCSID，建議您將其轉換為支援的 CCSID，或聯絡我們尋求協助。

CCSID	字元集	描述
37	IBM037, IBM-037, Cp037	主持人：美國、加拿大 (「GESA」)、荷蘭、葡萄牙、巴西、澳洲、紐西蘭
273	IBM273, IBM-273, Cp273	主機：奧地利、德國

CCSID	字元集	描述
277	IBM277, IBM-277, Cp277	主機：丹麥、挪威
278	IBM278, IBM-278, Cp278	主機：芬蘭、瑞典
280	IBM280, IBM-280, Cp280	主機：義大利
284	IBM284, IBM-284, Cp284	主持人：西班牙、拉丁美洲（西班牙文）
285	IBM285, IBM-285, Cp285	主機：英國
297	IBM297, IBM-297, Cp297	主機：法國
300	IBM-300	日本資料庫 EBCDIC
301	IBM-301	PC 資料：日本資料庫
437	IBM437, IBM-437、US-ASCII、ASCII、Cp437、US-ASCII	PC 資料：PC Base USA，許多其他國家/地區
500	IBM500, IBM-500, Cp500	主持人：比利時、加拿大 (AS/400)、瑞士、國際Latin-1
720	IBM-720	MSDOS ARABIC
737	IBM-737, x-IBM737	MSDOS GREEK
775	IBM775, IBM-775	MSDOS BALTIC
808	IBM-808	PC 資料：俄羅斯塞爾維里克，含歐元
813	ISO-8859-7、ISO8859_7	ISO 8859-7：希臘
819	ISO-8859-1、ISO8859_1	ISO 8859-1：Latin-1國家
833	IBM-833	韓國 EBCDIC

CCSID	字元集	描述
834	IBM-834, x-IBM834	KOREAN 資料庫 EBCDIC
835	IBM-835	T-CHINESE 資料庫 EBCD
836	IBM-836	S-CHINESE EBCDIC
837	IBM-837	S-CHINESE EBCDIC
850	IBM850, IBM-850, Cp850	PC 資料：Latin-1 國家/地區
855	IBM855, IBM-855, Cp855	PC 資料：Cyrillic
856	IBM-856, x-IBM856, Cp856	PC 資料：希伯來文
858	IBM00858, IBM-858, Cp858	PC 資料：Latin-1 國家/地區，使用歐元
859	IBM-859	PC 資料：LATIN-9
860	IBM860, IBM-860	PC 資料：葡萄牙文
861	IBM861, IBM-861	PC 資料：冰島
862	IBM862, IBM-862, Cp862	PC 資料：希伯來文（遷移）
863	IBM863, IBM-863	PC 資料：加拿大
865	IBM865, IBM-865, Cp865	PC 資料：Den/Norway
866	IBM866, IBM-866, Cp866	PC 資料：俄羅斯塞拉維奇
867	IBM-867	PC 資料：希伯來文與歐元
870	IBM870, IBM-870, Cp870	主機： Latin-2 多語言
871	IBM871, IBM-871, Cp871	主機：冰島
874	x-IBM874	PC 資料：泰文
875	IBM-875, x-IBM875, Cp875	主機：希臘

CCSID	字元集	描述
897	IBM-897	PC 資料：日本 SB
912	ISO-8859-2、ISO8859_2	ISO 8859-2：Latin-2 多語言
915	ISO-8859-5、ISO8859_5	ISO 8859-5：西里爾文
916	ISO-8859-8、ISO8859_8	ISO 8859-8：希伯來文
918	IBM918, IBM-918, Cp918	主機：Urdu
920	ISO-8859-9、ISO8859_9	ISO 8859-9： Latin-5 (ECMA-128，土耳其 TS-5881)
921	IBM-921, x-IBM921, Cp921	PC 資料：立陶宛拉脫維亞
922	IBM-922, x-IBM922, Cp922	PC 資料：愛沙尼亞
923	ISO-8859-15, Cp923、ISO 8859_15_FDIS	ISO 8859-15：Latin-9
924	IBM-924	ISO 8859-15：Latin-9
927	IBM-927	PC 資料：T-Chinese
930	IBM-930, x-IBM930, Cp930	Katakana 主機：擴充 SBCS。Kanji 主機：DBCS，包括 4370 個使用者定義字元
932	IBM-932	PC 資料：Japan Mix
933	IBM-933, x-IBM933, Cp933	主機：擴充 SBCS。主機：DBCS，包括 1880 個使用者定義的字元和 11172 個完整的 Hangul 字元
935	IBM-935, x-IBM935, Cp935	主機：擴充 SBCS。主機：DBCS，包括 1880 個使用者定義的字元。

CCSID	字元集	描述
937	IBM-937, x-IBM937, Cp937	主機：擴充 SBCS。主機：DBCS，包括 6204 個使用者定義字元
939	IBM-939, x-IBM939, Cp939	拉丁主機：擴充 SBCS。Kanji 主機：DBCS，包括 4370 個使用者定義的字元。
942	IBM-942, IBM-942C, x-IBM942, x-IBM942C, Cp942, Cp942C	PC 資料：擴充 SBCS。PC 資料：DBCS 包含 1880 個使用者定義的字元
943	IBM-943, IBM-943C、Shift_JIS、Window-31j、Window-932、x-IBM943, x-IBM943C, Cp943, Cp943C, MS932	PC 資料：SBCS。PC 資料：DBCS for Open 環境，包括 1880 個 IBM 使用者定義字元
947	IBM-947	T-CHINESE BIG-5
948	IBM-948, x-IBM948, Cp948	PC 資料：擴充 SBCS。PC 資料：DBCS，包括 6204 個使用者定義的字元
949	IBM-949, IBM-949C, x-IBM949, x-IBM949C, Cp949, Cp949C	IBM KS 程式碼 - PC 資料：SBCS。IBM KS 程式碼 - PC 資料：DBCS 包含 1880 個使用者定義的字元
950	Big5, IBM-950, x-IBM950, Cp950	PC 資料：SBCS (IBM BIG5)。PC 資料：DBCS，包括 13493 CNS、566 IBM 已選取、6204 個使用者定義字元
951	IBM-951	PC 資料：IBM KS

CCSID	字元集	描述
954	EUC-JP、IBM-954, IBM-954C	G0：JIS X201 Roman。G1：JIS X208-1990。G1：JIS X201 Katakana。G1：JIS X212
964	EUC-TW、IBM-964, x-IBM964, Cp964	G0：ASCII。G1：CNS 11643 平面 1。G1：CNS 11643 平面 2。
970	EUC-KR、x-IBM970, Cp970	G0：ASCII。G1：KSC X5601-1989，包括 1880 個使用者定義字元
971	IBM-971	韓國 EUC
1006	IBM-1006, x-IBM1006, Cp1006	ISO-8：烏爾都文
1025	IBM-1025, x-IBM1025, Cp1025	主機：西里爾文多語言
1026	IBM1026, IBM-1026, Cp1026	主機： Latin-5（土耳其）
1027	IBM-1027	日本 LATIN EBCD
1041	IBM-1041	PC 資料：日本
1043	IBM-1043	PC 資料：T-Chinese
1046	IBM-1046, IBM-1046S, x-IBM1046	ARABIC - PC
1047	IBM1047, IBM-1047	主機： Latin-1
1051	hp-roman8	HP 模擬
1088	IBM-1088	PC 資料： Korea KS

CCSID	字元集	描述
1089	ISO-8859-6、ISO8859_6	ISO 8859-6：阿拉伯文
1097	IBM-1097, x-IBM1097, Cp1097	主機：Fargis
1098	IBM-1098, x-IBM1098, Cp1098	PC 資料：Fargis
1112	IBM-1112, x-IBM1112, Cp1112	主持人：立陶宛拉脫維亞
1114	IBM-1114	PC 資料：T-CH SB
1115	IBM-1115	PC 資料：S-CH GB
1122	IBM-1122, x-IBM1122, Cp1122	主機：愛沙尼亞
1123	IBM-1123, x-IBM1123, Cp1123	主機：烏克蘭西里爾文
1124	IBM-1124, x-IBM1124, Cp1124	8 位元：西里爾文、白俄羅斯文
1140	IBM01140, IBM-1140, Cp1140	主持人：美國、加拿大 (「GESA」)、荷蘭、葡萄牙、巴西、澳洲、紐西蘭，使用歐元
1141	IBM01141, IBM-1141, Cp1141	主機：奧地利、德國，使用歐元
1142	IBM01142, IBM-1142, Cp1142	主機：丹麥、挪威，使用歐元
1143	IBM01143, IBM-1143, Cp1143	主持人：瑞典芬蘭，含歐元
1144	IBM01144, IBM-1144, Cp1144	主機：義大利，含歐元

CCSID	字元集	描述
1145	IBM01145, IBM-1145, Cp1145	主持人：西班牙、拉丁美洲（西班牙文），含歐元
1146	IBM01146, IBM-1146, Cp1146	主機：英國，含歐元
1147	IBM01147, IBM-1147, Cp1147	主機：法國，含歐元
1148	IBM01148, IBM-1148, Cp1148	主持人：比利時、加拿大 (AS/400)、瑞士、國際Latin-1 和歐元
1149	IBM01149, IBM-1149, Cp1149	主機：冰島，含歐元
1200	UTF-16BE	Unicode 搭配字元集 65535。如果沒有位元組順序標記 (BOM)，則假設為 UTF-16 BE (big-endian)。
1202	UTF-16LE	UTF-16 LE 搭配 IBM PUA
1204	UTF-16	UTF-16 搭配 IBM PUA
1208	UTF-8, UTF-8J, UTF8	Unicode 搭配字元集 65535。UTF-8。
1232	UTF-32BE	UTF-32 BE 搭配 IBM PUA
1234	UTF-32LE	UTF-32 LE 搭配 IBM PUA
1236	UTF-32	UTF-32 搭配 IBM PUA
1351	IBM-1351	日本開放
1362	IBM-1362	韓國 MS-WIN

CCSID	字元集	描述
1363	IBM-1363, IBM-1363C、Window-949、MS949	PC 資料：MS Windows 韓文 SBCS。PC 資料：MS Windows Koran DBCS，包括 11172 完整 Hangul
1364	IBM-1364	主機：擴充 SBCS。主機：DBCS，包括 1880 個使用者定義的字元和 11172 個完整的 Hangul 字元
1370	IBM-1370	PC 資料：擴充 SBCS，以歐元表示。PC 資料：DBCS 包含 6204 個使用者定義字元，以歐元表示
1371	IBM-1371	主機：擴充 SBCS，使用歐元。主機：DBCS 包含 6204 個使用者定義字元，以歐元表示
1375	Big5-HKSCS	適用於 HKSCS 的混合 Big-5 Ext
1380	IBM-1380	PC 資料：S-CH GB
1381	IBM-1381, x-IBM1381, Cp1381	PC 資料：擴充 SBCS (IBM GB)。PC 資料：DBCS (IBM GB)，包括 31 個 IBM 選取的 1880 個使用者定義字元
1382	IBM-1382	S-CHINESE EUC
1383	EUC-CN、GB2312, IBM-1383, x-IBM1383, Cp1383	G0：ASCII。G1：GB 2312-80 組
1385	IBM-1385	PC 資料：S-CH GBK

CCSID	字元集	描述
1386	GBK、IBM-1386、Windo w-936、MS936	PC 資料：S-Chinese GBK 和 T-Chinese IBM BIG-5。PC 資料：S-Chinese GBK
1388	IBM-1388	主機：擴充 SBCS。主機：DBCS，包括 1880 個使用者定義的字元
1390	IBM-1390	Katakana 主機：擴充 SBCS，以歐元表示。Kanji 主機：DBCS，包括 6205 個使用者定義字元
1399	IBM-1399	拉丁主機：擴充 SBCS，以歐元表示。Kanji 主機：DBCS，包括 4370 個使用者定義字元，以歐元表示
5050	JIS0201, JIS0208, JIS0212, JIS0201, JIS0208, JIS0212	G0：JIS X201 Roman。G1：JIS X208-1990。G1：JIS X201 Katakana。G1：JIS X212
5054	ISO-2022-JP	日文 TCP
5346	windows-1250、Cp1250	MS Windows： Latin-2，第 2 版，含歐元
5347	windows-1251、Cp1251	MS Windows：Cyrillic，第 2 版，含歐元
5348	windows-1252、Cp1252	MS Windows：Latin-1國家/地區，第 2 版，含歐元
5349	windows-1253、Cp1253	MS Windows：希臘，第 2 版，含歐元

CCSID	字元集	描述
5350	windows-1254、Cp1254	MS Windows：土耳其，第 2 版，含歐元
5351	windows-1255、Cp1255	MS Windows：希伯來文，第 2 版，含歐元
5352	windows-1256、 windows-1256S, Cp1256	MS Windows：阿拉伯文，第 2 版，含歐元
5353	windows-1257、Cp1257	MS Windows：波羅的海邊緣，第 2 版，含歐元
5354	windows-1258、Cp1258	MS Windows：越南文，第 2 版，含歐元
5488	GB18030	GB18030、1 位元組資料 GB18030、2 位元組資料 GB18030、4 位元組資料
9030	IBM-838, Cp838	主機：泰文延伸 SBCS
9066	IBM-874, Cp874	PC 資料：泰文延伸 SBCS
9400	CESU-8	CESU-8 搭配 IBM PUA
25546	ISO-2022-KR	KOREAN TCP
33722	IBM-33722, IBM-33722C	IBMeucJP

AWS 大型主機現代化應用程式測試中的資料保護

AWS 共同責任模型適用於 AWS 大型主機現代化應用程式測試中的資料保護。如此模型所述， AWS 負責保護執行所有 的全域基礎設施 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。

建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 來保護 AWS 帳戶 登入資料並設定個別使用者。因此，每個使用者只會獲得完成其任務所需的許可。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 中的所有預設安全控制 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 AWS 透過命令列界面或 API 存取 時需要 FIPS 140-2 驗證的密碼編譯模組，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。

建議您避免將任何機密或敏感資訊，例如客戶的電子郵件地址，用於標籤或自由格式的文字欄位 （例如名稱欄位）。這包括當您 AWS 服務 使用 AWS 大型主機現代化應用程式測試，或使用主控台、API AWS CLI、 或 AWS SDKs 的其他 時。您輸入標籤或用於名稱的任意格式文字欄位的任何資料都可能用於計費或診斷日誌。如果您提供 URL 給外部伺服器，請避免在 URL 中使用登入資料資訊來驗證您對該伺服器的請求。

AWS 大型主機現代化應用程式測試收集的資料

AWS Mainframe Modernization Application Testing 會收集多種類型的資料：

- `Resource definition`：資源定義表示當您建立或更新類型測試案例、測試套件或測試組態的資源時，傳遞給 Application Testing 的資料。
- `Scripts for replay`：這些是針對您的 AWS Mainframe Modernization 應用程式傳遞至 Application Testing 的指令碼。
- `Data for comparison`：這些是傳遞給 Application Testing 進行比較的資料集或資料庫變更資料擷取 (CDC) 檔案。

AWS Mainframe Modernization Application Testing 會以原生方式存放此資料 AWS。我們從您收集的資料會存放在 AWS 大型主機現代化應用程式測試管理的 Amazon S3 儲存貯體中。當您刪除資源時，相關聯的資料會從 Amazon S3 儲存貯體中移除。

當您開始測試執行以執行重播以測試互動式工作負載時， AWS 大型主機現代化應用程式測試會將指令碼下載到支援 Amazon ECS 管理的 Fargate 容器的暫時性儲存體，以執行重播。指令碼檔案會在重播

完成且指令碼產生的輸出檔案存放在您帳戶中的應用程式測試受管 Amazon S3 儲存貯體中時刪除。當您刪除測試執行時，重播輸出檔案會從 Amazon S3 儲存貯體中刪除。

同樣地，當您開始測試執行以比較檔案（資料集或資料庫變更）時，AWS 大型主機現代化應用程式測試會將檔案下載到支援 Amazon ECS 管理的 Fargate 容器，以執行比較。下載的檔案會在比較操作完成後立即刪除。比較輸出資料會儲存在您帳戶中的應用程式測試受管 Amazon S3 儲存貯體中。當您刪除測試執行時，輸出資料會從 S3 儲存貯體中刪除。

當您將資料放入 AWS 大型主機現代化應用程式測試用於比較檔案的 Amazon S3 儲存貯體時，您可以使用所有可用的 Amazon S3 Amazon S3加密選項來保護資料。

AWS 大型主機現代化應用程式測試的靜態資料加密

AWS Mainframe Modernization Application Testing 與 AWS Key Management Service (KMS) 整合，可在永久存放資料的所有相依資源上提供透明的伺服器端加密 (SSE)。資源範例包括 Amazon Simple Storage Service、Amazon DynamoDB 和 Amazon Elastic Block Store。 AWS 大型主機現代化應用程式測試會建立和管理 中的對稱加密 AWS KMS 金鑰 AWS KMS。

依預設加密靜態資料，有助於降低保護敏感資料所涉及的營運開銷和複雜性。同時，它可讓您測試需要嚴格加密合規和法規要求的應用程式。

您無法在建立測試案例、測試套件或測試組態時停用此加密層或選取替代加密類型。

您可以使用自己的客戶受管金鑰進行比較檔案和 AWS CloudFormation 範本，以加密 Amazon S3。您可以使用此金鑰來加密在應用程式測試中為測試執行建立的所有資源。

> **Note**
>
> DynamoDB 資源一律使用 Application Testing 服務帳戶中 AWS 受管金鑰 的 進行加密。您無法使用客戶受管金鑰加密 DynamoDB 資源。

AWS Mainframe Modernization Application Testing 會將您的客戶受管金鑰用於下列任務：

- 從應用程式測試將資料集匯出至 Amazon S3。
- 將比較輸出檔案上傳至 Amazon S3。

如需更多資訊，請參閱 AWS Key Management Service 開發人員指南中的客戶受管金鑰。

建立客戶受管金鑰

您可以使用 AWS Management Console 或 AWS KMS APIs 來建立對稱客戶受管金鑰。

建立對稱客戶受管金鑰

請依照《AWS Key Management Service 開發人員指南》中建立對稱客戶受管金鑰的步驟進行。

金鑰政策

金鑰政策會控制客戶受管金鑰的存取權限。每個客戶受管金鑰都必須只有一個金鑰政策，其中包含決定誰可以使用金鑰及其使用方式的陳述式。在建立客戶受管金鑰時，可以指定金鑰政策。

以下是使用 ViaService 縮小存取範圍的金鑰政策範例，允許應用程式測試在您的帳戶中寫入重播和比較產生的資料。呼叫 `StartTestRun` API 時，您應該將此政策連接至 IAM 角色。

Example

```
{
    "Sid": "TestRunKmsPolicy",
    "Action": ["kms:Decrypt", "kms:GenerateDataKey"],
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/TestRunRole"
    },
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:ViaService": ["s3.amazonaws.com"]
        },
        "ForAnyValue:StringEquals": {
            "kms:EncryptionContextKeys": "aws:apptest:testrun"
        }
    }
}
```

如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的管理客戶受管金鑰的存取。

如需有關故障診斷金鑰存取的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》。

為 AWS 大型主機現代化應用程式測試指定客戶受管金鑰

建立測試組態時，您可以輸入 KEY ID 來指定客戶受管金鑰。應用程式測試使用 來加密測試執行期間上傳至 Amazon S3 儲存貯體的資料。

- KEY ID — 客戶受管金鑰的金鑰識別符。輸入金鑰 ID、金鑰 ARN、別名名稱或別名 ARN。

若要在使用 建立測試組態時新增客戶受管金鑰 AWS CLI，請指定 kmsKeyId 參數，如下所示：

```
create-test-configuration --name test \
--resources '[{
    "name": "TestApplication",
    "type": {
        "m2ManagedApplication": {
            "applicationId": "wqju4m2dcz3rhny5fpdozrsdd4",
            "runtime": "MicroFocus"
        }
    }
}]' \
--service-settings '{
    "kmsKeyId": "arn:aws:kms:us-west-2:111122223333:key/05d467z6-c42d-40ad-
b4b7-274e68b14013"
}'
```

AWS Mainframe Modernization Application Testing 加密內容

加密內容是一組選用的金鑰值對，包含資料的其他相關內容資訊。

AWS KMS 使用加密內容做為額外的已驗證資料，以支援已驗證的加密。當您在加密資料的請求中包含加密內容時， 會將加密內容 AWS KMS 繫結至加密的資料。若要解密資料，您必須在請求中包含相同的加密內容。

AWS Mainframe Modernization Application Testing 加密內容

AWS Mainframe Modernization Application Testing 會在與測試執行相關的所有 AWS KMS 密碼編譯操作中使用相同的加密內容，其中金鑰為 ，aws:apptest:testrun而值為測試執行的唯一識別符。

Example

```
"encryptionContext": {
        "aws:apptest:testrun": "u3qd7uhdandgdkhhi44qv77iwq"
```

```
}
```

使用加密內容進行監控

當您使用對稱客戶受管金鑰來加密測試執行時，您也可以使用稽核記錄和日誌中的加密內容，來識別上傳資料至 Amazon S3 時如何使用客戶受管金鑰。

監控 AWS 大型主機現代化應用程式測試的加密金鑰

當您將 AWS KMS 客戶受管金鑰與 AWS Mainframe Modernization Application Testing 資源搭配使用時，您可以使用 AWS CloudTrail 來追蹤 AWS Mainframe Modernization Application Testing 在上傳物件時傳送至 Amazon S3 的請求。

傳輸中加密

對於定義測試交易工作負載步驟的測試案例，執行硒指令碼的 Application Testing 受管終端機模擬器與 AWS Mainframe Modernization 應用程式端點之間的資料交換不會在傳輸中加密。 AWS Mainframe Modernization Application Testing 會使用 AWS PrivateLink 連線到您的應用程式端點，以私有交換資料，而不會透過公有網際網路暴露流量。

AWS Mainframe Modernization Application Testing 使用 HTTPS 來加密服務 APIs。 AWS Mainframe Modernization Application Testing 中的所有其他通訊都受到服務 VPC 或安全群組以及 HTTPS 的保護。

根據預設，傳輸中的基本加密是設定，但不適用於以TN3270通訊協定為基礎的互動式工作負載測試。

AWS Mainframe Modernization 應用程式測試如何與 IAM 搭配使用

在您使用 IAM 管理 AWS 大型主機現代化應用程式測試的存取權之前，請先了解有哪些 IAM 功能可與 AWS 大型主機現代化應用程式測試搭配使用。

可與 AWS 大型主機現代化應用程式測試搭配使用的 IAM 功能

IAM 功能	AWS 大型主機現代化應用程式測試支援
身分型政策	是
資源型政策	否
政策動作	是

IAM 功能	AWS 大型主機現代化應用程式測試支援
政策資源	是
政策條件索引鍵	是
ACL	否
ABAC (政策中的標籤)	是
臨時憑證	是
轉送存取工作階段 (FAS)	是
服務角色	否
服務連結角色	否

若要全面了解 AWS Mainframe Modernization Application Testing 和其他 AWS 服務如何與大多數 IAM 功能搭配使用，請參閱《IAM 使用者指南》中的AWS 與 IAM 搭配使用的 服務。

AWS 大型主機現代化應用程式測試的身分型政策

支援身分型政策：是

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。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的透過客戶管理政策定義自訂 IAM 許可。

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。

AWS Mainframe Modernization 應用程式測試的身分型政策範例

若要檢視 AWS Mainframe Modernization Application Testing 身分型政策的範例，請參閱 AWS Mainframe Modernization 的身分型政策範例。

AWS 大型主機現代化應用程式測試中的資源型政策

支援資源型政策：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。委託人可以包括帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

如需啟用跨帳戶存取權，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，做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。當委託人和資源位於不同位置時 AWS 帳戶，信任帳戶中的 IAM 管理員也必須授予委託人實體（使用者或角色）存取資源的許可。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 中的快帳戶資源存取。

AWS Mainframe Modernization 應用程式測試的政策動作

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。政策動作通常具有與相關聯 AWS API 操作相同的名稱。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 AWS 大型主機現代化應用程式測試動作的清單，請參閱《服務授權參考》中的 AWS 大型主機現代化應用程式測試定義的動作。

AWS Mainframe Modernization Application Testing 中的政策動作在動作之前使用下列字首：

```
apptest
```

如需在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "apptest:CreateTestCase",
      "apptest:StartTestRun"
]
```

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 `List` 文字的所有動作，請包含以下動作：

```
"Action": "apptest:List*"
```

若要檢視 AWS Mainframe Modernization 身分型政策的範例，請參閱 AWS Mainframe Modernization 的身分型政策範例。

AWS Mainframe Modernization 應用程式測試的政策資源

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": "*"
```

您可以使用特定 AWS 大型主機現代化應用程式測試資源ARNs 來識別 IAM 政策適用的資源，藉此限制對特定大型主機現代化應用程式測試資源的存取。如需有關 ARN 格式的詳細資訊，請參閱《AWS 一般參考》中的 Amazon Resource Name (ARN)。

例如， AWS 大型主機現代化應用程式測試測試案例具有下列 ARN。

```
"Resource": "arn:aws:apptest:regionId:accountId:testcase/service-generated-unique-identifier"
```

AWS Mainframe Modernization Application Testing 測試組態具有下列 ARN。

```
"Resource": "arn:aws:apptest:regionId:accountId:testconfiguration/service-generated-unique-identifier"
```

AWS Mainframe Modernization Application Testing 套件具有下列 ARN。

```
"Resource": "arn:aws:apptest:regionId:accountId:testsuite/service-generated-unique-identifier"
```

AWS Mainframe Modernization Application Testing 測試執行具有下列 ARN。

```
"Resource": "arn:aws:apptest:regionId:accountId:testrun/service-generated-unique-identifier"
```

並非所有 AWS 大型主機現代化應用程式測試動作都支援資源層級許可。對於不支援資源層級許可的動作，您必須使用萬用字元 (*)。

下列 AWS 大型主機現代化應用程式測試動作不支援資源層級許可。

```
ListTestCases
ListTestConfigurations
ListTestRuns
ListTestSuites
ListTagsForResource
```

若要查看 AWS Mainframe Modernization Application Testing 資源類型及其 ARNs，請參閱《服務授權參考》中的 AWS Mainframe Modernization Application Testing 定義的資源。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱 AWS Mainframe Modernization Application Testing 定義的動作。

若要檢視 AWS Mainframe Modernization 身分型政策的範例，請參閱 AWS Mainframe Modernization 的身分型政策範例。

AWS Mainframe Modernization 應用程式測試的政策條件索引鍵

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， 會使用邏輯OR操作 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的條件金鑰。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的AWS 全域條件內容索引鍵。

若要查看 AWS Mainframe Modernization Application Testing 條件索引鍵的清單，請參閱《服務授權參考》中的 AWS Mainframe Modernization Application Testing 的條件索引鍵。若要了解您可以使用條件金鑰的動作和資源，請參閱 AWS Mainframe Modernization Application Testing 定義的動作。

若要檢視 AWS Mainframe Modernization 身分型政策的範例，請參閱 AWS Mainframe Modernization 的身分型政策範例。

AWS Mainframe Modernization Application Testing 中的存取控制清單 (ACLs)

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。

具有 AWS 大型主機現代化應用程式測試的屬性型存取控制 (ABAC)

支援 ABAC (政策中的標籤)：是

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。您可以將標籤連接至 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。

如需根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，在政策的條件元素中，提供標籤資訊。

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》中的使用 ABAC 授權定義許可。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 AWS 大型主機現代化應用程式測試使用暫時登入資料

支援臨時憑證：是

當您使用臨時登入資料登入時，有些 AWS 服務 無法運作。如需詳細資訊，包括哪些 AWS 服務 使用臨時登入資料，請參閱《AWS 服務 IAM 使用者指南》中的 使用 IAM。

如果您 AWS Management Console 使用使用者名稱和密碼以外的任何方法登入 ，則使用臨時登入資料。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取 時，該程序會自動建立臨時登入資料。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》中的從使用者切換至 IAM 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。

AWS 大型主機現代化應用程式測試的轉送存取工作階段

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，您會被視為委託人。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 使用呼叫 的委託人許可 AWS 服務，結合 AWS 服務 請求向下游服務提出請求。只有當服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時，才會提出 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。

> **Important**
>
> 這些字符可讓 AWS 大型主機現代化應用程式測試在沒有您明確同意的情況下存取客戶資料；例如， AWS 大型主機現代化應用程式測試會在未取得客戶明確許可的情況下，將測試結果交付給客戶 Amazon S3 儲存貯體。您可能需要相應地更新任何合規文件。

AWS Mainframe Modernization 應用程式測試的服務角色

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。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可權給 AWS 服務。

AWS 大型主機現代化應用程式測試的服務連結角色

支援服務連結角色：否

服務連結角色是連結至 的一種服務角色 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 中 AWS 帳戶 ，並由服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

使用 AWS Blu Age 自動重構應用程式

使用 AWS Blu Age 自動重構可提供end-to-end解決方案，以遷移和現代化您的大型主機應用程式。重構程序中的步驟如下：

- 分析庫存
- 分析相依性
- 自動轉換程式碼
- 擷取和管理測試案例

您可以在 Blu Insights 工具中完成上述步驟，可從 AWS Mainframe Modernization 主控台透過單一登入取得。如需 Blu Insights 的詳細資訊，請參閱 Blu Insights 文件。

當您對轉換後的原始程式碼感到滿意時，就可以移至 AWS，您將在其中完成以下步驟：

- 建置和部署重構的應用程式。
- 在 AWS Mainframe Modernization 中部署和監控您的應用程式。

AWS Blu Age 提供各種執行時間選項，以符合不同的部署案例和操作偏好設定。這些包括受管和非受管執行期，每個執行期都有自己的一組功能和部署目標。

如需可用 AWS Blu Age Runtime 選項的完整概觀，包括受管和非受管版本、部署目標及其個別特性，請參閱 the section called “AWS Blu Age 執行時間選項” 文件。

本指南將協助您了解執行時間選項之間的差異，並為您的現代化專案選擇最適合的選項。

主題

- AWS 藍卡年齡版本
- AWS Blu Age 執行期概念
- 設定 AWS Blu Age Runtime 的組態
- AWS Blu Age 執行時間 APIs
- 設定 AWS Blu Age 執行期 （ 非受管 ）
- 使用 Blu Age Developer IDE 修改原始程式碼
- AWS 藍年常見問答集

AWS 藍卡年齡版本

AWS Blu Age 引擎有多種版本供您選擇。此頁面概述 AWS Blu Age 版本控制的運作方式、每個版本版本有哪些變更、版本升級說明、如何與客戶溝通 AWS Blu Age 更新，以及這些版本的生命週期。

the section called “AWS Blu Age 版本控制” 此頁面詳細說明 版本的相關資訊，以及主要和次要版本如何識別每個版本。the section called “AWS Blu Age 版本備註” 此頁面具有每個主要和次要版本的深入版本備註。 the section called “AWS Blu Age 安全性漏洞”頁面提及 AWS Blu Age 如何處理常見漏洞與暴露 (CVE)。 the section called “升級 AWS 藍髮年齡”詳細說明 AWS Blu Age 版本的升級說明。並the section called “AWS 藍卡存留期生命週期”包含 AWS Blu Age Runtime 主要版本生命週期結束 (EOL) 日期的所有詳細資訊。

主題

- AWS Blu Age 版本控制
- AWS Blu Age 執行時間選項
- AWS Blu Age 版本備註
- AWS Blu Age 安全性漏洞
- 升級 AWS Blu Age 的說明
- AWS 藍卡存留期生命週期

AWS Blu Age 版本控制

AWS Blu Age Transformation 和 Runtime 產品是使用符合 semver （語意版本控制） 規範的方案進行版本控制。若要部署應用程式，您需要使用與現代化程式碼相容的對應執行時間版本。如果您對要使用的版本有任何疑問，請聯絡您的 AWS Blu Age 交付管理員。

推出

每個版本都會以**`[Major].[Minor].[Patch]`**模式識別。例如，使用 AWS Blu Age Runtime 版本時`4.1.0`，主要版本為 4，次要版本為 1，修補程式版本為 0。

我們打算每月發行新的 AWS Blu Age Runtime 次要版本，以及當產品或其相依性發生重大變更時的新主要版本。

如需每個版本中可用新功能的詳細資訊，請參閱 the section called “AWS Blu Age 版本備註”。

Alpha 發行前版本

每個 alpha 發行前版本都會以 **[Major].[Next_Minor].0-alpha.[pre-release]** 模式識別。例如，在發行前 中`4.2.0-alpha.1`， 中可用的變更`alpha.1`將在下一個次要版本 中發行`4.2.0`。

Alpha 發行前版本是頻繁的短期版本，旨在並在現代化專案期間快速迭代。新的 Alpha 發行前版本沒有固定版本節奏，並在開發和測試時提供。

如需版本控制、升級和支援的詳細資訊，請參閱 元件生命週期。

> **Important**
>
> Alpha 發行前版本應僅在現代化專案階段使用，而非用於生產或關鍵工作負載。

AWS Blu Age 執行時間選項

AWS Blu Age 提供三種類型的執行期選項，以滿足現代化旅程的不同階段和營運需求。此頁面說明每個選項、其特性、使用案例，以及如何存取它們。

非受管執行期

透過 AWS Blu Age Runtime （非受管），您可以在自己的 AWS 帳戶 中部署現代化應用程式，讓您管理自己的基礎設施。此選項同時提供發行版本和發行前版本，讓您靈活地以您想要的方式操作執行現代化應用程式所需的所有技術元件。您可以選擇生產環境的穩定版本，或用於測試和開發用途的預先發行版本。

非受管執行期由客戶部署和管理，可對執行期環境提供更多控制。它提供自動重構功能，適合自訂部署案例。

使用情況

非受管執行時間適用於測試和生產環境，當需要特定自訂執行時間環境時特別有用。

如何存取

若要請求存取非受管執行期成品，請參閱加入 AWS 藍卡年齡執行期。

部署

AWS Blu Age Runtime （非受管） 可用於部署：

- Amazon EC2

- Amazon EC2 上的 Amazon ECS
- Amazon EC2 上的 Amazon EKS
- 由 管理的 Amazon ECS AWS Fargate

在 Amazon EC2 上部署可以直接在執行個體中或透過 Docker 容器化應用程式完成，這是使用 Amazon ECS 或 Amazon EKS 時的偏好方式。

如需詳細部署說明，請參閱設定 AWS Blu Age Runtime （非受管） 文件。

受管執行期

使用 AWS Blu Age Runtime 受管，您可以將現代化應用程式部署到 AWS 受管環境，以簡化您的體驗，因此您不需要管理執行現代化應用程式的基礎基礎設施。

受管執行期具有 管理的基礎設施 AWS，具有自動更新和修補程式。它提供簡化的操作和維護。受管執行時間中只有發行版本可用，可確保生產環境的穩定性和可靠性。

使用情況

受管執行時間非常適合生產環境，最適合您偏好實作方法進行執行時間管理的情況。

如何存取

若要存取 AWS Blu Age Runtime Managed 版本，您只需要存取下列 AWS 服務：

- Amazon S3
- AWS 大型主機現代化

透過適當的帳戶許可，您可以與受管執行期環境無縫互動。此簡化存取可確保您可以利用 Blu Age Runtime 的完整功能，同時受益 AWS於 的受管服務。

部署

若要了解如何設定和使用受管執行期，請參閱設定 AWS Blu Age 的受管執行期文件。

開發人員執行期 (BluInsights 工具箱)

開發人員執行期可從 BluInsights Toolbox 存取，專為開發和測試階段而設計，並經常使用最新功能進行更新。此執行時間同時包含發行版本和 Alpha 發行前版本，讓開發人員能夠存取穩定的組建和尖端功能。其主要目的是支援本機開發環境中的特定測試或開發活動，通常是來自 IDE。重要的是，此執行時間僅限 2 小時使用，因此適合集中開發工作階段，而非長期或生產部署。

使用情況

開發人員執行時間在初始開發和現代化專案期間，以及快速迭代和測試現代化應用程式時非常理想。

如何存取

存取 BluInsights 工具箱是 AWS 做為 Blu Age 專案參與的一部分。開發人員執行期可透過 AWS Blu Age 工具箱請求取得。核准後，您就可以存取特定的 S3 儲存貯體： `s3://toolbox-dev-runtime-<region>`

這些儲存貯體可在 us-east-1 和 us-east-2 區域取得。若要使用可用的儲存貯體，請將區域附加至儲存貯體名稱 （ 例如 `s3://toolbox-dev-runtime-us-east-1`)。

如需如何請求存取和設定必要許可的詳細說明，請參閱開發和特殊 AWS Blu Age Runtimes 文件。

部署

部署開發人員執行期：

1. 使用 列出可用的版本 AWS CLI：

```
aws s3 ls s3://toolbox-dev-runtime
```

2. 選擇特定版本並列出其內容：

```
aws s3 ls s3://toolbox-dev-runtime/[version]/
```

3. 下載執行時間成品：

```
aws s3 cp s3://toolbox-dev-runtime/[version]/gapwalk-[version]-dev.tar.gz
```

4. 根據您的專案需求擷取和設定執行時間。

如需更詳細的部署說明和使用準則，請參閱開發和特殊 AWS Blu Age Runtimes 文件。

> **Note**
>
> 請確定您在 上擁有 AWS 帳戶 存取這些儲存貯體所需的 S3 讀取許可。您可以在開發和特殊 AWS Blu Age Runtimes 文件中找到 IAM 政策範例。

AWS Blu Age 版本備註

本節包含自 3.5.0 版起的 AWS Blu Age Runtime 和 Modernization Tools 版本備註，以版本編號整理的最新版本為先。

> **Note**
>
> 如需本文件前的版本備註，請聯絡 AWS Blu Age 交付服務。如需有關最新 Blu Insights 功能的資訊，請參閱 Blu Insights 版本。

主題

- 版本備註 4.8.0
- 執行期 4.8.0 版
- AWS Blu Age Transformation Engine 4.8.0
- 版本備註 4.7.0
- 執行期 4.7.0 版
- AWS Blu Age Transformation Engine 4.7.0
- 版本備註 4.6.0
- 執行期 4.6.0 版
- AWS Blu Age 轉換引擎 4.6.0
- 版本備註 4.5.0
- 執行期 4.5.0 版
- AWS Blu Age 轉換引擎 4.5.0
- 版本備註 4.4.0
- 執行期 4.4.0 版
- AWS Blu Age 轉換引擎 4.4.0
- 版本備註 4.3.0
- 執行期 4.3.0 版
- 現代化工具 4.3.0 版
- 版本備註 4.2.0
- 執行期 4.2.0 版
- 現代化工具 4.2.0 版

- 版本備註 4.1.0
- 執行期 4.1.0 版
- 現代化工具 4.1.0 版
- 版本備註 4.0.0
- 執行期 4.0.0 版
- 現代化工具 4.0.0 版
- 版本備註 3.10.0
- 執行期 3.10.0 版
- 現代化工具 3.10.0 版
- 版本備註 3.9.0
- 執行期 3.9.0 版
- 現代化工具 3.9.0 版
- 版本備註 3.8.0
- 執行期 3.8.0 版
- 現代化工具 3.8.0 版
- 版本備註 3.7.0
- 執行期 3.7.0 版
- 現代化工具 3.7.0 版
- 版本備註 3.6.0
- 執行期 3.6.0 版
- 現代化工具 3.6.0 版
- 版本備註 3.5.0
- 執行期 3.5.0 版
- 現代化工具 3.5.0 版

版本備註 4.8.0

版本日期：2025 年 4 月 23 日

此版本的 AWS Blu Age Runtime 和 Transformation Engines 推出關鍵更新，以增強資料庫轉換功能和效能：

- IBM IMS 資料庫轉換 — 新增將 IBM IMS 資料庫轉換為 JHDB 自訂 Blu Age 解決方案的支援。

- 記憶體內快取功能 — 新增記憶體內快取功能，可讓使用者快取記憶體內的唯讀資料，改善資料密集型程式的效能。

我們使用以下堆疊測試此版本的 AWS Blu Age Runtime。

元件	測試版本
Java	Java 17
Presentation layer	Node JS 22.11.0
	Npm 10.9.0
	Angular 19.1.3
Service layer	Spring Boot 3.4.2
	Spring Core 6.2.2
	Spring Session 3.4.1
	Spring statemachine 4.0.0
Persistence layer	PostgreSQL engine 15.10
	Oracle 21c
Application server	Apache Tomcat 10.1.17

執行期 4.8.0 版

zOS

新功能

- 引進透過 YML 屬性 定義安全交易自訂標頭的支援`gapwalk-application.security.customAllowedHeaders`。此屬性與值為 的`gapwalk-application.identity`屬性相關`oauth`。
- 新增一項功能，以允許客戶根據 Blusam 資料庫資料表中的實際記錄重建 Blusam 中繼資料
- 使用 AWS Secrets Manager 整合的安全 JHDB 連線

改進

- COBOL
  - 改善對 READ 檔案陳述式上 AT END 條件的支援
  - 改善對 MOVE 簽章數字常值到英數欄位的支援，而無需移動符號
  - 改善對 UNSTRING 陳述式的支援，以在找不到分隔符號且沒有值移至其他變數時，處理非數值到數值欄位轉換並保留低值
  - 改善動態連結實體的地址繫結操作
  - 透過包含對傳入/傳出位元組 (SOSI) 的支援，改善字串串連操作
  - 使用 DAY-OF-WEEK 來源改善對 COBOL ACCEPT 陳述式的支援
  - 新增支援 SYSIN 和 SYSOUT 做為 SPECIAL NAMES 值
  - 改善對 STRING 串連處理表單饋送字元的支援
- JCL - IDCAMS
  - 新增使用不含 GDG 參數的基本名稱刪除 gdg 生成的支援
  - 已改善支援將 VOLUME 參數作為 DEFINE CLUSTER 陳述式標準 VOLUMES 參數的變體處理
- JCL - SORT
  - 使用 JOINKEYS 命令新增對 SYMNAMES 的支援
- JCL - IEBGENER
  - 透過引入驗證檢查和自動回復至 ICEGENER，改善對 IEBGENER 處理無效 SYSIN 內容的支援，即使控制陳述式不正確，也能確保持續操作
- JCL - INFUTILB
  - 改善卸載 INTO 和 FROM 子句的處理
- JCL - DSNUTILB
  - 新增對 DOUBLE PRECISION 資料類型的支援
- JCL - 其他
  - 改善在任務的子程序中使用 DD 名稱覆寫檔案組態
  - DSNTIAUL/INFUTILB - 增強了用於卸載公用程式的 SYSREC 填補位元組的自訂
  - 新增 DISPLAY 陳述式重新導向的 OUTDD 選項支援
  - 已改善對固定區塊序列檔案寫入的支援。當 COBOL 程式中的 LRECL 與 JCL LRECL 不同時，JCL LRECL 優先。
  - 新增對 SORTOUT 檔案目錄 DCB 屬性的支援，但僅適用於永久資料集
  - 增強對條件式 JCL 步驟轉換的支援

  - 已改善對 SYSIN 的 ACCEPT 支援，以及結尾空格超過目標大小的行
  - 已改善對 SET 陳述式的支援，以處理不包含變數名稱的命令，內部定義的命令包含應該可在 JCL 內存取的成員檔案，而命令包含內嵌註解
  - 新增支援，以在 JCL 檢查點中繼資料中保留任務層級 'scriptContext' 參數，以及在重新啟動時使用的任務內容
- Blusam
  - 改善載入空資料集後寫入時 Blusam 大型索引資料集上的記錄插入
  - 啟用暖機機制並引入選用的記錄預先擷取時段，以及索引和頁面的本機儲存體，以改善大型資料集的效能
  - 改善對匯出資料集的支援，以使用 處理更大的資料集 AWS Key Management Service
- SQL
  - 改善對 JDBC BLOB 資料類型的支援
- 訊息佇列整合
  - 新增在啟用 XA 時對 MQ PUT 命令上的使用者識別符的支援
  - 新增 XA 啟用時對並行 MQ 連線的支援
- IMS
  - 新增 YML 屬性`jhdb.keepParent`，以控制 IMS ISRT 呼叫在設為 true 時是否在資料庫中填入 _parent 和 _logicalparent 欄位
  - 透過移除無效交易程式碼的訊息佇列來改善穩健性

AS400

新功能

- 新增記憶體內快取功能，可讓使用者在記憶體內快取唯讀資料，改善資料密集型程式的效能。

改進

- CL
  - 改善對 OPNQRYF 命令中 QRYSLT 參數的支援，以處理 RANGE 關鍵字、在字串資料欄純數值時剖析值，以及處理空字串值
  - 新增對 FTP SENDPASV 命令的支援
  - 改善對使用 Qcmdexc 執行之命令 DSPDBR 中查詢形成的支援

- 改善對 MONMSG 例外狀況的支援，以在執行堆疊中正確傳播
- 改善 CPYF 上資料欄計數和資料類型不同之資料表之間的記錄複製
- 改善訊息監控優先順序：命令層級的訊息現在可正確優先於程序層級的訊息
- 改善對 CPYF 命令的支援，以考慮 BLU4IVLIB 定義的程式庫
- 改善命令 CHKOBJ 中 MonMsgs 的支援，以在檔案不存在於程式庫時處理案例

- 螢幕
  - 透過手動建置值並將其轉換為要顯示的數字，而不是讓數字四捨五入，改善處理超過 15 位數的 BigDecimal 值
- Misc (雜項)
  - 已改善在 UserSpace 已存在時，將取代為 NO 的 UserSpace 支援
  - 增強的 BLU4IVLIB 功能，可支援擷取和寫入記錄以覆寫程式庫
  - 改善對使用 DIM 和 OCCURS 操作的多維陣列欄位的支援
  - 增強對 DAO 操作中作為索引鍵欄位值傳遞的低值的支援
  - 移除可由本機和全域監視器處理之訊息的 MONMSG 指令衝突
  - 改善在重複記錄插入時設定 FileStatus
  - 先使用登錄檔檢查使用者空間是否存在，以減少使用者空間鎖定衝突並降低建立失敗風險
  - 已改善對 readPriorEqual() 在 delete() 之後略過記錄的支援
  - 改善了承諾控制下的記錄鎖定機制
  - 改善子常式中 RPG 陳述式 RETURN 的處理
  - 改善對 SharedContext 中使用者資訊的支援

AWS Blu Age Transformation Engine 4.8.0

zOS

改進

- COBOL
  - 改善多個檔案的印表機模式產生。
  - 改善對 SPECIAL-NAMES 的支援，以處理選用的端點點和多個點。
  - 已改善對具有相對路徑 /REPLACE OFF/COPY-REPLACING 陳述式的 COPY 支援。
  - 新增對 COL 做為有效欄位名稱的支援。

- 改善剖析以處理 SPECIAL-NAMES、X-COMPUTER 和 DECLARATIVES 之後的空間。
- 新增對 S01 至 S05 和 C01 至 C12 的支援作為有效的段落名稱。
- 新增對 LENG 函數的支援，做為 LENGTH 函數的別名。
- 改善 cobolMaxCodeWidth 組態值與其 COBOL 包含項目不同時的 Copybooks 剖析
- 已改善對使用逗號和操作剖析表達式的支援，例如 MIN(A * B， (3 + 1) / 2， 3 + 4)。
- 在 PROGRAM ID 子句中新增對 IS INITIAL PROGRAM 的支援。
- 新增對 SEARCH ALL 陳述式上第 88 級條件的支援。
- 新增支援報告部分。

- SQL
  - 新增支援從 IBM IMS 資料庫進行 DDL 轉換。

AS400

改進

- CL
  - 透過將轉換設定為 JAVA 作為預設支援來增強對 CL 的支援
  - 新增使用中繼模型，改善轉換至 groovy 指令碼
  - 新增關鍵字 CONTINUE 和 FOR 至預留關鍵字清單
- DDS
  - 已改善對相對於先前定義之 DSPF 欄位的支援
  - 改善對具有關鍵字 VARLEN 的 DDS 字元欄位的支援
  - 透過在轉換和 updateJPAFromVo 方法中產生重新命名欄位的明確設定，改善 LF 實體 DAOs 的轉換/產生。
- RPG
  - 改善對 PREFIX 關鍵字參數的支援，以處理值沒有引號的情況
  - 改善陣列索引中巢狀方法呼叫的處理
- COBOL400
  - 使用指令 COPY DDS-ALL-FORMATS 改善從 DSPF 匯入的記錄實體的產生
- SQL
  - 已改善具有非封閉 OR 條件的 SQL WHERE 子句轉換。

版本備註 4.7.0

版本日期：2025 年 3 月 12 日

此版本的 AWS Blu Age Runtime 和 Transformation Engines 推出核心相依性的金鑰更新： AWS Blu Age Runtime 已升級為使用最新的 Spring Boot 3.4.2 版，而現代化前端應用程式現在使用最新的 Angular 19 版。

我們已使用下列堆疊測試此版本的 AWS Blu Age Runtime。其他版本也可能相容。

元件	測試版本
Java	Java 17
Presentation layer	Node JS 22.11.0
	Npm 10.9.0
	Angular 19.1.3
Service layer	Spring Boot 3.4.2
	Spring Core 6.2.2
	Spring Session 3.4.1
Persistence layer	Spring statemachine 4.0.0
	PostgreSQL engine 14
Oracle 21c	
Application server	Apache Tomcat 10.1.17

執行期 4.7.0 版

zOS

新功能

- 新增 YML 屬性`dataSimplifier.doubleFloatingType`，以定義編碼和解碼浮點數時要使用的格式 - 允許的值為 `IEEE_754`（預設）/ `HFP`（適用於十六進位浮點數格式）

- 新增對數值 PICTURE 上小數擴展位置 P 的支援

改進

- COBOL
  - 已改善對 AFTER INITIAL 和 INITIAL 子句之前的 INSPECT 陳述式的支援
  - 升級的數值編輯類型支援，搭配最佳化的位元組轉換，可提高整體資料處理效率
  - 透過英數字元比較新增對 FUNCTION MIN & MAX 的支援
  - 改善對負值 FUNCTION INTEGER 的支援
- PL/I
  - 使用 FBA/VBA SYSPRINT 資料集增強對 PUT 陳述式的支援
- JCL - SORT
  - 新增支援 OUTREC 選項 SQZ 搭配子參數 SHIFT、LEAD、TRAIL、MID、PRECLANK 和 PAIR
  - SORTOUT 檔案的增強型 DCB 屬性目錄
- JCL - DSNUTILB
  - DECIMAL EXTERNAL 資料類型的增強支援
  - 改善對與其他資料欄值/位置相關的 NULLIF 條件的支援
  - 改善 BYTEA 資料欄的支援
- JCL - INFUTILB
  - 改善對 UNLOAD 操作中關鍵字 CURRENT 【DATE/TIME/TIMESTAMP】 的支援
- JCL - 改善對稱為 PROCs 之合格傳回碼的支援
- 資料處理
  - 處理從英數字元重新定義為數值類型的空白值欄位時，增強的數值比較邏輯，確保正確評估零
- Blusam
  - 改善對 START 命令的支援，以使用主索引鍵的區段處理部分索引鍵搜尋，提供更靈活的記錄擷取功能
- SQL
  - 新增支援 ON 子句中傳遞的輸入參數
  - 新增對 REPEAT 函數的支援
  - 使用涉及資料欄名稱的指派改善對 UPDATE 陳述式的支援
  - 改善多列游標擷取中的 SQLCODE 和 SQLSTATE 處理

    - 新增對 DB2 資料庫目標的 DB2 XML 函數的支援
- 訊息佇列整合
    - 在 MQ GET 操作中增強錯誤處理，為空白訊息內文提供更清楚的狀態指示 MQCC_FAILED
    - 透過在 XA 交易內容中處理 MQRFH2 標頭，增強對 MQ PUT 操作的支援
    - 改善對 AWS Blu Age Runtime 秘密的支援，以處理 JMS MQ 的其他屬性
    - 增強的 JMS MQ 連線集區組態容量
- Misc (雜項)
    - 系統整合 - 新增對 DFSRRC00 程式呼叫中結構描述環境的支援
    - 新增停用 Blusam 時的 IMS 交易相容性
    - 改善對循序end-of-file條件的支援，以符合舊版規則「EOF 視為讀取失敗」

AS400

新功能

- 新增對 RUNSQLSTM 命令的支援，以執行存放在外部來源檔案中的 SQL 查詢

改進

- CL
    - 透過 ALCOBJ/DLCOBJ 命令新增檔案鎖定機制
    - 改善 CPYF 命令功能，以在檔案複製期間維持記錄序列完整性
    - 改善 CPYF 命令，以支援輸出為一般檔案
    - 增強的參數處理，可在程式呼叫期間自動調整資料長度
    - 改善處理範圍參考的參數支援
    - 新增對 DLTDTAARA 命令上 MONMSG CPF2105 的支援
    - 新增對 DATAAREA 上命令 CHKOBJ 的 QTEMP 程式庫支援
    - 增強對 ADDLFM 命令的支援，改善邏輯檔案成員管理功能
    - 新增對 DLTDTAARA 命令上一般資料區域名稱的支援
    - 新增對 SBMJOB 的支援，以透過 SWS 參數處理傳遞切換
    - 將讀取和寫入呼叫放入同步區塊，以改善資料區域寫入邏輯
    - 增強的資料區域並行管理，以防止透過實作精細鎖定機制同時寫入，消除平行操作期間的錯誤

  - 在 RTVDTAARA 命令的 DTAARA 參數中新增對關鍵字 *ALL 的支援
- 螢幕
  - 改善了前端白色屬性的顏色
- RPG
  - 改善對 TESTB 操作的支援，以更好地處理不同的編碼
  - 已改善對編輯碼 'Y' 和 'P' 之 EDITC 操作的支援
  - 已改善對代碼 K、Q 和 Z 的 EDITCODE 支援
  - 使用參數在本機程序上實作對 Prototype ExtProc 關鍵字支援的支援
  - 外部程序的值傳遞的已處理參數
- COBOL
  - 改善 SORT 陳述式以處理目標程式庫
  - 在輸出模式下開啟關聯的 DAO 時截斷分割區
  - 新增對 PREFIX 關鍵字的支援
- DataQueue
  - 新增 RabbitMQ 資料和混合訊息佇列的金鑰存放區和信任存放區組態選項，在生產部署中啟用安全訊息
  - 透過消除備援資料佇列存在檢查，最佳化伺服器重新啟動效能
  - 透過移除 DataQueueRegistry 和 DataQueueConnectionProvider 元件來簡化架構
  - 透過直接處理來簡化資料佇列刪除，無需進行登錄驗證
  - 透過確保適當的交換建立和queue-to-exchange繫結，改善佇列建立程序
- Misc (雜項)
  - 新增交易請求大小驗證，以防止與超出範圍請求相關的系統過載
  - 改善任務資源清理的鎖定管理
  - 改善了 DAO 鎖定機制，以防止在 INPUT 模式下開啟檔案時鎖定記錄
  - 最佳化動態實體映射的效能
  - 改善 JPA 持續性內容的處理，以增強 DAO 操作的效能
  - 透過新增時間戳記資料欄來處理記錄中繼資料，改善資料區域支援
  - 透過實作同步的讀寫操作來改善資料區域可靠性，以防止並行存取問題
  - 改善對 DateHelper#moveDate 的支援date-to-timestamp轉換
- DDS

  - 改善與 SST 或 CONCAT 等合成操作結合的 Select/Omit 條件支援
- SQL
  - 最佳化 SQLExecutorBuilder 以更有效率地處理部分記錄擷取，特別是在擷取超過一個擷取限制的多列時

AWS Blu Age Transformation Engine 4.7.0

zOS

改進

- COBOL
  - 新增對 CODE-SET 陳述式的支援
  - 新增對 WRITE 陳述式中選項 NOT END OF PAGE 的支援
  - 新增對 ALTER 上新語法的支援
  - 新增支援 COPY REPLACING 陳述式的多行取代
  - 改善不同子句定義中隱含的 CHARACTERS 關鍵字剖析
  - 改善了當金鑰名稱重複但父系不同時，對輸入輸出區段中涉及之備用金鑰的支援
  - 透過比較大型欄位與空格，改善對 IF 條件的支援
  - 改善邏輯檔案參考靜態解析度，使其不區分大小寫
  - 增強程式碼轉換，以自動為每個包含 LINAGE 子句的 FD 項目產生 LINAGE-COUNTER 隱含欄位
  - 改善對 IDCAMS 叢集定義的支援，其名稱由簡單引號括住
  - 改善對基本格式的 PERFORM 陳述式支援，其 TIMES 片語格式和格式涉及區段的資格
  - 啟用具有多個 GIVING 檔案的 SORT 陳述式轉換，並新增例外狀況，以在執行時追蹤此案例
  - 改善使用 REDEFINES 子句在群組上指定時的 RENAMES 轉換
- PL/1
  - 新增支援 PUT STRING 的 DATA 或 LIST 選項
  - 新增對多項目程式支援的支援

AS400

改進

- CL

  - 改善 QCMDEXC 參數的剖析，使其能夠透過 QCMDEXC 執行 ALCOBJ/DLCOBJ
  - 新增對 CL 上 ElseIf 陳述式的支援
  - 新增對內建函數 %SWITCH 的支援，以設定任務切換。
- DDS
  - 改善對程式描述的 DSPF 記錄的支援，以處理輸入規格中的虛擬記錄
  - 在 CREATE INDEX SQLDDS 檔案的情況下，改善了 DAO 世代 WHERE 子句條件
- RPG
  - 新增對未簽署整數上 ZEROES 初始化 INZ(*ZEROS) 的支援
  - 改善對命令 EXCEPT 與工作站和 SQL 索引檔案類型的支援
  - 改善對外部定義之數值欄位 RPG 內部資料類型的支援
  - 改善對內建 %ERROR 的支援，以在對檔案執行 操作後取得目前的錯誤狀態
  - 改善對 COMMIT 檔案關鍵字的支援，以處理多重格式邏輯檔案
  - 增強多格式邏輯檔案的 SELECT/OMIT 處理
  - 改善處理 MOVEA，以使用空白設定陣列，將已封裝、區域和二進位的固定陣列移至相同類型的另一個陣列。
  - 改善 Z-ADD *ZEROS 和 MOVEA *ZEROS 的陣列處理
  - 改善從數值和字元組合移動時的 MOVE/MOVEL 操作
  - 使用參數在本機程序上實作對 RPGLE Prototype ExtProc 關鍵字支援的支援
  - 改善指派陳述式中 *HIVAL 圖形常數的處理
  - 改善對 PREFIX 關鍵字處理角色替換的支援
- COBOL400
  - 改善動態存取模式上 REWRITE 和 UPDATE 操作的支援
  - 新增對 CBLLE 中 ACCEPT 陳述式 FOR 子句的支援
  - 改善對內建 %STATUS 的支援，以在 操作後取得檔案的目前狀態
- Misc (雜項)
  - 改善了轉換程式，其定義與指定 PF/LF 中其中一個欄位共用相同名稱的資料結構
  - 改善 kanji 變數的陣列存取產生

版本備註 4.6.0

版本日期：2025 年 1 月 24 日

我們已使用下列堆疊測試此版本的 AWS Blu Age Runtime。其他版本也可能相容。

元件	測試版本
Java	Java 17
Presentation layer	Node JS 22.11.0
	Npm 10.9.0
	Angular 18
Service layer	Spring Boot 3.3.5
	Spring Core 6.1.14
	Spring statemachine 4.0.0
Persistence layer	PostgreSQL engine 14
	Oracle 21c
Application server	Apache Tomcat 10.1.17

執行期 4.6.0 版

zOS

改進

- COBOL
  - 增強WRITE ADVANCING的功能可提高循序檔案行寫入的準確性，支援多個內容 (BEFORE > 、AFTER和隱含使用 ） 和完整的PAGE陳述式實作
  - 增強對巢狀資料表 FILLER 用作將資料表作為子系的群組之案例FILLER的支援
  - 改善對一個區段內不明確父子的存取
  - 新增對數字編輯類型的支援，其中圖片='-----'
  - 改善 BINARY 類型資料的顯示處理
- PL/I
  - 改善指派陳述式中二進位常值的轉換

- JCL — SORT
  - 改善對相同OUTFIL陳述式中連續OVERLAY參數的支援
- JCL — DSNUTILB
  - 最佳化載入機制，使資料擷取時間加快 25%
  - 增強對外部業務資料來源 XA 交易的支援
- JCL — INFUTILB
  - UNLOAD - 新增對 FLOAT8 資料類型的支援
- JCL — IDCAMS
  - 最佳化IDCAMS處理命令的傳回碼
  - 新增根據 GDG 基本名稱刪除所有 GDG 世代的支援
  - 新增不使用 NONVSAM 參數刪除檔案的支援
- JCL — 其他
  - 增強型批次重新啟動中繼資料處理，以改善重新啟動模式期間的工作流程狀態管理
- Blusam
  - 新增支援 Ehcache 和 Redis 實作中的 TTL for Blusam 快取
  - 改善對 Blusam KSDS 檔案 COBOL 檔案描述DEPENDING ON欄位FD的支援
  - 增強 Redis Blusam 讀取操作中的執行緒安全性，可同時執行多工作
  - 改善了 Blusam 結構描述的建立，以提高資料庫使用者權限的穩定性
  - 改善變數區塊串連輸入資料集右側的填補 READ
- BAC
  - 新增支援在多結構描述模式中建立資料集，包括新的「結構描述」資料欄，以指示每個資料集的結構描述關聯
- MFS
  - 改善使用者資訊從前端傳播到共用內容，確保適當傳播到 JHDB 內容
  - 新增對 XA 交易上 IBM MQ IMS 資訊標頭的支援
- SQL
  - 當所有資料欄值為 NULL 時，增強SQLCODE處理以在游標擷取期間設定 305
  - 新增對涉及WHERE條件OCCURS參數的 IN 子句的支援
  - 新增對DECLARE GLOBAL暫存資料表陳述式的支援
  - 根據目標DB2引擎，在執行時透過專用轉換，延長對午夜 24H DB2 特定時間戳記格式的 DB2 SQL 支援

- Misc (雜項)
  - 增強的 IBM930 字元集，允許 Unicode 字元 U+2014 和 U+2015 對應至 EBCDIC 中的 x'44x4A'
  - TDQUEUE - 重構 SQS 實作以支援多執行緒
  - 改善 GDG 資料集名稱解析，以允許客戶封存具有相同 GDG 字首的檔案 （例如，A.B.C.G0002V00 是目前檔案， A.B.C.G0001V00.1236是封存檔案）
  - 增強SQLConverter::toPgmDate/Time/Timestamp以根據舊版格式對齊日期運算

AS400

新功能

- 新增對平面檔案和重複實體動態建立的 AS400 資料表的支援，允許存取透過 CRTPF、CRTDUPOBJ 和 CPYF 等 CL 命令建立的資料表
- 新增服務，透過處理每個資料表預設程式庫的登錄檔來支援程式庫清單

改進

- CL
  - CLRPFM - 改善呼叫 QTEMP 程式庫命令時的成員處理
  - SMBJOB - 改善對 參數的支援PARM，以處理動態建置的引數
  - CPYFRMIMPF - 新增對參數 TIMFMT、 ERRRCDFILE和 的支援 ERRRCDOPT
  - CPYFRMIMPF - 改善對包含單引號的資料庫英數值的支援
  - CPYF - 使用 改善多成員FROM檔案的命令查詢建構 TOMBR(*ALL)
  - CPYF - 增強支援處理 的 FMTOPT 參數 MAP DROP
  - CPYTOIMPF - 改善FROMFILE對處理資料表MEMBER的參數支援
  - RTVUSRPRF - 新增對 參數的支援 RTNUSRPRF
  - DSPDBR - 大修命令，以符合列印資料表上存在之檢視的預期舊版行為，以及其所屬的程式庫和成員
  - DSPFD - 改善對 參數的支援 FILE
  - DSPFD - 改善對參數TYPE MBR輸出的支援，以包含其他值：mbfile、mblib、mbfcdt、mfccn
- 螢幕
  - 改善 的游標位置優先順序 DSPATR(PC)
  - 忽略「受保護」欄位的前端驗證，以改善子檔案記錄欄位的驗證

- 改善在具有多個陣列欄位共用元件名稱的工作站中初始化記錄的支援
- 增強對DSPF關鍵字 (SFLMSG、 SFLMSGIDCHANGE和 命令索引鍵） 中回應指標的支援

- RPG
  - 增強的程式週期支援更好地處理從主要/次要檔案讀取的欄位
  - 新增支援主要/次要檔案讀取的分割控制欄位
  - 增強%SUBST的內建方法，可處理比較陳述式中的雙位元組欄位
  - 改善對 MVR 操作的 ZERO 指標支援
- DDS
  - 新增對記錄格式參照相同實體記錄的多重格式邏輯檔案的支援
- DataQueue
  - 透過在中斷期間清除取用者，改善等待資料佇列訊息之任務的任務中斷處理
  - 從 RabbitMQ 遷移至 Spring-AMQP，以獲得更好的頻道管理和執行緒擴展
- Misc (雜項)
  - 改善 SQLExecutorBuilder，以支援具有多個空格的查詢，並開啟沒有前置空格的括號
  - 改善 DAO 支援，以在變更讀取方向時正確處理游標位置
  - 擷取和刪除操作後精簡金鑰初始化，以確保在插入更新的記錄之前正確移除相關記錄
  - 最佳化 DAO 映射器產生的程式碼，以改善時間執行效能

AWS Blu Age 轉換引擎 4.6.0

zOS

改進

- COBOL
  - 使用選用AREA/AREAS常值改善子RESERVE句的剖析
  - 使用選用DATA DIVISION宣告增強 COBOL 支援，支援簡化的測試案例
  - 透過新增對 ALPHABET、 和 CLASS子句SYMBOLIC、交換器和FORMFEED變數的支援來改善特殊名稱段落
  - 在 ACCEPT陳述式中新增支援 SYSIN做為 Mnemonic Name
  - 增強子PICTURE句支援邏輯大小計算中的 "$"、"0"、"CR"、"DB" PIC 符號
  - 改善多個檔案案例的USE陳述式轉換

  - 多個變更的增強型ALTER陳述式轉換
  - 在 delimited by子句ZEROHIGH-VALUELOW-VALUES中新增對圖形常數的支援
- SQL
  - 改善 postgreSQL 目標預設值的轉換，以處理CURRENT_TIMESTAMP預設值周圍的引號
  - SQL 檢視的處理WITH CHECK OPTION子句

AS400

改進

- DDS
  - 改善對多次參考相同實體記錄的多重格式邏輯檔案的支援
- RPG
  - 增強 MOVE和 MOVEL操作，以更好地處理填補零
  - 在評估和條件中增強處理巢狀函數呼叫
- COBOL400
  - 新增在SELECT陳述式中轉換IN關鍵字的支援
  - 改善對資料描述項目中遺失點的支援，使其與遺失時假設點的最新 COBOL 版本保持一致
  - REWRITE 操作上的增強型游標定位
  - 增強對 START陳述式的支援，以將記錄鎖定在目前的檔案位置
  - 改善對編譯器指令的支援COPY DDS，以產生所有輸入/輸出資料結構
- Misc (雜項)
  - StateMachines - 改善轉換以增強複合狀態宣告，以符合 stateless4j 範例
  - 已改善包含特殊字元之 LF 檔案的淨化
  - 已改善*ALL對具有十六進位值之圖形的支援
  - 改善從數值到字元類型的隱含轉換MOVE的操作支援
  - 最佳化報告產生依相關聯的印表機名稱排序，防止重複或衝突的名稱
  - 改善了 與 EXTFILE結合的關鍵字支援USROPN，以處理常值和格式 libname/filename

版本備註 4.5.0

版本日期：2024 年 12 月 20 日

此版本的 AWS Blu Age Runtime 和 AWS Blu Age Transformation Engines 包含下列主要功能。

- JCL 支援 — 現在可在執行時間內容中即時產生和執行 JCL 指令碼。此功能可在批次任務處理中增加彈性和自動化。我們已使用一組 SORT、ICETOOL、INFUTILB 和 IDCAMS 的改進功能，更新了執行時間對 JCL 公用程式的支援（請參閱以下章節的詳細資訊）。這些增強功能提供更強大且更有效率的資料處理功能。
- 繫結目錄和啟用群組 支援 AS/400 現代化應用程式 — 繫結目錄可透過管理匯出的程序參考來增強系統組織，同時啟用群組可簡化執行內容管理。這些功能可改善精確度和可靠性、強大的資源管理，以及最佳化的系統互動。結果是更具彈性、有條不紊且高效率的系統，適用於現代化 AS400 應用程式。
- 相依性更新：將所有前端架構 (BAC/JAC 和現代化應用程式） 更新為長期支援 (LTS) 版本。將 Angular 從 v17 更新至 v18 引入新的回應模型和簡化的狀態管理，降低複雜性並改善開發人員的應用程式維護。Node.JS 也已從 v20 更新至 v22。

我們已使用下列堆疊測試此版本的 AWS Blu Age Runtime。其他版本也可能相容。

元件	測試版本
Java	Java 17
Presentation layer	Node JS 22.11.0
	Npm 10.9.0
	Angular 18
Service layer	Spring Boot 3.3.5
	Spring Core 6.1.14
	Spring statemachine 4.0.0
Persistence layer	PostgreSQL engine 14
	Oracle 21c
Application server	Apache Tomcat 10.1.17

執行期 4.5.0 版

zOS

新功能

- JCL — 新增從線上程式調用批次任務的能力。當現代化程式即時產生 JCL 指令碼時，我們新增了一項服務來處理存放在專用 TDQueue 中的 JCL 指令碼。此服務可讓您重建 JCL 訊息、將此訊息重構為 groovy 指令碼，並執行此 groovy 指令碼。
- ADABAS — 新增對 ADABAS 計劃的支援。透過此支援，執行時間會模擬資料庫存取的 ADABAS 命令 （僅適用於 Oracle)。

改進

- COBOL
  - 利用 NO ADVANCING 選項增強對 DISPLAY 陳述式的支援
  - 提高貨幣符號管理的準確性，讓使用者從更準確的轉換 COBOL 結構中受益
  - 改善將未簽署欄位移至已簽署欄位時對值指派的支援，反之亦然
  - 改善對 GDG 檔案和串連檔案區塊大小的支援
- CICS
  - 新增對 OpenStatus 和 EnableStatus of Blusam 資料集的支援
  - 新增對 `SET DATASET`命令的支援
- JCL — SORT
  - 改善資料集記錄大小的處理
  - 改善對 `OUTFIL`陳述式的支援，以根據 STARTREC 和 `ENDREC` 選項中指定的值，產生僅包含輸入檔案記錄的輸出檔案
  - 增強對 `OVERLAY` 陳述式的支援
  - 已改善對 `OUTREC`陳述式處理`EDIT`選項變體的支援。除了 之外`EDIT(...)`，我們現在還支援 `EDIT=(...)`
  - 新增對算術操作中 模式 (p、m、f、OPERATOR、p2、m2、f2) 的支援
  - 您可以從 JCL 使用`SORT`程式`DUMMY`的檔案子句來處理空的輸入檔案，並受益於空的檔案產生
- JCL — ICETOOL
  - 透過 `SORT`程式增強對 `SORT FIELDS=COPY`陳述式的支援
- JCL — INFUTILB

  - 如果未在 JCL 中指定，且屬性 DFSIGDCB 已停用，則改善對記錄大小運算的支援
  - 更新精確度並根據 擴展為子句欄位，以 INTO 子句改善 DECIMAL 的 UNLOAD
  - 改善 VarcharFormatter 中的格式方法
  - 使用新的可設定選項增強支援，允許使用者控制在資料卸載期間如何處理 VARCHAR 欄位，因為填補行為可確保資料擷取程序的靈活性和準確性。
- JCL — IDCAMS
  - 改善刪除具有萬用字元尾碼的檔案，以及直接由括號或簡單引號括住的名稱
  - 提高準確性以利用 MAXCC 傳回程式碼
- JCL – IKJEFT01 - 新增功能旗標 `systin.encoding` (預設值 = `ASCII`)，以支援 SYSTSIN 檔案資料集的特定編碼
- JCL — 改善對 JCL 步驟和後續步驟中產生輸出檔案的 BDW 屬性的支援，使用相同的檔案系統做為輸入和 `DISP=PASS`
- MF
  - 已改善對記錄序列檔案的 2 位元組標頭的支援
  - 改善處理 DELETE 命令的傳回碼
  - 改善記錄序列檔案的寫入提升行
- Redis
  - 改善 JCL 檢查點和 Jics TSQueues 的 Redis 範本初始化
  - 改善 Redis 資料集記錄鎖定資訊的可存取性和可讀性
- SQL
  - 使用 REFERENCES 子句改善 FOREIGN KEY 的剖析
  - 提供可擴展的快取功能，可將原始舊版圖形類型存放在資料庫中，增強資料可追蹤性並促進圖形運算
  - 增強對跨執行時間公用程式的 SQL 查詢 CASE WHEN 模式的剖析支援
  - 改進了 SQL Postgres Blu Age 內建函數 gwdecimal，執行時間依賴 以符合 DECIMAL DB2 內建函數。
- Misc (雜項)
  - 使用 SIGN 運算元增強對 NumericEditedType 的支援
  - 改善現代化應用程式中 SpringBootLauncher 的主要資料來源組態產生
  - 增強靈活性，可將應用程式日誌與與呼叫任務相關的路徑隔離。
  - 改善比較 NumberUtils 欄位時對空白值的支援

- FILE — 改善對基礎檔案中變數區塊資料集的支援
- MQ — 改善了 MQ 連線管理，以實現高可用性環境
- 新增對非 JMS 用戶端的支援，以增強編碼和字元集處理，進而改善 MQ 佇列相容性
- 改善對 Ebcdic 檔案的 ANSI 控制字元的支援

AS400

新功能

- 新增對 Bound 程式中匯出資料的支援
- 新增將 ILE 特定支援除以零

改進

- COBOL400
  - 改善檔案狀態中 EOF 的支援
  - 提高 Cobol START 陳述式的精確度支援 EQUAL 關鍵字到 KEY IS 子句
- CL
  - 新增對命令 UPDENVPARM 的支援
  - CRTPF - 新增對使用分割區存取之資料表的支援
  - RCVF - 改善對具有覆寫的邏輯檔案的支援
  - FTP - 改善對具有 OVRDBF 和增強型 OUTPUT 日誌的邏輯 I/O 檔案的支援，並新增對工作目錄中 I/O 檔案的支援
  - CPYFRMIMPF - 新增對參數 ERRRCDFILE、TIMFMT、 的支援 ERRRCDOPT
  - CPYF - 改善 QTEMP 分割區建立
  - CPYF - 新增 *FROM 檔案為空時的監控訊息
  - OVRPRTF - 新增對新參數的支援：PAGESIZE、OUTQ、DEV、LIP、CPI、OVRFLOW、LVLCHK、FORMTYPE、 HOLD
  - 將 FMTOPT 參數與 CPYF 命令中的 MAP和 DROP選項搭配使用時提高準確性，以允許將具有額外資料欄的資料從來源檔案複製到目標檔案
  - 提高在 RMVLNK命令中管理檔案系統路徑萬用字元模式映射的準確性
  - RMVM （移除虛擬機器） 命令已增強，可處理DROP分割區資料表，確保完整清除相關資源。
  - OPNQRYF - 改善對命令的參數 *FILE 的支援

  - 實作 CPF0000 處理，以涵蓋所有 CPFx 訊息
  - CHGDTAARA - 新增對 *ALL 關鍵字的支援，以變更整個資料區域
- 螢幕
  - 透過提高捲動和游標位置/優先順序的準確性來改善資料表/子檔案顯示
  - 改善非數值CHECK(RZ)和非簽署欄位的 和 CHECK(RB)功能
  - 改善支援關鍵字的說明畫面功能 HLPARA
- RPG
  - 改善對內建 的支援 %SubDt
  - 改善使用外部描述的本機資料結構的程序支援
  - 新增對選用錯誤碼參數 QMHSNDPM、 QMHRMVPM 和 的支援 QMHRCVPM
  - 增強對%SUBST內建方法的支援，以更好地處理雙位元組欄位。
  - 新增對內建 %TLOOKUP 及其變體 (%TLOOKUPGE、%TLOOKUPGT、%TLOOKUPLE、%TLOOKUPLT) 的支援
- 資料區域
  - 改善在 factor1 為空白時的 OUT 操作支援
  - 改善相同資料區域的並行讀取
  - 新增組態變數blu4iv.dtaara.library.disable以停用資料區域的程式庫
  - 擴展對透過資料區域操作利用具名程式庫的支援，讓使用者能夠視需要建構資料區域位置。
- DataQueue
  - 改善 RabbitMQ 頻道用量
  - 改善 RabbitMQ 取用者僅嘗試取消取用者一次
  - 改善資料佇列從 RabbitMQ 擷取，只需在等待時間為 0 時嘗試 basicGet
- Misc (雜項)
  - 使用者空間 - 改善多個任務同時嘗試擷取相同使用者空間時的行為
  - 改善對承諾控制下未遞交記錄刪除的支援
  - 實體 - 改善對連續省略的支援，因為 OMIT 具有隱含AND意義
  - 新增對實體映射器設定器中駱駝案例的支援，以處理透過額外重構定義的名為 的慣例
  - 改善透過整個應用程式從 AS400 環境交易傳播使用者資訊。
  - 在發生中斷情況時，改善了在結束由 Square 排程的任務時的準確性。
  - 改善承諾控制支援，使其成為程式範圍

AWS Blu Age 轉換引擎 4.5.0

zOS

改進

- JCL - 根據 LISTCAT 剖析改善 KSDS 資料集的雜湊產生
- COBOL
  - 改善剖析COPY-REPLACING陳述式，以便在出現此子欄位名稱的模棱兩可時，處理取代合格的子欄位
  - 改善對 SPECIAL-NAMES陳述式中SYSOUT定義 的支援
  - 改善對 ADD n TO ZERO陳述式中圖形化 ZEROES 的支援
  - 透過扁平化多行索引鍵和文字區塊，改善對 REPLACE陳述式處理多行問題的支援
  - 使用 GIVING子句改善對算術操作 ADD/SUBTRACT/MULTIPLY/DIVIDE 的支援
  - 啟動了對 REPORT 部分及其相關動作的剖析支援 (INITIATE、TERMINATE、GENERATE 報告）
- 其他 - 改善天氣報告的產生和穩定性

AS400

改進

- DDS
  - 改善對類型 DATE 隱含長度的支援
  - 改善對關鍵字 EDITWORD 上 stop-zero-suppression 字元的支援
  - 已改善對資料欄名稱 DESC 的支援，因為它在資料庫中是保留字
- RPG
  - 改善對內建 %TIME 的支援
  - 改善 EVALR 陳述式的產生，以處理從字串值到較短長度變數的指派，以及更好的右側調整
  - 選項設定的增強型 SQL 剖析
  - 改善對 NOMAIN RPGLE 程式中 PSDS 初始化的支援
  - 改善對關鍵字 LIKE 的支援，將 DDS 數值欄位定義為已封裝，無論其外部描述為何
  - 透過將「$」取代為「DL」來改善檔案名稱的清理
  - 改善對內建 %SUBST 的支援，以處理雙位元組值

- COBOL400
  - 畫面 - 改善對 I/O 操作的 DSPF 記錄支援
- CL
  - 改善預留變數名稱的重新命名
  - 改善對 Select/Omits 條件的支援，以處理多種格式的檔案
- Misc (雜項)
  - 減少檔案操作 (EOF、FOUND、EQUAL) 的重複實體
  - 已改善產生 QPRINT 的 JRXML 檔案，這是 AS/400 上的標準印表機。使用時，建立的 JSON 檔案不會包含對程式或檔案的任何參考。只會產生一個 JRXML 檔案 (QPRINT-QPRINT.jrxml)
  - 改善顯示來自程式佇列訊息之元件的其他訊息資訊

版本備註 4.4.0

版本日期：2024 年 11 月 13 日

此版本的 AWS Blu Age Runtime 和 Transformation Engines 著重於升級關鍵相依性和支援的技術，同時提升多種功能的效能。此版本中的一些主要功能和變更包括：

- 相依性更新：主控台應用程式 (BAC 和 JAC) 和現代化應用程式現在正在引導 5 上執行。 AWS Blu Age Runtime 現在採用 Spring Boot 3.3.5 架構。
- 效能：改善狀態機器執行的效能（最多快 10 倍），這要歸功於將 Spring State Machine 程式庫從 2.5.1 版升級至 4.0.0 版之後克服效能降低的新實作。此升級不是選用的，因為 2.5.1 版本不再維護，且包含關鍵和高 CVEs。它包括在平台上對新程式庫的執行期狀態機器實作，具有輕量且高效率的狀態機器實作，不含 CVE，且整體效能更佳。
- 簡化資料庫存取：完成了用於存取資料庫的元件的重大大修，包括 DAOs、JPA 實體、DS DataSimplifier 實體和 Mappers。此重新設計的驅動因素是需要為 AS400 專案中常見的 OVRDBF（覆寫資料庫檔案）功能提供更好的支援。它允許 使用所產生程式碼的簡化架構來處理更多案例。

我們使用以下堆疊測試此版本的 AWS Blu Age Runtime。其他元件版本也可能相容。

元件	測試版本
Java	Java 17
Presentation layer	Node JS 18.18

	Npm 9.8
	Angular 17
Service layer	Spring Boot 3.3.5
	Spring Core 6.1.14
	Spring statemachine 4.0.0
Persistence layer	PostgreSQL engine 14
	Oracle 21c
Application server	Apache Tomcat 10.1.17

如需此版本中包含之變更的詳細資訊，請參閱下列各節。

執行期 4.4.0 版

zOS

新功能

- COBOL - 新增對 JSON GENERATE 陳述式的支援
- COBOL - 新增對控制區塊的支援
- MF - 新增對 FCDREG 編譯器指令的支援
- Blusam - 新增功能 VSAM 檔案集，其中包含以資料庫結構描述為基礎的實作 - 僅支援 PostgresSQL
- Blusam - 新增處理 Blusam 快取資料項目 (Redis 快取引擎） TTL （存留時間） 的支援
- JCL - IDCAMS - 新增了新屬性`idcams.encoding.forced`，以強制用來解碼 SYSIN 卡的字元集
- JICS - 從 擴充 `jics.db.dataScriptLocation` 屬性`application-main.yml`，以接受檔案和資料夾路徑的清單。清單的順序很重要。第一個 SQL 檔案會先執行，依此類推。執行資料夾時，其包含的 SQL 指令碼不會有明確的順序執行。
- 新增對 CEE3ABD 公用程式的支援

改進

- Blusam - 為使用 PostgreSQL 引擎的客戶改善從傳統大型資料集到 Blusam 的載入時間和記憶體使用量（我們觀察到大型資料集的載入速度增加高達 8 倍）
- Blusam - 改善支援登入資料的 exportDataSetToS3 API
- Blusam - 改善 LISTCAT 上傳檔案以建立資料集
- Blusam - 使用明確 KEY 改善對動態 READ 的支援
- Blusam - 改善了幕後寫入機制邏輯
- JCL - 增強的 JES 支援可改善平行執行中的檔案鎖定
- JCL - 新增對 陳述式的支援 `INCLUDE MEMBER`
- JCL - DNSUTILB - 改善支援在主索引鍵包含空格時處理特殊情況的重複索引鍵
- JCL - DSNUTILB - 改善 LoadTask 以在載入 GRAPHIC 資料時最佳化效能
- JCL - INFUTILB - 新增了對 `chunksize`未定義`fetchsize`時的支援
- JCL - INFUTILB - 改善對查詢傳回空白結果集的支援
- JCL - INFUTILB - 改善在 CHUNK 中處理資料時的穩健性
- JCL - INFUTILB - 改善對具有 nullable 欄位的卸載的支援
- JCL - INFUTILB - 已改善對數值類型的支援
- JCL - INFUTILB - 改善 Nullable 欄位的卸載
- JCL - SORT - 改善對 OUTREC 語法的支援
- JCL - SORT - 改善了陳述式 DATE1 的剖析
- JCL - SORT - 改善對 RDW 的 INREC PARSE 子句的支援
- JCL - SORT - 使用編輯遮罩改善欄位格式
- JCL - SORT - 已改善對 OUTREC 中 'SubString' 的支援
- JCL - SORT - 改善對 MF 相容卡的支援
- JCL - UNLOAD - 透過 Postgresql 改善對欄位大小的支援
- JCL - IDCAMS - 透過引入大量模式來改善檔案載入 VSAM 資料集的效能
- PL/1 - 增強對 NumericEditedType 格式的支援，以防止擴展差異
- IMS - 已改善對 NodeSorter 中 IMS Database _right 欄的支援
- CICS - 改善具有 `SET`和 否 `RECEIVE MAP` 的命令 `INTO`
- BMS - 改善對欄位初始值的支援
- SQL - 改善`ddMMMyy`模式的 DateTimeFormat 剖析
- COBOL - 改善在取得值時未考慮小數點時對 NumericEditedType 值的支援
- 改善對行序列檔案中讀取變數長度欄位的支援

- 改善對 GDG 檔案資料集目錄中記錄大小繼承的支援
- 允許可自訂的進階行，改善列印報告的支援
- 已改善變數區塊 (VB) 檔案的記錄資料的初始化

GS21

新功能

- 畫面 - 新增對 PSAM 檔案的支援
- 畫面 - 新增對 ATTR2 的支援
- 新增對 AIM （進階資訊管理員） 生態系統的支援。
- 新增 AIM 中的 PED 支援

改進

- 改善 BitUtils 簽章來處理 RangeReference
- 改善對 DummyFileConfiguration 新增 recordSize / rdw / bdw / blksize / blkszlim 屬性的支援
- 改善對 VPOINT 陳述式的支援，以處理找不到記錄的情況
- 在存取記錄位元組陣列時新增了穩定性
- 改進的 JEF 字元集字元映射
- 改善在 JDBC 映射中處理陣列和條件的支援
- 改善對不同 NDB 陳述式中 SQL 請求的支援，使用 SQL 查詢每個部分的常數更好地處理 SQL 語法的變化。
- 已改善對 GS21 PackedType 最後一個小數為 C、D 或 F 的數值驗證支援
- 畫面 - 改善對 SPA 和 ENTER 的 ACSAPI 和 DefaultPsamController 的支援
- 畫面 - 增強對 ACSAPI 和 NDB 動詞的支援

AS400

新功能

- 新增對多記錄格式資料庫檔案的支援
- 重新設計 AS400 資料庫存取架構
  - 有關檔案覆寫的增強功能

  - 移除過時的元件並降低複雜性
  - 簡化從舊版程式產生的程式碼
  - 將 DAOCycleManager 元件整合到 Blu4iv 外掛程式中，讓我們能夠利用自訂執行時間的 AS400-specific功能。
- JOB - 改善對任務管理 (Quartz) 的支援，以新增中斷任務/任務群組的能力。新增 REST API 端點以中斷具有指定執行 ID 的任務（每個任務都是唯一的，因為它是主索引鍵）。成功中斷後，執行時間會將任務的狀態更新為「INTERRUPTED」。
- 新增公用程式程式 CEERAN0 的支援
- 新增對被動模式的支援。新增 YAML `configuration gapwalk-application.cl:ftpservice:passive`以啟用被動模式
- 新增建立 QTEMP 工作階段和延遲 QTEMP 清除的功能
- 新增對編譯功能 BNDDIR 的支援，以定義程式之間的明確相依性
- 新增對 Activation Groups 機制的支援

改進

- CL - 改善程式訊息佇列上的 RMVMSG 命令，以處理 *PREV 關鍵字
- CL - 改善對 OPNQRYF 中覆寫的支援
- CL - 新增對 RTVMSG 命令 MSGLEN 和 SECLVLLEN 參數的支援
- CL - 改善對 CRTDUPOBJ 的支援，以在未傳遞 NEWOBJ 並新增一般資料表名稱的支援時管理案例
- CL - 改善對 FTP 處理參數 GET、RMTSYS 和 BINARY 的支援
- CL - 改善 CLRPFM 查詢效能，並新增使用 TRUNCATE 而非 DELETE 的選項
- CL - 改善 SBMJOB 以正確處理 USER 參數，以在提交任務時將其用作 USER
- CL - 改善 DLTOVR 命令支援以處理 *ALL
- 資料區域 - 新增例外狀況處理的記錄，改善對 Blu4DataArea 的支援
- 資料區域 - 改善對 Blu4DataArea 的支援，以為每個執行緒擷取新的 DataAreaDao 執行個體
- 資料區域 - 改善資料區域鎖定，避免在記錄層級鎖定，改用新實作的鎖定機制
- 資料區域 - 當未取得鎖定並提供錯誤指標時，資料區域寫入操作現在會繼續執行
- 報告 - 改善對列印報告的報告輸出路徑/命名慣例的支援。允許客戶自訂報告輸出路徑和名稱。客戶可以指定自己的路徑和命名慣例，而不會影響任何其他專案。
- JOB - 改善對任務管理 (Quartz) 的支援，以在任務終止異常時更新任務狀態。例如：Tomcat 的「關閉」或「異常關閉」

- 畫面 - 改善了對具有減號的編輯字詞欄位的數值的處理
- 畫面 - 改善僅具有 titleColorTop 的轉譯快顯視窗
- 畫面 - 改善在找不到一般說明項目時，擷取處理案例的說明資訊支援
- 畫面 - 改善在子檔案訊息行上按下 F1 時顯示「其他資訊」畫面
- 畫面 - 改善顯示 SFLMSG 的訊息行頁尾
- 畫面 - 改善前端，以在新記錄與記錄重疊時完全移除記錄
- 佇列 - 改善 RabbitMQ 訊息擷取，以使用較少的資源
- 佇列 - 改善 RabbitMQ 資料佇列實作，一次只能擷取一則訊息。
- SQL - 針對動態 CREATE 和 DROP 資料表查詢改善 SQLCODE 的 SQLExecutorBuilder 處理
- SQL - 改善對查詢的 OVRDBF 支援
- SQL - 改善 SQLExecutorBuilder，以便將 OVRDBF 覆寫套用至預備陳述式
- RPG - 已改善對程式描述磁碟檔案的輸入和輸出規格的支援
- RPG - 改善對主要和次要檔案讀取搭配 MR （配對記錄） 指標的支援。已改善具有相符欄位之 DAO 週期的擷取順序。
- RPG - 改善對主要和次要檔案的支援。改善更新主要檔案和輸出次要檔案更新/寫入程式碼產生。
- RPG - 新增對自由格式 RETURN 陳述式的支援
- RPG - 改善數值小數指派的轉換和執行時間處理，
- RPG - 改善二進位變數的產生
- RPG - 改善對 EDITC 的支援
- RPG - 改善本機資料區域的處理
- 改善對多個裝置類型 (DISK、WORKSATION、PrintER) 共用的 DDS 欄位的支援
- 改善覆寫處理，讓 PFs 上的覆寫不再影響 LFs
- 改善 Blu4ivWebController 不會將使用者名稱和 userid 重設為預設值
- 改善讀取方向變更時記錄讀取期間的索引調整
- 改善更新/刪除操作後記錄讀取上的游標放置
- 改善讀取方向變更時對多實體 DAO 讀取的支援
- 改善對使用者空間的支援，以避免所有執行緒重複使用執行個體，而不是每個執行緒都有自己的執行個體
- 改善對記錄讀取時多執行緒並行存取的支援
- 改善透過 YML 組態在 SharedContext 中存放使用者名稱/使用者

- 使用更新的值改善鎖定記錄版本
- 新增對 NEXT SENTENCE 陳述式 OPM 編譯器特定行為的支援

橫向功能

新功能

- 新增中繼資料。ini 屬性`legacy.compilerto`指定要轉換之成品的舊版編譯器。某些 COBOL 陳述式的支援，例如 NEXT SENTENCE，取決於您設定的值。
  - z/OS 舊版系統的「ZOS」。
  - AS400 系統的「ILE」或「OPM」。預設 = "ILE" 當 `legacy.system` = "as400"

改進

- 前端 - 重新設計螢幕欄位元件，以擴展支援欄位類型的範圍。此增強功能可讓執行時間適應 AS400 中涉及的更多使用者輸入和資料需求。
- 改善 ZonedType 上個別簽署位元組`isValid()`的方法
- 已改善對涉及 CRLF 之串連`StringConcatenationBuilder::withPointer`的支援
- 改善對特定雙位元組編碼的支援，使其可安全執行緒
- 透過整合新的架構來改善狀態機器效能
- 改善指派最佳化的演算法，以防止意外重寫

AWS Blu Age 轉換引擎 4.4.0

zOS

改進

- LISTCAT - 改善剖析器以防止重複項目
- LISTCAT - 改善對 JCL / Groovy 中檔案系統的 ESDS 支援
- CICS - 改善對 CICS 陳述式 LENGTH OF 的支援

AS400

改進

- DDS 記錄產生增強功能
  - 改善 DDS 記錄的支援，以產生對應至 DDS 記錄結構的實體
  - 支援更符合舊版的共用欄位和映射函數
  - 改善外部描述和程式描述檔案的處理
- RPG - 改善僅自由形式模組的 RPG 偵測
- RPG - 已改善對 COPY 陳述式的支援，以忽略關鍵字*LIBL/作為字首來尋找應用程式複製本
- RPG - PF - 使用來自 pfile 的實體記錄改善對輸入規格的支援
- RPG - 新增對 On-Exit 陳述式的支援
- RPG - 改善對 LikeRec 關鍵字的支援
- RPG - 改善重新命名的 DSPF 欄位的映射
- CL - 改善欄位名稱解析
- COBOL - 改善支援從十六進位轉換為角色
- 改善對小數類型生成的支援
- 改善對不支援舊版程式碼的 FIXME 訊息支援 （顯示整個舊版行）
- 改善 AWS 轉換引擎的效能 (AS400 剖析步驟）
- 改善對關鍵字 LikeRec 的支援，使其與檔案規格保持一致
- 改善對內建函數 %Diff 的支援
- 新增對 DSPF 標籤上特殊字元貨幣符號的支援

版本備註 4.3.0

版本日期：2024 年 9 月 16 日

此版本的 AWS Blu Age Runtime 和 Modernization Tools 著重於擴展功能和涵蓋範圍，以現代化大型主機功能。此版本中的一些主要功能和變更包括：

- CICS：其他支援從終端機交換資料，並透過支援 SEND MAP 命令搭配映射參考來執行與傳入資料的交易。
- JCL：新功能，允許 從先前失敗的 JCL/PROC 步驟重新啟動最近執行的批次任務，或略過先前執行的步驟來觸發延遲重新啟動。這可讓您使用持久的步驟層級檢查點更好地控制批次處理。
- AS400：CPYF、OVRDBF、SBMJOB 和 OPNQRYF 等常用命令的其他程式庫支援、增強的效能和穩健性。

我們使用以下堆疊測試此版本的 AWS Blu Age Runtime。其他元件版本也可能相容。

元件	測試版本
Java	Java 17
Presentation layer	Node JS 18.18
	Npm 9.8
	Angular 17
Service layer	Spring Boot 3.2.5
	Spring Core 6.1.5
	Spring statemachine 4.0.0
Persistence layer	PostgreSQL engine 14
	Oracle 21c
Application server	Apache Tomcat 10.1.17

如需此版本中包含之變更的詳細資訊，請參閱下列各節。

執行期 4.3.0 版

zOS

新功能

- CICS - 在 SEND MAP 命令中新增對映射參考的支援
- CICS - 新增對 RECEIVE 命令的支援，並支援使用來自`JicsTransactionRunner`畫面的資料執行交易
- 新增對 JMS 訊息 IIH 標頭的支援
- COBOL - 在 Pseudo-text for REPLACING 陳述式中新增對多個內嵌空間的支援
- COBOL - 新增對 JSON PARSE 陳述式的支援
- Blusam - 新增對 KMS 的支援，以具有「匯出資料集」功能

- BAC - 新增 的組態`application-main.yaml`來定義記錄大小，以篩選與此記錄大小相符的已載入遮罩
- JCL - INFUTILB - 新增對關鍵字 INTO 的支援，作為 BMC 控制陳述式的一部分
- GS21 - 新增 JEF 編碼的 SOSI 處理
- GS21 - JCL - 新增 KDJBR14 做為 IEFBR14 的別名
- GS21 - JCL - 新增 KQCAMS 做為 IDCAMS 的別名
- MF - 新增對 COBOL MF 相容檔案的支援 dependingOn 欄位支援
- MF - 新增對 COBOL MF 相容檔案 SORT 機制的支援
- MF - 新增對 COBOL MF 相容開啟非選用遺失檔案的支援

改進

- JCL - DSNUTILB - 使用 ZONED DECIMAL 類型改善 LOAD 操作
- JCL - DSNUTILB - 新增對重複金鑰的支援
- JCL - DSNUTILB - 新增對 LOAD 命令上復原機制的支援
- JCL - INFUTILB - 使用新屬性 FETCHSIZE 和 CHUNKSIZE 改善 UNLOAD
- JCL - IKJEFT1A - 新增目前字元集來增強 SYSTSIN 檔案讀取
- JCL - DFSORT - 新增對選項 DATE4 和 DATE5 的支援
- JCL - DFSORT - 新增對變數區塊類型作為輸入和固定區塊類型作為輸出的案例的支援
- JCL - DFSORT - 新增對 ALTSEQ 的支援
- JCL - 使用任務 Web 識別符增強檢查點中繼資料
- JCL - 已改善 REDIS 的批次重新啟動檢查點清除
- IMS - 實作 PURGE 命令的 EXPRESS 函數
- IMS - 新增對選項的支援PCBNAME 和適用於PCB 陳述式的 LIST
- COBOL - 新增對沒有目標的 GO TO 陳述式的支援
- CICS - 改善對 READQ TS 中 RecordAdaptable 的 INTO 陳述式的支援
- CICS - 改善對 INQUIRE TRANSACTION 命令的支援
- CICS - 改善對 READNEXT 命令中 setBytes 的支援
- CICS - 改善不使用 CHANNEL 選項的 START 命令支援
- CICS - 新增支援查詢 TSQueue 的參考類型
- CICS - 改善地圖和地圖集為參考時的 RECEIVE MAP 命令支援

- CICS - 改善對選項 FROM 和 LENGTH for RECEIVE MAP 命令的支援
- CICS - 新增對 RecordAdaptable 屬性的支援
- CICS - 改善對 RECEIVE 命令處理溢位的支援
- CICS - 新增對 CICS 陳述式中配量規則的支援
- CICS - 改善對連結結構 DFHCOMMAREA 和 DFHEIBLK 的支援。轉換引擎支援更隱含的定義
- CICS - 新增對選項 START、NEXT 和 END for INQUIRE CONNECTION 命令的支援
- CICS - 新增對 RECEIVE 命令選項 LENGTH 類型 "int" 和 "reference" 的支援
- CICS - 增強對剖析 INQUIRE NETNAME 命令的支援
- CICS - 新增對 JicsQueueBuilder 群組名稱的支援
- Blusam - 新增支援以一般金鑰開頭的索引檔案
- Blusam - 改進的 Blusam 載入器
- BAC - 當 Redis 用於集中快取值時，改善對多執行個體環境中資料同步的支援，包括實際資料和鎖定
- BAC - 改善的 UI （樣式、標誌、核取方塊）
- BAC 和 JAC - 新增 `application-main.yaml` 的組態，透過指定 ARN，從 AWS Secrets Manager 擷取秘密中預設超級管理員使用者的使用者名稱和密碼
- BAC 和 JAC - 將相依性升級至引導 5
- 改善 JCL 檢查點和 JICS TSQueues Redis 範本組態
- 根據 AMode 改善對指標大小的支援
- 新增對 NumericEditedType 零比較的支援
- 記錄前強制執行的 Slf4j MDC 屬性
- 改善對檔案讀取的支援，以處理多個空白行
- MF - 已改善對初始化 COBOL MF 編譯器指令 initPtr 指標變數的支援
- Redis - 透過以 Redisson 為基礎的實作，改善在同意方面的 GwFileLock 功能

AS400

新功能

- CL - 新增對 CHGPF 命令的支援
- RPG - 新增對函數 %HOURS、%MINUTES 和 %SECONDS 的支援

- COBOL - 新增支援具有 Blu4IV DAO 架構的 SORT 檔案

改進

- CL - 改進 PgmClose 以註冊為程式並接受 OPNID 參數的各種物件
- CL - 重構 RTVMBRD 來處理多個程式庫和成員
- CL - 新增對 MOVOBJ 命令上 TOLIB 參數的支援
- CL - 改善對 CPYFRMSTMF 命令上分割區的支援
- CL - 新增對 SNDMSG 參數 TOUSR 的支援
- CL - 改善對 OVRDBF 命令的支援
- CL - 改善 OVRDBF 命令的效能 - 更新 srcfile 和成員的預設值
- CL - 使用 CPYF 命令改善檔案複製
- CL - 重新設計的 CPYF 命令更強大，更能處理 QTEMP、CRTFILE、 FROMRCD & TORCD、MBROPT 和 FMTOPT(MAP & DROP)
- CL - 改善對 FROMFILE 和 TOFILE 資料欄不相符之案例的 CPYF 命令支援
- CL - 改善了指定 REPLACE 時 CPYF NOCHK 處理不同名稱資料欄的方式
- CL - 在邏輯檔案上新增 CRTDUPOBJ 命令的空白實作
- CL - 使用 CHGDTAARA 命令處理子字串索引問題
- CL - 改善對 SBMJOB 命令的支援
- CL - 使 OverrideManager 和 OpnqryfHelper 映射不區分大小寫
- 畫面 - 改善未指定游標時第一個可編輯欄位的初始焦點
- 畫面 - 改善關閉後和使用說明選單時的焦點位置
- 畫面 - 在資料表元件中按下頁面向上/向下後改善游標焦點
- 畫面 - 改善對多個欄位錯誤訊息和焦點的支援
- 畫面 - 改善子檔案欄位的行號運算
- 畫面 - 改善對使用 SFLINZ 初始化的子檔案的支援
- 畫面 - 改善僅數值項目的支援
- 畫面 - 改善了使用 3 個參數處理 DSPF 中的 WINDOW 關鍵字
- 畫面 - 改善包含超過 1 行記錄之資料表頁尾的位置
- 畫面 - 改善頁面導覽，以將輪換訊息粘貼在向上/向下頁面
- 已改善編輯程式碼 3 的 EDITC 功能

- 改善 Blu4iv 資料區域鎖定機制，以便在沒有鎖定可解除鎖定時不執行任何動作，而不是擲回例外狀況
- 新增支援，以傳回 StraightQueryBuilder 中受影響的資料列數
- 改善 QTEMP 日誌機制
- 改善 DAOManager 讀取reads/writes/deletes檔案上的使用案例，以不同的檔案 + 程式庫覆寫

橫向功能

新功能

- 新增依組態管理 SSL/TLS 相關系統屬性的集中方式，允許使用 AWS Secrets Manager
- 使用 增強 IBMMQ 資源的組態 AWS Secrets Manager
- JCL - 透過 YML 屬性 tempFilesDirectory 新增執行期解析的 groovy 檔案的暫時位置組態，並新增 功能，以指定是否要在應用程式啟動時透過 YML 屬性 cleanTempFilesDirectoryAtStartup 清除暫存檔案資料夾的內容
- 為所有 Redis 登入資料新增 AWS 秘密

改進

- 改善從英數字元類型轉換為已編輯的數字類型
- 改善 PackedType 的 DataUtils：：isNumeric 檢查
- 增強型日誌檔案時間戳記
- 已處理單獨的登入 ZonedType.decodeAsString
- COBOL - 改善對陳述式 INITIALIZE 的支援
- 改善對 DataUtils.compareAlphInt 的支援，以處理 AS400 和 ZOS 的前尾空間
- SQL - 改進了隱含唯讀游標執行時間驗證
- SQL - 改善中繼資料快取機制
- 從 Gapwalk 應用程式移除 Jics/Blusam 資料庫連線 `application-main.yml`

現代化工具 4.3.0 版

zOS

新功能

- GS21 - 新增 COBOL GS21 CONSTANT 章節的支援
- GS21 - 將 JEF 編碼新增至可用的字元集

改進

- CICS - 新增剖析 DOCUMENT CREATE 命令的支援
- CICS - 新增對剖析 CICS WEB EXTRACT 命令的支援
- CICS - 新增了剖析 WEB WRITE 命令的支援
- CICS - 新增了對 DB2CONN SIGNIN 和 PLAN 的轉換支援
- CICS - 透過忽略選項 TERMINAL 增強對剖析 SEND MAP 命令的支援
- CICS - 透過忽略選項 ENDACTIVITY 增強對剖析 RETURN 命令的支援
- MFS - 改善對產生具有特定副檔名的MFS 檔案的支援
- COBOL - 改善對 REPLACE 陳述式的支援
- COBOL - 已處理的動態路徑和 MF 編譯器指令
- COBOL - 改善對 CALL 陳述式中 OMITTED 值的支援
- COBOL - 改善多維欄位存取，以支援簽章值
- COBOL - 新增對 FILE STATUS 陳述式的 OF 子句的支援
- COBOL - 改善 RESULT-SET-LOCATOR 陳述式的剖析
- JCL - IDCAMS - 新增對 RECORDS 縮寫的支援

AS400

新功能

- CL - 新增對 CL 轉換中指標型和定義變數的支援
- CL - 新增對 DCLF 中特殊字元的支援
- 新增擷取呼叫堆疊 (QWVRCSTK) API 的支援

改進

- RPG - 使用`likeds`關鍵字改善程序參數的轉換
- RPG - 檢閱關鍵字 EXTNAME 的支援

- RPG - 改善支援常值 *ALL
- RPG - 改善對輸出規格和程式描述檔案的支援
- DDS - 改善 LF 中參考參考字典 PF 之 PF 的 DDS 欄位解析度
- 畫面 - 使用 CLEAR 陳述式從 DSPF 清除記錄時清除的指標
- CL - 使用元素清單改善 CL 參數的轉換/產生

橫向功能

改進

- SQL - 已改善產生包含 N 與波狀符號字元的 SQL 查詢
- COBOL - 改善對群組欄位 LENGTH OF 陳述式的支援
- COBOL - 改善對使用複製手冊的 REDEFINED 欄位的支援

版本備註 4.2.0

版本日期：2024 年 7 月 10 日

此版本的 AWS Blu Age 執行期和現代化工具著重於效能和安全性。此版本中的一些主要功能和變更包括：

- 我們改善了轉型效能，尤其是具有超過 3，000 萬行程式碼的大型專案。我們實作了一組改進，我們取得的結果顯示時間減少超過 150%，並在幾分鐘內完成執行，而不是數小時。我們實作的關鍵改進是逾時機制的組態，以限制分配給分析的最長時間，以便略過偵測到問題的檔案。我們會標記略過的檔案，以便您稍後在必要時進行調查。
- 我們新增了對 AS400 專案分散式鎖定管理系統的支援。在高可用性環境中（多節點），其中應用程式的多個執行個體以相同的資料庫為目標，在整個執行個體生命週期中維持資料一致性是一項重大挑戰。為了有效地解決此挑戰，我們新增 Redis 做為共用和外部快取伺服器，以在批次模式下執行時協調所有執行個體。
- 我們新增了資料表元件的新動態分頁功能。此功能的目標是改善回應時間，並減少具有大量資料列之資料表的記憶體用量。此功能允許資料表元件僅載入部分資料，並在導覽頁面時隨需擷取更多記錄。為了進一步改善體驗，平台也支援預先擷取資料。這項新的動態分頁功能可為具有大型資料集的應用程式提供更有效率且回應靈敏的使用者體驗。
- 為了解決經常出現的關鍵挑戰，我們新增了對巢狀 COBOL 計劃的支援。先前，現代化巢狀 COBOL 程式的解決方法，涉及手動將程式分成不同的檔案、透過連結區段進行連結，並讓它們使用必要的引

數互相呼叫。此程序不僅耗時，而且容易出錯。您現在可以將巢狀 COBOL 程式現代化，而不需要手動分離。

我們使用以下堆疊測試此版本的 AWS Blu Age Runtime。其他元件版本也可能相容。

元件	測試版本
Java	Java 17
Presentation layer	Node JS 18.18
	Npm 9.8
	Angular 17
Service layer	Spring Boot 3.2.4
	Spring Core 6.1.5
	Spring statemachine 4.0.0
Persistence layer	PostgreSQL engine 14
	Oracle 21c
Application server	Apache Tomcat 10.1.17

如需此版本中包含之變更的詳細資訊，請參閱下列各節。

執行期 4.2.0 版

zOS

新功能

- DB2 - 新增對 SQL 查詢中沒有結構描述限定詞之預存程序調用的支援
- COBOL - 新增對 HEX-OF 函數的支援
- COBOL - 新增對巢狀程式的支援
- COBOL - 新增對 FUNCTION TEST-DATE-YYYYMMDD 和 TEST-DAY-YYYYDDD 的支援

- CICS - 在 SET TERMINAL 命令中新增對選項 UCTRANST 的支援
- CICS - 新增對 INQUIRE DB2CONN 命令的支援
- BluSam - 新增對動態存取 VSAM 上金鑰刪除的支援
- IMS - 新增對 TERM 命令的支援
- BAC - 新增所有 BAC REST 端點的授權檢查
- BAC - 透過 新增組態`application-main.yaml`以定義記錄大小，以篩選符合該記錄大小的已載入遮罩
- BAC 和 JAC：透過 新增組態`application-main.yaml`，`command`透過指定 ARN 從秘密中擷取預設超級管理員使用者的使用者名稱和密碼

改進

- JCL - SORT - 增強對 OMIT 子句的支援，以使用 Shiftin 和 ShiftOut 字元處理條件
- JCL - SORT - 改善對 BDW 欄位的支援
- JCL - SORT - 使用 BDW 欄位改善對多個 GDG 串連的支援
- JCL - DFSORT - 新增對 INREC PARSE STARTAFT / STARTAT 子句的支援
- JCL - IEBGENER - 增強型 recordSize 處理輸出檔案
- JCL - INFUTILB - 以 YML 為基礎的已停用 NULL 指標 - 修正圖形大小寫
- JCL - 改善對 FormatterParser 處理 OUTREC 欄位中常數的支援
- JCL - DSNUTILB 程式公用程式中圖形類型的增強型載入資料
- JCL - SORT - 增強對 Zoned Decimal 格式的支援
- JCL - SORT - 增強對 OMIT 子句的支援，以使用 Shiftin 和 ShiftOut 字元處理條件
- MQ - 改善 MQ 連線的處理，以符合數個業務工作流程
- CICS - 增強支援 EXEC CICS READ SET (ptr-ref) 陳述式的指標參考
- COBOL - 改善對 ADDRESS OF 連結區段記錄的支援
- COBOL - 新增對 EXP 和 EXP10 函數的支援
- COBOL - 改善對使用複製手冊的 REPLACE 陳述式的支援
- COBOL - 改善多維欄位存取以支援簽章值
- MF COBOL - 新增對變數格式序列檔案的支援
- IMS - 改善 IMS YML 檔案組態的讀取，以便能夠使用環境變數

- IMS - 已處理指定區段號碼的其他方式
- IMS - 從程式設計開始的交易呼叫 IMS 程式時新增了穩健性
- IMS - 改善搜尋條件 SSA 組建，以便在未提供隱含區段長度時，將 WHERE 子句的目前長度納入考量
- IMS - 改善 IMS YML 檔案組態的讀取，以允許使用環境變數
- 改善對 NumericEditedType 中 VALUE 子句的支援
- 改進了對字串串連的支援，以在要串連的第一個字串為空白、空白或空格時處理案例

AS400

新功能

- 新增對資料表元件內分頁的支援；當載入具有大量資料列的資料表元件時，專案可以使用此功能來減少回應時間和大小
- 新增了對 AS400 應用程式上 SQL 查詢的程式庫支援；由於程式庫在現代應用程式中會轉換為分割區，因此我們調整了執行時間以相應地重寫查詢
- RPG - 新增對 SQL 查詢 QTEMP 程式庫的支援
- RPG - 在 CONVERT 函數中新增編碼，以處理空的輸入值
- RPG - 新增對 %HOURS、%MINUTES 和 %SECONDS 函數的支援
- CL - 新增了 CHGPFM 命令
- CL - 在 CRTDUPOBJ 命令中新增對 *FROMLIB 關鍵字的支援
- CL - 針對超過 9 個字元的資料表名稱新增對資料表和分割區建立的支援
- CL - 新增支援刪除 DLTF 命令子資料夾中的平面檔案

改進

- 畫面 - 改善要與特定欄位繫結並新增至 ArrayMessageLine 的 ErrorMessage ArrayMessageLine
- 畫面 - 改善錯誤訊息游標
- 畫面 - 改善 ArrayMessageLine 但不包含在 Tab 順序中
- 畫面 - 改善 AS400 畫面的錯誤訊息陣列顯示
- SQL - 改善游標在關閉時遞交交易的支援，以避免分割區建立時的死結
- CL - 新增對 PgmCall 命令的支援，並改善 QCMDEXC 不支援的模式

- CL - 改善對 CHKOBJ 命令處理 OBJTYPE PGM 的支援
- CL - 改善了對 CPYF 和其他處理程式庫和分割區的 CL 命令的多程式庫支援
- CL - 新增支援在 CALL PGM 命令中傳遞程式名稱變數
- CL - 處理物件類型預設類型的案例
- CL - 新增對 CRTDUPOBJ 命令的多程式庫支援
- CL - 多個命令上的增強型資料庫連線處理
- CL - 改善對 RMVLNK 在找不到檔案或目錄和 CPF0000 監控訊息時處理案例的支援
- CL - 改善 CLRPFM，以在移除記錄時將程式庫納入考量
- CL - CPYF - 改善支援 QTEMP 程式庫、FmtOpt(*NoChk) 參數和控制字元的命令
- CL - 修正 RMVLNK 和 CPY 命令中引號和遺失參數的處理
- RPG - 增強型變數範圍；DataArea 現在處於工作範圍，而不是連結範圍
- RPG - 改善了在沒有交易的情況下執行的 DAO 讀取查詢，以避免死結
- 透過在資料庫查詢中將修剪新增至 MSGQ 來增強 MQ 訊息查詢
- 已移除資料庫連線支援上不必要的交易宣告
- 在發生例外狀況時，改善了更新 Jetqua 任務狀態
- 新增支援，以在指標陣列未初始化時處理案例

橫向功能

新功能

- Redis - 新增所有 Redis 快取的全域 Redis 組態
- 新增工作階段追蹤功能，透過將資料保留到 Redis 來儲存工作階段追蹤資訊 （工作階段 ID、相關聯的使用者名稱、建立時間戳記和節點 ID)
- 透過 YML 屬性 新增執行時間解析的 groovy 檔案的暫時位置組態`tempFilesDirectory`；也新增了 功能，以指定是否要在應用程式啟動時透過 YML 屬性清除暫存檔案資料夾的內容 `cleanTempFilesDirectoryAtStartup`

改進

- 增強對公用程式資料來源連線集區實作組態屬性的支援
- 根據 ADVANCING 子句和 WRITE BEFORE 子句的使用，改善對印表機模式和 ANSI 換行控制的支援

- 更新了適用於現代化專案之前端應用程式的角度版本
- 資料庫 DB2 的增強型秘密管理員 URL 語法建構
- 增強 DataUtils.compareAlphInt 方法，新增對結尾空格的支援
- 改善對 Blob 類型輸出的 SQL 支援
- 透過 Post/script 端點為任務觸發新增了穩健性

現代化工具 4.2.0 版

zOS

新功能

- CICS - 新增了剖析 WEB CICS 命令的支援
- CICS - 新增對 MONITOR 命令轉換的支援
- CICS - 新增了剖析 CICS 命令 SEND MRO 的支援
- COBOL - 新增了剖析 NO REWIND 陳述式的支援
- COBOL - 新增對 CICS 命令 SET TERMINAL 中 UCTRANST 選項數量類型的支援
- COBOL - 在 I-O-SECTION 中新增支援 MULTIPLE FILE 子句
- CSD - 新增對轉換多個 CSD 檔案的支援
- CSD - 新增支援從多個 CSD 檔案產生 jicsFileAix.json
- IDCAMS - 新增支援建立相對記錄資料集 (RRDS)

改進

- 改善運算 SQL 遮罩時的效能
- COBOL - 改善 FILE-CONTROL 中無用 RESERVE 子句的剖析
- COBOL - 改善章節和類別的剖析
- COBOL - 改善 DFHRESP 處理
- COBOL - 透過執行增強對 EXIT PARAGRAPH 的支援
- IMS - 改善對使用雙括號指定區段名稱的支援
- IMS - 在叫用 SCHD 和 TERM 時充實狀態碼的產生
- COBOL - 改善 DEPENDING ON 欄位的產生

- COBOL - 改善 TO_TIMESTAMP DB2 內建函數的轉換

AS400

新功能

- 新增在 SQL 指令碼中將英數字元欄位轉換為 CHAR 的支援
- COBOL400 - 新增對程式描述的 DATABASE 檔案的支援

改進

- DDS - 改善對 ALIAS 名稱的支援
- 增強對沒有初始值的類型浮點數的支援
- COBOL 400 - 改善已簽署區域類型的大小運算

橫向功能

改進

- 改善有關 DDS 和 SQL 剖析的錯誤 ID 報告
- 改善條件分支的程式碼產生
- 改善產生天氣報告的效能

版本備註 4.1.0

版本日期：2024 年 5 月 31 日

此版本的 AWS Blu Age 執行期和現代化工具著重於效能和安全性。此版本中的一些主要功能和變更包括：

- 轉換和效能：為了允許具有大型程式碼庫 (+5000 萬行程式碼） 的專案成功轉換，我們最佳化了整個轉換機制的效能和記憶體使用量。
- BAC/JAC： 的安全 AWS 是最高優先順序。使用 AWS Blu Age 進行現代化的應用程式必須符合安全標準。我們已對 BluSam 管理主控台 (BAC) 和 JICS 管理主控台 (JAC) 進行一些重大升級，以使其更加安全：
  - 將應用程式更新為 Angular v17。

  - 除了原生支援 AWS Cognito 之外，我們也新增了 OAuth 的一般支援，可讓客戶更靈活地使用自己選擇的身分提供者。
  - 使用適當的標頭設定和擴充安全功能。
- AS400 - 資料庫鎖定機制的多節點支援。提供插入共用和外部快取伺服器 (Redis) 以在多個執行個體上執行批次應用程式的可能性，例如受管 AWS 大型主機現代化。

此版本的 Blu Age 執行期已使用下列堆疊進行測試。其他版本也可能相容。

元件	測試版本
Java	Java 17
Presentation layer	Node JS 18.18
	Npm 9.8
	Angular 16.1
Service layer	Spring Boot 3.2.5
	Spring Core 6.1.5
	Spring statemachine 4.0.0
Persistence layer	PostgreSQL engine 14
	Oracle 21c
Application server	Apache Tomcat 10.1.17

如需此版本中包含之變更的詳細資訊，請參閱下列各節。

執行期 4.1.0 版

zOS

新功能

- 新增動態 OAuth2 供應商處理的組態。引進 SECRET_OAUTH2_PROVIDER_NAME_KEY 來指定提供者。已更新秘密擷取方法以處理多個供應商。確保從 安全地擷取秘密 AWS Secrets Manager。

- 新增對 上 DB2 SSL 屬性的支援， AWS Secrets Manager 讓您能夠定義 SSL 憑證 (sslTrustStoreLocation) 和密碼 (sslTrustStorePassword) 來解鎖金鑰存放區檔案。
- 新增對外部業務資料來源的支援。
- JCL - 新增對批次重新啟動檢查點機制的支援。
- JCL - 新增對 DCB 參數記錄大小和 RDW 的支援。
- JCL - 為產生的暫存檔案新增動態資料夾名稱組態。
- REDIS - 在 JICS 的 Redis 組態中新增集區組態。
- REDIS - 在目錄和 JICS 的 Redis 組態中新增資料庫索引。
- BatchScript - 新增了執行程式執行的步驟名稱傳播。
- CICS - 新增對 ADDRESS SET 命令的支援。
- CICS - 新增對 PURGE MESSAGE 和 JUSTIFY 的支援。

改進

- JCL - INFUTILB - 增強支援根據 YML 屬性停用 null 指標。
- JCL - INFUTILB - 改善對 CHAR/BPCHAR 資料類型的支援。
- JCL - ICEGENER - 新增將多行輸入串流複製到 檔案的支援。
- JCL - IEBGENER - 改善處理從變數區塊轉換為固定區塊檔案的支援。
- JCL - DFSORT - 改善對操作 DATE 上多位數參數的支援。
- JCL - DFSORT - 新增對 INCLUDE=ALL 子句的支援。
- JCL - 改善 SORT 公用程式處理輸出中 BDW 欄位的支援。
- JCL - 改善對 DD 串連的支援。
- JCL - 改善對輸入串流的支援。
- JCL - DSNUTILB - 改善對 NULLIF() 陳述式的支援。
- JCL - INFUTILB - 新增支援使用 NOPAD 選項卸載資料。
- JCL - INFUTILB - 增強對 INFUTILB 中目前日期的支援。
- JCL - 在使用檔案之前新增檔案存在和大小檢查。
- JCL - GDG - 改善處理 GDG 的子目錄。
- MQ - 改善 JMS 實作中的連線開啟。
- MQ - 改善了 XA datasoure GET 訊息的資料長度設定。

- MQ - 分解 CMQV 標準複製手冊，以防止編譯錯誤和重構用途。
- BluSam - 改善對不存在資料集的刪除請求的支援。
- 改善對 ALLOCATE 陳述式的支援。
- 改善了 TS-QUEUE 命名的穩定性。
- BatchScript - 在任務重新執行中增強先前步驟傳回程式碼的保留。
- 資料集 - 改善檔案存在且為暫時性時的檔案存在檢查。
- 資料集 - 改善尋找要刪除的 GDG 檔案時的並行。
- 資料集 - 新增取得 GDG 資料集記錄大小的支援。
- CICS - 改善 INQUIRE TASK LIST 命令中 SUSPENDED 選項的支援。
- CICS - 使用 ADDRESS OF 陳述式改善對 LOAD SET 的支援。
- CICS - 改善 CICS INQUIRE 時未處理的 CICS 引數 REMOTESYSTEM。
- CICS - 增強對 GETMAIN 命令的支援，以使用 OF 關鍵字定義的指標來處理 SET 選項。
- JICS - 透過新增交易狀態檢查來改善 jicsXAPrepare() 方法的穩健性。
- JICS XA - 新增了交易狀態和增強型交易執行緒終止的檢查。
- BAC - 用戶端的增強型角色型身分驗證，以及重構/集中所有 API 呼叫。
- BAC - 實作一項功能，根據組態封鎖對 BAC 和 JAC 的公開存取
- BAC - 依存項目的升級：角度 17。
- BAC - 改善與 OAuth2 的安全整合。
- BAC - 增強型休眠產生的 DDL。
- BAC - 改善了匯出資料集機制。
- JAC - 更新為 Angular 17，並報告 BAC 的所有詳細資訊 (ROLE、sadmin conf、XSRF、logut)。
- COBOL - 新增對 CHAR 和 ORD-MIN 函數的支援。
- 增強的 FileFactory 可將目錄記錄大小保留在 MOD 處置中。
- 針對 JICS 交易使用 MDC 啟用記錄。
- 改善針對產生臨機操作結果集的預存程序產生的 SQLCA > SQLSTATE。
- 已改善與上次 Spring 升級相關的任務排程支援。

AS400

新功能

- 新增使用 Redis 鎖定資料庫記錄的多節點支援。
- 新增對 DDS 類型的 BINARY CHARACTER 支援。
- CL - 新增對產生自訂報告檔案的支援。
- RPG - 新增對主要/次要檔案上 RENAME 關鍵字的支援。

改進

- 改善使用 JOIN 子句處理 CTID 資料欄的資料庫支援。
- 已改善多個 DSPATR(PC) 的游標位置。
- 改善讀取例外狀況的記錄。
- 改善了 Jetson 任務記錄，將任務屬性包含到 MDC。
- 已改善 AS400 說明畫面的支援。
- CL - 改善對 RMVJOBSCDE 命令的支援，以接受結尾空格的項目號碼。
- CL - 改善對 RMVJOBSCDE 命令的支援，以使用一般任務名稱移除任務排程。
- CL - 改善對 SAVOBJ 命令的支援，以按資料表索引鍵排序記錄。
- CL - 改善對 CPYF 命令的支援，以為資料庫查詢建立新的連線。
- CL - 已改善使用 SNDPGMMSG 在佇列訊息中插入查詢訊息。
- CL - 改善任務佇列組態以指定預設任務佇列。
- CL - 改善 CRTPF 命令以支援 QTEMP 程式庫和 RCDLEN 參數。
- CL - 改善對 CHKOBJ 命令的支援 - 檢查具有程式庫的分割區。
- CL - 改善 RTVMGS，以便在找不到檔案/ID 時傳送 CPF2407 和 CPF2419。
- CL - 改善傳統格式化參數的 CPYTOIMPF 和 CPYFRMIMPF 解釋。
- CL - 新增對 OVRPRTF 參數 USRDTA 的支援。
- CL - 改善 CPYTOIMPF CL 命令以建立新的連線，以避免關閉現有的結果集。
- CL - 改善了 CHGDTAARA，使其在更新內容時不會再修改資料區域長度。
- CL - 改善了 ClCommand 資料庫連線處理。
- 前端和後端之間的最佳化互動。
- COBOL - 更新轉換以處理複製手冊中的 FILLER。
- 已改善傳送至前端之自訂訊息的其他訊息資訊顯示。
- 更新 app.component.ts 中選取器的預設值。

- 改善split-dynamic-field顯示中的文字分割。
- 已改善錯誤訊息的顯示，其中包含多個寫入，後面接著一個讀取。

橫向功能

新功能

新增對 OAuth2 供應商秘密動態組態的支援。

改進

- 列印 - 改善處理引號的 QCMDEXC 參數支援，並改善報告名稱的形成
- 改善對 RecordAdaptable 上分隔語法的支援。
- 增強型 InspectBuilder 錯誤記錄，以新增來源字串的相關內容。
- DataSimplifier - 新增了 ByteArray 影響的穩健性。
- 使用新的執行期屬性增強 MDC 記錄。

現代化工具 4.1.0 版

zOS

新功能

- 新增對多個 CSD 檔案轉換的支援
- COBOL - 新增對 CICS ALLOCATE 陳述式的支援。
- COBOL - 在 ADD CORRESPONDING 陳述式中新增對 ON SIZE ERROR 的支援。
- COBOL - 新增對 EXIT PARAGRAPH 的支援。

改進

- COBOL - 改善對 -INC 複製手冊的支援。
- COBOL - 增強對 FILLER 初始化的支援。
- COBOL - 改善對圖元值比較的支援。
- COBOL - 在缺少中介程式碼區塊的連續 WHEN 子句中增強對 WHEN ANY 的支援。
- COBOL - 改善對圖形常數的支援。

- COBOL - 改善對封裝類型大小運算的支援。
- COBOL - 改善 SPOOLCLOSE 的未處理 CICS 引數 KEEP。
- COBOL - 改善了 TEST-NUMVAL 函數的產生。
- COBOL - 改善 INSPECT 架構支援的 Java 產生引數。
- CICS - 已改善定義 DFHCOMMAREA 的支援。

AS400

新功能

- RPG - 新增了錯誤擷取機制來產生（不完整）DDS，因此不會封鎖程式產生。
- 新增對 INCLUDE 檔案描述規格關鍵字的支援。

改進

- RPG - 改善了完全免費的剖析。
- RPG - 新增了強健性與錯誤捕捉。
- RPG - 使用匯出關鍵字改善欄位/DS 的初始化。
- RPG - 改善處理指標的 DAO 操作。
- RPG - 使用 CTDATA 處理 PERRCD 的預設值。
- RPG - 升級 Free-RPG 剖析器，以記錄每個剖析規則的唯一錯誤。
- PRTF - PRTF 和 JRXML 之間的已處理名稱衝突。
- COBOL - 改善對 LIKE 關鍵字的支援。

橫向功能

改進

- 新增 ErrorID API 的穩健性
- 大型專案轉換的效能最佳化。例如：略過封鎖檔案的逾時、重複使用 Blu Insights 中的分類，以及更好的記憶體配置。
- 在 COBOL/PL1 轉換期間最佳化記憶體使用量。
- 已修正第三方 (jQuery 和引導）上的 CVE。
- TC 中的受管 timeoutParser 選項。

- 改善 SQL 查詢上的多個空格重寫。
- 改善了具有敏感度屬性的唯讀游標。

版本備註 4.0.0

版本日期：2024 年 4 月 8 日

如需如何從 AWS Blu Age Runtime 3.10.0 遷移至 4.0.0 的說明，請參閱 the section called “從 3.10.0 遷移至 4.0.0”。

此版本的 AWS Blu Age Runtime 和 Modernization Tools 著重於升級關鍵相依性和支援的技術，同時提升多種功能的效能。此版本中的一些主要功能和變更包括：

- • 從 Spring Boot 2.7 升級至 3.2.4、Spring Core 5.3 升級至 6.1.5，以及 Tomcat 9.0 升級至 10.1.17，透過使用主動修補和維護的版本來改善安全性、效能和可維護性。
  - 在前端應用程式上延遲載入，以使用超過 2000 個螢幕建置更快速的大型專案，並將顯示初始化從 10 秒縮短為 300 毫秒。
  - 支援前端應用程式的 DBCS 顯示，以增強對雙位元組字元的支援，以提供處理雙位元組和單位元組字元的新字型，防止雙位元組欄位中的單位元組輸入，以及處理混合雙位元組和單位元組字元的欄位。
  - AS400 Online 應用程式以平行方式執行 AS400 應用程式的執行緒監控功能。 AS400
  - 透過新增可設定的機制來預先初始化程式內容，從而改善內容和 RunUnit 初始化的效能，減少載入傳統複雜性中固有複雜結構的影響。

此版本的 AWS Blu Age Runtime 已使用下列堆疊進行測試。其他版本也可能相容。

元件	測試版本
Java	Java 17
呈現層	節點 JS 18.18
	Npm 9.8
	角度 16.1
服務層	彈簧開機 3.2.4

	Spring Core 6.1.5
	彈簧狀態機器 4.0.0
持久性層	PostgreSQL 引擎 14
	Oracle 21
應用程式伺服器	Apache Tomcat 10.1.17

如需此版本中包含之變更的詳細資訊，請參閱下列各節。

執行期 4.0.0 版

zOS

新功能

- 新增對包含陳述式 '-INC CPYNAME' 的支援。
- CICS - 新增對 PUSH/POP HANDLE 陳述式的支援。
- COBOL - 新增對 "ASSIGN TO DYNAMIC" 的支援。
- 新增使用 INFUTILB 對 DB2 UNLOAD 的支援。
- 在 INREC 陳述式的 OVERLAY 中新增對關鍵字 SEQNUM 的支援。

改進

- SORT - 新增對排序字串常值 C'....' 中特殊字元 （括號和星號） 的支援。
- SORT - 改善對 OUTFIL NOMATCH-(..) 引數的支援。
- SORT - 新增對 SYMNAMES 資料定義的支援。
- SORT - 改善 TO= 和 LENGTH= 引數的處理。
- SORT - 改善處理 MOD 處置。
- SORT - 新增對 HIT=NEXT 引數的支援。
- 增強的 ICEGENER 可新增對特定輸出檔案編碼的支援。
- INFUTILB - 增強對 WITH UR 子句的支援。
- INFUTILB - 增強支援在 writeNullIndicator 為 false 時卸載。

- DSNUTILB - 當 NULLIF 關鍵字在選用 SQL 關鍵字之後時，增強載入步驟的穩定性。
- DSNUTILB - 增強對隔離資料欄名稱的支援。
- DSNUTILB - 新增將空檔案載入資料表的支援。
- DNSUTILB - 新增對 DNSUTILB SYSDISC 檔案 MOD 處置的支援。
- IDCAMS - 增強型評論支援。
- JCL - 新增對 LoadTask 中雙引號資料欄的支援。
- JCL - 有關白速移除的增強型 UNLOAD SQL 查詢處理。
- JCL - 在處理時發生例外狀況時增強對 Groovy 指令碼的回應，以確保 JSON 格式。
- JCL - 在 DISP=NEW 和 DISP=OLD 的情況下改善了檢查檔案處置。
- JCL - 增強支援在 GDG 基本名稱中使用特殊字元處理多個 GDG 產生參考。
- JCL - 增強支援載入虛擬檔案。
- JCL - 增強對 tempFilesDirectory YML 參數的支援。
- JCL - 當需要逸出字串元素 中的雙引號時，改善了 JSON 傳回。
- JCL - 增強的 FileUtils 支援 GDG 基本名稱。
- JCL - DB2 多個查詢執行的增強型 DSNTEP 程式。
- 新增對 Spring Bean 的支援。
- 增強 SQLConverter 以避免更正錯誤的日期。
- 改善 YYYYDDD 的 JicsTimeBuilder 處理。
- 允許從 groovy 存取自訂 jar。
- IMS - 增強 IMS 資料庫實作中跨記錄的導覽。
- IMS - 增強 CBLTDLI 以能夠啟動程式使用清除。
- IMS - DFSRRC00 能夠將參數從 groovy 傳遞至後端程式。
- 新增對未透過 transactionRunner 叫用之 JICS 命令的支援。
- JICS - 使用可設定的快取來改善效能。
- BluSam - 新增在開啟時停用暖機 BluSam 的支援，以增強大型資料集的效能。
- BluSam - 改善一般 BluSam 資料集上的刪除/重新命名行為。
- BluSam - 增強記錄操作的效能。
- 用於判斷字串是否為低值之方法的增強型資料簡化器。
- 增強對 Packed-Decimal 和排序順序問題的支援。

- 使用 AWS Secrets 增強 DB2 做為主要資料來源的組態。
- 增強型 FileSystem API 以公開檔案狀態。
- 使用 lineSeparator 增強型 DynamicFileBuilder 讀取串流輸入。
- 當 處理 CUSTOM930 字元集時，用於判斷字串是否為低值的方法的增強型資料簡化器。
- SQL - 改善 SQL 預存程序輸出處理。
- SQL - 改善了具有別名之多個資料表的 lambda 映射。
- COBOL - 改進了對 LENGTH OF 陳述式的支援。
- COBOL - 新增對 TRANSFORM 陳述式的支援。
- COBOL - 新增對 9 個新數學函數的支援。
- COBOL - 改善對 INTEGER-OF-DAY FUNCTION 的支援。
- COBOL - 已改善對涉及圖形值的 88 層級的支援。
- COBOL - 已改善 SET ADDRESS 陳述式的轉換。

AS400

新功能

- 已移除重複的指標實體。
- 新增對 DBCS 字元的支援。
- 引進處理子檔案記錄控制的 HELP 關鍵字。
- 新增組態參數，以切換管道字元上的資料欄名稱大寫和分割註解資料欄內容。
- 新增支援使用 0x0c 做為已封裝類型欄位的最後一個小數。
- RPG - 使用 ExtProc('system') 宣告的已處理原型。
- CL - cl-manmand RMVMSG 的已處理 'CLEAR' 參數 + 引入記憶體內非程式訊息佇列。
- CL - 正在傳遞給 SBMJOB CMD() 呼叫的已處理一般陳述式。
- CL - 新增命令 STRCMTCTL 和 ENDCMTCTL。修改鎖定機制並清除交易和鎖定。
- CL - 新增對 CPYTOIMPF 命令 RCDDLM 參數的支援。
- CL - 在 SAVOBJ 命令中新增填補零的處理。
- CL - 新增處理包含在 RTVOBJD 的 OBJ 參數合格名稱中的程式庫。
- CL - 新增對 CPYTOIMPF 命令參數 STRDLM、STRESCCHR 和 RMVBLANK 的支援。

- CL - 當找不到檔案/id 時，增強的 RTVMGS 可傳送 CPF2407 和 CPF2419。
- CL - 改進了 RCVF 命令，以從 DEV 參數中任何提供的程式庫接收記錄。

改進

- 變更 Blu4iv 任務執行器的預設值，以允許預設更好的擴展。
- 參數協助程式已修改，將字串清單和 ElementaryRangeReference 轉換為字串。
- 增強 CTID 來處理 POSTGRE 中不存在的資料欄。
- 新增強健性以支援使用者空間 API "QUSPTRUS"。
- 新增對使用者空間 APIs QUSRUSAT 和 QUSCUSAT 的支援。
- 增強支援無錯誤碼的使用者空間 API (QUSPTRUS)。
- 已新增對使用 Jetson 的 CRON 任務排程的支援。
- 增強對 RPG 程式週期的支援。
- 改善 Blu4iv 交易管理。
- 已改善相同交易中承諾控制下檔案的記錄鎖定。
- 改善子檔案初始化的處理。
- 改善訊息行捲動指標的顯示。
- 防止透過資料佇列傳送的號碼尾隨零。
- 改善了其他訊息資訊畫面。
- 改善 JPA 寫入操作，以考慮目前的程式庫。
- 改善不使用參數執行程式時 ProgramJobExecutor 的行為。
- 新增直接從前端連結將引數傳遞至後端指令碼的功能。
- 改善任務中繼資料的交易處理。
- CL - 新增對 RTVMSG 中參數 SECLVL 的支援。
- CL - 新增 CLRLIB 的空白實作。
- CL - 改善了從資料庫和 CSV 複製的 CPYFRMIMPF 支援。
- CL - 改善 CPYFRMIMPF 實作以忽略額外的資料欄。
- CL - 改善傳統格式參數的 CPYTOIMPF 和 CPYFRMIMPF 解譯。
- CL - 新增參數 removeDecimalPoint 以格式化 SAVOBJ 中的數值。
- CL - 改善 RCVF 命令以正確處理 EOF 條件。

- CL - RTVSYSVAL - 實作 SYSVAL = QDATETIME。
- CL - OVRDBF 命令已修改，以取得欄位做為預設資料表名稱。
- CL - RTVJOBA 無法使用的參數值：USRLIBL。
- CL - 處理 SNDPGMMSG MSGF 參數中的正斜線。
- CL - 已改善命令 DSPFFD 中 sourcefile 中萬用字元的支援。
- CL - 改善處理 RCVMSG 和 SNDPGMMSG 中的參數 PGMQ。
- CL - 將 RTVMSG 參數 MSG 設為選用，以與舊版文件保持一致。

橫向功能

新功能

- 改善在 OPEN 游標的 USING 子句傳遞參數時的功能。
- 效能：改善內容的預先初始化和 RunUnit 的效能調校。

改進

- 改善從 INFUTILB 公用程式的 UNLOAD 命令傾印低值的機制。
- 新增支援資料來源秘密管理員的目前結構描述選項。
- 增強執行時間，以便在不需要參數時，不考慮在開啟游標處傳遞的參數。
- 已改善數值欄位的數值格式驗證。
- 改善高度平行執行環境中的 SQL Diagnostic。
- 引進 unicode for codepage 位元組序列 (FE FD)。
- DataSimplifier 效能最佳化 - 增強型指派陳述式。
- DataSimplifier 效能最佳化 - 改善數值類型初始化的預設值，以防止無用的 BigDecimal 用量。

現代化工具 4.0.0 版

zOS

新功能

- 新增處理 Abend PROGRAM 的支援。
- 已改善產生 AIX 資料集的支援。

- COBOL - 新增對 ALPHANUMERIC/ALPHABETIC/GRAPHIC 欄位的 JUSTIFIED 子句的支援。

改進

- 已改善 TRANSCLASS 資源定義的 PURGETHRESH 屬性處理。
- 已改善對資料定義和 MOVE 陳述式的支援。
- CICS - 在選項 MILLISECS 上增強對 DELAY 命令的支援。
- 改善了具有別名之多個資料表的 SQL lambda 映射。
- 改善對父欄位調查結果的支援。
- 已改善 COMMIT 和 ROLLBACK 操作的 SQLCA sqlstate 集。
- COBOL - 透過評論過時的段落來增強剖析
- COBOL - 增強對 REPLACING 子句的支援。
- COBOL - 新增對數學函數 ASIN ACOS LOG TAN 的支援。
- COBOL - 在 PERFORM VARYING 中新增對多個 AFTER 陳述式的支援。
- COBOL - 增強對 RENAMES （層級 66) 欄位的支援。
- COBOL - 增強 LENGTH OF 方法，可在陣列欄位中取得特定索引的長度。
- COBOL - 在 PERFORM VARYING 陳述式中新增對多個 AFTER 子句的支援。
- COBOL - 增強對 RENAMES 子句的支援。
- COBOL - 增強對 PICTURE 關鍵字的支援。
- COBOL - 增強對 Level 88 欄位剖析的支援。
- COBOL - 根據具有資料表資料項目的條件，改善前往。

AS400

新功能

- 新增將引數傳遞至前端 Java 呼叫的功能。
- CL - 改善了 %SST 產生，包括支援使用 CL→Java 的 *LDA。
- RPG - 新增支援 DISK 檔案的程式描述記錄。

改進

- 改善顯示檔案，使用「REFFLD」關鍵字解析參考欄位。

- 改善對顯示檔案關鍵字 SETOF-CSRLOC 的支援。
- 關閉後從承諾控制移除檔案。
- 確保在相同程式執行時，資料表上並行讀取和寫入操作的一致行為。
- 已處理 SizePrefixedAlphanumericType 子字串的指派。
- 使用不同長度的字串參數處理傳遞資料結構至程序。
- 已改善在 onBlur 事件時保留無效的數值，並僅為有效欄位建立事件接聽程式。
- 改善螢幕上的錯誤訊息，並反白輸入無效的欄位。
- 改善以指標為條件的畫面欄位處理。
- 使用滑鼠滾輪啟用捲動。
- 新增支援說明畫面的 函數索引鍵。
- 已改善對split-dynamic-field元件中長文字的支援。
- 改善重新命名記錄時多記錄 LF 檔案的處理。
- CL - 改善 RTVJOBD 命令來處理 LF 檔案（檢視）。
- CL - 改善在多筆記錄 LF 上使用時的 OVRDBF 命令。
- RPG - 程序定義與重新命名參數同名變數的已處理案例。
- RPG - 改善初始化簽署的 binaryInteger 時 *ZEROS 的處理。
- RPG - 改善非本機（參考）變數指標的處理。
- RPG - 改善 IFxx 陳述式後的 ELSEIF 陳述式處理。
- RPG - 新增對 LIKE on 原型所定義欄位的支援。
- RPG - 改善對 LIKEREC 建立之欄位的 LIKE 關鍵字的支援。
- RPG - 改善了使用圖形運算子的產生。
- RPG - 改善陣列表達式 xxx(*) 的剖析，並在 %lookup 中支援。
- RPG - 改善了具有高和等於（或低和等於）指標的 LookUp 操作程式碼。
- RPG - 改善自由格式剖析。
- RPG - 改善遵循 I-card 記錄格式的 I-card 命名常數剖析。
- RPG - 已改善對 INTEGER 和 UNSIGNED 類型的支援。
- COBOL - 新增支援 COPY DDS 陳述式中 DSPF 格式的 INDIC 子句。
- COBOL - 改善 DISPLAY 和 ACCEPT 陳述式的語法，以解除封鎖轉換和產生。
- COBOL - 新增支援 fro DISK 檔案。

- COBOL - 改善 DDS 顯示檔案支援程式。
- COBOL - 新增對 LIKE 子句的支援。
- COBOL - 新增對程式描述 DISK 檔案的支援。
- COBOL - 新增對具有尾碼的檔案名稱的支援。

橫向功能

新功能

- 處理 Web 專案地圖元件的延遲載入。

改進

- 改善 SQL 指標參數的 Java 產生。
- 改善處理 SET DB2 陳述式中涉及變數的容量。
- 改善在輸出為單一實體陣列時，在擷取游標結束時引發錯誤的情況。
- Linux 中的受管路徑。
- Data Migrator 會管理漏洞並移除未使用的相依性。

版本備註 3.10.0

此版本的 AWS Blu Age 執行期和現代化工具專注於跨產品的核心基準升級和改進，以提高所有轉換和執行步驟的效能和穩定性。此版本中的一些主要功能和變更包括：

- 從 Java 8 到 Java 17 的版本升級，提高安全性和效能，並允許客戶部署和執行以更現代的語言實作的應用程式，並使用最新的第三方架構版本。
- 使用者或任務之間管理大型共用記憶體空間的其他支援，可在應用程式或執行個體重新啟動後儲存可重複使用的資料。
- 使用分頁機制更快速地存取 Blusam 中的大型資料集，以逐步擷取一部分的記錄。

如需此版本中包含之變更的詳細資訊，請參閱下列各節。

執行期 3.10.0 版

此執行時間以 Java17, Spring2.7 和 Angular16 為基礎。

zOS

新功能

- Blusam - 透過分頁機制新增對大型資料集的支援，其中使用頁面存放和載入索引

改進

- 增強 DataUtils.compare 來處理從字串到數字的較低優先順序轉換
- 新增支援，透過 YML 屬性 dataSimplifier.byteRangeBoundsCheck 來檢查未建立任何 ByteRange 且值不正確 dataSimplifier.byteRangeBoundsCheck
- 增強型 removeSOSI() 支援初始化具有空白字元的 GraphicAlphanumericType
- 新增了任務操作和安全 GDG 狀態讀取的穩健性
- Blusam - 新增透過名為 CoreBluesamManager.removeCache() 的新方法清除 Blusam 資料集的 Ehcache 支援
- Blusam - 改善一般 Blusam 資料集的刪除/重新命名行為
- Redis - 增強對解除鎖定資料集和清除記錄鎖定的支援
- JICS - 已改善失敗請求的錯誤訊息
- JCL - 根據點字元新增對 ControlM 變數串連的支援
- JCL - 新增對 GDG 檔案寫入建議 (ADV) 的支援
- JCL - 刪除所有 GDG 檔案後增強對目前世代號碼的支援
- JCL - 增強對資料集建立時目錄讀取 rdw/recordSize 的支援
- JCL - 新增在開啟具有資料輸出記錄大小的檔案時更新資源物件 （來自 AbstractSequentialFile) 的支援
- JCL - 改善的 IDCAMS 效能
- JCL - 新增「CHAR」做為「CHARACTER」的別名，增強對列印陳述式的支援
- SORT - 增強支援將操作從 Blusam 固定長度資料集複製到可變長度的資料集
- SORT - 增強的排序文法來處理某些特定陳述式

AS400

新功能

- 新增對使用者空間及其相關 APIs支援

- 新增對 SNDPGMMSG 的 TOMSGQ 參數和實作的訊息佇列的支援
- CL - 新增對 OVRPRTF 命令的 FILE 和 SPLFNAME 參數的支援
- CL - 新增支援使用 CPYF 命令處理對應分割區資料表的程式庫
- CL - 新增處理 CHGCURLIB 命令並在建置查詢時考量目前程式庫的支援
- CL - 新增在呼叫堆疊追蹤中處理 cl 命令的支援

改進

- 改善 MessageHandlingBuilder，以更好地處理呼叫堆疊追蹤項目
- 改善 contextPreconstruct 功能的平行執行
- 改善 SFLINZ 建立記錄時的顯示屬性
- 改善 SAVOBJ 以允許處理多個輸出檔案
- 從 Java 程式呼叫 groovy 程式時，透過將其新增至 programCallStack 來改善處理
- 改善協助模態的最高定位偵測
- 針對 SNDPGMMSG 提供 toMsgQ 參數時改善 toPgmQ 功能
- 改善擷取預先定義的訊息和訊息載入器的功能
- 改善內容中分隔符號字元的 CPYTOIMPF 處理
- 改善 READ 記錄的版本鎖定

橫向功能

新功能

- 新增前端系統訊息的翻譯
- 在 ExecutionContext 中新增了傳回程式呼叫堆疊的新方法
- 無論實際環境為何，設定行分隔符號 （用於資料簡化器）
- 新增設定 SQL 模型 JSON 路徑的可能性

改進

- 改善了涉及填補時的比較方法 DataUtils.compareAlphInt()
- 建立旗標以允許游標查詢中例外狀況的自訂行為

- 改善圖形 LOWVALUES db 轉換

第三方

- 升級以緩解 CVE-2024-21634, CVE-2023-34055, CVE-2023-34462, IN1-JAVA-ORGSPRINGFRAMEWORKSECURITY-5905484, CVE-2023-46120, CVE-2023-6481, CVE-2023-6378, CVE-2023-5072)

現代化工具 3.10.0 版

zOS

改進

- COBOL - 新增對 ABS 函數的支援
- JCL - 增強型變數範圍：連接至 STEP 而非 JOB
- 低/高值的增強游標參數注入
- 改善 CSD 剖析，特別是遠端交易

AS400

改進

- 已移除控制層級指示器的空白檢查
- 新增支援 IMPORT/EXPORT 關鍵字的外部名稱
- 新增對欄位 %LEN 的支援
- CL - 新增對 CLLE 語言新運算子的支援
- CL - 新增對巢狀 IF 的支援
- COBOL - 改善與多個金鑰搭配使用時的 START 命令處理
- DSPF - 使用記錄編號改善游標位置的處理
- DSPF - 已改善已簽章的數值、僅限數值欄位和大規模欄位的格式
- DSPF - 已改善 Screen General Help 標題的判斷
- DSPF - 改善對輸入/輸出規格的支援
- DSPF - 改善了在驗證數值欄位期間對分組分隔符號的處理

- 改善映射輸出/DDS 記錄
- 改善印表機檔案 REFFLT 關鍵字解析參考欄位的能力
- RPG - 增強對「ALL free」陳述式的支援
- RPG - 改善了條件剖析，並新增了在沒有結果 TAG 的情況下處理 CABXX 的支援
- RPG - 改善數值欄位的輸入規格處理
- RPG - 在 IF/ELSEIF/WHEN 條件下改善程序呼叫的處理
- RPG - 改善在 dspf 檔案上呼叫 READ 命令時的處理
- RPG - 改善對參考不存在 DDS 的檔案的支援
- 改善傳遞實體記錄格式名稱時的 REFFLD 處理
- 新增使用 'return' 做為 db 資料欄名稱的支援

橫向功能

新功能

- Oracle - 讓 SYS 能夠定義使用者來存放內建函數

改進

- 從 v8 升級到 v17 的 Java 版本
- 使用叢集資料欄名稱改善 SQL 條件
- 從檢視新增對 ORDER BY 子句的支援

版本備註 3.9.0

此版本的 AWS Blu Age Runtime 和 Modernization Tools 專注於跨產品的多個橫向增強功能，努力提高高可用性架構的效能，以及將任務執行提升到下一個層級的新功能。此版本中的一些主要功能和變更包括：

- 從 Angular 13 升級至 Angular 16 的版本，可提高安全性，並可存取改善客戶線上應用程式效能的新功能。
- 新增對 AS400 中跨任務功能的支援，其主要重點是任務可以同步傳送查詢訊息，從而在現代化任務中解耦。
- 改善 Redis 使用效能，包括連線集區最佳化、連線高度安全性和升級的資料集鎖定機制。

如需此版本中包含之變更的詳細資訊，請參閱下列各節。

執行期 3.9.0 版

zOS

新功能

- 排序程式：以固定長度更新 VSAM 輸入
- JHDB 資料庫：新增可設定的逾時

改進

- 如果用於檔案串連，則增強支援將行分隔符號串流
- 增強了開啟串連序列檔案的支援。開啟檔案後初始化 DataSetIndex
- 當 NumericEditedType 影響到數值時，增強對虛擬小數分隔符號的支援
- 增強對負值 NumericEditedType 的支援
- IDCAMS：SYSIN 卡現在使用 application-utility-pgm.yml 中定義的「編碼」屬性進行讀取
- IDCAMS：更新文法以支援 DEFINE CLUSTER 陳述式中的 FILE(..) 引數
- INFUTILB：新增對 DFSIGDCB 引數的支援，以覆寫 DD SYSREC 的 DCB 參數
- INFUTIL：增強對 "DFSIGDCB YES" 參數的支援
- 改善 SPLICE 以處理巨型輸入檔案
- DFSORT：改善備註欄位處理
- DFSORT：新增對（已簽署/未簽署）免費格式數字格式 (SFF/UFF) 的支援
- SORT：新增 OPTION PRINT 和 OPTION ROUTE 陳述式的剖析支援
- SORT/ICEMAN：新增對封閉分割操作的支援（具有 DIV 運算子的欄位）
- 使用一般金鑰增強對 CICS READ 的支援
- Function StringUtils.chargraphic 已修正，以從圖形類型中移除 SOSI
- 增強 DataUtils.isDoubleByteEncoding 的效能
- JCL：增強對臨時資料集的 KEEP 處置模式的支援。系統會將處置變更為 PASS
- JCL：動態處理 DCB 參數
- JCL：不正確值的增強型 SUM FIELDS 輸出
- JCL：CommonDDUtils：：getContent 現在會在目錄中搜尋 recordSize

- JCL：建立資料集時從目錄讀取 rdw/recordSize 屬性
- JCL：新增對 DCB= 的支援。MYDD 將 DD 的 DCB 參數複製到相同任務步驟中的另一個
- JCL：改善記錄大小繼承系統
- JCL：新增 (Redis) 專屬資料集鎖定
- Redis：新增獨立模式的 SSL 支援
- Redis：新增具有鎖定的同步 Redis 鎖定計數
- Redis：Redis 鎖定支援的集區參數
- Redis：使用 Redis 最佳化中繼資料重新整理
- Redis：改善 Redis 叢集支援
- 改善使用 IO 模式的開啟鎖定
- 改善的資料集會鎖定效能並清除未使用的鎖定
- 取消註冊檔案期間資料集的增強路徑
- 已改善預先擷取視窗快取失效
- 新增對執行緒安全公用程式資料來源提供者使用的支援
- 增強datasetState Nullity 檢查
- 增強對未重新開啟已開啟資料集的支援
- 新增了任務最終操作的穩健性
- 增強支援索引鍵的索引順序，允許重複
- 增強對略過清單序列化順序的支援
- 新增對偵錯傾印功能的支援，以協助診斷索引順序問題
- 增強對中繼資料重新整理的支援
- 增強對 Blusam 大量讀取的支援

AS400

新功能

- 建立應用程式內容登錄檔
- 支援 DSPF 關鍵字 CLRL(NO) 支援記錄鎖定監控
- 支援鍵控 DataQueue
- 批次任務的 INQUIRY 訊息支援

- 新增對 AS400 COBOL 的程式描述印表機檔案的支援
- 處理 RMVJOBSCDE cl 命令
- RUNSQL/DLYJOB 的改善
- CHKOBJ：提高參數 LIB 的舊版錯誤碼
- SNDPGMMSG：支援字串參數
- RTVDTAARA：改善 LDA 中的子字串
- DSPFD：針對特定檔案名稱新增支援的 FILE 參數
- RUNQRY：支援 QRY PARAM 中的 sql 檔案
- CRTDUPOB：支援在資料區域之間複製資料
- SBMJOB：將指示轉換為使用 JobQueueManager
- OPNQRYF：新增對 Qtemp 程式庫的支援
- CRTDUPOBJ：改善複製分割區內容的邏輯
- CRTDUPOBJ：新增對檢視 Qtemp 的支援
- RTVSYSVAL：支援 CL 命令中的 SYSVAL 值、QDATFMT
- CHKOBJ：新增對 OUTQ 的支援
- RTVJOBA：支援 SWS 參數
- SNDPGMMSG 和 RCVMSG：支援
  MSGF、MSGFLIB、MSGDTA、MSGTYPE、KEYVAR、MSGKEY、MSGID 的其他參數

改進

- 改善 WORKSTATION I/O 卡支援
- 改善對設定訊息重疊上一個訊息的處理
- 支援 array-messageline 上的其他訊息資訊
- 改善了 EVAL、SortA、圖形中的獨立陣列包裝函式存取
- 改善線上應用程式結束時DAOs 清理
- 新增對其他日期格式的支援，並改善字串輸入的處理
- 透過從 CL 命令 SbmJob 新增系統值協助程式類別解碼和建置參數，改善 SYSVAL 的 CVTDAT 處理
- 從 gapwalk-cl-command 元件掃描中移除套件 com.netfective.bluage.gapwalk.rt.blu4iv

- 改善對訊息佇列 API 預先定義訊息的支援
- 改善對在另一個程式中寫入記錄的 retrieveSubfileRecord 支援
- 改善對訊息佇列 API 立即訊息的支援
- 改善提交任務時本機資料區域的處理
- 伺服器啟動時自動啟動 JobQueues
- 使用 applicationContext 組態來解碼 SBMJOB 的參數
- 改善系統提供的錯誤訊息
- 允許 RTVMSG 在巢狀子目錄中搜尋 .properties 檔案
- 處理繫結至錯誤/無效指標的實體重設
- 改善 MessageHandlingBuilder，將 msgId 和 MsgFile 名稱顯示為 RCVMSG 的字串
- 改善訊息佇列 API 的 withMsgFileName 方法
- 改善資料區域鎖定機制
- RTVMBRD：支援參數 FILE 的大小寫
- CRTDUPOBJ：改善檢視的處理
- CPYTOSTMF：改善連線的處理
- CPYF：改善從一般檔案複製時的處理目錄名稱
- RCVF：為 groovy 和 java 正確處理 DEV/RCDFMT 參數和 RCDFMT 的轉換
- RCVF：處理後續呼叫並避免重設游標
- CPYF：新增從一般檔案寫入的支援
- CRTDUPOBJ：新增使用 Qtemp 程式庫處理新物件
- CHGDTAARA：資料區域長度上限從 256 增加至 2000
- SAVOBJ：確保儲存的記錄按插入順序排列
- RTVDTAARA：擷取的值（不裁剪）
- CHKOBJ：當成員不存在時傳回正確的監控訊息
- RTVDTAARA：新增對 LDA 子字串的支援
- RTVDTAARA：傳回最多為 RTNVAR 參數中指定變數長度的空格
- RTVDTAARA：支援啟動和長度的整數參數，並支援最新的轉換格式
- CHGDTAARA：新增對參數的支援，其中包含下限和上限
- CHKOBJ：處理參數物件類型的 VIEW 值

- CHKOBJ：如果檢視存在，無論成員為何，結果都會設為 true

橫向功能

新功能

- 處理產生報告至 .txt 檔案
- 將 currentSchema XA 資料來源屬性新增至秘密管理員
- 新增 database.cursor.raise.already.opened.error YAML 屬性，以在已開啟的游標開啟時啟用架構來引發 SQLCODE 錯誤 502

改進

- 將 gapwalk poms 新增至 Amazon EC2 封裝上的 AWS Blu Age
- 預設使用新的訊號處理常式範例
- 在處置為 MOD 或 OLD 時新增鎖定支援
- 新增快取以存放資料庫日期時間模式
- 改善 PackedType 的檢查功能
- 使用 VariableSizeArray 改善記錄的 DataUtils.setTo 函數
- 處理與執行單位相關的 MQ SYNCPOINT 選項
- 啟用架構以在轉返交易上設定 SQLCODE
- 根據引擎金鑰秘密新增自動驅動程式類別名稱
- 程式/交易逾時
- 存取游標時，在轉返之後還原游標位置

第三方

- 升級 SnakeYAML、Redisson 和 Amazon SDK、移除 YamlBeans（緩解 CVE-2022-25857, CVE-2023-24621, CVE-2023-42809, CVE-2023-44487)

現代化工具 3.9.0 版

zOS

改進

- 增強支援 XML-TEXT 做為字串類型目標的來源
- 增強的 STM 到 UML 工作流程，以支援 X/(Y/Z) 分割模式
- JHDB 資料庫：在任何資料庫更新之前接受 ROLLBACK 呼叫
- JHDB 資料庫：接受 ROLLBACK，即使交易已終止 (NOP)
- JCL：改善步驟驗證函數
- SORT：使用區域小數負值處理 SUM 函數
- COBOL：新增對字串常值中單/雙引號逸出的支援

AS400

改進

- 新增前導零，改善編輯程式碼 X 的內建函數 %editc 處理
- 改善僅輸入欄位初始值的處理
- 新增動作金鑰以協助對話方塊
- 動態資料表的頁尾記錄顯示在底部
- 為指定實際 RECORD-KEY 的檔案處理了不含 KEY PHASE 的 START 命令
- 新增浮點數和 NumberUtils：：pow 類型的預設值
- 新增使用 LIKE(IN) 定義變數的支援
- 更新 FOR 迴圈處理，以支援省略選用元素
- 更新 RPG 剖析，將記錄與 CTDATA 陣列名稱建立關聯
- 改善 CABxx 陳述式指標的處理
- 支援 COMMIT 關鍵字上的選用參數
- 改進了 LF 中的 FORMAT 關鍵字支援
- 具有高和相等（或低和相等）指標的受管 LookUP 操作程式碼
- 在雙引號中宣告的已處理 PF 金鑰名稱
- 改善 EDTCDE X 的處理，以免抑制前導零
- 改善未產生未命名標籤的印表機檔案中 MSGCON 的支援
- 欄位內容由多個資料結構共用
- 已處理的 ERRSFL 參數搭配 SFLMSG/SFLMSGID
- 改善完全免費 rpg 的 proc 宣告範圍之前的主要程式碼

- 新增剖析條件化控制規格
- 改善對 dataholdermapper 中 setErrSfl() 方法的支援
- 改善內部建立變數的類型解析度
- 改善對 Z-ADD opcode 的支援
- 改善使用 DFT 值處理常數欄位
- 改善程式狀態 ds 內整數欄位的支援
- ENTRY 參數中的已處理指標指派
- 改善透過 ref/reffield 關鍵字傳播的關鍵字篩選條件
- 支援的未命名 DataArea 資料結構
- 改善指標資料類型的處理
- 用於使用 LIKE 關鍵字定義變數的陣列處理元素支援輸出欄位中的陣列存取
- 改善對已簽章數字的支援，僅顯示數字
- O 卡上支援的邏輯關係
- 英數字元上 %CHAR 的測試案例
- 支援的控制項規格關鍵字主要
- 印表機檔案中有兩個參數的 EDTCDE
- 改善 FullFreeRPG 剖析
- 增強動態資料表，以確保頁尾已正確定位
- 新增使用 ALL 圖形常數初始化數值類型的支援
- 改善參考相同實體檔案的多個 RPG 邏輯檔案的處理
- 改善現代畫面中修改欄位的偵測
- 與動態欄位的模態同步
- 改善僅處理輸出簽署的數值欄位
- 改善 WORKSTATION I/O 卡支援

橫向功能

新功能

- Data Migrator 工具：新增了 ebcdicFilesWithVarcharInVB 屬性，允許讀取位元組時考慮 VARCHAR 2 位元組長度

- 實作常見的 API 來記錄錯誤
- 實作 BluAgeErrorDictionaryUtils，並使用常見的 API 在 COBOL2Model、RPGCycleBuilder、ReDefinitions2Model 和 FieldsProcessor 中記錄錯誤和/或資訊
- 改善 SQL 文法以支援不同的隔離子句定義

改進

- 已將角度版本升級至 v16
- 角度：將 ajv 版本從 6 升級至 8.9

第三方

- 升級 Groovy 至 2.4.15 版

版本備註 3.8.0

此版本的 AWS Blu Age Runtime 和 Modernization Tools 專注於跨產品的多個橫向增強功能，以提高其品質和安全性，以及改善快取效能和在單一分佈中支援的命令統一。此版本中的一些主要功能和變更包括：

- 從 Spring 2.5 升級至 Spring 2.7 的版本，增加平台的維護支援、效能和安全性。
- 整合了超過 82 個 CL 命令，支援 over-the-counter分發的一部分，以促進先前使用 CL 指令碼的現代化應用程式的使用和部署。
- 新的 APIs 可供操作並與 BluSAM 資料集互動，例如整合匯入至受管服務，以及列出資料集中繼資料資訊的功能。
- Redis 用量的效能改善和延伸，包括叢集模式中的可用性、高可用性資料擷取、秘密用量的標準化。

如需此版本中包含之變更的詳細資訊，請參閱下列各節。

執行期 3.8.0 版

zOS

新功能

- 將金鑰定義作為 DynamicFileBuilder 的字串來處理

- DFSORT：新增對 OUTFIL TRAILER1 + DFSORT 文法初始化中多個項目的支援
- CommonDDUtils 工具：處理串流內資料中的記錄大小
- 索引檔案：處理 GENKEY 選項

改進

- 不同 jar 中的外部化 BluSAM 載入服務
- 新增設定位置以存放暫存檔案的支援
- 改善多節點案例的共用快取機制
- 共用快取用量：IDCAMS 驗證最佳化
- 改善內嵌 select 的 ROWID 注入
- JCL：每個串流任務程序現在都會在不同的 Groovy 檔案中產生
- 確保 IDCAMS JCL 卡上的 card-demo-v2 涵蓋範圍
- BluSAM：使用多個執行個體時，避免重複warmUp
- 減少快取補充的記憶體使用量
- Jedis 集區組態支援
- 如果在檔案串連中使用，將行分隔符號新增至串流
- 支援 IDCAMS 公用程式中的 EBCDIC 卡 + 區塊註解 (/* ... /)
- 資料庫支援查詢：支援將 level49 轉換為 SQL 的雙位元組字串
- DFSORT 文法：實作 17 個控制陳述式 + 其中 2 個的整合 (OMIT/INCLUDE)
- 增強 GRAPHIC 資料欄擷取 INFUTILB
- 支援使用變數大小資料表讀取檔案
- 支援簽署 nibble 的 ZonedType，其中最後一個位元組的第一個位元為 'E'
- 如果記錄不符合任何 CHANGE 尋找常數，DFSORT/ICETOOL 新增對 NOMATCH=(..) 引數的支援
- Redis 叢集相容性
- 根據 groovy 結束代碼處理任務狀態（失敗）
- 改善 CICS SYNCPOINT ROLLBACK 支援。
- 預先擷取視窗以最佳化 Redis 快取用量
- JCL/GROOVY： DISP=(，PASS) 時，繼承上一個步驟資料集的 isRDW 屬性
- 使用可變大小陣列處理資料的部分複本

AS400

新功能

- 支援顯示檔案的 I/O 卡
- 支援 DSPF 關鍵字 ERRMSGID 和 CHKMSGID 的其他訊息資訊
- 支援前端畫面上的多個錯誤訊息
- 新增或改善對 gapwalk-cl-command 應用程式中 82 個 CL 命令的支援

改進

- 在承諾控制下改善對 DELETE 和 READ 的支援
- 內建 %dec 內的 ConvertDate
- 強制執行的 XSS 安全標頭
- 改善 STM 生成的穩健性和一致性（更好的處理方式：以自由格式 rpg 表示的接續行、小數部分的逗號、定義/宣告中的自由格式區塊）
- 改善 DataHolderMapper 產生
- 在 DataAreaFactory 中新增了穩健性和變更範圍
- 改善索引標籤索引鍵上的焦點轉移
- 改善產生 Jasper 報告的效能
- 改善了填補 0s 的小數顯示
- 改善對 INFDS 中 ROW/COL 欄位的支援
- 從畫面改善對修改欄位的支援
- 為產生的報告名稱和路徑新增 getter
- 已改善 Dataqueue 長度
- 改善任務佇列的自動設定，以符合 Spring Boot 2.7 中的新標準
- 改善多個並行工作階段的工作站更新

橫向功能

新功能

- 對 Packed 無無效資料容錯的支援

- 新增分頁/篩選以列出資料集端點

改進

- 資料欄與空白字串比較中的增強 ORACLE 查詢轉換策略
- 使用 DSNTEP 和 INFUTILB 公用程式處理 BLOB DB2。BLOB DB2 現在已現代化為 BYTEA 類型貼文。
- 改善刪除游標的最後一個項目
- 增強對刪除 RRDS 檔案的支援
- 改善 AWS Blusam 秘密效能
- 改善 SQL 架構中資料庫連線的處理
- 標準化 AWS 多資料來源秘密管理員金鑰
- 效能迴歸修正
- 改善 PackedType 的檢查函數
- 改善 PackedType 的 LOW-VALUE 處理
- 已升級 cognito 連線的彈簧安全封裝
- 未在DB2 目標資料庫上套用程式碼轉移點編碼和解碼

第三方

- 從 2.5 到 2.7 的 Spring Boot 升級

現代化工具 3.8.0 版

zOS

新功能

- JCL：使用歸位字元處理串流 "\r"

改進

- 改善日誌記錄，以防止使用 ON SIZE ERROR 子句現代化 DIVIDE 時除以零
- JCL：在程序中呼叫程序的增強支援

- 有不明確欄位時，支援 FORMATTIME CICS 命令中的 OF 關鍵字
- JCL：支援變數中的 Â¥ 字元
- JCL：根據先前的步驟計算 RC
- 使用 PL1 SUBSTR 時比較位元組而非字串
- 改善從單一來源初始化多維陣列
- 改善在 IF 區塊中涉及單一 SQL 查詢時的 COBOL 剖析

AS400

新功能

- 支援 CL 中的巢狀 IF 陳述式
- 改善對 RPG 自由格式中 ENDDO 陳述式的支援

改進

- 改善對調節控制層級的支援
- 使用 LIKE 改善原型傳回
- 改善處理函數 %months、%year、%days 的支援
- 支援整個畫面的說明功能
- 處理以參數傳遞的推論 BLANKS
- 使用 "" 運算子改善表達式 EVAL
- 在沒有 KEY 階段的情況下處理 START 命令
- 改善處理關鍵字 LIKEREC
- 改善未命名的子欄位
- 改善傳回未簽署類型的程序
- 改善對 RESET 操作（免費 RPG)、%CHAR 和 %DEC 內建的支援
- 改善內建函數 %LOOKUPXX
- 改善在沒有原型的程序中對 LIKEDS 關鍵字的支援
- 處理 Dim 關鍵字陣列類型 (VAR、AUTO)
- 改善對 XFOOT 的支援

- COBOL：改善對 RENAMES 欄位的支援
- CL：支援 while(true) 條件
- 使用 LIKE 關鍵字改善獨立陣列的處理
- 改善內建函數 %INT
- 改善 RPG 完整免費剖析
- 改善對連結中陣列的支援
- CL2GROOVY：支援 Select 陳述式
- 改善了 DSPF 關鍵字 "ERRMSGID"
- 改善使用前導零初始化位元組的處理
- 改善數值欄位的 authorizedValues
- 處理適用於自由形式 EVAL 陳述式的擴展器 H
- CL 到 Groovy：支援 LDA 的子字串
- 改善對記錄中 RESET 的支援
- 使用參考改善 EDTCDE 和 EDTWRD 的處理
- 改善 DDS 欄位的輸入欄位映射
- 已改善對 IN 陣列中 MOVEA 字元的支援
- 使用 LIKEDS 關鍵字改善原型
- 已改善對 DSPF 關鍵字 DSPATR 的支援
- 改善使用 +/- 的 D 卡剖析
- 在程式呼叫中新增了穩健性
- 在欄位解析程序中新增了穩健性

橫向功能

改進

- FrontEnd：模擬 IME 輸入的貼上事件

第三方

- 從 2.5 到 2.7 的 Spring Boot 升級

版本備註 3.7.0

此版本的 AWS Blu Age Runtime 和 Modernization Tools 主要包括更完善支援命令和公用程式的增強功能、與 AWS Secrets Manager 整合的功能，以及新的監控功能。此版本中的一些重要變更如下：

- 多個執行期元件現在可以使用 AWS Secrets Manager 來提高現代化應用程式的安全性設定，主要與公用程式資料來源、適用於 TS 佇列的 Redis、BluSam 快取和鎖定相關。
- 監控端點，允許 擷取交易、批次和 JVM 指標，以進行資源用量最佳化和操作管理，例如狀態、持續時間、磁碟區等。
- 支援 RPG 中 IBM MQ 呼叫的新功能，並提高 JCL SORT 和 IDCAMS 轉換涵蓋範圍。

如需此版本中包含之變更的詳細資訊，請參閱下列各節。

執行期 3.7.0 版

主題

- zOS
- AS400
- 橫向功能

zOS

新功能

- 使用 SQL 例如文法，改善程式公用程式應用程式中涉及的剖析查詢。(V7-9401)
- 偏移時處理索引的變數大小陣列 (V7-9904)
- 支援將 SQL TIME 資料欄插入 DB2，格式為 24：00：00 (V7-10023)
- 支援使用 FOR ROWS 和 ATOMIC 選項 (V7-10105) 的陣列的 INSERT SQL 查詢
- JCL SORT - 增強 TranscodeTool 以使用 IFTHEN 支援 OUTREC (V7-10124)
- JCL SORT - 在 OUTREC 命令 (V7-10125) 中新增對 DATE 關鍵字的支援
- JCL - 新增對串流內程序的支援 (V7-10223)

改進

- 所有任務步驟 (V7-9504) 都應提供標示「PASS」處置的資料集

- 支援 JCL 屬性 SCHENV (V7-9570)
- 支援 SEND 搭配 CTLCHAR 選項 (V7-9714)
- COBOL - 處理 ACCEPT 陳述式中的不同行分隔符號字元集 (V7-9875)
- 避免多次轉返 (V7-9958)
- 允許使用 MOD 處置附加在 GDG 檔案結尾 (V7-10031)
- 最佳化： putAll refactoring (V7-10063)
- PutAll 重構：新增分頁 (V7-10063)
- 讓 Jedis 用戶端讀取逾時可設定 (V7-10063)
- 獨立模式的 UseSsl 支援 (V7-10114)
- 成功開啟檔案後支援 EIBDS (V7-10147)
- 在檔案控制請求後支援 EIBDS (V7-10147)
- 改善 CICS SYNCPOINT 支援 (V7-10187)
- BluesamRedisSerializer：metadataPersistence (V7-10202) 的問題
- 支援 TS 佇列的 Redis AWS Secrets Manager (V7-10204)
- 支援 JCLBCICS 自訂 DD 名稱大小 (V7-10224)
- 在 IDCAMS DELETE 陳述式 (V7-10308) 中新增對絕對路徑的支援

AS400

新功能

- 實作 AS400 畫面的說明功能 (V7-9673)

改進

- INFDS (V7-9377) 中的記錄數量

橫向功能

新功能

- 支援 EC2 上的執行期將日誌傳送至 Amazon CloudWatch (D87990246)
- 新增端點以擷取有關批次、交易和 JVM (D88393832) 的指標

改進

- 支援公用程式 pgm 的資料來源 AWS Secrets Manager (V7-9570)
- 新增對 DSNUTILB DISCARD (V7-9798) 的 Db2 支援
- 支援在預設 SYSPRINT 和 SYSPUNCH 檔案 (V7-10098) 中寫入記錄器而非預設系統輸出串流
- 支援 BluSam Redis 快取並鎖定 AWS Secrets Manager (V7-10238) 中的連線屬性
- 支援 Db2 XA AWS 秘密上的 SSL 連線 (V7-10258)
- 更新 IDCAMS REPRO 和 VERIFY (V7-10281) 的中繼資料
- 改善 IDCAMS 中止傳回程式碼管理 (V7-10307)

現代化工具 3.7.0 版

主題

- zOS
- AS400
- 橫向功能

zOS

新功能

- PLI - 已改善陣列區段和二維陣列的指派 (V7-9830)

AS400

新功能

- 處理控制層級指標 (V7-9227)
- 支援 EXTNAME 參數 *INPUT (V7-9897)
- 增強型 Goto 重寫：支援 SELECT OTHER 陳述式 (V7-9973) 中的標籤
- 支援 REFSHIT DSPF 關鍵字 (V7-10049)

改進

- 改善處理檔案描述關鍵字 EXTIND(*INUx) (V7-7404)

- 改善 SQLDDS 檔案轉換 (V7-7687)
- 不再為 AS400 檔案產生檔案物件 (V7-9062)
- 改善處理檔案描述關鍵字 EXTDESC (V7-9268)
- 改善處理 %CHAR 內建 (V7-9311)
- 改善對不使用 SFLEND (V7-9322) 之最後一筆記錄的分頁支援
- 改善對字首資料結構的支援 (V7-9436)
- 支援使用 %SIZE (V7-9472) 定義的維度
- 支援處理雙引號內宣告的 PF 欄位名稱 (V7-9557)
- 改善檔案操作 - 不區分大小寫 (V7-9785)
- 支援初始化為 *USER (V7-9806) 的欄位
- 支援 AS400 (V7-9840) 中的 COMP 類型
- 改善 (Not)InvalidKey (V7-9922) 上的 COBOL400 剖析
- 改善 SCAN 操作的處理 (V7-9971)
- 改善對 GOTO opcode (V7-9973) 的支援
- 改善 EXCEPT 操作的處理 (V7-9977)
- 改善字首支援 (V7-10000)
- 支援 RPG 中的 MQ 呼叫 (V7-10007)
- 已改善 %LOOKUP 內建 （金鑰陣列資料結構） (V7-10022)
- 支援關閉 *所有操作 (V7-10036)
- 支援 UPDATE AS ROW CHANGE SQLDDS 陳述式 (V7-10051)
- 改善處理常值類型長 (V7-10073)
- 改善 RPG 文法 （使用關鍵字 INZ 做為子常式的名稱） (V7-10074)
- 改善 RPG 文法，以支援具有空分數部分的數值 (V7-10077)
- 改善對 CL 和外部檔案 (V7-10081) 之間共用欄位的支援
- 改善對 DDS 條件式指標的支援 (V7-10084)
- 支援搭配 COBOL 程式的 DDS 二進位類型 (V7-10100)
- 改善與連結的名稱衝突 (V7-10109)
- 支援混合主要和匯出程序 (V7-10112)
- 改善子程序中 DataStructure 的支援 (V7-10113)

- 改善對 CLEAR (V7-10126) 的支援
- 改善對 DO 迴圈的支援 (V7-10134)
- 在全免費 RPG 中支援 SQLTYPE (V7-10151)
- 改善 DDS 關鍵字 (V7-10155) 的條件剖析
- 改善 DSL 產生 (V7-10163)
- 當條件是二進位表達式時，processIndicators 的改進。(V7-10164)
- 改善具有 Else 條件GOTOs (V7-10168)
- 支援在 DSPF 中鍵入時間和時間戳記 (V7-10173)
- 改善 DDS (V7-10183) 接續行的剖析
- COBOL 支援 RENAMES FLD OF RECORD (V7-10195)
- 改善了 DSPF 欄位的條件式指標剖析 (V7-10221)
- 支援剖析 DDS 關鍵字 NOALTSEQ (V7-10288)
- 支援說明功能表和隱藏欄位 (V7-10314)
- 改善的 DSPF 協助關鍵字健全狀態檢查 (V7-10328)
- 不再傳播 Ref 欄位上的所有關鍵字 (V7-10347)

橫向功能

新功能

- Data Migrator - 處理 CLOB 資料 (V7-9665)

改進

- 透過 JobContext (V7-10225) 將 JCL 屬性 SCHENV 從 JOB 傳播至 PROC GROOVY 定義
- FrontEnd - 在沒有邊界的情況下調整視窗大小 (V7-10358)

版本備註 3.6.0

此版本的 AWS Blu Age Runtime 和 Modernization Tools 為 zOS 和 AS400 舊版遷移提供了新功能，主要用於擴展 CICS 支援機制、補充 JCL 功能、最佳化並行和大量功能的效能，以及新增multi-data-source功能。此版本中的一些重要變更包括：

- 增強 JCL 動態檔案處理、擴展目前陳述式和管理串連資料集、在單一區塊中執行多個陳述式，以及從批次到程式的資料傳輸。
- 增強對多個 CICS 命令的支援，包括查詢多種 CICS 資源類型。
- 使用 Blu Age Runtime Utilities 時具有不同資料庫的功能，最適合業務資料分散到多個來源的情況。

如需此版本中包含之變更的詳細資訊，請參閱下列各節。

執行期 3.6.0 版

主題

- zOS
- AS400
- 橫向功能

zOS

新功能

- JCL - DynamicFileBuilder - 增強型檔案處理管理 (V7-9408)
- 呼叫 INFUTILB UNLOAD 公用程式 (V7-9554) 時，某些內建 SQL DB2 函數上的增強型格式轉換
- 增強型 PLI 多維陣列指派 (V7-9592)
- 處理 sysout 重新導向至檔案 (V7-9992)

改進

- 新增 DB2 RDBMS (V7-9155) 預存程序的觸發
- SORT 處理轉換為 PDF 格式 (V7-9286)
- JCL/GROOVY - 增強 REPRO 陳述式以支援 DUMMY 資料集 (V7-9424)
- 改善 CICS UNLOCK 支援 (V7-9606)
- 處理 Union (V7-9648) 的預設值大小
- JCL/GROOVY 處理串連資料集中的不同終止/處置 (V7-9653)
- 讓 pageSize 適用於 Blusam 資料集 (V7-9680)
- DSNUTIL - 允許在 DB2LUW (V7-9697) 中載入 24：00：00 作為有效的 TIME

- 在 NumberUtils.ne() NumberUtils.eq()(V7-9731) 中支援 HIGH-VALUES (0xff) 比較
- JCL/GROOVY - 支援 DO ... IDCAMS IF-THEN-ELSE 子句中的 THEN 關鍵字，可在單一區塊 (V7-9750) 中執行多個陳述式
- 稱為 JHDBBatchRunner (V7-9782) 外部程式的無效 JHDB
- 支援 SORT OUTFIL 控制卡 (V7-9808) 中的空格字元
- 改善 CICS READ PREV 支援 (V7-9845)
- 改善資料集索引的並行存取 (V7-9864)
- 改善 CICS REWRITE 支援 (V7-9873)
- COBOL - 支援 ACCEPT 陳述式中的多行 SYSIN，將資料從批次 (JCL) 傳遞至程式 (COBOL) (V7-9875)
- Groovy - 在檔案建立步驟中更好地處理 ConcatenatedFileConfiguration (V7-9876)
- IDCAMS UTILITY - DEFINE PATH 陳述式的處理 (V7-9878)
- SORT 建置 - 調整 TRAN 選項並處理隱含空白 (V7-9925)
- 透過 GENERIC 選項支援 (V7-9939) 改善 CICS DELETE
- 改善 CICS STARTBR 和 ENDBR 支援 (V7-9952)
- 改善並行存取的關閉效能 (V7-9953)
- 改善啟動時的檔案狀態處理 (V7-9991)
- Groovy - 允許呼叫 ConcatenatedFileConfiguration (V7-10012) 上的 getDisposition()/ getNormalTermination()/getAbnormalTermination() ConcatenatedFileConfiguration

AS400

新功能

- 支援 COMMIT 關鍵字上的外部指標 (V7-6035)
- SFLCTL 寫入後重設 ReadC 迴圈 (V7-8061)
- CALL 中的支援 LR 指標 (V7-9250)
- 新增動態欄位（分割）的新類型，以處理多行 (V7-9370) 上的輸入欄位
- 支援主要/次要檔案 (V7-9390)
- 本機資料區域現在會在提交任務時傳遞至呼叫的任務 (V7-9775)
- 支援資料區域的 QTEMP 和資料rea 值建立的支援。(V7-9916)
- 承諾控制 - 支援啟用/停用承諾控制 (V7-9956)

- 支援 COMMIT 關鍵字上的外部指標

改進

- 改善 0 值顯示和 EDTWRD (V7-8933)
- 支援 DSPF 關鍵字 "CHKMSGID" (V7-9125)
- 批次終止時的 SQL 遞交交易 (V7-9232)
- 改善對欄位和資料結構的關鍵字 EXPORT 和 IMPORT 支援 (V7-9265)
- 在 DateHelper (V7-9461) 中支援小寫
- 支援將 *CYMD 轉換為 *ISO （數值）(V7-9488)
- 改善不同欄位的內建 %len 控制代碼 （表達式的左側和右側）(V7-9733)
- 改善對內建函數 '%LOOKUPXX' XX ("LE"、"LT"、"GE"、"GT") (V7-10064) 的支援

橫向功能

新功能

- CICS - 改善選項狀態的查詢交易 (V7-9712)
- JCL - 使用系統輸出檔案改善 sysprint 的負載 (V7-9797)
- CICS - 改善 INQUIRE TSQUEUE (V7-9823)
- CICS - 改善選項 userid 的查詢終端機 (V7-9906)

改進

- 改善與空白比較的處理方式 (V7-8047)
- 改善 Jics 和 Blusam (V7-8847) 的記錄
- 支援動態欄位 (V7-8857) 的 BMS 延伸屬性 SOSI 和程式設計符號 F8
- 處理程式參數中的緩衝區溢位 (V7-9138)
- 改善 Blusam 鎖定登錄檔 (V7-9505) 的執行緒寫入並行
- 支援 Utility-pgm (V7-9570) 的多個資料來源組態
- Blusam 僅記錄層級鎖定模式 (V7-9626)
- 確保中繼資料持久性拒絕伺服器重新啟動 (V7-9748)
- 改善例外狀況時的 DAO 清除 （瀏覽器關閉）(V7-9790)

- 支援 INFUTILB SYSPUNCH (V7-9799) 的 DummyFile
- 增強對 NumericEditedType (V7-9935) 負值的支援

現代化工具 3.6.0 版

主題

- zOS
- AS400
- 橫向功能

zOS

新功能

- JCL - 增強程序結束的記錄 (V7-8509)
- PL1 - 增強 PakedLong (V7-8917) 資料類型的包產生
- JCL - 當檔案包含「結束」標記 // (V7-9509) 時，增強程序結束的記錄
- PL1 - 使用固定點和 SYSIN 串流增強對 GET EDIT 的支援 (V7-9593)
- DB2 - 增強對 VARGRAPHIC DB2 類型的支援 (V7-9809)
- CICS - 改善選項 LOGMESSAGE (V7-9969) 的命令 QUERY SECURITY
- PL1 - 改善 CHARG/chargraphic built-in (V7-9989) 的包產生

改進

- PL1- 增強對 INCLUDEX 關鍵字的支援 (V7-9588)
- PL/I - 將 CHARGRAPHIC 關鍵字視為任何方法呼叫的有效參數 (V7-9589)
- 以特定字元 @ # $ § 命名時改善 PL1 主機變數解析度。(V7-9654)
- COBOL - 支援 C01...C12 和 S01...S05 關鍵字做為剖析步驟 (V7-9669) 中 WRITE ADVANCING 陳述式的參數

AS400

新功能

- 在 Analyzer 中支援 SQL-DDS 轉換 (V7-7687)

- 自動化 SQL-DDS 檔案偵測 (V7-7687)
- 實作 SQL-DDS 預先處理 (V7-7687)
- 支援 ALIGN 關鍵字 (V7-9254)
- 支援 ExtName 到 DSPF 和多晶片陣列 (V7-9663)
- 在 COBOL WRITE (V7-9793) 上支援 InvalidKey 陳述式

改進

- TESTB opcode (V7-8865) 的改善
- 改善 DECFMT 對焦點的支援 (V7-8933)
- 在 MOVE 上處理產生的指標 (V7-9224)
- 改善對欄位和資料結構關鍵字 TEMPLATE 的支援 (V7-9278)
- 改善 LIKEDS （使用 LIKEDS 定義的DS 會自動符合資格） (V7-9302)
- COBOL - 改善指標結構的產生 (V7-9423)
- 原型中的 Const 參數不是唯讀 (V7-9437)
- 使用編輯碼 "Y" (V7-9443) 改善 EDTCDE 關鍵字
- 支援在 PSDS 和 INFDS (V7-9487) 中產生 *ROUTINE 欄位
- 將重寫欄位 XXX 改善為獨立 （重寫時遺失預設值） (V7-9522)
- 改善對 DSPF 關鍵字的支援 (V7-9658)
- 處理二進位 (V7-9666) 上的 ZEROES 預設值
- 支援隱含指標 (V7-9719)
- 使用一個參數 (V7-9730) 改善內建呼叫 %size 的處理
- 改善處理內建呼叫中的資料結構參考 (%ELEM) (V7-9736)
- 使用定義規格 (V7-9738) 中的 LIKE 參考改善欄位的已簽署長度處理
- REWRITE (V7-9791) 的改善
- 改善從 DDS 檔案產生索引 (V7-9803)
- 使用無效的數值 (V7-9813) 改善映射器的穩定性
- 改善 SQLModel 和 allIndexes 檔案產生 (V7-9818)
- 改善合格的 DS 支援 (V7-9863)
- 改善對 lookUP 的支援 （使用獨立欄位 LIKE a DS in 參數） (V7-9961)

- 改善 LIKE on 指標 (V7-9985)
- 在 MVR 上處理產生的指標 (V7-9995)
- 支援字元 N 搭配波狀符號 (V7-10021)
- 改善從 SQLDDS 舊版檔案產生現代 DDL 檔案 (V7-10067)

橫向功能

新功能

- 使用 yml 屬性自訂資源位置 (D88816105)
- COBOL - 支援 EXIT PERFORM 陳述式退出內嵌 PERFORM，而無需使用 GO TO / PERFORM ... 透過 (V7-9582)
- 指定要考慮全域中繼資料的預設舊版編碼。(V7-9883)

改進

- 改善遮罩產生 (V7-9602)
- 改善內容暖機期 (V7-9621)
- 讓 Charset CUSTOM930 執行緒安全。(V7-9674)
- MOVEA (V7-9773) 的改善

版本備註 3.5.0

此版本的 AWS Blu Age Runtime 和 Modernization Tools 為 zOS 和 AS400 舊版遷移提供了新功能，主要面向資料集和簡訊最佳化，以及延伸的 Java 功能作為轉換程序的結果資產。此版本中的一些重要變更包括：

- 除了現有的 groovy 指令碼功能之外，還能夠將 CL 程式遷移至 Java，以促進其與其他現代化程式的整合，並透過整合產生的程式設計語言來簡化客戶學習曲線。
- 使用新的資料大量功能，減少和最佳化 Redis 中資料集負載的效能。
- 能夠在任務步驟中操作和傳遞資料集，以現代化傳統資料集行為。
- 延伸 SQL 遷移以支援 VB 輸入檔案和 Java 11 簡化遷移。
- 與 IBM MQ 更快速整合的多種新機制，包括其他標頭、延伸 GET/PUT 支援和佇列中繼資料的自動擷取。

- 資料集中繼資料的 REST 端點，以及從 S3 儲存貯體匯入資料集。

如需此版本中包含之變更的詳細資訊，請參閱下列各節。

執行期 3.5.0 版

主題

- zOS
- AS400
- 橫向功能

zOS

新功能

- JCL SORT - 處理新的關鍵字浮水印 (V7-9409)
- ZOS COBOL - 增強對浮動字元的支援 (V7-9404)
- RedisJicsTSQueue 的連接埠至 RedisTemplate 和 ListOperations (V7-9212)
- ZOS JCL - 如果透過 UserDefinedParameters (V7-9012) 定義，請使用檔案目錄增強暫時目錄的路徑
- 使用 ALL 處理 FUNCTION ORD-MAX （所有陣列項目） (V7-9366)
- 在 Redis (V7-9212) 中存放 TS 佇列時，現在使用字首和人類可讀金鑰
- 新增 Blusam API 的取得資料集端點
- JCL - ADD 對批次任務的支援，其名稱涉及特殊字元，例如 # (V7-9136)
- TSModel 擷取現在可依需求穩健執行 (V7-9212)

改進

- LNK 檔案 (V7-6022) 中的非版本 INCLUDE 支援
- MQ - 增強編碼支援 (V7-9652)
- 改善對不同字元類型的雙位元組或混合字元集的支援 (V7-9596)
- JCL - 支援 IDCAMS 中的 filesDirectory 組態刪除 NONVSAM 陳述式 (V7-9609)
- 支援從檔案載入 ESDS 和 RRDS 資料集的大量模式 (V7-8639)
- 在輸入模式中處理空 ESDS 的開啟。(V7-9287)

- 使用 ORD/UNORD 縮寫支援增強 DEFINE CLUSTER 陳述式 (V7-9451)
- Blusam Redis 鎖定效能改善 (V7-8639)
- 增強 DEFINE CLUSTER 陳述式，以支援 DATA() 引數範圍 (V7-9337) 中提供的 RECORDSIZE
- 新增對 DEFINE CLUSTER 陳述式 (V7-9419) 上 BUFFERSPACE/UNIQUE 屬性的支援
- 改善可變長度記錄資料集的 Blusam 讀取操作。(V7-9391)
- CICS ADDRESS 正確將缺少的 CWA 表示為 null (V7-9491)
- 移除終端鎖定時不必要的寫入 (V7-8639)
- 在快取中處理 Redis 快取範本注入 (V7-9510)
- 正確解碼 BPXWDYN 參數 (V7-9417)
- LISTCAT 匯出耗用量的改善 (V7-9201)
- Blusam TS 佇列名稱 (V7-9212) 中不可列印的字元支援
- 處理具有映射集 null (V7-9486) 欄位的接收映射建置
- 改善動態存取模式的 BluesamRelativeFile 刪除和重寫操作。(V7-8989)

AS400

新功能

- 新增功能，透過標準 DS/STM 樞紐 (V7-9427) 將 CL 檔案產生為 Java 程式
- 支援具有 ADD 模式的輸入檔案 (V7-9378)
- 改善排序順序和擷取管理，以支援 cl 命令 OPNQRYF （開放查詢檔案），並在 OverrideItem 中新增對 SHARE 參數的支援。(V7-9364)

改進

- UpdateSubfile (V7-8061) 支援 SFLNXTCHG
- 執行 CL 命令 (V7-9624) 時修改 CL 內容的範圍
- 處理程式 BPXWDYN (V7-9417) 的傳回碼
- 清除本機監視器。(V7-9624)
- 支援 DSPF 關鍵字 RTNCSRLOC (V7-9389)
- setOnGreaterOrEqual() 不設定為等於 1 (V7-9342)
- UpdateSubfileRecord (V7-9376) 上的更新欄位快取

- 改善支援 SFLNXTCHG (V7-8061)

橫向功能

新功能

- 忽略常值圖形字串上的 G 字首。(V7-9420)
- ZOS COBOL - 增強對某些特殊結構 (V7-9485) 的 Fiedl.initialize() 支援
- 允許以非同步方式初始化內容，以改善程式啟動的效能 (V7-9446)
- SQL Release 明確表示開啟的準備陳述式和 ResulSet。(V7-9422)
- 增強 JMS MQ - 支援 MQ PUT 的 MQRFH2 / V7-7085 - 支援預設佇列管理員 (V7-9400)
- SQL 管理 - 啟用 SET 命令 (V7-9492) 參數的 Lambda 轉換
- ZOS MQ JMS - 將支援新增至 MQCOMIT 和 MQBACK (V7-9399)
- ZOS IBMMQ - 增強對 MQINQ (V7-9544) 的支援
- 使用雙位元組編碼時，使用位元組而非字串處理 CONCAT 操作。(V7-8932)
- ZOS IBMMQ - 使用選項 SET_ALL_CONTEXT (V7-9544) 增強對 PUT 命令的支援

改進

- 使用 $ 字元處理 gdg 檔案名稱 (V7-9066)
- 當先前的 SQL 陳述式成功時，SQL Diagnostic 會將 1 傳回為 NUMBER 子句。(V7-9410)
- 非 null 長度欄位的概述 (V7-7536)
- 支援內建 PL1 GRAPHIC 函數 (V7-9245)
- MQ - 新增 MQGMO 欄位設定的版本支援 (V7-9500)
- JMS MQ GET - 訊息傳回dataLength改進 (V7-9502)
- 設定 sqlerrd(3)，其中包含 ROWSET 內容中擷取的項目數量。(V7-9371)

現代化工具 3.5.0 版

主題

- zOS
- AS400
- 橫向功能

zOS

新功能

- ZOS PLI - 支援以二進位表達式指派的星號索引 (V7-9178)
- JCL 到 BatchScript - "//" 表示任務執行結束 (V7-9304)
- ZOS PLI - 增強對浮動字元的支援，並登入已編輯的數值類型 (V7-8982)
- COBOL - 支援內建 SUM 函數 (V7-9367)
- JCL - 選擇性，在 null 陳述式 (//) (V7-9202) 之後註解無效代碼
- JCL- 條件陳述式中運算子 '|' 的支援 (V7-9499)
- PL/I - 預先處理步驟中編譯前指令的註解，以防止剖析例外狀況 (V7-9507)

改進

- 使用分隔符號處理串流定義 (V7-9615)
- 改善 LISTCAT 匯出處理。(V7-9201)
- PL/I- 增強以支援隱含 'null' 引數 (V7-9204)

AS400

新功能

- 支援 DDS 關鍵字 CONCAT (V7-9439)
- 重構為 DSPF 關鍵字產生的 Java 程式碼。(V7-7700)
- 在資料結構定義 (V7-9029) 內的欄位上支援變動關鍵字

改進

- 改善邏輯關係和/或的剖析 (V7-9352)
- COBOL 改善 vo 和 dsEntity (V7-9449) 之間的映射
- 如果數值輸入聚焦，則顯示空白值 (V7-9374)
- SQL Declare 游標中的本機變數 (V7-9456)
- 空白 DS 的範圍問題 (V7-9466)
- 剖析 (V7-9632) 之前，在 col 80 之後截斷行

- 改善定義規格 (V7-9358) 中關鍵字 (DIM、LIKE...) 中欄位參考和內建呼叫的處理方式
- 支援 SQL 註解 (--) (V7-9632)
- FullFree 剖析，輸入Date/Time/Timestamp (V7-9542)
- 從 FullFree 剖析 (V7-9333) 包含 SQLCA
- 改善控制層級的支援。(V7-9610)
- 使用 *BLANKS (V7-9668) 處理 DS 比較
- 改善 DDS (V7-9318) 中多個指標的支援
- 改善對多個 DSPF 計畫的支援 (V7-9657)
- 使用 LIKE 改善欄位的處理方式（陣列中類似資料結構和類似資料結構的情況）(V7-9213)
- 免費 RPG，處理常值接續 (V7-9686)
- 改善對程式結束記錄的支援 (V7-9452)
- CALL 陳述式中的 LINKAGE 片語支援。(V7-9685)
- CASXX 操作代碼（不含 CASXX 群組的 CASBB) (V7-9357)
- 改善 FullFreeRPG 剖析 (V7-9457)
- 內建 %LEN 不支援 DS 做為引數 (V7-9267)
- 當因素 2 為 *ALL'X...' 時的 MOVEA 改善 (V7-9228)
- 支援使用 RENAME 欄位指派 (V7-9385)

橫向功能

新功能

- SQL Migrator 工具 - 在 ebcdic 載入步驟為可變記錄長度新增 OID 選項。(V7-9380)
- SQL Migrator 工具 - 支援 Java 11 on OID 選項 (V7-9599)

改進

- 改善對巢狀陣列的支援 (V7-9595)
- 如果原始編碼支援 Â¬，請將 Â¬ 字元取代為 ！。(V7-9465)
- JCL - 支援 PASS 正常終止以在任務步驟之間共用資料集 (V7-9504)
- 處理 VARCHAR 和 nullable db 資料欄類型時，將 ON NULL 套用至 ORACLE 上的資料欄定義。(V7-9681)
- 改善 Spring Injection 合規 (V7-9635)

AWS Blu Age 安全性漏洞

常見漏洞與暴露 (CVE) 是公開已知網路安全漏洞的參考清單。每個項目都包含識別碼、說明，以及至少一份公開參考內容。

我們建議您一律升級至最新的 AWS Blu Age 版本，以防止已知漏洞。使用 Amazon Inspector 持續執行安全性掃描，並根據問題清單在 NIST 中的嚴重性進行分類。

下列清單詳細說明了每個可用次要版本中修正的 CVEs，這是由於使用相依性所導致：

版本	CVE
4.8.0	CVE-2025-22228
4.7.0	CVE-2025-25193, CVE-2025-24970, CVE-2024-47535, CVE-2024-38820, CVE-2024-38809, CVE-2024-22262, CVE-2022-45868
4.6.0	CVE-2024-12801, CVE-2024-12798, CVE-2024-50379, CVE-2024-56337
4.5.0	CVE-2024-47535, CVE-2024-52316, CVE-2024-47535, CVE-2024-38827
4.4.0	CVE-2024-38820, CVE-2024-38821, CVE-2024-38809, CVE-2024-38816, CVE-2024-47554, CVE-2024-6484, CVE-2024-6485
4.3.0	CVE-2024-43788, CVE-2022-25898, CVE-2021-30246, CVE-2024-21484, CVE-2024-34750
4.2.0	CVE-2020-11023, CVE-2023-26364 , CVE-2019-11358, CVE-2020-11022, CVE-2021-23358 , CVE-2017-18214, CVE-2022-24785, CVE-2022-31129, CVE-2023-48631

版本	CVE
4.1.0	CVE-2024-29025, CVE-2024-23080, CVE-2024-22262, CVE-2024-30171, CVE-2024-29857, CVE-2024-30172
4.0.0	CVE-2016-1000027, CVE-2022-1471, CVE-2024-1597, CVE-2024-22243, CVE-2024-22233, CVE-2024-22234, CVE-2024-22259, CVE-2024-22257, CVE-2024-29131, CVE-2024-29133

> **Note**
>
> 如需先前發行版本中修正CVEs 詳細資訊，請聯絡您的 AWS Blu Age 交付管理員

升級 AWS Blu Age 的說明

此頁面包含升級 AWS Blu Age 版本的指示。

常見升級

在大多數情況下，升級 AWS Blu Age Runtime （非受管） 版本時，您應該將舊版的成品 (WARs、組態檔案、指令碼等） 取代為新版本中提供的內容，並重新啟動應用程式。升級後，請務必對現代化應用程式執行廣泛的迴歸測試。您也可以聯絡 AWS Blu Age 交付管理員，以取得適用於您應用程式的特定指示。

若要升級 AWS Blu Age Runtime （受管） 版本，請參閱 受管執行期環境。

某些升級可能需要額外的組態，以確保相容性。在這種情況下，請遵循該特定升級的指示。

從 3.10.0 遷移至 4.0.0

4.0.0 的主要變更是從 Spring Boot 2.7 遷移至 Spring Boot 3.2，以及從 Tomcat 9 遷移至 Tomcat 10。

程式碼變更

本節列出使現代化程式碼與 AWS Blu Age Runtime 4.0.0 相容所需的變更。如果您決定使用 Blu Insights (Transformation Center) 上的 4.0.0 版本啟動新一代，可以略過本節。

POM 變更

群組	ArtifactId	變更
org.slf4j	slf4j-api	移除（是暫時性相依性）
org.yaml	snakeyaml	移除（是暫時性相依性）
org.springframework.boot	spring-boot-starter-web	- 升級 spring.boot.version 至 3.2.4 - 移除 log4j-to-slf4j 的排除
org.springframework.boot	spring-boot-starter-jta-atomikos	變更為 com.atomikos：transactions- spring-boot3-starter：6.0.0
org.apache.commons	commons-dbcp2	升級至 2.10.0
org.postgresql	postgreql	升級到 42.7.2
com.microsoft.sqlserver	mssql-jdbc	升級至 12.4.211
com.oracle.database.jdbc	ojdbc8	變更為 ojdbc11 23.3.0.23.09 版

從 Javax 遷移至雅加達

tomcat 升級隨附從 Javax Java 套件遷移至雅加達的作業。請務必相應地將匯入從 javax.* 更新至 jakarta.*。

Javax 套件中幾乎所有舊的參考類別都可以在雅加達找到。已知例外狀況是 `javax.sql`和 `javax.xml`套件，這些套件仍然保持不變。

Atomikos 變更

由於上述相依性變更，`org.springframework.boot.jta.atomikos.AtomikosDataSourceBean`必須將 的參考變更為 `com.atomikos.spring.AtomikosDataSourceBean`。

PostgreSQL 方言移除

自訂類別PostgreSQLDialect.java已移除。也必須移除主要啟動器中的參考。

部署 (AWS Blu Age Runtime （非受管））

Tomcat

此版本與 Tomcat 相容10.1.17。需要將 Tomcat 伺服器升級至此版本，才能執行 Blu Age Runtime 4.0.0。請務必移植舊的組態變更（特別是 Catalina 屬性）。

共用相依性

執行時間共用資料夾包含up-to-date相依性。

額外相依性

如果您使用額外的相依性（不包含在執行時間中），您可能需要更新它們。額外資料夾中的讀我檔案會列出支援的版本。

AWS 藍卡存留期生命週期

本節定義 AWS Blu Age Runtime 主要版本的生命週期結束 (EOL) 日期。這可讓您規劃版本升級，以便隨時掌握最新的維護和功能。若要升級您的版本，請參閱 the section called “升級 AWS 藍髮年齡”。

建議您每 3 個月檢查新版本，並經常升級至最新版本。對於每次升級，您應該在生產或關鍵部署之前，對現代化應用程式執行非迴歸測試。

> **Note**
>
> 生命週期結束日期可能會因重大安全補救措施而變更。如需詳細資訊，請參閱元件生命週期。

AWS 藍卡年齡執行期生命週期結束 (EOL)

下表摘要說明每個主要版本的 EOL 日期。

主要版本	生命週期結束日期
第 3 版	2024 年 7 月 8 日
第 4 版	但要發佈

> **Note**
>
> 主要版本 4 的 EOL 日期將與下一個主要版本的可用性保持一致。

若要了解次要版本的支援模型，請參閱 元件生命週期。

AWS Blu Age 執行期概念

了解 AWS Blu Age Runtime 的基本概念可協助您了解應用程式如何透過自動重構進行現代化。

主題

- AWS Blu Age Runtime 高階架構
- AWS 現代化應用程式的 Blu Age 結構
- 什麼是 AWS 藍年資料簡化器
- AWS 藍本年齡藍調
- 公用程式 Web 應用程式中的可用程式
- AWS Blu Age Blusam 管理主控台

AWS Blu Age Runtime 高階架構

作為將舊版程式現代化為 Java 的 AWS Blu Age 解決方案的一部分， AWS Blu Age Runtime 透過提供舊版建構和程式程式碼組織的標準化程式庫，為現代化應用程式提供統一的 REST 型進入點，以及此類應用程式的執行架構。

這類現代化應用程式是將大型主機和中階程式（在下列文件中稱為「舊版」）現代化為 Web 架構的 AWS Blu Age Automated Refactor 程序的結果。

AWS Blu Age 執行期目標為複製舊版程式行為（異功能）、效能（與程式執行時間和資源耗用相關），以及 Java 開發人員維護現代化程式的便利性，不過使用如 tomcat、Spring、getters/setters、Fluent APIs 等熟悉的環境和慣用語。

主題

- AWS Blu Age 執行期元件
- 執行環境
- 無狀態和工作階段處理

- 高可用性和無狀態

AWS Blu Age 執行期元件

AWS Blu Age Runtime 環境由兩種元件組成：

- 一組 Java 程式庫 (jar 檔案） 通常參考為「共用資料夾」，並提供舊版建構和陳述式。
- 一組 Web 應用程式 (war 檔案），其中包含 Spring 型 Web 應用程式，為現代化程式提供一組常見的架構和服務。

以下各節詳細說明這兩個元件的角色。

AWS Blu Age 程式庫

AWS Blu Age 程式庫是一組存放在標準 tomcat classpath `shared/`子資料夾中的 jar 檔案，以便讓所有現代化 Java 程式都能使用。他們的目標是提供 Java 程式設計環境中原生且不易使用的功能，但傳統開發環境的典型功能。這些功能會盡可能以熟悉的方式公開給 Java 開發人員 (getters/setters、類別型、流暢APIs)。資料簡化程式程式程式庫是重要的範例，它為 Java 程式提供舊版記憶體配置和操作建構 （以 COBOL、PL1 或 RPG 語言出現）。這些 jar 是從舊版程式產生的現代化 Java 程式碼的核心相依性。如需 Data Simplifier 的詳細資訊，請參閱 什麼是 AWS 藍年資料簡化器。

Web 應用程式

Web Application Archives (WARs) 是將程式碼和應用程式部署到 tomcat 應用程式伺服器的標準方式。作為 AWS Blu Age 執行期的一部分提供的執行架構旨在提供一組重現舊版環境和交易監控器 (JCL 批次、CICS、IMS...) 和相關必要服務的執行架構。

最重要的是 `gapwalk-application`（通常簡稱為「Gapwalk」），它提供一組統一的 REST 型進入點來觸發和控制交易、程式和批次執行。如需詳細資訊，請參閱AWS Blu Age 執行時間 APIs。

此 Web 應用程式會配置 Java 執行緒和資源，以在其設計內容中執行現代化程式。以下章節會詳細說明這類重現環境的範例。

其他 Web 應用程式會新增至執行環境 （更精確地說，是以下所述的「程式登錄檔」) 程式，模擬舊版程式可用的程式，以及可從中呼叫的程式。其中兩個重要的類別包括：

- 模擬作業系統提供的程式：JCL 驅動的批次尤其預期能夠呼叫各種檔案和資料庫處理程式，作為其標準環境的一部分。範例包括 `SORT`/`DFSORT` 或 `IDCAMS`。為此目的，Java 程式提供重現這類行為的程式，並且可以使用與舊版相同的慣例進行呼叫。

- 「驅動程式」，這是執行架構或中介軟體提供的特殊程式，做為進入點。範例為 CBLTDLI，在 IMS 環境中執行的 COBOL 程式依賴於存取 IMS 相關服務 (IMS 資料庫、透過MFS 的使用者對話方塊等）。

程式登錄檔

為了參與和利用這些建構、架構和服務，從舊版的 Java 程式現代化遵循 中記錄的特定結構AWS 現代化應用程式的 Blu Age 結構。啟動時， AWS Blu Age Runtime 會在常見的「程式登錄檔」中收集所有這類程式，以便之後可以叫用（和呼叫彼此）。程式登錄檔提供鬆散耦合和分解的可能性（因為彼此呼叫的程式不必同時進行現代化）。

執行環境

提供經常遇到的舊版環境和編排：

- JCL 驅動的批次，一旦現代化為 Java 程式和 Groovy 指令碼，就可以同步（封鎖）或非同步（刪除）方式啟動。在後一種情況下，可以透過 REST 端點監控其執行。
- AWS Blu Age 子系統透過下列方式提供類似 CICS 的執行環境：
  - 一個進入點，用於啟動 CICS 交易並執行相關聯的程式，同時遵守 CICS「執行層級」編排，
  - 資源定義的外部儲存、
  - 一組同質的 Java 流利 APIs可重新產生EXEC CICS陳述式，
  - 一組可重新產生 CICS 服務的可插拔類別，例如暫存儲存佇列、暫存資料佇列或檔案存取（通常提供多個實作，例如 Amazon Managed Service for Apache Flink、Amazon Simple Queue Service 或 RabbitMQ for TD Queues)。
  - 對於面向使用者的應用程式，BMS 螢幕描述格式會現代化為「角度 Web 應用程式」，並支援對應的「虛擬對等」對話方塊。
- 同樣地，另一個子系統提供以 IMS 訊息為基礎的編排，並支援以 Citrix 格式現代化 UI 畫面。
- 此外，第三個子系統允許在類似 iSeries 的環境中執行程式，包括將 DSPF（顯示檔案）指定的畫面現代化。

所有這些環境都以常見的作業系統層級服務為基礎，例如：

- 模擬舊版記憶體配置和配置 (資料簡化程式)，
- Java 執行緒式 COBOL "run units" 執行和參數傳遞機制的重製 (CALL 陳述式），
- 模擬平面、串連、VSAM（透過 Blusam 程式庫集）和 GDG 資料集組織，

- 存取資料存放區，例如 RDBMS (EXEC SQL 陳述式）。

無狀態和工作階段處理

AWS Blu Age Runtime 的一項重要功能是在執行現代化程式時啟用高可用性 (HA) 和水平可擴展性案例。

其基石是無狀態，其為 HTTP 工作階段處理的重要範例。

工作階段處理

Tomcat 以 Web 為基礎，這是 HTTP 工作階段處理（如 tomcat 和 Spring 所提供）和無狀態設計的重要機制。因此，無狀態設計是以下列項目為基礎：

- 使用者透過 HTTPS 連線，
- 應用程式伺服器會部署在負載平衡器後方，
- 當使用者第一次連線到應用程式時，將會進行身分驗證，而應用程式伺服器將會建立識別符（通常是在 Cookie 內）
- 此識別符將用作金鑰，以儲存和擷取使用者內容至外部快取（資料存放區）或從中擷取內容。

Cookie 管理由 AWS Blu Age 架構和基礎 tomcat 伺服器自動完成，這對使用者來說是透明的。使用者網際網路瀏覽器會自動管理此項目。

Gapwalk Web 應用程式可能會在各種資料存放區中存放工作階段狀態（內容）：

- Amazon ElastiCache (Redis OSS)
- Redis 叢集
- 記憶體映射中的（僅適用於開發和獨立環境，不適用於 HA)。

高可用性和無狀態

更普遍地說， AWS Blu Age 架構的設計宗旨是無狀態：大多數需要非暫時性狀態才能重現舊版程式行為，不會存放在應用程式伺服器中，而是透過外部常見的「單一事實來源」共用。

這類狀態的範例包括 CICS 的暫時儲存佇列或資源定義，而這些一般外部儲存是與 Redis 相容的伺服器或關聯式資料庫。

此設計結合負載平衡和共用工作階段，會導致大多數使用者面向的對話方塊 (OLTP、「線上交易處理」) 可在多個「節點」（此處為 tomcat 執行個體）之間分佈。

事實上，使用者可以在任何伺服器上執行交易，而不在意下一個交易呼叫是否在不同伺服器上執行。然後，當產生新的伺服器時（由於自動擴展或取代運作狀態不佳的伺服器），我們可以保證任何可連線且運作狀態良好的伺服器都能以適當的結果（預期的傳回值、資料庫中預期的資料變更等）執行交易。

AWS 現代化應用程式的 Blu Age 結構

本文件提供有關現代化應用程式結構的詳細資訊（使用 AWS 大型主機現代化重構工具），以便開發人員可以完成各種任務，例如：

- 順利導覽至應用程式。
- 開發可從現代化應用程式呼叫的自訂程式。
- 安全地重構現代化應用程式。

我們假設您已具備下列基本知識：

- 舊版常見編碼概念，例如記錄、資料集及其存取模式，以記錄 -- 編製索引、循序 --、VSAM、執行單位、jcl 指令碼、CICS 概念等。
- 使用 Spring 架構進行 Java 編碼。
- 在整個文件中，我們使用 `short class names`來提供可讀性。如需詳細資訊，請參閱AWS Blu Age 完整名稱映射擷取 AWS Blu Age 執行時間元素的對應完整名稱第三方完整名稱映射，以及擷取第三方元素的對應完整名稱。
- 所有成品和範例都取自範例 COBOL/CICS CardDemo 應用程式的現代化程序輸出。

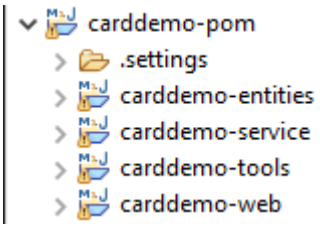
主題

- 成品組織
- 執行和呼叫程式
- 撰寫您自己的程式
- 完整名稱映射

成品組織

AWS Blu Age 現代化應用程式封裝為 Java Web 應用程式 (.war)，您可以在 JEE 伺服器上部署。通常，伺服器是嵌入 AWS Blu Age Runtime 的 Tomcat 執行個體，目前建置在 Springboot 和 Angular（適用於 UI 部分）架構上。

戰會彙總數個元件成品 (.jar)。每個 jar 都是專用 Java 專案的編譯（使用 maven 工具）的結果，其元素是現代化程序的結果。



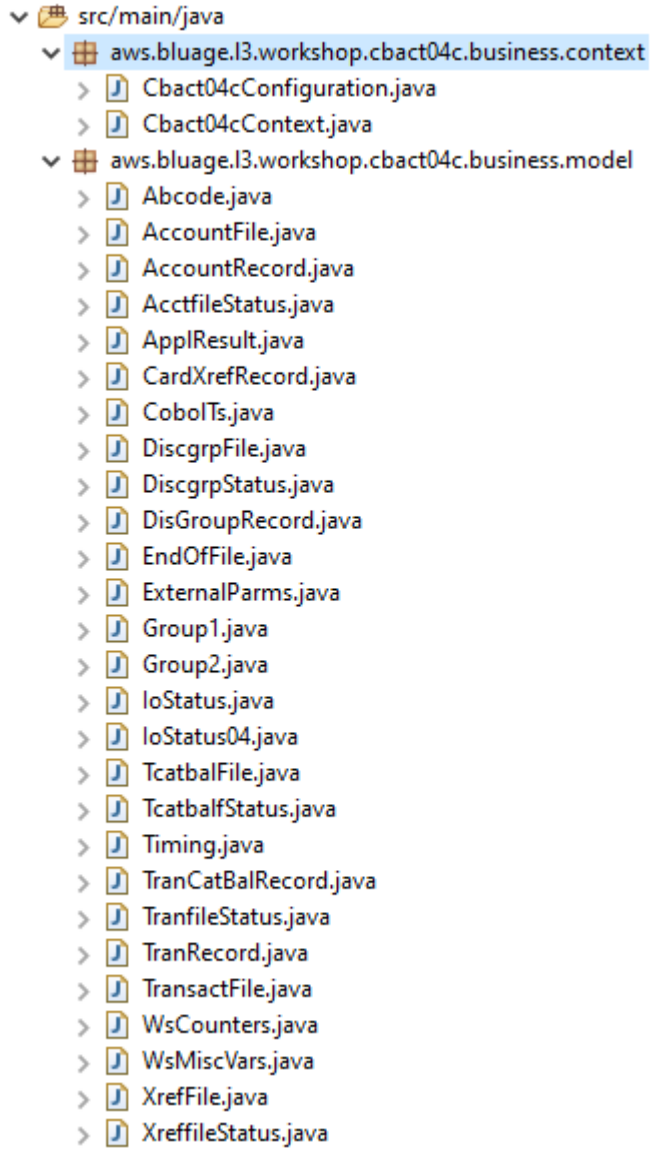
基本組織依賴下列結構：

- 實體專案： 包含商業模型和內容元素。專案名稱通常以「-entities」結尾。一般而言，對於指定的舊版 COBOL 程式，這對應於 I/O 區段（資料集）和資料分割的現代化。您可以有一個以上的實體專案。
- 服務專案： 包含傳統商業邏輯現代化元素。通常，COBOL 程式的程序劃分。您可以有一個以上的服務專案。
- 公用程式專案： 包含其他專案使用的共用常見工具和公用程式。
- Web 專案： 包含適用的 UI 相關元素現代化。不適用於僅限批次的現代化專案。這些 UI 元素可能來自 CICSBMS 映射、IMSMFS 元件和其他大型主機 UI 來源。您可以有一個以上的 Web 專案。

實體專案內容

> **Note**
>
> 下列描述僅適用於 COBOL 和 PL/I 現代化輸出。RPG 現代化輸出是以不同的配置為基礎。

在任何重構之前，實體專案中的套件組織會與現代化程式繫結。您可以用幾種不同的方式來完成此操作。偏好的方式是使用重構工具盒，在您觸發程式碼產生機制之前，該工具盒會運作。這是進階操作，在 BluAge 訓練中會加以說明。如需詳細資訊，請參閱重構研討會。此方法可讓您保留稍後重新產生 Java 程式碼的功能，以受益於未來的進一步改善，例如）。另一種方式是直接在產生的原始程式碼上，使用您可能想要套用的任何 Java 重構方法，執行定期 Java 重構，風險由您自負。



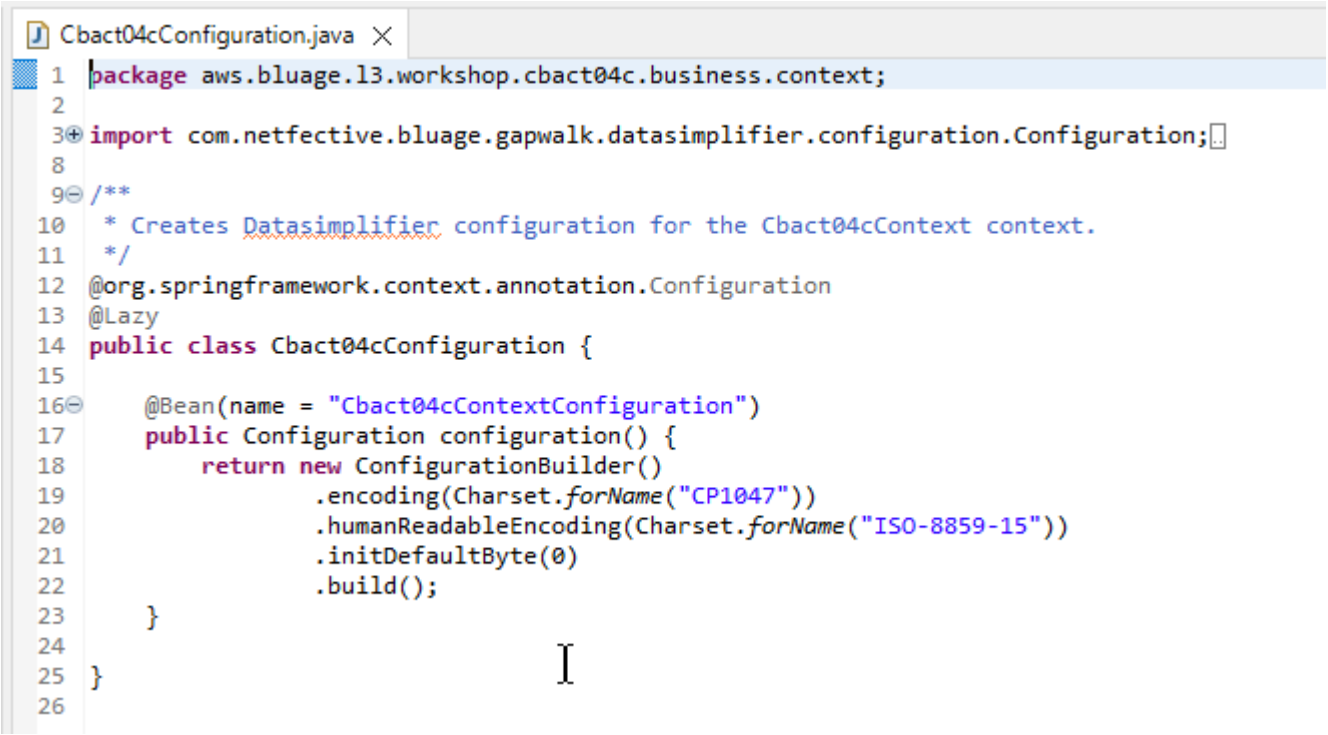
程式相關類別

每個現代化計畫都與兩個套件相關，一個是 business.context，另一個是 business.model 套件。

- *base package*.*program*.business.context

  business.context 子套件包含兩個類別，一個組態類別和一個內容類別。

  - 程式的一個組態類別，其中包含指定程式的特定組態詳細資訊，例如用來表示字元型資料元素的字元集、填補資料結構元素的預設位元組值等。類別名稱以「組態」結尾。它以`@org.springframework.context.annotation.Configuration`註釋標記，並包含必須傳回正確設定`Configuration`物件的單一方法。



Cbact04cConfiguration.java

```java
package aws.bluage.l3.workshop.cbact04c.business.context;

import com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration;

/**
 * Creates Datasimplifier configuration for the Cbact04cContext context.
 */
@org.springframework.context.annotation.Configuration
@Lazy
public class Cbact04cConfiguration {

    @Bean(name = "Cbact04cContextConfiguration")
    public Configuration configuration() {
        return new ConfigurationBuilder()
                .encoding(Charset.forName("CP1047"))
                .humanReadableEncoding(Charset.forName("ISO-8859-15"))
                .initDefaultByte(0)
                .build();
    }

}

```

- 一種內容類別，做為程式服務類別（請參閱下文）與模型子套件中的資料結構 (Record) 和資料集 (File) 之間的橋樑（請參閱下文）。類別名稱以「內容」結尾，是 RuntimeContext 類別的子類別。

Cbact04cContext.java

```java
@Component("aws.bluage.l3.workshop.cbact04c.business.context.Cbact04cContext")
@Import({
    aws.bluage.l3.workshop.cbact04c.business.model.TcatbalFile.class
,   aws.bluage.l3.workshop.cbact04c.business.model.XrefFile.class
,   aws.bluage.l3.workshop.cbact04c.business.model.DiscgrpFile.class
,   aws.bluage.l3.workshop.cbact04c.business.model.AccountFile.class
,   aws.bluage.l3.workshop.cbact04c.business.model.TransactFile.class
})
@Lazy
@Scope("prototype")
public class Cbact04cContext extends JicsRuntimeContext {

    @Autowired
    private TcatbalFile tcatbalFile;

    @Autowired
    private XrefFile xrefFile;

    @Autowired
    private DiscgrpFile discgrpFile;

    @Autowired
    private AccountFile accountFile;

    @Autowired
    private TransactFile transactFile;

    private IndexedFile tcatbalFileFile;

    private IndexedFile xrefFileFile;

    private IndexedFile discgrpFileFile;

    private IndexedFile accountFileFile;

    private SequentialFile transactFileFile;

    private TranCatBalRecord tranCatBalRecord;
    private TcatbalfStatus tcatbalfStatus;
    private CardXrefRecord cardXrefRecord;
```

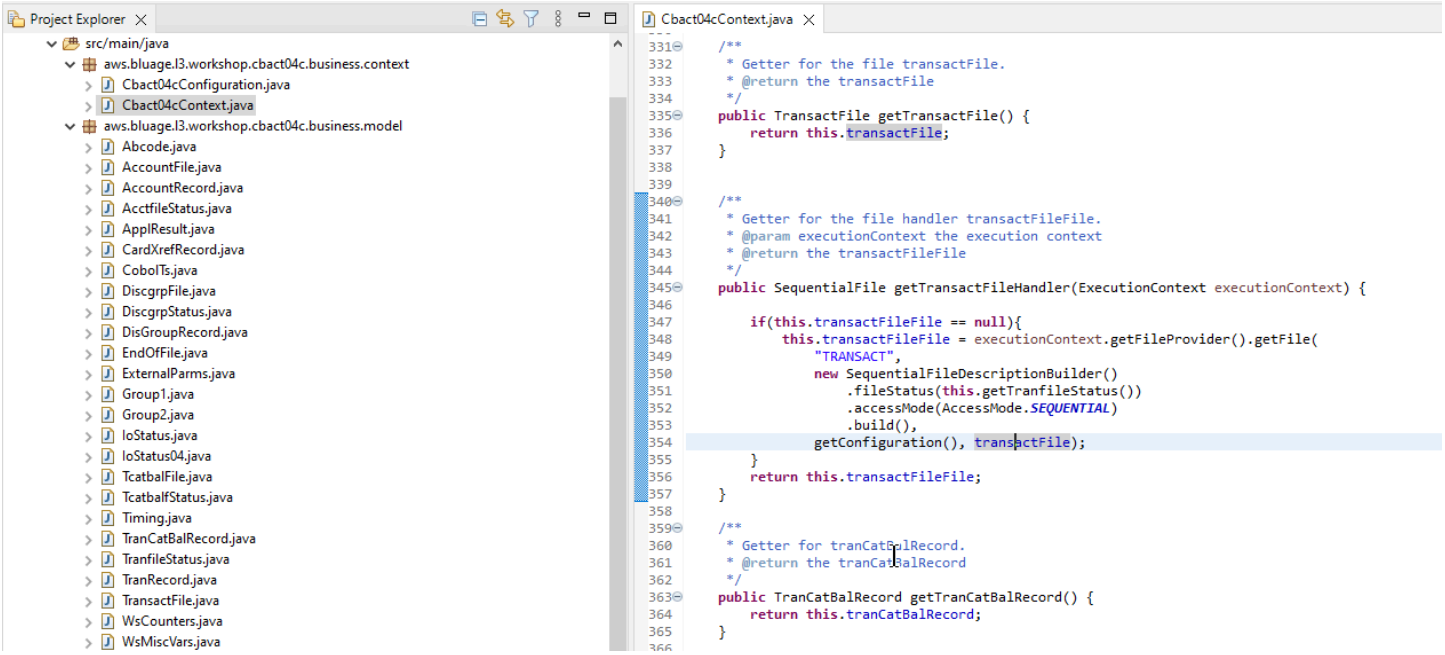
- *base package*.*program*.business.model

  模型子套件包含指定程式可以使用的所有資料結構。例如，任何 01 層級 COBOL 資料結構對應至模型子套件中的類別（較低層級的資料結構是其擁有 01 層級結構的屬性）。如需如何現代化 01 資料結構的詳細資訊，請參閱什麼是 AWS 藍年資料簡化器。

DiscgrpFile.java

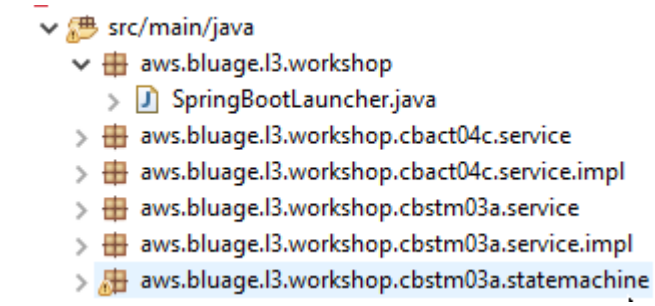
```java
package aws.bluage.l3.workshop.cbact04c.business.model;

import com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration;
import com.netfective.bluage.gapwalk.datasimplifier.data.structure.Elementary;
import com.netfective.bluage.gapwalk.datasimplifier.data.structure.Group;
import com.netfective.bluage.gapwalk.datasimplifier.entity.ElementaryRangeReference;
import com.netfective.bluage.gapwalk.datasimplifier.entity.RangeReference;
import com.netfective.bluage.gapwalk.datasimplifier.entity.RecordEntity;
import com.netfective.bluage.gapwalk.datasimplifier.metadata.type.AlphanumericType;
import com.netfective.bluage.gapwalk.datasimplifier.metadata.type.ZonedType;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Lazy;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

/**
 * Data simplifier file DiscgrpFile.
 *
 * <p>About 'fdDiscgrpRec' field, <br>uml entity: aws.bluage.l3.workshop.cbact04c.business.model.FdDiscgrpRec
 * <br></p>
 *
 */
@Component("aws.bluage.l3.workshop.cbact04c.business.model.DiscgrpFile")
@Lazy
@Scope("prototype")
public class DiscgrpFile extends RecordEntity {

    private final Group root = new Group(getData());
    private final Group fdDiscgrpRec = new Group(root);
    private final Group fdDiscgrpKey = new Group(fdDiscgrpRec);
    private final Elementary fdDisAcctGroupId = new Elementary(fdDiscgrpKey,new AlphanumericType(10));
    private final Elementary fdDisTranTypeCd = new Elementary(fdDiscgrpKey,new AlphanumericType(2));
    private final Elementary fdDisTranCatCd = new Elementary(fdDiscgrpKey,new ZonedType(4, 0, false));
    private final Elementary fdDiscgrpData = new Elementary(fdDiscgrpRec,new AlphanumericType(34));
```

所有類別都會擴展 RecordEntity類別，代表對業務記錄表示法的存取。有些記錄具有特殊用途，因為它們繫結至 File。在建立檔案物件時， Record和 之間的繫結File是在內容類別中找到的對應 *FileHandler 方法中進行。例如，以下清單顯示 TransactfileFile File 如何繫結至 transactFile Record（從模型子套件）。



服務專案內容

每個服務專案都隨附專用的 Springboot 應用程式，做為架構的骨幹。這是透過名為 的類別具體化SpringBootLauncher，位於服務 Java 來源的基本套件中：



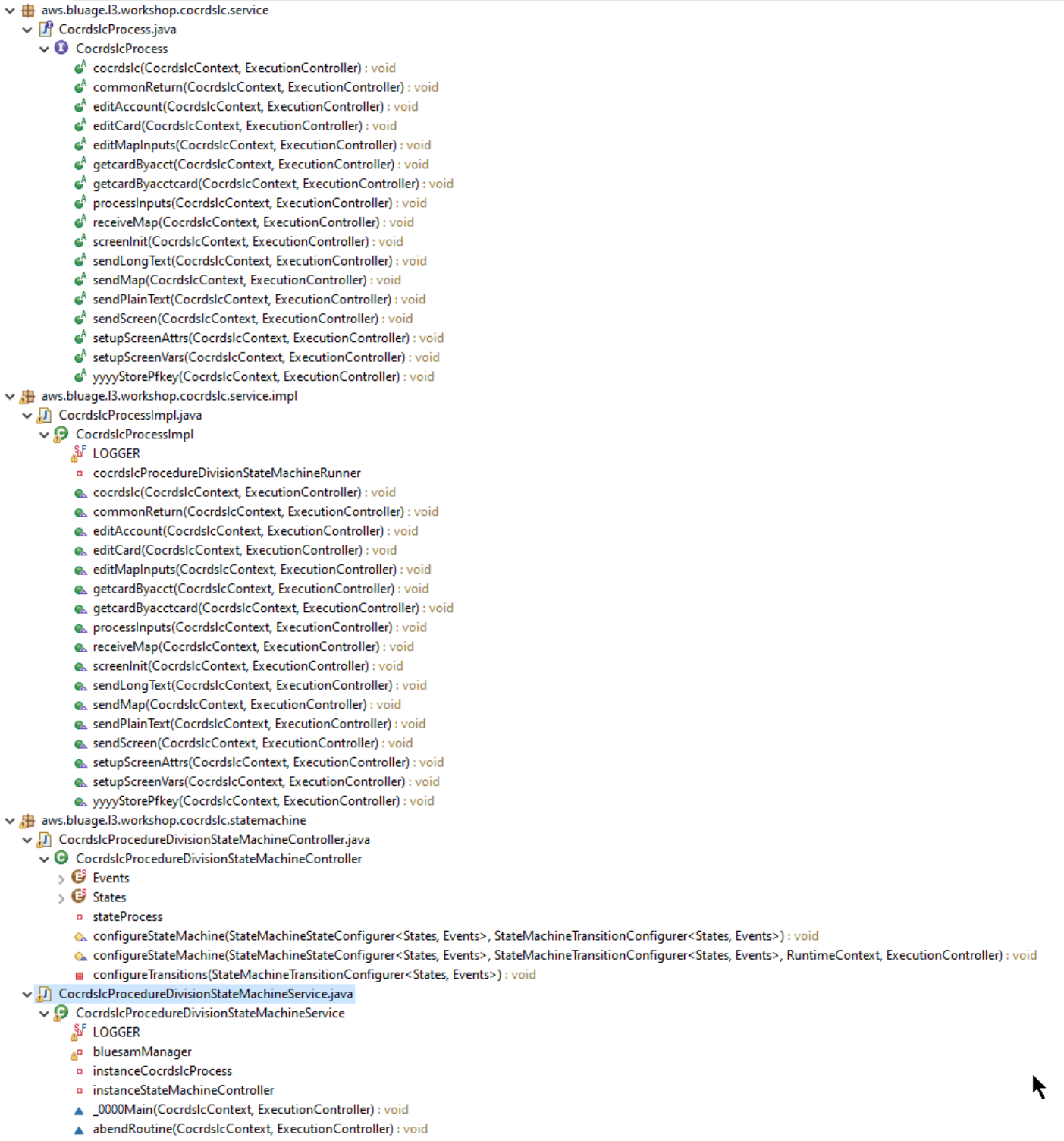
此類別明顯負責：

- 在程式類別與受管資源 （資料來源/交易管理員/資料集映射/等） 之間建立黏性。
- ConfigurableApplicationContext 將 提供給程式。
- 探索標示為彈簧元件 () 的所有類別@Component。
- 確保程式在 中正確註冊 ProgramRegistry -- 請參閱負責此註冊的初始化方法。

```
/**
 * Initialization method called when the spring application is ready.
 * Register all programs and services to the gapwalk shared context.
 * @param event the application ready event
 */
@EventListener
public void initialize(ApplicationReadyEvent event) {
    Map<String, ProgramContainer> programContainers = event.getApplicationContext().getBeansOfType(ProgramContainer.class);
    programContainers.values().forEach(ProgramRegistry::registerProgram);
    Map<String, ServiceContainer> serviceContainers = event.getApplicationContext().getBeansOfType(ServiceContainer.class);
    serviceContainers.values().forEach(ServiceRegistry::registerService);
}
```

程式相關成品

若未事先重構，商業邏輯現代化輸出會依每個舊版程式的兩個或三個套件進行組織：



最詳盡的案例將包含三個套件：

- *base package*.*program*.service： 包含名為 Program Process 的界面，其具有處理商業邏輯的業務方法，可保留舊版執行控制流程。

- *base package*.*program*.service.impl：包含名為 ProgramProcessImpl ProcessImpl 的類別，這是先前描述的程序界面實作。這是舊版陳述式「轉譯」為 Java 陳述式的地方，依賴 AWS Blu Age 架構：

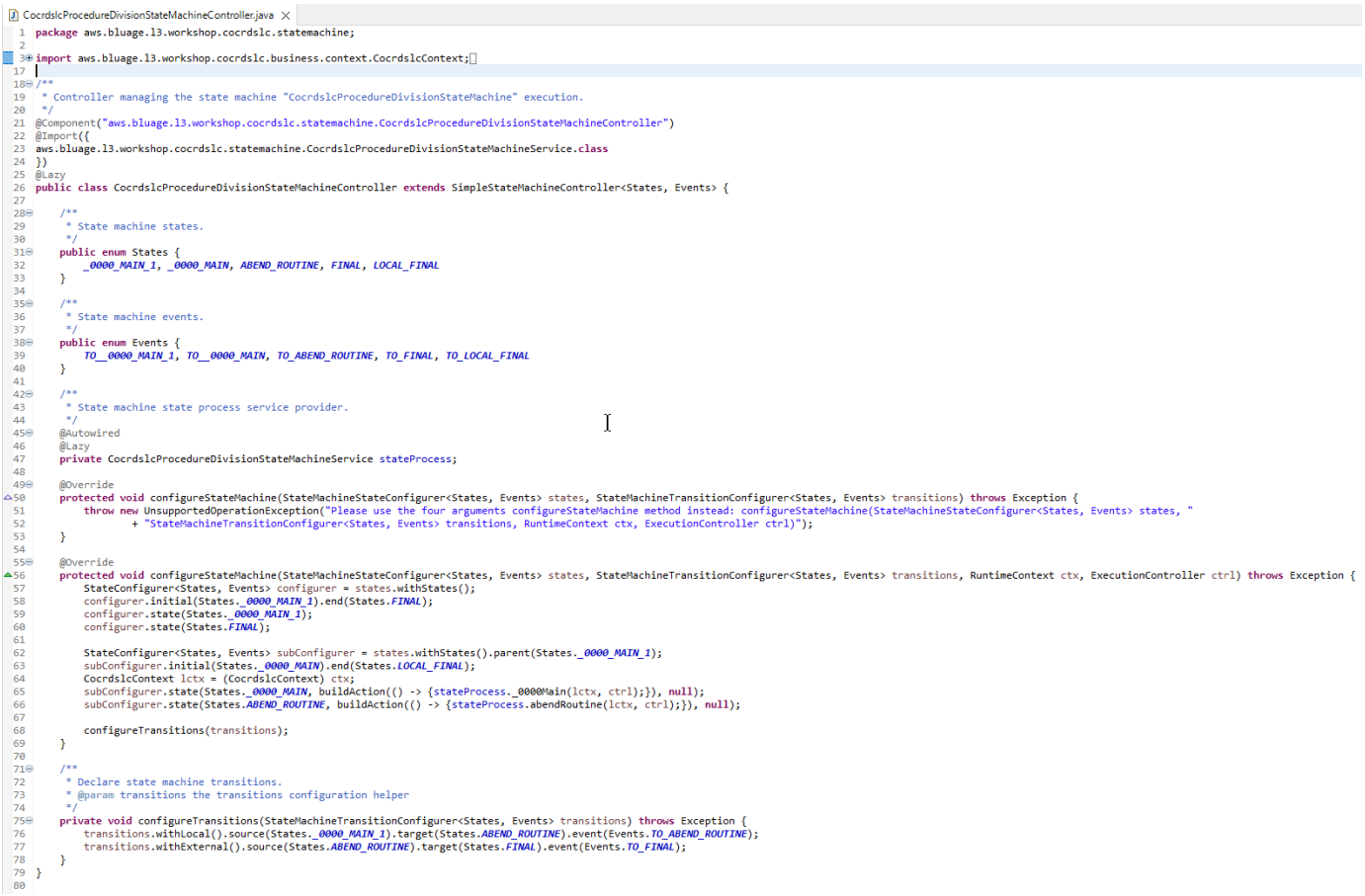
CocrdslcProcessImpl.java

```
210     /**
211      * Process operation sendScreen.
212      *
213      * @param ctx
214      * @param ctrl
215      */
216     @Override
217     public void sendScreen(final CocrdslcContext ctx, final ExecutionController ctrl) {
218         ctx.getCcWorkAreas().setCcardNextMapset(ctx.getWsLiterals().getLitThismapset());
219         ctx.getCcWorkAreas().setCcardNextMap(ctx.getWsLiterals().getLitThismap());
220         ctx.getCarddemoCommarea().setCdemoPgmReenter(true);
221         SendMapBuilder.newInstance(ctx.getDfheiblk(), ctx)
222         .withMap(ctx.getCcWorkAreas().getCcardNextMap())
223         .withMapset(ctx.getCcWorkAreas().getCcardNextMapset())
224         .withData(ctx.getGroup1().getCcrdslaoReference())
225         .withCursor()
226         .withErase()
227         .withFreeKB()
228         .execute();
229         ctx.getWsMiscStorage().setWsRespCd(ctx.getDfheiblk().getEibresp());
230     }
231
232     /**
233      * Process operation processInputs.
234      *
235      * @param ctx
236      * @param ctrl
237      */
238     @Override
239     public void processInputs(final CocrdslcContext ctx, final ExecutionController ctrl) {
240         receiveMap(ctx, ctrl);
241         editMapInputs(ctx, ctrl);
242         ctx.getCcWorkAreas().setCcardErrorMsg(ctx.getWsMiscStorage().getWsReturnMsg());
243         ctx.getCcWorkAreas().setCcardNextProg(ctx.getWsLiterals().getLitThispgm());
244         ctx.getCcWorkAreas().setCcardNextMapset(ctx.getWsLiterals().getLitThismapset());
245         ctx.getCcWorkAreas().setCcardNextMap(ctx.getWsLiterals().getLitThismap());
246     }
247
```

- *base package*.*program*.statemachine：此套件可能不一定總是存在。當傳統控制流程的現代化必須使用狀態機器方法（即使用 Spring StateMachine 架構) 來正確涵蓋傳統執行流程時，這是必要的。

  在這種情況下，狀態機器子套件包含兩個類別：

  - *Program*ProcedureDivisionStateMachineController：擴展實作 StateMachineController（定義控制狀態機器執行所需的操作）和 StateMachineRunner（定義執行狀態機器所需的操作）介面的類別，用於驅動 Spring 狀態機器機制；例如，SimpleStateMachineController如範例案例所示。



```java
package aws.bluage.l3.workshop.cocrdslc.statemachine;

import aws.bluage.l3.workshop.cocrdslc.business.context.CocrdslcContext;

/**
 * Controller managing the state machine "CocrdslcProcedureDivisionStateMachine" execution.
 */
@Component("aws.bluage.l3.workshop.cocrdslc.statemachine.CocrdslcProcedureDivisionStateMachineController")
@Import({
aws.bluage.l3.workshop.cocrdslc.statemachine.CocrdslcProcedureDivisionStateMachineService.class
})
@Lazy
public class CocrdslcProcedureDivisionStateMachineController extends SimpleStateMachineController<States, Events> {

    /**
     * State machine states.
     */
    public enum States {
        _0000_MAIN_1, _0000_MAIN, ABEND_ROUTINE, FINAL, LOCAL_FINAL
    }

    /**
     * State machine events.
     */
    public enum Events {
        TO__0000_MAIN_1, TO__0000_MAIN, TO_ABEND_ROUTINE, TO_FINAL, TO_LOCAL_FINAL
    }

    /**
     * State machine state process service provider.
     */
    @Autowired
    @Lazy
    private CocrdslcProcedureDivisionStateMachineService stateProcess;

    @Override
    protected void configureStateMachine(StateMachineStateConfigurer<States, Events> states, StateMachineTransitionConfigurer<States, Events> transitions) throws Exception {
        throw new UnsupportedOperationException("Please use the four arguments configureStateMachine method instead: configureStateMachine(StateMachineStateConfigurer<States, Events> states, "
                + "StateMachineTransitionConfigurer<States, Events> transitions, RuntimeContext ctx, ExecutionController ctrl)");
    }

    @Override
    protected void configureStateMachine(StateMachineStateConfigurer<States, Events> states, StateMachineTransitionConfigurer<States, Events> transitions, RuntimeContext ctx, ExecutionController ctrl) throws Exception {
        StateConfigurer<States, Events> configurer = states.withStates();
        configurer.initial(States._0000_MAIN_1).end(States.FINAL);
        configurer.state(States._0000_MAIN_1);
        configurer.state(States.FINAL);

        StateConfigurer<States, Events> subConfigurer = states.withStates().parent(States._0000_MAIN_1);
        subConfigurer.initial(States._0000_MAIN).end(States.LOCAL_FINAL);
        CocrdslcContext lctx = (CocrdslcContext) ctx;
        subConfigurer.state(States._0000_MAIN, buildAction(() -> {stateProcess._0000Main(lctx, ctrl);}), null);
        subConfigurer.state(States.ABEND_ROUTINE, buildAction(() -> {stateProcess.abendRoutine(lctx, ctrl);}), null);

        configureTransitions(transitions);
    }

    /**
     * Declare state machine transitions.
     * @param transitions the transitions configuration helper
     */
    private void configureTransitions(StateMachineTransitionConfigurer<States, Events> transitions) throws Exception {
        transitions.withLocal().source(States._0000_MAIN_1).target(States.ABEND_ROUTINE).event(Events.TO_ABEND_ROUTINE);
        transitions.withExternal().source(States.ABEND_ROUTINE).target(States.FINAL).event(Events.TO_FINAL);
    }
}

```

狀態機器控制器會定義可能的可能不同狀態及其之間的轉換，以重現指定程式的舊版執行控制流程。

建置狀態機器時，控制器是指在狀態機器套件中相關聯的服務類別中定義的方法，如下所述：

```
subConfigurer.state(States._0000_MAIN, buildAction(() ->
 {stateProcess._0000Main(lctx, ctrl);}), null);
subConfigurer.state(States.ABEND_ROUTINE, buildAction(() ->
 {stateProcess.abendRoutine(lctx, ctrl);}), null);
```

- *Program*ProcedureDivisionStateMachineService：此服務類別代表一些業務邏輯，這些邏輯必須與狀態機器控制器建立的狀態機器繫結，如前所述。

  此類別方法中的程式碼使用狀態機器控制器中定義的事件：

CocrdslcProcedureDivisionStateMachineService.java

```
59      /**
60       * State process operation _0000Main.
61       *
62       * @param ctx
63       * @param ctrl
64       */
65      void _0000Main(CocrdslcContext ctx, ExecutionController ctrl) {
66          ctx.getDfheiblk().bind(ArgUtils.get(ctx, 0));
67          ctx.getDfhcommarea().bind(ArgUtils.get(ctx, 1));
68
69          /*
70          ****************************************************************
71          Program:     COCRDSLC.CBL                                     *
72          Layer:       Business logic                                   *
73          Function:    Accept and process credit card detail request    *
74          ****************************************************************
75          Copyright Amazon.com, Inc. or its affiliates.
76          All Rights Reserved.
77          Licensed under the Apache License, Version 2.0 (the \"License\").
78          You may not use this file except in compliance with the License.
79          You may obtain a copy of the License at
80          http://www.apache.org/licenses/LICENSE-2.0
81          Unless required by applicable law or agreed to in writing,
82          software distributed under the License is distributed on an
83          \"AS IS\" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
84          either express or implied. See the License for the specific
85          language governing permissions and limitations under the License
86          ****************************************************************
87          Ver: CardDemo_v1.0-15-g27d6c6f-68 Date: 2022-07-19 23:16:00 CDT */
88          instanceStateMachineController.registerSignalHandler(Events.TO_ABEND_ROUTINE, "!ABEND");
89          HandleAbendBuilder.newInstance(ctx.getDfheiblk(), ctx).execute().handleException();
90          ctx.getCcWorkAreas().getCcWorkAreaReference().getField().initialize();
91          ctx.getWsMiscStorage().getField().initialize();
92          DataUtils.initialize(ctx.getWsCommarea().getWsCommareaReference());
93
```

CocrdslcProcedureDivisionStateMachineService.java

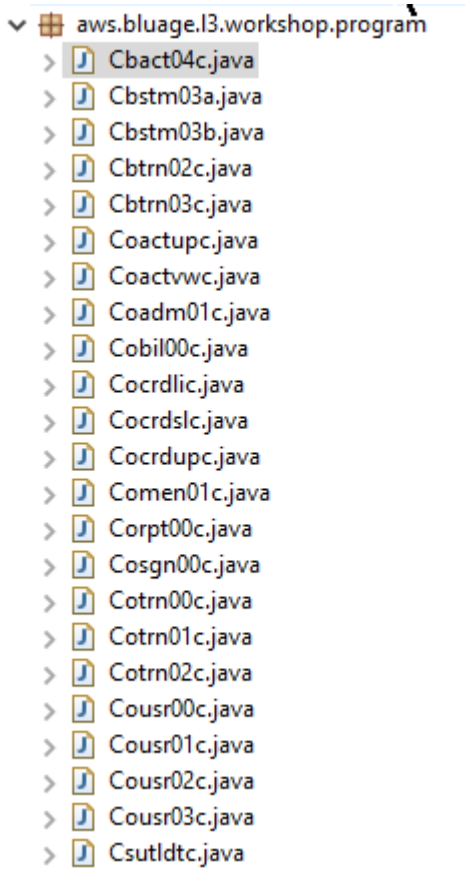
```
221     *
222     * @param ctx
223     * @param ctrl
224     */
225    void abendRoutine(CocrdslcContext ctx, ExecutionController ctrl) {
226        if (DataUtils.isLowValue(ctx.getAbendData().getAbendMsgReference())) {
227            ctx.getAbendData().setAbendMsg("UNEXPECTED ABEND OCCURRED.");
228        }
229        ctx.getAbendData().setAbendCulprit(ctx.getWsLiterals().getLitThispgm());
230        SendTextBuilder.newInstance(ctx.getDfheiblk(), ctx)
231        .withData(ctx.getAbendData())
232        .withLength(134)
233        .execute();
234        HandleAbendBuilder.newInstance(ctx.getDfheiblk(), ctx).cancel().execute().handleException();
235        AbendBuilder.newInstance(ctx.getDfheiblk(), ctx).withAbendCode("9999").execute().handleException();
236
237        /*
238        Ver: CardDemo_v1.0-15-g27d6c6f-68 Date: 2022-07-19 23:12:33 CDT */
239        instanceStateMachineController.sendEvent(Events.TO_FINAL);
240
241    }
242 }
243
```

狀態機器服務也會呼叫上述的程序服務實作：

CocrdslcProcedureDivisionStateMachineService.java

```
166
167             /*
168             ************************************************************
169             COMING FROM CREDIT CARD LIST SCREEN
170             SELECTION CRITERIA ALREADY VALIDATED
171             ************************************************************ */
172         } else if (ctx.getCarddemoCommarea().isCdemoPgmEnter() && DataUtils.compare(ctx.getCarddemoCommarea().getCdemoFromProgramReference(), ctx.getWsLiterals().getLitCclistpgmReference()) == 0) {
173             ctx.getWsMiscStorage().setInputOk(true);
174             ctx.getCcWorkAreas().setCcAcctIdN(ctx.getCarddemoCommarea().getCdemoAcctId());
175             ctx.getCcWorkAreas().setCcCardNumN(ctx.getCarddemoCommarea().getCdemoCardNum());
176             instanceCocrdslcProcess.getcardByacctcard(ctx, ctrl);
177             instanceCocrdslcProcess.sendMap(ctx, ctrl);
178             instanceCocrdslcProcess.commonReturn(ctx, ctrl);
179         } else if (ctx.getCarddemoCommarea().isCdemoPgmEnter()) {
180
181             /*
182             ************************************************************
183             COMING FROM SOME OTHER CONTEXT
184             SELECTION CRITERIA TO BE GATHERED
185             ************************************************************ */
186             instanceCocrdslcProcess.sendMap(ctx, ctrl);
187             instanceCocrdslcProcess.commonReturn(ctx, ctrl);
188         } else if (ctx.getCarddemoCommarea().isCdemoPgmReenter()) {
189             instanceCocrdslcProcess.processInputs(ctx, ctrl);
190             if (ctx.getWsMiscStorage().isInputError()) {
191                 instanceCocrdslcProcess.sendMap(ctx, ctrl);
192                 instanceCocrdslcProcess.commonReturn(ctx, ctrl);
193             } else {
194                 instanceCocrdslcProcess.getcardByacctcard(ctx, ctrl);
195                 instanceCocrdslcProcess.sendMap(ctx, ctrl);
196                 instanceCocrdslcProcess.commonReturn(ctx, ctrl);
197             }
198         } else {
199             ctx.getAbendData().setAbendCulprit(ctx.getWsLiterals().getLitThispgm());
200             ctx.getAbendData().setAbendCode("0001");
201             DataUtils.setToBlank(ctx.getAbendData().getAbendReasonReference());
202             ctx.getWsMiscStorage().setWsReturnMsg("UNEXPECTED DATA SCENARIO");
203             instanceCocrdslcProcess.sendPlainText(ctx, ctrl);
204         }
205
206         /*
207         If we had an error setup error message that slipped through
208         Display and return */
209         if (ctx.getWsMiscStorage().isInputError()) {
210             ctx.getCcWorkAreas().setCcardErrorMsg(ctx.getWsMiscStorage().getWsReturnMsg());
211             instanceCocrdslcProcess.sendMap(ctx, ctrl);
212             instanceCocrdslcProcess.commonReturn(ctx, ctrl);
213         }
214         instanceCocrdslcProcess.commonReturn(ctx, ctrl);
```

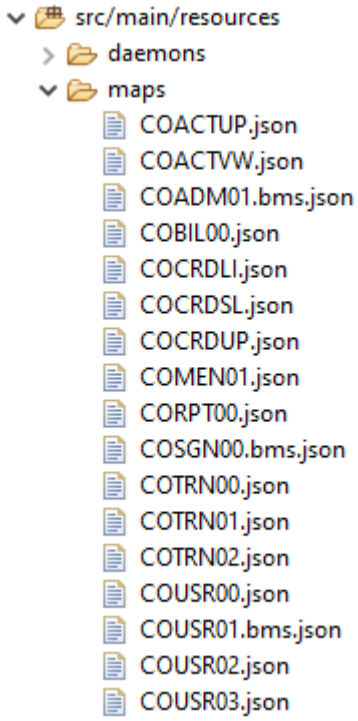
此外，名為 的套件*base package*.program扮演重要角色，因為每個程式會收集一個類別，做為程式進入點（稍後會更詳細地說明）。每個類別都會實作Program界面，即程式進入點的標記。



其他成品

- Adobe MAPs配套

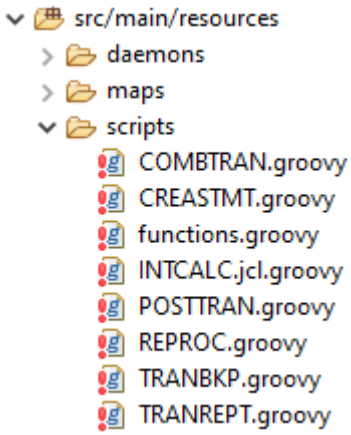
除了程式相關的成品之外，服務專案還可以包含其他成品以供各種用途使用。在 CICS 線上應用程式的現代化情況下，現代化程序會產生 json 檔案，並放入 /src/main/resources 資料夾的映射資料夾中：



Blu Age 執行時間會使用這些 json 檔案，將 SEND MAP 陳述式所使用的記錄與畫面欄位繫結在一起。

- Groovy 指令碼

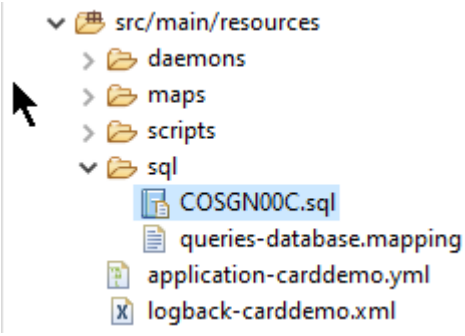
  如果舊版應用程式具有 JCL 指令碼，則這些指令碼已現代化為 groovy 指令碼，並存放在 /src/main/resources/scripts 資料夾 （稍後會更詳細地說明該特定位置）：



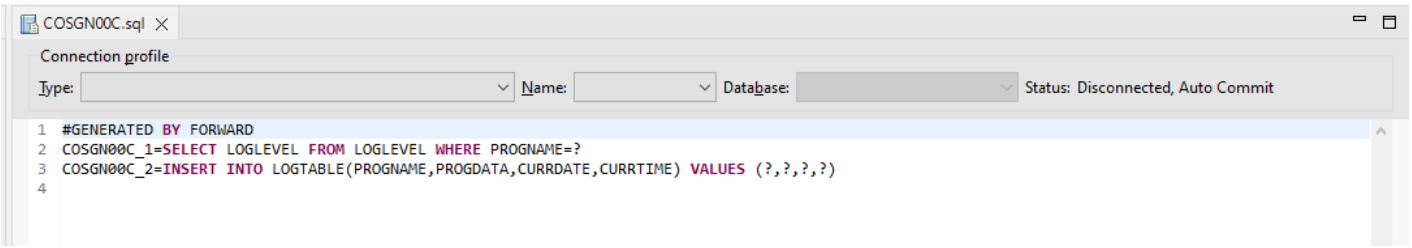
  這些指令碼用於啟動批次任務 （專用、非互動式、密集 cpu 的資料處理工作負載）。

- SQL 檔案

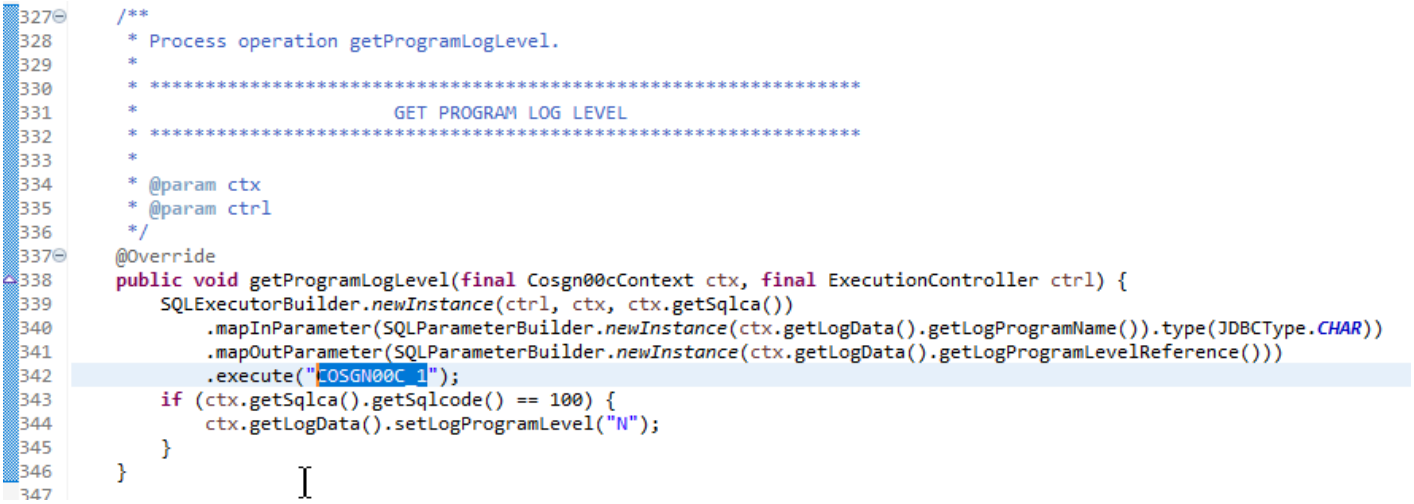
如果舊版應用程式使用 SQL 查詢，則會使用命名模式 program.sql，在專用屬性檔案中收集對應的現代化 SQL 查詢，其中程式是使用這些查詢的程式名稱。



這些 sql 檔案的內容是 (key=query) 項目的集合，其中每個查詢都與唯一索引鍵相關聯，而現代化程式會使用此索引鍵來執行指定的查詢：

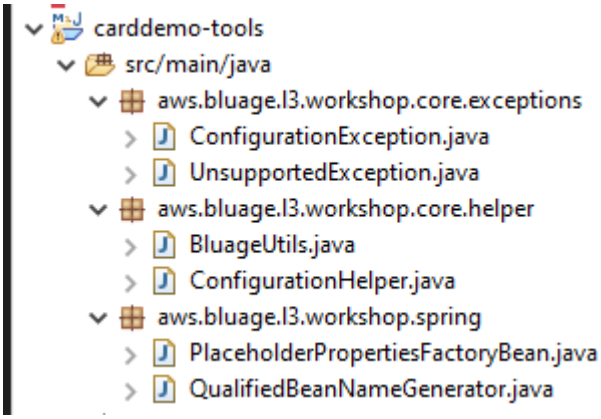


例如，COSGN00C 程式正在執行金鑰為 "COSGN00C_1" 的查詢 (sql 檔案的第一個項目）：



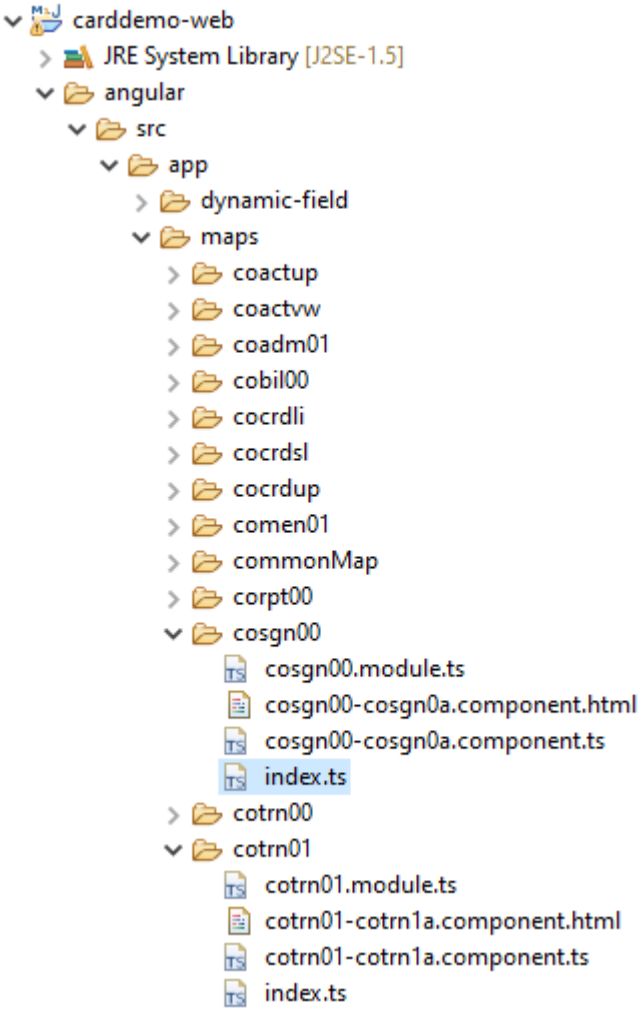
公用程式專案內容

名稱以「工具」結尾的公用程式專案包含一組技術公用程式，所有其他專案可能使用。



Web 專案 （ 多個 ） 內容

Web 專案只有在現代化舊版 UI 元素時才存在。用來建置現代化應用程式的現代 UI 元素是以 Angular 為基礎。用於顯示現代化成品的範例應用程式是 COBOL/CICS 應用程式，在大型主機上執行。CICS 系統使用 MAPs 來代表 UI 畫面。對於每個映射，對應的現代元素將是 html 檔案，並附帶 Typescript 檔案：



Web 專案僅負責應用程式的前端方面 服務專案依賴於公用程式和實體專案，可提供後端服務。前端與後端之間的連結是透過名為 Gapwalk-Application 的 Web 應用程式建立，這是標準 AWS Blu Age 執行期分佈的一部分。

執行和呼叫程式

在舊版系統上，程式會編譯為獨立的可執行檔，可透過 CALL 機制呼叫自己，例如 COBOL CALL 陳述式，並在需要時傳遞引數。現代化應用程式提供相同的功能，但使用不同的方法，因為涉及成品的性質與舊版成品不同。

在現代化方面，程式進入點是實作`Program`界面的特定類別，是 Spring 元件 (@Component)，位於名為 的服務專案中*`base package`*`.program`。

程式註冊

每次啟動託管現代化應用程式的 Tomcat 伺服器時，也會啟動服務 Springboot 應用程式，這會觸發程式註冊。名為 的專用登錄`ProgramRegistry`檔會填入程式項目，每個使用其識別符註冊的程式，每個已知的程式識別符各一個項目，這表示如果程式是由數個不同的識別符所知道，登錄檔會包含任意數量的項目，只要有識別符即可。

指定程式的註冊依賴於 getProgramIdentifiers() 方法傳回的識別符集合：

Cbact04c.java

```java
package aws.bluage.l3.workshop.program;

import aws.bluage.l3.workshop.SpringBootLauncher;

/**
 * Reference the spring application of program CBACT04C.
 *  Provides an access to the contained program for the run unit.
 */
@Component
@Import({
    aws.bluage.l3.workshop.cbact04c.business.context.Cbact04cConfiguration.class,
    aws.bluage.l3.workshop.cbact04c.business.context.Cbact04cContext.class,
    aws.bluage.l3.workshop.cbact04c.service.impl.Cbact04cProcessImpl.class
})
public class Cbact04c implements Program {
    /**
     * Unique identifiers for the contained program.
     */
    private static final Set<String> programIdentifiers = Collections.unmodifiableSet(Stream.of("CBACT04C").collect(Collectors.toSet()));

    /**
     * Main program identifier for the contained program.
     */
    private static final String programIdentifier = "CBACT04C";
    @Autowired
    PlatformTransactionManager transactionManager;

    @Autowired
    Map<String, DataSource> datasources;
    @Autowired
    BeanFactory beanFactory;
    /**
     * {@inheritDoc}
     */
    @Override
    public ConfigurableApplicationContext getSpringApplication() {
        return SpringBootLauncher.getCac();
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public void updateExecutionContext(ExecutionContext executionContext) {
        executionContext.setDatasources(datasources);
        executionContext.setDatabaseSupport(ExecutionContext.DatabaseSupport.POSTGRE);
        executionContext.setSqlcaVersion(ExecutionContext.SqlcaVersion.getEnum("ansi-comp5"));
        executionContext.setTransactionManager(transactionManager);
        executionContext.setUseSQLDateNewParadigm(true);
        executionContext.setUseSQLTrimStringType(false);
    }

    /**
     * {@inheritDoc}
     */
    @Override
    public Set<String> getProgramIdentifiers() {
        return programIdentifiers;
    }
```

在此範例中，程式註冊一次，名稱為「CBACT04C」（查看 programIdentifiers 集合的內容）。tomcat 日誌會顯示每個程式註冊。程式註冊僅取決於宣告的程式識別符，而非程式類別名稱本身（雖然通常符合程式識別符和程式類別名稱。

相同的註冊機制適用於各種公用程式 AWS Blu Age Web 應用程式帶來的公用程式程式，這些應用程式屬於 AWS Blu Age 執行期分佈的一部分。例如，Gapwalk-Utility-Pgm Webapp 提供 z/OS 系統公用程式 (IDCAMS、ICEGENER、SORT 等）的功能對等項目，並且可由現代化程式或指令碼呼叫。在 Tomcat 啟動時註冊的所有可用公用程式都會記錄在 Tomcat 日誌中。

指令碼和協助程式註冊

在 Tomcat 啟動時間，位於 /src/main/resources/scripts 資料夾階層的 groovy 指令碼會發生類似的註冊程序。指令碼資料夾階層會周遊，而探索到的所有 groovy 指令碼（特殊函數除外.groovy 預留指令碼）都會在 中註冊`ScriptRegistry`，並使用其簡短名稱（指令碼檔案名稱的一部分，位於第一個點字元之前）做為擷取金鑰。

> **Note**
>
> - 如果多個指令碼的檔案名稱會導致產生相同的註冊金鑰，則只會註冊最新的 ，並覆寫該指定金鑰先前遇到的任何註冊。
> - 考量上述注意事項，當使用子資料夾時，當註冊機制扁平化階層，並可能導致非預期的覆寫時，請特別注意。階層不會計入註冊程序中：通常 /scripts/A/myscript.groovy 和 /scripts/B/myscript.groovy 會導致 /scripts/B/myscript.groovy 覆寫 /scripts/A/myscript.groovy。

/src/main/resources/daemons 資料夾中的 groovy 指令碼處理方式略有不同。它們仍然註冊為一般指令碼，但此外，它們會在 Tomcat 啟動時間以非同步方式直接啟動一次。

在 中註冊指令碼後`ScriptRegistry`，REST 呼叫可以使用 Gapwalk-Application 公開的專用端點啟動指令碼。如需詳細資訊，請參閱對應的文件。

呼叫程式的程式

每個程式都可以將另一個程式稱為子程式，將參數傳遞給它。程式使用 `ExecutionController`界面的實作來執行此操作（大多數情況下，這會是`ExecutionControllerImpl`執行個體），以及名為的流暢 API 機制`CallBuilder`來建置程式呼叫引數。

所有程式方法都會同時採用 `RuntimeContext`和 `ExecutionController`作為方法引數，因此 `ExecutionController` 一律可用於呼叫其他程式。

例如，請參閱下圖，其中顯示 CBST03A 程式如何將 CBST03B 程式呼叫為子程式，並將參數傳遞給程式：

CbstmO3aProcessImpl.java

```java
    /**
     * Process operation xreffileGetNext.
     *
     * ----------------------------------------------------------------*
     *
     * @param ctx
     * @param ctrl
     */
    @Override
    public void xreffileGetNext(final Cbstm03aContext ctx, final ExecutionController ctrl) {
        ctx.getWsM03bArea().setWsM03bDd("XREFFILE");
        ctx.getWsM03bArea().setM03bRead(true);
        DataUtils.setToZeroes(ctx.getWsM03bArea().getWsM03bRcReference());
        DataUtils.setToBlank(ctx.getWsM03bArea().getWsM03bFldtReference());
        ctrl.callSubProgram("CBSTM03B", CallBuilder.newInstance()
            .byReference(ctx.getWsM03bArea())
            .getArguments(), ctx);
        if (DataUtils.compare(ctx.getWsM03bArea().getWsM03bRcReference(), "00") == 0) {

            /*
            Do nothing */
        } else if (DataUtils.compare(ctx.getWsM03bArea().getWsM03bRcReference(), "10") == 0) {
            ctx.getMiscVariables().setEndOfFile("Y");
        } else {
            if (LOGGER.isInfoEnabled()) LOGGER.info("ERROR READING XREFFILE");
            if (LOGGER.isInfoEnabled()) LOGGER.info("{}{}" , "RETURN CODE: " , ctx.getWsM03bArea().getWsM03bRc());
            abendProgram(ctx, ctrl);
        }
        ctx.getCardXrefRecord().setBytes(ctx.getWsM03bArea().getWsM03bFldtReference().getBytes());
    }
```

- 的第一個引數`ExecutionController.callSubProgram`是要呼叫之程式的識別符 （ 也就是用於程式註冊的識別符之一 -- 請參閱上述段落 ）。
- 第二個引數是 建置的結果`CallBuilder`，是 的陣列`Record`，對應於從來電者傳遞給受話方的資料。
- 第三個和最後一個引數是呼叫者`RuntimeContext`執行個體。

所有三個引數都是強制性的，不能為 null，但第二個引數可以是空的陣列。

只有在原本設計為這麼做時，受話方才能處理傳遞的參數。對於舊版 COBOL 程式，這表示具有 LINKAGE 區段和 USING 子句，讓程序分區使用 LINKAGE 元素。

例如，請參閱對應的 CBSTM03B.CBL COBOL 來源檔案：

github.com/aws-samples/aws-mainframe-modernization-carddemo/blob/main/app/cbl/CBSTM03B.CBL

```
 98
 99          LINKAGE SECTION.
100          01  LK-M03B-AREA.
101              05  LK-M03B-DD          PIC X(08).
102              05  LK-M03B-OPER        PIC X(01).
103                88  M03B-OPEN       VALUE 'O'.
104                88  M03B-CLOSE      VALUE 'C'.
105                88  M03B-READ       VALUE 'R'.
106                88  M03B-READ-K     VALUE 'K'.
107                88  M03B-WRITE      VALUE 'W'.
108                88  M03B-REWRITE    VALUE 'Z'.
109              05  LK-M03B-RC          PIC X(02).
110              05  LK-M03B-KEY         PIC X(25).
111              05  LK-M03B-KEY-LN      PIC S9(4).
112              05  LK-M03B-FLDT        PIC X(1000).
113
114          PROCEDURE DIVISION USING LK-M03B-AREA.
115
```

因此，CBSTM03B 程式採用單一 Record做為參數 （大小為 1 的陣列）。這是 CallBuilder 建置的內容，使用 byReference() 和 getArguments() 方法鏈結。

CallBuilder 流暢的 API 類別有多種方法可用於填入要傳遞給受話方的引數陣列：

- asPointer(RecordAdaptable)：新增指標類型的引數，以參考方式表示。指標代表目標資料結構的地址。
- byReference(RecordAdaptable)：依參考新增 引數。發起人將看到被發起人執行的修改。
- byReference(RecordAdaptable)：上一個方法的 varargs 變體。
- byValue(Object)：新增 引數，Record依值轉換為 。發起人不會看到被發起人執行的修改。
- byValue(RecordAdaptable)：與先前的方法相同，但引數可直接做為 使用RecordAdaptable。
- byValueWithBounds(Object， int， int)：新增 引數，轉換為 Record，依值擷取指定邊界定義的位元組陣列部分。

最後，getArguments 方法會收集所有新增的引數，並以 陣列傳回這些引數Record。

> **Note**
>
> 呼叫者的責任是確保引數陣列具有所需的大小，以確保項目在記憶體配置方面具有連結元素的預期配置，正確排序且相容。

指令碼呼叫程式

從 groovy 指令碼呼叫已註冊的程式需要使用實作`MainProgramRunner`界面的類別執行個體。通常，透過 Spring 的 ApplicationContext 使用量來取得這類執行個體：

REPROC.groovy

```groovy
// Import
import com.netfective.bluage.gapwalk.rt.provider.ScriptRegistry
import com.netfective.bluage.gapwalk.rt.call.MainProgramRunner
import com.netfective.bluage.gapwalk.io.support.FileConfigurationUtils
import com.netfective.bluage.gapwalk.rt.job.support.DefaultJobContext
import com.netfective.bluage.gapwalk.rt.utils.GroovyUtils
import com.netfective.bluage.gapwalk.rt.io.support.FileConfiguration
import com.netfective.bluage.gapwalk.rt.shared.AbendException
import com.netfective.bluage.gapwalk.rt.call.exception.GroovyExecutionException
// Variables
mpr = applicationContext.getBean("com.netfective.bluage.gapwalk.rt.call.ExecutionController", MainProgramRunner.class)
TreeMap mapTransfo = [:]
```

`MainProgramRunner` 界面可用後，請使用 runProgram 方法來呼叫程式，並將目標程式的識別符做為參數傳遞：

REPROC.groovy

```
50 //*******************************************************************
51 //*                              STEPS                              *
52 //*******************************************************************
53 // STEP PRC001 - PGM - IDCAMS*****************************************************
54 def stepPRC001(Object shell, Map params, Map programResults){
55     shell.with {
56         if (checkValidProgramResults(programResults)) {
57             return execStep("PRC001", "IDCAMS", programResults, {
58                 mpr
59                     .withFileConfigurations(new FileConfigurationUtils()
60                         .systemOut("SYSPRINT")
61                         .output("*")
62                         .build()
63                         .bluesam("FILEIN")
64                         .dataset("NULLFILE")
65                         .disposition("SHR")
66                         .build()
67                         .bluesam("FILEOUT")
68                         .dataset("NULLFILE")
69                         .disposition("SHR")
70                         .build()
71                         .fileSystem("SYSIN")
72                         .path("&CNTLLIB(REPROCT)")
73                         .disposition("SHR")
74                         .build()
75                         .getFileConfigurations(fcmap))
76                     .withParameters(params)
77                     .runProgram("IDCAMS")
78                 })
79         }
80     }
81 }
```

在先前的範例中，任務步驟會呼叫 IDCAMS （檔案處理公用程式程式），提供實際資料集定義與其邏輯識別符之間的映射。

處理資料集時，舊版程式大多使用邏輯名稱來識別資料集。從指令碼呼叫程式時，指令碼必須映射邏輯名稱與實際的實體資料集。這些資料集可能位於檔案系統、Blusam 儲存體中，甚至由內嵌串流、數個資料集的串連或 GDG 的產生所定義。

使用 withFileConfiguration 方法建置邏輯到資料集的實體映射，並將其提供給呼叫的程式。

撰寫您自己的程式

撰寫自己的指令碼程式或其他現代化程式來呼叫 是常見的任務。一般而言，在現代化專案中，當可執行式舊版程式以現代化程序不支援的語言撰寫，或來源已遺失 （是，可能會發生這種情況），或程式是來源無法使用的公用程式時，您會撰寫自己的程式。

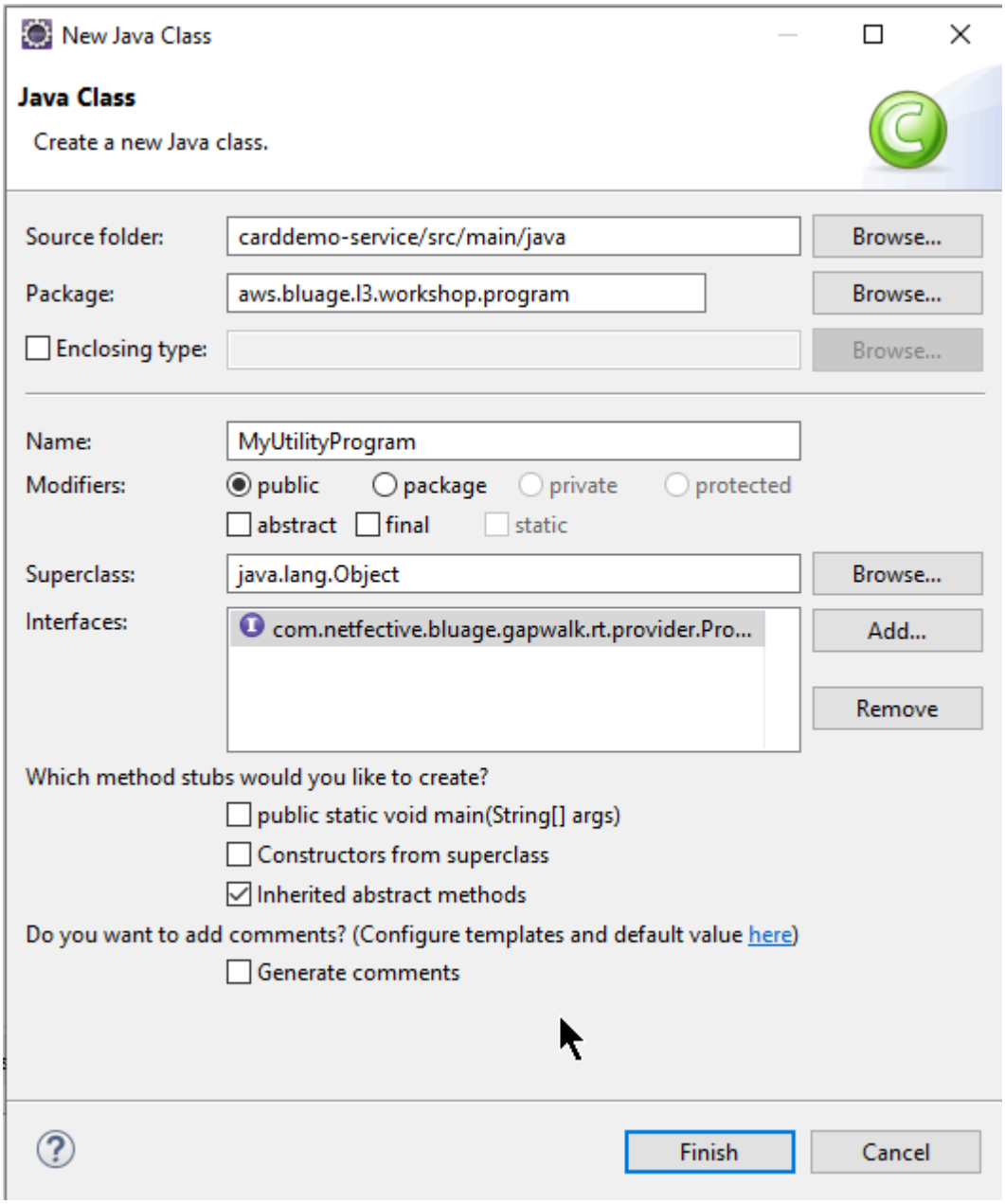
在這種情況下，您可能必須自行在 Java 中撰寫缺少的程式（假設您對程式預期的行為、程式引數的記憶體配置有的話，以此類推）。您的 Java 程式必須符合本文件所述的程式機制，以便其他程式和指令碼可以執行。

若要確保程式可供使用，您必須完成兩個必要步驟：

- 撰寫可正確實作`Program`界面的類別，以便註冊和呼叫。
- 請確定您的程式已正確註冊，以便從其他程式/指令碼中可見。

撰寫程式實作

使用您的 IDE 來建立新的 Java 類別，以實作`Program`界面：



下圖顯示 Eclipse IDE，負責建立要實作的所有強制性方法：

MyUtilityProgram.java

```java
package aws.bluage.l3.workshop.program;

import java.util.Set;

public class MyUtilityProgram implements Program {

    @Override
    public ConfigurableApplicationContext getSpringApplication() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public Set<String> getProgramIdentifiers() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public Context getContext() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public void run(ExecutionController ctrl) {
        // TODO Auto-generated method stub

    }

}
```

彈簧整合

首先，必須將 類別宣告為 Spring 元件。使用 註釋來@Component註釋 類別：

```java
import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.stereotype.Component;

import com.netfective.bluage.gapwalk.rt.call.ExecutionController;
import com.netfective.bluage.gapwalk.rt.context.Context;
import com.netfective.bluage.gapwalk.rt.provider.Program;

import aws.bluage.l3.workshop.SpringBootLauncher;

@Component
public class MyUtilityProgram implements Program {
```

接著，正確實作所需的方法。在此範例內容中，我們將 MyUtilityProgram 新增至已包含所有現代化程式的套件。該置放允許程式使用現有的 Springboot 應用程式來提供 getSpringApplication 方法實作ConfigurableApplicationContext所需的 ：

```
public class MyUtilityProgram implements Program {

    @Override
    public ConfigurableApplicationContext getSpringApplication() {
        return SpringBootLauncher.getCac();
    }
```

您可以為自己的程式選擇不同的位置。例如，您可以在另一個專用服務專案中找到指定的程式。確定指定的服務專案有自己的 Springboot 應用程式，這可讓您擷取 ApplicationContext （應該是 `ConfigurableApplicationContext`)。

將身分提供給程式

若要讓其他程式和指令碼呼叫，程式必須至少提供一個識別符，這不得與系統內的任何其他現有已註冊程式衝突。識別符選擇可能是由需要涵蓋現有舊版程式取代所驅動；在這種情況下，您必須使用預期的識別符，如同在舊版程式中發現的 CALL 事件中所滿足。在舊版系統中，大部分的程式識別符長度為 8 個字元。

在程式中建立一組無法修改的識別符，是執行此操作的一種方式。下列範例顯示選擇「MYUTILPG」作為單一識別符：

```
@Component
public class MyUtilityProgram implements Program {

    /**
     * Unique identifiers for the contained program.
     */
    private static final Set<String> programIdentifiers = Collections.unmodifiableSet(Stream.of("MYUTILPG").collect(Collectors.toSet()));

    public ConfigurableApplicationContext getSpringApplication() {

    @Override
    public Set<String> getProgramIdentifiers() {
        return programIdentifiers;
    }
```

將程式與內容建立關聯

程式需要配套`RuntimeContext`執行個體。對於現代化程式， AWS Blu Age 會使用屬於舊版程式的資料結構，自動產生配套內容。

如果您要撰寫自己的程式，您也必須撰寫配套內容。

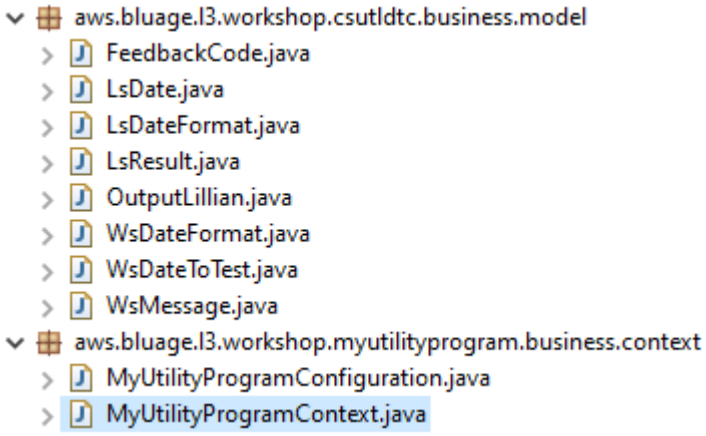
參考 程式相關類別，您可以看到程式至少需要兩個配套類別：

- 組態類別。
- 使用 組態的內容類別。

如果公用程式程式使用任何額外的資料結構，則應一併寫入並由內容使用。

這些類別應位於套件中，屬於套件階層的一部分，該階層將在應用程式啟動時掃描，以確保內容元件和組態將由 Spring 架構處理。

讓我們在*base package*.myutilityprogram.business.context套件中撰寫在實體專案中新建立的最小組態和內容：



以下是組態內容。它使用的組態組建類似於附近的其他 -- 現代化 -- 程式。您可能必須根據您的特定需求自訂此項目。

MyUtilityProgramConfiguration.java

```
1  package aws.bluage.l3.workshop.myutilityprogram.business.context;
2
3  import java.nio.charset.Charset;
4
5  import org.springframework.context.annotation.Bean;
6  import org.springframework.context.annotation.Lazy;
7
8  import com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration;
9  import com.netfective.bluage.gapwalk.datasimplifier.configuration.ConfigurationBuilder;
10
11 /**
12  * Creates Datasimplifier configuration for the MyUtilityProgram context.
13  */
14 @org.springframework.context.annotation.Configuration
15 @Lazy
16 public class MyUtilityProgramConfiguration {
17
18     @Bean(name = "MyUtilityProgramContextConfiguration")
19     public Configuration configuration() {
20         return new ConfigurationBuilder()
21                 .encoding(Charset.forName("CP1047"))
22                 .humanReadableEncoding(Charset.forName("ISO-8859-15"))
23                 .initDefaultByte(0)
24                 .build();
25     }
26 }
27
```

備註：

- 一般命名慣例是 ProgramName 組態。
- 它必須使用 @org.springframework.context.annotation.Configuration 和 @Lazy 註釋。
- Bean 名稱通常遵循 ProgramNameContextConfiguration ContextConfiguration慣例，但這並非強制性。請務必避免整個專案發生 Bean 名稱衝突。
- 要實作的單一方法必須傳回`Configuration`物件。使用`ConfigurationBuilder`流暢的 API 來協助您建立一個。

以及相關聯的內容：

MyUtilityProgramContext.java

```
2
3  import org.springframework.beans.factory.annotation.Qualifier;
4  import org.springframework.context.annotation.Lazy;
5  import org.springframework.context.annotation.Scope;
6  import org.springframework.stereotype.Component;
7
8  import com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration;
9  import com.netfective.bluage.gapwalk.rt.context.RuntimeContext;
10
11 @Component("aws.bluage.l3.workshop.mytuilityprogram.business.context.MyUtilityProgramContext")
12 @Lazy
13 @Scope("prototype")
14 public class MyUtilityProgramContext extends RuntimeContext{
15
16     protected MyUtilityProgramContext(@Qualifier("MyUtilityProgramContextConfiguration") Configuration configuration) {
17         super(configuration);
18     }
19
20     @Override
21     public void cleanUp() {
22         // TODO implement clean-up of associated data structures if any
23     }
24
25     @Override
26     protected void doReset() {
27         // TODO implement reset of associated data structures if any
28     }
29
30 }
31
```

備註

- 內容類別應擴展現有的`Context`界面實作（`RuntimeContext`或 `JicsRuntimeContext`，這是`RuntimeContext`使用 JICS 特定項目增強的）。
- 一般命名慣例是 ProgramName 內容。
- 您必須將其宣告為原型元件，並使用 @Lazy 註釋。
- 建構函數是指相關聯的組態，使用 @Qualifier 註釋以適當的組態類別為目標。
- 如果公用程式使用一些額外的資料結構，它們應該是：

- 寫入並新增至*base package*.business.model套件。
- 在內容中參考 。查看其他現有的內容類別，了解如何參考資料細分類別，並視需要調整內容方法（建構函式/清除/重設）。

現在有可用的專用內容，讓新程式使用它：

MyUtilityProgram.java

```
19 import aws.bluage.l3.workshop.SpringBootLauncher;
20
21 @Component
22 @Import({
23     aws.bluage.l3.workshop.myutilityprogram.business.context.MyUtilityProgramConfiguration.class,
24     aws.bluage.l3.workshop.myutilityprogram.business.context.MyUtilityProgramContext.class
25 })
26 public class MyUtilityProgram implements Program {
27
28     @Autowired
29     BeanFactory beanFactory;
30
31     /**
32      * Unique identifiers for the contained program.
33      */
34     private static final Set<String> programIdentifiers = Collections.unmodifiableSet(Stream.of("MYUTILPG").collect(Collectors.toSet()));
35
36     private static final String programIdentifier = "MYUTILPG";
37
38     @Override
39     public ConfigurableApplicationContext getSpringApplication() {
40         return SpringBootLauncher.getCac();
41     }
42
43     @Override
44     public Set<String> getProgramIdentifiers() {
45         return programIdentifiers;
46     }
47
48     /**
49      * {@inheritDoc}
50      */
51     @Override
52     public String getProgramMainIdentifier() {
53         return programIdentifier;
54     }
55
56
57     @Override
58     public Context getContext() {
59         return ProgramContextStore.getOrCreate(
60         getProgramMainIdentifier(),
61         aws.bluage.l3.workshop.myutilityprogram.business.context.MyUtilityProgramContext.class,
62         beanFactory);
63     }
64
```

備註：

- getContext 方法必須嚴格實作，如下所示，使用委派至 ProgramContextStore類別的 getOrCreate 方法和自動有線彈簧 BeanFactory。單一程式識別符用於在 中存放程式內容ProgramContextStore；此識別符參考為「程式主要識別符」。
- 必須使用@Import彈簧註釋來參考配套組態和內容類別。

實作商業邏輯

當程式骨架完成時，請實作新公用程式的商業邏輯。

在程式的 run方法中執行此操作。每當呼叫程式時，就會執行此方法，無論是透過其他程式或指令碼。

編碼愉快！

處理程式註冊

最後，請確定新的程式已在 中正確註冊ProgramRegistry。如果您已將新程式新增至已包含其他程式的套件，則無需再執行任何操作。新程式會在應用程式啟動時，取得並註冊其所有鄰近程式。

如果您選擇程式的其他位置，則必須確保程式在 Tomcat 啟動時已正確註冊。如需有關如何執行此操作的一些啟發，請查看服務專案中產生的 SpringbootLauncher 類別的初始化方法（請參閱服務專案內容)。

檢查 Tomcat 啟動日誌。系統會記錄每個程式註冊。如果您的程式已成功註冊，您會找到相符的日誌項目。

當您確定您的程式已正確註冊時，您可以開始反覆運算商業邏輯編碼。

完整名稱映射

本節包含用於現代化應用程式的 AWS Blu Age 和第三方完整名稱映射清單。

AWS Blu Age 完整名稱映射

短名稱	完整名稱
CallBuilder	com.netfective.bluage.gapwalk.runtime.statements.CallBuilder
Configuration	com.netfective.bluage.gapwalk.datasimplifier.configuration.Configuration
ConfigurationBuilder	com.netfective.bluage.gapwalk.datasimplifier.configuration.ConfigurationBuilder

短名稱	完整名稱
ExecutionController	com.netfective.bluage.gapwalk.rt.call.ExecutionController
ExecutionControllerImpl	com.netfective.bluage.gapwalk.rt.call.internal.ExecutionControllerImpl
File	com.netfective.bluage.gapwalk.rt.io.File
MainProgramRunner	com.netfective.bluage.gapwalk.rt.call.MainProgramRunner
Program	com.netfective.bluage.gapwalk.rt.provider.Program
ProgramContextStore	com.netfective.bluage.gapwalk.rt.context.ProgramContextStore
ProgramRegistry	com.netfective.bluage.gapwalk.rt.provider.ProgramRegistry
Record	com.netfective.bluage.gapwalk.datasimplifier.data.Record
RecordEntity	com.netfective.bluage.gapwalk.datasimplifier.entity.RecordEntity
RuntimeContext	com.netfective.bluage.gapwalk.rt.context.RuntimeContext
SimpleStateMachineController	com.netfective.bluage.gapwalk.rt.statemachine.SimpleStateMachineController

短名稱	完整名稱
StateMachineController	com.netfective.bluage.gapwalk.rt.statemachine.StateMachineController
StateMachineRunner	com.netfective.bluage.gapwalk.rt.statemachine.StateMachineRunner

第三方完整名稱映射

短名稱	完整名稱
@Autowired	org.springframework.beans.factory.annotation.Autowired
@Bean	org.springframework.context.annotation.Bean
BeanFactory	org.springframework.beans.factory.BeanFactory
@Component	org.springframework.stereotype.Component
ConfigurableApplicationContext	org.springframework.context.ConfigurableApplicationContext
@Import	org.springframework.context.annotation.Import
@Lazy	org.springframework.context.annotation.Lazy

什麼是 AWS 藍年資料簡化器

在大型主機和中階系統上（在下列主題中稱為「舊版」系統），經常使用的程式設計語言，例如 COBOL、PL/I 或 RPG，可提供記憶體的低階存取。此存取著重於透過原生類型存取的記憶體配置，例如分區、封裝或英數字元，也可能透過群組或陣列彙總。

透過輸入欄位和直接存取位元組（原始記憶體）來混合存取指定記憶體片段，並存在於指定程式中。例如，COBOL 程式會將引數以連續位元組集 (LINKAGE) 的形式傳遞給發起人，或以相同方式從檔案讀取/寫入資料（記錄），同時解譯這類記憶體範圍，其中包含以複製手冊整理的類型欄位。

這類記憶體原始和結構化存取的組合、對精確位元組層級記憶體配置的依賴，以及舊版類型，例如分區或封裝，都是 Java 程式設計環境中原生或輕鬆可用的功能。

作為將舊版程式現代化為 Java 的 AWS Blu Age 解決方案的一部分，Data Simplifier 程式庫提供此類建構給現代化 Java 程式，並以盡可能熟悉的方式向 Java 開發人員公開這些建構 (getters/setters、byte 陣列、類別型）。這是從此類程式產生的現代化 Java 程式碼的核心相依性。

為求簡化，下列大部分說明皆以 COBOL 建構為基礎，但您可以將相同的 API 用於 PL1 和 RPG 資料配置現代化，因為大多數概念都類似。

主題

- 主要類別
- 資料繫結和存取
- 討論的 Java 類型的 FQN

主要類別

為了方便閱讀，本文件使用 AWS Blu Age API 介面和類別的 Java 短名。如需詳細資訊，請參閱討論的 Java 類型的 FQN。

低階記憶體表示

在最低層級，`Record` 界面代表記憶體（以快速、隨機方式存取的連續位元組範圍）。此界面基本上是固定大小的位元組陣列抽象。因此，它提供設定程式和取得程式能夠存取或修改基礎位元組。

結構化資料表示

若要表示結構式資料，例如「01 資料項目」或「01 複製手冊」，如 COBOL DATA DIVISION 中所找到，會使用 `RecordEntity` 類別的子類別。這些通常不是手動撰寫，而是由對應的舊版建構中的

AWS Blu Age 現代化工具產生。了解其主要結構和 API 仍然很有用，因此您可以了解現代化程式中的程式碼如何使用它們。在 COBOL 的情況下，該程式碼是從 PROCEDURE DIVISION 產生的 Java。

產生的程式碼代表具有RecordEntity子類別的每個「01 資料項目」；每個基本欄位或彙總編寫以私有 Java 欄位表示，組織為樹狀目錄（每個項目都有父項，但根欄位除外）。

為方便說明，以下是 COBOL 資料項目範例，後面接著對應的 AWS Blu Age 產生的程式碼，可對其進行現代化：

```
01 TST2.
 02 FILLER PIC X(4).
 02 F1     PIC 9(2) VALUE 42.
 02 FILLER PIC X.
 02        PIC 9(3) VALUE 123.
 02 F2     PIC X VALUE 'A'.
```

```
public class Tst2 extends RecordEntity {

    private final Group root = new Group(getData()).named("TST2");
    private final Filler filler = new Filler(root,new AlphanumericType(4));
    private final Elementary f1 = new Elementary(root,new ZonedType(2, 0, false),new
 BigDecimal("42")).named("F1");
    private final Filler filler1 = new Filler(root,new AlphanumericType(1));
    private final Filler filler2 = new Filler(root,new ZonedType(3, 0, false),new
 BigDecimal("123"));
    private final Elementary f2 = new Elementary(root,new
 AlphanumericType(1),"A").named("F2");



    /**
     * Instantiate a new Tst2 with a default record.
     * @param configuration the configuration
     */
    public Tst2(Configuration configuration) {
        super(configuration);
        setupRoot(root);
    }
    /**
     * Instantiate a new Tst2 bound to the provided record.
     * @param configuration the configuration
     * @param record the existing record to bind
     */
    public Tst2(Configuration configuration, RecordAdaptable record) {
```

```
        super(configuration);
        setupRoot(root, record);
    }

    /**
     * Gets the reference for attribute f1.
     * @return the f1 attribute reference
     */
    public ElementaryRangeReference getF1Reference() {
        return f1.getReference();
    }

    /* *
     * Getter for f1 attribute.
     * @return f1 attribute
     */
    public int getF1() {
        return f1.getValue();
    }

    /**
     * Setter for f1 attribute.
     * @param f1 the new value of f1
     */
    public void setF1(int f1) {
        this.f1.setValue(f1);
    }
    /**
     * Gets the reference for attribute f2.
     * @return the f2 attribute reference
     */
    public ElementaryRangeReference getF2Reference() {
        return f2.getReference();
    }

    /**
     * Getter for f2 attribute.
     * @return f2 attribute
     */
    public String getF2() {
        return f2.getValue();
    }

    /**
```

```
     * Setter for f2 attribute.
     * @param f2 the new value of f2
     */
    public void setF2(String f2) {
        this.f2.setValue(f2);
    }
}
```

基本欄位

類別欄位 Elementary（或 Filler，當未命名時）代表舊版資料結構的「分葉」。它們與基礎位元組 ("range") 的連續範圍相關聯，通常具有類型（可能參數化），表示如何解譯和修改這些位元組（分別透過「解碼」和「編碼」從/到位元組陣列的值）。

所有基本類型都是 的子類別RangeType。常見類型為：

COBOL 類型	資料簡化器類型
PIC X(n)	AlphanumericType
PIC 9(n)	ZonedType
PIC 9(n) COMP-3	PackedType
PIC 9(n) COMP-5	BinaryType

彙總欄位

彙總欄位會組織其內容的記憶體配置（其他彙總或基本欄位）。它們本身沒有基本類型。

Group 欄位代表記憶體中的連續欄位。每個包含的欄位在記憶體中以相同的順序排列，第一個欄位與記憶體中的群組欄位位置0處於偏移，第二個欄位位於偏移 0 + (size in bytes of first field)等。它們用於表示相同包含欄位下的 COBOL 欄位序列。

Union 欄位代表存取相同記憶體的多個欄位。每個包含的欄位都會以與記憶體中聯集欄位位置0的位移來配置。例如，它們用於代表 COBOL "REDEFINES" 建構（第一個聯集子是重新定義的資料項目，第二個子是其第一個重新定義等）。

陣列欄位 (的子類別Repetition) 代表其子欄位（無論是彙總本身或基本項目）配置的重複記憶體。它們在記憶體中配置了指定數量的這類子配置，每個配置都處於偏移 index * (size in bytes of child)。它們用於代表 COBOL "OCCURS" 建構。

基本概念

在某些現代化案例中，「基本」也可以用來呈現獨立的「根」資料項目。這些在使用 時非常類似RecordEntity，但不來自它，也不是以產生的程式碼為基礎。而是由 AWS Blu Age 執行時間直接提供做為Primitive界面的子類別。這類提供的類別範例為 Alphanumeric或 ZonedDecimal。

資料繫結和存取

結構式資料與基礎資料之間的關聯可以透過多種方式完成。

此用途的重要界面是 RecordAdaptable，用於取得RecordAdaptable基礎資料的Record「可寫入檢視」。如以下所示，多個類別實作 RecordAdaptable。互斥地， AWS 藍本存留期 APIs和程式碼處理低階記憶體 （ 例如程式引數、檔案 I/O 記錄、CICS 通訊區域、配置的記憶體...) 通常預期RecordAdaptable會作為該記憶體的控制代碼。

在 COBOL 現代化案例中，大多數資料項目與記憶體相關聯，這些項目將在對應的程式執行生命週期內修正。因此，RecordEntity子類別會在產生的父物件 （ 程式內容） 中執行個體化一次Record，並根據RecordEntity位元組大小來執行個體化其基礎 。

在其他 COBOL 案例中，例如將 LINKAGE 元素與程式引數建立關聯，或現代化 SET ADDRESS OF 建構，RecordEntity執行個體必須與提供的 建立關聯RecordAdaptable。為此，有兩個機制：

- 如果RecordEntity執行個體已存在，則方法 RecordEntity.bind(RecordAdaptable)（繼承自 Bindable) 可用來將此執行個體設為「點」至此 RecordAdaptable。RecordEntity 然後，在 上呼叫的任何 getter 或 setter 都會由基礎位元組進行支援 （ 位元組讀取或寫入 ）RecordAdaptable。
- 如果要RecordEntity執行個體化 ，RecordAdaptable則可使用產生且接受 的建構函數。

相反地，可以存取Record目前繫結至結構化資料的 。因此， RecordEntity會實作 RecordAdaptable，因此getRecord()可以在任何這類執行個體上呼叫 。

最後，許多 COBOL 或 CICS 動詞需要存取單一欄位，以供讀取或寫入之用。RangeReference 類別用於表示此類存取。其執行個體可從RecordEntity產生的getXXXReference()方法 (XXX 為存取欄位） 取得，並傳遞至執行時間方法。 RangeReference 通常用於存取整個 RecordEntity或 Group，而其子類別ElementaryRangeReference代表對Elementary欄位的存取。

請注意，上述大多數觀察都適用於Primitive子類別，因為它們會努力實作類似的行為，RecordEntity如同由 AWS Blu Age 執行時間 （ 而不是產生的程式碼） 所提供。為此目的， 的所有子類別都會Primitive實作 RecordAdaptable、 ElementaryRangeReference和 Bindable 介面，以用來取代RecordEntity子類別和基本欄位。

討論的 Java 類型的 FQN

下表顯示本節討論的 Java 類型的完整名稱。

短名稱	完整名稱
Alphanumeric	com.netfective.bluage.gapwalk.datasimplifier.elementary.Alphanumeric
AlphanumericType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.AlphanumericType
BinaryType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.BinaryType
Bindable	com.netfective.bluage.gapwalk.datasimplifier.data.Bindable
Elementary	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Elementary
ElementaryRangeReference	com.netfective.bluage.gapwalk.datasimplifier.entity.ElementaryRangeReference
Filler	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Filler
Group	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Group

短名稱	完整名稱
PackedType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.PackedType
Primitive	com.netfective.bluage.gapwalk.datasimplifier.elementary.Primitive
RangeReference	com.netfective.bluage.gapwalk.datasimplifier.entity.RangeReference
RangeType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.RangeType
Record	com.netfective.bluage.gapwalk.datasimplifier.data.Record
RecordAdaptable	com.netfective.bluage.gapwalk.datasimplifier.data.RecordAdaptable
RecordEntity	com.netfective.bluage.gapwalk.datasimplifier.entity.RecordEntity
Repetition	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Repetition
Union	com.netfective.bluage.gapwalk.datasimplifier.data.structure.Union

短名稱	完整名稱
ZonedDecimal	com.netfective.bluage.gapwalk.datasimplifier.elementary.ZonedDecimal
ZonedType	com.netfective.bluage.gapwalk.datasimplifier.metadata.type.ZonedType

AWS 藍本年齡藍調

在大型主機系統上（在下列主題中稱為「舊版」），商業資料通常會使用 VSAM（虛擬儲存存取方法）儲存。資料存放在「記錄」（位元組陣列），屬於「資料集」。

有四個資料集組織：

- KSDS：金鑰序列資料集 - 記錄會依主索引鍵（不允許重複的索引鍵）編製索引，也可以選擇使用其他的「備用」索引鍵。所有索引鍵值都是記錄位元組陣列的子集，每個索引鍵都由以下項目定義：
  - 位移（以 0 為基礎，0 是記錄位元組陣列內容的開頭，以位元組為單位）
  - a length（以位元組表示）
  - 是否容忍重複值
- ESDS：進入序列資料集 - 記錄主要是循序存取（根據其在資料集中的插入順序），但可以使用其他備用金鑰存取；
- RRDS：相對記錄資料集 - 使用「跳躍」存取記錄，使用相對記錄編號；跳躍可以向前或向後完成；
- LDS：線性資料集 - 其中沒有記錄，只是以頁面整理的位元組串流。主要用於舊版平台上的內部用途。

現代化舊版應用程式時，使用 AWS Blu Age Refactoring 方法，現代化應用程式不再旨在存取 VSAM 儲存的資料，同時保留資料存取邏輯。Blusam 元件是答案：它允許從舊版 VSAM 資料集匯出匯入資料，為現代化應用程式提供 API，以搭配專用的管理 Web 應用程式操作它們。請參閱 the section called “Blusam 管理主控台”。

> **Note**
>
> Blusam 僅支援 KSDS、ETS 和 RRDS。

Blusam API 可讓您保留資料存取邏輯（循序、隨機和相對讀取；插入、更新和刪除記錄），而依賴快取策略和 RDBMS 型儲存的元件架構則允許具有有限資源的高輸送量 I/O 操作。

Blusam 基礎設施

Blusam 依賴 PostgreSQL RDBMS 儲存資料集，包括原始記錄資料和金鑰索引（如適用）。最愛的選項是使用 Amazon Aurora PostgreSQL 相容引擎。本主題中的範例和圖例是以此引擎為基礎。

> **Note**
>
> 在伺服器啟動時，Blusam 執行時間會檢查是否存在某些強制性技術資料表，並在找不到它們時建立它們。因此，在組態中用來存取 Blusam 資料庫的角色必須獲得建立、更新和刪除資料庫資料表（資料列和資料表定義本身）的權限。如需如何停用 Blusam 的資訊，請參閱 the section called “Blusam 組態”。

快取

除了儲存體本身之外，Blusam 在與快取實作結合時運作速度更快。

目前支援兩個快取引擎：EhCache 和 Redis，每個引擎都有自己的使用案例：

- EhCache：獨立內嵌揮發性本機快取
  - 不符合 AWS 大型主機現代化受管環境部署的資格。
  - 通常在單一節點，例如單一 Apache Tomcat 伺服器，用於執行現代化應用程式時使用。例如，節點可能專用於託管批次任務。
  - 揮發性：EhCache 快取執行個體為揮發性；其內容會在伺服器關閉時遺失。
  - 內嵌：EhCache 和伺服器共用相同的 JVM 記憶體空間（定義託管機器的規格時要考量）。
- Redis：共用持久性快取
  - 符合 AWS 大型主機現代化受管環境部署的資格。
  - 通常用於多節點情況，特別是當多個伺服器落後於負載平衡器時。快取內容在所有節點之間共用。
  - Redis 是持久性的，不繫結至節點生命週期。它會在自己的專用機器或服務上執行（例如，Amazon ElastiCache)。快取是所有節點的遠端快取。

鎖定

為了處理資料集和記錄的並行存取，Blusam 依賴可設定的鎖定系統。鎖定可套用至兩個層級：資料集和記錄：

- 鎖定用於寫入目的的資料集，可防止所有其他用戶端在任何層級（資料集或記錄）對其執行寫入操作。
- 在記錄層級鎖定以進行寫入，將阻止其他用戶端僅對指定記錄執行寫入操作。

應根據快取組態相應地設定 Blusam 鎖定系統：

- 如果選擇 EhCache 做為快取實作，則不需要進一步鎖定組態，因為應該使用預設的記憶體內鎖定系統。
- 如果選擇 Redis 做為快取實作，則需要 Redis 型鎖定組態，以允許從多個節點同時存取。用於鎖定的 Redis 快取不必與用於資料集的 Redis 快取相同。如需設定 Redis 型鎖定系統的相關資訊，請參閱 the section called “Blusam 組態”。

Blusam 內部和資料從舊版遷移

儲存資料集：記錄和索引

匯入至 Blusam 的每個舊版資料集都會儲存到專用資料表；資料表的每一列都代表記錄，使用兩欄：

- 數值 ID 欄，大整數類型，即資料表主索引鍵，用於存放記錄的相對位元組地址 (RBA)。RBA 代表從資料集開始的位元組位移，從 0 開始。
- 位元組陣列記錄欄，用於存放原始記錄的內容。

例如，請參閱 CardDemo 應用程式中使用的 KSDS 資料集內容：

```
SELECT * FROM public.aws_m2_carddemo_acctdata_vsam_ksds
ORDER BY id ASC
```

Data output Messages Notifications

	id [PK] bigint	record bytea
1	0	[binary data]
2	300	[binary data]
3	600	[binary data]
4	900	[binary data]
5	1200	[binary data]
6	1500	[binary data]
7	1800	[binary data]
8	2100	[binary data]
9	2400	[binary data]
10	2700	[binary data]
11	3000	[binary data]
12	3300	[binary data]
13	3600	[binary data]

- 此特定資料集具有固定長度記錄，長度為 300 位元組 （因此，ID 的集合為 300 的倍數）。
- 根據預設，用於查詢 PostgreSQL 資料庫的 pgAdmin 工具不會顯示位元組陣列資料欄內容，而是列印 【二進位資料】 標籤。
- 原始記錄內容符合從舊版匯出的原始資料集，無需任何轉換。特別是，不會發生字元集轉換；這表示記錄的英數字元部分必須由使用舊版字元集的現代化應用程式解碼，很可能是 EBCDIC 變體。

關於資料集中繼資料和索引鍵：每個資料集都與名為 的資料表中的兩列相關聯`metadata`。這是預設的命名慣例。若要了解如何自訂，請參閱 the section called “Blusam 組態”。

	name [PK] text	metadata oid
1	AWS_M2_CARDDEMO_ACCTDATA_VSAM_KSDS	66320
2	AWS_M2_CARDDEMO_ACCTDATA_VSAM_KSDS___internal	66321

- 第一列具有資料集名稱做為名稱資料欄的值。中繼資料資料欄是二進位資料欄，其中包含指定資料集之一般中繼資料的二進位序列化。如需詳細資訊，請參閱the section called “一般資料集中繼資料屬性”。
- 第二列具有資料集名稱，其尾碼__internal'為名稱欄的值。中繼資料資料欄二進位內容取決於資料集的「權重」。
  - 對於小型/中型資料集，內容是壓縮序列化：
    - 資料集所使用的金鑰定義；主金鑰定義（適用於 KSDS) 和替代金鑰定義，如適用（適用於 KSDS / ESDS)
    - 索引鍵索引（如適用，使用備用索引鍵定義）：用於記錄的索引瀏覽；索引鍵索引會將索引鍵值映射至記錄的 RBA；
    - 記錄長度映射：用於記錄的循序/相對瀏覽；
  - 對於大型/極大型資料集，內容是壓縮序列化：
    - 資料集所使用的金鑰定義；主金鑰定義（適用於 KSDS) 和替代金鑰定義，如適用（適用於 KSDS / ESDS)

此外，使用分頁機制存放大型/非常大型的資料集索引（如適用）；索引頁面二進位序列化會以專用資料表的資料列（每個資料集索引鍵一個資料表）存放。索引的每一頁都會儲存在列中，並具有下列資料欄：

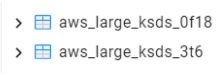
- ID：索引頁面的技術識別符（數字主索引鍵）；
- firstkey：存放在索引頁面中第一個（最低）索引鍵值的二進位值；
- lastkey：存放在索引頁面中最後一個（最高）索引鍵值的二進位值；
- 中繼資料：索引頁面的二進位壓縮序列化（將索引鍵值映射至記錄 RBAs)。

	id [PK] bigint	firstkey bytea	lastkey bytea	metadata oid
1	1	[binary data]	[binary da...	6458928
2	2	[binary data]	[binary da...	6458929
3	3	[binary data]	[binary da...	6458930
4	4	[binary data]	[binary da...	6458931
5	5	[binary data]	[binary da...	6458932
6	6	[binary data]	[binary da...	6458933
7	7	[binary data]	[binary da...	6458934
8	8	[binary data]	[binary da...	6458935
9	9	[binary data]	[binary da...	6458936

資料表名稱是資料集名稱和金鑰內部名稱的串連，其中包含金鑰的相關資訊，例如金鑰位移、金鑰是否接受重複（設定為 true 以允許重複）以及金鑰長度。例如，假設名為 "AWS_LARGE_KSDS" 的資料集具有下列兩個已定義的金鑰：

- 主索引鍵 【偏移：0，重複：false， length：18】
- 備用索引鍵 【偏移：3，重複： true，長度：6】

在此情況下，下表會存放與兩個索引鍵相關的索引。



使用寫入落後機制最佳化 I/O 輸送量

為了最佳化插入/更新/刪除操作效能，Blusam 引擎依賴可設定的寫入落後機制。該機制建立在專用執行緒集區上，該集區使用大量更新查詢來處理持久性操作，以最大化 Blusam 儲存的 I/O 輸送量。

Blusam 引擎會收集應用程式對記錄執行的所有更新操作，並建置要分派給專用執行緒進行處理的記錄批次。然後，這些批次會保留到 Blusam 儲存體，使用大量更新查詢，避免使用原子持久性操作，確保盡可能使用最佳的網路頻寬。

機制使用可設定的延遲（預設為一秒）和可設定的批次大小（預設為 10000 筆記錄）。一旦符合下列兩個條件中的第一個，就會執行建置持久性查詢：

- 已超過設定的延遲，且批次不是空的
- 批次中要處理的記錄數量達到設定的限制

若要了解如何設定寫入落後機制，請參閱 the section called “可選屬性”。

挑選適當的儲存方案

如上一節所示，資料集的儲存方式取決於其「權重」。但是，對於資料集，什麼被認為是小型、中型或大型？ 何時選擇分頁儲存策略，而不是一般儲存策略？

該問題的答案取決於下列各項。

- 託管將使用這些資料集之現代化應用程式的每個伺服器上可用的記憶體數量。
- 快取基礎設施上的可用記憶體數量（如果有的話）。

使用非分頁索引儲存方案時，每個資料集的完整索引和記錄大小集合會在資料集開啟時間載入伺服器記憶體。此外，如果涉及快取，則所有資料集記錄可能會使用一般方法預先載入快取，這可能會導致快取基礎設施端的記憶體資源耗盡。

根據已定義的金鑰數量、金鑰值的長度、記錄數量和同時開啟的資料集數量，可針對指定的已知使用案例大致評估耗用記憶體數量。

如需詳細資訊，請參閱 the section called “估計指定資料集的記憶體使用量”。

Blusam 遷移

為指定的資料集選取適當的儲存方案後，必須透過遷移舊版資料集來填入 blusam 儲存。

若要達成此目的，使用者必須使用舊版資料集的原始二進位匯出，而不需在匯出程序期間使用任何字元集轉換。從舊版系統傳輸資料集匯出時，請務必不要損毀二進位格式。例如，使用 FTP 時強制執行二進位模式。

原始二進位匯出僅包含記錄。匯入機制不需要匯出索引鍵/索引，因為匯入機制正在即時重新計算所有索引鍵/索引。

一旦資料集二進位匯出可用，便存在幾個將其遷移至 Blusam 的選項：

在 AWS 大型主機現代化受管環境上：

- 使用專用功能匯入資料集。請參閱 the section called “匯入應用程式的資料集”。

或

- 使用資料集大量匯入設施。請參閱 the section called “資料集定義參考” 和 the section called “VSAM 的資料集請求格式範例”。

或

- 使用 groovy 指令碼，使用專用載入服務匯入資料集。

> **Note**
> 現在只能使用 groovy 指令碼在大型主機現代化受管環境中匯入 LargeKSDS 和 LargeESDS。

Amazon EC2 上的 AWS 「藍時執行期」：

- 使用 匯入資料集the section called “Blusam 管理主控台”。

或

- 使用 groovy 指令碼，使用專用載入服務匯入資料集。

使用 Groovy 指令碼匯入資料集

本節將協助您撰寫 groovy 指令碼，將舊版資料集匯入 Blusam。

它從一些強制性匯入開始：

```
import com.netfective.bluage.gapwalk.bluesam.BluesamManager
import com.netfective.bluage.gapwalk.bluesam.metadata.Key;
import com.netfective.bluage.gapwalk.rt.provider.ServiceRegistry
import java.util.ArrayList; //used for alternate keys if any
```

之後，針對要匯入的每個資料集，程式碼會建置在指定的模式上：

1. 建立或清除映射物件
2. 使用所需的屬性填入映射（這隨資料集類型而異 -- 如需詳細資訊，請參閱下文）
3. 在服務登錄檔中擷取用於資料集類型的適當載入服務
4. 使用映射做為引數來執行服務

有 5 個服務實作可以使用下列識別符從服務登錄檔擷取：

- `"BluesamKSDSFileLoader"`：適用於小型/中型 KSDS

- "BluesamESDSFileLoader" 適用於中小型 ESDS
- "BluesamRRDSFileLoader"：適用於 RRDS
- "BluesamLargeKSDSFileLoader"：適用於大型 KSDS
- "BluesamLargeESDSFileLoader"：適用於大型 ESDS

是否選擇 KSDS/ESDS 的一般版本和大型版本服務，取決於資料集的大小，以及您要套用的儲存策略。若要了解如何選擇適當的儲存策略，請參閱 the section called “挑選適當的儲存方案”。

若要能夠成功將資料集匯入 Blusam，必須將適當的屬性提供給載入服務。

常見屬性：

- 強制性（適用於所有類型的資料集）
  - "bluesamManager"：預期值為 `applicationContext.getBean(BluesamManager.class)`
  - "datasetName"：資料集的名稱，做為字串
  - 「inFilePath」：舊版資料集匯出的路徑，做為字串
  - "recordLength"：固定記錄長度或變數記錄長度資料集的 0，以整數表示
- 選用
  - 不支援大型資料集：
    - "isAppend"：布林值旗標，表示匯入正在附加模式中發生（將記錄附加至現有的 blusam 資料集）。
    - 「useCompression」：布林值旗標，表示將使用壓縮來存放中繼資料。
  - 僅適用於大型資料集：
    - 「indexingPageSizeInMb」：每個索引頁面的每個索引鍵的大小，以絕對正整數表示

資料集類型相依屬性：

- KSDS/大型 KSDS：
  - mandatory
    - "primaryKey"：使用`com.netfective.bluage.gapwalk.bluesam.metadata.Key`建構函數呼叫的主要金鑰定義。
  - 選用：
    - "alternateKeys"：使用建構函數呼叫建置的備用金鑰定義的清單 `com.netfective.bluage.gapwalk.bluesam.metadata.Key` (`java.util.List`)。

- ESDS/大型 ESDS：
  - 選用：
    - "alternateKeys" ：使用建構函數呼叫建置的備用金鑰定義的清單 `com.netfective.bluage.gapwalk.bluesam.metadata.Key` (`java.util.List`)。
- RRDS：
  - 無。

金鑰建構函數呼叫：

- `new Key(int offset, int length)`： 會建立金鑰物件，其中包含指定的金鑰屬性（偏移和長度），且不允許重複。此變體應用於定義主索引鍵。
- `new Key(boolean allowDuplicates, int offset, int length)`： 會建立金鑰物件，其中包含指定的金鑰屬性（偏移和長度）和允許標記的重複項目。

下列 Groovy 範例說明各種載入案例。

使用兩個備用金鑰載入大型 KSDS：

```
import com.netfective.bluage.gapwalk.bluesam.BluesamManager
import com.netfective.bluage.gapwalk.bluesam.metadata.Key;
import com.netfective.bluage.gapwalk.rt.provider.ServiceRegistry
import java.util.ArrayList;

// Loading a large KSDS into Blusam
def map = [:]
map.put("bluesamManager", applicationContext.getBean(BluesamManager.class));
map.put("datasetName", "largeKsdsSample");
map.put("inFilePath", "/work/samples/largeKsdsSampleExport");
map.put("recordLength", 49);
map.put("primaryKey", new Key(0, 18));
ArrayList altKeys = [new Key(true, 10, 8), new Key(false, 0, 9)]
map.put("alternateKeys", altKeys);
map.put("indexingPageSizeInMb", 25);
def service = ServiceRegistry.getService("BluesamLargeKSDSFileLoader");
service.runService(map);
```

載入不含備用索引鍵的可變記錄長度 ESDS：

```
import com.netfective.bluage.gapwalk.bluesam.BluesamManager
```

```
import com.netfective.bluage.gapwalk.bluesam.metadata.Key;
import com.netfective.bluage.gapwalk.rt.provider.ServiceRegistry

// Loading an ESDS into Blusam
def map = [:]
map.put("bluesamManager", applicationContext.getBean(BluesamManager.class));
map.put("datasetName", "esdsSample");
map.put("inFilePath", "/work/samples/esdsSampleExport");
map.put("recordLength", 0);
def service = ServiceRegistry.getService("BluesamESDSFileLoader");
service.runService(map);
```

可變記錄長度資料集匯出將包含強制性的 Record Decriptor Word (RDW) 資訊，以允許在讀取時分割記錄。

載入固定記錄長度 RRDS：

```
import com.netfective.bluage.gapwalk.bluesam.BluesamManager
import com.netfective.bluage.gapwalk.bluesam.metadata.Key;
import com.netfective.bluage.gapwalk.rt.provider.ServiceRegistry

// Loading a RRDS into Blusam
def map = [:]
map.put("bluesamManager", applicationContext.getBean(BluesamManager.class));
map.put("datasetName", "rrdsSample");
map.put("inFilePath", "/work/samples/rrdsSampleExport");
map.put("recordLength", 180);
def service = ServiceRegistry.getService("BluesamRRDSFileLoader");
service.runService(map);
```

在多結構描述模式中載入資料集：

多結構描述模式：在某些舊版系統中，VSAM 檔案會組織成檔案集，允許程式存取和修改指定分割區內的資料。現代系統會將每個檔案集視為結構描述，啟用類似的資料分割和存取控制。

若要在 `application-main.yml` 檔案中啟用多結構描述模式，請參閱 the section called “Blusam 組態”。在此模式中，資料集可以使用共用內容載入至特定結構描述，該內容是執行時間資訊的記憶體內登錄檔。若要將資料集載入特定結構描述，請在資料集名稱前加上相關結構描述名稱。

將 KSDS 檔案載入至多結構描述模式的特定結構描述：

```
import com.netfective.bluage.gapwalk.bluesam.BluesamManager
import com.netfective.bluage.gapwalk.bluesam.metadata.Key;
```

```
import com.netfective.bluage.gapwalk.rt.provider.ServiceRegistry
import java.util.ArrayList;
import com.netfective.bluage.gapwalk.rt.shared.SharedContext;

// Loading a KSDS into Blusam
def map = [:]
String schema = "schema1";
String datasetName = schema+"|"+"ksdsSample";
SharedContext.get().setCurrentBlusamSchema(schema);
schema = SharedContext.get().getCurrentBlusamSchema();
map.put("bluesamManager", applicationContext.getBean(BluesamManager.class));
map.put("datasetName", datasetName);
map.put("inFilePath", "/work/samples/ksdsSampleExport");
map.put("recordLength", 49);
map.put("primaryKey", new Key(0, 18));
map.put("indexingPageSizeInMb", 25);
def service = ServiceRegistry.getService("BluesamKSDSFileLoader");
service.runService(map);
```

將大型 KSDS 檔案載入多結構描述模式的特定結構描述：

```
import com.netfective.bluage.gapwalk.bluesam.BluesamManager
import com.netfective.bluage.gapwalk.bluesam.metadata.Key;
import com.netfective.bluage.gapwalk.rt.provider.ServiceRegistry
import java.util.ArrayList;
import com.netfective.bluage.gapwalk.rt.shared.SharedContext;

// Loading a Large KSDS into Blusam
def map = [:]
String schema = "schema1";
String datasetName = schema+"|"+"largeKsdsSample";
SharedContext.get().setCurrentBlusamSchema(schema);
schema = SharedContext.get().getCurrentBlusamSchema();
map.put("bluesamManager", applicationContext.getBean(BluesamManager.class));
map.put("datasetName", datasetName);
map.put("inFilePath", "/work/samples/LargeKsdsSampleExport");
map.put("recordLength", 49);
map.put("primaryKey", new Key(0, 18));
map.put("indexingPageSizeInMb", 25);
def service = ServiceRegistry.getService("BluesamLargeKSDSFileLoader");
service.runService(map);
```

此外，組態項目（要在`application-main.yml`組態檔案中設定）可用來微調匯入程序：

- `bluesam.fileLoading.commitInterval`：嚴格正整數，定義一般 ESDS/KSDS/RRDS 匯入機制的遞交間隔。不適用於大型資料集匯入。預設為 100000。

Blusam 組態

設定 Blusam 發生在`application-main.yml`組態檔案中（或在`application-bac.yml`組態檔案中，用於獨立部署 Blusam 管理主控台 -- BAC -- 應用程式）。

Blusam 必須設定兩個層面：

- Blusam 儲存和快取存取組態
- Blusam 引擎組態

Blusam 儲存和快取存取組態

如需如何使用秘密管理員或資料來源來設定對 Blusam 儲存和快取的存取的資訊，請參閱 the section called “AWS Blu Age 執行期組態”。

> **Note**
>
> 關於對 Blusam 儲存體的存取，使用的登入資料將指向具有相應權限的連線角色。對於 Blusam 引擎能夠如預期般運作，連線角色必須具有下列權限：

- 連線至資料庫
- 建立/刪除/更改/截斷資料表和檢視
- 選取/插入/刪除/更新資料表和檢視中的資料列
- 執行函數或程序

Blusam 引擎組態

停用 Blusam 支援

首先，讓我們提到，可以透過將 `bluesam.disabled` 屬性設定為 來完全停用 Blusam 支援`true`。應用程式啟動時，伺服器日誌中會顯示一則資訊訊息，提醒停用 Blusam：

```
BLUESAM is disabled. No operations allowed.
```

在這種情況下，不需要進一步設定 Blusam，而且任何嘗試使用 Blusam 相關功能 （以程式設計方式或透過 REST 呼叫） 的嘗試都會在 Java 程式碼執行UnsupportedOperationException中引發 ，並顯示有關 Blusam 停用的相關說明訊息。

Blusam 引擎屬性

Blusam 引擎組態屬性會在 bluesam 金鑰字首下重新分組：

強制性屬性

- `cache`：使用所選的快取實作進行評價。有效的 值如下：
  - `ehcache`：適用於本機內嵌 ehcache 使用。請參閱上述相關的使用案例限制。
  - `redis`：用於共用遠端 redis 快取用量。這是 AWS Mainframe Modernization 受管使用案例的偏好選項。
  - none：停用儲存快取
- `persistence`：以 pgsql 值 (PostgreSQL 引擎：最低版本 10.0 – 建議版本 >=14.0
- 資料來源參考： `<persistence engine>.dataSource` 將指向連接至 Blusam 儲存體的 dataSource 定義，此定義在組態檔案中的其他位置定義。通常命名為 `bluesamDs` 。

> **Note**
>
> 每當 Redis 用作快取機制時，無論是用於資料或鎖定 （請參閱下文），都必須設定對 Redis 執行個體的存取。如需詳細資訊，請參閱the section called “可用的 Redis 快取屬性”。

可選屬性

Blusam Locks：屬性字首為 `locks`

- `cache`：只有可用值為 `redis` ，以指定將使用 redis 型鎖定機制 （當 blusam 儲存快取也以 redis 型時，將使用）。如果 屬性遺失或未設定為 `redis` ，則會改用預設記憶體內鎖定機制。
- `lockTimeOut`：正長整數值，在嘗試鎖定已鎖定的元素標記為失敗之前，以毫秒表示的逾時。預設為 `500` 。
- `locksDeadTime`：正長整數值，代表以毫秒表示的最大時間，應用程式可以保留鎖定。鎖定會自動標示為過期，並在經過時間後釋放。預設為 `1000` ；
- `locksCheck`：字串，用於定義目前 blusam 鎖定管理員使用的鎖定檢查策略，關於過期的鎖定移除。在以下值中挑選：

- off：不會執行任何檢查。不建議，因為可能會發生無效鎖定。
- reboot：檢查會在重新啟動或應用程式開始時間執行。屆時會釋出所有過期的鎖定。此為預設值。
- timeout：檢查會在重新啟動或應用程式開始時間，或在嘗試鎖定資料集期間逾時過期時執行。過期的鎖定會立即釋放。

落後的機制：屬性的字首為write-behind索引鍵：

- enabled： true （預設和建議值）或 false ，以啟用或停用幕後寫入機制。停用 機制將大幅影響寫入效能，而且不建議這麼做。
- maxDelay：要觸發的執行緒的最大持續時間。預設為 "1s"（一秒）。除非特定條件需要調校此值，否則保留預設值通常是個好主意。在任何情況下，該值都應該保持低 (3 秒內）。延遲字串的格式為： <time unit>要在下列值中挑選<integer value><optional whitespace><time unit>：
  - "ns"：奈秒
  - "µs"：微秒
  - "ms"：毫秒
  - "s"： 秒
- threads：專用寫入落後執行緒的數量。預設為 5 。您需要根據執行 Blusam 引擎之主機的運算能力來調整此值。這與使用更高值無關，希望提高效能，因為限制因素將成為儲存 RDBMS 處理許多並行批次查詢的能力。建議值通常在 4-8 的範圍內。
- batchSize：正整數，代表批次中要分派以進行大量處理到執行緒的記錄數目上限。其值必須介於 1 到 32767 之間。預設為 10000 。使用 1做為值會破壞機制的目的，以避免使用原子更新查詢；使用的適當最小值大約是 1000 。

內嵌 EhCache 微調：屬性的字首為ehcache索引鍵：

- resource-pool:
  - size：內嵌快取的配置記憶體大小，以字串表示。預設為 "1024MB"(1 GB)。針對託管 Blusam 引擎之機器的可用記憶體，以及應用程式正在使用的資料集大小進行調整。大小字串的格式為：<memory-unit>要在下列值中挑選<integer value><optional whitespace><memory unit>：
    - B：位元組
    - KB：KB

  - MB：MB
  - GB： GB
  - TB：TB
- heap： true或 false ，指出快取是否會耗用 JVM 堆積記憶體。預設為 true（快取效能的最快選項，但快取儲存會耗用 JVM 堆積上 RAM 記憶體的記憶體）。將此屬性設定為 false 會切換到離區記憶體，因為與 JVM 堆積進行必要的交換，這會較慢。
- timeToLiveMillis：快取項目在被視為過期和移除之前保留在快取中的持續時間 （以毫秒為單位）。如果未指定此屬性，快取項目預設不會自動過期。

多結構描述組態屬性

- multiSchema：false （預設值） 或 true，以停用或啟用 Blusam 的多結構描述模式 - 可用的起始版本 4.4.0。
- pgsql:
  - schemas：應用程式將在多結構描述模式中用於 Blusam 的結構描述名稱清單。
  - fallbackSchema：用於多結構描述模式的備用結構描述名稱。如果在目前的結構描述內容中找不到資料集，則此結構描述將用於該資料集上的藍毛相關操作。

範例組態程式碼片段：

```
dataSource:
  bluesamDs:
    driver-class-name: org.postgresql.Driver
    ...
    ...
bluesam:
  locks:
    lockTimeOut: 700
  cache: ehcache
  persistence: pgsql
  ehcache:
    resource-pool:
      size: 8GB
  write-behind:
    enabled: true
    threads: 8
    batchsize: 5000
  pgsql:
```

```
    dataSource : bluesamDs
```

範例組態程式碼片段（針對 Blusam 啟用多結構描述模式）：

```
dataSource:
  bluesamDs:
    driver-class-name: org.postgresql.Driver
    ...
    ...
bluesam:
  locks:
    lockTimeOut: 700
  cache: ehcache
  persistence: pgsql
  ehcache:
    resource-pool:
      size: 8GB
  write-behind:
    enabled: true
    threads: 8
    batchsize: 5000
  multiSchema: true
  pgsql:
    dataSource : bluesamDs
    schemas:
      - "schema1"
      - "schema2"
      - "schema3"
    fallbackSchema: schema3
```

> **Note**
>
> 藍毛中繼資料結構描述，包括多結構描述模式`application-main.yml`檔案中列出的結構描述，如果不存在且使用者具有足夠的權限，則會在藍毛資料庫中建立。

Blusam 管理主控台

Blusam 管理主控台 (BAC) 是一種 Web 應用程式，用於管理 Blusam 儲存。如需 BAC 的相關資訊，請參閱 the section called “Blusam 管理主控台”。

附錄

一般資料集中繼資料屬性

一般資料集中繼資料序列化屬性清單：

- 名稱（資料集的）
- 類型 (KSDS、 LargeKSDS、ESDS、 LargeESDS 或 RRDS)
- 快取暖機旗標（是否應在伺服器啟動時預先載入快取中的資料集）
- 壓縮用量旗標（以壓縮或原始格式存放記錄）
- 建立日期
- 上次修改日期
- 固定長度記錄旗標（資料集記錄是否都具有相同的長度）
- 記錄長度 - 僅對固定記錄長度有意義
- 頁面大小（用於自訂用於在需要時預先載入快取的分頁 sql 查詢）
- 大小（資料集的大小 - 記錄的累積長度）
- 上次偏移（偏移，即新增至資料集之最新記錄的 RBA)
- 下一個位移（下一個可用位移用於將新記錄新增至資料集）
- 如果有意義的，定義資料集所使用的金鑰；每個金鑰都由其類型（備用金鑰集合的主要或部分）和三個屬性定義：
  - 位移：鍵值開始位元組記錄中的位置；
  - length ：金鑰值的位元組長度。因此，索引鍵值是位元組陣列，這是記錄的子集，從位置 開始`key offset`到結束`key offset + length - 1`；
  - 允許的重複旗標：金鑰是否接受重複（設定為 true 以允許重複）。

估計指定資料集的記憶體使用量

對於中小型資料集，中繼資料（各種索引鍵的大小和索引）將完全載入記憶體。為託管用於執行現代化應用程式的伺服器之機器配置適當的資源，需要找出 Blusam 資料集所引發的記憶體耗用，特別是中繼資料。本節提供相關運算子的實際答案。

指定的公式僅適用於 Blusam 中小型資料集，而不是使用「大型」儲存策略。

Blusam 資料集中繼資料

對於 Blusam 資料集，中繼資料分為兩個部分：

- 核心中繼資料 ：保留有關資料集的全域資訊。與內部中繼資料相比，這可視為可忽略的記憶體使用量。
- 內部中繼資料： 保留記錄大小和索引鍵的資訊；當資料集不是空的時，這是載入託管現代化應用程式的應用程式伺服器時耗用記憶體的原因。以下各節詳細說明耗用記憶體如何隨著記錄數量而成長。

計算內部中繼資料使用量

記錄大小映射

首先，內部中繼資料會儲存映射，以保留每個記錄的大小（以整數表示），並指定其 RBA（相對位元組地址，以長數字儲存）。

該資料結構的記憶體使用量，以位元組為單位： `80 * number of records`

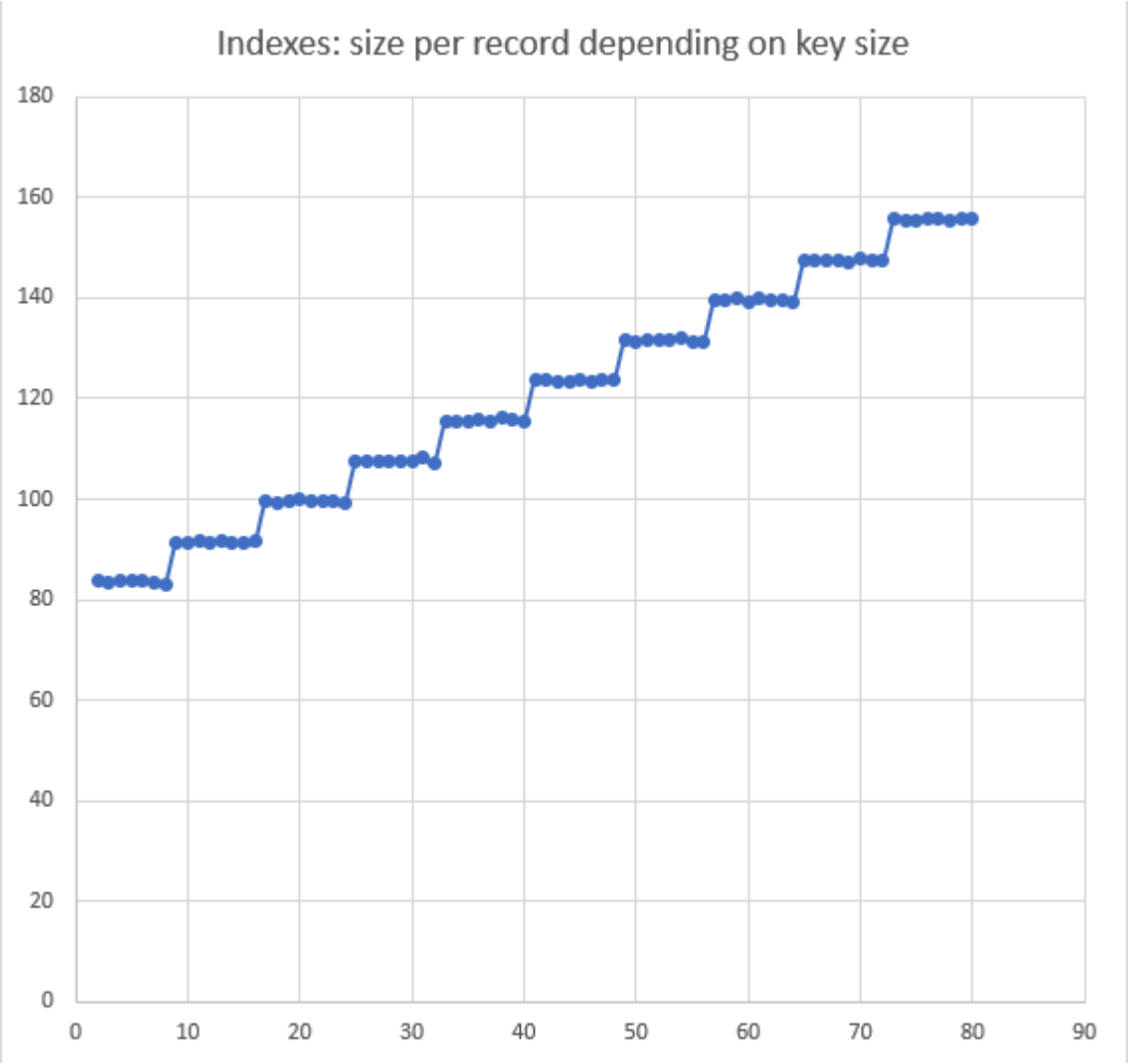
這適用於所有資料集類型。

索引

關於 KSDS 的主索引鍵或 ESDS 和 KSDS 上的備用索引鍵，用量的計算取決於兩個因素：

- 資料集中的記錄數目；
- 金鑰的大小，以位元組為單位。

下圖顯示根據金鑰大小 (x 軸） 的每筆記錄金鑰索引大小 (y 軸）。



評估資料集之指定索引鍵的佔用量的對應公式為：

```
index footprint = number of records * ( 83 + 8 (key length / 8))
```

其中 '/' 代表整數分割。

範例：

- 資料集 1：
  - 記錄數目 = 459 996
  - 金鑰長度 = 15，因此 （金鑰長度 / 8) = 1
  - 索引使用量 = 459 996 * (83 + (8*1)) = 41 859 636 位元組 (= 約 39 MB)
- 資料集 2：

- 記錄數目 = 13 095 783
- 金鑰長度 = 18，因此 （ 金鑰長度 / 8) = 2
- 索引使用量 = 13 095 783 * (83 + (8*2)) = 1 296 482 517 位元組 （ 大約 = 1.2 GB)

指定資料集的總佔用量是所有索引鍵的所有佔用量和記錄大小映射的佔用量總和。

例如，以只有單一金鑰的範例資料集 2 為例，全域足跡為：

- 記錄大小映射：13 095 783 * 80 = 1 047 662 640 位元組
- 索引鍵：1 296 482 517 位元組 （請參閱上述）
- 總使用量 = 2 344 145 157 位元組 (= 2.18 GB 左右)

公用程式 Web 應用程式中的可用程式

公用程式 Web 應用程式支援各種舊版平台公用程式，例如 IDCAMS、INFUTILB、SORT 等。若要設定對應用程式的存取，請參閱 設定受管應用程式的公用程式存取權。

程式清單

- JCLBCICS 公用程式 - 批次用於將 bluesam 資料集狀態設定為開啟/啟用或關閉/停用。

JCLBCICS 公用程式

JCLBCICS 是一種 JCL 公用程式，旨在將 bluesam 資料集設定為開啟/啟用或關閉/停用。開啟/啟用狀態會封鎖從批次程式存取資料集，而關閉/停用狀態會使資料集無法存取 JICS 線上服務。

用途

- JCLBCICS 會根據 DD 名稱上的 groovy 組態，變更 Jics FILE_TABLE 資料表中的 STATUS 資料欄和 Bluesam BLUESAM_STATUS 資料表中的 OPEN_STATUS 資料欄。

```
.open(ddName) -> ENABLED in Jics FILE_TABLE table, OPEN in Bluesam BLUESAM_STATUS
 table
.close(ddName) -> DISABLED in Jics FILE_TABLE table, CLOSED in Bluesam BLUESAM_STATUS
 table
```

- DD 名稱大小可在組態檔案中全域`application-utility-pgm.yml`設定。

```
jclbcics.ddname.size: 7
```

- 全域 DD 名稱大小可以在個別步驟中覆寫，方法是在 groovy 中提供具有下列行的覆寫大小，然後使用 stepParams 作為該步驟的參數。

```
TreeMap stepMapTransfo = [:]
Map stepParams = ["MapTransfo":stepMapTransfo]
stepParams["MapTransfo"]["JCLBCICS_OVERRIDDEN_SIZE"] = '7'
...
.withParameters(stepParams)
.runProgram("JCLBCICS")
```

- 設定 DD 名稱大小時，最大有效 DD 名稱大小為 8。
- 如果 ddName 長度大於提供的 DD 名稱大小，則會從結尾截斷以符合 DD 名稱大小。
- 如果 * （星號） 附加到 ddName ddName 結尾或 ddName 長度小於 8，則 ddName 中支援萬用字元。

```
.open("DTSNAME*")
```

範例程式碼

```
// DD name with overridden size of 7 bytes
 def stepSTEP007(Object shell, Map params, Map programResults) {
   shell.with {
     if (checkValidProgramResults(programResults)) {
       TreeMap stepMapTransfo = [:]
       Map stepParams = ["MapTransfo":stepMapTransfo]
       stepParams["MapTransfo"]["JCLBCICS_OVERRIDDEN_SIZE"] = '7'
       return execStep("STEP007", "JCLBCICS", programResults, {
         mpr
           .withDatasetsConfiguration(new DatasetsConfiguration()
           .close("DTSNAME"))
           .withParameters(stepParams)
           .runProgram("JCLBCICS")
        })
     }
   }
}
```

AWS Blu Age Blusam 管理主控台

Blusam 管理主控台 (BAC) 是處理 Blusam 資料集的安全 Web 應用程式。本指南涵蓋 BAC 使用者介面。如需透過 REST 端點的遠端管理，請參閱 the section called “Blusam 應用程式主控台 REST 端點”。

主題

- 部署 BAC
- 使用 BAC
- LISTCAT JSON 格式

部署 BAC

BAC 以安全單一 Web 應用程式的形式提供，使用 Web 封存格式 (.war)。它旨在與 BluAge Gapwalk-Application 一起部署在 Apache Tomcat 應用程式伺服器中，但也可以部署為獨立應用程式。如果存在，BAC 會從 Gapwalk-Application 組態繼承對 Blusam 儲存體的存取權。

BAC 有自己的專用組態檔案，名為 `application-bac.yml`。如需組態詳細資訊，請參閱 the section called “BAC 專用組態檔案”。

BAC 是安全的。如需安全組態的詳細資訊，請參閱 the section called “設定 BAC 的安全性”。

BAC 專用組態檔案

獨立部署：如果 BAC 單獨部署為 Gapwalk-Application，則必須在 application-bac.yml 組態檔案中設定與 Blusam 儲存體的連線。

用於瀏覽資料集記錄的資料集組態預設值必須在組態檔案中設定。請參閱 the section called “從資料集瀏覽記錄”。記錄瀏覽頁面可以使用選用的遮罩機制，以便在記錄的內容上顯示結構化檢視。某些屬性會影響使用遮罩時的記錄檢視。

下列可設定屬性必須在組態檔案中設定。BAC 應用程式不會假設這些屬性的任何預設值。

金鑰	Type	描述
`bac.crud.limit`	integer	正整數值，表示瀏覽記錄時傳回的記錄數目上限。使用 `0`表示無限制。建議值：`10`（然後

金鑰	Type	描述
		調整瀏覽頁面上資料集的值資料集，以符合您的需求）。
bac.crud.encoding	string	預設字元集名稱，用於將記錄位元組解碼為英數字元內容。提供的字元集名稱必須與 Java 相容（請參閱 Java 文件以取得支援的字元集）。建議值：用於舊版平台上的舊版字元集，其中資料集來自該平台；這在大多數情況下都會是 EBCDIC 變體。
bac.crud.initCharacter	string	用來初始化資料項目的預設字元（位元組）。可使用兩個特殊值："LOW-VALUE" 、0x00 位元組（建議值）和 "HI-VALUE" 、0xFF 位元組。套用遮罩時使用。
bac.crud.defaultCharacter	string	預設字元（位元組），做為一個字元字串，用於填補記錄（右側）。建議值：" "（空格）。套用遮罩時使用。
bac.crud.blankCharacter	string	預設字元（位元組）做為一個字元字串，用於表示記錄中的空白。建議值：" "（空格）。套用遮罩時使用。
bac.crud.strictZoned	boolean	指出用於記錄的區域模式的旗標。如果 true，則會使用嚴格區域模式；如果 false，則會使用修改的區域模式。建議值：true。套用遮罩時使用。

金鑰	Type	描述
bac.crud.decimalSeparator	string	在數值編輯欄位中用作小數分隔符號的字元（套用遮罩時使用）。
bac.crud.currencySign	string	套用格式時（套用遮罩時使用），預設字元做為一個字元字串，用於表示數字編輯欄位中的貨幣。
bac.crud.pictureCurrencySign	string	預設字元作為一個字元字串，用於在數值編輯欄位圖片中表示貨幣（套用遮罩時使用）。

下列範例是組態檔案程式碼片段。

```
bac.crud.limit: 10
bac.crud.encoding: ascii
bac.crud.initCharacter: "LOW-VALUE"
bac.crud.defaultCharacter: " "
bac.crud.blankCharacter: " "
bac.crud.strictZoned: true
bac.crud.decimalSeparator: "."
bac.crud.currencySign: "$"
bac.crud.pictureCurrencySign: "$"
```

設定 BAC 的安全性

設定 BAC 的安全性取決於本文件頁面中詳述的機制。身分驗證機制為 OAuth2，並提供 Amazon Cognito 或 Keycloak 的組態詳細資訊。

雖然可以套用一般設定，但此處需要詳細說明 BAC 的一些詳細資訊。對 BAC 功能的存取會使用角色型政策進行保護，並依賴下列角色。

- ROLE_USER：
  - 基本使用者角色
  - 不允許匯入、匯出、建立或刪除資料集
  - 無法控制快取政策

  - 不允許管理功能
- ROLE_ADMIN：
  - 繼承 ROLE_USER 許可
  - 允許的所有資料集操作
  - 允許快取政策管理

安裝遮罩

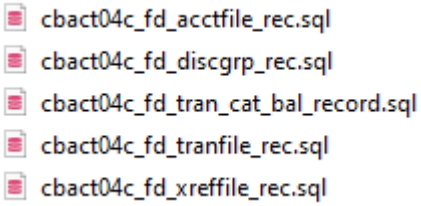
在 Blusam 儲存中，資料集記錄會存放在資料庫中的位元組陣列欄中，以因應多樣化和效能考量。根據應用程式觀點，使用業務記錄的欄位存取結構化檢視是 BAC 的便利功能。這依賴於 BluAge 驅動的現代化過程中產生的 SQL 遮罩。

若要產生 SQL 遮罩，請務必將 BluInsights 轉換中心組態中的相關選項 (`export.SQL.masks`) 設為 true：



遮罩是現代化成品的一部分，可以從 BluInsights 下載給定專案。它們是由現代化程式整理的 SQL 指令碼，為資料集記錄提供適用的觀點。

例如，使用 AWS CardDemo 範例應用程式，您可以在此應用程式的現代化結果下載的成品中找到下列 CBACT04C.cbl 程式的 SQL 遮罩：



每個 SQL 遮罩名稱都是程式名稱和程式中指定資料集的記錄結構名稱的串連。

例如，查看 【CBACT04C.cbl 程式，指定的檔案控制項目：

```
    FILE-CONTROL.
        SELECT TCATBAL-FILE ASSIGN TO TCATBALF
               ORGANIZATION IS INDEXED
               ACCESS MODE  IS SEQUENTIAL
               RECORD KEY   IS FD-TRAN-CAT-KEY
               FILE STATUS  IS TCATBALF-STATUS.
```

與指定的 FD 記錄定義相關聯

```
       FILE SECTION.
       FD  TCATBAL-FILE.
       01  FD-TRAN-CAT-BAL-RECORD.
           05 FD-TRAN-CAT-KEY.
              10 FD-TRANCAT-ACCT-ID             PIC 9(11).
              10 FD-TRANCAT-TYPE-CD             PIC X(02).
              10 FD-TRANCAT-CD                  PIC 9(04).
           05 FD-FD-TRAN-CAT-DATA               PIC X(33).
```

名為 的相符 SQL 遮罩cbact04c_fd_tran_cat_bal_record.SQL是遮罩，可在名為 的 FD 記錄上提供程式 CBACT04C.cbl 的觀點FD-TRAN-CAT-BAL-RECORD。

其內容為：

```
-- Generated by Blu Age Velocity
-- Mask : cbact04c_fd_tran_cat_bal_record

INSERT INTO mask (name, length) VALUES ('cbact04c_fd_tran_cat_bal_record', 50);
  INSERT INTO mask_item (name, c_offset, length, skip, type, options, mask_fk) VALUES
 ('fd_trancat_acct_id', 1, 11, false, 'zoned', 'integerSize=11!fractionalSize=0!
signed=false', (SELECT MAX(id) FROM mask));
  INSERT INTO mask_item (name, c_offset, length, skip, type, options, mask_fk) VALUES
 ('fd_trancat_type_cd', 12, 2, false, 'alphanumeric', 'length=2', (SELECT MAX(id) FROM
 mask));
  INSERT INTO mask_item (name, c_offset, length, skip, type, options, mask_fk)
 VALUES ('fd_trancat_cd', 14, 4, false, 'zoned', 'integerSize=4!fractionalSize=0!
signed=false', (SELECT MAX(id) FROM mask));
```

```
 INSERT INTO mask_item (name, c_offset, length, skip, type, options, mask_fk) VALUES
('fd_fd_tran_cat_data', 18, 33, false, 'alphanumeric', 'length=33', (SELECT MAX(id)
FROM mask));
```

遮罩會使用兩個資料表存放在 Blusam 儲存體中：

- 遮罩：用於識別遮罩。質量資料表的資料欄如下：
  - 名稱：用於存放遮罩識別（用作主索引鍵，因此必須是唯一的）
  - 長度：記錄遮罩的大小，以位元組為單位
- mask_item：用於存放遮罩詳細資訊。FD 記錄定義中的每個基本欄位都會在 mask_item 資料表中產生一列，其中包含如何解譯指定記錄部分的詳細資訊。mask_item 資料表的資料欄為：
  - name：記錄欄位的名稱，根據基本名稱，使用小寫並以底線取代破折號
  - c_offset：記錄子部分以 1 為基礎的偏移，用於欄位內容
  - length：記錄子部分的位元組長度，用於欄位內容
  - 略過：在檢視簡報中指出是否應略過指定記錄部分
  - 類型：欄位類型（根據其舊版圖片子句）
  - 選項：其他類型選項 -- type-dependant
  - mask_fk：參考要將此項目附加至 的遮罩識別符

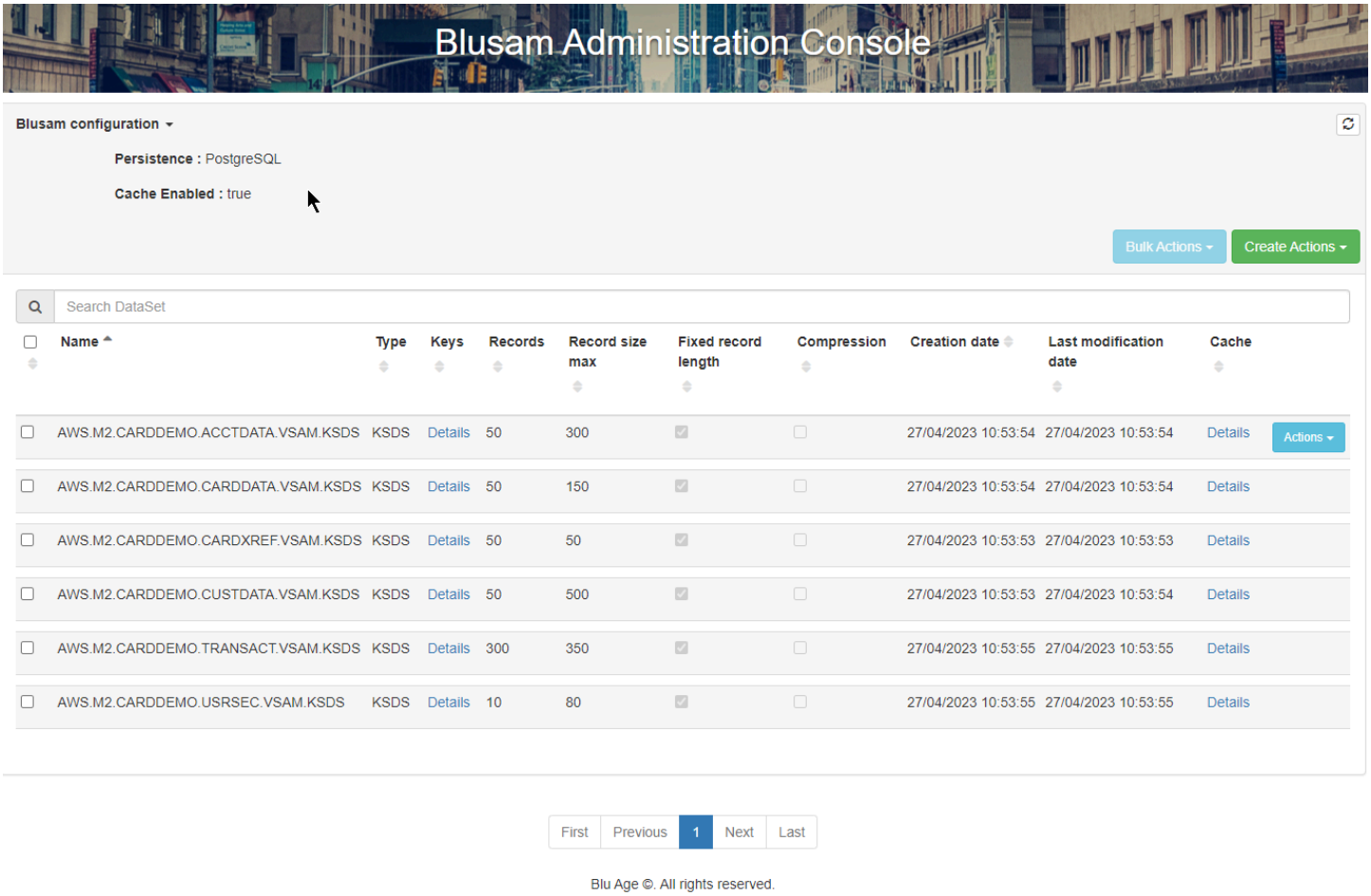
注意下列事項：

- SQL 遮罩代表程式對資料集記錄的觀點：數個程式對指定資料集可能有不同的觀點；僅安裝您認為與您目的相關的遮罩。
- SQL 遮罩也可以代表根據 WORKING STORAGE 區段 01 資料結構的程式觀點，而不只是從 FD 記錄。SQL 遮罩會根據其性質組織成子資料夾：
  - FD 記錄型遮罩將位於名為 的子資料夾中 `file`
  - 01 資料結構型遮罩將位於名為 的子資料夾中 `working`

  雖然 FD 記錄定義一律符合資料集的記錄內容，但 01 資料結構可能不會對齊，或可能僅代表資料集記錄中的子集。使用它們之前，請先檢查程式碼並了解可能的缺點。

使用 BAC

由於 BAC 是受保護的，並提供根據使用者角色使用 功能的許可，因此存取應用程式的第一步是自行驗證。在驗證步驟之後，系統會將您重新導向至 首頁。首頁會顯示 Blusam 儲存中找到的資料集分頁清單：



若要返回含有資料集清單的首頁，請選擇應用程式任何頁面左上角的 Blu Age 標誌。下圖顯示 標誌。

可摺疊標頭，標示為「BluSam 組態」，包含使用之 BluSam 儲存組態的相關資訊：

- `Persistence`：持久性儲存引擎 (PostgreSQL)
- `Cache Enabled`：是否啟用儲存快取

在 標頭的右側，有兩個下拉式清單，每個清單都會列出與資料集相關的操作：

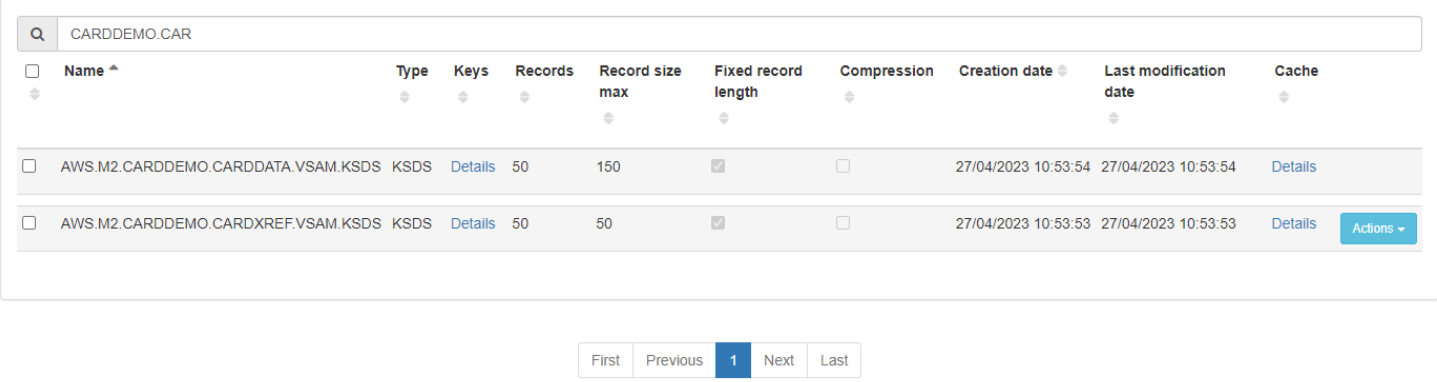
- 大量動作

- 建立動作

若要了解這些清單的詳細內容，請參閱 the section called “現有的資料集操作”。

未進行任何資料集選取時，會停用大量動作按鈕。

您可以使用搜尋欄位，根據資料集名稱將清單存檔：



以下分頁清單顯示每個資料表資料列一個資料集，其中包含下列資料欄：

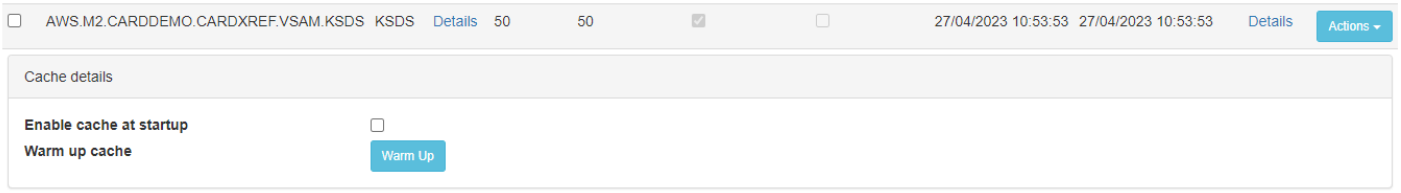
- 選取核取方塊：選取目前資料集的核取方塊。
- 名稱：資料集的名稱。
- 類型：資料集的類型，下列其中一項：
  - KSDS
  - ESDS
  - RRDS
- 金鑰：顯示或隱藏金鑰詳細資訊的連結 （如果有的話）。例如，指定的 KSDS 具有強制性主索引鍵和一個替代索引鍵。



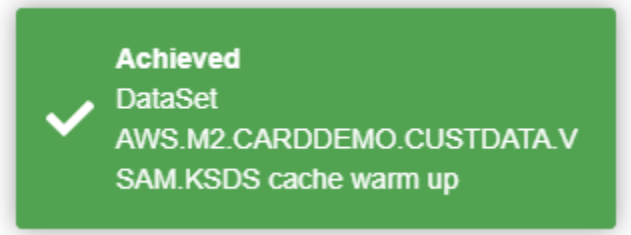
每個金鑰有一個資料列，其中包含下列資料欄。所有欄位都無法編輯。

- 金鑰本質：主金鑰或其他金鑰
- 名稱：金鑰的名稱

  - 唯一：金鑰是否接受重複的項目
  - 位移：記錄中索引鍵開始的位移
  - 長度：記錄中金鑰部分的位元組長度
- 記錄：資料集中的記錄總數。
- 記錄大小上限：記錄的大小上限，以位元組表示。
- 固定記錄長度：指出記錄為固定長度（已選取）或可變長度（未選取）的核取方塊。
- 壓縮：指出壓縮是否套用（選取）至已儲存索引的核取方塊。
- 建立日期：在 Blusam 儲存中建立資料集的日期。
- 上次修改日期：在 Blusam 儲存中上次更新資料集的日期。
- 快取：顯示或隱藏套用至此資料集之快取策略詳細資訊的連結。



  - 在啟動時啟用快取：指定此資料集啟動快取策略的核取方塊。如果選取，資料集將在啟動時載入快取。
  - 暖機快取：將指定資料集載入快取的按鈕，立即開始（但補充快取需要一些時間，取決於資料集大小和金鑰數量）。將資料集載入快取後，會出現類似下列的通知。

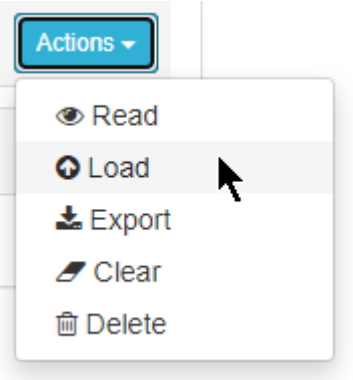


- 動作：可能資料集操作的下拉式清單。如需詳細資訊，請參閱the section called “現有的資料集操作”。

在頁面底部，有一個一般分頁導覽小工具，用於瀏覽資料集清單的頁面。

現有的資料集操作

對於分頁清單中的每個資料集，有一個動作下拉式清單，其中包含下列內容：



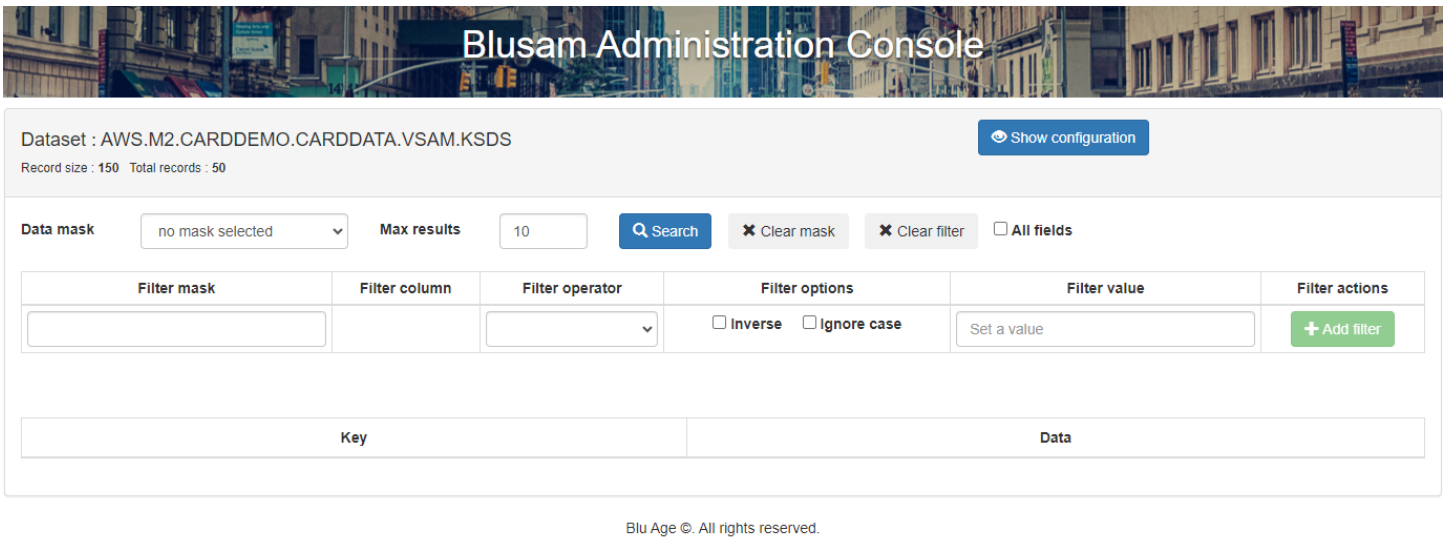
清單中的每個項目都是作用中連結，可讓您對資料集執行指定的動作：

- 讀取：瀏覽資料集中的記錄
- 載入：從舊版資料集檔案匯入記錄
- 匯出：將記錄匯出至一般檔案（與舊版系統相容）
- 清除：從資料集移除所有記錄
- 刪除：從儲存體移除資料集

以下各節提供每個動作的詳細資訊。

從資料集瀏覽記錄

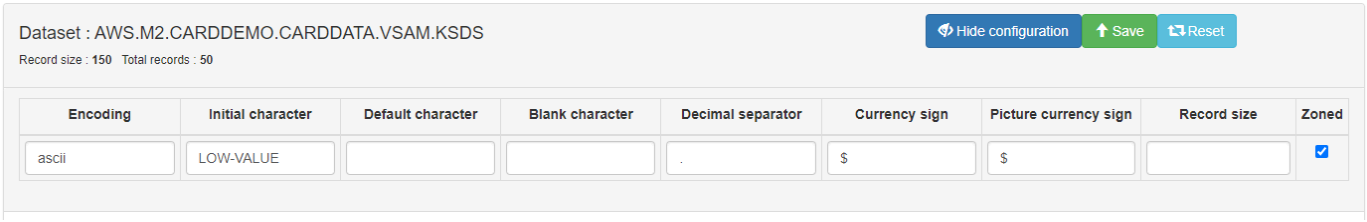
當您為指定的資料集選擇讀取動作時，您會看到以下頁面。



頁面是由以下項目組成：

- 標頭，具有：

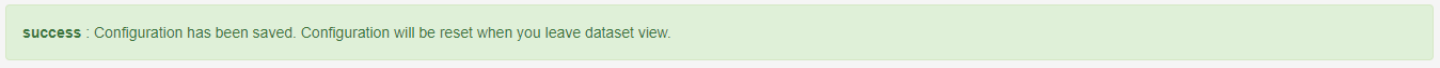
- 資料集：資料集名稱
- 記錄大小：固定記錄長度，以位元組表示
- 總記錄：為此資料集存放的記錄總數
- 顯示組態按鈕 （右側）：顯示/隱藏資料集組態的切換按鈕。一開始，組態會隱藏。使用 按鈕時，您會看到組態的組態，如下圖所示。



  顯示組態時，兩個新按鈕：儲存和重設，分別用於：

  - 儲存此資料集和目前工作工作階段的組態
  - 將所有欄位的組態重設為預設值。
- 可設定的屬性清單，可量身打造指定資料集的瀏覽體驗。

可設定的屬性符合 中所述的組態屬性the section called “BAC 專用組態檔案”。請參閱該區段以了解每個資料欄和適用值的意義。您可以在這裡為資料集重新定義每個值，並為工作工作階段儲存 （使用儲存按鈕）。儲存組態後，會出現類似下圖所示的橫幅。



橫幅指出當您離開目前頁面時，工作工作階段會結束。

有一個額外的可設定屬性，未記錄在組態區段中：記錄大小。這用於指定指定的記錄大小，以位元組表示，這會篩選此資料集適用的遮罩：只有總長度符合指定記錄大小的遮罩才會列在資料遮罩下拉式清單中。

從資料集擷取記錄是由搜尋按鈕觸發，使用附近的所有選項和篩選條件。

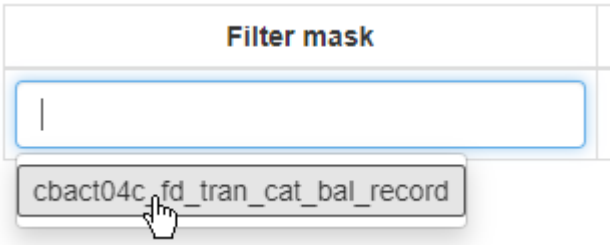
第一行選項：

- 資料遮罩下拉式清單會顯示適用的遮罩 （遵守記錄大小）。請注意，比對記錄大小不足以成為有效的適用遮罩。遮罩定義也必須與記錄內容相容。此處選擇的資料遮罩具有
- 最大結果： 限制搜尋擷取的記錄數。將資料集的無限制 （分頁） 結果設為 0。
- 搜尋按鈕：使用篩選條件和選項啟動記錄擷取
- 清除遮罩按鈕：如果有的話， 會清除已使用的遮罩，並將結果頁面切換回原始金鑰/資料呈現。

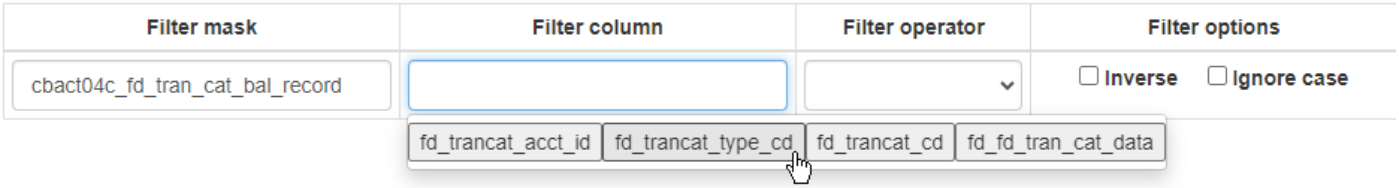
- 清除篩選條件按鈕： 將清除使用過的篩選條件（如果有的話），並相應地更新結果頁面。
- 所有欄位切換：選取時，無論如何skip = true都會顯示以 定義的遮罩項目，否則skip = true會隱藏以 遮罩項目。

下一行篩選條件：可以根據從指定遮罩套用至欄位（資料欄）的篩選條件使用情況來定義篩選條件清單，如下圖所示。

- 篩選遮罩：要從中挑選篩選資料欄的遮罩名稱。當您選擇 欄位時，會顯示適用的遮罩清單。您可以從該清單中選擇您想要的遮罩。



- 篩選條件欄：遮罩中的欄位名稱（資料欄），用於篩選記錄。當您選擇 欄位時，會出現遮罩資料欄清單。若要填入篩選條件資料欄欄位，請選擇所需的儲存格。



- 篩選條件運算子：要套用至所選資料欄的運算子。下列運算子可供使用。
  - 等於：記錄的資料欄值必須等於篩選條件值
  - 開頭為：記錄的資料欄值必須以篩選條件值開頭
  - 結尾為：記錄的資料欄值必須以篩選條件值結尾
  - 包含：記錄的資料欄值必須包含篩選條件值
- 篩選條件選項：
  - 反向：套用篩選條件運算子的反向條件；例如，「等於」被「不等於」取代；
  - 忽略大小寫：忽略篩選條件運算子的英數字元比較大小寫
- 篩選條件值：篩選條件運算子與篩選條件資料欄比較所使用的值。

設定最少數量的篩選條件項目後（至少必須設定篩選條件遮罩、篩選條件欄、篩選條件運算子和篩選條件值），即會啟用新增篩選條件按鈕，然後按一下該按鈕，在擷取的記錄上建立新的篩選條件。頂

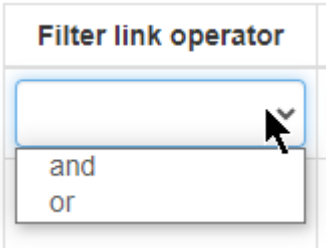
部會新增另一個空白篩選條件列，而新增的篩選條件具有移除篩選條件按鈕，可用於隱藏指定的篩選條件：

Filter link operator	Filter mask	Filter column	Filter operator	Filter options	Filter value	Filter actions
				Inverse Ignore case	Set a value	+ Add filter
	cbact04c_fd_tran_cat_bal_record	fd_trancat_type_cd	equals	Inverse Ignore case	77	− Remove filter

當您啟動搜尋時，篩選的結果會出現在分頁資料表中。

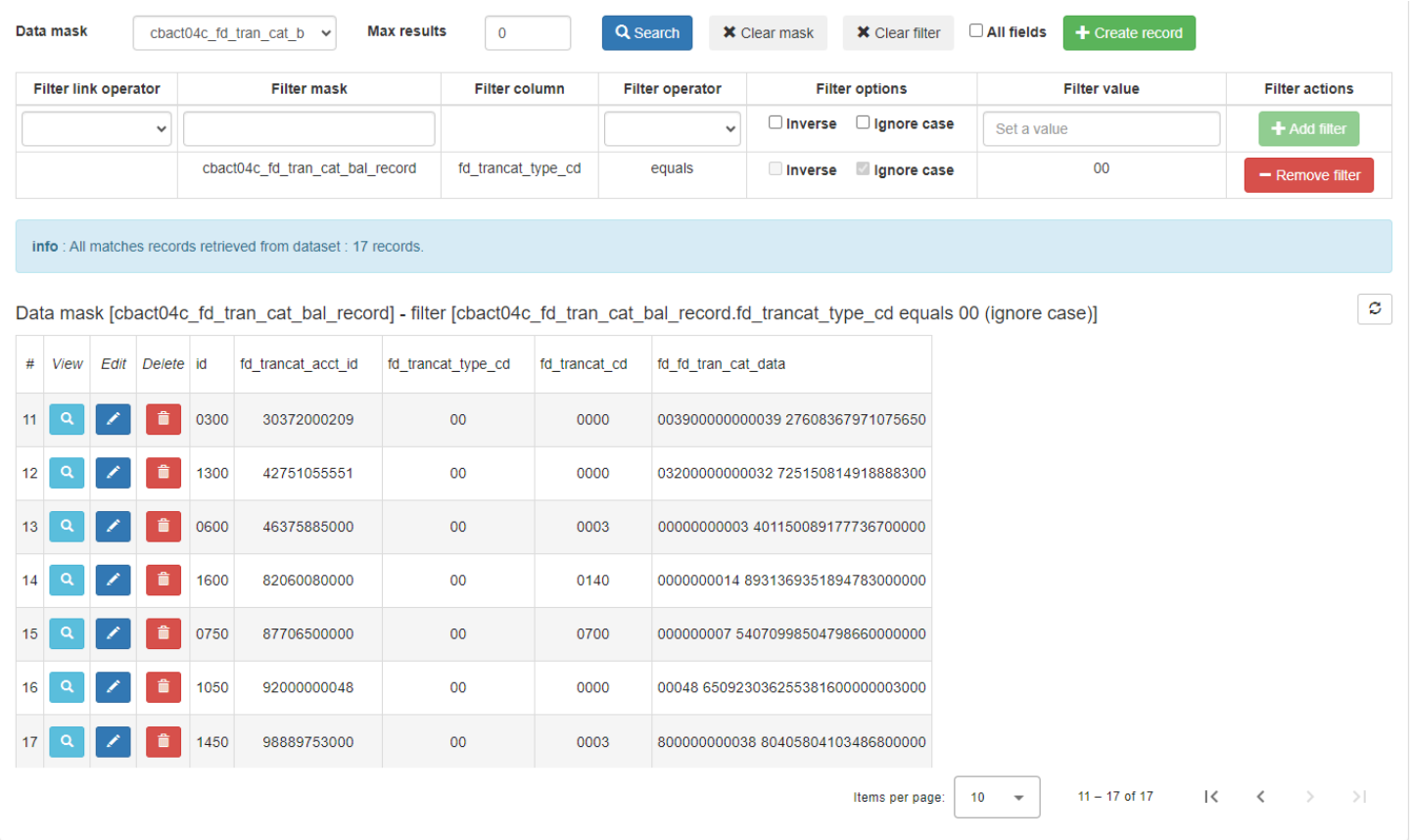
注意

- 和 或 會連結後續篩選條件。每個新的篩選條件定義都從設定連結運算子開始，如下圖所示。



- 可能沒有任何記錄符合指定的篩選條件。

否則，結果表看起來會與下圖中的結果表類似。



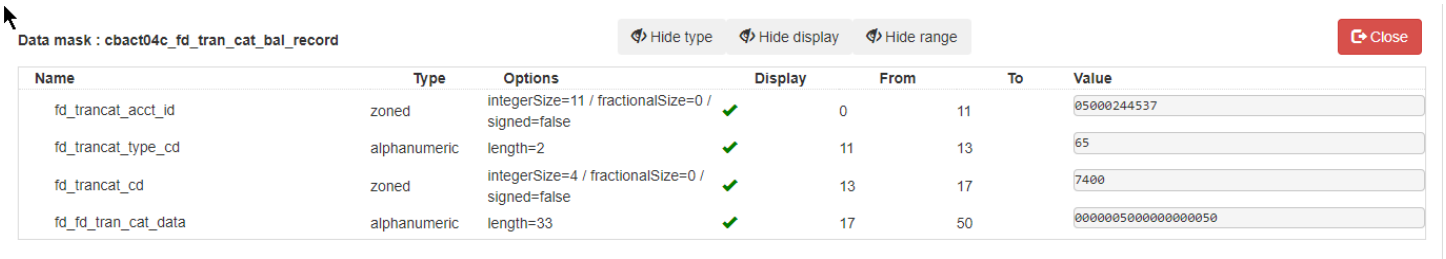
標頭表示符合篩選條件的記錄總數。在 標頭之後，您會看到以下內容。

- 提醒已使用的資料遮罩 （如果有的話） 和篩選條件。
- 重新整理按鈕，可用來觸發整個結果資料表的重新整理，其中包含來自 Blusam 儲存體的最新值 （因為可能已由其他使用者更新，例如）。

對於每個擷取的記錄，資料表都有一列，顯示將資料遮罩套用至記錄內容的結果。每個資料欄都是根據資料欄類型 （以及使用選取的編碼） 對記錄子部分的解釋。每一列左側有三個按鈕：

- 放大鏡按鈕： 會導向顯示詳細記錄內容的專用頁面
- 筆按鈕： 會導向記錄內容的專用編輯頁面：
- 垃圾桶按鈕：用於從 blusam 儲存中刪除指定的記錄

檢視記錄內容的詳細資訊：



- 三個切換按鈕，用於隱藏或顯示一些資料欄：
  - 隱藏/顯示 類型
  - 隱藏/顯示顯示旗標
  - 隱藏/顯示範圍
- 若要離開此專用頁面並返回結果資料表，請選擇關閉。
- 每一列代表資料遮罩中的資料欄，其中包含下列資料欄：
  - 名稱：資料欄的名稱
  - 類型：資料欄的類型
  - 顯示：顯示指示器；如果使用 定義相符的遮罩項目，則會顯示綠色檢查`skip = false`，否則會顯示紅色交叉
  - 從 & To：記錄子部分的 0 型範圍
  - 值：記錄子部分的解譯值，使用類型和編碼

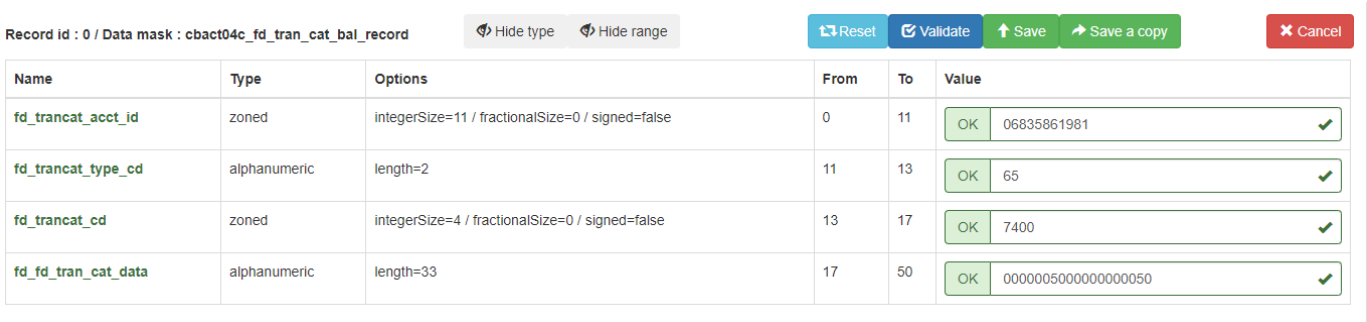
編輯記錄的內容：



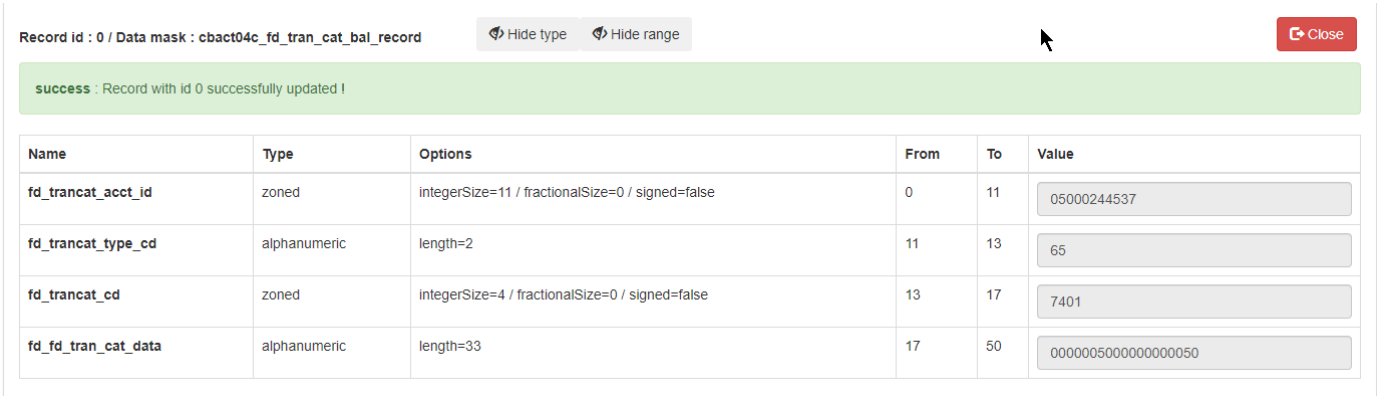
編輯頁面類似於上述檢視頁面，但遮罩項目值可編輯。三個按鈕控制更新程序：

- 重設：將可編輯的值重設為初始記錄值 （在任何版本之前）；
- 驗證： 驗證遮罩項目類型的輸入。對於每個遮罩項目，將使用視覺化標籤列印驗證結果 (OK如果驗證成功，則為 核取方塊，如果驗證失敗，則為ERROR紅色交叉，以及提供驗證失敗提示的錯誤訊息）。如果驗證成功，則會顯示兩個新按鈕：

- 儲存：嘗試將現有記錄更新為 Blusam 儲存
- 儲存副本：嘗試在 Blusam 儲存中建立新的記錄



- 如果成功將記錄儲存至儲存體，則會顯示訊息，且頁面會切換至唯讀模式（無法再編輯遮罩項目值）：

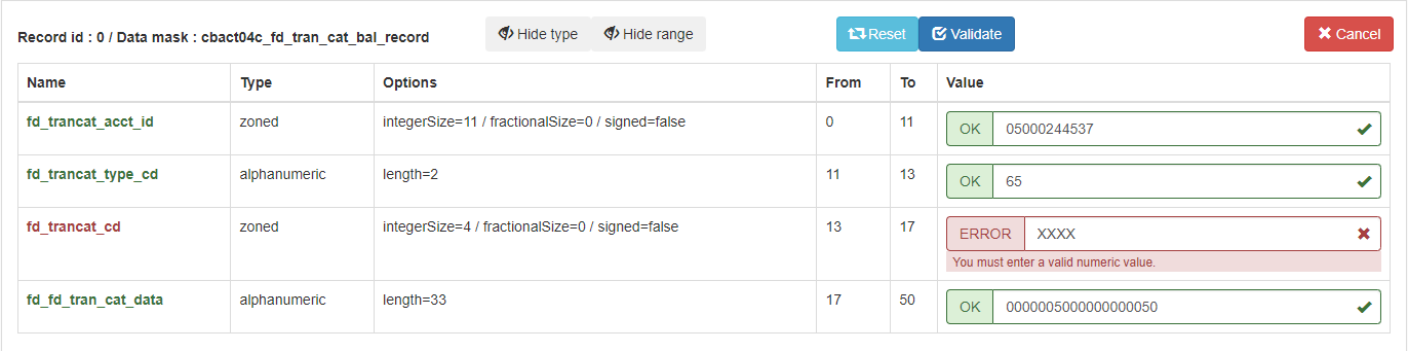


- 如果記錄因任何原因持續儲存失敗，則會顯示紅色錯誤訊息，並提供失敗原因。最常見的失敗案例是，儲存記錄會導致金鑰損毀（無效或重複的金鑰）。如需圖例，請參閱下列備註。
- 若要結束，請選擇關閉按鈕。

- 取消：結束編輯工作階段、關閉頁面，並帶您返回記錄清單頁面。

請注意：

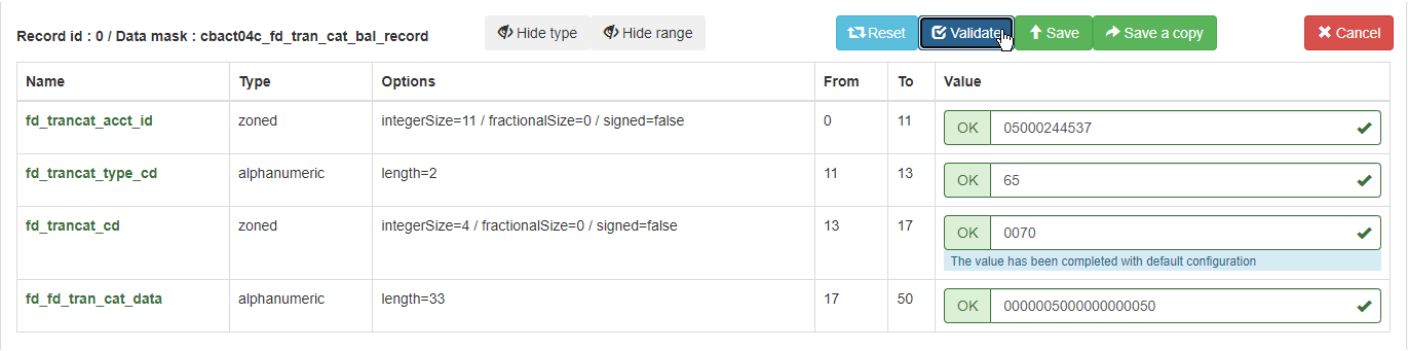
- 驗證機制只會檢查遮罩項目值是否與遮罩項目類型正式相容。例如，請參閱對數值遮罩項目的失敗驗證：



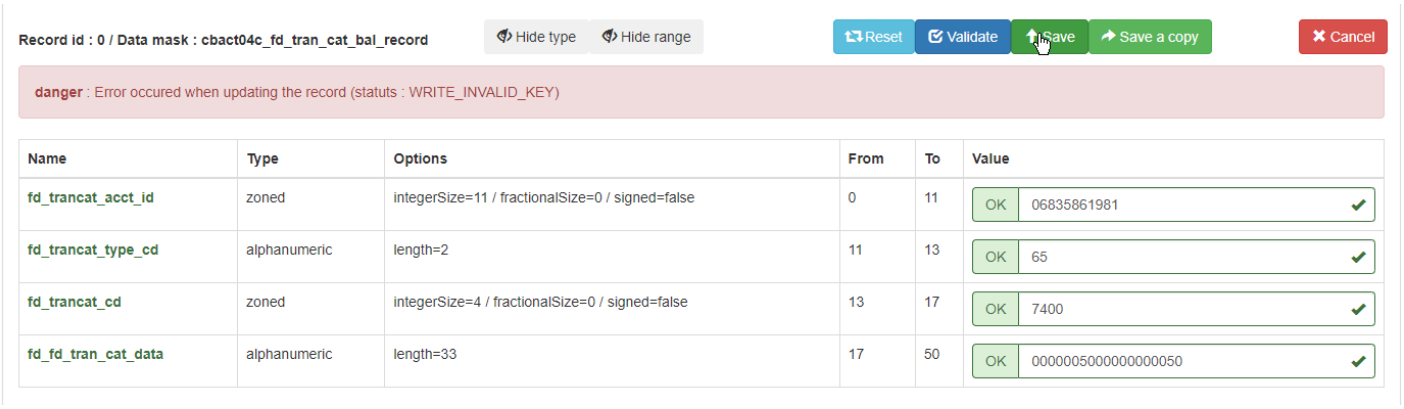
- 驗證機制可能會嘗試自動更正無效輸入，以藍色顯示資訊訊息，指出已根據其類型自動更正值。例如，輸入 7XX0 做為數值`fd_trncat_cd`遮罩項目中的數值：



呼叫驗證會導致下列情況：

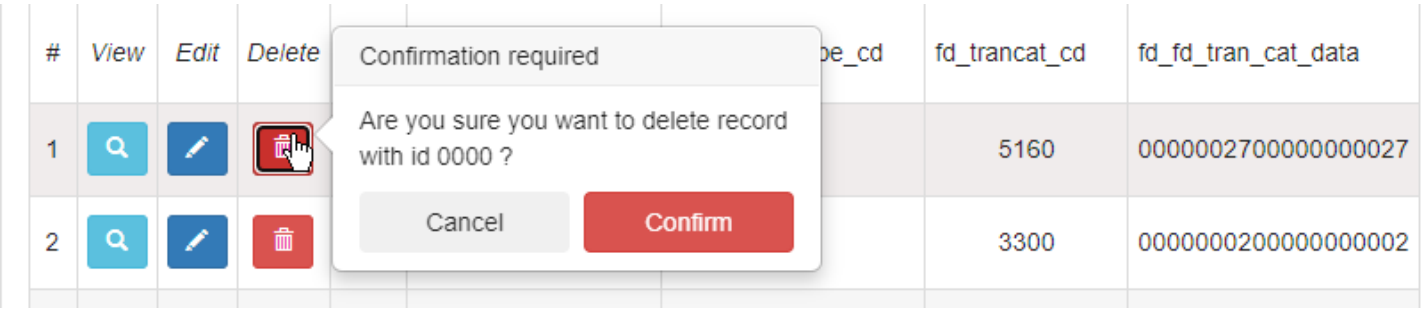


- 驗證機制不會檢查指定值在金鑰完整性方面是否有效（如果指定資料集涉及任何唯一金鑰）。例如，儘管驗證成功，如果提供的值導致無效或重複的金鑰情況，持久性將會失敗，並顯示錯誤訊息：



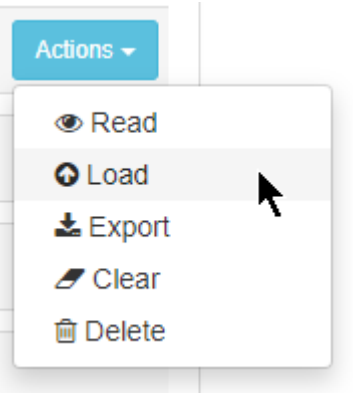
刪除記錄：

若要刪除記錄，請選擇垃圾桶按鈕：

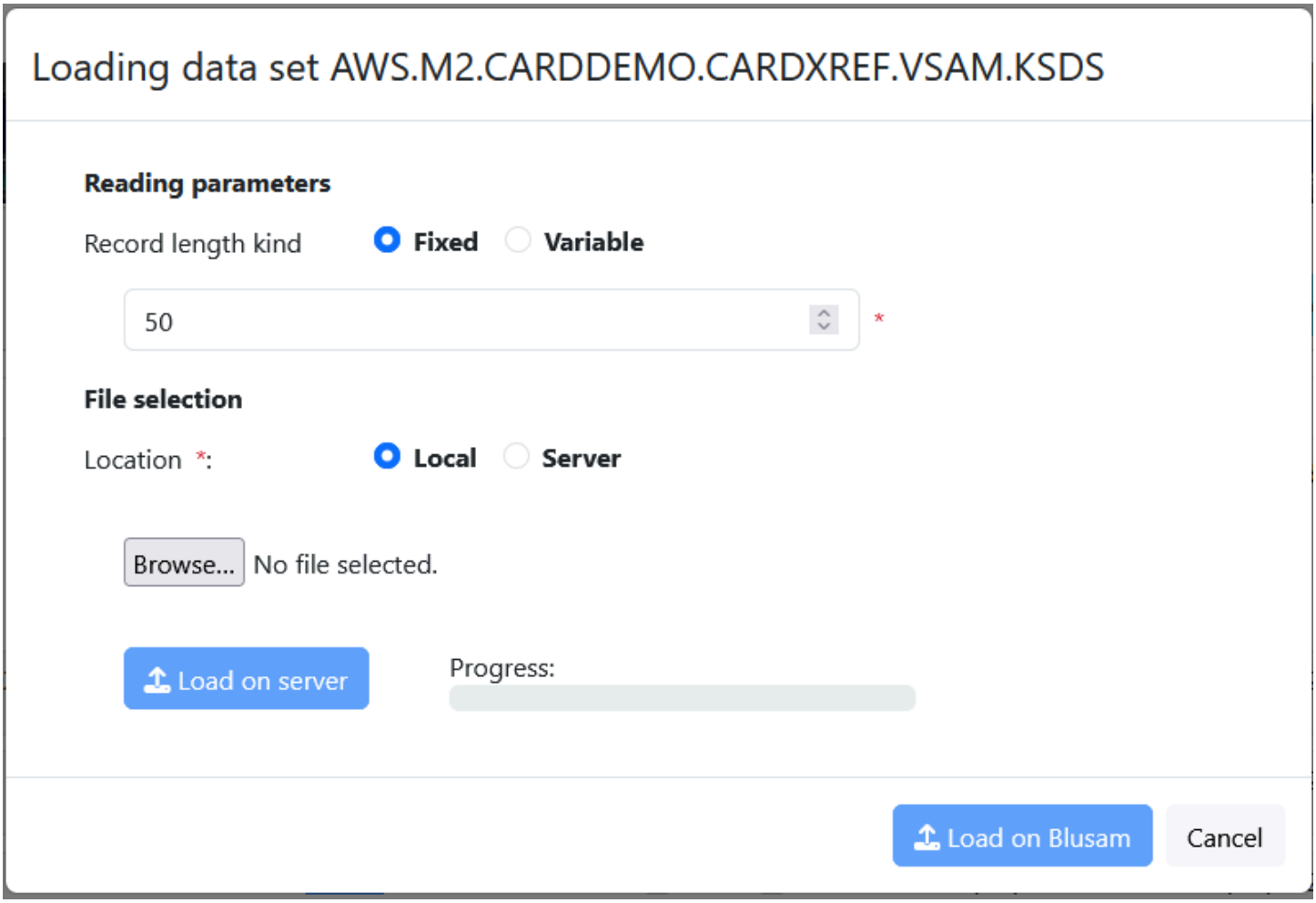


將記錄載入資料集

若要將記錄載入資料集，請選擇動作，然後選擇載入。



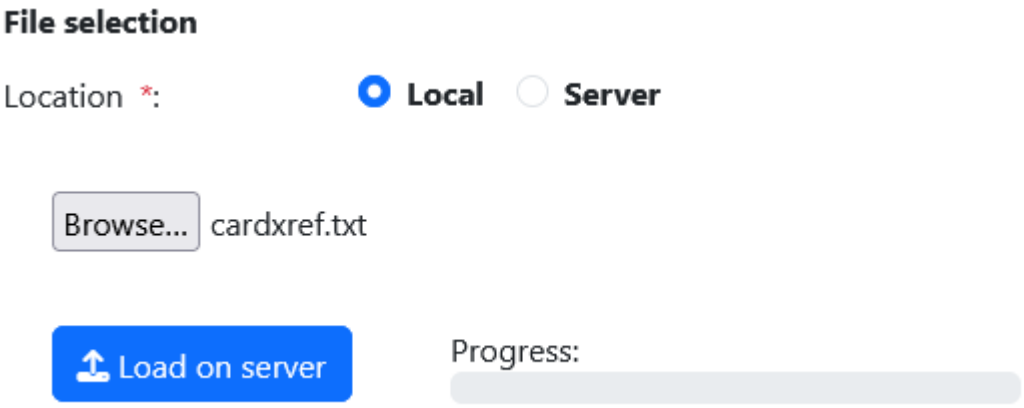
隨即出現具有載入選項的視窗。



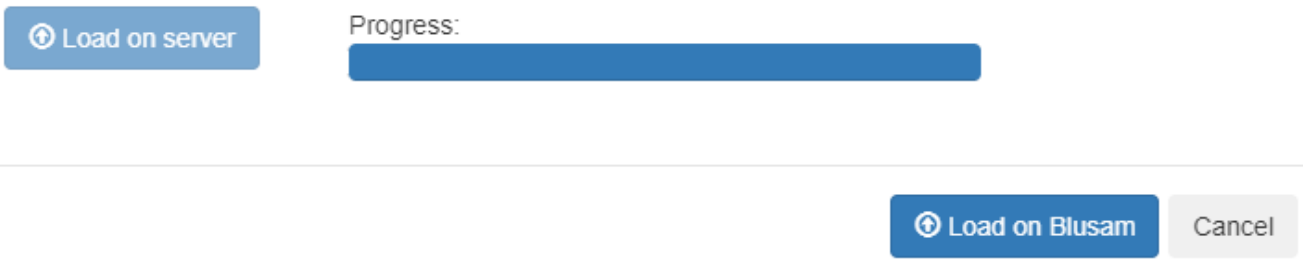
一開始，伺服器負載和藍上負載按鈕都會停用。

讀取參數：

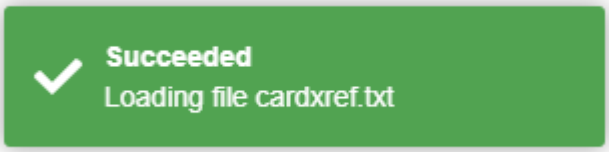
- 記錄長度類型：
  - 固定或可變記錄長度：使用選項按鈕指定舊版資料集匯出是否使用固定長度記錄或可變長度記錄（記錄預期以 RDW 位元組開頭）。如果您選擇固定，則必須在輸入欄位中將記錄長度指定為正整數值 （以位元組為單位）。值應該由來自資料集的資訊預先填入。如果您選擇變數，則指定的輸入欄位會消失。
  - 檔案選擇：
    - 本機：從本機電腦選擇資料集檔案，使用下面的檔案選擇器 （注意：檔案選擇器使用瀏覽器的地區設定來列印其訊息 -- 此處為法文，但在您這方面可能看起來不同，這是預期的）。進行選取後，視窗會更新為資料檔案名稱，並啟用伺服器上載入按鈕：



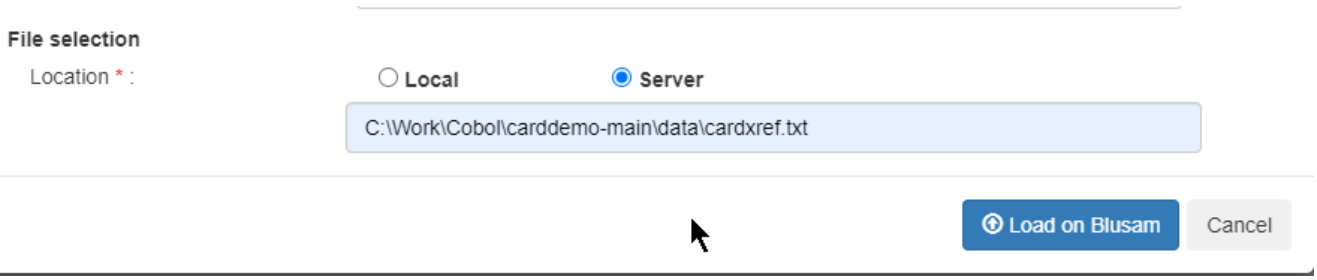
選擇在伺服器上載入。進度列到達結尾後，Load on Blusam 按鈕就會啟用：



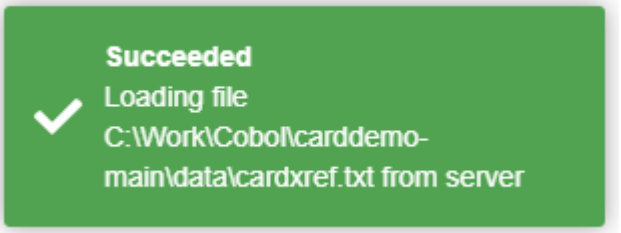
若要完成 Blusam 儲存的載入程序，請選擇 Load on Blusam。否則，請選擇 Cancel (取消)。如果您選擇繼續載入程序，載入程序完成後，右下角會顯示通知：



- 伺服器：選擇此選項會在伺服器上載入按鈕消失時顯示輸入欄位。輸入欄位是您必須在 Blusam 伺服器上指定資料集檔案的路徑（假設您已先將指定檔案傳輸至 Blusam 伺服器）。指定路徑後，Load on Blusam 會啟用：

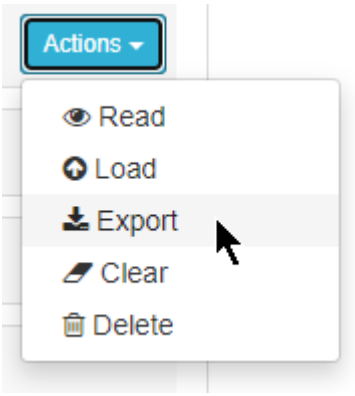


若要完成載入程序，請選擇載入藍本。否則，請選擇 Cancel (取消)。如果您選擇繼續載入，載入程序完成後就會顯示通知。通知與瀏覽器的負載不同，因為它會顯示資料檔案伺服器路徑，後面接著伺服器的字詞：

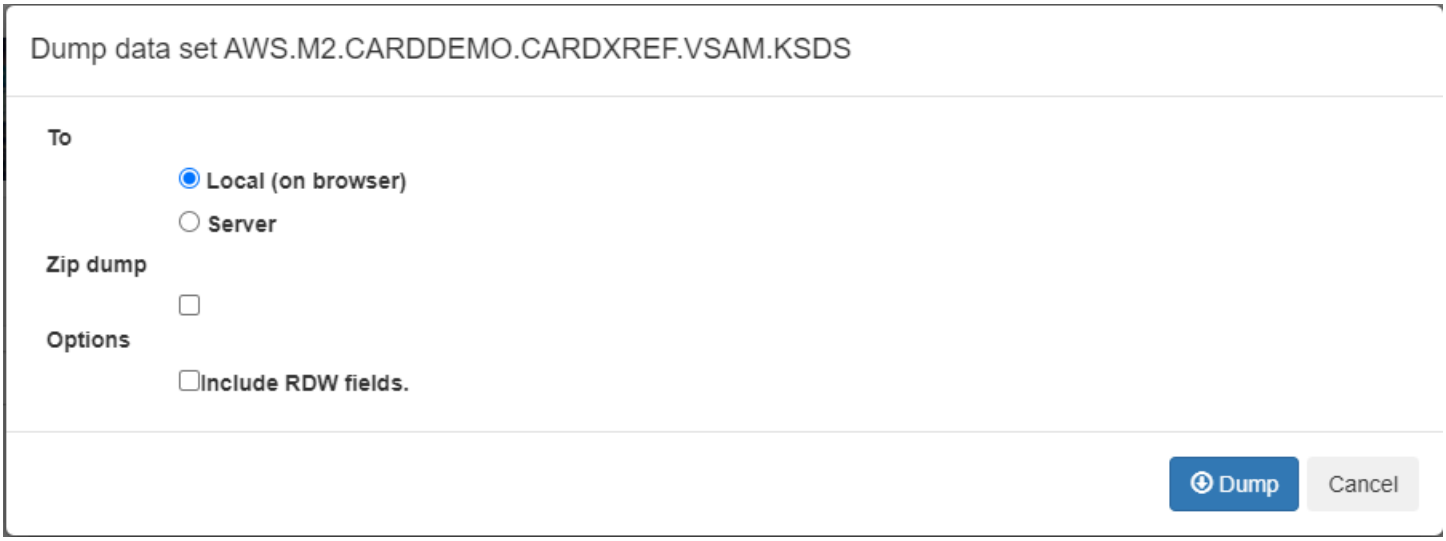


從資料集匯出記錄

若要匯出資料集記錄，請選擇目前資料集列中的動作，然後選擇匯出：



出現下列快顯視窗。



選項：

若要 ：選項按鈕選擇，以選擇匯出目的地，無論是在瀏覽器 (本機 （ 在瀏覽器上 ）) 中下載，或是託管 BAC 應用程式之伺服器上的特定資料夾。如果您選擇使用伺服器選擇匯出，則會顯示新的輸入欄位：



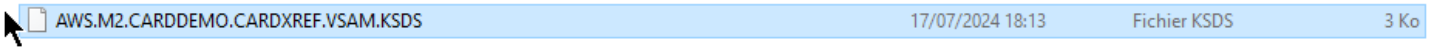
如輸入欄位右側的紅色星號所示，您必須在伺服器上提供有效的資料夾位置 （ 傾印按鈕會在未提供資料夾位置時處於非作用中狀態 ）。

若要匯出至伺服器，如果您計劃在匯出後操作匯出的資料集檔案，您必須擁有足夠的伺服器檔案系統存取權。

壓縮傾印：產生壓縮封存而非原始檔案的核取方塊。

選項：若要在可變長度記錄資料集的情況下，在匯出的資料集的每個記錄開頭包含記錄描述項字詞 (RDW)，請選擇包含 RDW 欄位。

若要啟動資料集匯出程序，請選擇傾印。如果您選擇匯出至瀏覽器，請檢查匯出資料集檔案的下載資料夾。檔案的名稱將與資料集相同：

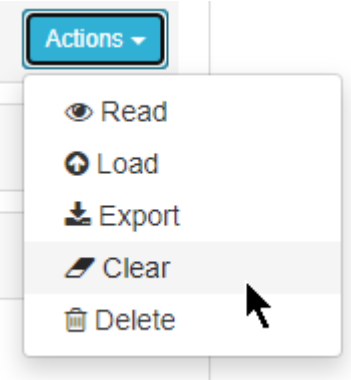


請注意：

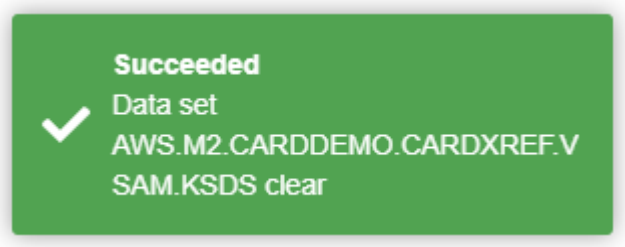
- 對於 KSDS，記錄將按照主索引鍵順序匯出。
- 對於 ESDS 和 RRDS，記錄將按照 RBA （ 相對位元組地址 ） 順序匯出。
- 對於所有資料集類型，記錄都會匯出為原始二進位陣列 （ 不會進行任何種類的轉換 ），以確保與舊版平台的直接相容性。

從資料集清除記錄

若要清除資料集中的所有記錄，請選擇動作，然後選擇清除：

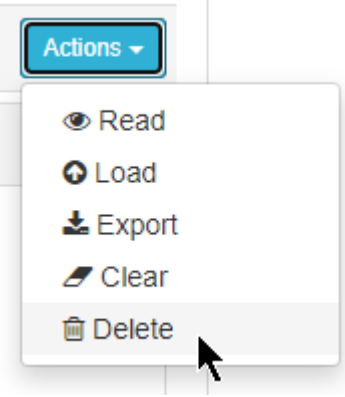


從資料集移除所有記錄後，會出現下列通知。

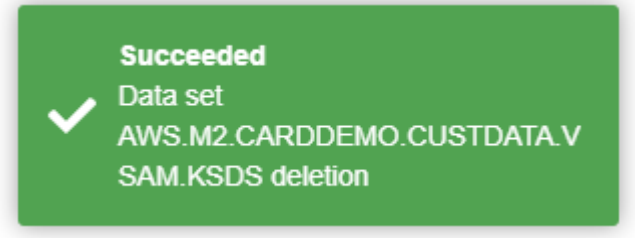


刪除資料集

若要刪除資料集，請選擇動作，然後選擇刪除：



刪除資料集後，會出現下列通知：

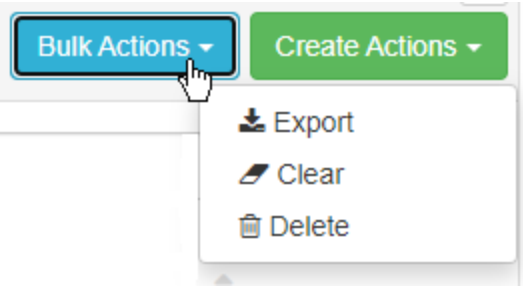


大量操作

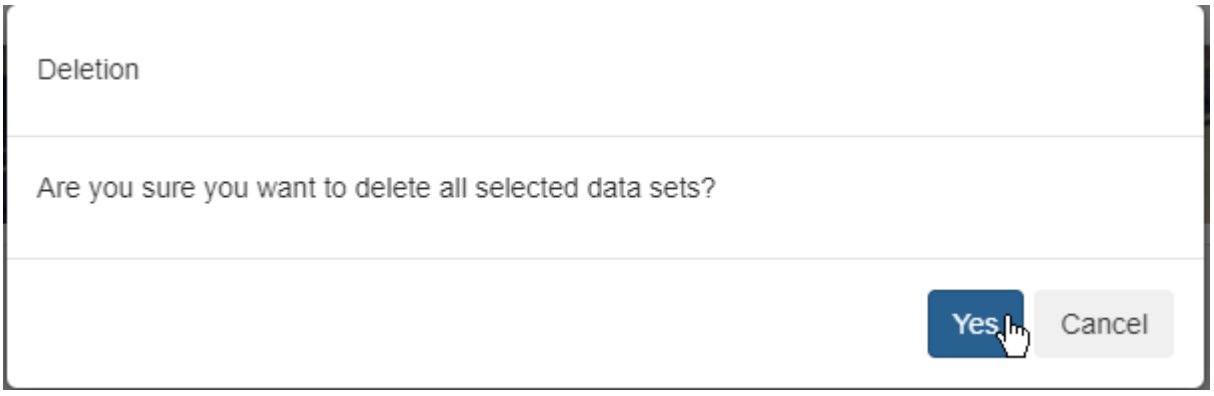
資料集有三種大量操作可用：

- 匯出
- Clear
- Delete

大量操作只能套用至資料集的選擇（至少需要選取一個資料集）；選取資料集是透過資料集清單資料表左側的勾選選擇核取方塊來完成。選取至少一個資料集將啟用大量動作下拉式清單：



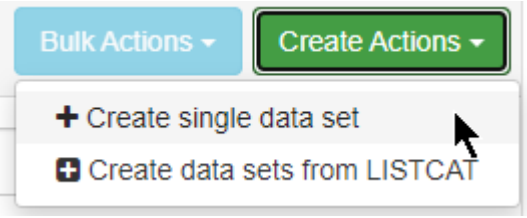
除了特定動作適用於資料集的選擇，而不是單一資料集之外，這些動作與上述動作類似，因此請參閱專用動作文件以取得詳細資訊。快顯視窗文字內容將略有不同，以反映大量性質。例如，嘗試刪除數個資料集時，快顯視窗會如下所示：



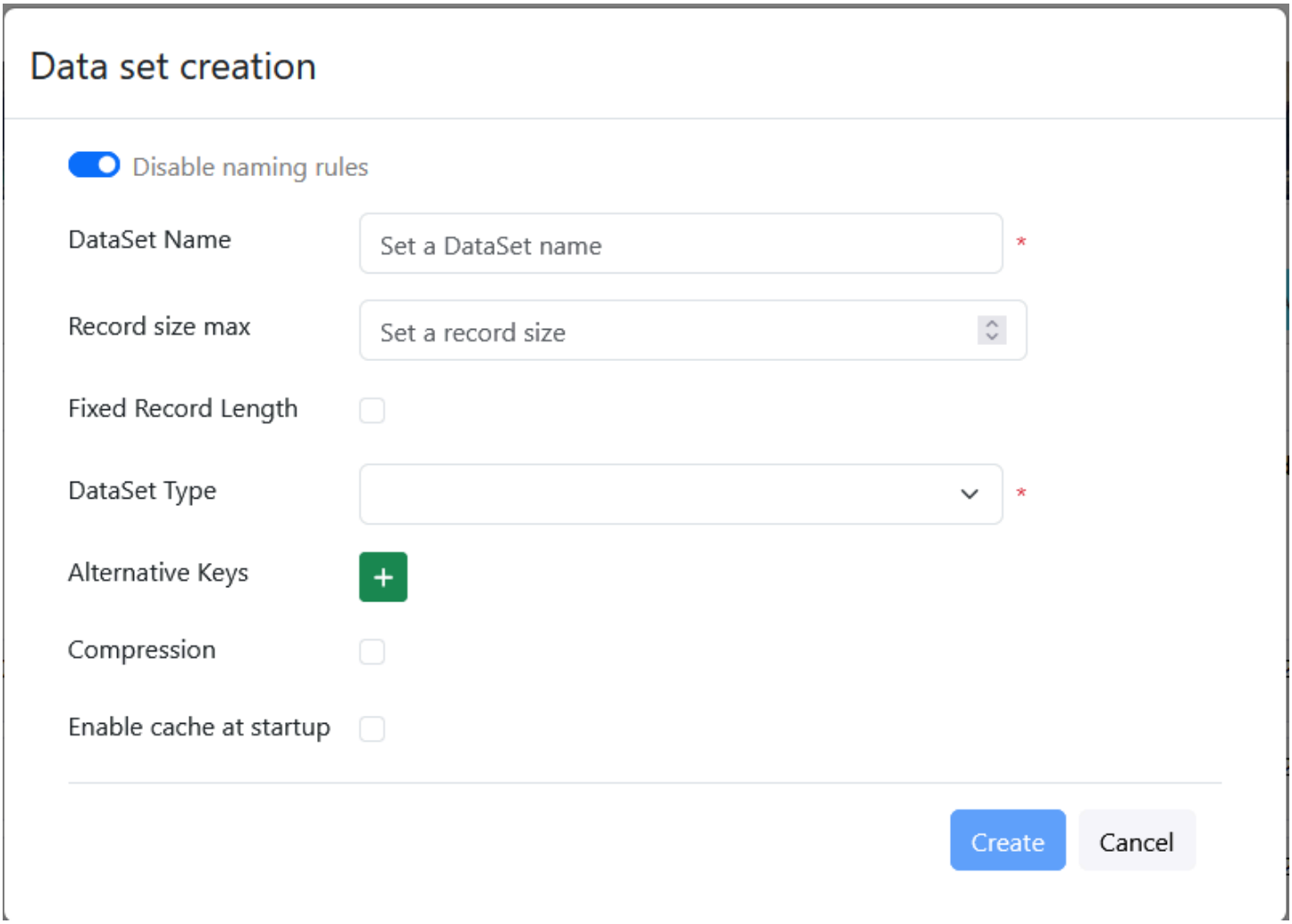
建立操作

建立單一資料集

選擇動作，然後選擇建立單一資料集：



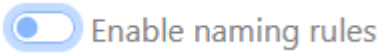
然後，資料集建立表單會顯示為快顯視窗：



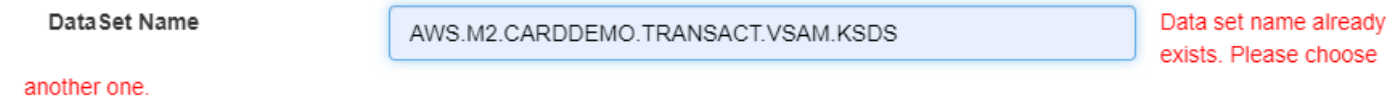
您可以為資料集定義指定下列屬性：

- 啟用和停用命名規則：使用「停用命名規則/啟用命名規則」切換小工具來停用和啟用資料集命名慣例。建議您將切換保留在預設值上，並啟用資料集命名規則（切換小工具應顯示「停用命名規則」）：

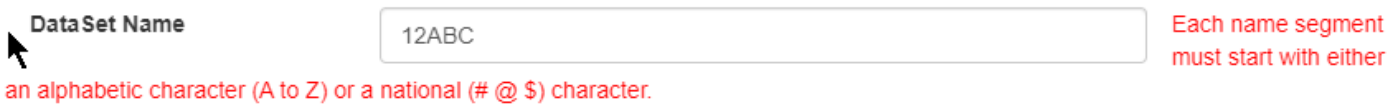
  

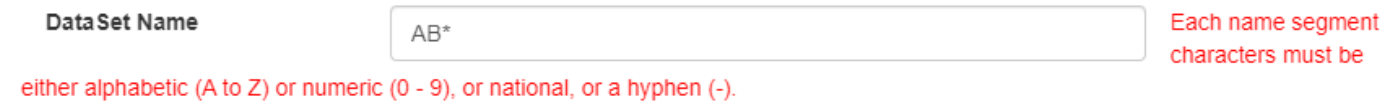


- 資料集名稱：資料集的名稱。如果您指定的名稱已在使用中，則會出現下列錯誤訊息。

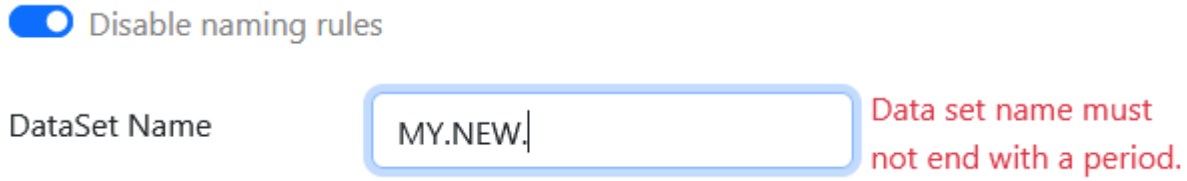


  如果已啟用，名稱也必須遵守命名慣例：





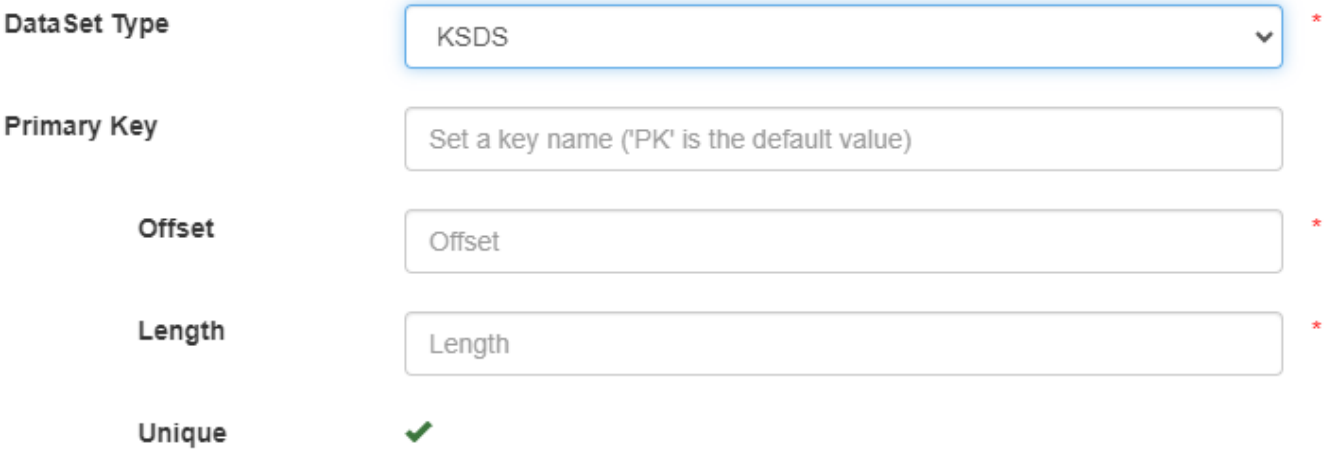




- 記錄大小上限：這必須是正整數，代表具有固定長度記錄之資料集的記錄大小。對於具有可變長度記錄 的資料集，您可以將其保留空白。
- 固定長度記錄：指定記錄長度是固定的還是變數的核取方塊。如果選取，資料集將具有固定長度的記錄，否則記錄長度將是可變的。

  當您將舊版資料匯入變數長度記錄資料集時，提供的舊版記錄必須包含記錄描述項 (RDW)，以提供每個記錄的長度。
- 資料集類型：用於指定目前資料集類型的下拉式清單。支援下列類型。
  - ESDS
  - LargeESDS
  - KSDS

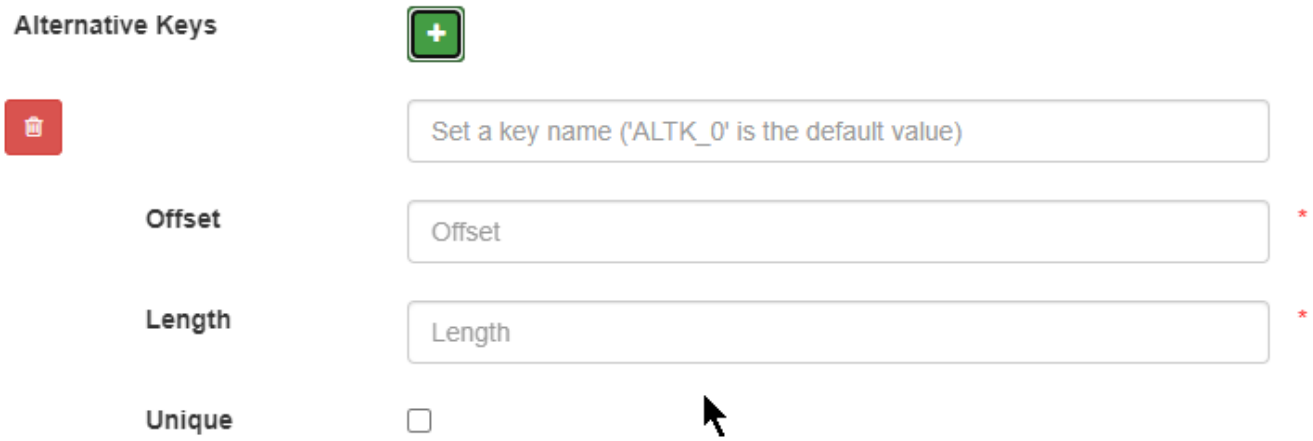
  對於 KSDS，您必須指定主索引鍵：



針對主索引鍵，指定下列項目：

- 名稱：此欄位為選用。預設值為 **PK**。
- 位移：記錄中主索引鍵的 0 型位移。位移必須是正整數。此欄位為必填。
- 長度：主索引鍵的長度。此長度必須是正整數。此欄位為必填。

對於 KSDS 和 ESDS，您可以選擇選擇備用金鑰標籤前面的加號按鈕來定義備用金鑰的集合。每次選擇該按鈕時，新的備用金鑰定義區段會出現在資料集建立表單中：



對於每個備用金鑰，您需要提供：

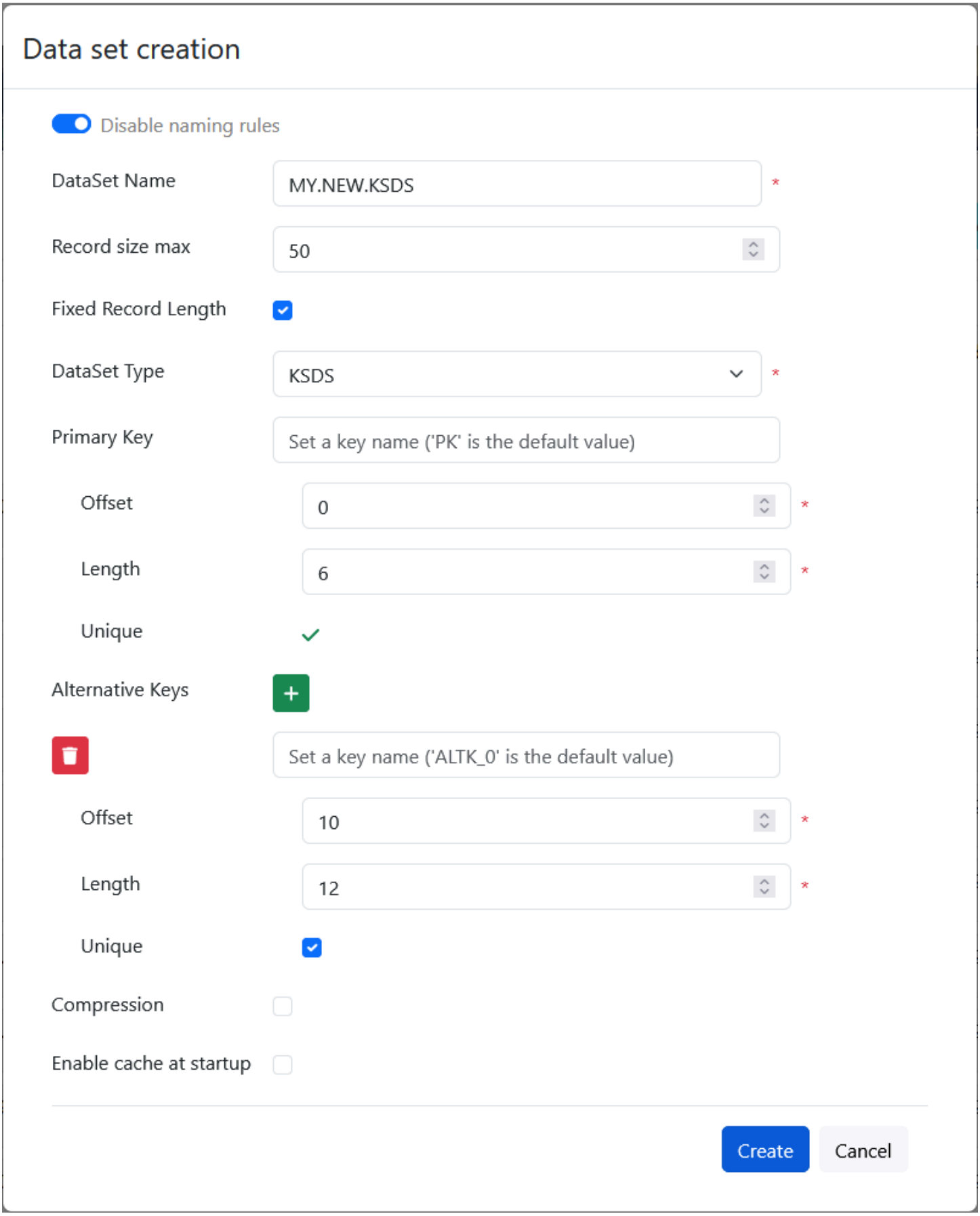
- 名稱：此欄位為選用。預設值為 **ALTK_#**，其中 # 代表從 0 開始的自動遞增計數器。
- 位移：記錄中替代索引鍵的 0 型位移。必須是正整數。此欄位為必填。
- 長度：備用金鑰的長度。此長度必須是正整數。此欄位為必填。

- 唯一：核取方塊，指出替代索引鍵是否接受重複的項目。如果選取，備用金鑰將定義為唯一（不接受重複的金鑰項目）。此欄位為必填。

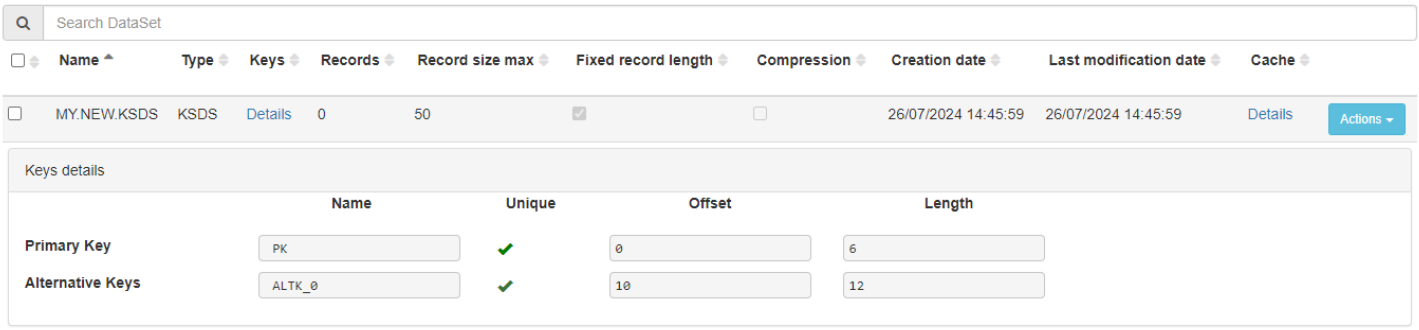
  若要移除備用金鑰定義，請使用左側的垃圾桶按鈕。

- 壓縮：指定是否使用壓縮來存放資料集的核取方塊。
- 啟動時啟用快取：核取方塊，指定是否應在應用程式啟動時將資料集載入快取。

指定屬性定義之後，請選擇建立以繼續：


Data set creation
Disable naming rules
DataSet Name
MY.NEW.KSDS
Record size max
50
Fixed Record Length
DataSet Type
KSDS
Primary Key
Set a key name ('PK' is the default value)
Offset
0
Length
6
Unique
Alternative Keys
Set a key name ('ALTK_0' is the default value)
Offset
10
Length
12
Unique
Compression
Enable cache at startup
Create
Cancel

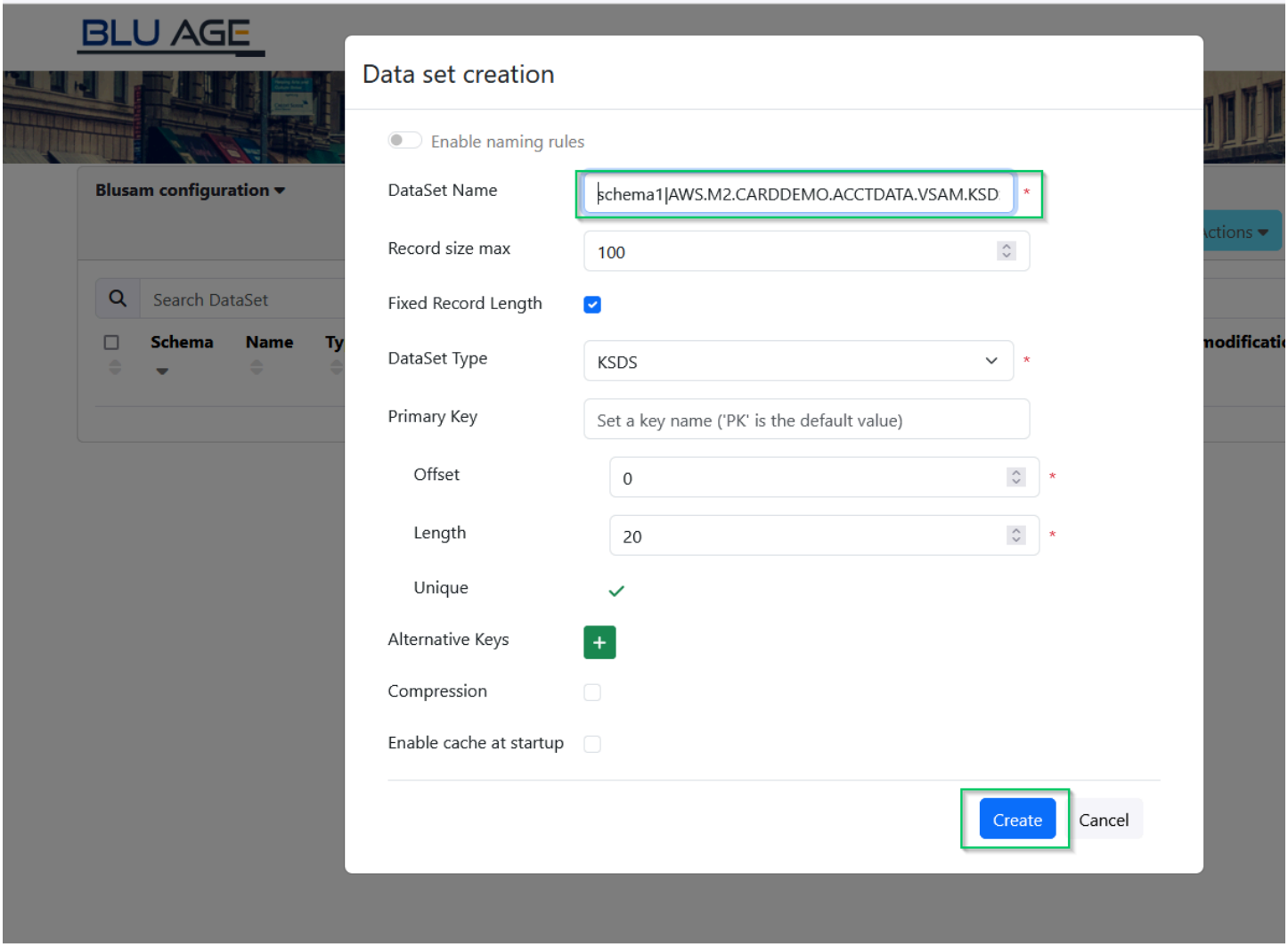
建立視窗將會關閉，並顯示顯示資料集清單的首頁。您可以檢視新建立資料集的詳細資訊。



在多結構描述模式中建立單一資料集

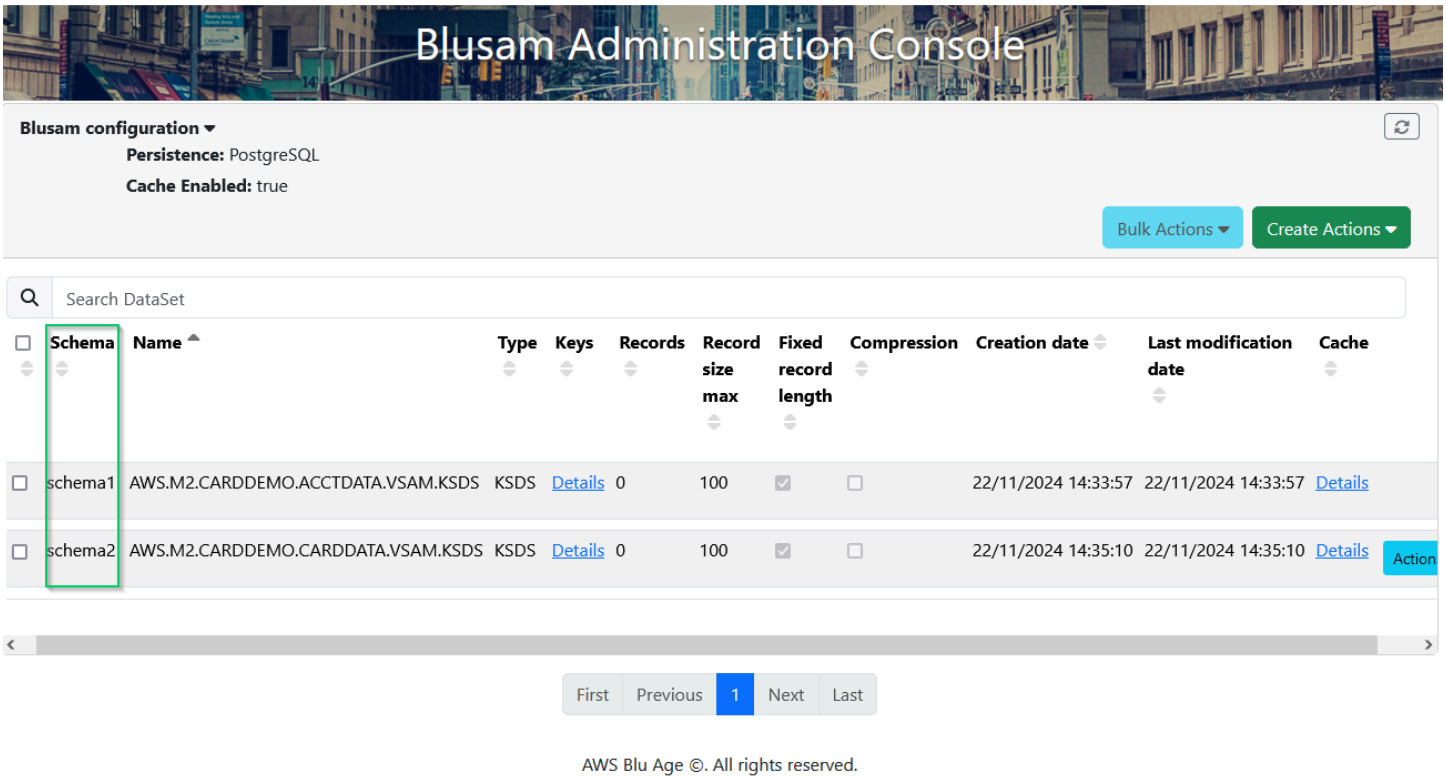
資料集可以在多結構描述模式中建立，方法是在資料集名稱前面加上結構描述名稱，後面接著管道 (|) 符號（例如，`schema1|AWS.M2.CARDDEMO.ACCTDATA.VSAM.KSDS`)。

> **Note**
>
> 用於建立資料集的結構描述應在`application-main.yml`組態中指定。如需詳細資訊，請參閱the section called “多結構描述組態屬性”。



如果未提供結構描述字首，則會在 Blusam 資料來源組態中 Blusam 資料來源 URL 中指定的預設結構描述中建立資料集。如果 Blusam 資料來源 URL 中未指定結構描述，則預設會使用「公有」結構描述。

> **Note**
>
> 在多結構描述模式中，BAC 主控台會在第一欄中顯示資料集的結構描述資訊。



從 LISTCAT 建立資料集

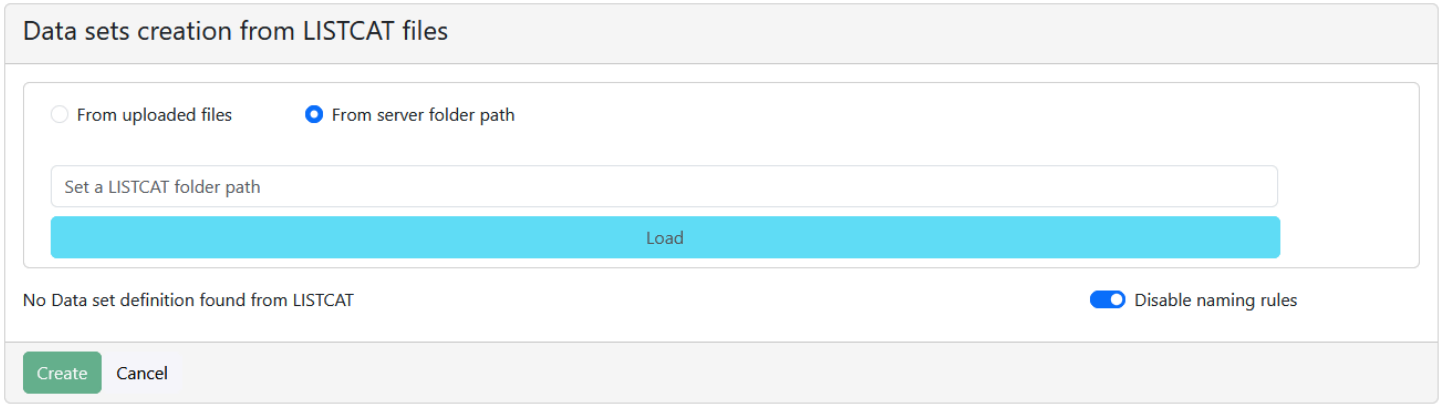
此功能可讓您利用使用 BluInsights Transformation Center 在 BluAge 轉換程序期間建立的 LISTCAT JSON 檔案，因為剖析了舊版平台的 LISTCAT 匯出：LISTCAT 匯出會剖析並轉換為包含資料集定義的 JSON 檔案（名稱、資料集類型、金鑰定義，以及記錄長度是固定的還是可變的）。

擁有 LISTCAT JSON 檔案可讓您直接建立資料集，而無需手動輸入資料集所需的所有資訊。您也可以直接建立資料集的集合，而不必逐一建立資料集。

如果您的專案沒有可用的 LISTCAT JON 檔案（例如，因為轉換時沒有可用的 LISTCAT 匯出檔案），只要您遵守附錄中詳述的 LISTCAT JSON 格式，您一律可以手動建立一個。

從建立動作下拉式清單中，選擇從 LISTCAT 建立資料集。

將顯示以下專用頁面：



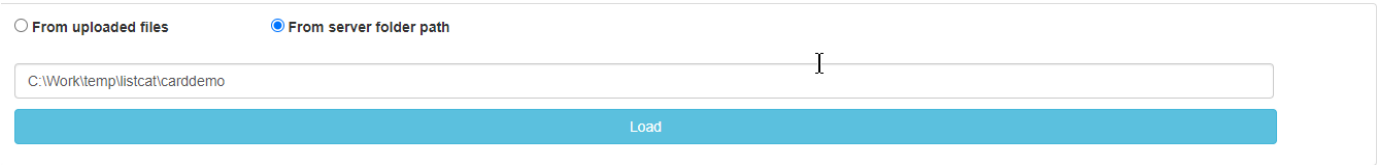
在此階段，載入按鈕會停用，這是預期的。

使用選項按鈕指定您要提供 LISTCAT JSON 檔案的方式。有兩個選項：

- 您可以使用瀏覽器上傳 JSON 檔案。
- 您可以從伺服器上的資料夾位置選取 JSON 檔案。若要選擇此選項，您必須先將 JSON 檔案複製到伺服器上具有適當存取權限的指定資料夾路徑。

在伺服器上使用 JSON 檔案

1. 在伺服器上設定資料夾路徑，指向包含 LISTCAT JSON 檔案的資料夾：



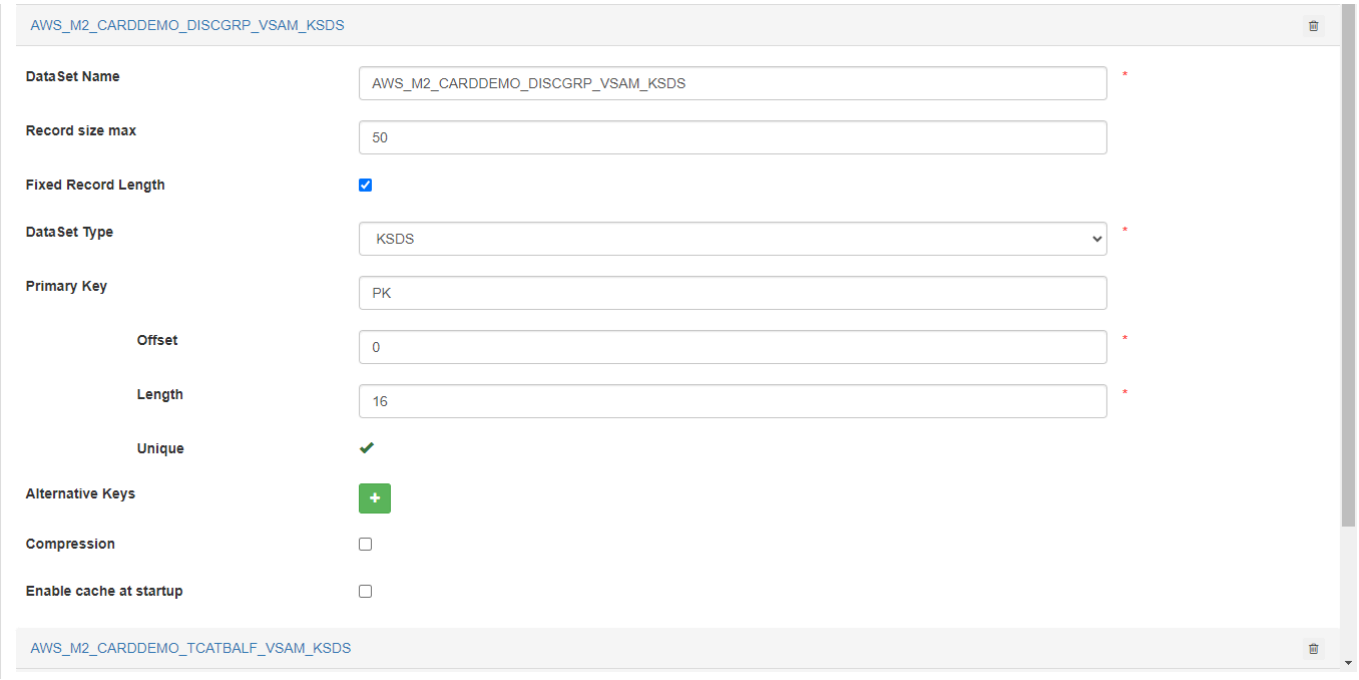
2. 選擇載入按鈕。所有已辨識的資料集定義都會列在表格中：



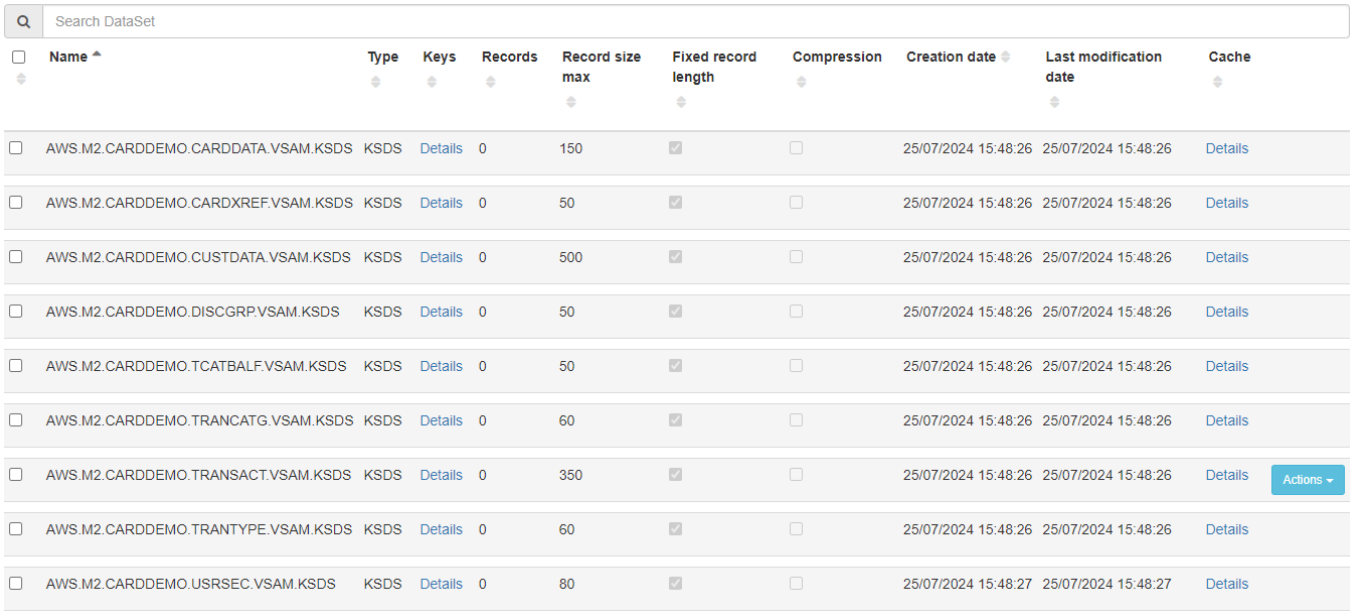
每一列代表資料集定義。您可以使用垃圾桶按鈕從清單中移除資料集定義。

> **Important**
>
> 從清單中移除是立即的，沒有警告訊息。

3. 左側的名稱是連結。您可以選擇它來顯示或隱藏資料集定義的詳細資訊，這是可編輯的。您可以自由修改定義，從剖析的 JSON 檔案開始。

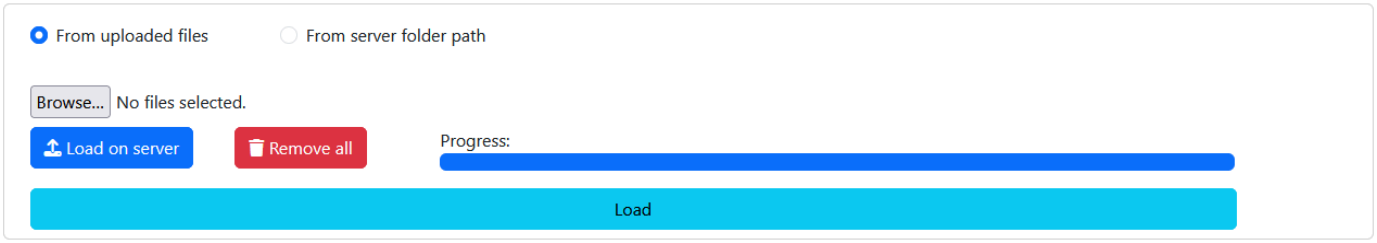


4. 若要建立所有資料集，請選擇建立。所有資料集都會建立，並顯示在資料集結果頁面上。新建立的資料集都會有 0 筆記錄。



將檔案上傳至伺服器

1. 此選項類似於使用伺服器資料夾路徑中的檔案，但在這種情況下，您必須先使用檔案選取器上傳檔案。選取要從本機電腦上傳的所有檔案，然後選擇在伺服器上載入。



2. 當進度列到達結尾時，所有檔案都已成功上傳到伺服器，並啟用載入按鈕。選擇載入按鈕，並使用前述探索到的資料集定義。

LISTCAT JSON 格式

LISTCAT JSON 格式由下列屬性定義：

- 選用的「catalogId」：舊版目錄的識別符做為字串，或預設目錄的「預設」。
- "identifier"：資料集名稱，做為字串。
- "isIndexed"：指出 KSDS 的布林值旗標：KSDS 為 true，否則為 false。
- "isLinear"：指出 ESDS 的布林值旗標：ESDS 為 true，否則為 false。
- 「isRelative」：指出 RRDS 的布林值旗標：RRDS 為 true，否則為 false

- 注意：「isIndexed」、「isLinear」和「isRelative」是互斥的。
- "isFixedLengthRecord"：布林值旗標：如果固定長度記錄資料集，則設為 true，否則為 false。
- "avgRecordSize"：以位元組為單位的平均記錄大小，以正整數表示。
- "maxRecordSize"：以位元組為單位的最大記錄大小，以整數表示。應等於固定長度記錄大小的 avgRecordSize。
- 僅限 KSDS：強制性主要金鑰定義（做為巢狀物件）
  - 已標記 "primaryKey"
  - "offset"：記錄中主索引鍵的以 0 為基礎的位元組位移。
  - 「長度」：以主索引鍵位元組為單位的長度。
  - 「唯一」：主索引鍵必須設為 true。
- 對於 KSDS/ESDS，收集備用金鑰（做為巢狀物件的集合）：
  - 已標記 "alternateKeys"
  - 對於每個備用金鑰：
    - "offset"：記錄中備用索引鍵的 0 位元組偏移。
    - "length"：備用金鑰的位元組長度。
    - 「唯一」：如果金鑰不接受重複的項目，則必須為備用金鑰設為 true，否則為 false。
- 如果不存在備用金鑰，請提供空集合：

```
alternateKeys: []
```

以下是範例 KSDS LISTCAT JSON 檔案。

```
{
  "catalogId": "default",
  "identifier": "AWS_M2_CARDDEMO_CARDXREF_VSAM_KSDS",
  "isIndexed": true,
  "isLinear": false,
  "isRelative": false,
  "isFixedLengthRecord": true,
  "avgRecordSize": 50,
  "maxRecordSize": 50,
  "primaryKey": {
    "offset": 0,
    "length": 16,
    "unique": true
```

```
  },
  "alternateKeys": [
    {
      "offset": 25,
      "length": 11,
      "unique": false
    }
  ]
}
```

設定 AWS Blu Age Runtime 的組態

AWS Blu Age 執行期和用戶端程式碼是使用 Spring Boot 架構的 Web 應用程式。它利用 Spring 功能來提供組態，具有數個可能的位置和優先順序規則。提供許多其他檔案也有類似的優先順序規則，例如 groovy 指令碼、sql 等。

AWS Blu Age Runtime 也包含額外的選用 Web 應用程式，可視需要選擇加入。

主題

- 應用程式組態基本概念
- 應用程式優先順序
- 資料庫的 JNDI
- AWS Blu Age 執行期秘密
- 其他檔案 (groovy、sql 等)
- 其他 Web 應用程式
- 啟用 AWS Blu Age Runtime 的屬性
- AWS Blu Age Runtime 中的可用 Redis 快取屬性
- 設定 Gapwalk 應用程式的安全性

應用程式組態基本概念

處理應用程式組態的預設方法是使用應用程式伺服器`config`資料夾中提供的專用 YAML 檔案。有兩個主要 YAML 組態檔案：

- `application-main.yaml`
- `application-`*`profile`*`.yaml` （在應用程式產生期間設定*`profile`*值的位置）。

第一個檔案會設定架構，即 Gapwalk-application.war，而第二個檔案是專門用於用戶端應用程式的其他選項。這適用於使用彈簧設定檔：Gapwalk 應用程式使用main設定檔，而用戶端應用程式使用*profile*設定檔。

下列範例顯示典型的主要 YAML 檔案。

```
########################################
##### JICS datasource configuration #####
########################################
datasource:
  jicsDs:
    driver-class-name : org.postgresql.Driver
    url: jdbc:postgresql://localhost/jics
    username: jics
    password: jics
    type : org.postgresql.ds.PGSimpleDataSource

#####################################################
##### Embedded Bluesam datasource configuration #####
#####################################################
  bluesamDs :
    driver-class-name : org.postgresql.Driver
    url : jdbc:postgresql://localhost/bluesam
    username : bluesam
    password : bluesam
    type : org.postgresql.ds.PGSimpleDataSource

########################################
##### Embedded Bluesam configuration #####
########################################
bluesam :
  remote : false
  cache : ehcache
  persistence : pgsql #pgsql, mssql, xodus...
  ehcache:
    resource-pool:
      size: 4GB
  write-behind:
```

下列範例顯示典型的用戶端 YAML 檔案。

```
# Logback context logger integration.
logging.config : classpath:logback-[redacted].xml
# Limits Spring logger output.
logging.level.org.springframework.beans.factory.support.DefaultListableBeanFactory : WARN
logging.level.org.springframework.statemachine : WARN
# If the datasource support mode is not static-xa, spring JTA transactions autoconfiguration must me disabled
spring.jta.enabled : false

spring:
  aws:
    client:
      datasources:
        names: primary
        primary:
          secret: arn:aws:secretsmanager:[redacted]

spring.jta.atomikos.datasource.primary.unique-resource-name: primary
spring.jta.atomikos.datasource.primary.xa-data-source-class-name: org.postgresql.xa.PGXADataSource
spring.jta.atomikos.datasource.primary.maxPoolSize: 20
spring.jta.atomikos.datasource.primary.autoCommit: false
```

如需 YAML 檔案內容的資訊，請參閱 啟用 AWS Blu Age Runtime 的屬性。

應用程式優先順序

對於這些組態檔案，適用 Spring 優先順序規則。值得注意的是：

- `application-main` YAML 檔案會顯示在具有預設值的 Gapwalk 主戰檔案中，資料夾中的 YAML 檔案會`config`取代它。
- 用戶端應用程式組態也應該這樣做
- 其他參數可能會在伺服器啟動時傳遞到命令列。它們會覆寫 YAML。

如需詳細資訊，請參閱官方 Spring Boot 文件。

資料庫的 JNDI

在 Tomcat 的 context.xml 檔案中，可能會提供 JNDI 資料庫組態。任何這類組態都會覆寫 YAML 組態。但請注意，使用此功能將不允許 將您的登入資料包裝在秘密管理員中（請參閱下文）。

下列範例顯示 JICS 和 BluSam 資料庫的範例組態。

```
<Resource auth="Container" driverClassName="org.postgresql.Driver" initialSize="0"
 maxIdle="5"
   maxOpenPreparedStatements="-1" maxTotal="10" maxWaitMillis="-1" name="jdbc/jics"
   poolPreparedStatements="true" testOnBorrow="false" type="javax.sql.DataSource"
   url="jdbc:postgresql://XXXX.rds.amazonaws.com:5432/XXXX" username="XXXX"
 password="XXXX" />
```

jdbc/jics

將為 `jdbc/jics` JICS 資料庫和 `jdbc/bluesam`（請注意 'e') 為 blusam 資料庫。

url="jdbc：postgresql：//XXXX.rds.amazonaws.com：5432/XXXX" username="XXXX" password="XXXX"

資料庫 URL、使用者名稱和密碼。

AWS Blu Age 執行期秘密

某些包含登入資料的資源組態可以使用 AWS 秘密進一步保護。其目的是將關鍵資料存放在 AWS 秘密中，並在 YAML 組態中參考秘密，以便在 Apache Tomcat 啟動時即時取得秘密內容。

Aurora 的秘密

Aurora 資料庫組態（適用於 JICS、Blusam、Customer db 等）將使用內建資料庫秘密，這會自動從對應的資料庫填入所有相關欄位。

> **Note**
>
> `dbname` 金鑰是選用的，取決於您的資料庫組態，它是否會進入秘密。您可以手動將其新增至該處，或將名稱提供給 YAML 檔案。

其他秘密

其他秘密適用於具有單一密碼（特別是受密碼保護的 redis 快取）的資源。在這種情況下，必須使用其他類型的秘密。

秘密的 YAML 參考

`application-main.yml` 可以參考各種資源的秘密 ARN：

JICS 資料庫

使用 的 JICS 資料庫登入資料 `spring.aws.jics.db.secret`

```
spring:
   aws:
     jics:
       db:
```

```
        dbname: jics
        secret: arn:aws:secretsmanager:XXXX
```

支援的 JICS 資料庫私密金鑰：

私密金鑰	私密金鑰描述
託管	主機名稱
port	連接埠
dbname	資料庫的名稱
使用者名稱	使用者名稱
密碼	密碼
engine	資料庫引擎：Postgres、Oracle、Db2、Microsoft SQL Server
currentSchema	要使用的特定結構描述（僅限 Db2 支援）
sslConnection	是否使用 SSL 連線（僅限 Db2 支援）
sslTrustStoreLocation	用戶端上信任存放區的位置（僅限 Db2 支援）
sslTrustStorePassword	用戶端上信任存放區的密碼（僅限 Db2 支援）

> **Note**
>
> 資料庫的名稱會在秘密或 yaml 參考 中提供`spring.aws.jics.db.dbname`。

Blusam 資料庫

使用 的 Blusam 資料庫登入資料 `spring.aws.client.bluesam.db.secret`

```
spring:
   aws:
     client:
```

```
      bluesam:
        db:
          dbname: bluesam
          secret: arn:aws:secretsmanager:XXXX
```

支援的 Blusam 資料庫私密金鑰：

私密金鑰	私密金鑰描述
託管	主機名稱
port	連接埠
dbname	資料庫的名稱
使用者名稱	使用者名稱
密碼	密碼
engine	資料庫引擎：Postgres

> **Note**
>
> 資料庫的名稱會在秘密或 yaml 參考 中提供`spring.aws.client.bluesam.db.dbname`。

用戶端資料庫

用戶端`application-profile.yml`可以參考用戶端資料庫的秘密 ARN。這需要額外的 屬性來列出資料來源名稱 `spring.aws.client.datasources.names`。對於每個資料來源名稱，請在下列屬性中`ds_name`指定秘密 ARN：`spring.aws.client.datasources.ds_name.secret`。範例：

```
spring:
   aws:
     client:
       datasources:
         names: primary,host
         primary:
           secret: arn:aws:secretsmanager:XXXX
         host:
```

```
            dbname: hostdb
            secret: arn:aws:secretsmanager:XXXX
```

名稱：主要、主機：

一個範例，其中有兩個用戶端資料來源名為主要資料來源和主機，每個資料來源都有其資料庫和登入資料。

dbname：hostdb：

在此範例中，「主機」資料庫的名稱不在秘密中，而是在此處提供，而對於「主要」資料庫，它在秘密中。

支援的用戶端資料庫私密金鑰：

私密金鑰	私密金鑰描述
託管	主機名稱
port	連接埠
dbname	資料庫的名稱
使用者名稱	使用者名稱
密碼	密碼
engine	資料庫引擎：Postgres、Oracle、Db2、Microsoft SQL Server
currentSchema	要使用的特定結構描述（僅限 Db2 支援）
sslConnection	是否使用 SSL 連線（僅限 Db2 支援）
sslTrustStoreLocation	用戶端上信任存放區的位置（僅限 Db2 支援）
sslTrustStorePassword	用戶端上信任存放區的密碼（僅限 Db2 支援）

PGM 公用程式資料庫

`application-utility-pgm.yml` 可以參考各種資源的秘密 ARN。

- spring.aws.client.datasources.primary
  - secret

    應用程式資料庫的秘密 ARN。

類型：字串

- type

要使用的連線集區實作的完整名稱。

類型：字串

預設：com.zaxxer.hikari.HikariDataSource

- spring.aws.client.utility.pgm.datasources
  - names

資料來源名稱的清單。

類型：字串

- dsname
  - dbname

主機的名稱。

類型：字串

- secret

主機資料庫的秘密 ARN。

類型：字串

- type

要使用的連線集區實作的完整名稱。

類型：字串

預設：com.zaxxer.hikari.HikariDataSource

對於多資料來源秘密：

```
spring:
   aws:
     client:
       primary:
         secret: arn:aws:secretsmanager:XXXX
         type: dataSourceType
       utility:
         pgm:
           datasources:
             names: dsname1,dsname2,dsname3
               dsname1:
                 dbname: dbname1
                 secret: arn:aws:secretsmanager:XXXX
                 type: dataSourceType
               dsname2:
                 dbname: dbname2
                 secret: arn:aws:secretsmanager:XXXX
                 type: dataSourceType
               dsname3:
                 dbname: dbname3
                 secret: arn:aws:secretsmanager:XXXX
                 type: dataSourceType
```

沒有 XA 支援的私密金鑰

- 引擎 (postgres/oracle/db2/mssql)
- port
- dbname
- currentSchema
- 使用者名稱
- 密碼
- url
- sslConnection

- sslTrustStoreLocation
- sslTrustStorePassword

postgres 僅針對sslMode私密金鑰值 (disable/allow/prefer/require/verify-ca/verify-full) 和 spring.aws.rds.ssl.cert-path YAML 屬性，可讓您使用 SSL 連線。

XA 支援的私密金鑰

如果用戶端資料庫使用 XA，則透過秘密值支援子 xa-properties。

- 託管
- port
- dbname
- currentSchema
- 使用者名稱
- 密碼
- url
- sslConnection (true/false)
- sslTrustStoreLocation
- sslTrustStorePassword

不過，對於其他 xa-properties （ 例如 maxPoolSize或 driverType)，仍spring.jta.atomikos.datasource.XXXX.unique-resource-name必須提供一般 YAML 金鑰。

秘密值會覆寫 YAML 屬性。

預設超級管理員 BAC 和 JAC

您也可以設定 application-main.yml，透過指定 ARN，從 AWS Secrets Manager 擷取秘密中預設超級管理員使用者的使用者名稱和密碼。下列範例顯示如何在 YAML 檔案中宣告此秘密。

```
spring:
   aws:
     client:
       defaultSuperAdmin:
```

```
        secret: arn:aws:secretsmanager:XXXX
```

支援的預設超級管理員資料庫私密金鑰：

私密金鑰	私密金鑰描述
使用者名稱	使用者名稱。
密碼	密碼。

OAuth2

您也可以 AWS Secrets Manager 指定提供者和 ARN，將 `application-main.yml` 設定為從 擷取 OAuth2 用戶端秘密。供應商屬性的預設值為 Amazon Cognito。以下是 OAuth2 供應商 Keycloak 的範例組態：

```
spring:
   aws:
     client:
       provider: keycloak
       keycloak:
         secret: arn:aws:secretsmanager:XXXX
```

在此範例中，OAuth2 供應商 Keycloak 的用戶端秘密會從 AWS Secrets Manager 中指定的 ARN 擷取。此組態透過動態解析提供者名稱和對應的秘密 ARN 來支援多個提供者。

支援的 OAuth2 私密金鑰：

私密金鑰	私密金鑰描述
client-secret	授權伺服器在應用程式註冊過程中產生的秘密。

Redis 快取的 Secret Manager

`application-main.yml` 檔案可以參考 Redis 快取的秘密 ARN。支援的項目為：

- 使用 的 Gapwalk Redis 登入資料 `spring.aws.client.gapwalk.redis.secret`
- 使用 的 Bluesam Redis 登入資料 `spring.aws.client.bluesam.redis.secret`

- Bluesam 使用 鎖定 Redis 登入資料 `spring.aws.client.bluesam.locks.redis.secret`
- 使用 的資料集目錄 Redis 登入資料 `spring.aws.client.dataset.catalog.redis.secret`
- 使用 的 JICS Redis 登入資料 `spring.aws.client.jics.redis.secret`
- 使用 的工作階段 Redis 登入資料 `spring.aws.client.jics.redis.secret`
- 使用 的工作階段追蹤器 Redis 登入資料 `spring.aws.client.session.tracker.redis.secret`
- 使用 的 JICS TS Queues Redis 登入資料 `spring.aws.client.jics.queues.ts.redis.secret`
- 使用 的 JCL 檢查點 Redis 憑證 `spring.aws.client.jcl.checkpoint.redis.secret`
- Gapwalk 檔案使用 鎖定 Redis 登入資料 `spring.aws.client.gapwalk.files.locks.redis.secret`
- Blu4IV 使用 鎖定 Redis 登入資料 `spring.aws.client.blu4iv.locks.redis.secret`

下列範例顯示如何在 YAML 檔案中宣告這些秘密。

```
spring:
   aws:
     client:
       gapwalk:
         redis:
           secret: arn:aws:secretsmanager:XXXX
       bluesam:
         locks:
           redis:
             secret: arn:aws:secretsmanager:XXXX
         redis:
           secret: arn:aws:secretsmanager:XXXX
       dataset:
         catalog:
           redis:
             secret: arn:aws:secretsmanager:XXXX
       jics:
         redis:
           secret: arn:aws:secretsmanager:XXXX
       session:
         tracker:
           redis:
             secret: arn:aws:secretsmanager:XXXX
       jics:
```

```
        queues:
          ts:
            redis:
              secret: arn:aws:secretsmanager:XXXX
      jcl:
        checkpoint:
          redis:
            secret: arn:aws:secretsmanager:XXXX
      gapwalk:
        files:
          locks:
            redis:
              secret: arn:aws:secretsmanager:XXXX
      blu4iv:
        locks:
          redis:
            secret: arn:aws:secretsmanager:XXXX
```

支援的 Redis 私密金鑰：

私密金鑰	私密金鑰描述
hostname	Redis 伺服器主機名稱。
port	Redis 伺服器連接埠。
使用者名稱	使用者名稱。
密碼	密碼。

SSL 密碼設定的秘密管理員

`application-main.yml` 檔案可以參考 SSL 密碼設定的秘密 ARN。支援以下項目。

- 使用 的 Gapwalk SSL 登入資料 `spring.aws.client.ssl.secret`

下列範例顯示如何在 YAML 檔案中宣告這些秘密。

```
spring:
   aws:
     client:
```

```
      ssl:
        secret: arn:aws:secretsmanager:XXXX
```

私密金鑰	私密金鑰描述
trustStorePassword	信任存放區密碼。
keyStorePassword	金鑰存放區密碼。

IBM MQ 密碼設定的秘密管理員

`application-main.yml` 檔案可以參考 IBM MQ 的秘密 ARN 設定。支援以下項目。

- IBM MQ 連線定義為清單，登入資料也是：

  `mq.queues.jmsMQQueueManagers[N].secret:`

  第一個連線的 N 會從 0 開始。

下列範例顯示如何在 YAML 檔案中宣告這些秘密。

```
mq.queues.jmsMQQueueManagers[0].secret: Secret-0-ARN
mq.queues.jmsMQQueueManagers[1].secret: Secret-1-ARN
```

如需秘密 ARNs 的資訊，請參閱 Secrets Manager 秘密中的內容？

秘密中定義的屬性會覆寫其在 `jmsMQ` YAML 組態中的對應值。

如果在秘密中`queueManager`設定 ，則會覆寫 YAML 檔案中`mq.queues.jmsMQQueueManagers[N].jmsMQQueueManager`的值。

私密金鑰	私密金鑰描述
queueManager	IBM MQ 佇列管理員名稱。
appName	IBM MQ 應用程式名稱。
通道	IBM MQ 頻道名稱。

私密金鑰	私密金鑰描述
託管	IBM MQ 主機名稱。
port	IBM MQ 連接埠。
userId	IBM MQ 使用者名稱。
密碼	IBM MQ 使用者密碼。
maxPoolSize	IBM MQ 集區大小上限。
sslCipherKey	IBM MQ SSL 密碼套件。

其他檔案 (groovy、sql 等)

客戶專案使用的其他檔案使用與彈簧組態類似的優先順序規則。範例：

- Groovy 指令碼是 `scripts` 資料夾或子資料夾中`.groovy`的檔案。
- SQL 指令碼是 `sql` 資料夾或子資料夾中`.sql`的檔案。
- 協助程式指令碼是 `daemons` 資料夾或子資料夾中`.groovy`的檔案。
- 查詢 資料庫映射檔案是`sql`資料夾子資料夾中名為 `queries-database.mapping` 檔案的檔案。
- Jasper 範本是 `templates` 資料夾或子資料夾中`.jrxml`的檔案。
- 資料集目錄是 `catalog` 資料夾中`.json`的檔案。
- Lnk 檔案是 `lnk` 資料夾中`.json`的檔案。

所有這些位置都可以透過系統屬性或用戶端 YAML 屬性覆寫。

- 對於 Groovy 指令碼： `configuration.scripts`
- 針對 SQL 指令碼： `configuration.sql`
- 對於精靈指令碼： `configuration.daemons`
- 對於查詢資料庫映射檔案： `configuration.databaseMapping`
- 針對 Jasper 範本： `configuration.templates`
- 對於資料集目錄： `configuration.catalog`
- 對於 Lnk 檔案： `configuration.lnk`

如果找不到 屬性，則會從上述預設位置取得檔案。查詢首先會以 tomcat 工作目錄做為根目錄完成，最後在應用程式戰檔案中完成。

其他 Web 應用程式

AWS Blu Age Runtime 在其webapps-extra資料夾中包含其他 Web 應用程式。根據預設，tomcat 伺服器不會提供這些應用程式。

選擇加入這些 Web 應用程式取決於現代化專案，並透過將所需的 war 檔案從 webapps-extra 資料夾移至 webapps 資料夾來完成。之後，該戰會在下一次啟動時由 tomcat 伺服器提供。

某些專案特定的額外組態也可以新增到 YAML 組態檔案中，以進行每次額外爭用，如 application-main.yml 檔案中所完成和上述說明。其他的戰鬥包括：

- gapwalk-utility-pgm.war： 包含對 ZOS 公用程式的支援，並使用 application-utility-pgm.yaml做為其組態。
- gapwalk-cl-command.war： 包含對 AS/400 公用程式的支援，並使用 application-cl-command.yaml做為其組態。
- gapwalk-hierarchical-support.war：包含 IMS/MFS 交易支援，並使用 application-jhdb.yaml做為其組態

啟用 AWS Blu Age Runtime 的屬性

在 Spring Boot 應用程式中， application-main.yml是組態檔案，其中定義了不同類型的屬性，例如接聽連接埠、資料庫連線等。您可以使用此頁面來了解 AWS Blu Age Runtime 的可用屬性，以及如何啟用這些屬性。

主題

- YML 表示法
- 快速入門/使用案例
- 主要應用程式的可用屬性
- 選用 Web 應用程式的可用屬性
- 用戶端應用程式的可用屬性

YML 表示法

在下列文件中， 等屬性parent.child1.child2=true會以 YAML 格式撰寫如下。

```
parent:
 child1:
   child2: true
```

快速入門/使用案例

下列使用案例顯示適用金鑰和值的範例。

- 預設 application-main.yml

```
----
#### DEFAULT APPLICATION-MAIN.YML FILE      #####
#### SHOWING USEFUL CONFIGURATION ELEMENTS #####
#### SHOULD BE OVERRIDDEN AND EXTERNALIZED   #####

###################################
##### Logging configuration #####
###################################
logging:
  config: classpath:logback-main.xml
  level.org.springframework.beans.factory.support.DefaultListableBeanFactory : WARN

###################################
##### Spring configuration #####
###################################
spring:
  quartz:
    auto-startup: false
    scheduler-name: Default
    properties:
      org.quartz.threadPool.threadCount: 1
  jta:
    enabled: false
    atomikos.properties.maxTimeout : 600000
    atomikos.properties.default-jta-timeout : 100000
  jpa:
# DISABLE OpenEntityManagerInViewInterceptor
    open-in-view: false
    # Fix Postgres JPA Error:
    # Method org.postgresql.jdbc.PgConnection.createClob() is not yet implemented.
    properties.hibernate.temp.use_jdbc_metadata_defaults : false
#####################################
##### Jics tables configuration #####
```

```
#####################################

    # The dialect should match the jics datasource choice
    database-platform : org.hibernate.dialect.PostgreSQLDialect #
 org.hibernate.dialect.PostgreSQLDialect, org.hibernate.dialect.SQLServerDialect

    # those properties can be used to create and initialize jics tables
 automatically.
#    properties:
#      hibernate:
#      globally_quoted_identifiers: true
#        hbm2ddl:
#           import_files_sql_extractor :
 org.hibernate.tool.hbm2ddl.MultipleLinesSqlCommandExtractor
#           import_files : file:./setup/initJics.sql
#           auto : create

##########################
###### Level 2 cache #####
##########################
#        cache:
#          use_second_level_cache: true
#          use_query_cache: true
#          region:
#            factory_class: org.hibernate.cache.ehcache.EhCacheRegionFactory
#      javax:
#        persistence:
#          sharedCache:
#            mode: ENABLE_SELECTIVE
##########################
###### Redis settings #####
##########################
  session:
    store-type: none #redis

# Secret manager configuration for global Redis cache
    aws:
      client:
        gapwalk:
          redis:
            secret: arn:aws:secretsmanager:XXXX

########################################
##### JICS datasource configuration #####
```

```
########################################
datasource:
  jicsDs:
    driver-class-name : org.postgresql.Driver # org.postgresql.Driver,
 com.microsoft.sqlserver.jdbc.SQLServerDriver
    url: jdbc:postgresql://localhost/jics # jdbc:postgresql://localhost:5433/jics,
 jdbc:sqlserver://localhost\SQLEXPRESS:1434;databasename=jics;
    username: jics
    password: jics
    type : org.postgresql.ds.PGSimpleDataSource #
 org.postgresql.ds.PGSimpleDataSource,
 com.microsoft.sqlserver.jdbc.SQLServerDataSource

#########################################################
##### Embedded Bluesam datasource configuration #####
#########################################################
  bluesamDs :
    driver-class-name : org.postgresql.Driver
    url : jdbc:postgresql://localhost/bluesam
    username : bluesam
    password : bluesam
    type : org.postgresql.ds.PGSimpleDataSource

########################################
##### Embedded Bluesam configuration #####
########################################
bluesam :
  remote : false
  cache : ehcache
  persistence : pgsql
  ehcache:
    resource-pool:
      size: 4GB
  write-behind:
    enabled: true
  pgsql :
    dataSource : bluesamDs

########################
##### Jics settings #####
########################
rabbitmq.host: localhost
jics:
  cache: false #redis
```

```
  resource-definitions.store-type: jpa # default value: jpa, other possible value:
 redis

jics.disableSyncpoint : false
#jics.initList:
#jics.parameters.datform: DDMMYY
#jics.parameters.applid: VELOCITY
#jics.parameters.sysid: CICS
#jics.parameters.eibtrmid: TERM
#jics.parameters.userid: MYUSERID
#jics.parameters.username: MYUSERNAME
#jics.parameters.opid: XXX
#jics.parameters.cwa.length: 0
#jics.parameters.netname: MYNETNAME
#jics.parameters.jobname: MJOBNAME
#jics.parameters.sysname: SYSNAME

################################################
##### Jics RunUnitLauncher pool settings #####
################################################
#jics.runUnitLauncherPool.enable: false
#jics.runUnitLauncherPool.size: 20
#jics.runUnitLauncherPool.validationInterval: 1000

#########################
##### Jhdb settings #####
#########################
#jhdb.lterm: LTERMVAL
#jhdb.identificationCardData: SomeIDData

###################################
##### DateHelper configuration ####
###################################
#forcedDate: "2013-08-26T12:59:58+01:57"

#############################
##### Sort configuration ####
#############################
#externalSort.threshold: 256MB

###################################
##### Server timeout (10 min)  ####
###################################
spring.mvc.async.request-timeout: 600000
```

```
################################
##### DATABASE STATISTICS  ####
################################
databaseStatistics : false

######################
##### CALLS GRAPH ####
######################
callGraph : false

####################################
#####     SSL configuration    #####
####################################
gapwalk.ssl.enabled : true
gapwalk.ssl.trustStore : "./config/clientkey.jks"
gapwalk.ssl.trustStorePassword : mysslcertifpassword

##################################
##### MQ settings #####
##################################
mq.queues: jmsmq
mq.queues.jmsMQQueueManagers[0].jmsMQQueueManager: QM1
mq.queues.jmsMQQueueManagers[0].jmsMQAppName: Gapwalk
mq.queues.jmsMQQueueManagers[0].jmsMQChannel: DEV.APP.SVRCONN
mq.queues.jmsMQQueueManagers[0].jmsMQHost: localhost
mq.queues.jmsMQQueueManagers[0].jmsMQPort: 1415
mq.queues.jmsMQQueueManagers[0].jmsMQUserid: app
mq.queues.jmsMQQueueManagers[0].jmsMQSSLCipher: "*TLS12ORHIGHER"
mq.queues.jmsMQQueueManagers[1].jmsMQQueueManager: QM2
mq.queues.jmsMQQueueManagers[1].jmsMQAppName: Gapwalk
mq.queues.jmsMQQueueManagers[1].jmsMQChannel: DEV.APP.SVRCONN
mq.queues.jmsMQQueueManagers[1].jmsMQHost: localhost
mq.queues.jmsMQQueueManagers[1].jmsMQPort: 1415
mq.queues.jmsMQQueueManagers[1].jmsMQUserid: app

################################
##### SQL SHIFT CODE POINT ####
################################
# Code point 384 match unicode character \u0180
sqlCodePointShift : 384

################################
##### LOCK TIMEOUT RECORD  ####
```

```
################################
# Blu4IV record lock timeout
lockTimeout : 100

################################
##### REPORTS OUTPUT PATH ####
################################
reportOutputPath: reports

################################
##### TASK EXECUTOR       ####
################################
taskExecutor:
  corePoolSize: 5
  maxPoolSize: 10
  queueCapacity: 50
  allowCoreThreadTimeOut: false

################################
##### PROGRAM NOT FOUND   ####
################################
stopExecutionWhenProgNotFound: false

#########################################################
#####  DISP DEFAULT VALUE (to be removed one day) ####
#########################################################
defaultKeepExistingFiles: true

#########################################################
#####  BLOCKSIZE DEFAULT VALUE  ####
#########################################################
#blockSizeDefault: 32760

#####################################
#####  JOBQUEUE CONFIGURATION    ####
#####################################
jobqueue:
  api.enabled: false
  impl: none # possible values: quartz, none
  schedulers:  # list of schedulers
    -
      name: queue1
      threadCount: 5
    -
```

```
      name: queue2
      threadCount: 5

################################################################################
##### QUERY BUILDING                                                       ##
# useConcatCondition : false by default
# if true, in the query, the where condition is build with key concatenation  ##
################################################################################
# query.useConcatCondition: true

########################################
#####  JCL Batch Restart Mechanism  ####
########################################
jcl:
checkpoint:
enabled: false
#expireTimeout: -1
#expireTimeoutUnit: SECONDS # Supported values: java.util.concurrent.TimeUnit
#provider: redis

----
```

- 搭配 LISTCAT 命令使用可變長度檔案

```
[**/*.*]
encoding=IBM930
reencoding=false

[global]
listcat.variablelengthpreprocessor.enabled=true
listcat.variablelengthpreprocessor.type=rdw
# use "rdw" if your .listcat file contains a set of records (RDW)
# use "bdw" if your .listcat file contains a set of blocks (bdw)
```

- 在 LOAD/UNLOAD 公用程式中提供 Null 位元組指標值

```
# Unload properties
# For date/time: if use database configuration is enabled, formats are ignored
# For nbi; use hexadecimal syntax to specify the byte value
```

```
# - When the value is null in database : the value dumped to the file is filled by
 low value characters and the NBI is
# equal to the byte 6F (the ? character)
# - When the value is not null in database and the column is nullable: the NBI is
 equal to the byte 00 (low value) and NOT
# equal to the byte 40 (space)
unload:
  sqlCodePointShift: 0
  nbi:
    whenNull: "6F"
    whenNotNull: "00"
  useDatabaseConfiguration: false
  format:
    date: MM/dd/yyyy
    time: HH.mm.ss
    timestamp: yyyy-MM-dd-HH.mm.ss.SSSSSS
```

主要應用程式的可用屬性

此表格提供金鑰/值參數的詳盡檢視。

金鑰	Type	預設值	描述	發行版本
logging.config	路徑	classpath：logback-main.xml	日誌傳回組態檔案參考的標準金鑰。其他標準記錄金鑰也可供使用。	
spring.jta.enabled	boolean	false	標準金鑰。如果資料來源支援模式不是靜態 xa，則必須停用彈簧 JTA 交易自動設定。	
datasource.jicsDs+-driver-	具有子索引鍵的標準彈簧資料來源		包含 Jics 資料庫的連線資訊。或者，強烈建	

金鑰	Type	預設值	描述	發行版本
`class-name + -url + -username + -password + -type`			議使用 AWS 秘密，如 中所述the section called “JICS 資料庫”。	
`datasource.bluesamDs + -driver-class-name + -url + -username + -password + -type`	具有子索引鍵的標準彈簧資料來源		包含 Blusam 資料庫的連線資訊。或者，強烈建議使用 AWS 秘密，如 中所述the section called “Blusam 資料庫”。	
`bluesam.disabled`	boolean	false	是否完全停用 Blusam。	
`bluesam.cache`	string		如果未設定，將不會使用 Blusam 快取。可能的值（快取實作）為快取和 redis (the section called “Redis 快取屬性”)。	
`bluesam.maxBluesamDisablingThreadpoolSize`	number	10	指定用於停用 bluesam 資料集以進行批次處理的執行緒集區大小上限。	4.5.0

金鑰	Type	預設值	描述	發行版本
bluesam.bluesamStatusPollingInterval	number	1000	指定輪詢 bluesam 狀態以檢查線上活動時，每次反覆運算之間的等待時間（以毫秒為單位）。	4.5.0
bluesam.maxBluesamStatusPollingRetry	number	3	指定輪詢 bluesam 狀態失敗時的重試次數上限。	4.5.0
bluesam.checkBluesamStatus	boolean	false	指定是否要在存取 bluesam 資料集之前檢查其狀態。	4.5.0
spring.aws.client.bluesam.redis.secret	string	null	指定 Bluesam Redis 快取的登入資料秘密 ARN，請參閱 the section called “AWS BluAge 執行期秘密”。	
spring.aws.client.bluesam.locks.redis.secret	string	null	指定 Bluesam 鎖定 Redis 快取的登入資料秘密 ARN，請參閱 the section called “AWS BluAge 執行期秘密”。	

金鑰	Type	預設值	描述	發行版本
forcedDate	string		如果有日期，強制日期到提供的日期。	
frozenDate	boolean	true	指定是否凍結日期。只有在 也設定 時才forcedDate 適用。	
externalSort.threshold	datasize （範例：12 MB)		排序閾值：何時切換到外部 （合併） 排序。	
blockSizeDefault	number	32760	用於 BDW 位元組的預設區塊大小。	
jics.parameters.datform	string	MMDDYY	日期表單。	

金鑰	Type	預設值	描述	發行版本
`jics.initList`	string		初始化 JICS 清單，以逗號分隔。如果存在，它會定義清單的逗號分隔名稱，以便在 CICS 清單之間的 Apache Tomcat 啟動時啟用。範例值：`$UUU,DFH$IVPL,PEZ1` 。這將串聯到這些清單中包含的群組及其基礎資源定義，然後執行時間將可見這些定義。預設為空白。	
`jics.parameters.applid`	string	VELOCITY	套用以識別 JICS 中的應用程式（至少 4 個字元，無長度上限）。	
`jics.parameters.sysid`	string	CICS	系統識別 (SYSID)。	
`jics.parameters.eibtrmid`	string	TERM	終端機識別符（最多 4 個字元，最少 1 個字元）。	

金鑰	Type	預設值	描述	發行版本
jics.parameters.userid	string		使用者 ID（最多 8 個字元，沒有最小值）。未提供任何值時（預設為空白），HTTP 工作階段 ID 會用作使用者 ID。	
jics.parameters.username	string	MYUSERNAME	使用者名稱（最多 10 個字元，最少 1 個字元）。	
jics.parameters.netname	string	MYNETNAME	網路名稱（最多 8 個字元，最少 1 個字元）。	
jics.parameters.opid	string	XXX	3 字元運算子識別。	
jics.parameters.jobname	string	MJOBNAME	任務名稱。	
jics.parameters.sysname	string	SYSNAME	AS400 系統名稱 (sysname)。	
jics.parameters.cwa.length	number	0	常用工作區域 (CWA) 的長度。	
jics.parameters.charset	string	CP037	JICS 全域使用的字元集。	

金鑰	Type	預設值	描述	發行版本
`jics.parameters.tsqimpl`	string	藍山	JICS 暫時儲存佇列 (TSQ) 實作（允許的值為`bluesam`/`memory`/`redis`）	
`jics.queues.ts.redis.*`	支援的 Redis 屬性		指定 JICS TS Queues Redis 伺服器的組態屬性，請參閱 the section called “支援的 Redis 屬性”。	
`spring.aws.client.jics.queues.ts.redis.secret`	string	null	指定 JICS TS Queues Redis 伺服器的登入資料秘密 ARN，請參閱 the section called “AWS BluAge 執行期秘密”。	
`lockTimeout`	number	500	鎖定逾時，以毫秒為單位。	

金鑰	Type	預設值	描述	發行版本
sqlCodePointShift	number		選用。sql 程式碼點轉移。將舊版 RDBMS 資料遷移至現代 RDBMS 時，我們可能遇到的控制角色的程式碼點轉移。例如，您可以指定384來比對 Unicode 字元\u0180。	
sqlIntegerOverflowAllowed	boolean	false	指定是否允許 SQL 整數溢出，表示是否允許在主機變數中放置較大的值。	
database.cursor.overflow.allowed	boolean	true	指定是否允許游標溢位。將 設定為true ，以對游標執行其位置的下一個呼叫。設定為false ，檢查游標是否位於最後一個位置，再對游標執行下一個呼叫。只有在游標為 SCROLLABLE (SENSITIVE 或 INSENSITIVE) 時，才啟用 。	

金鑰	Type	預設值	描述	發行版本
reportOutputPath	string	/reports	報告輸出路徑。	
spring.session.store-type	string	無	高可用性環境的工作階段快取。可能的值為none或 redis。預設為 none。	
stopExecutionWhenProgNotFound	boolean	true	指定在找不到程式時是否停止執行。如果設定為 true，則如果找不到程式，會中斷執行。	
forceHR	boolean	false	指定要在主控台或檔案輸出上使用人類可讀 SYSPRINT。	
rollbackOnRTE	boolean	false	指定是否要在執行時間例外狀況時轉返隱含執行單位交易。	
sctThreadLimit	long	5	觸發指令碼的執行緒限制。	

金鑰	Type	預設值	描述	發行版本
dataSimplifier.onInvalidNumericData	string	拒絕	解碼無效數值資料時如何反應。允許的值為reject /tolerates paces /tolerates paceslowvalues /toleratemost 。預設為reject。	
filesDirectory	string		批次輸入/輸出檔案的目錄。	
ims.messages.extendedSize	boolean	false	指定是否要在 IMS 訊息上設定擴展大小。	
defaultKeepExistingFiles	boolean	false	指定是否設定資料集預設的先前值。	
jics.db.ddlScriptLocation	string		Jics DDL 指令碼位置。可讓您使用 .sql 指令碼啟動 Jics 資料庫結構描述。預設空白。例如，./jics/sql/jics.sql 。	

金鑰	Type	預設值	描述	發行版本
`jics.db.schemaTestQueryLocation`	string		應包含唯一查詢的 sql 檔案位置，該查詢會傳回 jics 結構描述中的物件數目（如果有的話）。	
`jics.db.dataScriptLocation`	string		定義用於初始化 JICS 資料庫之 SQL 指令碼的路徑。接受以逗號分隔的檔案和目錄清單，允許指定多個指令碼和資料夾。	
`jics.db.dataTestQueryLocation`	string		包含單一 sql 查詢的 sql 指令碼位置，預期會傳回物件計數（例如：計算 jics 程式資料表中的記錄數）。如果計數等於 0，則會使用`jics.db.dataScriptLocation` 指令碼載入資料庫，否則會略過資料庫載入。	

金鑰	Type	預設值	描述	發行版本
jics.data.dataJsonInitLocation	string			
jics.xa.agent.timeout	number			
query.useConcatCondition	boolean	false	指定金鑰條件是否由金鑰串連建置。	
system.qdecfmt	string			
disposition.checkexistence	boolean	false	指定是否針對具有 DISP SHR 或 OLD 的資料集，釋出檔案存在的檢查。	
useControlMVariable	boolean	false	指定是否使用 control-M 規格取代變數。	
card.encoding	string	CP1145	卡片編碼：要與 搭配使用useControlMVariable 。	
mapTransfo.prefixes	string	&,@,%%	轉換 controlM 變數時要使用的字首清單。每個以逗號分隔。	

金鑰	Type	預設值	描述	發行版本
checkinputfilesize	boolean	false	指定是否要在檔案大小為記錄大小的倍數時釋出檢查。	
stepFailWhenAbend	boolean	true	指定在步驟失敗或完成執行時是否引發無效。	
bluesam.fileLoading.commitInterval	number	100000	bluesam 遞交間隔。	
uppercaseUserInput	boolean	true	指定使用者輸入是否為大寫。	
jhdb.lterm	string		在 IMS 模擬的情況下，可讓您強制使用常見的邏輯終端機 ID。如果未設定，則會使用 sessionId。	
jhdb.identificationCardData	string		用來將一些「運算子識別卡資料」硬式編碼至 CARD 參數指定的 MID 欄位。根據預設，空白，無輸入限制。	

金鑰	Type	預設值	描述	發行版本
encoding	string	ASCII	專案中使用的編碼（非 groovy 檔案）。預期有效的編碼CP1047IBM930.	
cl.configuration.context.encoding	string	CP297	CL 檔案的編碼。預期有效的編碼CP1047IBM930. 預設值為CP297	
cl.zonedMode	string	EBCDIC_STRICT	編碼或解碼控制語言 (CL) 命令的模式。允許的值為EBCDIC_STRICT /EBCDIC_MODIFIED /AS400。	
ims.programs	string		要使用的 IMS 程式清單。以分號 (;) 分隔每個參數，並以逗號 () 分隔每個交易,。例如：PCP008,PCT008;PCP054,PCT054;PCP066,PCT066;PCP068,PCT068;	

金鑰	Type	預設值	描述	發行版本
jhdb.configuration.context.encoding	string	CP297	JHDB (Java 階層式資料庫) 編碼。預期有效的編碼字串CP1047IBM930.	
jhdb.metadata.extrapath	string	file：./setup/	組態參數，指定 psbs 和 dbds 資料夾的額外執行時間特定根資料夾。	

金鑰	Type	預設值	描述	發行版本
jhdb.checkpointPersistence	string	無	檢查點持續性模式。允許的值為none /add /end。使用add在建立新的檢查點並新增至登錄檔時保留檢查點。在伺服器關閉時，使用end過於持久的檢查點。任何其他值都會停用持久性。請注意，每次將新的檢查點新增至登錄檔時，所有現有的檢查點都會序列化，並且檔案也會清除。它不是附加到 檔案中的現有資料。因此，根據檢查點的數量，它可能會對效能產生一些影響。	

金鑰	Type	預設值	描述	發行版本
jhdb.checkpointPath	string	file：./setup/	如果jhdb.checkpointPersistence 不是none，則此參數可讓您設定檢查點持久性路徑 (checkpoint.dat 檔案儲存位置），則登錄檔中包含的所有檢查點資料都會序列化，並備份在提供的資料夾中的檔案 (checkpoint.dat) 中。請注意，只有檢查點資料 (scriptId、stepId、資料庫位置和檢查點區域） 會考慮此備份。	
jhdb.navigation.cachenexts	number	5000	RDBMS 階層導覽中使用的快取持續時間 （以毫秒為單位）。	
jhdb.use-db-prefix	boolean	true	指定是否在 RDBMS 的階層導覽中啟用資料庫字首。	

金鑰	Type	預設值	描述	發行版本
`jhdb.query.limitJoinUsage`	boolean	true	指定是否在 RDBMS 圖形上使用限制聯結用量參數。	
`taskExecutor.corePoolSize`	number	5	當終端機中的交易透過 groovy 指令碼啟動時，會建立新的執行緒。使用此參數來設定核心集區大小。	
`taskExecutor.maxPoolSize`	number	10	當終端機中的交易透過 groovy 指令碼啟動時，會建立新的執行緒。使用此參數來設定最大集區大小（平行執行緒的數量上限）。	
`taskExecutor.queueCapacity`	number	50	當終端機中的交易透過 groovy 指令碼啟動時，會建立新的執行緒。使用此參數來設定佇列大小。(= 達到時`taskExecutor.maxPoolSize` 待處理交易的最大數量）	

金鑰	Type	預設值	描述	發行版本
taskExecutor.allowCoreThreadTimeOut	boolean	false	指定是否允許核心執行緒在 JCIS 中逾時。這可讓動態成長和縮減，即使與非零佇列結合（因為最大集區大小只會在佇列已滿時成長）。	
jics.runUnitLauncherPool.enable	boolean	false	指定是否要在 JICS 中啟用執行單位啟動器集區。	
jics.runUnitLauncherPool.size	number	20	JICS 中的執行單位啟動器集區大小。	
jics.runUnitLauncherPool.validationInterval	number	1000	調整集區大小之任務每次執行之間的間隔。	
jics.runUnitLauncherPool.parallelism	number	2	調整任務執行時，用於在佇列中產生遺失執行個體的執行緒數目。	
context.preconstruct.enable	boolean	false	指定是否啟用程式內容的預先建構。	

金鑰	Type	預設值	描述	發行版本
`context.preconstruct.frequencyInMillis`	number	100	調整集區大小之任務每次執行之間的間隔。	
`context.preconstruct.parallelism`	number	5	調整任務執行時，用於佇列中產生遺失執行個體的執行緒數目。	
`context.preconstruct.minInstances`	number	2	第一次需要內容時要建立的執行個體數量。	
`spring.aws.application.credentials`	string	null	從 JICS 中的 AWS 登入資料設定檔載入登入資料。	
`jics.queues.sqs.region`	string	eu-west-1	Amazon Simple Queue Service AWS 的區域，用於 JICS。	
`jics.jcl.rt.encoding`	string	CP037	在專用 JICS 佇列中寫入的 JCL 指令碼編碼。	
`jics.jcl.rt.queue`	string	JICS	在執行時間，JCL 指令碼可以逐行寫入的佇列名稱。	

金鑰	Type	預設值	描述	發行版本
mq.queues.sqs.region	string	eu-west-3	AWS SQS MQ 服務的 AWS 區域。	
quartz.scheduler.stand-by-if-error	boolean	false	指定是否在任務排程器處於待命模式時觸發任務執行。如果為 true，則啟用任務執行時不會觸發。	
databaseStatistics	boolean	false	指定是否允許 SQL 建置器收集和顯示統計資料資訊。	
dbDateFormat	string	yyyy-MM-dd	db 目標日期格式。	
dbTimeFormat	string	HH：mm：ss	db 目標時間格式。	
dbTimestampFormat	string	yyyy-MM-dd HH：mm：ss.SSSSSS	db 目標時間戳記格式。	
dateTimeFormat	string	ISO	dateTimeFormat 描述如何將資料庫日期時間戳記類型溢出至資料簡化器實體。允許的值為ISO /EUR /EUR /USA /LOCAL	

金鑰	Type	預設值	描述	發行版本
localDateFormat	string		本機日期格式清單。使用 分隔每個格式\。	
localTimeFormat	string		本機時間格式的清單。使用 分隔每個格式\	
localTimestampFormat	string		本機時間戳記格式的清單。使用分隔每個格式\。	
pgmDateFormat	string	yyyy-MM-dd	日期時間格式。	
pgmTimeFormat	string	HH.mm.ss	用於 pgm （程式） 執行的時間格式。	
pgmTimestampFormat	string	yyyy-MM-dd-HH.mm.ss.SSSSSS	時間戳記格式。	
cacheMetadata	boolean	true	指定是否快取資料庫中繼資料。	
forceDisableSQLTrimStringType	boolean	false	指定是否停用所有 sql 字串參數的修剪。	
fetchSize	number		游標的 fetchSize 值。依載入/卸載 utils 使用區塊擷取資料時使用 。	

金鑰	Type	預設值	描述	發行版本
check-groovy-file	boolean	true	指定是否要在註冊之前檢查 groovy 檔案內容。	
qtemp.uuid.length	number	9	QTEMP 唯一 ID 長度。	
qtemp.dblog	boolean	false	是否啟用 QTEMP 資料庫記錄。	
qtemp.cleanup.threshold.hours	number	0	指定啟用 的時間qtemp.dblog 。db 分割區生命週期（以小時為單位）。	
sort.function	string		blu4iv 資料庫的排序函數名稱。	
invalidDataTolerence	boolean	true	指定是否容忍封裝類型的無效資料。	
program.timeout	number	-1	指定任何程式/交易執行的逾時，以秒為單位。在此期間之後，系統會嘗試中斷程式。	

金鑰	Type	預設值	描述	發行版本
gapwalk.line.separator	string	null	在 gapwalk 中指定行分隔符號類型。允許的值為 WIN (CRLF) / UNIX (LF) / LINUX (LF)。系統會忽略其他值，並使用 System line.separator 屬性。	
enableActivePgmIdCache	boolean	false	指定是否啟用作用中的程式 ID 本機快取。請謹慎使用此功能，因為 JICS 資源可以在程式和使用者之間共用。任何管理員都可以在外部變更這些資源，且放置的本機快取可能會失效。	

金鑰	Type	預設值	描述	發行版本
mq.queues.default.syncpoint	boolean	false	指定未設定 MQPMO_SYNCPOINT 或 MQPMO_NO_SYNCPOINT 時 MQ PUT 命令的預設行為。設為 true 時，它會做為 ，MQPMO_SYNCPOINT 且訊息不會在 PUT 命令期間直接遞交。設定為 false 時，它會做為 ，MQPMO_NO_SYNCPOINT 並在 PUT 命令期間直接遞交訊息。	
dataSimplifier.byteRangeBoundsCheck	boolean	false	設為 true 時，可確保不會使用不正確的值建立 ByteRange。預設值為 false。	
file.stdoutIntoLogger	boolean	false	指定是否要在預設 SYSPRINT和 SYSPUNCH 檔案中啟用寫入記錄器，而非預設系統輸出串流。	
tempFilesDirectory	string	null	指定產生的暫存檔案的資料夾位置名稱。	

金鑰	Type	預設值	描述	發行版本
cleanTempFilesDirectoryAtStartup	boolean	true	指定是否要在應用程式啟動時清除暫存檔案資料夾的內容。	

金鑰	Type	預設值	描述	發行版本
tempFolderPattern	string	null	指定模式，用於根據下列預先定義且可自訂的資訊，動態建置暫時資料夾的名稱。 HOST：主機名稱。 JOBID：任務的 ID。 HASHCODE：任務內容的雜湊碼。 TIMESTAMP：取得時間戳記時要使用的模式。暫時資料夾的目標名稱為 TMP_DIR_{tempFolderPattern}。例如，在下列模式的情況下，名稱將以任務 ID 開頭，並以「時間戳記」：tempFolderPattern：JOBID，HOST=xxxx，HASHCODE，TIMESTAMP=yyyymm	

金鑰	Type	預設值	描述	發行版本
			ddhhmmsss 結尾。如果屬性tempFolderPattern 未新增至 YAML 檔案或為空，則暫時資料夾的名稱將為 "TMP_DIR_" + this.hashCode() (DefaultJobContext)。	
database.cursor.raise.already.opened.error	boolean	false	指定當已開啟的游標開啟時是否啟用提升 SQLCODE 錯誤 502。	
jics.spool.smtp.hostname	string	null	指定 SMTP 伺服器主機。範例：smtp.xxx.com	
jics.spool.smtp.port	string	null	指定 SMTP 伺服器連接埠。範例：25	
jics.spool.smtp.password	string	null	指定 SMTP 伺服器的登入密碼。	
jics.spool.smtp.username	string	null	指定 SMTP 伺服器的使用者名稱。	

金鑰	Type	預設值	描述	發行版本
jics.spool.smtp.debug	boolean	false	指定 SMTP 伺服器的偵錯模式。	
gapwalk-application.security	string	disabled	切換全域安全組態 (XSS、CORS、CSRF、OAUTH 身分驗證...)。允許的值為 disabled 和 enabled。	
gapwalk-application.identity	string	null	全域身分驗證方法。建議值為 oauth。允許的值為 json 和 oauth。當 gapwalk-application.security 為時，此選項為必要項目enabled。	
gapwalk-application.security.issuerUri	string	null	身分提供者 (IdP) 的發行者 URI。當 gapwalk-application.identity 為時，此選項為必要項目oauth。	

金鑰	Type	預設值	描述	發行版本
gapwalk-application.security.allowedOrigins	string【】	null	要允許的原始伺服器清單。此選項gapwalk-application.identity 需要設定為oauth。	
gdgDirectoryPath	string	output/gdg	GDG 目錄路徑是存放 gdg 檔案的目錄。	4.6.0
gapwalk-application.security.claimGroupName	string	cognito:groups	包含使用者所屬所有群組清單的宣告屬性。cognito:groups 用於 Amazon Cognito，或外部 IdP 的任何其他字串。	
gapwalk-application.security.userAttributeName	string	username	用於識別使用者請求的宣告屬性名稱。將 username 用於 Amazon Cognito、preferı_username 將用於 Keycloak，或將 用於外部 IdP 的任何其他字串。	

金鑰	Type	預設值	描述	發行版本
gapwalk-application.security.localhostWhitelistingEnabled	boolean	true	指定是否從任何localhost 請求啟用身分驗證。	
gapwalk-application.defaultSuperAdminUserName	string	sadmin	當 gapwalk-application.security 停用時，指定預設的本機超級使用者名稱。	
gapwalk-application.defaultSuperAdminUserPwd	string	sadmin	當 gapwalk-application.security 停用時，指定預設的本機超級使用者密碼。	
gapwalk-application.security.filterURIs	string	disabled	切換篩選 URIs 組態。允許的值為 disabled 和 enabled。	

金鑰	Type	預設值	描述	發行版本
gapwalk-application.security.blockedURIs	string【】	null	要封鎖的 URIs 清單。當 gapwalk-application.security.filterURIs 為時，此選項為必要enabled。	
jics.redis.*	支援的 Redis 屬性		指定 JICS Redis 伺服器連線工廠的組態屬性，請參閱 the section called “支援的 Redis 屬性”。	
spring.aws.client.jics.redis.secret	string	null	指定 JICS Redis 伺服器連線工廠的登入資料秘密 ARN，請參閱 the section called “AWS BluAge 執行期秘密”。	

金鑰	Type	預設值	描述	發行版本
jcl.checkpoint.enabled	boolean	false	指定是否啟用 JCL 檢查點機制以允許任務重新啟動。JCL 檢查點會在每個步驟或主要程式調用開始時建立並儲存至記憶體內登錄檔。如果定義持久性提供者，則所有步驟層級檢查點都會保留在任務結束時。	
jcl.checkpoint.expireTimeout	number	-1	指定在持久性提供者或記憶體內登錄檔中保留 JCL 檢查點的持續時間。	
jcl.checkpoint.expireTimeoutUnit	string	秒	指定 jcl.checkpoint.expireTimeout 屬性的持續時間單位。支援的列舉常數值：java.util.concurrent.TimeUnit。	
jcl.checkpoint.provider	string	null	指定 JCL 檢查點機制持續性提供者。允許的值為 redis。	

金鑰	Type	預設值	描述	發行版本
`jcl.checkpoint.redis.*`	支援的 Redis 屬性		指定 JCL 檢查點機制 REDIS 持續性提供者的組態屬性，請參閱 the section called “支援的 Redis 屬性”。	
`spring.aws.client.jcl.checkpoint.redis.secret`	string	null	指定 JCL 檢查點機制 Redis 持久性提供者的登入資料秘密 ARN，請參閱 the section called “AWS BluAge 執行期秘密”。	
`gapwalk.ssl.enabled`	boolean	false	如果應用程式啟動時尚未設定，則表示將下列`gapwalk.ssl.*` 屬性設定為目前的 JVM 系統屬性。	
`gapwalk.ssl.trustStore`	string	null	`javax.net.ssl.trustStore` 如果應用程式啟動時尚未設定 ，請將值設定為系統屬性。	

金鑰	Type	預設值	描述	發行版本
gapwalk.ssl.trustStorePassword	string	null	javax.net.ssl.trustStorePassword 如果應用程式啟動時尚未設定 ，請將值設定為系統屬性。或者，強烈建議使用 AWS 秘密，如 中所述the section called “SSL 密碼設定的秘密管理員”。	
gapwalk.ssl.trustStoreType	string	null	javax.net.ssl.trustStoreType 如果應用程式啟動時尚未設定 ，請將 值設定為系統屬性。	
gapwalk.ssl.keyStore	string	null	javax.net.ssl.keyStore 如果應用程式啟動時尚未設定 ，請將 值設定為系統屬性。	

金鑰	Type	預設值	描述	發行版本
gapwalk.ssl.keyStorePassword	string	null	javax.net.ssl.keyStorePassword 如果應用程式啟動時尚未設定 ，請將 值設定為系統屬性。或者，強烈建議使用 AWS 秘密，如 中所述the section called “SSL 密碼設定的秘密管理員”。	
mq.queues	string	sqs	指定sqs使用 Amazon SQS、rabbitmq使用內部部署 Rabbit MQ 或使用內部部署 IBMMQ 時，要在 jms之間使用的支援佇列緩衝器。	

金鑰	Type	預設值	描述	發行版本
mq.queues.jmsMQQueueManagers[N]			當 mq.queues 為 時jms，可讓 指定 IBM MQ 連線清單。mq.queues.jmsMQQueueManagers[0] 代表第一個連線，第二個mq.queues.jmsMQQueueManagers[1] 為 ，以此類推。	
mq.queues.jmsMQQueueManagers[N].jmsMQQueueManager	string	null	IBMMQ 佇列管理員名稱。	
mq.queues.jmsMQQueueManagers[N].jmsMQAppName	string	null	IBMMQ 應用程式名稱。	
mq.queues.jmsMQQueueManagers[N].jmsMQChannel	string	null	IBMMQ 頻道名稱。	

金鑰	Type	預設值	描述	發行版本
mq.queues.jmsMQQueueManagers[N].jmsMQHost	string	null	IBMMQ 主機名稱。	
mq.queues.jmsMQQueueManagers[N].jmsMQPort	number	null	IBMMQ 連接埠。	
mq.queues.jmsMQQueueManagers[N].jmsMQUserid	string	null	IBMMQ 使用者名稱。	
mq.queues.jmsMQQueueManagers[N].jmsMQPassword	string	null	IBMMQ 使用者密碼。或者，強烈建議使用 AWS 秘密，如 中所述the section called “IBM MQ 密碼設定的秘密管理員”。	
mq.queues.jmsMQQueueManagers[N].jmsMQMaxPoolSize	number	0	IBMMQ 集區大小上限 。使用 0 時，會啟用無限數量的實體連線。	

金鑰	Type	預設值	描述	發行版本
mq.queues.jmsMQQueueManagers[N].jmsMQSSLCipher	string	null	IBMMQ SSL 密碼套件。範例可以是 "*TLS12OR HIGHER" 。如需詳細資訊，請參閱適用於 JMS 的 IBM MQ 類別中的官方文件 TLS CipherSpecs 和 CipherSuites。	
mq.queues.non.jms.client	boolean	false	指出傳送訊息的目標用戶端是否為非 JMS。原生 MQ 格式將用於非 JMS 用戶端，而 RFH2 格式將用於 JMS。	4.5.0
			當 mq.queues 為時rabbitmq，IBM主機名稱。	
mq.queues.rabbitMQHost			Rabbit MQ 主機名稱。	
mq.queues.rabbitMQVirtualHost			Rabbit MQ 虛擬主機名稱。	

金鑰	Type	預設值	描述	發行版本
mq.queues.rabbitMQPort			Rabbit MQ 連接埠。	
mq.queues.rabbitMQUsername			Rabbit MQ 使用者。	
mq.queues.rabbitMQPassword			Rabbit MQ 密碼。	
mf.runtime.switch.N	boolean	true	為 MF 自然行序列檔案啟用 null 插入。	4.4.0
mf.runtime.switch.T	boolean	false	啟用在 MF 自然行序列檔案中插入標籤字元。	4.4.0
gapwalk.database.support.useSavePointToRestoreFail	boolean	false	在插入查詢上使用儲存點，在失敗時啟用交易復原。啟用此屬性可能會影響資料庫效能。您可以使用query-to-database映射組態覆寫特定查詢的此設定。	4.6.0

選用 Web 應用程式的可用屬性

視您的現代化應用程式而定，您可能需要設定一或多個選用的 Web 應用程式，這些應用程式代表對 z/OS、AS/400 或 IMS/MFS 等相依性的支援。下表包含用於設定每個選用 Web 應用程式的可用金鑰/值參數清單。

gapwalk-utility-pgm.war

此選用的 Web 應用程式包含對 Z/OS 公用程式的支援。

此表格提供此應用程式金鑰/值參數的詳盡檢視。

金鑰	Type	預設值	描述	發行版本
`logging.config`	路徑	classpath：logback-utility.xml	日誌傳回組態檔案參考的標準金鑰。您也可以使用其他標準日誌金鑰。	
`spring.jta.enabled`	boolean	false	標準金鑰。如果資料來源支援模式不是靜態 xa，則必須停用彈簧 JTA 交易自動組態。	
`spring.datasource.primary.jndi-name`	string	jdbc/primary	如果使用 JNDI，則主要資料來源的 JNDI 名稱 (Java 命名和目錄界面）。	
`primary.datasource -driver-class-name -url -username -password`	具有子索引鍵的標準彈簧資料來源		如果不使用 JNDI，則包含應用程式資料庫的連線資訊。必須具有與現代化應用程式 YAML 檔案中相同的組態。 或者，強烈建議使用 AWS 秘密，如 中所述the	

金鑰	Type	預設值	描述	發行版本
			section called “用戶端資料庫”。	
encoding	string	ASCII	公用程式中使用的編碼。預期有效的編碼CP1047、IBM930	
sysPunchEncoding	string	ASCII	syspunch 編碼字元集。預期有效的編碼CP1047IBM930	
systin.encoding	string	ASCII	SYSTIN 檔案資料集的編碼字元集。預期有效的編碼CP1047IBM930	4.5.0
zonedMode	string	EBCDIC_STRICT	編碼或解碼分區資料類型的模式。允許的值為EBCDIC_STRICT /EBCDIC_MODIFIED /AS400。	
idcams.encoding.forced	string		IDCAMS 公用程式程式中使用的編碼。預期有效的編碼CP1047、IBM930	4.4.0
unload.chunkSize	number	0	用於卸載公用程式的區塊大小。	

金鑰	Type	預設值	描述	發行版本
unload.computeRecordSizeIfNull	boolean	false	決定是否在未指定時計算記錄大小。如果指定，則值保持不變。	
unload.sqlCodePointShift	number	0	卸載公用程式的 SQL 程式碼指標轉移。執行轉移字元程序。當您來自 DB2 的目標資料庫為 Postgresql 時為必要。	
unload.columnFiller	string	空格	卸載公用程式資料欄填充。	
unload.varCharIsNull	boolean	false	在 INFTILB 程式中使用此參數，如果設定為true，則所有具有空白（空格）值的不可為 null 欄位都會傳回空字串。	
unload.useDatabaseConfiguration	boolean	false	指定是否要在卸載公用程式中使用 application-main.yml 的日期或時間組態。	

金鑰	Type	預設值	描述	發行版本
unload.format.date	string	MM/dd/yyyy	如果unload.useDatabaseConfiguration 已啟用 ，則在卸載公用程式中使用的日期格式。	
unload.format.time	string	HH.mm.ss	如果unload.useDatabaseConfiguration 已啟用 ，則在卸載公用程式中使用的時間格式。	
unload.format.timestamp	string	yyyy-MM-dd-HH.mm.ss.SSSSSS	如果unload.useDatabaseConfiguration 已啟用 ，則在卸載公用程式中使用的時間戳記格式。	
unload.nbi.whenNull	十六進位	6F	當資料庫的值為 Null 時，要新增的 Null Byte Indicator (NBI) 值。	
unload.nbi.whenNotNull	十六進位	00	當來自資料庫的值不是 null 時，要新增的 Null 位元組指標 (NBI) 值。	

金鑰	Type	預設值	描述	發行版本
unload.nbi.writeNullIndicator	boolean	false	指定是否要在卸載輸出檔案中寫入 null 指標。	
unload.bmc.useInto	boolean	false	指定是否處理卸載公用程式的 INTO bmc 控制關鍵字。	
unload.fetchSize	number	0	可讓您在卸載公用程式中處理游標時調整擷取大小。	
unload.noPad	boolean	true	指出變數長度字元 (VARCHAR) 欄位要卸載，而不會填補到長度上限。	4.5.0
treatLargeNumberAsInteger	boolean	false	指定是否要將大型數字視為 Integer。根據預設，它們會被視為 BigDecimal 。	
load.batchSize	number	0	負載公用程式批次大小。	
load.format.localDate	string	dd.MM.yyyy\dd/MM/yyyy\yyyy-MM-dd	要使用的負載公用程式本機日期格式。	

金鑰	Type	預設值	描述	發行版本
`load.format.localTime`	string	HH：mm：ss\HH.mm.ss	要使用的負載公用程式本機時間格式。	
`load.format.dbDate`	string	yyyy-MM-dd	要使用的負載公用程式資料庫格式。	
`load.format.dbTime`	string	HH：mm：ss	要使用的負載公用程式資料庫時間。	
`load.sqlCodePointShift`	number	0 秒	負載公用程式的 SQL 程式碼指標轉移。執行轉移字元程序。當您來自 DB2 的目標資料庫為 Postgresql 時為必要。	
`load.applyRollback`	boolean	false	將此參數設定為 ，`true`表示您希望服務在將資料載入資料庫時發生錯誤時轉返資料表變更。	
`forcedDate`	string		如果有日期，則強制日期到提供的日期。	

金鑰	Type	預設值	描述	發行版本
frozenDate	boolean	true	指定是否凍結日期。只有在 也設定 時才forcedDate 適用。	
jcl.type	string	mvs	.jcl 檔案類型。允許的值為jcl/vse。如果非 vse jcl 的檔案為空，IDCAMS 公用程式 PRINT/REPRO 命令會傳回 4。	
hasGraphic	boolean	false	INFUTILB 公用程式是否需要處理 GRAPHIC DB2 資料欄。	
convertGraphicDataToFullWidth	boolean	true	指定是否要將圖形資料轉換為完整寬度格式。	

gapwalk-cl-command.war

此選用 Web 應用程式包含 AS/400 公用程式的支援。

此表格提供此應用程式金鑰/值參數的詳盡檢視。

金鑰	Type	預設值	描述
logging.config	路徑	classpath：logback-utility.xml	日誌傳回組態檔案參考的標準金鑰。您也

金鑰	Type	預設值	描述
			可以使用其他標準日誌金鑰。
spring.jta.enabled	boolean	false	標準金鑰。如果資料來源支援模式不是靜態 xa，則必須停用彈簧 JTA 交易自動組態。
spring.datasource.primary.jndi-name	string	jdbc/primary	如果使用 JNDI，則主要資料來源的 JNDI 名稱 (Java 命名和目錄界面）。
primary.datasource + -driver-class-name + -url + -username + -password	具有子索引鍵的標準彈簧資料來源		如果不使用 JNDI，則包含應用程式資料庫的連線資訊。必須具有與現代化應用程式 YAML 檔案中相同的組態。 或者，強烈建議使用 AWS 秘密，如 中所述the section called “用戶端資料庫”。
encoding	string	ASCII	公用程式中使用的編碼。預期有效的編碼 CP1047IBM930、ASCIIUTF-
zonedMode	string	EBCDIC_STRICT	編碼或解碼分區資料類型的模式。允許的值為EBCDIC_STRICT /EBCDIC_MODIFIED /AS400。

金鑰	Type	預設值	描述
commands-off	string		要關閉的命令清單，以逗號分隔。允許的值為PGM_BASIC 、RCVMSGSNDRCVF、CHGVAR 當您想要停用或覆寫現有程式時很有用。 PGM_BASIC 是專為偵錯目的設計的特定 AWS Blu Age Runtime 程式。
forcedDate	string		如果有日期，則強制日期到提供的日期。

gapwalk-hierarchical-support.war

這個選用的 Web 應用程式包含 IMS/MFS 交易支援。

此表格提供此應用程式金鑰/值參數的詳盡檢視。

金鑰	Type	預設值	描述
logging.config	路徑	classpath：logback-utility.xml	日誌傳回組態檔案參考的標準金鑰。您也可以使用其他標準日誌金鑰。
spring.jta.enabled	boolean	false	標準金鑰。如果資料來源支援模式不是靜態 xa，則必須停用彈簧 JTA 交易自動組態。
jhdb.configuration	string		JHDB (Java 階層式資料庫） 編碼。

金鑰	Type	預設值	描述
.context.encoding			預期有效的編碼字串CP1047IBM930、ASCII、U
jhdb.checkpointPersistence	string	無	檢查點持久性模式。允許的值為none /add /end。使用add 在建立新的檢查點並新增至登錄檔時保留檢查點。在伺服器關閉時，使用end過於持久的檢查點。任何其他值都會停用持久性。請注意，每次將新的檢查點新增至登錄檔時，所有現有的檢查點都會序列化，並清除檔案。它不是附加到 檔案中的現有資料。因此，根據檢查點的數量，它可能會對效能產生一些影響。

用戶端應用程式的可用屬性

您的現代化應用程式可能需要用戶端 Spring 應用程式的特定屬性組態。這些屬性會初始化執行時間 JAR 檔案中封裝之類別的 Bean。在應用程式產生期間設定設定檔值application-profile.yaml的檔案，可讓您設定這些屬性。下表列出可用於設定用戶端 Web 應用程式的金鑰/值參數，該應用程式使用來自封裝在 Gapwalk 執行時間中的類別的 Bean

金鑰	Type	預設值	描述	發行版本
blu4iv.dtaara.libr	boolean	false	在資料區域操作的內容中控制程式庫的使用。如	4.5.0

金鑰	Type	預設值	描述	發行版本
ary.disab le			果設為 true，則會針對資料區域操作停用程式庫用量，但這不會影響 QTemp 的使用。如果設定為 false，則在執行資料區域的 CRUD 操作時，會考慮程式庫。	

AWS Blu Age Runtime 中的可用 Redis 快取屬性

您可以使用本文件來了解 AWS Blu Age Runtime 中的 Redis 快取，以及 Gapwalk 組態、支援的 Redis 屬性，以及application-main.yml檔案如何參考 Redis 快取的秘密 ARN。

AWS 藍卡存留期執行期中的 Redis 快取

Redis 伺服器可以用作 AWS Blu Age Gapwalk 應用程式中各種功能的快取，例如：

AWS 使用 Redis 快取的 Blu Age Runtime 功能	描述
Blusam 快取	一種 Redis Blusam 快取，使用先寫策略來有效讀取記錄，以最佳化批次承載上遇到的寫入密集型工作負載。
Blusam 鎖定	資料集和記錄的分散式鎖定快取。
資料集目錄	目錄資料集快取。
工作階段快取	HttpSession 的 Redis 快取。快取會儲存使用者名稱、與角前端的對話狀態，以及特定的「對話」(BMS、MFS、AS400) 資訊。
工作階段追蹤器	作用中工作階段的快取，其中包含相關聯的使用者名稱和session-creation-time資訊。

AWS 使用 Redis 快取的 Blu Age Runtime 功能	描述
JICS 快取	JICS 資源定義的快取。
TS 佇列	TS 佇列的儲存體。
JCL 檢查點	JCL 檢查點快取。
Gapwalk 檔案鎖定	依任務的分散式檔案鎖定快取。
Blu4iv 鎖定	Blu4iv 記錄鎖定的儲存體。

Redis Gapwalk 組態

如果 `redis` 指定為快取機制，且未針對特定功能提供 Redis 組態，則會使用全域 Redis 組態。此組態可讓您同時針對多個 Redis 快取使用相同的組態。

在下列範例中，Blusam 資料集快取和 JICS 快取使用 `gapwalk.redis(redis.server1)` 組態，因為其快取類型設定為 `redis`，而且 the section called “JICS 資源定義”和 下未指定隱含 Redis 屬性the section called “JICS 資源定義”。不過，由於已明確定義其 Redis 屬性，因此 Blusam 鎖定快取將使用不同的 Redis 組態 (`redis.server2`)。

```
...

 gapwalk:
   redis:
     hostName: redis.server1
  port: 6379
 ...

 bluesam:
   # Redis bluesam cache
   cache: redis
   # Redis locks cache
   locks:
     cache: redis
  hostName: redis.server2
  port: 6379
 ...
 # Redis jics cache
 jics:
```

```
  resource-definitions:
    store-type: redis
...
```

若要啟用全域 Redis 組態，請在 中新增下列組態`main-application.yml`。

```
gapwalk:
  redis:
    hostName: localhost
    port: 6379
    mode: standalone                          # Optional
    username:                                 # Optional
    password: ""                              # Optional
    useSsl: false                             # Optional
    database: 0                               # Optional
    maxTotal: 128                             # Optional
    maxIdle: 128                              # Optional
    minIdle: 16                               # Optional
    testOnBorrow: true                        # Optional
    testOnReturn: true                        # Optional
    testWhileIdle: true                       # Optional
    testOnCreate: true                        # Optional
    minEvictableIdleTimeMillis: 60000         # Optional
    timeBetweenEvictionRunsMillis: 30000      # Optional
    numTestsPerEvictionRun: -1                # Optional
    blockWhenExhausted: true                  # Optional
    nettyThreads: 32                          # Optional
    subscriptionsPerConnection: 10            # Optional
    subscriptionConnectionPoolSize: 100       # Optional
    pageSizeInBytes: 8192                     # Optional
    readTimeout: 2000                         # Optional
```

支援的 Redis 屬性

下表顯示 AWS Blu Age Runtime 上全域和特定 Redis 快取支援的 Redis 屬性。

屬性名稱	是否為必要？	描述	值	預設
`mode`	否	Redis 執行模式。	`standalone | cluster`	`standalone`

屬性名稱	是否為必要？	描述	值	預設
hostname	是	Redis 伺服器的主機名稱或 IP 地址。	string	null
port	是	Redis 伺服器接聽連線的連接埠號碼。	int	null
username	否	身分驗證的使用者名稱。	string	null
password	否	身分驗證的密碼。	string	空白字串
useSsl	否	指定是否啟用 Redis 連線的 SSL/TLS 加密。	boolean	false
database	否	要使用的 Redis 資料庫號碼。Redis 支援多個邏輯資料庫，此屬性會指定要使用哪個資料庫。	int	0
maxTotal	否	Redis 連線集區中允許的連線數目上限。	int	128
maxIdle	否	Redis 連線集區中允許的閒置連線數目上限。	int	128

屬性名稱	是否為必要？	描述	值	預設
minIdle	否	要在 Redis 連線集區中維護的閒置連線數目下限。	int	16
testOnBorrow	否	布林值，指出是否要在從集區借用連線之前驗證連線。	boolean	true
testOnReturn	否	布林值，指出是否要在將連線傳回集區之前驗證連線。	boolean	true
testWhileIdle	否	布林值，指出是否定期驗證集區中的閒置連線。	boolean	true
testOnCreate	否	布林值，指出是否要在建立連線時驗證連線。	boolean	true
minEvictableIdleTimeMillis	否	閒置連線必須保留在集區中才能移出的最短時間量（以毫秒為單位）。	long	60000L
timeBetweenEvictionRunsMillis	否	閒置連線偵測器執行緒連續執行之間的時間（以毫秒為單位）。	long	30000L

屬性名稱	是否為必要？	描述	值	預設
numTestsPerEvictionRun	否	每次執行閒置連線偵測器執行緒時要測試的連線數目上限。	int	-1
blockWhenExhausted	否	布林值，指出是否要在集區用盡時封鎖和等待連線變成可用。	boolean	true
nettyThreads	否	用於處理 Redis 連線的 Netty 執行緒數量。	int	32
subscriptionsPerConnection	否	每個 Redis 連線允許的訂閱數量上限。	int	10
subscriptionConnectionPoolSize	否	Redis 訂閱連線集區中允許的連線數目上限。	int	100
pageSizeInBytes	否	Redis 操作的預設頁面大小，以位元組為單位。	long	262144000
readTimeout	否	Redis 操作的讀取逾時，以毫秒為單位。	long	2000

屬性名稱	是否為必要？	描述	值	預設
timeToLiveMillis	否	快取項目在被視為過期和移除之前保留在快取中的持續時間（以毫秒為單位）。如果未指定此屬性，快取項目預設不會自動過期。	long	-1

Redis 快取屬性

Redis Blusam 快取

```
bluesam:
   cache: redis
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
   redis:
     hostName: localhost
     port: 6379
     mode: standalone                        # Optional
     username:                               # Optional
     password: ""                            # Optional
     useSsl: false                           # Optional
     database: 0                             # Optional
     maxTotal: 128                           # Optional
     maxIdle: 128                            # Optional
     minIdle: 16                             # Optional
     testOnBorrow: true                      # Optional
     testOnReturn: true                      # Optional
     testWhileIdle: true                     # Optional
     testOnCreate: true                      # Optional
     minEvictableIdleTimeMillis: 60000       # Optional
     timeBetweenEvictionRunsMillis: 30000    # Optional
     numTestsPerEvictionRun: -1              # Optional
     blockWhenExhausted: true                # Optional
     nettyThreads: 32                        # Optional
```

```
    subscriptionsPerConnection: 10          # Optional
    subscriptionConnectionPoolSize: 100     # Optional
    pageSizeInBytes: 8192                   # Optional
    readTimeout: 2000                       # Optional
    timeToLiveMillis: 60000                 # Optional
```

Redis Blusam 快取

```
bluesam:
   locks:
     cache: redis
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
       hostName: localhost
       port: 6379
       mode: standalone                        # Optional
       username:                               # Optional
       password: ""                            # Optional
       useSsl: false                           # Optional
       database: 0                             # Optional
       maxTotal: 128                           # Optional
       maxIdle: 128                            # Optional
       minIdle: 16                             # Optional
       testOnBorrow: true                      # Optional
       testOnReturn: true                      # Optional
       testWhileIdle: true                     # Optional
       testOnCreate: true                      # Optional
       minEvictableIdleTimeMillis: 60000       # Optional
       timeBetweenEvictionRunsMillis: 30000    # Optional
       numTestsPerEvictionRun: -1              # Optional
       blockWhenExhausted: true                # Optional
       nettyThreads: 32                        # Optional
       subscriptionsPerConnection: 10          # Optional
       subscriptionConnectionPoolSize: 100     # Optional
       pageSizeInBytes: 8192                   # Optional
       readTimeout: 2000                       # Optional
```

工作階段快取

```
 spring:
   session:
     store-type: redis
```

```
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
 jics:
   redis:
     hostName: localhost
     port: 6379
     mode: standalone                          # Optional
     username:                                 # Optional
     password: ""                              # Optional
     useSsl: false                             # Optional
     database: 0                               # Optional
     maxTotal: 128                             # Optional
     maxIdle: 128                              # Optional
     minIdle: 16                               # Optional
     testOnBorrow: true                        # Optional
     testOnReturn: true                        # Optional
     testWhileIdle: true                       # Optional
     testOnCreate: true                        # Optional
     minEvictableIdleTimeMillis: 60000         # Optional
     timeBetweenEvictionRunsMillis: 30000      # Optional
     numTestsPerEvictionRun: -1                # Optional
     blockWhenExhausted: true                  # Optional
     nettyThreads: 32                          # Optional
     subscriptionsPerConnection: 10            # Optional
     subscriptionConnectionPoolSize: 100       # Optional
     pageSizeInBytes: 8192                     # Optional
     readTimeout: 2000                         # Optional
```

JICS 資源定義

```
jics:
   resource-definitions:
     store-type: redis
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
   redis:
     hostName: localhost
     port: 6379
     mode: standalone                          # Optional
     username:                                 # Optional
     password: ""                              # Optional
     useSsl: false                             # Optional
     database: 0                               # Optional
```

```
    maxTotal: 128                          # Optional
    maxIdle: 128                           # Optional
    minIdle: 16                            # Optional
    testOnBorrow: true                     # Optional
    testOnReturn: true                     # Optional
    testWhileIdle: true                    # Optional
    testOnCreate: true                     # Optional
    minEvictableIdleTimeMillis: 60000      # Optional
    timeBetweenEvictionRunsMillis: 30000   # Optional
    numTestsPerEvictionRun: -1             # Optional
    blockWhenExhausted: true               # Optional
    nettyThreads: 32                       # Optional
    subscriptionsPerConnection: 10         # Optional
    subscriptionConnectionPoolSize: 100    # Optional
    pageSizeInBytes: 8192                  # Optional
    readTimeout: 2000                      # Optional
```

JICS TS 佇列

```
jics:
   parameters:
     tsqimpl: redis
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
   queues:
     ts:
       redis:
         hostName: localhost
         port: 6379
         mode: standalone                        # Optional
         username:                               # Optional
         password: ""                            # Optional
         useSsl: false                           # Optional
         database: 0                             # Optional
         maxTotal: 128                           # Optional
         maxIdle: 128                            # Optional
         minIdle: 16                             # Optional
         testOnBorrow: true                      # Optional
         testOnReturn: true                      # Optional
         testWhileIdle: true                     # Optional
         testOnCreate: true                      # Optional
         minEvictableIdleTimeMillis: 60000       # Optional
         timeBetweenEvictionRunsMillis: 30000    # Optional
```

```
        numTestsPerEvictionRun: -1              # Optional
        blockWhenExhausted: true                # Optional
        nettyThreads: 32                        # Optional
        subscriptionsPerConnection: 10          # Optional
        subscriptionConnectionPoolSize: 100     # Optional
        pageSizeInBytes: 8192                   # Optional
        readTimeout: 2000                       # Optional
```

工作階段追蹤器

```
session-tracker:
   store-type: redis
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
   redis:
     hostName: localhost
     port: 6379
     mode: standalone                       # Optional
     username:                              # Optional
     password: ""                           # Optional
     useSsl: false                          # Optional
     database: 0                            # Optional
     maxTotal: 128                          # Optional
     maxIdle: 128                           # Optional
     minIdle: 16                            # Optional
     testOnBorrow: true                     # Optional
     testOnReturn: true                     # Optional
     testWhileIdle: true                    # Optional
     testOnCreate: true                     # Optional
     minEvictableIdleTimeMillis: 60000      # Optional
     timeBetweenEvictionRunsMillis: 30000   # Optional
     numTestsPerEvictionRun: -1             # Optional
     blockWhenExhausted: true               # Optional
     nettyThreads: 32                       # Optional
     subscriptionsPerConnection: 10         # Optional
     subscriptionConnectionPoolSize: 100    # Optional
     pageSizeInBytes: 8192                  # Optional
     readTimeout: 2000                      # Optional
```

JCL 檢查點

```
jcl:
   checkpoint:
```

```
    provider: redis
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
   redis:
     hostname: localhost
     port: 6379
     mode: standalone                          # Optional
     username:                                 # Optional
     password: ""                              # Optional
     useSsl: false                             # Optional
     database: 0                               # Optional
     maxTotal: 128                             # Optional
     maxIdle: 128                              # Optional
     minIdle: 16                               # Optional
     testOnBorrow: true                        # Optional
     testOnReturn: true                        # Optional
     testWhileIdle: true                       # Optional
     testOnCreate: true                        # Optional
     minEvictableIdleTimeMillis: 60000         # Optional
     timeBetweenEvictionRunsMillis: 30000      # Optional
     numTestsPerEvictionRun: -1                # Optional
     blockWhenExhausted: true                  # Optional
     nettyThreads: 32                          # Optional
     subscriptionsPerConnection: 10            # Optional
     subscriptionConnectionPoolSize: 100       # Optional
     pageSizeInBytes: 8192                     # Optional
     readTimeout: 2000                         # Optional
```

Gapwalk 檔案鎖定

```
filesLocks:
   enabled: true
   retryTime: 1000
   MaxRetry: 5
   provider: redis
 # If the following redis properties are not specified gapwalk.redis configuration will
 be used for this cache
   redis:
     hostName: localhost
     port: 6379
     mode: standalone                            # Optional
     username:                                   # Optional
     password: ""                                # Optional
```

```
    useSsl: false                              # Optional
    database: 0                                # Optional
    pool:
      maxTotal: 128                            # Optional
      maxIdle: 128                             # Optional
      minIdle: 16                              # Optional
      testOnBorrow: true                       # Optional
      testOnReturn: true                       # Optional
      testWhileIdle: true                      # Optional
      testOnCreate: true                       # Optional
      minEvictableIdleTimeMillis: 60000        # Optional
      timeBetweenEvictionRunsMillis: 30000     # Optional
      numTestsPerEvictionRun: -1               # Optional
      blockWhenExhausted: true                 # Optional
      nettyThreads: 32                         # Optional
      subscriptionsPerConnection: 10           # Optional
      subscriptionConnectionPoolSize: 100      # Optional
      pageSizeInBytes: 8192                    # Optional
      readTimeout: 2000                        # Optional
```

Blu4iv 鎖定

```
blu4iv.lock: redis
blu4iv.lock.timeout: 10 #(in millisecondes)
 # If the following redis properties are not specified gapwalk.redis configuration
will be used for this cache
blu4iv.lock.redis:
      hostName: localhost
      port: 6379
      mode: standalone                         # Optional
      username:                                # Optional
      password: ""                             # Optional
      useSsl: false                            # Optional
      database: 0                              # Optional
      maxTotal: 128                            # Optional
      maxIdle: 128                             # Optional
      minIdle: 16                              # Optional
      testOnBorrow: true                       # Optional
      testOnReturn: true                       # Optional
      testWhileIdle: true                      # Optional
      testOnCreate: true                       # Optional
      minEvictableIdleTimeMillis: 60000        # Optional
```

```
      timeBetweenEvictionRunsMillis: 30000    # Optional
      numTestsPerEvictionRun: -1              # Optional
      blockWhenExhausted: true                # Optional
      nettyThreads: 32                        # Optional
      subscriptionsPerConnection: 10          # Optional
      subscriptionConnectionPoolSize: 100     # Optional
      pageSizeInBytes: 8192                   # Optional
      readTimeout: 2000                       # Optional
```

資料集目錄

```
datasimplifier:
   catalogImplementation: redis
   # If the following redis properties are not specified gapwalk.redis configuration
 will be used for this cache
   redis:
     hostName: localhost
     port: 6379
     mode: standalone                       # Optional
     username:                              # Optional
     password: ""                           # Optional
     useSsl: false                          # Optional
     database: 0                            # Optional
     maxTotal: 128                          # Optional
     maxIdle: 128                           # Optional
     minIdle: 16                            # Optional
     testOnBorrow: true                     # Optional
     testOnReturn: true                     # Optional
     testWhileIdle: true                    # Optional
     testOnCreate: true                     # Optional
     minEvictableIdleTimeMillis: 60000      # Optional
     timeBetweenEvictionRunsMillis: 30000   # Optional
     numTestsPerEvictionRun: -1             # Optional
     blockWhenExhausted: true               # Optional
     nettyThreads: 32                       # Optional
     subscriptionsPerConnection: 10         # Optional
     subscriptionConnectionPoolSize: 100    # Optional
     pageSizeInBytes: 8192                  # Optional
     readTimeout: 2000                      # Optional
```

Redis 快取的 Secret Manager

`application-main.yaml` 檔案可以參考 Redis 快取的秘密 ARN。如需如何在執行時間整合 AWS Secrets Manager 以安全地擷取 Redis 連線詳細資訊的詳細資訊，請參閱 the section called “AWS Blu Age 執行期秘密”。

設定 Gapwalk 應用程式的安全性

下列主題說明如何保護 Gapwalk 應用程式。

您有責任提供正確的組態，以確保使用 AWS Blu Age 架構是安全的。

預設會停用所有與安全相關的功能。若要啟用身分驗證（以及 CSRF、XSS、CSP 等），請將 `gapwalk-application.security` 設定為 `enabled`，並將 `gapwalk-application.security.identity`設定為 `oauth`。

主題

- 設定 Gapwalk 應用程式的 URI 可存取性
- 設定 Gapwalk 應用程式的身分驗證

設定 Gapwalk 應用程式的 URI 可存取性

本主題說明如何設定 Gapwalk 應用程式的 URIs 篩選。此功能不需要身分提供者 (IdP)。

若要封鎖 URIs 清單，請將以下兩行新增至`application-main.yml`現代化應用程式的 ，將 *`URI-1`*、*`URI-2`* 等取代為您要封鎖URIs。

```
gapwalk-application.security.filterURIs: enabled
gapwalk-application.security.blockedURIs: URI-1, URI-2, URI-3
```

設定 Gapwalk 應用程式的身分驗證

若要為您的 Gapwalk 應用程式設定 OAuth2 身分驗證，您需要設定身分提供者 (IdP)，並將其與您的應用程式整合。本指南涵蓋使用 Amazon Cognito 或 Keycloak 作為 IdP 的步驟。使用 Amazon Cognito，您可以使用 Cognito 使用者集區詳細資訊來更新應用程式的組態檔案。使用 Keycloak，您可以根據使用者指派的角色來控制對應用程式 APIs 和資源的存取。

主題

- 使用 Amazon Cognito 設定 Gapwalk OAuth2 身分驗證

- 使用 Keycloak 設定 Gapwalk OAuth2 身分驗證

使用 Amazon Cognito 設定 Gapwalk OAuth2 身分驗證

本主題說明如何使用 Amazon Cognito 做為身分提供者 (IdP)，為 Gapwalk 應用程式設定 OAuth2 身分驗證。

先決條件

在本教學課程中，我們將使用 Amazon Cognito 做為 IdP，並使用 PlanetDemo 做為現代化專案。

您可以使用任何其他外部身分提供者。ClientRegistration 資訊必須從您的 IdP 取得，且為 Gapwalk 身分驗證的必要項目。如需詳細資訊，請參閱 Amazon Cognito 開發人員指南。

ClientRegistration 資訊：

client-id

ClientRegistration 的 ID。在我們的範例中，它是 PlanetsDemo。

client-secret

您的用戶端秘密。

授權端點

授權伺服器的授權端點 URI。

權杖端點

授權伺服器的字符端點 URI。

jwks 端點

用來取得 JSON Web 金鑰 (JWK) 的 URI，其中包含驗證授權伺服器所發出 JSON Web 簽章的金鑰。

重新導向 URI

如果授予存取權，授權伺服器重新導向最終使用者的 URI。

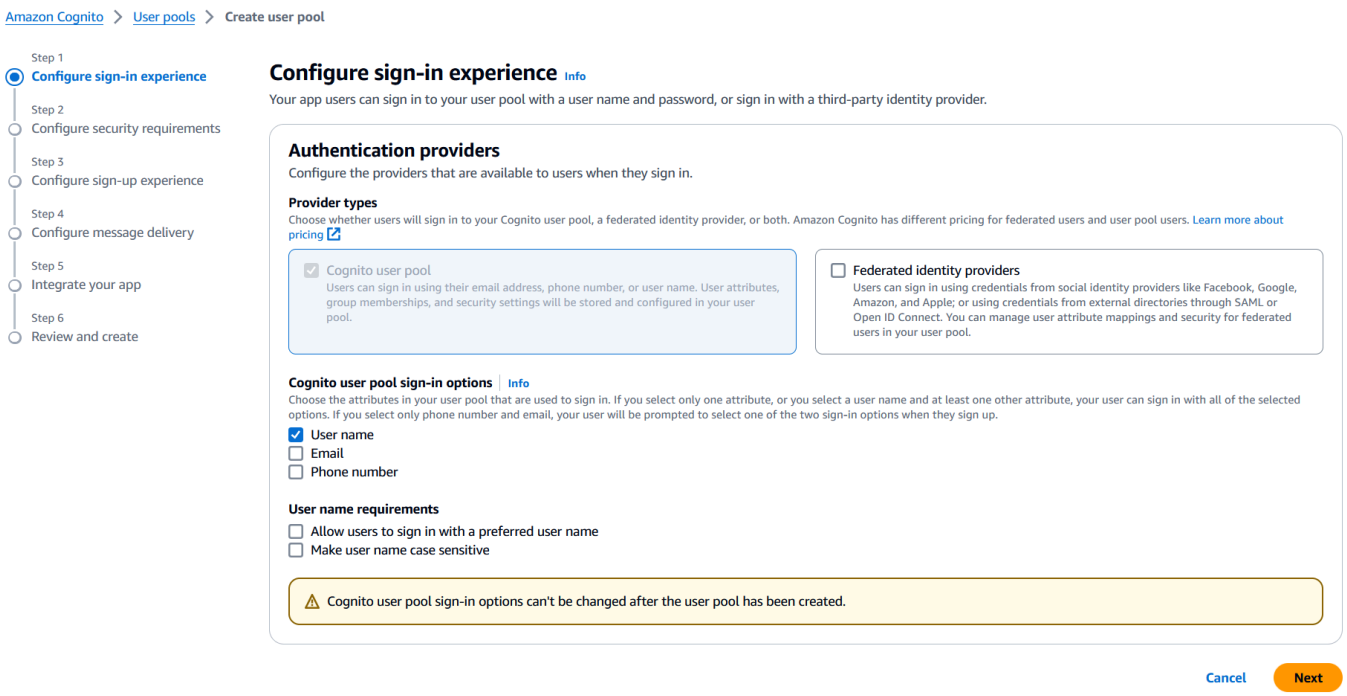
Amazon Cognito 設定

首先，我們將建立和設定 Amazon Cognito 使用者集區和使用者，我們將將其與已部署的 Gapwalk 應用程式搭配使用，用於測試目的。

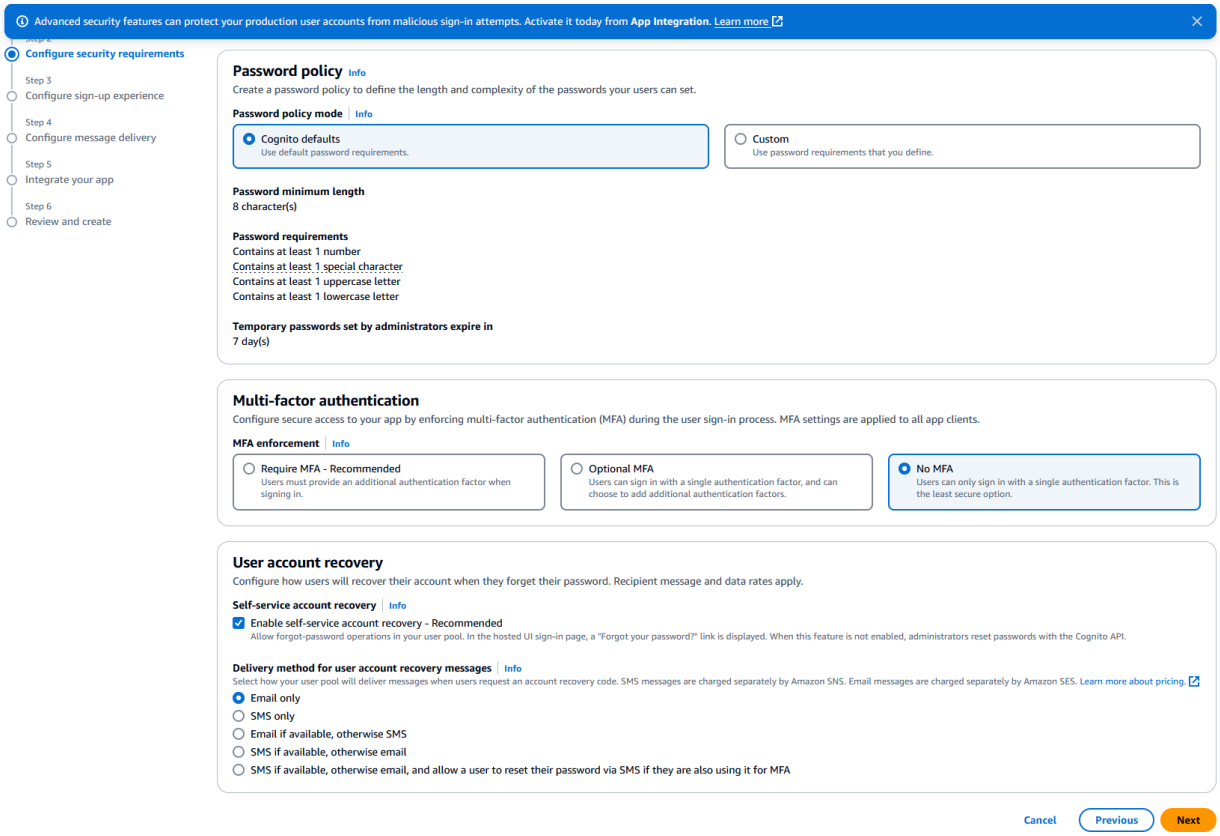
> **Note**
>
> 如果您使用的是另一個 IdP，則可以略過此步驟。

建立使用者集區

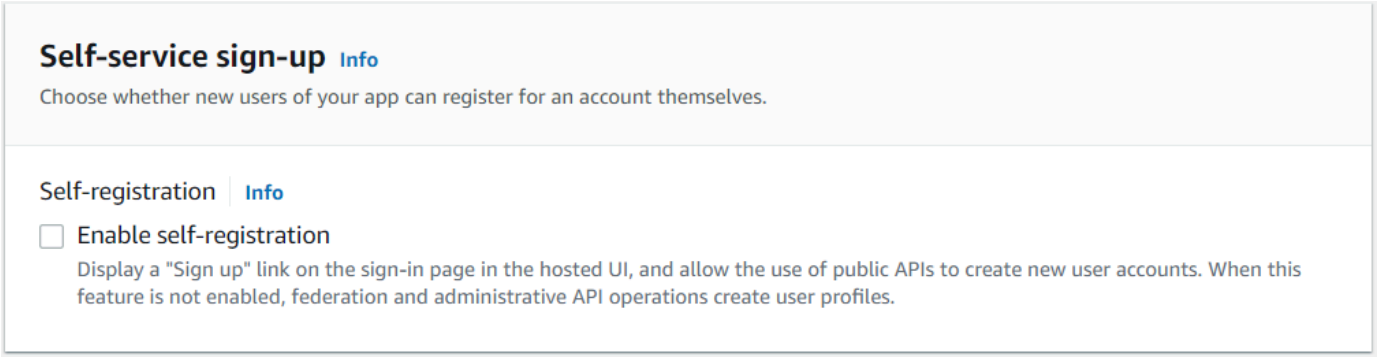
1. 前往 中的 Amazon Cognito AWS Management Console ，並使用您的 AWS 登入資料進行身分驗證。
2. 選擇 User Pools (使用者集區)。
3. 選擇 Create a user pool (建立使用者集區)。
4. 在設定登入體驗中，保留 Cognito 使用者集區預設提供者類型。您可以選擇一或多個 Cognito 使用者集區登入選項；現在，選擇使用者名稱，然後選擇下一步。



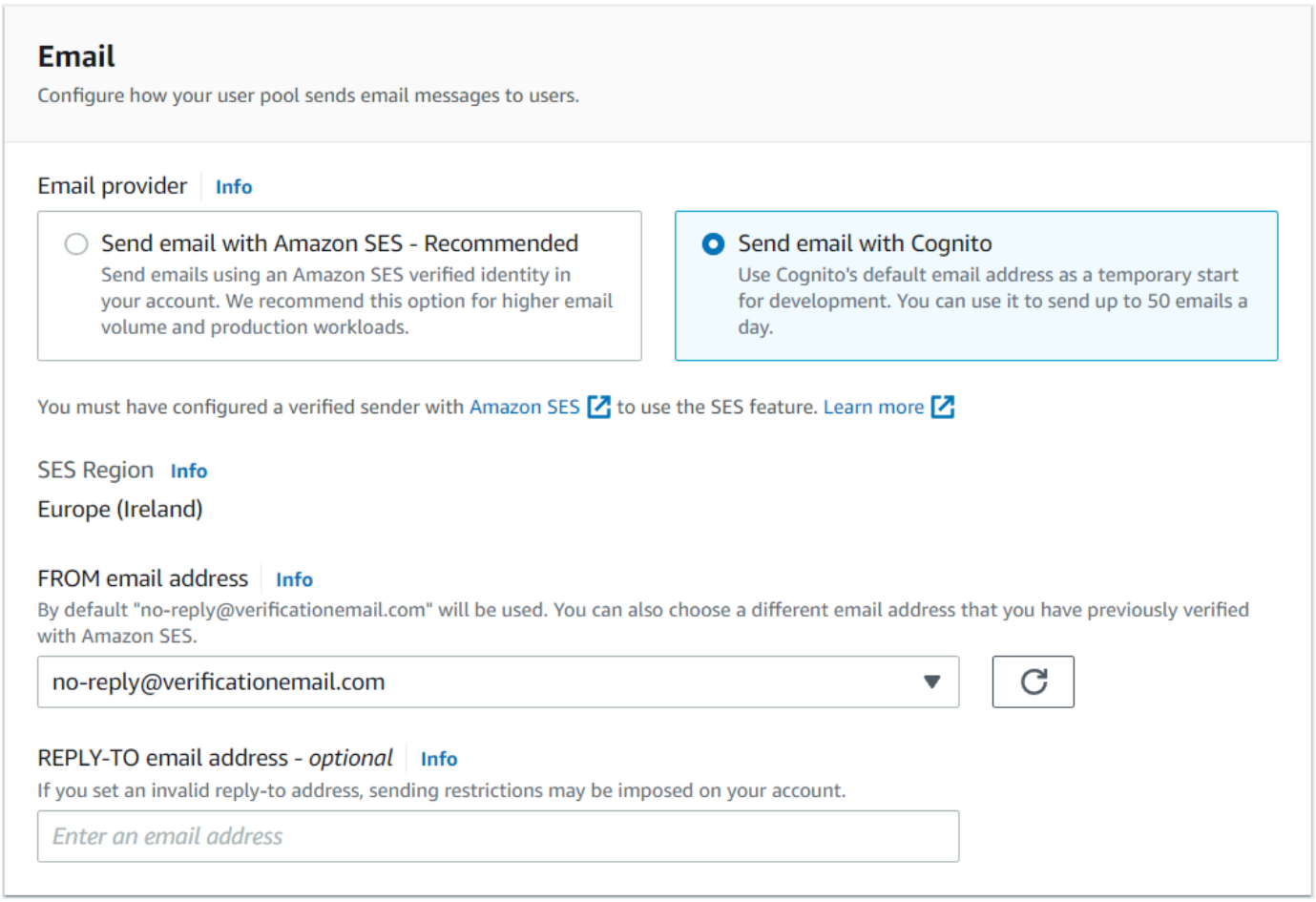
5. 在設定安全需求中，選擇無 MFA 來保持預設值並停用多重要素驗證，然後選擇下一步。



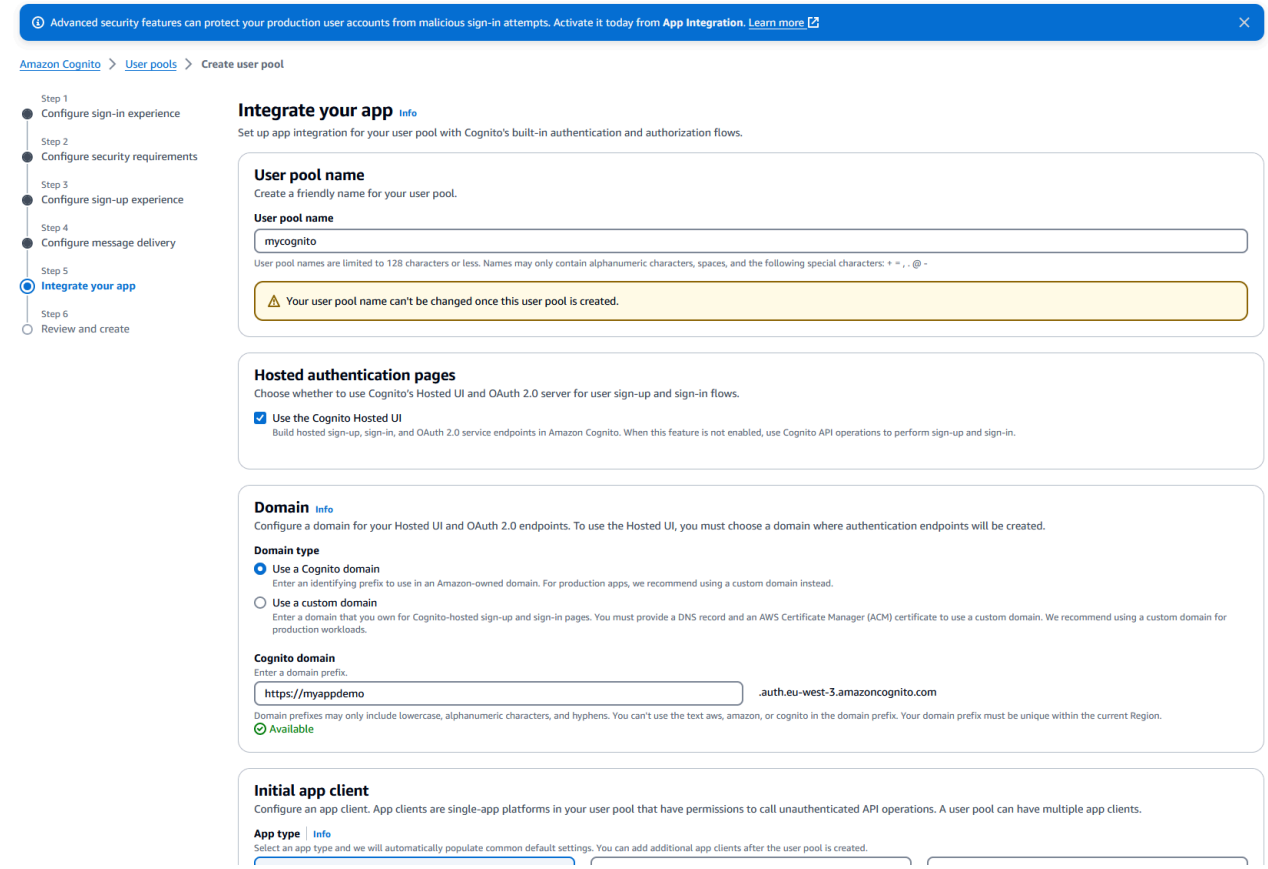
6. 做為安全措施，請停用啟用自我註冊，然後選擇下一步。



7. 選擇使用 Cognito 傳送電子郵件，然後選擇下一步。



8. 在整合您的應用程式中，為您的使用者集區指定名稱。在託管身分驗證頁面中，選擇使用 Cognito 託管 UI。



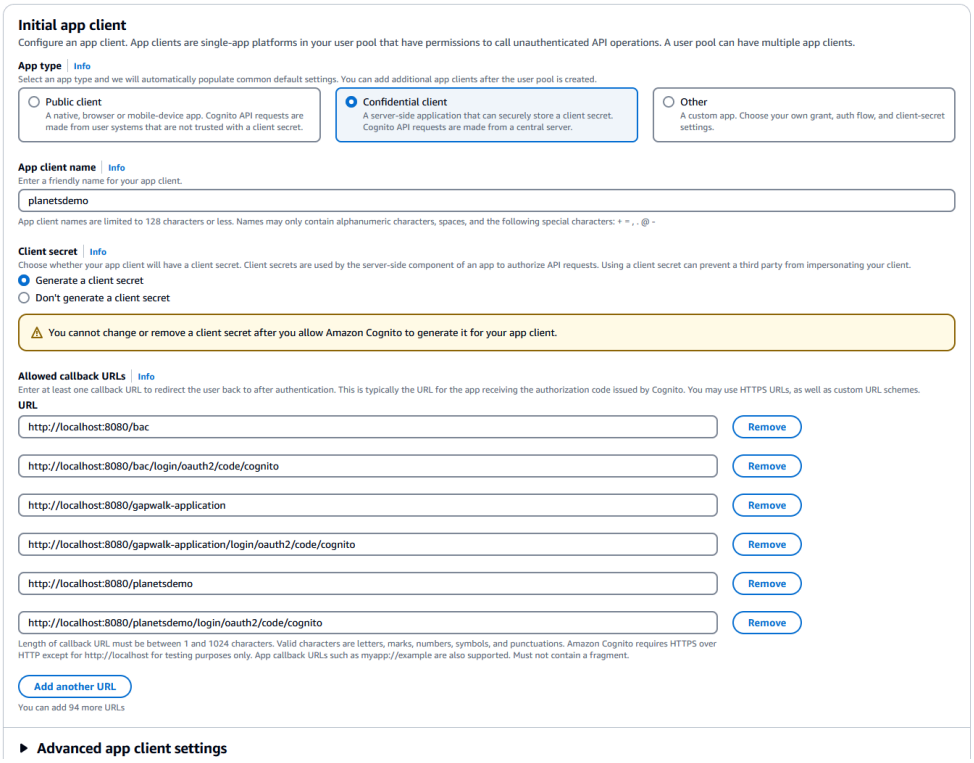
9. 為了簡化，在網域中，選擇使用 Cognito 網域並輸入網域字首；例如 `https://planetsdemo`。示範應用程式必須新增為用戶端。

   a. 在初始應用程式用戶端中，選擇機密用戶端。輸入應用程式用戶端名稱，例如 **`planetsdemo`**，然後選擇產生用戶端秘密。

   b. 在允許的回呼 URL 中，輸入 URL 以在身分驗證後將使用者重新導向至 。URL 必須以 結尾/`login/oauth2/code/cognito`。例如，對於我們的應用程式和後端 Gapwalk 和 BAC 應用程式：

```
http://localhost:8080/bac
      http://localhost:8080/bac/login/oauth2/code/cognito
      http://localhost:8080/gapwalk-application
      http://localhost:8080/gapwalk-application/login/oauth2/code/cognito
      http://localhost:8080/planetsdemo
      http://localhost:8080/planetsdemo/login/oauth2/code/cognito
```

   您可以稍後編輯 URL。



c. 在允許的登出 URLs 中，輸入您要 Amazon Cognito 在應用程式將使用者登出時重新導向至的登出頁面 URL。例如，對於後端 Gapwalk 和 BAC 應用程式：

```
http://localhost:8080/bac/logout
http://localhost:8080/gapwalk-application/logout
http://localhost:8080/planetsdemo/logout
```

您可以稍後編輯 URL。

d. 在進階應用程式用戶端設定和屬性讀取和寫入許可區段中保留預設值。

e. 選擇下一步。

10. 在檢閱和建立中，驗證您的選擇，然後選擇建立使用者集區。

如需詳細資訊，請參閱建立使用者集區。

使用者建立

由於自我註冊已停用，請建立 Amazon Cognito 使用者。在 中導覽至 Amazon Cognito AWS Management Console。選擇您建立的使用者集區，然後在使用者中選擇建立使用者。

在使用者資訊中，選擇傳送電子郵件邀請、輸入使用者名稱和電子郵件地址，然後選擇產生密碼。選擇 Create user (建立使用者)。

角色建立

在群組索引標籤中，建立 3 個群組 (SUPER_ADMIN、ADMIN 和 USER)，並將您的使用者與其中一或多個群組建立關聯。這些角色稍後會由 Gapwalk 應用程式映射至 ROLE_SUPER_ADMIN、ROLE_ADMIN 和 ROLE_USER，以便能夠存取一些受限的 API REST 呼叫。

將 Amazon Cognito 整合到 Gapwalk 應用程式

現在您的 Amazon Cognito 使用者集區和使用者已準備就緒，請移至現代化應用程式的 `application-main.yml` 檔案，並新增下列程式碼：

```
gapwalk-application.security: enabled
gapwalk-application.security.identity: oauth
gapwalk-application.security.issuerUri: https://cognito-idp.<region-id>.amazonaws.com/
<pool-id>
gapwalk-application.security.domainName: <your-cognito-domain>
gapwalk-application.security.localhostWhitelistingEnabled: false

spring:
  security:
    oauth2:
      client:
        registration:
          cognito:
            client-id: <client-id>
            client-name: <client-name>
            client-secret: <client-secret>
            provider: cognito
            authorization-grant-type: authorization_code
            scope: openid
            redirect-uri: "<redirect-uri>"
        provider:
          cognito:
            issuer-uri: ${gapwalk-application.security.issuerUri}
            authorization-uri: ${gapwalk-application.security.domainName}/oauth2/
authorize
            jwk-set-uri: ${gapwalk-application.security.issuerUri}/.well-known/
jwks.json
            token-uri: ${gapwalk-application.security.domainName}/oauth2/token
            user-name-attribute: username
      resourceserver:
        jwt:
```

```
        jwk-set-uri: ${gapwalk-application.security.issuerUri}/.well-known/jwks.json
```

依照說明取代下列預留位置：

1. 前往 中的 Amazon Cognito AWS Management Console ，並使用您的 AWS 登入資料進行身分驗證。
2. 選擇使用者集區，然後選擇您建立的使用者集區。您可以在使用者*## ID ##### pool-*id。
3. 選擇您可以在其中找到*your-cognito-domain* 的應用程式整合，然後前往應用程式用戶端和分析，然後選擇您的應用程式。
4. 在應用程式用戶端：yourApp 中，您可以找到 *client-name* 、 *client-id* 和 *client-secret* (顯示用戶端秘密)。
5. *region-id* 對應至您建立 Amazon Cognito 使用者和使用者集區的 AWS 區域 ID。範例：eu-west-3。
6. 針對 *redirect-uri* ，輸入您為允許回呼 URL 指定的 URI。在我們的範例中，它是 http://localhost:8080/planetsdemo/login/oauth2/code/cognito。

您現在可以部署您的 Gapwalk 應用程式，並使用先前建立的使用者來登入您的應用程式。

使用 Keycloak 設定 Gapwalk OAuth2 身分驗證

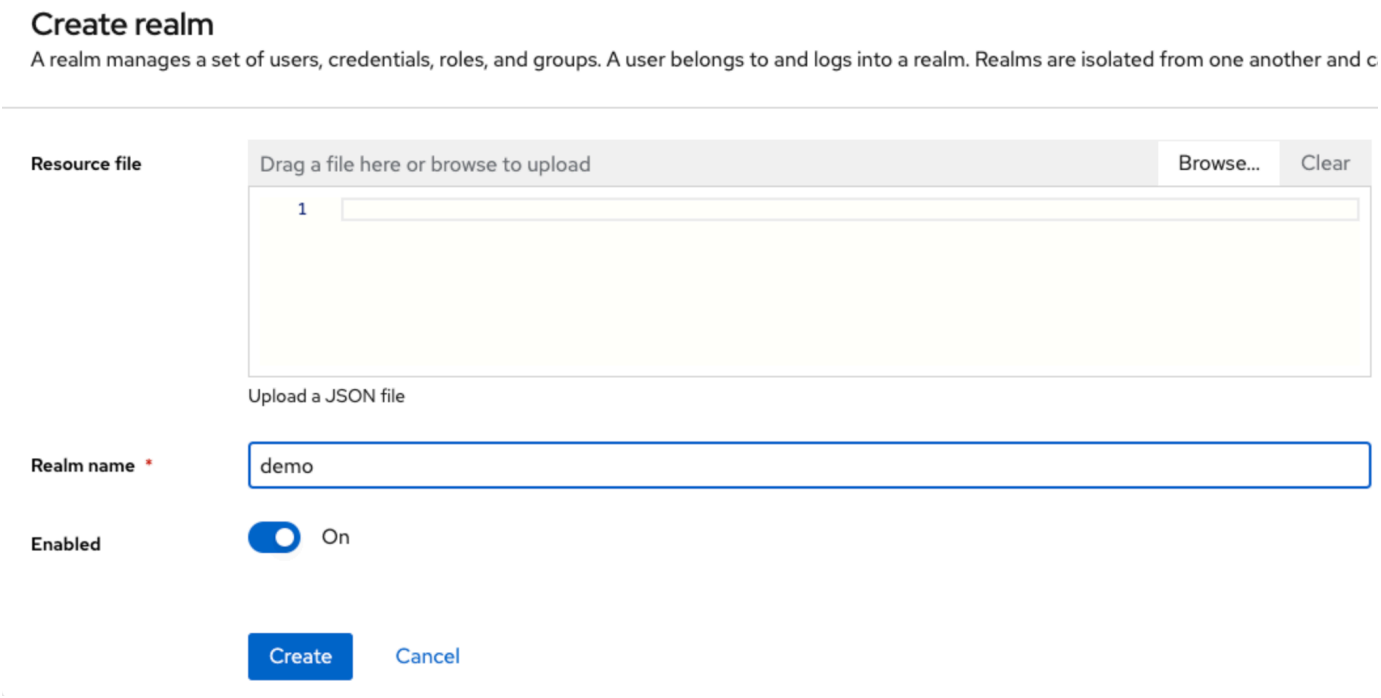
本主題說明如何使用 Keycloak 做為身分提供者 (IdP)，為 Gapwalk 應用程式設定 OAuth2 身分驗證。在本教學課程中，我們使用 Keycloak 24.0.0。

先決條件

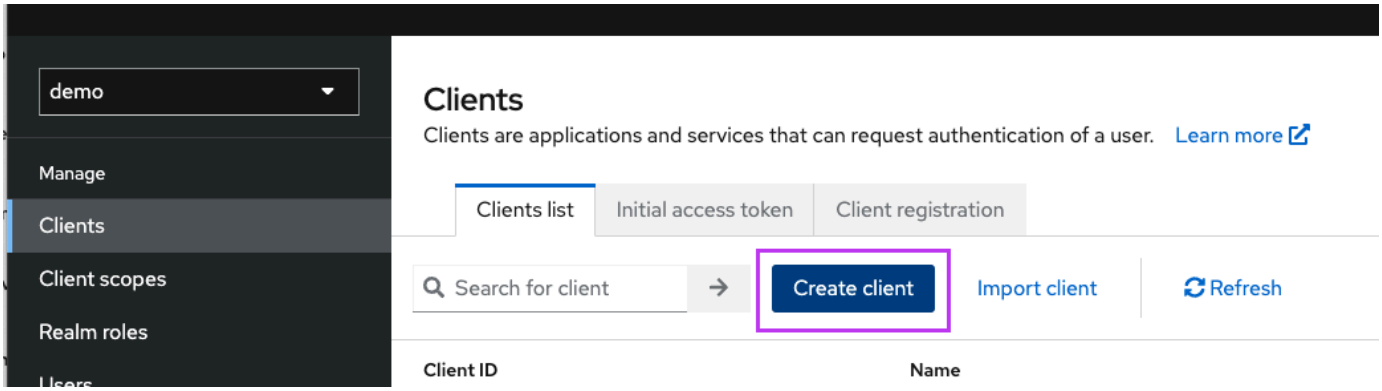
- Keycloak
- Gapwalk 應用程式

Keycloak 設定

1. 前往 Web 瀏覽器中的 Keycloak 儀表板。預設登入資料為 admin/admin。前往左上角導覽列，並建立名稱為 的領域**demo**，如下圖所示。



2. 建立名稱為 的用戶端**app-demo**。



`localhost:8080` 將 取代為 Gapwalk 應用程式的地址

General settings

Client ID * app-demo

Name

Description

Always display in UI Off

Access settings

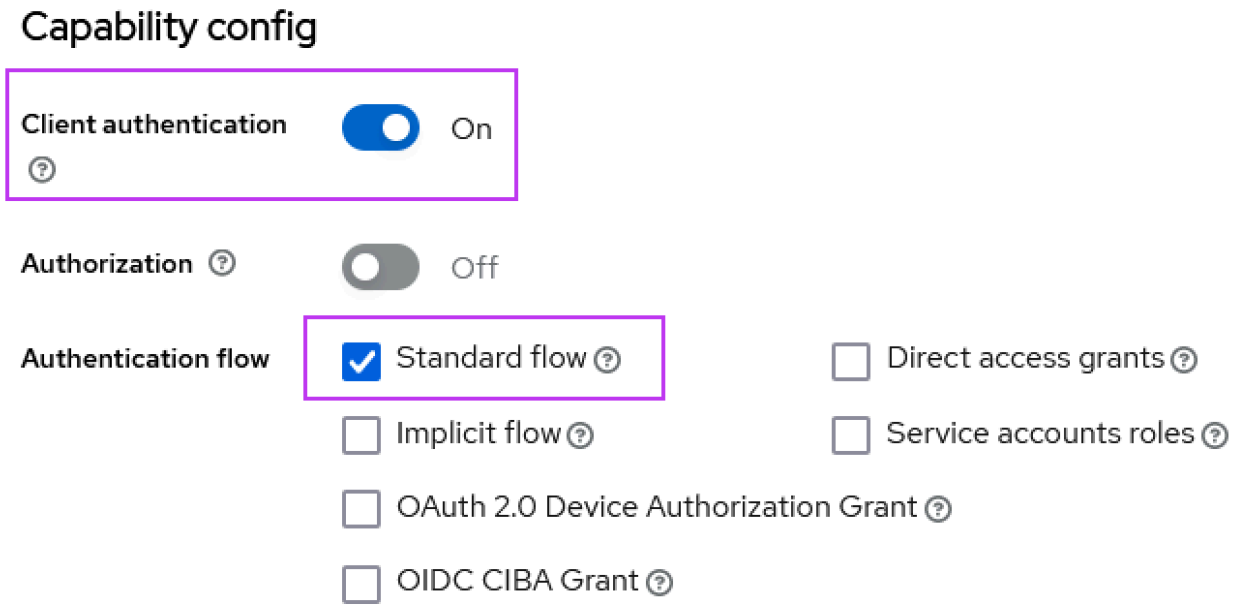
Root URL http://localhost:8080

Home URL

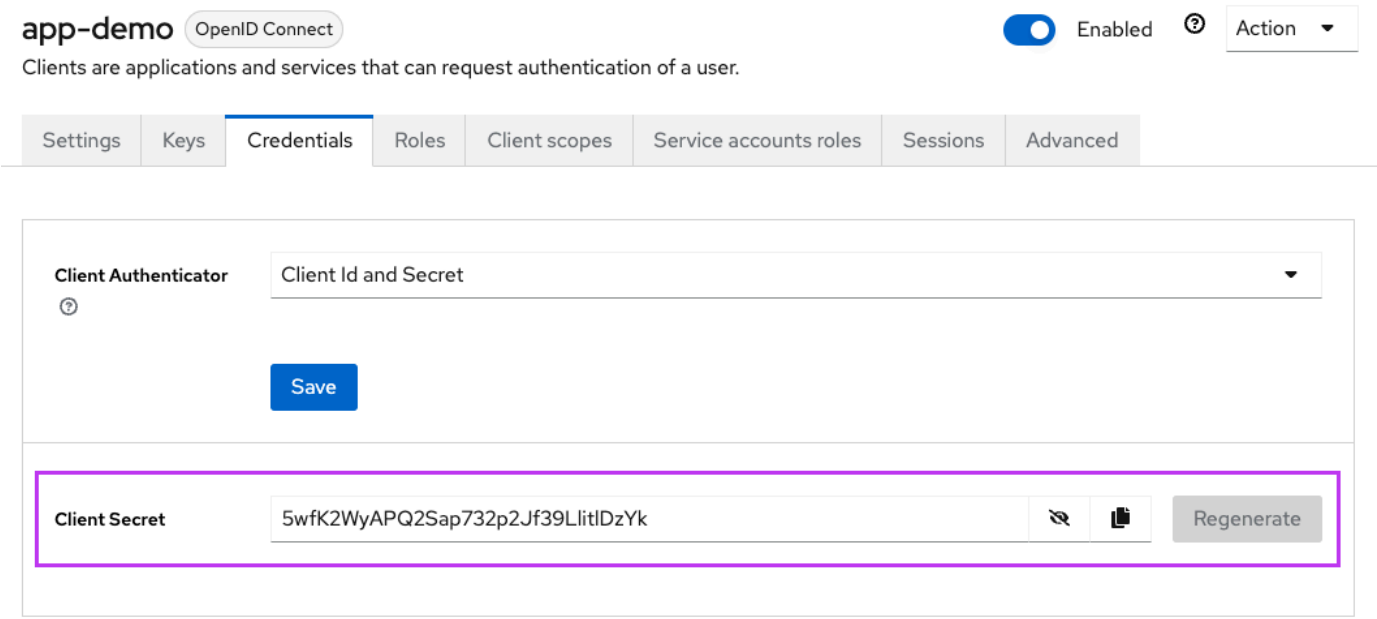
Valid redirect URIs http://localhost:8080/*
https://localhost:8080/*
Add valid redirect URIs

Valid post logout redirect URIs http://localhost:8080/*
https://localhost:8080/*
Add valid post logout redirect URIs

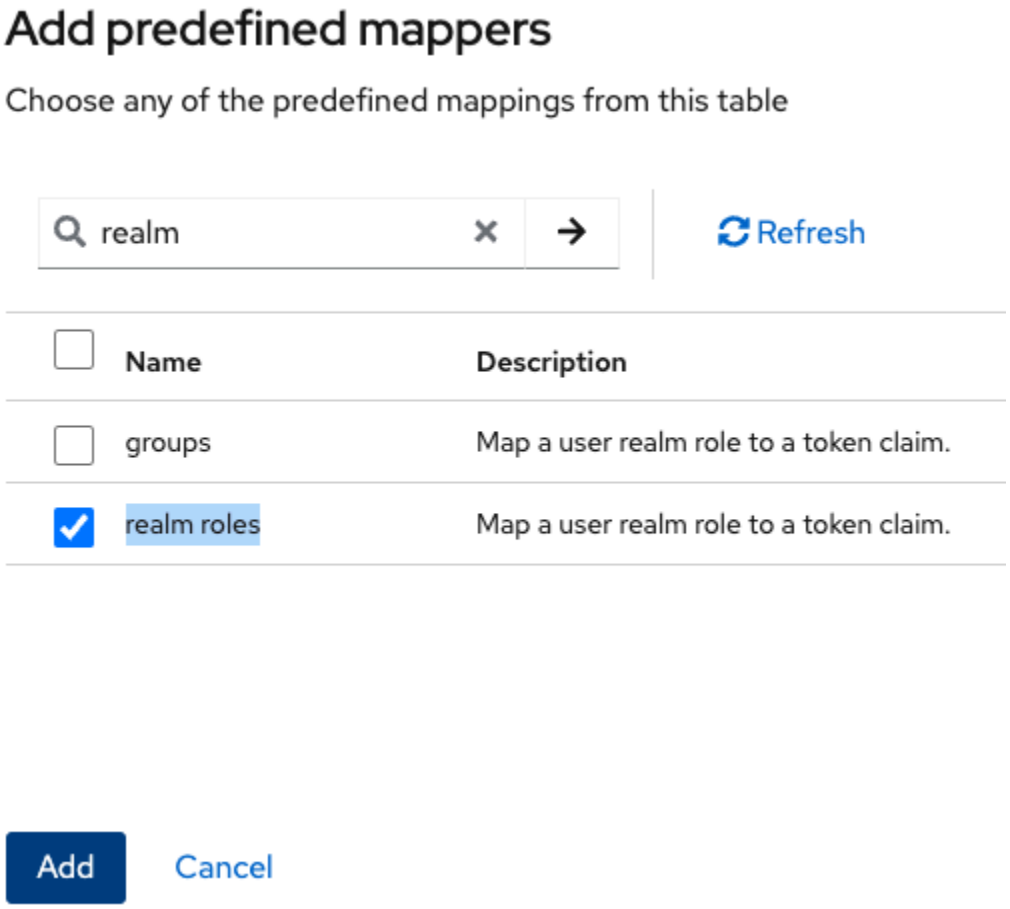
Web origins +
Add web origins



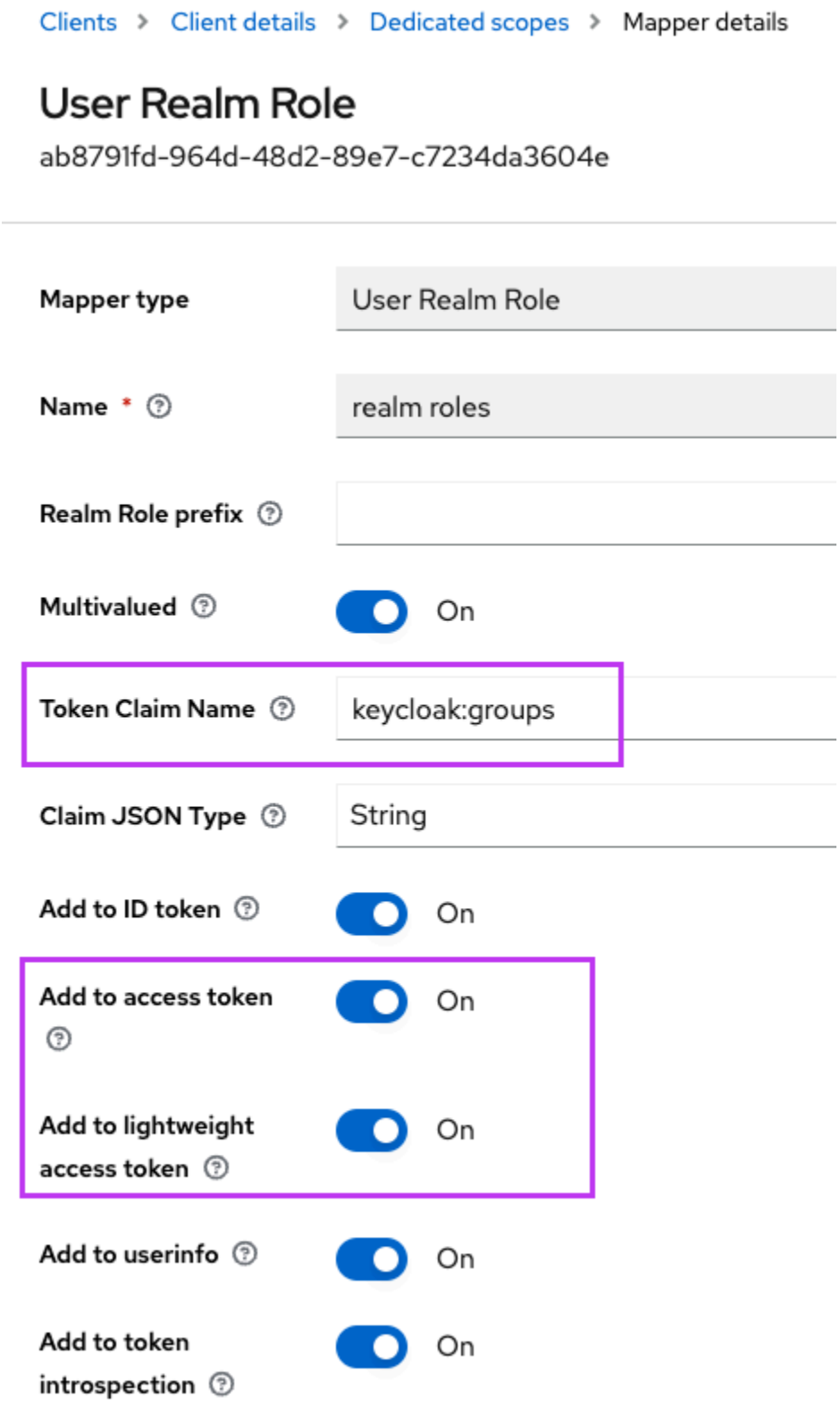
3. 若要取得您的用戶端秘密，請選擇用戶端，然後選擇應用程式示範，然後選擇登入資料。



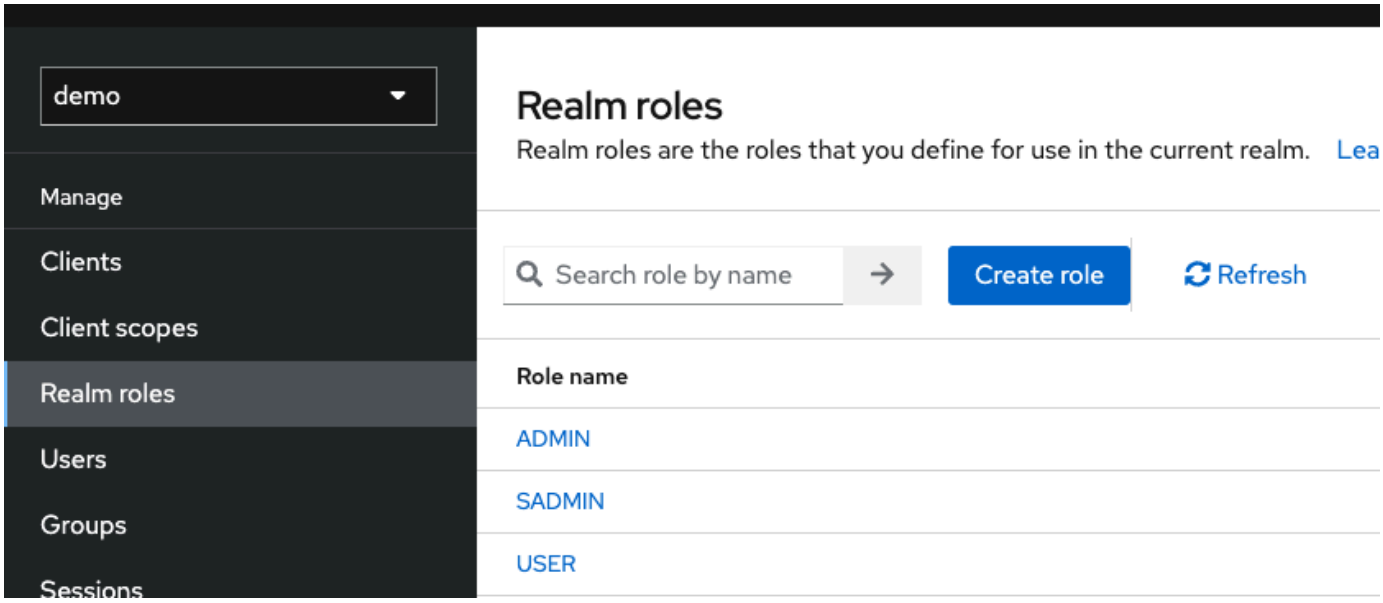
4. 選擇用戶端，然後選擇用戶端範圍，然後新增預先定義的映射器。選擇領域角色。



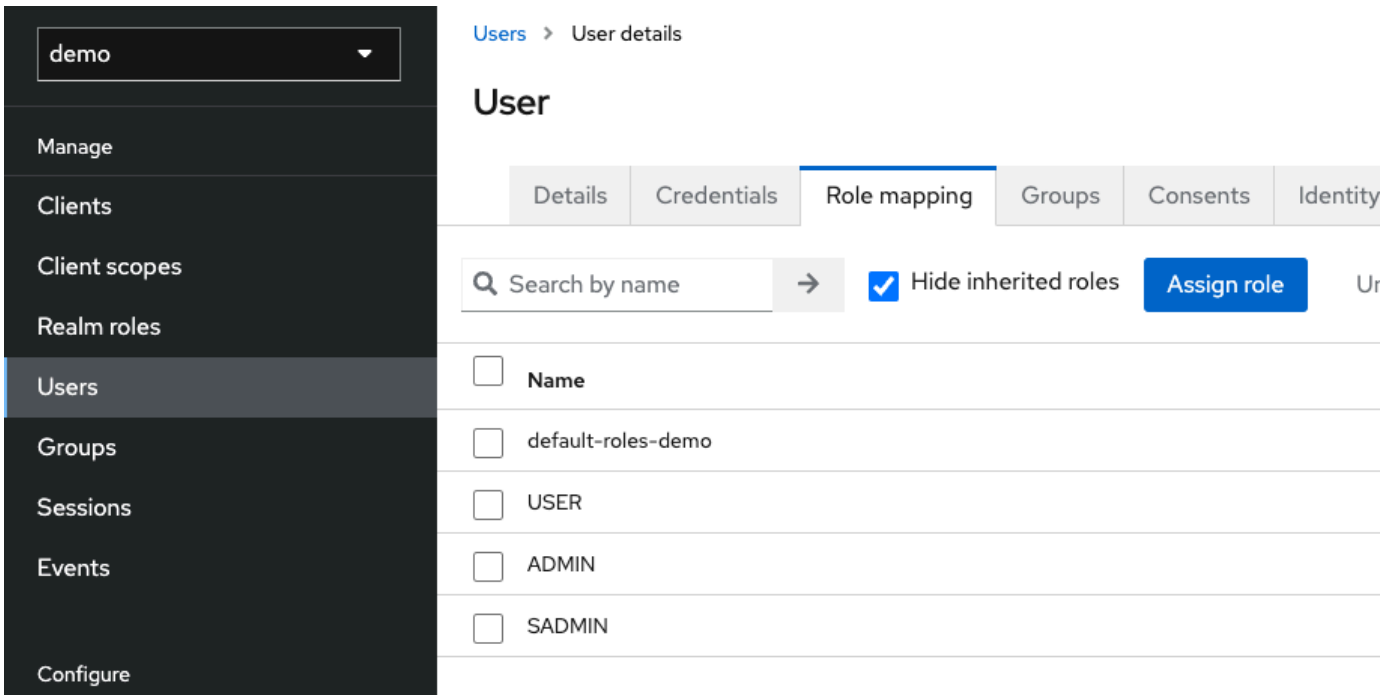
5. 使用下圖所示的組態編輯您的領域角色。



6. 記住定義的權杖宣告名稱。在 `gapwalk-application.security.claimGroupName` 屬性的 Gapwalk 設定定義中，您將需要此值。



7. 選擇領域角色，並建立 3 個角色：**ADMIN**、 **SUPER_ADMIN**和 **USER**。這些角色稍後會映射至 `ROLE_SUPER_ADMIN`、 和 `ROLE_ADMIN`，`ROLE_USER`並由 Gapwalk 應用程式存取一些受限的 API REST 呼叫。



將 Keycloak 整合到 Gapwalk 應用程式

編輯 `application-main.yml`，如下所示：

```
gapwalk-application.security: enabled
```

```
gapwalk-application.security.identity: oauth
gapwalk-application.security.issuerUri: http://<KEYCLOAK_SERVER_HOSTNAME>/realms/
<YOUR_REALM_NAME>
gapwalk-application.security.claimGroupName: "keycloak:groups"

gapwalk-application.security.userAttributeName: "preferred_username"
# Use "username" for cognito,
#     "preferred_username" for keycloak
#      or any other string
gapwalk-application.security.localhostWhitelistingEnabled: false

spring:
  security:
    oauth2:
      client:
        registration:
          demo:
            client-id: <YOUR_CLIENT_ID>
            client-name: Demo App
            client-secret: <YOUR_CLIENT_SECRET>
            provider: keycloak
            authorization-grant-type: authorization_code
            scope: openid
            redirect-uri: "{baseUrl}/login/oauth2/code/{registrationId}"
        provider:
          keycloak:
            issuer-uri: ${gapwalk-application.security.issuerUri}
            authorization-uri: ${gapwalk-application.security.issuerUri}/protocol/
openid-connect/auth
            jwk-set-uri: ${gapwalk-application.security.issuerUri}/protocol/openid-
connect/certs
            token-uri: ${gapwalk-application.security.issuerUri}/protocol/openid-
connect/token
            user-name-attribute: ${gapwalk-application.security.userAttributeName}
      resourceserver:
        jwt:
          jwk-set-uri: ${gapwalk-application.security.issuerUri}/protocol/openid-
connect/certs
```

將 *<KEYCLOAK_SERVER_HOSTNAME>*、*<YOUR_REALM_NAME>*、*<YOUR_CLIENT_ID>* 和 *<YOUR_CLIENT_SECRET>* 取代為您的 Keycloak 伺服器主機名稱、您的領域名稱、您的用戶端 ID 和您的用戶端秘密。

AWS Blu Age 執行時間 APIs

AWS Blu Age Runtime 使用多個 Web 應用程式公開 REST 端點，提供使用 REST 用戶端與現代化應用程式互動的方式（例如使用排程器呼叫任務）。

本文件的目的是列出可用的 REST 端點，並提供有關下列項目的詳細資訊：

- 其角色
- 正確使用它們的方式

端點清單會依提供服務的性質和公開端點的 Web 應用程式，依類別進行組織。

我們假設您已擁有使用 POSTMAN、Thunder Client、CURL、網頁瀏覽器等專用工具來使用 REST 端點的基本知識...) 或編寫自己的程式碼來進行 API 呼叫。

主題

- 建置 URLs時使用者可用的端點
- AWS 藍本時代的 Gapwalk 應用程式端點
- Blusam 應用程式主控台 REST 端點
- 在 AWS 藍年管理 JICS 應用程式主控台
- AWS Blu Age 使用者的資料結構

建置 URLs時使用者可用的端點

本主題列出端點根路徑的 URLs。以下每個 Web 應用程式都會定義根路徑，由所有端點共用。然後，每個端點都會新增自己的專用路徑。產生的 URL 是路徑串連的結果。例如，考量 Gapwalk 應用程式的第一個端點，我們有：

- `/gapwalk-application` 根 Web 應用程式路徑。
- `/scripts` 專用端點路徑。

產生的 URL 將是 `http://server:port/gapwalk-application/scripts`

伺服器

會指向伺服器名稱（託管指定 Web 應用程式者）。

port

伺服器公開的連接埠。

AWS 藍本時代的 Gapwalk 應用程式端點

在本主題中，了解 Gapwalk Web 應用程式的端點。它們使用根路徑 `/gapwalk-application`。

主題

- 批次任務（現代化 JCLs 等）相關的端點
- 指標端點
- 其他端點
- 任務佇列相關的端點

批次任務（現代化 JCLs 等）相關的端點

批次任務可以同步或非同步執行（請參閱下列詳細資訊）。正在執行批次任務的 groovy 指令碼，這是舊版指令碼 (JCL) 現代化的結果。

主題

- 列出部署的指令碼
- 同步啟動指令碼
- 以非同步方式啟動指令碼
- 列出觸發的指令碼
- 擷取任務執行詳細資訊
- 列出可刪除的非同步啟動指令碼
- 列出可刪除的同步啟動指令碼
- 刪除指定的任務執行
- 列出現有檢查點的可重新啟動性
- 重新啟動任務（同步）
- 重新啟動任務（非同步）
- 設定非同步任務執行的執行緒限制

列出部署的指令碼

- 支援的方法：GET
- 路徑： `/scripts`
- 引數：無
- 此端點會以字串的形式傳回伺服器上已部署的 groovy 指令碼清單。此端點主要用於 Web 瀏覽器，因為產生的字串是具有作用中連結的 HTML 頁面（每個可啟動指令碼的連結 -- 請參閱以下範例）。

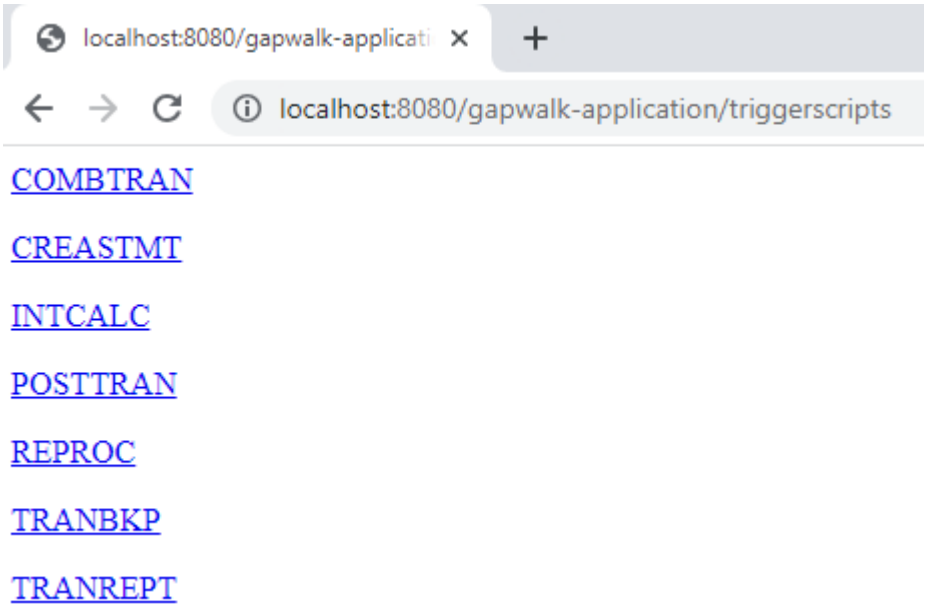
回應範例：

```
<p><a href=./script/COMBTRAN>COMBTRAN</a></p><p><a href=./script/CREASTMT>CREASTMT</
a></p><p><a href=./script/INTCALC>INTCALC</a></p><p><a href=./script/
POSTTRAN>POSTTRAN</a></p><p><a href=./script/REPROC>REPROC</a></p><p><a href=./script/
TRANBKP>TRANBKP</a></p><p><a href=./script/TRANREPT>TRANREPT</a></p><p><a href=./
script/functions>functions</a></p>
```

> **Note**
>
> 這些連結代表用來同步啟動每個所列指令碼的 URL。

- 支援的方法：GET
- 路徑： `/triggerscripts`
- 引數：無
- 此端點會以字串的形式傳回伺服器上已部署的 groovy 指令碼清單。此端點主要用於 Web 瀏覽器，因為產生的字串是具有作用中連結的 HTML 頁面（每個可啟動指令碼的連結 -- 請參閱以下範例）。

  與先前的端點回應相反，連結代表用於非同步啟動每個所列指令碼的 URL。



同步啟動指令碼

此端點有兩個變體，具有 GET 和 POST 用量的專用路徑（請參閱下方）。

- 支援的方法：GET
- 路徑：`/script/{scriptId:.+}`
- 支援的方法：POST
- 路徑：`/post/script/{scriptId:.+}`
- 引數：
  - 要啟動之指令碼的識別符
  - 選擇性：使用請求參數（視為）傳遞至指令碼的參數`Map<String,String>`。指定的參數會自動新增至調用的 groovy 指令碼的繫結。
- 呼叫將啟動具有指定識別符的指令碼，如果提供額外參數，則使用額外的參數，並等待指令碼執行完成，然後傳回訊息 (String)，該訊息將是：
  - 「完成。」（如果任務執行順利執行）。
  - JSON 錯誤訊息，其中包含任務執行期間發生錯誤的詳細資訊。您可以從伺服器日誌擷取更多詳細資訊，以了解任務執行的錯誤。

```
{
    "exitCode": -1,
    "stepName": "STEP15",
```

```
    "program": "CBACT04C",
    "status": "Error"
}
```

查看伺服器日誌，我們可以發現這是部署問題（預期的程式尚未正確部署，因此找不到，導致任務執行失敗）：

```
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.s.BatchWebController          - --> executing script INTCALC
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.s.BatchWebController          - Bound jobContext 419695287 - GDGEventsQueueHandler :907380469
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.s.ScriptControlTower          - Added jobExecutor [a65c2791-864f-43c9-972a-b5f2353389e6] to Sync Script Control Tower.
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.j.s.JobExecutor              - a65c2791-864f-43c9-972a-b5f2353389e6 - worker :Thread-26 [1547512424]
2023-06-09 10:27:28 default        INFO  - c.n.b.g.r.j.s.JobExecutor              - Triggered script: INTCALC - [a65c2791-864f-43c9-972a-b5f2353389e6] - jobContext [419695287]
2023-06-09_10-27-29-613 | [JOB] INTCALC - Started
2023-06-09_10-27-29-651 | [STEP] STEP15 - Started
2023-06-09 10:27:29 default        ERROR - c.n.b.g.r.c.i.ExecutionControllerImpl   - Could not find program "CBACT04C" in the program registry.
2023-06-09 10:27:29 default        ERROR - c.n.b.g.r.c.i.ExecutionControllerImpl   - Could not find program "CBACT04C" in the program registry.
2023-06-09_10-27-29-760 | Program not found => not executed !
2023-06-09_10-27-29-761 | [STEP] STEP15 - Ended
2023-06-09_10-27-29-772 | [JOB] INTCALC - Ended
2023-06-09 10:27:29 default        INFO  - c.n.b.g.r.j.s.DefaultJobContext         - Job [419695287] - starting final operation
2023-06-09 10:27:29 default        INFO  - c.n.b.g.r.j.s.DefaultJobContext         - End of job [419695287]
2023-06-09 10:27:29 default        INFO  - c.n.b.g.r.s.ScriptControlTower          - Removed jobExecutor [a65c2791-864f-43c9-972a-b5f2353389e6] from Script Control Tower.
2023-06-09 10:27:29 default        INFO  - c.n.b.g.r.s.ScriptControlTower          - Remaining jobExecutors:0
```

> **Note**
>
> 同步呼叫應保留為短時間執行任務。長時間執行的任務應該以非同步方式啟動（請參閱下列專用端點）。

以非同步方式啟動指令碼

- 支援的方法：GET / POST
- 路徑：`/triggerscript/{scriptId:.+}`
- 引數：
  - 要啟動之指令碼的識別符
  - 選擇性：使用請求參數（視為）傳遞至指令碼的參數`Map<String,String>`。指定的參數會自動新增至調用的 groovy 指令碼的 https://docs.groovy-lang.org/latest/html/api/groovy/lang/Binding.html繫結關係】。
- 與上述同步模式相反，端點不會等待任務執行完成以傳送回應。如果找到可用的執行緒，且回應會立即傳送給發起人，且任務執行 ID 是代表任務執行的唯一識別符，可用於查詢任務執行狀態或強制刪除應該故障的任務執行。回應的格式為：

```
Triggered script <script identifier> [unique job execution id] @ <date and time>
```

- 由於任務非同步執行依賴固定的有限執行緒數量，如果找不到可用的執行緒，則任務執行可能不會啟動。在這種情況下，傳回的訊息會更像是：

```
Script [<script identifier>] NOT triggered - Thread limit reached (<actual thread
 limit>) - Please retry later or increase thread limit.
```

請參閱以下settriggerthreadlimit端點，了解如何提高執行緒限制。

回應範例：

```
Triggered script INTCALC [d43cbf46-4255-4ce2-aac2-79137573a8b4] @ 06-12-2023 16:26:15
```

唯一的任務執行識別符允許 在需要時快速擷取伺服器日誌中的相關日誌項目。其他幾個端點也會使用它，詳述如下。

列出觸發的指令碼

- 支援的方法：GET
- 路徑：/triggeredscripts/{status:.+}、 /triggeredscripts/{status:.+}/{namefilter}
- 引數：
  - 狀態 （必要）：要擷取的已觸發指令碼狀態。可能的值為：
    - all ：顯示所有任務執行詳細資訊，無論任務是否仍在執行中。
    - running：僅顯示目前正在執行之任務的任務詳細資訊。
    - done：僅顯示執行已結束之任務的任務詳細資訊。
    - killed：僅顯示使用專用端點強制終止執行之任務的任務詳細資訊 （請參閱下文）。
    - triggered：僅顯示已觸發但尚未啟動之任務的任務詳細資訊。
    - failed：僅顯示執行標示為失敗之任務的任務詳細資訊。
    - _namefilter （選用）_ ：僅擷取指定指令碼識別符的執行。
- 以 JSON 傳回任務執行詳細資訊的集合。如需詳細資訊，請參閱任務執行詳細資訊訊息結構。

回應範例：

```
[
    {
      "scriptId": "INTCALC",
      "caller": "127.0.0.1",
```

```
      "identifier": "d43cbf46-4255-4ce2-aac2-79137573a8b4",
      "startTime": "06-12-2023 16:26:15",
      "endTime": "06-12-2023 16:26:15",
      "status": "DONE",
      "executionResult": "{ \"exitCode\": -1, \"stepName\": \"STEP15\", \"program\":
\"CBACT04C\", \"status\": \"Error\" }",
      "executionMode": "ASYNCHRONOUS"
    }
 ]
```

擷取任務執行詳細資訊

- 支援的方法：GET
- 路徑： `/getjobexecutioninfo/{jobexecutionid:.+}`
- 引數：
  - jobexecutionid （必要）：擷取對應任務執行詳細資訊的唯一任務執行識別符。
- 傳回代表單一任務執行詳細資訊的 JSON 字串（請參閱任務執行詳細資訊訊息結構），如果找不到指定識別符的任務執行詳細資訊，則傳回空白回應。

列出可刪除的非同步啟動指令碼

- 支援的方法：GET
- 路徑： `/killablescripts`
- 傳回任務執行識別符的集合，這些任務已非同步啟動，目前仍在執行中且可以強制終止（請參閱下列`/kill`端點）。

列出可刪除的同步啟動指令碼

- 支援的方法：GET
- 路徑： `/killablesyncscripts`
- 傳回已同步啟動之任務的任務執行識別符集合，目前仍在執行中，並可以強制終止（請參閱下列`/kill`端點）。

刪除指定的任務執行

- 支援的方法：GET

- 路徑： `/kill/{identifier:.+}`
- 引數：任務執行識別符 （必要）：指向要強制終止之任務執行的唯一任務執行識別符。
- 傳回詳細說明任務執行終止嘗試結果的文字訊息；訊息將包含指令碼識別符、任務執行唯一識別符，以及執行終止發生的日期和時間。如果找不到指定識別符的執行中任務執行，則會改為傳回錯誤訊息。

> ⚠ Warning
>
> - 執行時間會盡最大努力順利地終止目標任務執行。因此，來自 /kill 端點的回應可能需要一些時間才能到達發起人，因為 AWS Blu Age 執行時間會嘗試將終止任務的業務影響降至最低。
> - 不得輕而易舉地執行強制終止任務，因為它可能會帶來直接的業務後果，包括可能的資料遺失或損毀。應保留給特定任務執行已橫向完成且資料修復方法已清楚識別的情況。
> - 刪除任務應會導致進一步調查 （死後分析），以找出錯誤並採取適當的補救措施。
> - 無論如何，嘗試刪除執行中的任務都會記錄在伺服器日誌中，並顯示警告層級訊息。

列出現有檢查點的可重新啟動性

任務可重新啟動性依賴於指令碼在 中註冊檢查點的能力`CheckpointRegistry`，以追蹤任務執行進度。如果任務執行無法正常結束，且已註冊重新啟動檢查點，則可以直接從最後一個已知的已註冊檢查點重新啟動任務執行 （而不必在檢查點上方執行前一步驟）。

- 支援的方法：GET
- 路徑： `/restarts/{scriptId}/{jobId}`
- 引數：
  - scriptId （選用 - 字串）：正在重新啟動的指令碼。
  - jobId （選用 - 字串）：任務執行的唯一識別符。
- 傳回現有重新啟動點的 JSON 格式清單，可用來重新啟動執行未正常到達和結束的任務，或略過先前執行的步驟來觸發延遲重新啟動。如果有任何指令碼沒有註冊檢查點，頁面內容將是「沒有註冊檢查點」。

重新啟動任務 （同步）

- 支援的方法：GET

- 路徑： `/restart/{hashcode}/{scriptId}/{skipflag}`
- 引數：
  - 雜湊碼（整數 - 必要）：使用提供的雜湊碼作為檢查點值，重新啟動任務的最新執行（請參閱上述`/restarts`端點，了解如何擷取有效的檢查點值）。
  - scriptId（選用 - 字串）：正在重新啟動的指令碼。
  - skipflag（選用 - 布林值）：略過執行選取的（檢查點）步驟，並從立即的後續步驟（如果有的話）發出重新啟動。
- 傳回：請參閱上述`/script`的傳回描述。

重新啟動任務（非同步）

- 支援的方法：GET
- 路徑： `/triggerrestart/{hashcode}/{scriptId}/{skipflag}`
- 引數：
  - 雜湊碼（整數 - 必要）：使用提供的雜湊碼作為檢查點值，重新啟動任務的最新執行（請參閱上述`/restarts`端點，了解如何擷取有效的檢查點值）。
  - scriptId（選用 - 字串）：正在重新啟動的指令碼。
  - skipflag（選用 - 布林值）：略過執行選取的（檢查點）步驟，並從立即的後續步驟（如果有的話）發出重新啟動。
- 傳回：請參閱上述`/triggerscript`的傳回描述。

設定非同步任務執行的執行緒限制

任務非同步執行依賴 JVM 中的專用執行緒集區。該集區具有與可用執行緒數量相關的固定限制。使用的 能夠根據主機功能 (CPUs 數量、可用記憶體等）調整限制。根據預設，執行緒限制設定為 5 個執行緒。

- 支援的方法：GET
- 路徑： `/settriggerthreadlimit/{threadlimit:.+}`
- 引數（整數）：要套用的新執行緒限制。必須是嚴格正整數。
- 傳回訊息 (String)，提供新的執行緒限制和先前的執行緒限制，或如果提供的執行緒限制值無效（非嚴格正整數），則傳回錯誤訊息。

回應範例：

```
Set thread limit for Script Tower Control to 10 (previous value was 5)
```

計數目前執行的已觸發任務執行

- 支援的方法：GET
- 路徑： `/countrunningtriggeredscripts`
- 傳回訊息，指出非同步啟動的任務數量和執行緒限制（也就是可同時執行的已觸發任務數量上限）。

回應範例：

```
0 triggered script(s) running (limit =10)
```

> **Note**
>
> 這可用於在啟動任務之前，檢查是否尚未達到執行緒限制（這會阻止任務啟動）。

清除任務執行資訊

只要伺服器正常運作，任務執行資訊就會保留在伺服器記憶體中。清除記憶體中最舊的資訊可能很方便，因為它們不再相關；這是此端點的目的。

- 支援的方法：GET
- 路徑： `/purgejobinformation/{age:.+}`
- 引數：嚴格正整數值，代表要清除資訊時數的存留期。
- 傳回包含下列資訊的訊息：
  - 清除檔案的名稱，其中儲存了清除的任務執行資訊以供封存。
  - 清除的任務執行資訊數目。
  - 備註中剩餘的任務執行資訊數目

指標端點

JVM

此端點會傳回與 JVM 相關的可用指標。

- 支援的方法：GET
- 路徑： `/metrics/jvm`
- 引數：無
- 傳回包含下列資訊的訊息：
  - threadActiveCount：作用中執行緒的數量。
  - jvmMemoryUsed：Java 虛擬機器主動使用的記憶體。
  - jvmMemoryMax：Java 虛擬機器允許的記憶體上限。
  - jvmMemoryFree：Java 虛擬機器目前未使用中的可用記憶體。

Session (工作階段)

此端點會傳回與目前開啟的 HTTP 工作階段相關的指標。

- 支援的方法：GET
- 路徑： `/metrics/session`
- 引數：無
- 傳回包含下列資訊的訊息：
  - sessionCount：伺服器目前維護的作用中使用者工作階段數目。

批次

- 支援的方法：GET
- 路徑： `/metrics/batch`
- 引數：
  - startTimestamp （選用，數字）：資料篩選的開始時間戳記。
  - endTimestamp （選用，數字）：資料篩選的結束時間戳記。
  - page （選用， number)：分頁的頁碼。
  - pageSize （選用， number)：分頁中每頁的項目數。
- 傳回包含下列資訊的訊息：
  - 內容：批次執行指標的清單。
  - pageNumber：分頁中的目前頁碼。
  - pageSize：每頁顯示的項目數量。

- totalPages：可用的頁面總數。
- numberOfElements：目前頁面上的項目計數。
- last：最後一頁的布林值旗標。
- first：第一頁的布林值旗標。

交易

- 支援的方法：GET
- 路徑： `/metrics/transaction`
- 引數：
  - startTimestamp（選用，數字）：資料篩選的開始時間戳記。
  - endTimestamp（選用，數字）：資料篩選的結束時間戳記。
  - page（選用， number)：分頁的頁碼。
  - pageSize（選用， number)：分頁中每頁的項目數。
- 傳回包含下列資訊的訊息：
  - 內容：交易執行指標的清單。
  - pageNumber：分頁中的目前頁碼。
  - pageSize：每頁顯示的項目數量。
  - totalPages：可用的頁面總數。
  - numberOfElements：目前頁面上的項目計數。
  - last：最後一頁的布林值旗標。
  - first：第一頁的布林值旗標。

其他端點

使用這些端點列出已註冊的程式或服務、探索運作狀態，以及管理 JICS 交易。

主題

- 列出已註冊的程式
- 列出已註冊的服務
- 運作狀態
- 列出可用的 JICS 交易

- 啟動 JICS 交易
- 啟動 JICS 交易 （替代）
- 列出作用中工作階段

列出已註冊的程式

- 支援的方法：GET
- 路徑： `/programs`
- 以 html 頁面的形式傳回已註冊的程式清單。每個程式都由其主要程式識別符指定。清單中會傳回現代化舊版程式和公用程式程式 (IDCAMS、ISBGENER 等）。請注意，可用的公用程式將取決於已部署在 tomcat 伺服器上的公用程式 Web 應用程式。例如，z/OS 公用程式支援計劃可能無法用於現代化 iSeries 資產，因為它們不相關。

列出已註冊的服務

- 支援的方法：GET
- 路徑： `/services`
- 以 html 頁面的形式傳回已註冊的執行時間服務清單。指定服務由 AWS Blu Age 執行期做為公用程式提供，可用於 groovy 指令碼中的執行個體。Blusam 載入服務 （從舊式資料集建立 Blusam 資料集） 屬於該類別。

回應範例：

```
<p>BluesamESDSFileLoader</p><p>BluesamKSDSFileLoader</p><p>BluesamRRDSFileLoader</p>
```

運作狀態

- 支援的方法：GET
- 路徑： `/`
- 傳回簡單的訊息，指出 gapwalk-application 已啟動並執行 (`Jics application is running.`)

列出可用的 JICS 交易

- 支援的方法：GET
- 路徑： `/transactions`

- 傳回列出所有可用 JICS 交易的 html 頁面。這僅適用於具有 JICS 元素（舊版 CICS 元素的現代化）的環境。

回應範例：

```
<p>INQ1</p><p>MENU</p><p>MNT2</p><p>ORD1</p><p>PRNT</p>
```

啟動 JICS 交易

- 支援的方法：GET、POST
- 路徑：`/jicstransrunner/{jtrans:.+}`
- 引數：
  - JICS 交易識別符（字串、必要）：要啟動之 JICS 交易的識別符（長度上限為 8 個字元）。
  - 必要：要傳遞至交易的其他輸入資料，做為 Map<String，Object>。此映射的內容將用於饋送 JICS 交易將使用的 COMMAREA。如果不需要資料即可執行交易，則映射可以是空的。
  - 選用：Http 標頭項目，以自訂指定交易的執行環境。支援下列標頭索引鍵：
    - `jics-channel`：此交易啟動將啟動的程式將使用的 JICS CHANNEL 名稱。
    - `jics-container`：用於此 JICS 交易啟動的 JICS CONTAINER 名稱。
    - `jics-startcode`：在 JICS 交易開始時使用的 STARTCODE（字串，最多 2 個字元）。如需可能的值，請參閱 STARTCODE（向下瀏覽頁面）。
    - `jicxa-xid` ：「全域交易」(XA) 的 XID (X/Open 交易識別符 XID 結構），由發起人啟動，目前 JICS 交易啟動將參與其中。
- 傳回 `com.netfective.bluage.gapwalk.rt.shared.web.TransactionResultBean` JSON 序列化，代表 JICS 交易啟動的結果。

如需 結構詳細資訊的詳細資訊，請參閱 交易啟動結果結構。

啟動 JICS 交易（替代）

- 支援的方法：GET、POST
- 路徑：`/jicstransaction/{jtrans:.+}`
- 引數：

  JICS 交易識別符（字串、必要）

  要啟動之 JICS 交易的識別符（長度上限為 8 個字元）。

必要：要傳遞至交易的其他輸入資料，做為 Map<String，Object>

此映射的內容將用於饋送 JICS 交易將使用的 COMMAREA。如果不需要資料即可執行交易，則映射可以是空的。

選用：Http 標頭項目，以自訂指定交易的執行環境。

支援下列標頭索引鍵：

- `jics-channel`：此交易啟動將啟動的程式將使用的 JICS CHANNEL 名稱。
- `jics-container`：用於此 JICS 交易啟動的 JICS CONTAINER 名稱。
- `jics-startcode`：在 JICS 交易開始時使用的 STARTCODE （字串，最多 2 個字元）。如需可能的值，請參閱 STARTCODE （向下瀏覽頁面）。
- `jicxa-xid` ：「全域交易」(XA) 的 XID (X/Open 交易識別符 XID 結構），由發起人啟動，目前 JICS 交易啟動將參與其中。

- 傳回 `com.netfective.bluage.gapwalk.rt.shared.web.RecordHolderBean` JSON 序列化，代表 JICS 交易啟動的結果。您可以在 中找到結構的詳細資訊交易啟動記錄結果結構。

列出作用中工作階段

- 支援的方法：GET、POST
- 路徑： `/activesessionlist`
- 引數：無
- 傳回 JSON 序列化`com.netfective.bluage.gapwalk.application.web.sessiontracker.SessionTrackerOb`的 清單，代表作用中使用者工作階段的清單。停用工作階段追蹤時，將傳回空清單。

任務佇列相關的端點

任務佇列是 AS400 任務提交機制的 AWS Blu Age 支援。任務佇列用於 AS400，以在特定執行緒集區上執行任務。任務佇列是由名稱和對應於該佇列上可同時執行之程式數目上限的執行緒數目上限所定義。如果在佇列上提交的任務超過執行緒數量上限，任務會等待執行緒可用。

如需佇列上任務狀態的詳細清單，請參閱 佇列上任務的可能狀態。

任務佇列的操作是透過下列專用端點處理。您可以使用下列根 URL 從 Gapwalk 應用程式 URL 叫用這些操作：`http://server:port/gapwalk-application/jobqueue`。

主題

- 列出可用的佇列
- 啟動或重新啟動任務佇列
- 提交任務以啟動
- 列出所有提交的任務
- 釋出「保留」的所有任務
- 釋出指定任務名稱「保留」的所有任務
- 釋出任務編號的指定任務
- 依重複排程提交任務
- 列出所有提交的重複任務
- 取消重複任務的排程

列出可用的佇列

- 支援的方法：GET
- 路徑： `list-queues`
- 傳回可用佇列清單及其狀態，做為鍵值的 JSON 清單。

回應範例：

```
{"Default":"STAND_BY","queue1":"STARTED","queue2":"STARTED"}
```

任務佇列的可能狀態為：

STAND_BY

任務佇列正在等待啟動。

STARTED (已啟動)

任務佇列已啟動並執行。

UNKNOWN (不明)

無法判斷任務佇列狀態。

啟動或重新啟動任務佇列

- 支援的方法：POST
- 路徑： `/restart/{name}`
- 引數：要啟動/重新啟動的佇列名稱，做為字串 - 必要。
- 端點不會傳回任何內容，而是倚賴 http 狀態來指出開始/重新啟動操作的結果：

  HTTP 200

  開始/重新啟動操作進展良好：指定的任務佇列現在已啟動。

  HTTP 404

  任務佇列不存在。

  HTTP 503

  啟動/重新啟動嘗試期間發生例外狀況（應檢查伺服器日誌，以了解發生了什麼問題）。

提交任務以啟動

- 支援的方法：POST
- 路徑： `/submit`
- 引數：做為請求內文的必要項目，即`com.netfective.bluage.gapwalk.rt.jobqueue.SubmitJobMessage`物件的 JSON 序列化。如需詳細資訊，請參閱提交任務並排程任務輸入。
- 傳回包含原始 的 JSON，`SubmitJobMessage`以及指出任務是否已提交的日誌。

列出所有提交的任務

- 支援的方法：GET
- 路徑： `/list-jobs?status={status}&size={size}&page={page}&sort={sort}`
- 引數：
  - 頁面：要擷取的頁面號碼（預設值 = 1)
  - 大小：頁面大小（預設值 = 50，最大值 = 300)
  - sort： 任務的順序。（預設值 = “executionId”)。 “executionId” 目前是唯一支援的值
  - 狀態：（選用）如果存在，則會篩選狀態。
- 以 JSON 字串傳回所有排程任務的清單。如需回應範例，請參閱 排程任務回應的清單。

釋出「保留」的所有任務

- 支援的方法：POST
- 路徑： `/release-all`
- 傳回訊息，指出發行嘗試操作的結果。這裡有兩個可能的情況：
  - HTTP 200 和訊息「成功發佈所有任務！」 如果成功釋出所有任務。
  - HTTP 503 和訊息「未發佈任務」。發生未知的錯誤。如果發行嘗試發生錯誤，請參閱日誌以取得更多詳細資訊。

釋出指定任務名稱「保留」的所有任務

對於指定的任務名稱，可以使用不同的任務編號提交多個任務（任務執行的獨佔性是由兩個 <任務名稱，任務編號> 授予）。端點會嘗試釋出具有指定任務名稱的所有任務提交，這些名稱「保留中」。

- 支援的方法：POST
- 路徑： `/release/{name}`
- 引數：要尋找的任務名稱，做為字串。強制性。
- 傳回訊息，指出發行嘗試操作的結果。這裡有兩個可能的情況：
  - HTTP 200 和訊息「群組中的任務 <name> (< 已發行任務的數目 >) 成功發佈！」 任務已成功釋出。
  - HTTP 503 和訊息「群組 <name> 中的任務未發佈。發生未知錯誤。如果發行嘗試發生錯誤，請參閱日誌以取得更多詳細資訊。

釋出任務編號的指定任務

端點會嘗試針對指定的夫偶 <job name， job number> 釋出「on hold」的唯一任務提交。

- 支援的方法：POST
- 路徑： `/release/{name}/{number}`
- 引數：

  name

  要尋找的任務名稱，做為字串。強制性。

  number

  要尋找的任務編號，以整數表示。強制性。

傳回

指出發行嘗試操作結果的訊息。這裡有兩個可能的情況：

- HTTP 200 和訊息「成功發佈任務 <name/number>！」 如果任務已成功釋出。
- HTTP 503 和訊息「Job <name/number>>未發佈。發生未知錯誤。如果發行嘗試發生錯誤，請參閱日誌以取得更多詳細資訊。

依重複排程提交任務

使用重複排程來排程要執行的任務。

- 支援的方法：POST
- 路徑： `/schedule`
- 引數：請求內文必須包含`com.netfective.bluage.gapwalk.rt.jobqueue.SubmitJobMessage`物件的 JSON 序列化。

列出所有提交的重複任務

- 支援的方法：GET
- 路徑： `/schedule/list?status={status}&size={size}&page={page}&sort={sort}`
- 引數：
  1. 頁面：要擷取的頁碼（預設值 = 1)
  2. 大小：頁面大小（預設值 = 50，最大值 = 300)
  3. sort： 任務的順序。（預設值 = “id”)。 “id” 是目前唯一支援的值。
  4. 狀態：（選用） 如果存在，則會篩選狀態。可能的值是第 1 節所述的值。
  5. 狀態：（選用） 如果存在，則會篩選狀態。可能的值是第 1 節所述的值。
  6. 以 JSON 字串傳回所有排程任務的清單。

取消重複任務的排程

移除以重複排程建立的任務。任務排程狀態設定為非作用中。

- 支援的方法：GET
- 路徑： `/schedule/remove/{schedule_id}`

- 引數：schedule_id，要移除之排程任務的識別符。

Blusam 應用程式主控台 REST 端點

在本節中，您可以了解 Blusam 應用程式主控台，這是專為簡化現代化 VSAM 資料集管理而設計的 API。Blusam Web 應用程式的端點使用根路徑 /bac。

主題

- 資料集相關端點
- 大量資料集相關端點
- 記錄
- 遮罩
- 其他
- BAC 使用者管理端點

資料集相關端點

使用下列端點來建立或管理特定資料集。

主題

- 建立資料集
- 上傳檔案
- 載入資料集 (POST)
- 載入資料集 (GET)
- 從 Amazon S3 儲存貯體載入資料集
- 將資料集匯出至 Amazon S3 儲存貯體
- 清除資料集
- 刪除資料集
- 計數資料集記錄

建立資料集

您可以使用此端點來建立資料集定義。

- 支援的方法：POST
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑： `/api/services/rest/bluesamservice/createDataSet`
- 引數：

  name

  （必要，字串）：資料集的名稱。

  type

  （必要，字串）：資料集類型。可能的值為：ESDS、KSDS、RRDS。

  recordSize

  （選用，字串）：資料集每個記錄的大小上限。

  fixedLength

  （選用，布林值）：指出記錄長度是否固定。

  compression

  （選用，布林值）：指出資料集是否已壓縮。

  cacheEnable

  （選用，布林值）：指出是否已為資料集啟用快取。

  alternativeKeys

  （選用，索引鍵清單）：
  - 位移（必要，數字）
  - length（必要，數字）
  - name（必要，數字）
- 傳回代表新建立資料集的 JSON 檔案。

請求範例：

```
POST /api/services/rest/bluesamservice/createDataSet
{
  "name": "DATASET",
  "checked": false,
  "records": [],
```

```
  "primaryKey": {
    "name": "PK"
  },
  "alternativeKeys": [
    {
      "offset": 10,
      "length": 10,
      "name": "ALTK_0"
    }
  ],
  "type": "ESDS",
  "recordSize": 10,
  "compression": true,
  "cacheEnable": true
}
```

回應範例：

```
{
    "dataSet": {
      "name": "DATASET",
      "checked": false,
      "nbRecords": 0,
      "keyLength": -1,
      "recordSize": 10,
      "compression": false,
      "fixLength": true,
      "type": "ESDS",
      "cacheEnable": false,
      "cacheWarmup": false,
      "cacheEviction": "100ms",
      "creationDate": 1686744961234,
      "modificationDate": 1686744961234,
      "records": [],
      "primaryKey": {
        "name": "PK",
        "offset": null,
        "length": null,
        "columns": null,
        "unique": true
      },
      "alternativeKeys": [
        {
```

```
          "offset": 10,
          "length": 10,
          "name": "ALTK_0"
        }
      ],
      "readLimit": 0,
      "readEncoding": null,
      "initCharacter": null,
      "defaultCharacter": null,
      "blankCharacter": null,
      "strictZoned": null,
      "decimalSeparator": null,
      "currencySign": null,
      "pictureCurrencySign": null
    },
    "message": null,
    "result": true
  }
```

上傳檔案

您可以使用此端點將檔案上傳至伺服器。檔案會存放在對應至每個特定使用者的暫存資料夾中。每次您需要上傳檔案時，都使用此端點。

- 支援的方法：POST
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑： `/api/services/rest/bluesamservice/upload`
- 引數：

  file

  （必要，分段/表單資料）：要上傳的檔案。
- 傳回反映上傳狀態的布林值

載入資料集 (POST)

使用 `createDataSet`建立資料集定義後，您可以將與上傳檔案相關聯的記錄載入至特定資料集。

- 支援的方法：POST
- 需要身分驗證和 ROLE_ADMIN 角色。

- 路徑： `/api/services/rest/bluesamservice/loadDataSet`
- 引數：

  name

  （必要，字串）：資料集的名稱。
- 傳回請求的狀態和載入的資料集。

載入資料集 (GET)

- 支援的方法：GET
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑： `/api/services/rest/bluesamservice/loadDataSet`
- 引數：

  name

  （必要，字串）：資料集的名稱。

  資料集檔案

  （必要，字串）：資料集檔案名稱。
- 傳回請求的狀態和載入的資料集。

從 Amazon S3 儲存貯體載入資料集

使用 listcat 檔案從 Amazon S3 儲存貯體載入資料集。

- 支援的方法：GET
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑： `/api/services/rest/bluesamservice/loadDataSetFromS3`
- 引數：

  listcatFileS3Location

  （必要，字串）：Listcat 檔案的 Amazon S3 位置。

  datasetFileS3Location

  （必要，字串）：資料集檔案的 Amazon S3 位置。

region

（必要，字串）：存放檔案的 Amazon S3 AWS 區域 。

- 傳回新建立的資料集

請求範例：

```
/BAC/api/services/rest/bluesamservice/loadDataSetFromS3?region=us-
east-1&listcatFileS3Location=s3://bucket-name/listcat.json&datasetFileS3Location=s3://
bucket-name/dataset.DAT
```

將資料集匯出至 Amazon S3 儲存貯體

將資料集匯出至指定的 Amazon S3 儲存貯體。

- 支援的方法：GET
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑： `/api/services/rest/bluesamservice/exportDataSetToS3`
- 引數：

  s3Location

  （必要，字串）：要匯出資料集的 Amazon S3 位置。

  datasetName

  （必要，字串）：要匯出的資料集名稱。

  region

  （必要，字串）：Amazon S3 儲存貯 AWS 區域 體的 。

  kmsKeyId

  （選用，字串）：用於將匯出資料集加密至 Amazon S3 儲存貯體的 AWS KMS ID。
- 傳回匯出的資料集

請求範例：

```
/BAC/api/services/rest/bluesamservice/exportDataSetToS3?region=eu-
west-1&s3Location=s3://bucket-name/dump&datasetName=dataset
```

清除資料集

清除資料集中的所有記錄。

- 支援的方法：POST、GET
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑： `/api/services/rest/bluesamservice/clearDataSet`
- 引數：

  name

  （必要，字串）：要清除的資料集名稱。
- 傳回請求的狀態。

刪除資料集

刪除資料集定義和記錄。

- 支援的方法：POST
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑： `/api/services/rest/bluesamservice/deleteDataSet`
- 引數：

  name

  （必要，字串）：要刪除的資料集名稱。
- 傳回請求的狀態和已刪除的資料集。

計數資料集記錄

此端點會傳回與資料集相關聯的記錄數目。

- 支援的方法：POST
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/bluesamservice/countRecords`
- 引數：

name

（必要，字串）：資料集的名稱。

- 傳回：記錄數

大量資料集相關端點

使用下列端點一次建立或管理多個資料集。

主題

- 匯出資料集 (GET)
- 匯出資料集 (POST)
- 建立多個資料集
- 列出所有資料集
- 直接列出所有資料集
- 依頁面直接列出所有資料集
- 串流資料集
- 刪除所有資料集
- 從 listcat 檔案取得資料集定義
- 從上傳的清單目錄檔案取得資料集定義
- 取得資料集
- 從 JSON 檔案載入 listcat

匯出資料集 (GET)

- 支援的方法：GET
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/bluesamservice/exportDataSet`
- 引數：

datasetName

（必要，字串）：要匯出的資料集名稱。

datasetOutputFile

（必要，字串）：您要在伺服器上存放匯出資料集的資料夾路徑。

rdw

（必要，布林值）：您是否希望記錄描述項字詞 (RDW) 成為匯出記錄的一部分。如果資料集具有固定長度的記錄，則會忽略此參數的值。

- 傳回請求的狀態，以及包含匯出資料集之檔案的路徑（如果有的話）。如果回應中的資料集為 null，表示系統找不到具有指定名稱的資料集。

匯出資料集 (POST)

- 支援的方法：POST
- 需要身分驗證和 ROLE_USER 角色。
- 路徑：`/api/services/rest/bluesamservice/exportDataSet`
- 引數：

dumpParameters

（必要，BACReadParameters)：Bluesam 讀取參數。

- 傳回匯出資料集的狀態。

建立多個資料集

- 支援的方法：POST
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑：`/api/services/rest/bluesamservice/createAllDataSets`
- 引數：
  - 資料集清單

name

（必要，字串）：資料集的名稱。

type

（必要，字串）：資料集類型。可能的值為：ESDS、KSDS、RRDS。

recordSize

（選用，字串）：資料集的每個記錄的大小上限。

fixedLength

（選用，布林值）：指出記錄長度是否固定。

compression

（選用，布林值）：指出資料集是否已壓縮。

cacheEnable

（選用，布林值）：指出是否已為資料集啟用快取。

- 傳回：請求的狀態和新建立的資料集。

列出所有資料集

- 支援的方法：GET
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/bluesamservice/listDataSet`
- 引數：無
- 傳回：請求的狀態和資料集清單。

直接列出所有資料集

- 支援的方法：GET
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/bluesamservice/directListDataSet`
- 引數：無
- 傳回：請求的狀態和資料集清單。

依頁面直接列出所有資料集

- 支援的方法：GET
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/bluesamservice/directListDataSetByPage`

- 引數：

  datasetName

  （必要，字串）：資料集的名稱。

  pageNumber

  （必要，int)：頁碼。

  pageSize

  （必要，整數）：頁面大小。
- 傳回：請求的狀態和資料集清單。

串流資料集

- 支援的方法：GET
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑： `/api/services/rest/bluesamservice/streamDataset`
- 引數：

  datasetName

  （必要，字串）：資料集的名稱。
- 傳回：請求的資料集串流。

刪除所有資料集

- 支援的方法：POST
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑： `/api/services/rest/bluesamservice/removeAll`
- 引數：無
- 傳回：代表請求狀態的布林值。

從 listcat 檔案取得資料集定義

- 支援的方法：POST
- 需要身分驗證和 ROLE_ADMIN 角色。

- 路徑： `/api/services/rest/bluesamservice/getDataSetsDefinitionFromListcat`
- 引數：

  paramFilePath

  （必要，字串）：Listcat 檔案的路徑。
- 傳回：資料集清單

從上傳的清單目錄檔案取得資料集定義

- 支援的方法：POST
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑： `/api/services/rest/bluesamservice/getDataSetsDefinitionFromUploadedListcat`
- 引數：無
- 傳回：資料集清單

取得資料集

- 支援的方法：GET
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/bluesamservice/getDataSet`
- 引數：

  name

  （必要，字串）：資料集的名稱。
- 傳回請求的資料集。

從 JSON 檔案載入 listcat

- 支援的方法：GET
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑： `/api/services/rest/bluesamservice/loadListcatFromJsonFile`
- 引數：

filePath

（必要，字串）：Listcat 檔案的路徑。

- 傳回：資料集清單

記錄

使用下列端點來建立或管理資料集內的記錄。

主題

- 建立記錄
- 讀取資料集
- 刪除記錄
- 更新記錄
- 儲存記錄
- 驗證記錄
- 取得記錄樹狀目錄

建立記錄

您可以使用此端點來建立新的記錄。

- 支援的方法：POST
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/crud/createRecord`
- 引數：

  dataset

  （必要，DataSet)：資料集物件

  遮罩

  （必要，遮罩）：遮罩物件。
- 傳回請求的狀態和建立的記錄。

讀取資料集

您可以使用此端點來讀取資料集。

- 支援的方法：POST
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/crud/readDataSet`
- 引數：

  dataset

  （必要，DataSet)：資料集物件。
- 傳回請求的狀態和包含記錄的資料集。

刪除記錄

您可以使用此端點從資料集刪除記錄。

- 支援的方法：POST
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/crud/deleteRecord`
- 引數：

  dataset

  （必要，DataSet)：資料集物件

  record

  （必要，記錄）：要刪除的記錄
- 傳回刪除的狀態。

更新記錄

您可以使用此端點來更新與資料集相關聯的記錄。

- 支援的方法：POST
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/crud/updateRecord`

- 引數：

  dataset

  （必要，DataSet）：資料集物件

  record

  （必要，記錄）：要更新的記錄
- 傳回請求的狀態和包含記錄的資料集。

儲存記錄

您可以使用此端點將記錄儲存到資料集並使用遮罩。

- 支援的方法：POST
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/crud/saveRecord`
- 引數：

  dataset

  （必要，DataSet）：資料集物件

  record

  （必要，記錄）：要儲存的記錄
- 傳回請求的狀態和包含記錄的資料集。

驗證記錄

使用此端點來驗證記錄。

- 支援的方法：POST
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/crud/validateRecord`
- 引數：

  dataset

  （必要，DataSet）：資料集物件

- 傳回請求的狀態和包含記錄的資料集。

取得記錄樹狀目錄

使用此端點來取得記錄的階層樹狀結構。

- 支援的方法：POST
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/crud/getRecordTree`
- 引數：

  dataset

  （必要，DataSet)：資料集物件

  record

  （必要，記錄）：要擷取的記錄
- 傳回請求的狀態和請求記錄的階層樹狀結構。

遮罩

使用下列端點將遮罩載入或套用至資料集。

主題

- 載入遮罩
- 套用遮罩
- 套用遮罩篩選條件

載入遮罩

您可以使用此端點來擷取與特定資料集相關聯的所有遮罩。

- 支援的方法：POST
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/crud/loadMasks`
- 路徑變數：

recordSize：.../loadMasks/{recordSize}

（選用，數值）：記錄大小，篩選符合此記錄大小的載入遮罩

- 引數：

  dataset

  （必要，DataSet)：資料集物件
- 傳回請求的狀態和遮罩清單。

套用遮罩

您可以使用此端點將遮罩套用至特定資料集。

- 支援的方法：POST
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/crud/applyMask`
- 引數：

  dataset

  （必要，DataSet)：資料集物件

  遮罩

  （必要，遮罩）：資料集物件
- 傳回請求的狀態和已套用遮罩的資料集。

套用遮罩篩選條件

您可以使用此端點，將遮罩和篩選條件套用至特定資料集。

- 支援的方法：POST
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/crud/applyMaskFilter`
- 引數：

  dataset

  （必要，DataSet)：資料集物件

遮罩

（必要，遮罩）：資料集物件

- 傳回請求的狀態，以及具有套用遮罩和篩選條件的資料集。

其他

使用下列端點來管理資料集的快取或檢查資料集特性

主題

- 檢查暖機快取
- 檢查快取已啟用
- 啟用快取
- 檢查配置的 RAM 快取
- 檢查持久性
- 檢查支援的資料集類型
- 檢查伺服器運作狀態

檢查暖機快取

檢查是否針對特定資料集啟用暖機快取。

- 支援的方法：POST
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑： `/api/services/rest/bluesamservice/warmupCache`
- 引數：

  name

  （必要，字串）：資料集的名稱。
- 如果啟用暖機快取，則傳回： true，否則傳回 false。

檢查快取已啟用

檢查是否針對特定資料集啟用快取。

- 支援的方法：GET
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/bluesamservice/isEnableCache`
- 引數：無
- 如果啟用快取，則傳回 true。

啟用快取

- 支援的方法：POST
- 需要身分驗證和 ROLE_ADMIN 和 ROLE_SUPER_ADMIN 角色。
- 路徑： `/api/services/rest/bluesamservice/enableDisableCache/{enable}`
- 引數：

  啟用

  （必要，布林值）：如果設定為 true，則會啟用快取。
- 傳回無

檢查配置的 RAM 快取

您可以使用此端點來擷取配置的 RAM 快取記憶體。

- 支援的方法：GET
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/bluesamservice/allocatedRamCache`
- 引數：無
- 傳回：記憶體的大小做為字串

檢查持久性

- 支援的方法：GET
- 需要身分驗證和 ROLE_USER 角色。
- 路徑： `/api/services/rest/bluesamservice/persistence`
- 引數：無

- 傳回：用作字串的持久性

檢查支援的資料集類型

- 支援的方法：GET
- 路徑： `/api/services/rest/bluesamservice/getDataSetTypes`
- 需要身分驗證和 ROLE_USER 角色。
- 引數：無
- 傳回：支援的資料集類型清單做為字串清單。

檢查伺服器運作狀態

- 支援的方法：GET
- 路徑： `/api/services/rest/bluesamserver/serverIsUp`
- 引數：無
- 傳回：無。HTTP 回應狀態碼 200 表示伺服器已啟動並執行。

BAC 使用者管理端點

使用下列端點來管理使用者互動。

主題

- 在 中記錄使用者
- 驗證系統中是否至少有一個使用者
- 記錄新使用者
- 取得使用者資訊
- 列出使用者
- 刪除使用者
- 登出目前的使用者

在 中記錄使用者

- 支援的方法：POST
- 路徑： `/api/services/security/servicelogin/login`

- 引數：無
- 傳回com.netfective.bluage.bac.entities.SignOn物件的 JSON 序列化，代表目前請求中提供其登入資料的使用者。密碼會從傳回物件的檢視中隱藏。正在列出提供給使用 的角色。

回應範例：

```
{
     "login": "some-admin",
     "password": null,
     "roles": [
       {
         "id": 0,
         "roleName": "ROLE_ADMIN"
       }
     ]
   }
```

驗證系統中是否至少有一個使用者

- 支援的方法：GET
- 路徑： /api/services/security/servicelogin/hasAccount
- 引數：無
- true 如果至少已建立預設超級管理員使用者以外的一個使用者，則傳回布林值。false 否則傳回 。

記錄新使用者

- 支援的方法：POST
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑： /api/services/security/servicelogin/recorduser
- 引數：代表要新增至儲存體之使用者之com.netfective.bluage.bac.entities.SignOn物件的 JSON 序列化。必須定義使用者的角色，否則使用者可能無法使用 BAC 設施和端點。
- true 如果使用者已成功建立，則傳回布林值。false 否則傳回 。
- 請求 JSON 範例：

```
{
    "login": "simpleuser",
    "password": "simplepassword",
    "roles": [
      {
        "id": 2,
        "roleName": "ROLE_USER"
      }
    ]
  }
```

以下是 的兩個有效值`roleName`：

- `ROLE_ADMIN`： 可以管理 Blusam 資源和使用者。
- `ROLE_USER`： 可以管理 Blusam 資源，但無法管理使用者。

取得使用者資訊

- 支援的方法：GET
- 路徑： `/api/services/security/servicelogin/userInfo`
- 引數：無
- 傳回目前連線使用者的使用者名稱和角色

列出使用者

- 支援的方法：GET
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑： `/api/services/security/servicelogin/listusers`
- 引數：無
- 傳回 清單`com.netfective.bluage.bac.entities.SignOn`，序列化為 JSON。

刪除使用者

> **Important**
>
> 這個操作無法復原。刪除的使用者將無法再次連線至 BAC 應用程式。

- 支援的方法：POST
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑： `/api/services/security/servicelogin/deleteuser`
- 引數：代表要從儲存體中移除之使用者的`com.netfective.bluage.bac.entities.SignOn`物件 JSON 序列化。
- `true` 如果使用者已成功移除，則傳回布林值。

登出目前的使用者

- 支援的方法：GET
- 路徑： `/api/services/security/servicelogout/logout`
- 引數：無
- `{"success":true}` 如果目前使用者已成功登出，則傳回 JSON 訊息。相關的 HTTP 工作階段將會失效。

在 AWS 藍年管理 JICS 應用程式主控台

JICS 元件是對舊版 CICS 資源現代化的 AWS Blu Age 支援。JICS 應用程式主控台 Web 應用程式專用於管理 JICS 資源。下列端點允許 執行管理任務，而不必與 JAC 使用者介面互動。每當端點需要身分驗證時，請求就必須包含身分驗證詳細資訊（使用者名稱/密碼通常為基本身分驗證所需）。JICS 應用程式主控台 Web 應用程式的端點使用根路徑 `/jac/`。

主題

- JICS 資源管理
- 其他
- JAC 使用者管理端點

JICS 資源管理

下列所有端點都與 JICS 資源管理相關，讓 JICS 管理員每天處理資源。

主題

- 列出 JICS LISTS 和 GROUPS
- 擷取 JICS 資源

- 列出 JICS GROUPS
- 列出指定 LIST 的 JICS GROUPS
- 指定 GROUP 的 LIST JICS 資源
- 指定 GROUP 的 LIST JICS 資源 （使用名稱的替代資源）
- 編輯數個 LISTS 的擁有 GROUPS
- 刪除 LIST
- 刪除 GROUP
- 刪除交易
- 刪除程式
- 刪除檔案
- 刪除 TDQUEUE
- 刪除 TSMODEL
- 刪除元素
- 建立 LIST
- 建立 GROUP
- 常見的 RESOURCES 建立考量
- 建立交易
- 建立計畫
- 建立 檔案
- 建立 TDQUEUE
- 建立 TSMODEL
- 建立元素
- 更新 LIST
- 更新 GROUP
- 常見的 RESOURCES 更新考量
- 更新交易
- 更新程式
- 更新檔案

- 更新 TDQUEUE
- 更新 TSMODEL
- 更新元素
- Upsert 元素
- 擷取元素
- JICS CRUD 操作

列出 JICS LISTS 和 GROUPS

LIST 和 GROUPS 是 JICS 元件中擁有容器資源的主要擁有者。所有 JICS 資源都必須屬於 GROUP。群組可以屬於 LISTS，但這不是強制性的。LISTS 甚至可能不存在於指定的 JICS 環境，但大多數情況下，LISTS 會為資源提供多一層的組織。如需 CICS 資源組織的詳細資訊，請參閱 CICS 資源。

- 支援的方法：GET
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/listJicsListsAndGroups`
- 引數：無
- 傳回：序列化 JicsContainer 物件的清單，包括 LISTS 和 GROUPS，以 JSON 表示。

回應範例：

```
[
    {
      "name": "Resources",
      "children": [
        {
          "jacType": "JACList",
          "name": "MURACHS",
          "isActive": true,
          "children": [
            {
              "jacType": "JACGroup",
              "name": "MURACHS",
              "isActive": true,
              "children": []
            }
          ]
```

```
        },
        {
          "jacType": "JACGroup",
          "name": "TEST",
          "isActive": true,
          "children": []
        }
      ],
      "isExpanded": true
    }
  ]
```

擷取 JICS 資源

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/retrieveJicsResources`
- 引數：代表您要擷取之 JICS 資源的 JSON 承載。這是`com.netfective.bluage.jac.entities.request.RetrieveOperationRequest`物件的 JSON 序列化。
- 傳回：序列化 JicsResource 物件的清單。這些物件不會以特定順序傳回，且類型不同，例如 PROGRAM、TRANSACTION、FILE 等。

列出 JICS GROUPS

- 支援的方法：GET
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/listJicsGroups`
- 引數：無
- 以 JSON 傳回序列化 JicsContainer 物件 (GROUPS) 的清單。GROUPS 會傳回，而沒有其擁有的 LIST 資訊。

回應範例：

```
[
    {
      "jacType": "JACGroup",
```

```
    "name": "MURACHS",
    "isActive": true,
    "children": []
  },
  {
    "jacType": "JACGroup",
    "name": "TEST",
    "isActive": true,
    "children": []
  }
]
```

列出指定 LIST 的 JICS GROUPS

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/listGroupsForList`
- 引數：JSON 承載，代表您要尋找其 GROUPS 的 JICS LIST。這是`com.netfective.bluage.jac.entities.JACList`物件的 JSON 序列化。

  請求範例：

  ```
  {
      "jacType":"JACList",
      "name":"MURACHS",
      "isActive":true
    }
  ```

- 傳回序列化 JicsContainer 物件 (GROUPS) 的清單，做為 JSON，連接到指定的 LIST。GROUPS 會傳回，而沒有其擁有的 LIST 資訊。

  回應範例：

  ```
  [
      {
        "jacType": "JACGroup",
        "name": "MURACHS",
        "isActive": true,
        "children": []
      }
    ]
  ```

指定 GROUP 的 LIST JICS 資源

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/listResourcesForGroup`
- 引數：JSON 承載，代表您要尋找其資源的 JICS GROUP。這是`com.netfective.bluage.jac.entities.JACGroup`物件的 JSON 序列化。您不需要指定 GROUP 的所有欄位，但名稱是必要的。

  請求範例：

```
{
    "jacType":"JACGroup",
    "name":"MURACHS",
    "isActive":true
  }
```

- 傳回指定 GROUP 所擁有的序列化 JicsResource 物件清單。這些物件不會以特定順序傳回，且類型不同，例如 PROGRAM、TRANSACTION、FILE 等。

指定 GROUP 的 LIST JICS 資源（使用名稱的替代資源）

- 支援的方法：POST
- 需要身分驗證
- 路徑： `/api/services/rest/jicsservice/listResourcesForGroupName`
- 引數：擁有您要尋找之資源的 GROUP 名稱。
- 傳回：由指定 GROUP 擁有的序列化 JicsResource 物件清單。這些物件不會以特定順序傳回，且具有不同類型的物件，例如 PROGRAM、TRANSACTION、FILE 等。

編輯數個 LISTS 的擁有 GROUPS

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/editGroupsList`
- 引數：具有子 GROUPS 的 LISTS 集合的 JSON 表示法；

  請求範例：

```
[
  {
        "jacType": "JACList",
        "name": "MURACHS",
        "isActive": true,
        "children": [
          {
            "jacType": "JACGroup",
            "name": "MURACHS",
            "isActive": true,
            "children": []
          },
          {
            "jacType": "JACGroup",
            "name": "TEST",
            "isActive": true,
            "children": []
          }
        ]
  }
]
```

在此編輯之前，只有名為 "MURACHS" 的群組屬於名為 "MURACHS" 的 LIST。使用此編輯，您可以將名為 "TEST" 的群組新增至名為 "MURACHS" 的 LIST。

- 傳回布林值。如果值為 'true'，表示 LISTS 修改已成功保留到基礎 JICS 儲存體。

刪除 LIST

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/deleteList`
- 引數：JSON 承載，代表要刪除的 JICS LIST。這是`com.netfective.bluage.jac.entities.JACList`物件的 JSON 序列化。
- 傳回布林值。如果值為 'true'，表示已成功在基礎 JICS 儲存體上操作 LIST 刪除。

刪除 GROUP

- 支援的方法：POST

- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/deleteGroup`
- 引數：JSON 承載，代表要刪除的 JICS GROUP。這是`com.netfective.bluage.jac.entities.JACGroup`物件的 JSON 序列化。
- 傳回布林值。如果值為 'true'，表示已成功在基礎 JICS 儲存體上操作 GROUP 刪除。

刪除交易

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/deleteTransaction`
- 引數：JSON 承載，代表要刪除的 JICS 交易。這是`com.netfective.bluage.jac.entities.JACTransaction`物件的 JSON 序列化。
- 傳回布林值。如果值為 'true'，則已成功在基礎 JICS 儲存體上操作 TRANSACTION 刪除。

刪除程式

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/deleteProgram`
- 引數：JSON 承載，代表要刪除的 JICS 程式。這是`com.netfective.bluage.jac.entities.JACProgram`物件的 JSON 序列化。
- 傳回布林值。如果值為「true」，表示 PROGRAM 刪除已成功在基礎 JICS 儲存體上操作。

刪除檔案

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/deleteFile`
- 引數：JSON 承載，代表要刪除的 JICS 檔案。這是`com.netfective.bluage.jac.entities.JACFile`物件的 JSON 序列化。
- 傳回布林值。如果值為 'true'，表示已成功在基礎 JICS 儲存體上操作 FILE 刪除。

刪除 TDQUEUE

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/deleteTDQueue`
- 引數：JSON 承載，代表要刪除的 JICS TDQUEUE。這是 `com.netfective.bluage.jac.entities.JACTDQueue` 物件的 JSON 序列化。
- 傳回布林值。如果值為 'true'，則已成功在基礎 JICS 儲存體上操作 TDQUEUE 刪除。

刪除 TSMODEL

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/deleteTSModel`
- 引數：JSON 承載，代表要刪除的 JICS TSMODEL。這是 `com.netfective.bluage.jac.entities.JACTSModel` 物件的 JSON 序列化。
- 傳回布林值。如果值為「true」，TSMODEL 刪除已成功在基礎 JICS 儲存體上操作。

刪除元素

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/deleteElements`
- 引數：代表要刪除之 JICS 元素的 JSON 承載。
- 傳回布林值，其中 `true` 表示刪除已成功在基礎 JICS 儲存體中操作。

建立 LIST

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/createList`
- 引數：JSON 承載，代表要建立的 JICS LIST。這是 `com.netfective.bluage.jac.entities.JACList` 物件的 JSON 序列化。

- 傳回布林值。如果值為 'true'，表示已成功在基礎 JICS 儲存體中建立 LIST。

> **Note**
>
> LIST 將一律為空白建立。將 GROUPS 連接到 LIST 需要另一個操作。

建立 GROUP

- 支援的方法：POST
- 需要身分驗證和下列角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/createGroup`
- 引數：JSON 承載，代表要建立的 JICS GROUP。這是`com.netfective.bluage.jac.entities.JACGroup`物件的 JSON 序列化。
- 傳回布林值。如果值為「true」，表示已在基礎 JICS 儲存體中正確建立 GROUP。

> **Note**
>
> GROUP 將一律為空。將資源連接至 GROUP 需要額外的操作（建立資源會自動將其連接至指定的 GROUP。

常見的 RESOURCES 建立考量

下列所有端點都與 JICS RESOURCES 建立相關，並共用一些常見的限制條件：在要傳送至端點的請求承載中，必須重視 `groupName` 欄位。

GROUP 擁有權限制：

無法在未連接至現有群組的情況下建立資源，且端點會使用 groupName 擷取要連接此資源的群組。必須`groupName`指向現有 GROUP 的名稱。如果 `groupName` 未指向 JICS 基礎儲存體中的現有群組，則會傳送具有 HTTP STATUS 400 的錯誤訊息。

GROUP 內的 Unicity 限制條件：

具有指定名稱的指定資源在指定群組中必須是唯一的。每個資源建立端點都會執行單數檢查。如果指定的承載不遵守單一限制，端點將以 HTTP STATUS 400（錯誤請求）傳送回應 -- 請參閱以下範例回應。

承載範例：您嘗試在 'TEST' 群組中建立交易 'ARIT'，但具有該名稱的交易已存在於該群組中。

```
{
    "jacType":"JACTransaction",
    "name":"ARIT",
    "groupName":"TEST",
    "isActive":true
  }
```

您會收到下列錯誤回應：

```
{
    "timestamp": 1686759054510,
    "status": 400,
    "error": "Bad Request",
    "path": "/jac/api/services/rest/jicsservice/createTransaction"
  }
```

檢查伺服器日誌將確認問題的來源：

```
2023-06-14 18:10:54 default         TRACE - o.s.w.m.HandlerMethod
        - Arguments: [java.lang.IllegalArgumentException: Transaction already
 present in the group, org.springframework.security.web.header.HeaderWriterFilter
$HeaderWriterResponse@e34f6b8]
2023-06-14 18:10:54 default         ERROR - c.n.b.j.a.WebConfig                    -
 400
java.lang.IllegalArgumentException: Transaction already present in the group
 at
 com.netfective.bluage.jac.server.services.rest.impl.JicsServiceImpl.createElement(JicsServiceI
```

建立交易

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/createTransaction`
- 引數：JSON 承載，代表要建立的 JICS TRANSACTION。這是`com.netfective.bluage.jac.entities.JACTransaction`物件的 JSON 序列化。

- 傳回布林值。如果值為 'true'，表示已成功在基礎 JICS 儲存體中建立 TRANSACTION。

建立計畫

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/createProgram`
- 引數：JSON 承載，代表要建立的 JICS PROGRAM。這是`com.netfective.bluage.jac.entities.JACProgram`物件的 JSON 序列化。
- 傳回布林值。如果值為「true」，表示已成功在基礎 JICS 儲存中建立 PROGRAM。

建立 檔案

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/createFile`
- 引數：JSON 承載，代表要建立的 JICS 檔案。這是`com.netfective.bluage.jac.entities.JACFile`物件的 JSON 序列化。
- 傳回布林值。如果值為 'true'，表示已成功在基礎 JICS 儲存體中建立 FILE。

建立 TDQUEUE

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/createTDQueue`
- 引數：JSON 承載，代表要建立的 JICS TDQUEUE。這是`com.netfective.bluage.jac.entities.JACTDQueue`物件的 JSON 序列化。
- 傳回布林值。如果值為 'true'，則已成功在基礎 JICS 儲存體中建立 TDQUEUE。

建立 TSMODEL

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/createTSModel`

- 引數：JSON 承載，代表要建立的 JICS TSMODEL。這是`com.netfective.bluage.jac.entities.JACTSModel`物件的 JSON 序列化。
- 傳回布林值，其中 `true` 表示已成功在基礎 JICS 儲存體中操作元素的建立。

建立元素

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/createElements`
- 引數：代表要建立之 JICS 元素的 JSON 承載。
- 傳回布林值。如果值為「true」，則元素已成功在基礎 JICS 儲存中建立。

更新 LIST

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/updateList`
- 引數：JSON 承載，代表要更新的 JICS LIST。這是`com.netfective.bluage.jac.entities.JACList`物件的 JSON 序列化。您不需要提供 LIST 的子系；LISC 更新機制不會考慮子系。
- 傳回布林值。如果值為 'true'，表示已成功更新基礎 JICS 儲存中的 LIST。

更新 LIST 'isActive' 旗標將傳播至 LIST 的所有擁有元素，亦即 LIST 擁有的所有 GROUPS 和這些 GROUPS 擁有的所有資源。這是在多個 GROUPS 上透過單一操作停用許多資源的便利方式。

更新 GROUP

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/updateGroup`
- 引數：JSON 承載，代表要更新的 JICS GROUP。這是`com.netfective.bluage.jac.entities.JACGroup`物件的 JSON 序列化。不需要提供 GROUP 的子系，GROUP 更新機制不會考慮這一點。
- 傳回布林值。如果值為 'true'，表示已成功更新基礎 JICS 儲存中的 GROUP。

> **Note**
>
> 更新 GROUP 'isActive' 旗標將傳播到 GROUP 的所有擁有元素，即 GROUP 擁有的所有資源。這是在指定 GROUP 中使用單一操作停用許多資源的便利方式。

常見的 RESOURCES 更新考量

下列所有端點都是關於更新 JICS RESOURCES。使用 `groupName` 欄位，您可以變更任何 JICS 資源的擁有 GROUP，前提是欄位值指向基礎 JICS 儲存體中的現有 GROUP （否則，您會從端點收到 BAD REQUEST 回應 (HTTP STATUS 400))。

更新交易

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/updateTransaction`
- 引數：JSON 承載，代表要更新的 JICS TRANSACTION。這是`com.netfective.bluage.jac.entities.JACTransaction`物件的 JSON 序列化。
- 傳回布林值。如果值為 'true'，表示基礎 JICS 儲存體中的 TRANSACTION 已成功更新。

更新程式

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/updateProgram`
- 引數：JSON 承載，代表要更新的 JICS PROGRAM。這是`com.netfective.bluage.jac.entities.JACProgram`物件的 JSON 序列化。
- 傳回布林值。如果值為「true」，表示已成功更新基礎 JICS 儲存中的 PROGRAM。

更新檔案

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/updateFile`

- 引數：JSON 承載，代表要更新的 JICS 檔案。這是`com.netfective.bluage.jac.entities.JACFile`物件的 JSON 序列化。
- 傳回布林值。如果值為 'true'，表示已成功更新基礎 JICS 儲存中的 FILE。

更新 TDQUEUE

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/updateTDQueue`
- 引數：JSON 承載，代表要更新的 JICS TDQUEUE。這是`com.netfective.bluage.jac.entities.JACTDQueue`物件的 JSON 序列化。
- 傳回布林值。如果值為 'true'，則已成功更新基礎 JICS 儲存中的 TDQueue。

更新 TSMODEL

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/updateTSModel`
- 引數：JSON 承載，代表要更新的 JICS TSMODEL。這是`com.netfective.bluage.jac.entities.JACTSModel`物件的 JSON 序列化。
- 傳回布林值。如果值為 'true'，則基礎 JICS 儲存體中的 TSMODEL 已成功更新。

更新元素

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/updateElements`
- 引數：代表要更新的元素的 JSON 承載。
- 傳回布林值，其中 `true` 表示元素更新已成功在基礎 JICS 儲存體中操作。

Upsert 元素

- 支援的方法：POST

- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/upsertElements`
- 引數：代表 upsert 元素的 JSON 承載。
- 傳回布林值，其中 `true`表示元素 upsert 已成功在基礎 JICS 儲存體中操作。

擷取元素

- 支援的方法：GET
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/retrieveElements`
- 引數：無
- 傳回所有序列化 JICS 資源的清單。

JICS CRUD 操作

- 支援的方法：POST
- 需要身分驗證和下列其中一個角色：ROLE_ADMIN、ROLE_SUPER_ADMIN、ROLE_USER
- 路徑： `/api/services/rest/jicsservice/jicsCrudOperation`
- 引數：代表您要尋找的 JICS 資源的 JSON 承載。這是`com.netfective.bluage.jac.entities.request.JicsCrudOperationRequest`物件的 JSON 序列化。
- 傳回代表回應的 JSON 承載。這是`com.netfective.bluage.jac.entities.request.JicsCrudOperationResponse`物件的 JSON 序列化。

其他

主題

- JICS 伺服器運作狀態

JICS 伺服器運作狀態

- 支援的方法：GET

- 路徑： `/api/services/rest/jicsserver/serverIsUp`
- 引數：無
- 傳回：無。HTTP STATUS 200 回應表示伺服器已啟動並執行。

JAC 使用者管理端點

使用下列端點來管理使用者互動。

主題

- 記錄使用者
- 測試系統中是否至少存在 使用者
- 記錄新使用者
- 使用者資訊
- 列出使用者
- 刪除使用者
- 登出目前使用者

記錄使用者

- 支援的方法：POST
- 路徑： `/api/services/security/servicelogin/login`
- 引數：無
- 傳回`com.netfective.bluage.jac.entities.SignOn`物件的 JSON 序列化，代表目前請求中提供其登入資料的使用者。密碼會從傳回物件的檢視中隱藏。正在列出提供給使用 的角色。

回應範例：

```
{
    "login": "some-admin",
    "password": null,
    "roles": [
      {
        "id": 0,
        "roleName": "ROLE_ADMIN"
      }
```

```
    ]
  }
```

測試系統中是否至少存在 使用者

- 支援的方法：GET
- 路徑： `/api/services/security/servicelogin/hasAccount`
- 引數：無
- `true` 如果至少已建立預設超級管理員使用者以外的一個使用者，則傳回布林值。`false` 否則傳回 。

記錄新使用者

- 支援的方法：POST
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑： `/api/services/security/servicelogin/recorduser`
- 引數：`com.netfective.bluage.jac.entities.SignOn`物件的 JSON 序列化，代表要新增至儲存體的使用者。應定義使用者的 角色，否則使用者可能無法使用 JAC 設施和端點。
- `true` 如果使用者已成功建立，則傳回布林值。`false` 否則傳回 。

請求範例：

```
{
    "login": "simpleuser",
    "password": "simplepassword",
    "roles": [
      {
        "id": 2,
        "roleName": "ROLE_USER"
      }
    ]
  }
```

記錄新使用者時，只能使用下列角色：

- ROLE_ADMIN ：可以管理 JICS 資源和使用者。
- ROLE_USER ：可以管理 JICS 資源，但不能管理使用者。

使用者資訊

- 支援的方法：GET
- 路徑： `/api/services/security/servicelogin/userInfo`
- 引數：無
- 傳回目前連線使用者的使用者名稱和角色。

列出使用者

- 支援的方法：GET
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑： `/api/services/security/servicelogin/listusers`
- 引數：無
- 傳回 的清單`com.netfective.bluage.jac.entities.SignOn`，序列化為 JSON。

刪除使用者

- 支援的方法：POST
- 需要身分驗證和 ROLE_ADMIN 角色。
- 路徑： `/api/services/security/servicelogin/deleteuser`
- 引數：代表要從儲存體中移除之使用者的`com.netfective.bluage.jac.entities.SignOn`物件 JSON 序列化。
- `true` 如果使用者已成功移除，則傳回布林值。

> **Important**
>
> 這個操作無法復原。刪除的使用者將無法再次連線至 JAC 應用程式。

登出目前使用者

- 支援的方法：GET
- 路徑： `/api/services/security/servicelogout/logout`
- 引數：無

- `{"success":true}` 如果目前使用者已成功登出，則傳回 JSON 訊息。相關的 HTTP 工作階段將會失效。

AWS Blu Age 使用者的資料結構

您可以在下一節中了解 AWS Blu Age 引擎的各種資料結構。

主題

- 任務執行詳細資訊訊息結構
- 交易啟動結果結構
- 交易啟動記錄結果結構
- 佇列上任務的可能狀態
- 提交任務並排程任務輸入
- 排程任務回應的清單
- 重複任務回應的清單

任務執行詳細資訊訊息結構

每個任務執行詳細資訊都會有下列欄位：

scriptId

呼叫指令碼的識別符。

呼叫者

發起人的 I.P. 地址。

identifier

唯一的任務執行識別符。

startTime

任務執行開始的日期和時間。

endTime

任務執行結束的日期和時間。

status

任務執行的狀態。一個可能的數值：

- DONE：任務執行正常結束。
- TRIGGERED：任務執行已觸發，但尚未啟動。
- RUNNING：任務執行正在執行。
- KILLED：任務執行已終止。
- FAILED：任務執行失敗。

executionResult

訊息，以加總任務執行的結果。如果任務執行尚未完成，此訊息可以是簡單的訊息，也可以是具有下列欄位的 JSON 結構：

- exitCode：數字結束碼；負值表示失敗情況。
- 程式：任務啟動的最新程式。
- 狀態：一個可能的數值：
  - Error：當 exitCode = -1 時；這對應於任務執行期間發生的（技術）錯誤。
  - Failed：當 exitcode = -2 時；這對應於服務程式執行期間發生的失敗（例如 ABEND 情況）。
  - Succeeded：當 exitCode >= 0 時；
- stepName：任務中執行的最新步驟名稱。

executionMode

SYNCHRONOUS 或 ASYNCHRONOUS，取決於任務的啟動方式。

輸出範例：

```
{
    "scriptId": "INTCALC",
    "caller": "127.0.0.1",
    "identifier": "97d410be-efa7-4bd3-b7b9-d080e5769771",
    "startTime": "06-09-2023 11:42:41",
    "endTime": "06-09-2023 11:42:42",
    "status": "DONE",
    "executionResult": "{ \"exitCode\": -1, \"stepName\": \"STEP15\", \"program\":
 \"CBACT04C\", \"status\": \"Error\" }",
    "executionMode": "ASYNCHRONOUS"
```

```
  }
```

交易啟動結果結構

結構可能包含下列欄位：

outCome

代表交易執行結果的字串。可能值為：

- Success：交易執行正常結束。
- Failure：交易執行無法正常結束，遇到一些問題（某些問題）。

逗號

代表 COMMAREA 最終值的字串，以位元組 64 編碼的位元組陣列表示。可能是空字串。

containerRecord

（選用）字串，以位元組 64 編碼位元組陣列表示 CONTAINER 的記錄內容。

serverDescription

可能包含提供請求之伺服器的相關資訊（用於除錯目的）。可能是空字串。

abendCode

（選用）如果已中止啟動的交易所參考的程式，則抽象程式碼值會傳回為此欄位中的字串。

回應範例：

成功

```
{
    "outCome": "Success",
    "commarea": "",
    "serverDescription": ""
  }
```

失敗

```
{
    "outCome": "Failure",
    "commarea": "",
```

```
    "serverDescription": "",
    "abendCode": "AEIA"
  }
```

交易啟動記錄結果結構

結構可能包含下列欄位：

recordContent

代表 COMMAREA 記錄內容的字串，以位元組 64 編碼的位元組陣列表示。

containerRecord

字串，以位元組 64 編碼位元組陣列表示 CONTAINER 的記錄內容。

serverDescription

可能包含提供請求之伺服器的相關資訊（用於除錯目的）。可能是空字串。

回應範例：

成功

```
{
    "recordContent": "",
    "serverDescription": ""
}
```

佇列上任務的可能狀態

在佇列上，任務可以具有下列狀態：

ACTIVE

任務目前正在佇列上執行。

EXECUTION_WAIT

任務正在等待執行緒可用。

已排程

任務排定在特定日期和時間執行。

HOLD

任務正在等待釋放，然後再執行。

COMPLETED (已完成)

任務已成功執行。

失敗

任務執行失敗。

UNKNOWN (不明)

狀態未知。

提交任務並排程任務輸入

提交任務和排程任務輸入
是com.netfective.bluage.gapwalk.rt.jobqueue.SubmitJobMessage物件的 JSON 序列化。以下的範例輸入顯示這類 Bean 的所有欄位。

提交任務的範例輸入：

```
{
    "messageQueueName":null,
    "scheduleDate":null,
    "scheduleTime":null,
    "programName":"PTA0044",
    "programParams":
     {"wmind":"B"},
    "localDataAreaValue":"",
    "userName":"USER1",
    "jobName":"PTA0044",
    "jobNumber":9,
    "jobPriority":5,
    "executionDate":"20181231",
    "jobQueue":"queue1",
    "jobOnHold":false
}
```

排程任務的範例輸入：

```
{
```

```
    "scheduleCron": "*/2 * * * * ?",
    "programName":"LOGPGM",
    "programParams": {
        "cl_sbmjob_param_json": "[\"./output/schedule-job-log.txt\", \"Every 2
seconds!\"]"
    },
    "localDataAreaValue":"",
    "userName":"PVO",
    "jobName":"LOGGERJOB",
    "jobPriority":5,
    "jobQueue":"queue1",
    "scheduleMisfirePolicy": 4,
    "startTime": "2003/05/04 07:00:00.000 GMT-06:00",
    "endTime": "2003/05/04 07:00:07.000 GMT-06:00"
}
```

jobNumber

如果任務編號為 0，任務編號將使用任務編號序列中的下一個編號自動產生。該值應設為 0（測試用途除外）。

jobPriority

AS400 中的預設任務優先順序為 5。有效範圍為 0-9，0 是最高優先順序。

jobOnHold

如果任務在保留狀態下提交，則不會立即執行，只會在有人「發行」它時執行。您可以使用 REST API (/release 或 /release-all) 來釋放任務。

scheduleDate 和 scheduleTime

如果這些值不是 null，任務將在指定的日期和時間執行。

日期

可以提供格式 MMddyy 或 ddMMyyyy（輸入的大小將決定使用的格式）

時間

可以提供格式 HHmm 或 HHmmss（輸入的大小將決定使用的格式）

programParams

將做為映射傳遞至程式。

scheduleMisfirePolicy

定義觸發錯誤時所使用的策略。以下為可能值：

1. 釋放第一個不發射並捨棄另一個不發射。
2. 提交第一次不發射的暫停任務，並捨棄其他不發射。
3. 捨棄錯誤。
4. 釋出所有不發射。任務佇列將執行所有任務。

排程任務回應的清單

這是 list-jobs 任務佇列端點的結構。用於提交該任務的提交任務訊息是回應的一部分。這可用於追蹤或測試/重新提交目的。任務完成時，也會填入開始日期和結束日期。

```
[
  {
    "jobName": "PTA0044",
    "userName": "USER1",
    "jobNumber": 9,
    "jobPriority": 5,
    "status": "HOLD",
    "jobDelay": 0,
    "startDate": null,
    "endDate": null,
    "jobQueue": "queue1",
    "message": {
      "messageQueueName": null,
      "scheduleDate": null,
      "scheduleTime": null,
      "programName": "PTA0044",
      "programParams": {"wmind": "B"},
      "localDataAreaValue": "",
      "userName": "USER1",
      "jobName": "PTA0044",
      "jobNumber": 9,
      "jobPriority": 5,
      "executionDate": "20181231",
      "jobQueue": "queue1",
      "jobOnHold": true,
      "scheduleCron": null,
      "save": false,
      "scheduleMisfirePolicy": 4,
```

```
      "omitdates": null
    },
    "executionId": 1,
    "jobScheduledId": 0,
    "jobScheduledAt": null
  },
  {
    "jobName": "PTA0044",
    "userName": "USER1",
    "jobNumber": 9,
    "jobPriority": 5,
    "status": "COMPLETED",
    "jobDelay": 0,
    "startDate": "2022-10-13T22:48:34.025+00:00",
    "endDate": "2022-10-13T22:52:54.475+00:00",
    "jobQueue": "queue1",
    "message": {
      "messageQueueName": null,
      "scheduleDate": null,
      "scheduleTime": null,
      "programName": "PTA0044",
      "programParams": {"wmind": "B"},
      "localDataAreaValue": "",
      "userName": "USER1",
      "jobName": "PTA0044",
      "jobNumber": 9,
      "jobPriority": 5,
      "executionDate": "20181231",
      "jobQueue": "queue1",
      "jobOnHold": true,
      "scheduleCron": "*/20 * * * * ?",
      "save": false,
      "scheduleMisfirePolicy": 4,
      "omitdates": null
    },
    "executionId": 2,
    "jobScheduledId": 0,
    "jobScheduledAt": null
  }
]
```

重複任務回應的清單

這是 /schedule/list 任務佇列端點的結構。


```
[
  {
    "id": 1,
    "status": "ACTIVE",
    "jobNumber": 1,
    "userName": "PVO",
    "msg": {
      "messageQueueName": null,
      "scheduleDate": null,
      "scheduleTime": null,
      "startTime": "2024/03/07 21:12:00.000 UTC",
      "endTime": "2024/03/07 21:13:59.000 UTC",
      "programName": "LOGPGM",
      "programParams": {"cl_sbmjob_param_json": "[\"./output/schedule-job-log.txt\",
 \"Every 20 seconds!\"]"},
      "localDataAreaValue": "",
      "userName": "PVO",
      "jobName": "LOGGERJOB",
      "jobNumber": 1,
      "jobScheduleId": 1,
      "jobPriority": 5,
      "executionDate": null,
      "jobQueue": "queue1",
      "jobOnHold": false,
      "scheduleCron": "*/20 * * * * ?",
      "save": false,
      "scheduleMisfirePolicy": 4,
      "omitdates": null
    },
    "lastUpdatedAt": "2024-03-07T21:11:13.282+00:00",
    "lastUpdatedBy": ""
  }
]
```

設定 AWS Blu Age 執行期（非受管）

本節說明在 AWS 基礎設施上設定 AWS Blu Age Runtime（非受管）的步驟。設定應用程式的 AWS Blu Age 執行期（非受管）之前，請先了解先決條件、區域和儲存貯體，以及設定 CloudWatch 警示以設定和管理執行期環境。

主題

- AWS Blu Age 執行期先決條件
- 加入 AWS 藍卡年齡執行期
- AWS Blu Age Runtime 的基礎設施設定需求（非受管）
- AWS Blu Age 執行期成品
- 在 Amazon EC2 上部署 AWS Blu Age 執行期
- 在 Amazon ECS 和 Amazon EKS 上的容器上部署 AWS Blu Age Runtime
- 測試 PlanetsDemo 應用程式

AWS Blu Age 執行期先決條件

AWS Blu Age Runtime（非受管）提供多種the section called “AWS Blu Age 版本備註”發行版本。如果您有持續的現代化專案，您可能需要執行時間的增量版本，以實作和測試。若要定義您的需求，請聯絡您的 AWS Blu Age 交付管理員。

在開始 AWS 藍卡年齡執行期（非受管）加入程序之前，請執行下列動作：

- 請確定您擁有 AWS 帳戶。
- 請確定您有一個以 AWS Blu Age 重構的現代化應用程式。
- 選擇 AWS 區域，以及支援 AWS Blu Age Runtime（非受管）的其中一個運算選項。
- 選擇您要使用的 AWS Blu Age Runtime 版本。
- 檢閱the section called “基礎設施設定要求”並驗證執行 AWS Blu Age Runtime 所需的其他元件（非受管）。

> **Note**
>
> 如果您想要測試 AWS Blu Age Runtime（非受管）的功能，您可以使用示範應用程式 `Planets Demo`，您可以從 PlanetsDemo-v1.zip 下載該應用程式。

加入 AWS 藍卡年齡執行期

若要開始使用，請建立AWS 支援案例來請求加入以存取 AWS Blu Age 執行期。在您的請求中包含您的 AWS 帳戶 ID、您要使用的 AWS 區域，以及運算選擇，以及 AWS Blu Age Runtime 版本。如果您不確定需要哪個版本，請聯絡您的 AWS Blu Age 交付管理員。如果您已經有 AWS 大型主機現代化重構工具產生的應用程式程式碼來源，請將`gapwalk.version`標籤值記錄在現代化程式碼基礎內的`pom.xml`檔案中。

> **Note**
>
> AWS Blu Age Runtime 提供兩種主要類型：Alpha 發行前版本和發行版本。若要判斷要使用哪個版本，請參閱 the section called “AWS Blu Age 版本控制”，或聯絡您的 AWS Blu Age 交付管理員。

AWS Blu Age Runtime 的區域和儲存貯體 （非受管）

我們會依區域和運算選擇，將 AWS Blu Age Runtime （非受管） 成品存放在不同的 Amazon S3 儲存貯體中。若要存取 AWS 區域 for AWS Blu Age Runtime （非受管） 的 儲存貯體，請使用下表所列的名稱。

AWS 區域	發行儲存貯體	Alpha 發行前儲存貯體
美國東部 (俄亥俄)	aws-bluage-runtime-artifacts-055777665268-us-east-2	aws-bluage-runtime-artifacts-dev-055777665268-us-east-2
美國東部 (維吉尼亞北部)	aws-bluage-runtime-artifacts-139023371234-us-east-1	aws-bluage-runtime-artifacts-dev-139023371234-us-east-1
美國西部 (加利佛尼亞北部)	aws-bluage-runtime-artifacts-788454048782-us-west-1	aws-bluage-runtime-artifacts-dev-788454048782-us-west-1
美國西部 (奧勒岡)	aws-bluage-runtime-artifacts-836771190483-us-west-2	aws-bluage-runtime-artifacts-dev-836771190483-us-west-2
加拿大 (中部)	aws-bluage-runtime-artifacts-637423580979-ca-central-1	aws-bluage-runtime-artifacts-dev-637423580979-ca-central-1

AWS 區域	發行儲存貯體	Alpha 發行前儲存貯體
歐洲 (愛爾蘭)	aws-bluage-runtime-artifacts-925278190477-eu-west-1	aws-bluage-runtime-artifacts-dev-925278190477-eu-west-1
歐洲 (倫敦)	aws-bluage-runtime-artifacts-767397831990-eu-west-1	aws-bluage-runtime-artifacts-dev-767397831990-eu-west-1
Europe (Paris)	aws-bluage-runtime-artifacts-673009995881-eu-west-3	aws-bluage-runtime-artifacts-dev-673009995881-eu-west-3
歐洲 (法蘭克福)	aws-bluage-runtime-artifacts-485196800481-eu-central-1	aws-bluage-runtime-artifacts-dev-485196800481-eu-central-1
歐洲 (斯德哥爾摩)	aws-bluage-runtime-artifacts-654654484534-eu-north-1	aws-bluage-runtime-artifacts-dev-654654484534-eu-north-1
歐洲 (米蘭)	aws-bluage-runtime-artifacts-654654328338-eu-south-1	aws-bluage-runtime-artifacts-dev-654654328338-eu-south-1
歐洲 (西班牙)	aws-bluage-runtime-artifacts-905417994954-eu-south-2	aws-bluage-runtime-artifacts-dev-905417994954-eu-south-2
南美洲 (聖保羅)	aws-bluage-runtime-artifacts-737536804457-sa-east-1	aws-bluage-runtime-artifacts-dev-737536804457-sa-east-1
亞太區域 (東京)	aws-bluage-runtime-artifacts-445578176276-ap-northeast-1	aws-bluage-runtime-artifacts-dev-445578176276-ap-northeast-1
亞太區域 (首爾)	aws-bluage-runtime-artifacts-381492221498-ap-northeast-2	aws-bluage-runtime-artifacts-dev-381492221498-ap-northeast-2

AWS 區域	發行儲存貯體	Alpha 發行前儲存貯體
亞太區域 (大阪)	aws-bluage-runtime-artifacts-905418229615-ap-northeast-3	aws-bluage-runtime-artifacts-dev-905418229615-ap-northeast-3
亞太區域 (新加坡)	aws-bluage-runtime-artifacts-767397774613-ap-southeast-1	aws-bluage-runtime-artifacts-dev-767397774613-ap-southeast-1
亞太區域 (悉尼)	aws-bluage-runtime-artifacts-726160321909-ap-southeast-2	aws-bluage-runtime-artifacts-dev-726160321909-ap-southeast-2
亞太區域 (孟買)	aws-bluage-runtime-artifacts-905418353577-ap-south-1	aws-bluage-runtime-artifacts-dev-905418353577-ap-south-1
非洲 (開普敦)	aws-bluage-runtime-artifacts-992382777663-af-south-1	aws-bluage-runtime-artifacts-dev-992382777663-af-south-1
以色列 (特拉維夫)	aws-bluage-runtime-artifacts-471112516508-il-central-1	aws-bluage-runtime-artifacts-dev-471112516508-il-central-1

使用 AWS CLI 列出儲存貯體的內容

加入後，您可以在終端機中執行下列 AWS CLI 命令，列出儲存貯體的內容。

```
aws s3 ls bucket-name
```

bucket-name 將 取代為上一個資料表中 的 儲存貯 AWS 區域 體名稱。

此命令會傳回對應不同版本的 AWS Blu Age Runtime （非受管） 執行時間的資料夾清單，例如發行儲存貯體的下列資料夾：

```
PRE 3.10.0/
PRE 4.0.0/
```

或者，建置儲存貯體的下列項目：

```
PRE 4.1.0-alpha.8/
PRE 4.1.0-alpha.9/
```

建議您使用最新版本。如果無法這麼做，請使用在應用程式重構階段驗證的執行時間版本。若要列出特定版本的可用架構，請執行下列命令：

```
aws s3 ls s3://bucket-name/version/Framework/
```

bucket-name 將 取代為 的儲存貯體名稱 AWS 區域 ，並將 version取代為您想要的版本。以下是兩個範例。

對於發行儲存貯體：

```
aws s3 ls s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/4.0.0/
Framework/
```

命令會傳回架構清單，例如：

```
2024-04-08 16:11:19  152040176 aws-bluage-runtime-4.0.0.tar.gz
2024-04-08 16:11:50         45 aws-bluage-runtime-4.0.0.tar.gz.checksumSHA256
2024-04-08 16:11:52  176518889 aws-bluage-webapps-4.0.0.tar.gz
2024-04-08 16:12:28         45 aws-bluage-webapps-4.0.0.tar.gz.checksumSHA256
```

對於建置儲存貯體：

```
aws s3 ls s3://aws-bluage-runtime-artifacts-dev-139023371234-us-
east-1/4.1.0-alpha.9/Framework/
```

命令會傳回架構清單，例如：

```
2024-04-09 20:23:34  152304534 aws-bluage-runtime-4.1.0-alpha.9.tar.gz
2024-04-09 20:24:05         45 aws-bluage-runtime-4.1.0-alpha.9.tar.gz.checksumSHA256
2024-04-09 20:24:07  176262381 aws-bluage-webapps-4.1.0-alpha.9.tar.gz
2024-04-09 20:24:42         45 aws-bluage-webapps-4.1.0-alpha.9.tar.gz.checksumSHA256
```

下載架構

您可以下載 架構，例如在現有的 Amazon EC2 執行個體上升級 AWS Blu Age Runtime 版本。

```
aws s3 cp s3://bucket-name/version/Framework/ folder-of-your-choice --
recursive
```

其中：

folder-of-your-choice

您要下載架構的資料夾路徑。

例如：`aws s3 cp s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/4.0.0/Framework/ . --recursive`

此命令會產生以下的輸出：

```
download: s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/4.0.0/
Framework/aws-bluage-runtime-4.0.0.tar.gz.checksumSHA256 to ./aws-bluage-
runtime-4.0.0.tar.gz.checksumSHA256
download: s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/4.0.0/
Framework/aws-bluage-webapps-4.0.0.tar.gz.checksumSHA256 to ./aws-bluage-
webapps-4.0.0.tar.gz.checksumSHA256
download: s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/4.0.0/Framework/aws-
bluage-webapps-4.0.0.tar.gz to ./aws-bluage-webapps-4.0.0.tar.gz
download: s3://aws-bluage-runtime-artifacts-139023371234-us-east-1/4.0.0/Framework/aws-
bluage-runtime-4.0.0.tar.gz to ./aws-bluage-runtime-4.0.0.tar.gz
```

您可以列出架構檔案，如下所示：

`ls -l`

此命令會產生以下的輸出：

```
total 230928
-rw-rw-r-- 1 cloudshell-user cloudshell-user 152040176 Apr  8 16:11 aws-bluage-
runtime-4.0.0.tar.gz
-rw-rw-r-- 1 cloudshell-user cloudshell-user        45 Apr  8 16:11 aws-bluage-
runtime-4.0.0.tar.gz.checksumSHA256
-rw-rw-r-- 1 cloudshell-user cloudshell-user 176518889 Apr  8 16:11 aws-bluage-
webapps-4.0.0.tar.gz
-rw-rw-r-- 1 cloudshell-user cloudshell-user        45 Apr  8 16:12 aws-bluage-
webapps-4.0.0.tar.gz.checksumSHA256
```

> **Note**
>
> 基於安全考量，可能會暫時中斷對成品的存取，而且可能會移除版本。強烈建議您將使用的成品存放在您自己的帳戶中。本機版本應該用於內部架構中的參考。

AWS Blu Age Runtime 的基礎設施設定需求（非受管）

本主題說明執行 AWS Blu Age Runtime（非受管）所需的最低基礎設施組態。下列程序說明如何在您選擇的運算上設定 AWS Blu Age Runtime（非受管），以在 AWS Blu Age Runtime 上部署現代化應用程式。您建立的資源必須位於 Amazon VPC 中，該 VPC 具有專屬於您應用程式網域的子網路。

主題

- 基礎設施需求
- 適用於 AWS Blu Age Runtime 的 Amazon EC2 執行個體類型（在 Amazon EC2 上）
- 在 Amazon EC2 上執行 AWS Blu Age Runtime
- 在 Amazon EC2 上的 Amazon EC2 上執行 AWS Blu Age Runtime
- 在 Amazon EC2 上的 Amazon EKS 上執行 AWS 藍卡存留期
- 在由 管理的 Amazon ECS 上執行 AWS Blu Age Runtime AWS Fargate

基礎設施需求

建立安全群組

如果您計劃在 Amazon EKS 上使用 Amazon EC2 執行個體，請略過此程序，因為 Amazon EKS 叢集建立程序會代表您建立安全群組。在下列程序中使用該安全群組，而不是建立新的安全群組。

1. 在 https://console.aws.amazon.com/vpc/ 開啟 Amazon VPC 主控台。
2. 在左側導覽窗格中，於安全性下，選擇安全群組。
3. 在中央窗格中，選擇建立安全群組。
4. 在安全群組名稱欄位中，輸入 **`M2BluagePrivateLink-SG`**。
5. 在 Inbound rules (傳入規則) 區段中，選擇 Add rule (新增規則)。
6. 針對類型，選擇 HTTPS。
7. 針對來源輸入您的 VPC CIDR。

8. 在傳出規則區段中，選擇新增規則。
9. 針對類型，選擇 HTTPS。
10. 針對 Destination (目標)，輸入 **0.0.0.0/0**。
11. 選擇建立安全群組。

建立 Amazon VPC 端點

1. 在 https://console.aws.amazon.com/vpc/ 開啟 Amazon VPC 主控台。
2. 在左側導覽窗格的虛擬私有雲端下，選擇端點。
3. 在中央窗格中，選擇建立端點。
4. 在服務區段**SQS**中，在搜尋欄位中輸入 ，然後選取對應至您區域的 Amazon SQS 服務。
5. 在 VPC 區段中，選取您在上一個步驟中建立的 Amazon VPC。
6. 在子網路區段中，選取您為應用程式網域建立的子網路。
7. 在安全群組區段中，從上一個程序中選取安全群組。
8. 選擇建立端點。

建立 IAM 政策

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側導覽窗格的存取管理下，選擇政策。
3. 在中央窗格中，選擇建立政策。
4. 在政策編輯器區段中，選擇 JSON。
5. 使用下列 JSON 取代您在編輯器中看到的所有 JSON。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "sqs:GetQueueUrl",
                "sqs:ReceiveMessage",
                "sqs:SendMessage"
```

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

> **Note**
>
> 如果您需要進一步的詳細資訊來自訂政策，請聯絡您的 AWS Blu Age 交付經理或客戶經理。

6. 選擇下一步。
7. 輸入政策的名稱，然後選擇建立政策。

建立 IAM 角色

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側導覽窗格的存取管理下，選擇角色。
3. 在中央窗格中，選擇建立角色。
4. 在使用案例區段中，根據您的運算選擇，選擇下列其中一項：
   - EC2（適用於 Amazon EC2 和 Amazon EC2 上的 Amazon EKS)
   - Elastic Container Service 和 Elastic Container Service 的 EC2 角色（適用於 Amazon EC2 上的 Amazon EC2)
   - Elastic Container Service，然後 Elastic Container Service Task（適用於 Fargate 管理的 Amazon ECS)
5. 選擇下一步。
6. 在搜尋方塊中，輸入您先前建立的政策名稱。
7. 選取政策左側的核取方塊。

> **Note**
>
> 如果您無法新增政策，請完成建立角色，然後更新角色以新增政策。

8. 選擇下一步。
9. 輸入角色名稱，然後選擇建立角色。

適用於 AWS Blu Age Runtime 的 Amazon EC2 執行個體類型 （在 Amazon EC2 上）

以下是您在建立 Amazon EC2 執行個體或定義 Amazon EKS 工作者節點時，可用於 AWS Blu Age Runtime (Amazon EC2) 的 Amazon EC2 執行個體類型清單。

檢查您感興趣的執行個體是否在您計劃部署的所需區域中可用。

```
t3.small
t3.medium
t3.large
t3.xlarge
t3.2xlarge
t2.small
t2.medium
t2.large
t2.xlarge
t2.2xlarge
r7a.medium
r7a.large
r7a.xlarge
r7a.2xlarge
r7a.4xlarge
r7a.8xlarge
r7a.12xlarge
r7a.16xlarge
r7a.24xlarge
r7a.32xlarge
r7a.48xlarge
r7a.metal-48xl
r7i.large
r7i.xlarge
r7i.2xlarge
r7i.4xlarge
r7i.8xlarge
r7i.12xlarge
r7i.16xlarge
r7i.24xlarge
r7i.48xlarge
r7i.metal-24xl
r7i.metal-48xl
r6i.xlarge
r6i.large
r6i.4xlarge
```

```
r6i.2xlarge
r5b.xlarge
r5b.large
r5b.2xlarge
r3.xlarge
m6i.xlarge
m6i.large
m6i.8xlarge
m6i.4xlarge
m6i.2xlarge
m6i.16xlarge
m5zn.xlarge
m5zn.large
m5zn.3xlarge
m5zn.2xlarge
m5.xlarge
m5.large
m5.8xlarge
m5.4xlarge
m5.2xlarge
m5.16xlarge
m5.12xlarge
c6i.xlarge
c6i.large
c6i.8xlarge
c6i.4xlarge
c6i.2xlarge
c6i.16xlarge
c5.xlarge
c5.large
c5.9xlarge
c5.4xlarge
c5.2xlarge
c5.18xlarge
c5.12xlarge
```

在 Amazon EC2 上執行 AWS Blu Age Runtime

若要建立 Amazon EC2 執行個體，請使用下列步驟。

建立 Amazon EC2 執行個體

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。

2. 選擇啟動執行個體。
3. 針對執行個體類型，選擇 中列出的其中一種類型the section called “適用於 AWS Blu Age Runtime 的 Amazon EC2 執行個體類型 （ 在 Amazon EC2 上 ）”。
4. 在金鑰對區段中，選擇現有的金鑰對或建立新的金鑰對。
5. 在網路設定區段中，選擇選取現有的安全群組。
6. 針對一般安全群組，選擇 M2BluagePrivateLink-SG。
7. 展開進階詳細資訊區段。
8. 針對 IAM 執行個體描述檔，選擇您先前建立的 IAM 角色。
9. 選擇啟動執行個體。

在 Amazon EC2 執行個體上安裝應用程式

1. 當 Amazon EC2 執行個體的狀態變更為執行中時，請連線至執行個體。
2. 在執行個體上安裝下列軟體元件：
   - Java 執行期環境 (JRE) 17.
   - Apache Tomcat 10。
   - AWS Blu Age 執行期 （ 在 Amazon EC2 上 ）。在 Apache Tomcat 安裝資料夾的根目錄安裝 AWS Blu Age 執行時間 （ 有些檔案會新增，有些檔案則會覆寫 ）。

   若要安裝與 AWS Blu Age Runtime 封存一起交付的其他 Web 應用程式，請設定 Apache Tomcat 伺服器的次要執行個體，然後解壓縮該位置的 Web 應用程式封存。如需詳細說明，請參閱 the section called “AWS Blu Age 執行期成品”。

在 Amazon EC2 上的 Amazon EC2 上執行 AWS Blu Age Runtime

1. 建立 Amazon ECS 叢集，並將 Amazon EC2 執行個體做為基礎基礎設施。請參閱《Amazon Elastic Container Service 開發人員指南》中的 Windows on Amazon EC2 入門。
2. 指定您在先前步驟中建立的 IAM 角色。
3. 選擇 中列出的其中一個執行個體類型the section called “適用於 AWS Blu Age Runtime 的 Amazon EC2 執行個體類型 （ 在 Amazon EC2 上 ）”。
4. 在 Amazon EC2 執行個體的網路設定中，選擇您在先前步驟中建立的安全群組。

在 Amazon EC2 上的 Amazon EKS 上執行 AWS 藍卡存留期

1. 建立 Amazon EKS 叢集。請參閱《Amazon EKS 使用者指南》中的建立 Amazon EKS 叢集。
2. 如前所述，系統會代表您建立安全群組。您可以在建立 Amazon VPC 端點時使用該安全群組。
3. 建立節點群組。指定您在先前步驟中建立的 IAM 角色。
4. 選擇 中列出的其中一個執行個體類型the section called “適用於 AWS Blu Age Runtime 的 Amazon EC2 執行個體類型 （ 在 Amazon EC2 上 ）”。
5. Amazon EKS 會自動將安全群組指派給產生的 Amazon EC2 執行個體。

在由 管理的 Amazon ECS 上執行 AWS Blu Age Runtime AWS Fargate

使用 AWS Fargate （ 無伺服器 ） 作為基礎基礎設施來建立 Amazon ECS 叢集。請參閱《Amazon Elastic Container Service 開發人員指南》中的 Fargate 入門。

AWS Blu Age 執行期成品

AWS Blu Age Runtime 成品是部署和執行現代化應用程式的元件。本文件概述了可用的不同成品類型、儲存位置，以及如何存取它們。

AWS Blu Age 執行期 （ 非受管 ） 成品

成品存取和儲存

非受管部署的 AWS Blu Age Runtime 成品會存放在區域特定的 S3 儲存貯體中。每個版本都有自己的專用資料夾，讓版本管理和存取更為簡單。

儲存貯體有兩種類型：

發行儲存貯體

發行儲存貯體包含最近部署版本的目錄，並遵循命名慣例：`aws-bluage-runtime-artifacts-<accountId>-<region>`。

發行前儲存貯體

發行前儲存貯體包含對應至最新短期增量發行前版本的 alpha 版本的目錄，並遵循命名慣例：`convention: aws-bluage-runtime-artifacts-dev-<accountId>-<region>`。

對生產和發行前儲存貯體的存取會獨立授予。如需如何請求存取權的詳細資訊，以及 S3 儲存貯體組織的詳細資訊，請參閱 the section called “加入 AWS 藍卡年齡執行期 ”。

成品內容

在發行和發行前儲存貯體中，您會找到：

aws-bluage-runtime-x.y.z.tar.gz

此封存檔，其中 x.y.z 代表版本編號 (`major.minor.patch`根據語意版本控制，請參閱 the section called “AWS Blu Age 版本控制”)，並包含執行 AWS Blu Age 應用程式所需的核心 AWS Blu Age Runtime 元件，包括：

- Gapwalk： AWS Blu Age Runtime 的重要元件，旨在彌補傳統應用程式和現代雲端原生環境之間的差距。它可做為相容性層，讓透過 AWS Blu Age 現代化應用程式在現代平台上有效執行。
- bluage.bin： AWS Blu Age Runtime 的核心二進位檔案。此檔案對於執行時間的操作至關重要。
- Blu AWS Age Runtime 操作的所有必要程式庫和支援檔案。

aws-bluage-webapps-x.y.z.tar.gz

此封存，其中 x.y.z 遵循與上述相同的版本控制機制，包括管理和控制 AWS Blu Age 部署所需的 Web 應用程式和程式庫：

- BAC (Blusam 主控台） WAR 檔案，用於監控 Blusam 資料庫。
- JAC (JICS 主控台） WAR 檔案，用於監控 JICS 資料庫。
- 必要的支援程式庫。

其他檔案

- 檢查總和檔案可讓您依照命名慣例驗證兩個 Blu Age 封存的完整性：
  - 對於執行時間： `aws-bluage-runtime-x.y.z.tar.gz.checksumSHA256`
  - 針對 Webapp： `aws-bluage-webapps-x.y.z.tar.gz.checksumSHA256`
- CVE 報告檔案 （僅適用於發行版本） 會列出此版本目前的 CVEs並遵循命名慣例：
  - 對於執行時間： `Bluage-Runtime-x.y.z-CVEs.txt`
  - 對於 Webapp： `Bluage-Webapps-x.y.z-CVEs.txt`

如需如何解決安全漏洞的詳細資訊，請參閱 AWS 大型主機現代化重構搭配 AWS Blu Age 版本概觀。

> **Note**
>
> 雖然我們努力在沒有 CVEs 的情況下發佈產品，但之後可能會出現新的 CVEs。CVE 報告檔案會定期更新，以反映最新狀態。

Developer AWS Blu Age 執行期成品

成品存取和儲存

AWS Blu Age 開發人員執行期成品存放在專用 S3 儲存貯體中。此執行時間同時包含發行版本和 Alpha 發行前版本。這些成品的存取是透過 AWS Blu Age 工具箱請求來管理。處理並核准您的請求後，系統會授予您從請求中 AWS 帳戶 指定的 存取適當儲存貯體的權限。

開發人員執行期儲存貯體

開發人員執行時間的主要儲存貯體為： `s3://toolbox-dev-runtime`

如需請求存取和了解儲存貯體結構的詳細資訊，請參閱開發和特殊 AWS Blu Age Runtimes 文件。

成品內容

開發人員執行期成品通常包括：

gapwalk-x.y.z-dev.tar.gz

此封存包含 Gapwalk 元件的開發版本，這是 AWS Blu Age 執行期的重要部分。它旨在將舊版應用程式與現代雲端原生環境橋接。

gapwalk-runtime-x.y.z-javadoc.zip

此 zip 檔案包含 Gapwalk 執行時間的 JavaDoc 文件。JavaDoc 提供詳細的 API 文件，對於整合或延長 Gapwalk 執行時間的開發人員特別有用。

gapwalk-webapps-x.y.z-javadoc.zip

與執行期 JavaDoc 類似，此 zip 檔案包含專門用於 Gapwalk Web 應用程式的 JavaDoc 文件。本文件對於使用 或自訂 Gapwalk 系統 Web 型元件的開發人員至關重要。

在 Amazon EC2 上部署 AWS Blu Age 執行期

您可以了解如何在 Amazon EC2 上設定 AWS Blu Age Runtime （非受管）、如何更新執行時間版本、如何使用 Amazon CloudWatch 警示監控您的部署，以及如何新增授權相依性與本節中的主題。

這些指示適用於建立 Amazon EC2 執行個體，以及使用 Amazon EC2 上的 Amazon EC2或 Amazon EC2 上的 Amazon EC2 EKS 時。

主題

- 在 Amazon EC2 上設定 AWS Blu Age Runtime （非受管）
- 升級 Amazon EC2 上的 AWS Blu Age 執行期
- 設定 AWS Blu Age Runtime （在 Amazon EC2 上） Amazon CloudWatch 警示
- 在 Amazon EC2 上的 AWS Blu Age Runtime 中設定授權相依性

在 Amazon EC2 上設定 AWS Blu Age Runtime （非受管）

本主題說明如何在 Amazon EC2 上使用 AWS Blu Age Runtime （非受管） 設定和部署 PlanetsDemo 範例應用程式。

主題

- 先決條件
- 設定
- 測試部署的應用程式

先決條件

開始之前，請務必完成下列先決條件。

- AWS CLI 請依照設定 AWS CLI 中的步驟來設定 。
- 完成 the section called “AWS Blu Age 執行期先決條件”和 the section called “加入 AWS 藍卡年齡執行期 ”。
- 使用其中一個支援的執行個體類型建立 Amazon EC2 執行個體。如需詳細資訊，請參閱 Amazon EC2 Linux 執行個體入門。
- 請確定您可以成功連線至 Amazon EC2 執行個體，例如使用 SSM。

> **Note**
>
> 在本指南中，Tomcat 安裝路徑假設為 `/m2-anywhere/tomcat-gapwalk/velocity`。遵循下列指示時，請務必使用此路徑，或將下列指示調整為您選擇的路徑。

- 下載並擷取 AWS Blu Age Runtime （在 Amazon EC2 上）。將速度目錄的內容複製到 /m2-anywhere/tomcat-gapwalk/velocity。請務必將bluage.bin檔案精確地放在 Apache Tomcat 文件 CATALINA_HOME 和 CATALINA_BASE 下描述的 CATALINA_HOME 環境變數指定的位置。如需如何擷取 AWS Blu Age Runtime 成品的說明，包括儲存、存取和內容的相關資訊，請參閱 the section called “AWS Blu Age 執行期成品”。
- 下載 PlanetsDemo 應用程式封存。
- 解壓縮封存檔，並將應用程式上傳到您選擇的 Amazon S3 儲存貯體。
- 為 JICS 建立 Amazon Aurora PostgreSQL 資料庫。 AWS Blu Age Runtime 會在第一次啟動時自動執行PlanetsDemo-v1/jics/sql/initJics.sql指令碼。如需如何建立 Amazon Aurora PostgreSQL 資料庫的資訊，請參閱建立並連線至 Aurora PostgreSQL 資料庫叢集。

設定

若要設定 PlanetsDemo 範例應用程式，請完成下列步驟。

1. 連線至 Amazon EC2 執行個體，並前往 Apache Tomcat 10 安裝conf資料夾下的 資料夾。開啟 catalina.properties 檔案進行編輯，並以下列行取代開頭common.loader為 的行。

```
common.loader="${catalina.base}/lib","${catalina.base}/lib/
*.jar","${catalina.home}/lib","${catalina.home}/lib/*.jar","${catalina.home}/
shared","${catalina.home}/shared/*.jar","${catalina.home}/extra","${catalina.home}/
extra/*.jar"
```

2. 導覽至 /m2-anywhere/tomcat-gapwalk/velocity /webapps/webapps 資料夾。
3. 使用下列命令，從 Amazon S3 儲存貯體複製PlanetsDemo-v1/webapps/資料夾提供的 PlanetsDemo 二進位檔。

```
aws s3 cp s3://path-to-demo-app-webapps/ . --recursive
```

> **Note**
>
> path-to-demo-app-webapps 將 取代為您先前解壓縮 PlanetsDemo 封存的儲存貯體的正確 Amazon S3 URI。

4. 將PlanetsDemo-v1/config/資料夾的內容複製到 /m2-anywhere/tomcat-gapwalk/velocity /config/。

5. 在 application-main.yml 檔案的下列程式碼片段中提供您建立的資料庫連線資訊，做為先決條件的一部分。如需詳細資訊，請參閱建立並連線至 Aurora PostgreSQL 資料庫叢集。

```
datasource:
   jicsDs:
     driver-class-name :
     url:
     username:
     password:
     type :
```

6. 啟動 Apache Tomcat 伺服器並驗證日誌。

```
/m2-anywhere/tomcat-gapwalk/velocity/startup.sh

tail -f /m2-anywhere/tomcat-gapwalk/velocity/logs/catalina.log
```

如果您找到以 C 開頭的錯誤代碼，後面接著 CXXXX 等數字，請注意錯誤訊息。例如，錯誤代碼 C5102 是表示基礎設施組態不正確的常見錯誤。

測試部署的應用程式

如需如何測試 PlanetsDemo 應用程式的範例，請參閱 the section called “測試 PlanetsDemo 應用程式”。

升級 Amazon EC2 上的 AWS Blu Age 執行期

本指南說明如何升級 Amazon EC2 上的 AWS Blu Age 執行期。

主題

- 先決條件
- 升級 Amazon EC2 執行個體中的 AWS Blu Age 執行期
- 升級容器中的 AWS Blu Age 執行時間

先決條件

開始之前，請確定您符合下列先決條件。

- 若要檢查您的版本是否有特定指示，請參閱 the section called “升級 AWS 藍髮年齡”。

- 完成 the section called “AWS Blu Age 執行期先決條件”和 the section called “加入 AWS 藍卡年齡執行期 ”。
- 請確定您的 Amazon EC2 執行個體包含最新的 AWS Blu Age 執行期。如需詳細資訊，請參閱 Amazon EC2 Linux 執行個體入門。
- 請確定您可以成功連線至 Amazon EC2 執行個體，例如使用 SSM。
- 下載您要升級的 AWS Blu Age Runtime 版本。如需詳細資訊，請參閱the section called “ 設定 AWS Blu Age 執行期 （ 非受管 ）”架構包含兩個二進位檔案： `aws-bluage-runtime-x.x.x.x.tar.gz`和 `aws-bluage-webapps-x.x.x.x.tar.gz`。

升級 Amazon EC2 執行個體中的 AWS Blu Age 執行期

完成下列步驟以升級 AWS Blu Age 執行期。

1. 連線至您的 Amazon EC2 執行個體，並執行下列命令將使用者變更為 su。

   ```
   sudo su
   ```

   您需要超級使用者權限，才能在本教學課程中執行命令。
2. 建立兩個資料夾，每個二進位檔案各一個。
3. 使用與二進位檔案相同的名稱命名每個資料夾。
4. 將每個二進位檔案複製到對應的資料夾。

   > **Warning**
   >
   > 解壓縮每個二進位檔案會產生具有相同名稱的資料夾。因此，如果您逐一在相同位置擷取兩個二進位檔案，則會覆寫內容。

5. 若要擷取二進位檔，請使用下列命令。在每個資料夾中執行命令。

   ```
   tar xvf aws-bluage-runtime-x.x.x.x.tar.gz
   tar xvf aws-bluage-webapps-x.x.x.x.tar.gz
   ```

6. 使用以下命令停止 Apache Tomcat 服務。

   ```
   systemctl stop tomcat.service
   systemctl stop tomcat-webapps.service
   ```

7. 將 的內容取代<your-tomcat-path>/shared/為 的內容aws-bluage-runtime-x.x.x.x/velocity/shared/。
8. 使用 aws-bluage-runtime-x.x.x.x/velocity/webapps/gapwalk-application.war 取代 <your-tomcat-path>/webapps/gapwalk-application.war。
9. 將 中的 war 檔案取代為 中的相同檔案jac.war，<your-tomcat-path>/webapps/也就是bac.war和 aws-bluage-webapps-x.x.x.x/velocity/webapps/。
10. 執行下列命令來啟動 Apache Tomcat 服務。

```
systemctl start tomcat.service
systemctl start tomcat-webapps.service
```

11. 檢查日誌。

若要檢查已部署應用程式的狀態，請執行下列命令。

```
curl http://localhost:8080/gapwalk-application/
```

應該會顯示以下訊息。

```
Jics application is running
```

```
curl http://localhost:8181/jac/api/services/rest/jicsservice/
```

應該會顯示以下訊息。

```
Jics application is running
```

```
curl http://localhost:8181/bac/api/services/rest/bluesamserver/serverIsUp
```

回應應該是空的。

成功升級 AWS Blu Age 執行時間。

升級容器中的 AWS Blu Age 執行時間

完成下列步驟以升級 AWS Blu Age 執行期。

1. 使用所需的 AWS Blu Age Runtime 版本重建 Docker 映像。如需說明，請參閱the section called “在 Amazon EC2 上設定 AWS Blu Age Runtime （非受管）”。
2. 將 Docker 映像推送至 Amazon ECR 儲存庫。
3. 停止並重新啟動您的 Amazon ECS 或 Amazon EKS 服務。
4. 檢查日誌。

成功升級 AWS Blu Age Runtime。

設定 AWS Blu Age Runtime （在 Amazon EC2 上） Amazon CloudWatch 警示

您可以設定 CloudWatch 來接收您的應用程式日誌，並新增警示來警告您可能發生的錯誤。這可讓您在部署的應用程式遇到例外狀況時，擁有更多可見的通知。下列各節可協助您了解和了解 CloudWatch 記錄和警示設定的組態。

部署 CloudWatch 記錄

根據預設， AWS Blu Age Runtime 包含名為 的記錄檔案`logback-cloudwatch.yml`。此檔案會在 `application-main.yml` 檔案中參考，但此參考會註解。

```
# logging:
#  config: classpath:logback-cloudwatch.xml
```

這兩個檔案都在組態資料夾中，並且取消註解上述行，即可啟用該功能。您可以設定 CloudWatch 記錄，如以下各節所述。

CloudWatch 記錄的組態

預設`logback-cloudwatch.xml`檔案具有下列內容。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration>
<configuration>

    <appender name="console" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%date{yyyy-MM-dd HH:mm:ss.SSS,UTC}  %level --- [%thread{15}]
 %logger{40} : %msg%n%xThrowable</pattern>
        </encoder>
    </appender>
```

```
    <appender name="cloudwatch"
 class="com.netfective.bluage.runtime.cloudwatchlogger.CloudWatchAppender">
        <logGroup>BluAgeRuntimeOnEC2-Logs</logGroup>
        <logStream>%date{yyyy-MM-dd,UTC}.%instanceId.%uuid</logStream>
        <layout>
            <pattern>%date{yyyy-MM-dd HH:mm:ss.SSS,UTC}  %level --- [%thread{15}]
 %logger{40} : %msg%n%xThrowable</pattern>
        </layout>
        <appender-ref ref="console" />
    </appender>

    <root level="INFO">
        <appender-ref ref="cloudwatch" />
    </root>
</configuration>
```

<appender name="cloudwatch"/> 元素以外的所有項目都是標準日誌傳回組態。此檔案中有兩個附加元件：主控台附加元件將日誌傳送至主控台，以及 CloudWatch 附加元件將日誌傳送至 CloudWatch。

`root` 元素中的 `level` 屬性會指定整個應用程式的記錄層級。

標籤 <appender name="cloudwatch"/> 內的必要值為：

- <logGroup/>：在 CloudWatch 中設定日誌群組的名稱。如果未指定值，則預設為 `BluAgeRuntimeOnEC2-Logs`。如果日誌群組不存在，則會自動建立日誌群組。此行為可以透過組態進行變更，如下所述。
- <logStream/>：設定 CloudWatch 中 logStream （ 日誌群組內部 ） 的名稱。

選用值：

- <region/>：覆寫日誌串流將寫入的區域。根據預設，日誌會前往與 EC2 執行個體相同的區域。
- <layout/>：日誌訊息將使用的模式。
- <maxbatchsize/>：每個操作要傳送至 CloudWatch 的日誌訊息數量上限。
- <maxbatchtimemillis/>：允許寫入 CloudWatch 日誌的時間，以毫秒為單位。
- <maxqueuewaittimemillis/>：嘗試在內部日誌佇列中插入請求的時間，以毫秒為單位。
- <internalqueuesize/>：內部佇列的大小上限。
- <createlogdests/>：如果日誌群組不存在，請建立日誌群組和日誌串流。
- <initialwaittimemillis/>：您希望執行緒在啟動時休眠的時間量。此初始等待允許日誌的初始累積。

- <maxeventmessagesize/>：日誌事件的大小上限。不會傳送超過此大小的日誌。
- <truncateeventmessages/>：截斷過長的訊息。
- <printrejectedevents/>：啟用緊急附加元件。

CloudWatch 設定

為了讓上述組態正確將日誌推送至 CloudWatch，請更新您的 Amazon EC2 IAM 執行個體描述檔角色，以授予 `BluAgeRuntimeOnEC2-Logs` 日誌群組及其日誌串流的額外許可：

- `logs:CreateLogStream`
- `logs:DescribeLogStreams`
- `logs:CreateLogGroup`
- `logs:PutLogEvents`
- `logs:DescribeLogGroups`

警示設定

由於 CloudWatch 日誌，您可以根據您的應用程式和需求，設定不同的指標和警示。具體而言，您可以為用量提醒設定主動警示，以便在錯誤可能導致應用程式處於寬限期時收到警告（最後防止其完全運作）。若要達成此目的，您可以在 日誌中新增有關「錯誤 C5001」字串的指標，這會反白連線至 AWS Blu Age 控制系統的錯誤。然後，您可以定義對此指標做出反應的警示。

在 Amazon EC2 上的 AWS Blu Age Runtime 中設定授權相依性

本指南說明如何設定額外的授權相依性，讓您可以在 Amazon EC2 上使用 AWS Blu Age Runtime。

主題

- 先決條件
- 概觀
- 設定 JAC 和 BAC Webapp 的相依性

先決條件

開始之前，請務必完成下列先決條件。

- 完成 the section called “AWS Blu Age 執行期先決條件”和 the section called “加入 AWS 藍卡年齡執行期 ”。

- 請確定您的 Amazon EC2 執行個體包含最新的 AWS Blu Age 執行期（在 Amazon EC2 上）。如需詳細資訊，請參閱 Amazon EC2 Linux 執行個體入門。
- 請確定您可以成功連線至 Amazon EC2 執行個體，例如使用 SSM。
- 從來源取得下列相依性。

Oracle 資料庫

提供 Oracle 資料庫驅動程式。我們測試了 ojdbc11-23.3.0.23.09.jar 版的 AWS Blu Age Runtime（在 Amazon EC2 上）功能，但較新的版本可能相容。

IBM MQ 連線

提供 IBM MQ 用戶端。我們測試了使用 com.ibm.mq.jakarta.client-9.3.4.1.jar 版的 AWS Blu Age Runtime（在 Amazon EC2 上）功能，但較新的版本可能相容。

在此相依性版本中， 也提供下列可轉移相依性：

- bcprov-jdk15to18-1.76.jar
- bcpkix-jdk15to18-1.76.jar
- bcutil-jdk15to18-1.76.jar

DDS 印表機檔案

提供 Jasper 報告程式庫 (https://community.jaspersoft.com/download-jaspersoft/community-edition)。我們已使用 jasperreports-6.16.0.jar 測試 AWS 了 Blu Age Runtime（在 Amazon EC2 上）功能，但較新的版本可能相容。

在此相依性版本中， 也提供下列可轉移相依性：

- castor-core-1.4.1.jar
- castor-xml-1.4.1.jar
- commons-digester-2.1.jar
- ecj-3.21.0.jar
- itext-2.1.7.js8.jar
- javax.inject-1.jar
- jcommon-1.0.23.jar

- jfreechart-1.0.19.jar
- commons-beanutils-1.9.4.jar
- commons-collections-3.2.2.jar

概觀

若要安裝相依性，請完成下列步驟。

1. 連線至您的 Amazon EC2 執行個體，並執行下列命令將使用者變更為 su。

   ```
   sudo su
   ```

   您需要超級使用者權限，才能在本教學課程中執行命令。
2. 導覽至 <your-tomcat-path>/extra/ 資料夾。

   ```
   cd <your-tomcat-path>/extra/
   ```
3. 視需要在此資料夾複製上述任何相依性。
4. 執行下列命令來停止和啟動 tomcat.service。

   ```
   systemctl stop tomcat.service
   ```

   ```
   systemctl start tomcat.service
   ```
5. 檢查服務的狀態，以確保其正在執行。

   ```
   systemctl status tomcat.service
   ```
6. 驗證日誌。

設定 JAC 和 BAC Webapp 的相依性

1. 如果您的 JICS 資料庫託管在 Oracle 上，則需要在 中提供 Oracle 資料庫驅動程式<your-tomcat-path>/extra。
2. 如果資料夾尚未存在，請建立資料夾。
3. 停止並重新啟動您的 Apache Tomcat 伺服器。
4. 驗證日誌。

在 Amazon ECS 和 Amazon EKS 上的容器上部署 AWS Blu Age Runtime

您可以使用本節中的主題，了解如何在容器上設定 AWS Blu Age Runtime 以在 Amazon ECS （由 Amazon EC2 或 管理 AWS Fargate) 和 Amazon EC2 管理的 Amazon EKS 上部署、如何更新執行時間版本、如何使用 Amazon CloudWatch 警示監控部署，以及如何新增授權相依性。

> **Note**
>
> 這與 管理的 Amazon EKS 不相容 AWS Fargate。

主題

- 在容器上設定 AWS Blu Age 執行期
- 升級 容器上的 AWS Blu Age 執行期
- 在容器上設定 AWS 藍時執行期的 Amazon CloudWatch 警示
- 在容器的 AWS 藍時空執行期中設定授權相依性

在容器上設定 AWS Blu Age 執行期

本主題說明如何在 Docker 容器上使用 AWS Blu Age Runtime 來設定和部署 PlanetsDemo 範例應用程式。

AWS 容器上的 Blu Age Runtime 適用於由 Amazon EC2 管理的 Amazon ECS AWS Fargate、由 管理的 Amazon ECS 和由 Amazon EC2 管理的 Amazon EKS。它與 管理的 Amazon EKS 不相容 AWS Fargate。

主題

- 先決條件
- 設定
- 測試部署的應用程式

先決條件

開始之前，請務必完成下列先決條件。

- AWS CLI 請依照設定 AWS CLI 中的步驟來設定 。

- 完成 the section called “AWS Blu Age 執行期先決條件”和 the section called “加入 AWS 藍卡年齡執行期 ”。
- 下載 AWS Blu Age Runtime 二進位檔。如需說明，請參閱the section called “加入 AWS 藍卡年齡執行期 ”。
- 下載 Apache Tomcat 10 二進位檔。
- 下載 PlanetsDemo 應用程式封存。
- 為 JICS 建立 Amazon Aurora PostgreSQL 資料庫，並在資料庫上執行`PlanetsDemo-v1/jics/sql/initJics.sql`查詢。如需有關如何建立 Amazon Aurora PostgreSQL 資料庫的資訊，請參閱建立並連線至 Aurora PostgreSQL 資料庫叢集。

設定

若要設定 PlanetsDemo 範例應用程式，請完成下列步驟。

1. 下載 Apache Tomcat 二進位檔後，擷取內容，然後移至 `conf` 資料夾。開啟 `catalina.properties` 檔案進行編輯，並以下列行取代開頭`common.loader`為 的行。

```
common.loader="${catalina.base}/lib","${catalina.base}/lib/
*.jar","${catalina.home}/lib","${catalina.home}/lib/*.jar","${catalina.home}/
shared","${catalina.home}/shared/*.jar","${catalina.home}/extra","${catalina.home}/
extra/*.jar"
```

2. 使用 tar 命令建置 `tar.gz` 封存，以壓縮 Apache Tomcat 資料夾。
3. 準備 Dockerfile，根據提供的執行時間二進位檔和 Apache Tomcat 伺服器二進位檔來建置自訂映像。請參閱下列範例 Dockerfile。目標是安裝 Apache Tomcat 10，接著安裝擷取自 Apache Tomcat 10 安裝目錄根目錄的 AWS Blu Age Runtime （適用於 管理的 Amazon ECS AWS Fargate)，然後安裝名為 PlanetsDemo 的範例現代化應用程式。

> **Note**
>
> 在這個範例 Dockerfile 中使用的 install-gapwalk.sh：//https：/install-app.sh：//https：//https：//https：//https：//https：//https：//https：//https：//https：//https：//https：//https：//http：///http

```
FROM --platform=linux/x86_64 amazonlinux:2
```

```
RUN mkdir -p /workdir/apps
WORKDIR /workdir
COPY install-gapwalk.sh .
COPY install-app.sh .
RUN chmod +x install-gapwalk.sh
RUN chmod +x install-app.sh

# Install Java and AWS CLI v2-y
RUN yum install sudo java-17-amazon-corretto unzip tar -y
RUN sudo yum remove awscli -y
RUN curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
 "awscliv2.zip"
RUN sudo unzip awscliv2.zip
RUN sudo ./aws/install

# Installation dir
RUN mkdir -p /usr/local/velocity/installation/gapwalk
# Copy PlanetsDemo archive to a dedicated apps dir
COPY PlanetsDemo-v1.zip /workdir/apps/

# Copy resources (tomcat, blu age runtime) to installation dir
COPY tomcat.tar.gz /usr/local/velocity/installation/tomcat.tar.gz
COPY aws-bluage-runtime-4.x.x.tar.gz /usr/local/velocity/installation/gapwalk/
gapwalk.tar.gz

# run relevant installation scripts
RUN ./install-gapwalk.sh
RUN ./install-app.sh

EXPOSE 8080
EXPOSE 8081
# ...

WORKDIR /bluage/tomcat.gapwalk/velocity
# Run Command to start Tomcat server
CMD ["sh", "-c", "sudo bin/catalina.sh run"]
```

以下是 的內容 `install-gapwalk.sh`.

```


# Vars
TEMP_DIR=/bluage-on-fargate/tomcat.gapwalk/temp
```

```
# Install
echo "Installing Gapwalk and Tomcat"
sudo rm -rf /bluage-on-fargate
mkdir -p ${TEMP_DIR}
# Copy Blu Age runtime and tomcat archives to temporary extraction dir
sudo cp /usr/local/velocity/installation/gapwalk/gapwalk.tar.gz ${TEMP_DIR}
sudo cp /usr/local/velocity/installation/tomcat.tar.gz ${TEMP_DIR}
# Create velocity dir
mkdir -p /bluage/tomcat.gapwalk/velocity
# Extract tomcat files
tar -xvf ${TEMP_DIR}/tomcat.tar.gz -C ${TEMP_DIR}
# Copy all tomcat files to velocity dir
cp -fr ${TEMP_DIR}/apache-tomcat-10.x.x/* /bluage/tomcat.gapwalk/velocity
# Remove default webapps of Tomcat
rm -f /bluage-on-fargate/tomcat.gapwalk/velocity/webapps/*
# Extract Blu Age runtime at velocity dir
tar -xvf ${TEMP_DIR}/gapwalk.tar.gz -C /bluage/tomcat.gapwalk
# Remove temporary extraction dir
sudo rm -rf ${TEMP_DIR}
```

以下是 的內容`install-app.sh`。

```
#!/bin/sh

APP_DIR=/workdir/apps
TOMCAT_GAPWALK_DIR=/bluage-on-fargate/tomcat.gapwalk

unzip ${APP_DIR}/PlanetsDemo-v1.zip -d ${APP_DIR}
cp -r ${APP_DIR}/webapps/* ${TOMCAT_GAPWALK_DIR}/velocity/webapps/
cp -r ${APP_DIR}/config/* ${TOMCAT_GAPWALK_DIR}/velocity/config/
```

4. 在位於 `{TOMCAT_GAPWALK_DIR}/config` 資料夾的 `application-main.yml` 檔案的下列程式碼片段中提供您建立的資料庫連線資訊，做為先決條件的一部分。如需詳細資訊，請參閱建立並連線至 Aurora PostgreSQL 資料庫叢集。

```
datasource:
   jicsDs:
     driver-class-name :
     url:
     username:
     password:
     type :
```

5. 建置映像並將其推送到您的 Amazon ECR 儲存庫。如需說明，請參閱《Amazon Elastic Container Registry 使用者指南》中的推送 Docker 映像。然後，根據您的情況，使用 Amazon ECR 映像建立 Amazon EKS Pod 或 Amazon ECS 任務定義，並將其部署到您的叢集。如需建立這些範例，請參閱《Amazon Elastic Container Service (Amazon ECS) 開發人員指南》中的使用主控台建立任務定義，以及《Amazon EKS 使用者指南》中的部署範例應用程式。
6. 具體而言，對於由案例管理的 Amazon ECS AWS Fargate，在建立任務定義時，請使用您在初始基礎設施設定中建立的 IAM 角色。然後，在建立服務時，展開聯網區段，並設定您在初始基礎設施設定中建立的 VPC、子網路和安全群組。請參閱 AWS Blu Age Runtime （非受管）的基礎設施設定要求。

測試部署的應用程式

如需如何測試 PlanetsDemo 應用程式的範例，請參閱 the section called “測試 PlanetsDemo 應用程式”。

升級 容器上的 AWS Blu Age 執行期

本指南說明如何升級容器上的 AWS Blu Age 執行期。若要這樣做，您需要先完成一些先決條件，然後使用 Docker 映像來升級 AWS Blu Age 執行期。

主題

- 先決條件
- 升級 AWS Blu Age 執行期

先決條件

開始之前，請確定您符合下列先決條件。

- 完成 the section called “AWS Blu Age 執行期先決條件”和 the section called “加入 AWS 藍卡年齡執行期 ”。
- 下載您要升級的 AWS Blu Age Runtime 版本。如需詳細資訊，請參閱the section called “加入 AWS 藍卡年齡執行期 ”。框架包含兩個二進位檔案： `aws-bluage-runtime-x.x.x.x.tar.gz`和 `aws-bluage-webapps-x.x.x.x.tar.gz`。

升級 AWS Blu Age 執行期

完成下列步驟以升級 AWS Blu Age 執行期。

1. 使用所需的 AWS Blu Age Runtime 版本重建 Docker 映像。如需說明，請參閱the section called “在容器上設定 AWS Blu Age 執行期”。
2. 將 Docker 映像推送至 Amazon ECR 儲存庫。
3. 停止並重新啟動您的 Amazon ECS 或 Amazon EKS 服務。
4. 驗證日誌。

成功升級 AWS Blu Age Runtime。

在容器上設定 AWS 藍時執行期的 Amazon CloudWatch 警示

您可以設定 CloudWatch，以便在部署的應用程式遇到例外狀況時收到更多可見的通知。這可協助您監控重新導向至 CloudWatch 的應用程式日誌，並新增警示來警告您可能發生的錯誤。

警示設定

使用 CloudWatch 日誌，您可以根據您的應用程式和需求，設定任意數量的指標和警示。

具體而言，您可以直接在叢集建立期間設定主動警示，以便在發生錯誤時收到通知。若要反白連線至 AWS Blu Age 控制系統的錯誤，請在日誌中新增有關字串 "Error C" 的指標。然後，您可以定義對此指標做出反應的警示。

在容器的 AWS 藍時空執行期中設定授權相依性

本主題說明如何設定可在容器上使用的 AWS Blu Age Runtime 的其他授權相依性。

主題

- 先決條件
- 概觀

先決條件

開始之前，請務必完成下列先決條件。

- 完成 the section called “AWS Blu Age 執行期先決條件”和 the section called “加入 AWS 藍卡年齡執行期 ”。
- 從來源取得下列相依性。

Oracle 資料庫

提供 Oracle 資料庫驅動程式。例如，ojdbc11-23.3.0.23.09.jar。

IBM MQ 連線

提供 IBM MQ 用戶端。例如，https：//com.ibm.mq..jakarta.client-9.3.4.1.jar。

在此相依性版本中， 也提供下列可轉移相依性：

- bcprov-jdk15to18-1.76.jar
- bcpkix-jdk15to18-1.76.jar
- bcutil-jdk15to18-1.76.jar

DDS 印表機檔案

提供 Jasper 報告程式庫 (https://community.jaspersoft.com/download-jaspersoft/community-edition)。例如，jasperreports-6.16.0.jar，但較新的版本可能相容。

在此相依性版本中， 也提供下列可轉移相依性：

- castor-core-1.4.1.jar
- castor-xml-1.4.1.jar
- commons-digester-2.1.jar
- ecj-3.21.0.jar
- itext-2.1.7.js8.jar
- javax.inject-1.jar
- jcommon-1.0.23.jar
- jfreechart-1.0.19.jar
- commons-beanutils-1.9.4.jar
- commons-collections-3.2.2.jar

概觀

若要安裝相依性，請完成下列步驟。

1. 視需要將上述任何相依性複製到 Docker 映像建置資料夾。

2. 如果您的 JICS 資料庫託管在 Oracle 上，請在 中提供 Oracle 資料庫驅動程式*your-tomcat-path*/extra。
3. 在您的 Dockerfile 上，將這些相依性複製到 *your-tomcat-path*/extra。
4. 建置您的 Docker 映像，然後將其推送至 Amazon ECR。
5. 停止並重新啟動您的 Amazon ECS 或 Amazon EKS 服務。
6. 檢查日誌。

測試 PlanetsDemo 應用程式

若要檢查已部署的 PlanetsDemo 應用程式的狀態，請在將 load-balancer-DNS-name、listener-port和 取代為設定web-binary-name正確的值後執行下列命令。

```
curl http://load-balancer-DNS-name:listener-port/gapwalk-application/
```

如果應用程式正在執行，您會看到下列輸出訊息：Jics application is running。

接著，執行下列命令。

```
curl http://load-balancer-DNS-name:listener-port/jac/api/services/rest/jicsservice/
```

如果應用程式正在執行，您會看到下列輸出訊息：Jics application is running。

```
Jics application is running
```

如果您已設定 Blusam，則在執行下列命令時，您可以預期會有空的回應。

```
curl http://load-balancer-DNS-name:listener-port/bac/api/services/rest/bluesamserver/
serverIsUp
```

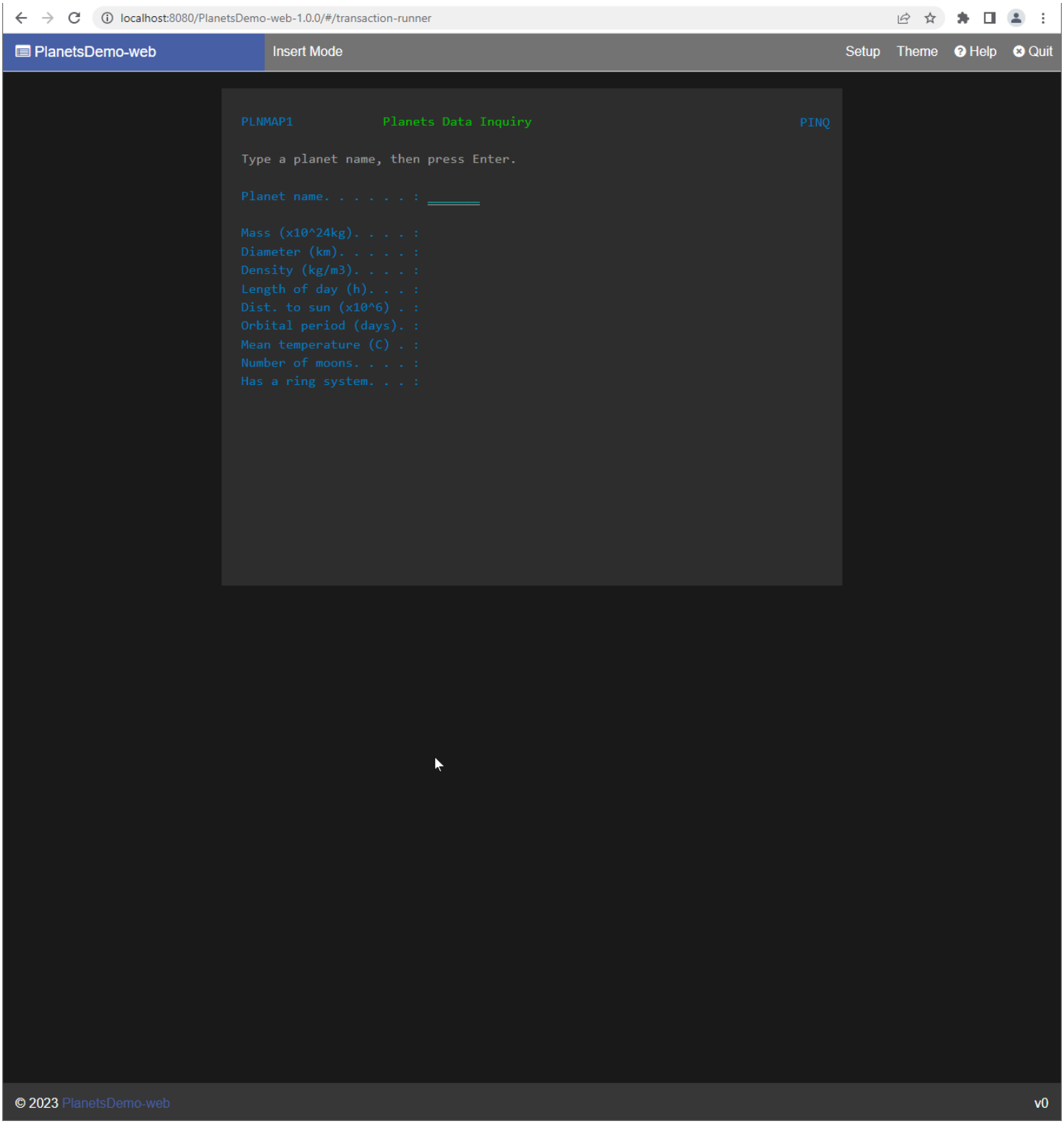
請注意 Web 二進位檔的名稱 (PlanetsDemo-web-1.0.0，如果未變更）。若要存取 PlanetsDemo 應用程式，請使用下列格式的 URL。

```
https://load-balancer-DNS-name:listener-port/web-binary-name
```

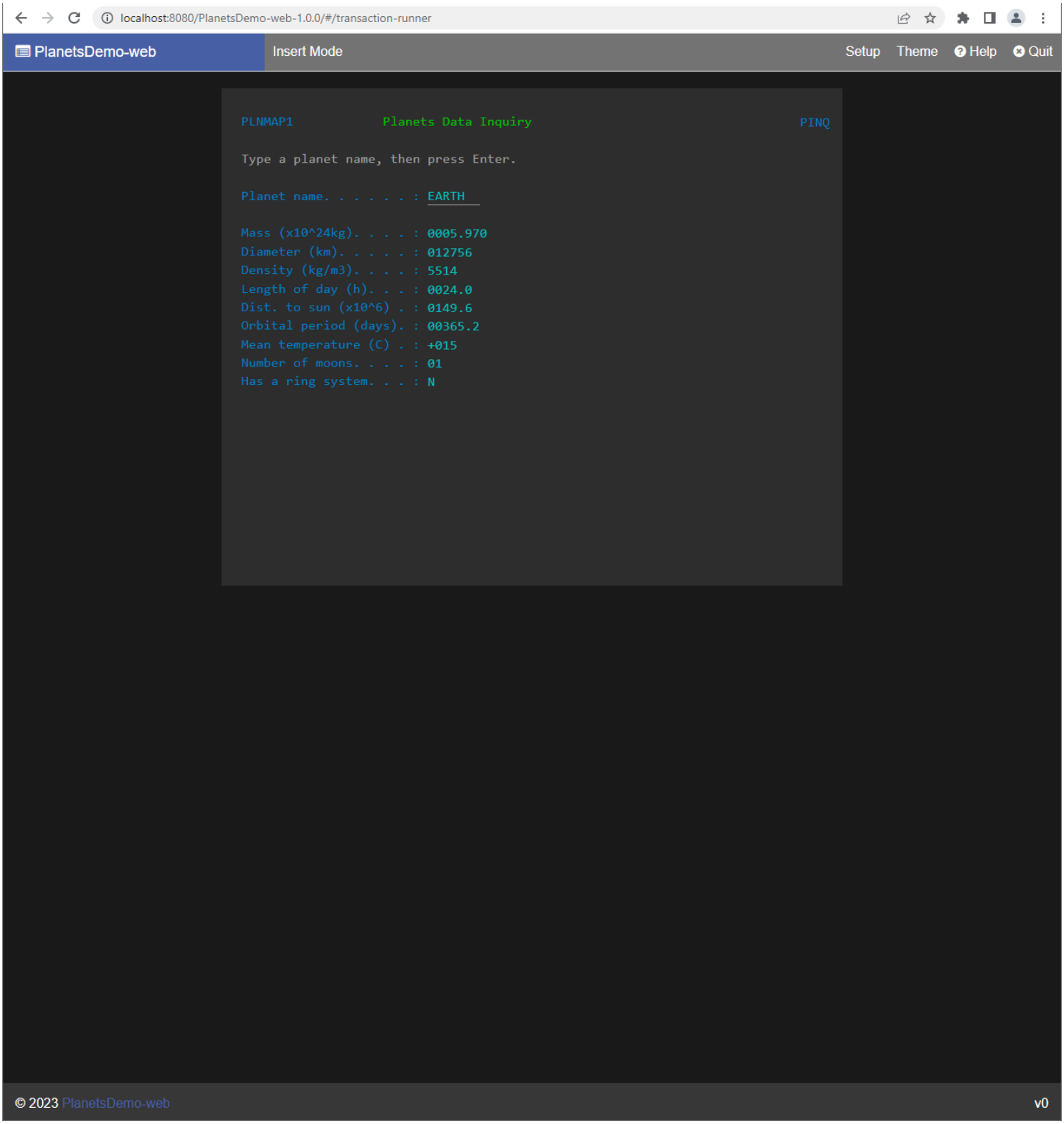
啟動 PlanetsDemo 應用程式後，會顯示首頁。



在文字方塊中輸入 PINQ，然後按 Enter。資料查詢頁面隨即顯示。



例如，在 PlanetsDemo 名稱欄位中輸入 EARTH，然後按 Enter。隨即顯示您輸入的行星頁面。



AWS Blu Age Runtime 可在下列區域使用： 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）、 加拿大（中部）、 歐洲（愛爾蘭）區域、歐洲（倫敦）區域、 歐洲（巴黎）區域、 歐洲（法蘭克福）、 歐洲（斯德哥爾摩）區域、 歐洲（米蘭）區域、 歐洲（西班牙）區域、 南美洲（聖保羅）、 亞太區域（東京），亞太區域（首

爾）、 亞太區域 （大阪）、 亞太區域 （新加坡）、 亞太區域 （雪梨）， 亞太區域 （孟買）， 非洲（開普敦）、 和以色列 （特拉維夫）。

使用 Blu Age Developer IDE 修改原始程式碼

如果您使用的是 AWS受管 AWS 的 Blu Age 執行期引擎，您可以使用 Blu Age Developer 來修改產生的原始程式碼。如果您因為某些原因而需要更新現代化程式碼，或舊版原始碼的一部分無法進行現代化，建議您執行此操作。您可以透過 Amazon AppStream 2.0 存取 Blu Age Developer。本節說明如何在 AppStream 2.0 上設定 Blu Age Developer。它還說明如何使用 Blu Age Developer 更新原始程式碼，並使用範例應用程式 PlanetsDemo。

主題

- 教學課程：為 AWS 藍年開發人員 IDE 設定 AppStream 2.0
- 教學課程：在 AppStream 2.0 上使用 AWS Blu Age Developer

教學課程：為 AWS 藍年開發人員 IDE 設定 AppStream 2.0

AWS Mainframe Modernization 透過 Amazon AppStream 2.0 提供數種工具。AppStream 2.0 是一項全受管、安全的應用程式串流服務，可讓您將桌面應用程式串流給使用者，而無需重寫應用程式。AppStream 2.0 可讓使用者立即存取所需的應用程式，並在他們選擇的裝置上提供回應靈敏、流暢的使用者體驗。使用 AppStream 2.0 託管執行期引擎特定工具，可讓客戶應用程式團隊直接從 Web 瀏覽器使用工具，並與存放在 Amazon S3 儲存貯體或 CodeCommit 儲存庫中的應用程式檔案互動。

如需有關 AppStream 2.0 中瀏覽器支援的資訊，請參閱《Amazon AppStream 2.0 管理指南》中的系統需求和功能支援 (Web 瀏覽器）。如果您在使用 AppStream 2.0 時遇到問題，請參閱《Amazon AppStream 2.0 管理指南》中的對 AppStream 2.0 使用者問題進行故障診斷。 Amazon AppStream 2.0

本文件說明如何在 AppStream 2.0 機群上設定 AWS Blu Age Developer IDE。

主題

- 先決條件
- 步驟 1：建立 Amazon S3 儲存貯體
- 步驟 2：將政策連接至 S3 儲存貯體
- 步驟 3：將檔案上傳至 Amazon S3 儲存貯體
- 步驟 4：下載 AWS CloudFormation 範本

- 步驟 5：使用 建立機群 AWS CloudFormation
- 步驟 6：存取執行個體
- 清除資源

先決條件

第一次使用時，請執行下列動作：

1. 導覽至 AppStream 2.0 主控台，網址為 https：//https://console.aws.amazon.com/appstream2/home。
2. 選擇開始使用。
3. 選擇 Skip (跳過)。

> **Important**
>
> Amazon AppStream 2.0 使用 IAM 角色來管理您的 AppStream 2.0 資源 AWS ，並在您執行此操作時建立這些角色。

然後，下載封存檔案，其中包含您在 AppStream 2.0 下設定 AWS Blu Age Developer IDE 所需的成品。

> **Note**
>
> 這是大型檔案。如果您在操作逾時時遇到問題，建議您使用 Amazon EC2 執行個體來改善上傳和下載效能。如需啟動和連線至 Amazon EC2 執行個體的詳細資訊，請參閱開始使用 Amazon EC2。

步驟 1：建立 Amazon S3 儲存貯體

在與您要建立的 AppStream 2.0 機群 AWS 區域 相同的 中建立 Amazon S3 儲存貯體。此儲存貯體將包含完成本教學課程所需的成品。如需儲存貯體的詳細資訊，請參閱建立儲存貯體。

步驟 2：將政策連接至 S3 儲存貯體

將下列政策連接至您為此教學課程建立的 儲存貯體。如需將政策連接至 S3 儲存貯體的詳細資訊，請參閱新增儲存貯體政策。

請務必amzn-s3-demo-bucket將 取代為您建立之儲存貯體的實際名稱。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "AllowAppStream2.0ToRetrieveObjects",
        "Effect": "Allow",
        "Principal": {
            "Service": "appstream.amazonaws.com"
        },
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    }]
}
```

步驟 3：將檔案上傳至 Amazon S3 儲存貯體

解壓縮您在先決條件中下載的檔案，並將appstream資料夾上傳至您的儲存貯體。上傳此資料夾會在您的儲存貯體中建立正確的結構。如需詳細資訊，請參閱《Amazon S3 使用者指南》中的上傳物件。

步驟 4：下載 AWS CloudFormation 範本

下載下列 AWS CloudFormation 範本。您需要這些範本來建立和填入 AppStream 2.0 機群。

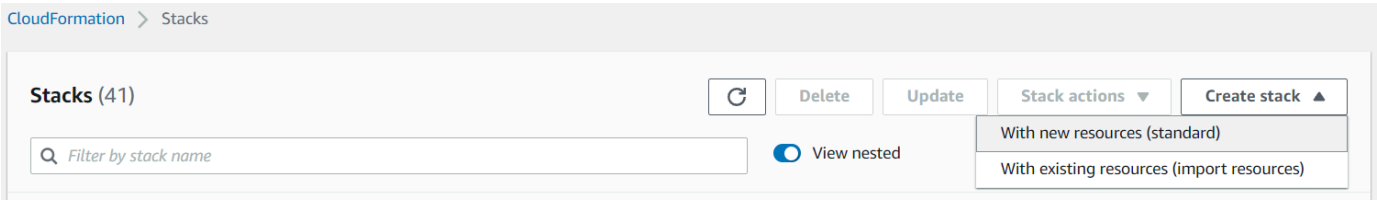
- cfn-m2-appstream-elastic-fleet-linux.yaml
- cfn-m2-appstream-bluage-dev-tools-linux.yaml
- cfn-m2-appstream-bluage-shared-linux.yaml
- cfn-m2-appstream-chrome-linux.yaml
- cfn-m2-appstream-eclipse-jee-linux.yaml
- cfn-m2-appstream-pgadmin-linux.yaml

步驟 5：使用 建立機群 AWS CloudFormation

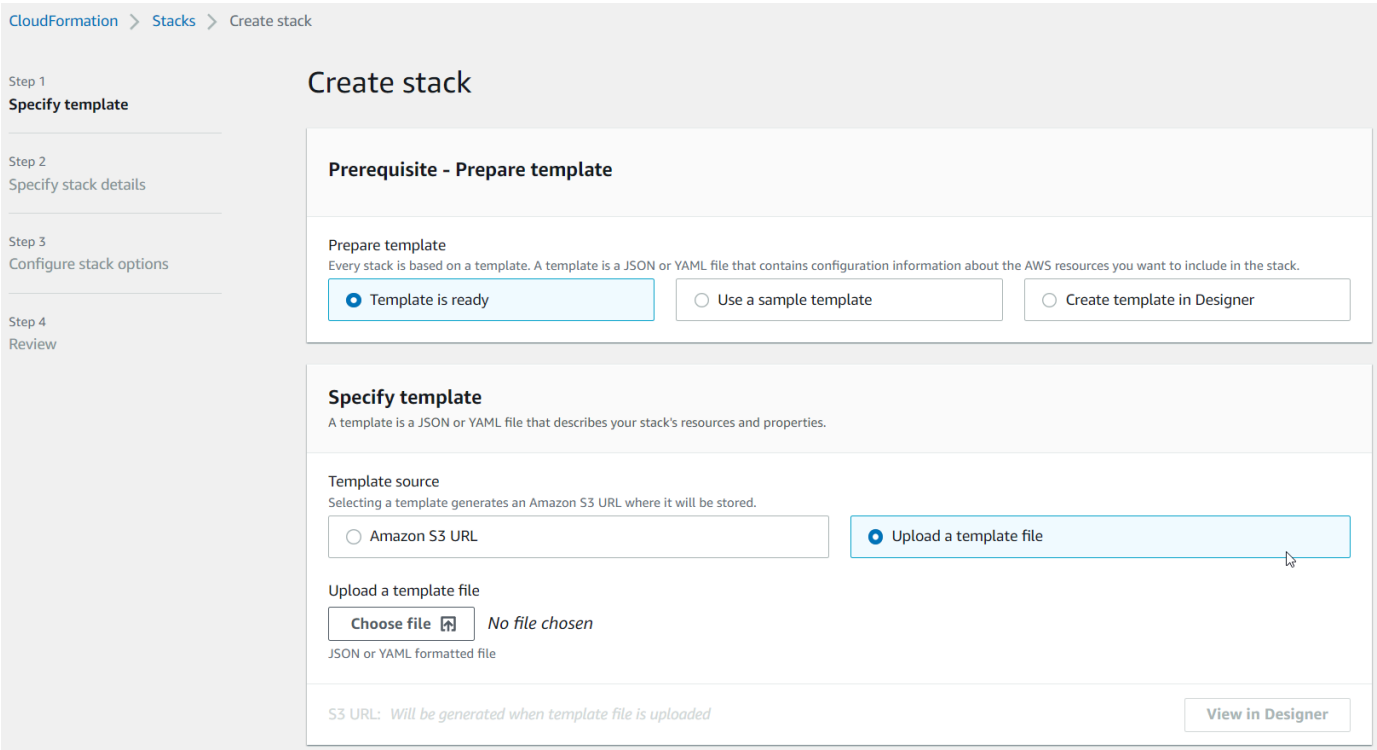
在此步驟中，您會使用 `cfn-m2-appstream-elastic-fleet-linux.yaml` AWS CloudFormation 範本來建立 AppStream 2.0 機群，並使用堆疊來託管 AWS Blu Age 開發人員 IDE。建立機群和堆疊之後，您將執行您在上一個步驟中下載的其他 AWS CloudFormation 範本，以安裝 Developer IDE 和其他必要的工具。

1. 在 AWS 管理主控台 AWS CloudFormation 中導覽至 ，然後選擇 Stacks。

2. 在堆疊中，選擇建立堆疊和使用新資源（標準）：



3. 在建立堆疊中，選擇選擇現有範本並上傳範本檔案：



4. 選擇選擇檔案，然後導覽至檔案 `cfn-m2-appstream-elastic-fleet-linux.yaml`。選擇下一步。

5. 在指定堆疊詳細資訊中，提供下列資訊：

   - 堆疊的名稱。
   - 您的預設安全群組和該安全群組的兩個子網路。

> **Note**
>
> 安全群組的兩個子網路必須位於不同的可用區域。

6. 選擇下一步。

7. 向下導覽頁面，然後選擇我確認 AWS CloudFormation 可能會使用自訂名稱建立 IAM 資源。

8. 選擇下一步。
9. 檢閱詳細資訊，然後選擇提交。
10. 建立機群後，請使用所有其他下載的範本建立 CloudFormation 堆疊，以完成應用程式設定。每次都請務必更新 BucketName，以指向正確的 S3 儲存貯體。您可以在 CloudFormation 主控台中編輯 BucketName。或者，您可以直接編輯範本檔案並更新 S3Bucket 屬性。

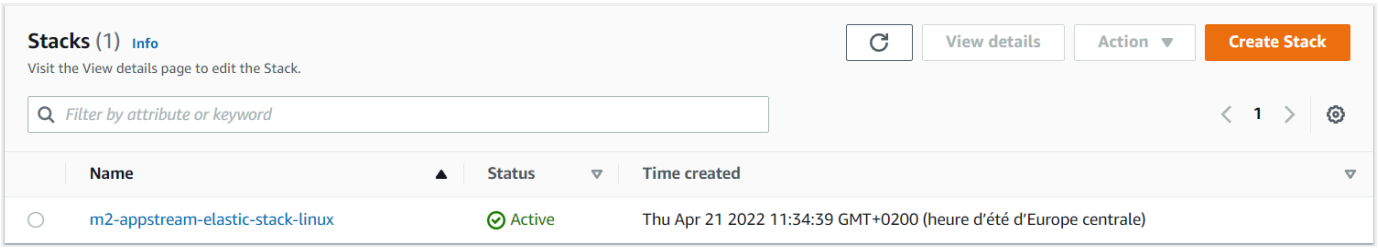
> **Note**
>
> 下載的範本預期會在 S3 儲存貯體中找到名為 的資料夾結構資產`appstream/bluage/developer-ide/`。儲存貯體必須與您建立的機群 AWS 區域 位於相同的 中。

> **Important**
>
> 執行步驟 4 中下載的所有 CloudFormation 指令碼，以正確設定您的應用程式。

步驟 6：存取執行個體

建立並啟動機群後，您可以建立臨時連結，透過原生用戶端存取機群。

1. 導覽至 中的 AppStream 2.0， AWS Management Console 然後選擇先前建立的堆疊：



2. 在堆疊詳細資訊頁面上，選擇堆疊，然後選擇關聯機群。
3. 在提示中，選擇您先前建立並開始的機群。
4. 選擇關聯。
5. 選擇相關聯的堆疊，然後從動作功能表中，選擇建立串流 URL，輸入任意使用者 ID 和 URL 過期時間，然後選擇取得 URL。您可以取得 URL，可用來串流到瀏覽器或原生用戶端。我們建議您串流到原生用戶端。

清除資源

如需清除已建立堆疊和機群的程序，請參閱建立 AppStream 2.0 機群和堆疊。

當您刪除 AppStream 2.0 物件時，您或帳戶管理員也可以清除應用程式設定和主資料夾的 S3 儲存貯體。

> **Note**
>
> 指定使用者的主資料夾在所有機群中都是唯一的，因此如果相同帳戶中的其他 AppStream 2.0 堆疊處於作用中狀態，您可能需要保留它。

您無法使用 AppStream 2.0 主控台刪除使用者。反之，您必須搭配 使用 服務 API AWS CLI。如需詳細資訊，請參閱《Amazon AppStream 2.0 管理指南》中的使用者集區管理。 Amazon AppStream 2.0

教學課程：在 AppStream 2.0 上使用 AWS Blu Age Developer

本教學課程說明如何在 AppStream 2.0 上存取 AWS Blu Age Developer，並將其與範例應用程式搭配使用，以便您可以試用功能。完成本教學課程後，您可以搭配自己的應用程式使用相同的步驟。

主題

- 步驟 1：建立資料庫
- 步驟 2：存取環境
- 步驟 3：設定執行時間
- 步驟 4：啟動 Eclipse IDE
- 步驟 5：設定 Maven 專案
- 步驟 6：設定 Tomcat 伺服器
- 步驟 7：部署至 Tomcat
- 步驟 8：建立 JICS 資料庫
- 步驟 9：啟動並測試應用程式
- 步驟 10：偵錯應用程式
- 清除資源

步驟 1：建立資料庫

在此步驟中，您會使用 Amazon RDS 建立受管 PostgreSQL 資料庫，示範應用程式會使用此資料庫來存放組態資訊。

1. 開啟 Amazon RDS 主控台。
2. 選擇資料庫 > 建立資料庫。
3. 選擇標準建立 > PostgreSQL，保留預設版本，然後選擇免費方案。
4. 選擇資料庫執行個體識別符。
5. 針對登入資料設定，選擇在 中管理主要登入 AWS Secrets Manager資料。如需詳細資訊，請參閱《Amazon RDS 使用者指南》中的使用 Amazon RDS 和 AWS Secrets Manager進行密碼管理。
6. 請確定 VPC 與您用於 AppStream 2.0 執行個體的 VPC 相同。您可以向管理員詢問此值。
7. 針對 VPC 安全群組，選擇建立新。
8. 將公有存取權設定為是。
9. 保留所有其他預設值。檢閱這些值。
10. 選擇建立資料庫。

若要從執行個體存取資料庫伺服器，請在 Amazon RDS 中選取資料庫伺服器。在連線和安全性下，選擇資料庫伺服器的 VPC 安全群組。此安全群組先前已為您建立，且應具有類似 RDS 管理主控台所建立之安全群組的說明。選擇動作 > 編輯傳入規則，選擇新增規則，然後建立 PostgreSQL 類型的規則。對於規則來源，請使用安全群組預設值 您可以開始在來源欄位中輸入來源名稱，然後接受建議的 ID。最後，選擇儲存規則。

步驟 2：存取環境

在此步驟中，您會存取 AppStream 2.0 上的 AWS Blu Age 開發環境。

1. 如需存取 AppStream 2.0 執行個體的適當方式，請聯絡您的管理員。如需可能用戶端和組態的一般資訊，請參閱《Amazon AppStream 2.0 管理指南》中的 AppStream 2.0 存取方法和用戶端。Amazon AppStream 2.0 請考慮使用原生用戶端以獲得最佳體驗。
2. 在 AppStream 2.0 中，選擇桌面。

步驟 3：設定執行時間

在此步驟中，您會設定 AWS Blu Age 執行時間。您必須在第一次啟動時設定執行時間，並在收到執行時間升級通知時再次設定。此步驟會填入您的 `.m2`資料夾。

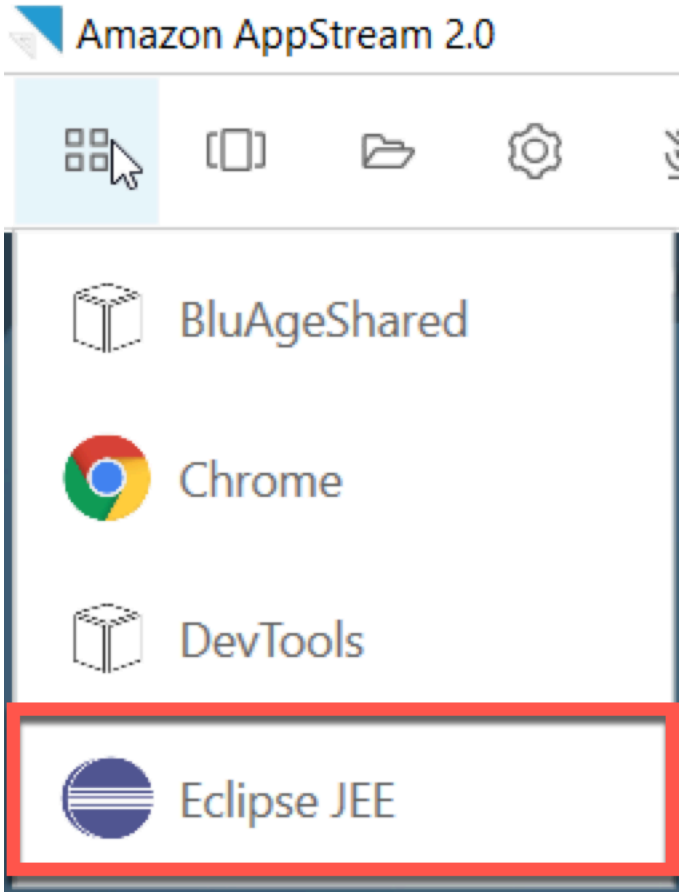
1. 從選單列選擇應用程式，然後選擇終端機。
2. 輸入以下命令：

```
~/_install-velocity-runtime.sh
```

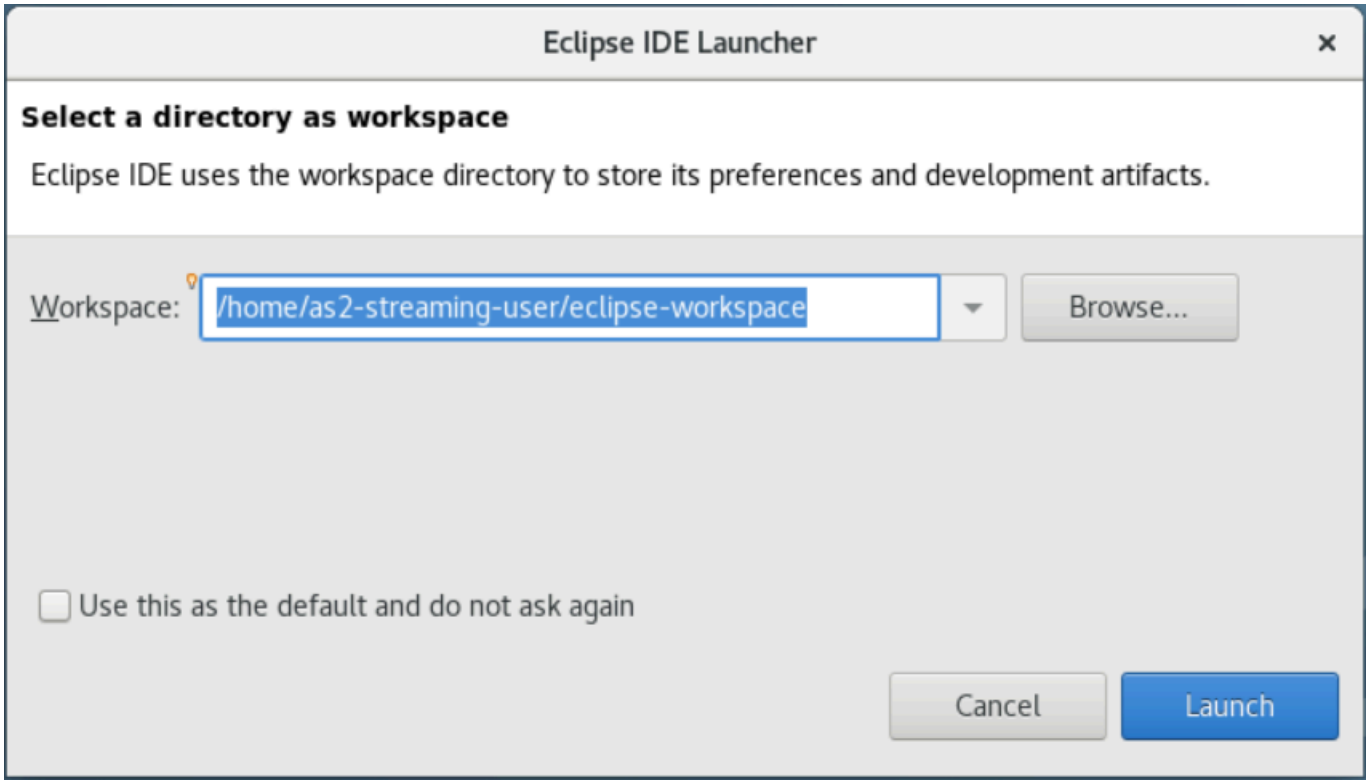
步驟 4：啟動 Eclipse IDE

在此步驟中，您會啟動 Eclipse IDE，並選擇您要建立工作區的位置。

1. 在 AppStream 2.0 中，選擇工具列上的啟動應用程式圖示，然後選擇 Eclipse JEE。



2. 啟動器開啟時，輸入您要建立工作區的位置，然後選擇啟動。



或者，您可以從命令列啟動 Eclipse，如下所示：

```
~/eclipse &
```

步驟 5：設定 Maven 專案

在此步驟中，您會為地球示範應用程式匯入 Maven 專案。

1. 將 PlanetsDemo-pom.zip 上傳至您的主資料夾。您可以使用原生用戶端「我的檔案」功能來執行此操作。
2. 使用`unzip`命令列工具來擷取檔案。
3. 在解壓縮資料夾內導覽，並在文字編輯器中開啟專案`pom.xml`的根目錄。
4. 編輯 `gapwalk.version` 屬性，使其符合已安裝的 AWS Blu Age 執行時間。

   如果您不確定安裝的版本，請在終端機中發出下列命令：

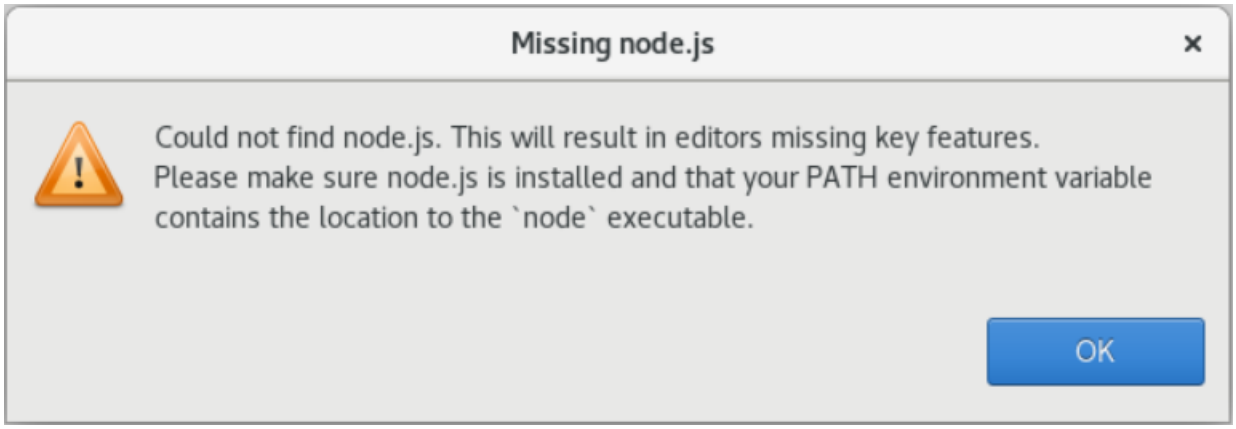
   ```
   cat ~/runtime-version.txt
   ```

   此命令會列印目前可用的執行時間版本，例如 `3.1.0-b3257-dev`。

> **Note**
>
> 請勿在 -dev 中包含尾碼gapwalk.version。例如，有效值為 <gapwalk.version>3.1.0-b3257</gapwalk.version>。

5. 在 Eclipse 中，選擇檔案，然後選擇匯入。在匯入對話方塊中，展開 Maven，然後選擇現有 Maven 專案。選擇下一步。
6. 在匯入 Maven 專案中，提供解壓縮檔案的位置，然後選擇完成。

   您可以安全地忽略下列快顯視窗。Maven 會下載 的本機副本node.js，以建置專案的角度 (*-web) 部分：

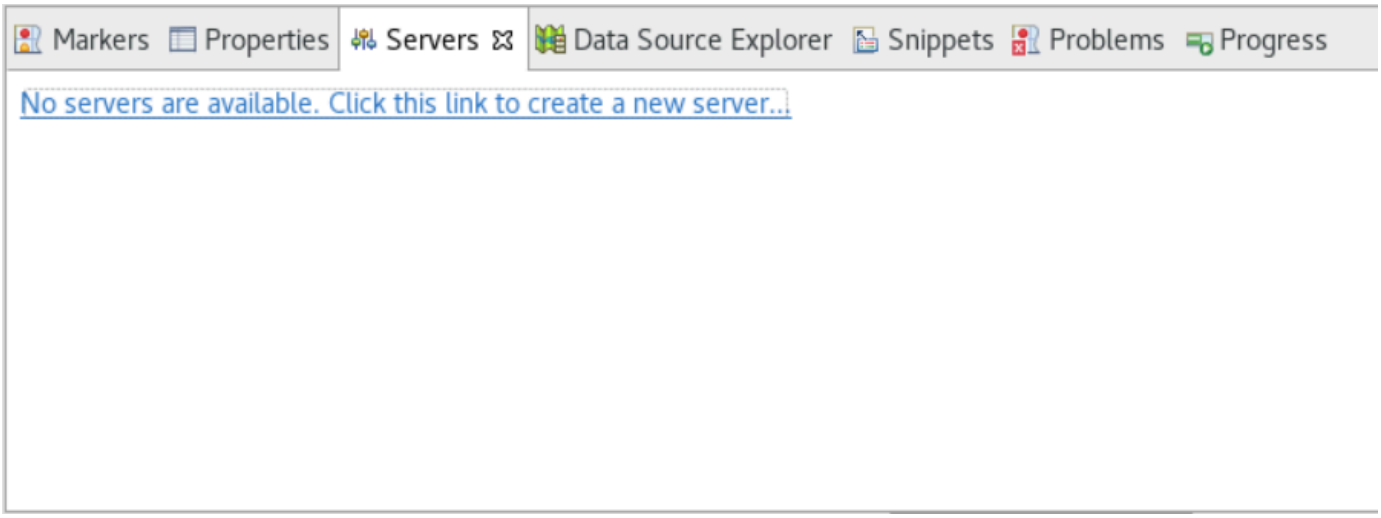


   等到建置結束。您可以在進度檢視中遵循建置。
7. 在 Eclipse 中，選取專案，然後選擇執行為。然後選擇 Maven 安裝。Maven 安裝成功後，它會在 下建立 war 檔案PlanetsDemoPom/PlanetsDemo-web/target/PlanetsDemo-web-1.0.0.war。

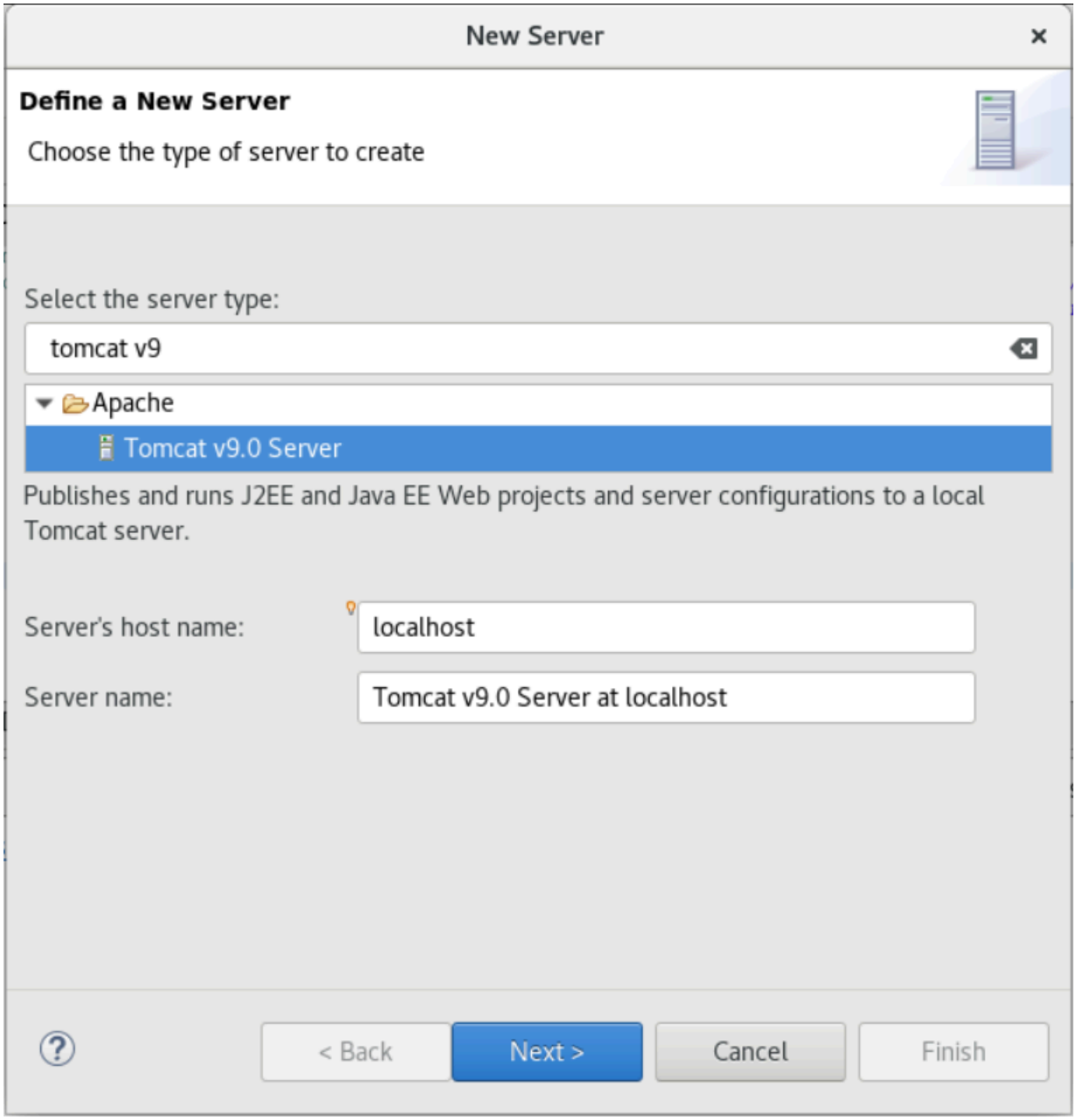
步驟 6：設定 Tomcat 伺服器

在此步驟中，您會在其中設定 Tomcat 伺服器，以部署和啟動已編譯的應用程式。

1. 在 Eclipse 中，選擇視窗 > 顯示檢視 > 伺服器以顯示伺服器檢視：

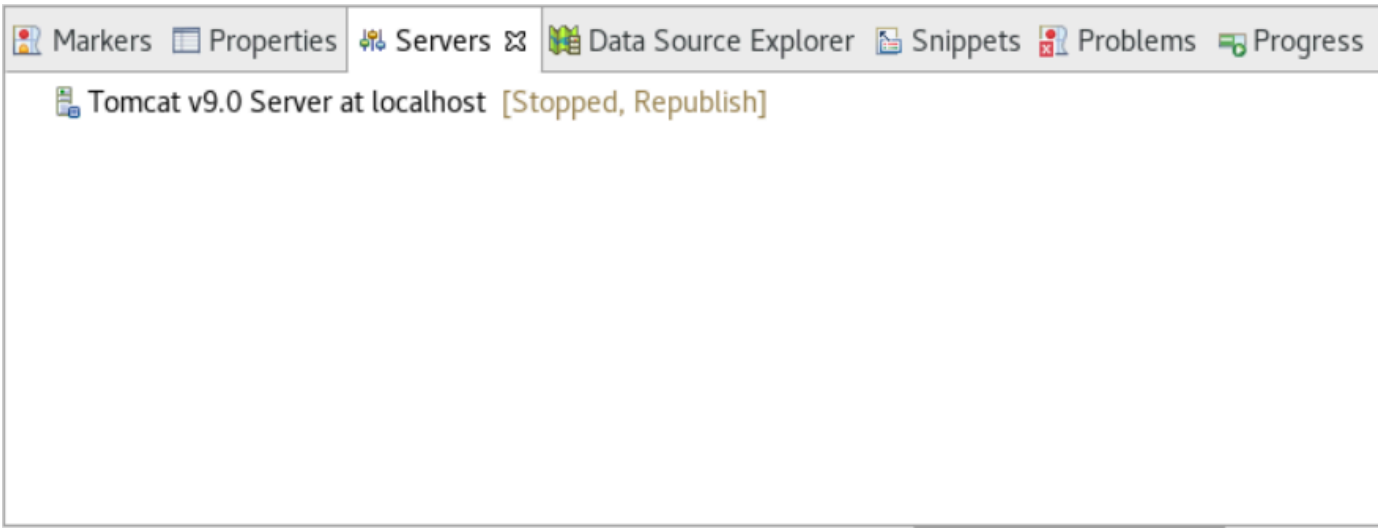


2. 選擇沒有可用的伺服器。按一下此連結以建立新的伺服器...。新伺服器精靈隨即出現。在精靈的選取伺服器類型欄位中，輸入 tomcat v9 ，然後選擇 Tomcat v9.0 Server。然後選擇下一步。



3. 選擇瀏覽，然後選擇主資料夾根目錄的 tomcat 資料夾。將 JRE 保留為預設值，然後選擇完成。

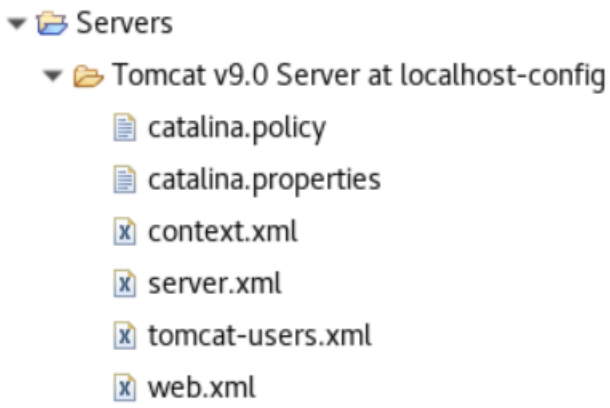
   伺服器專案會在工作區中建立，而 Tomcat v9.0 伺服器現在可在伺服器檢視中使用。這是將部署和啟動已編譯應用程式的位置：



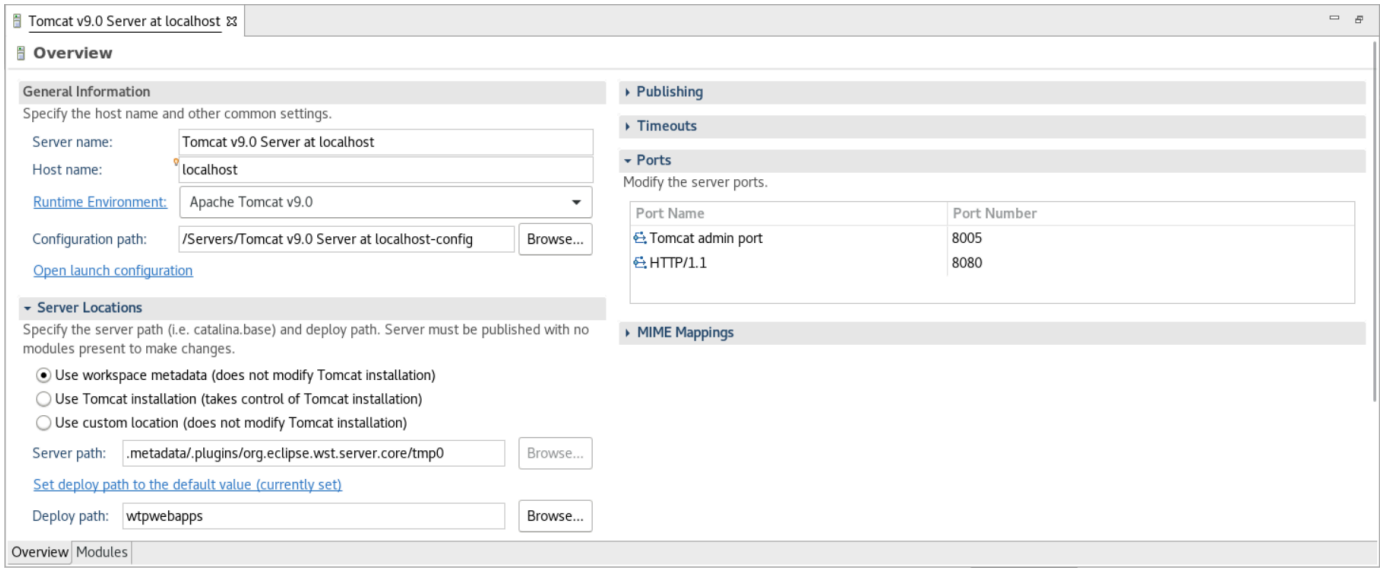
步驟 7：部署至 Tomcat

在此步驟中，您會將地球示範應用程式部署至 Tomcat 伺服器，以便執行應用程式。

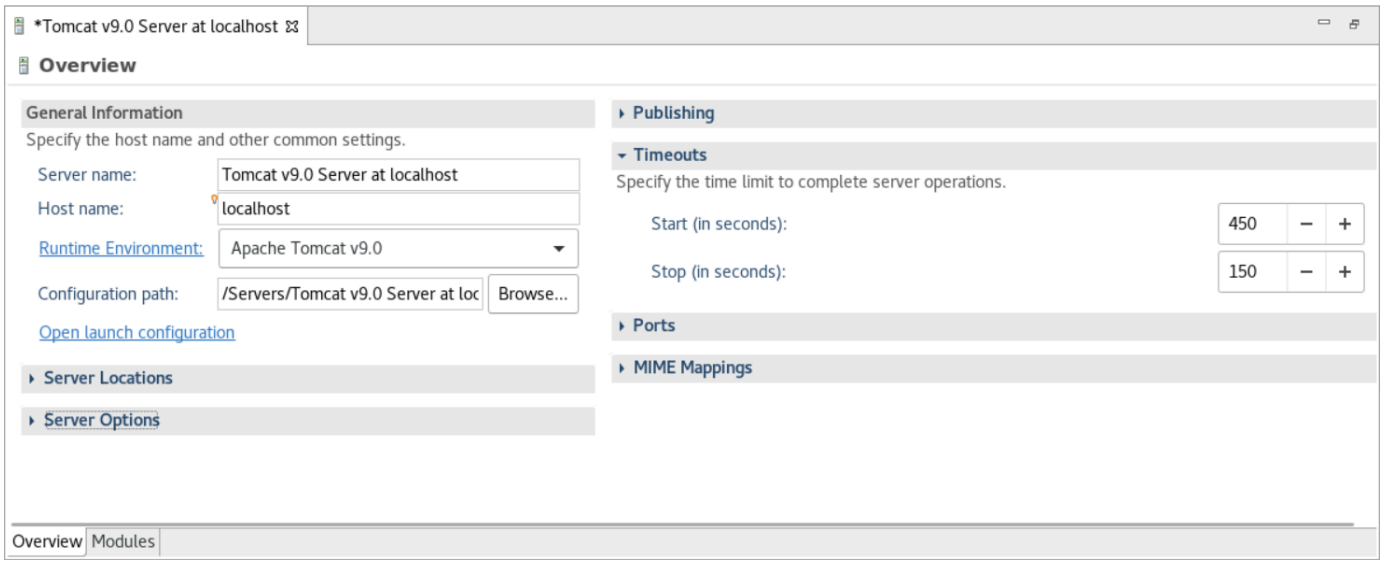
1. 選取 PlanetsDemo-web 檔案，然後選擇執行身分 > Maven 安裝。PlanetsDemo-web 再次選取 ，然後選擇重新整理，以確保 npm 編譯的前端正確編譯為 .war 並由 Eclipse 注意到。
2. 將 PlanetsDemo-runtime.zip 上傳至執行個體，並在可存取的位置解壓縮檔案。這可確保示範應用程式可以存取所需的組態資料夾和檔案。
3. 將 的內容複製到您為 Tomcat 伺服器建立的PlanetsDemo-runtime/tomcat-configServers/Tomcat v9.0...子資料夾中：



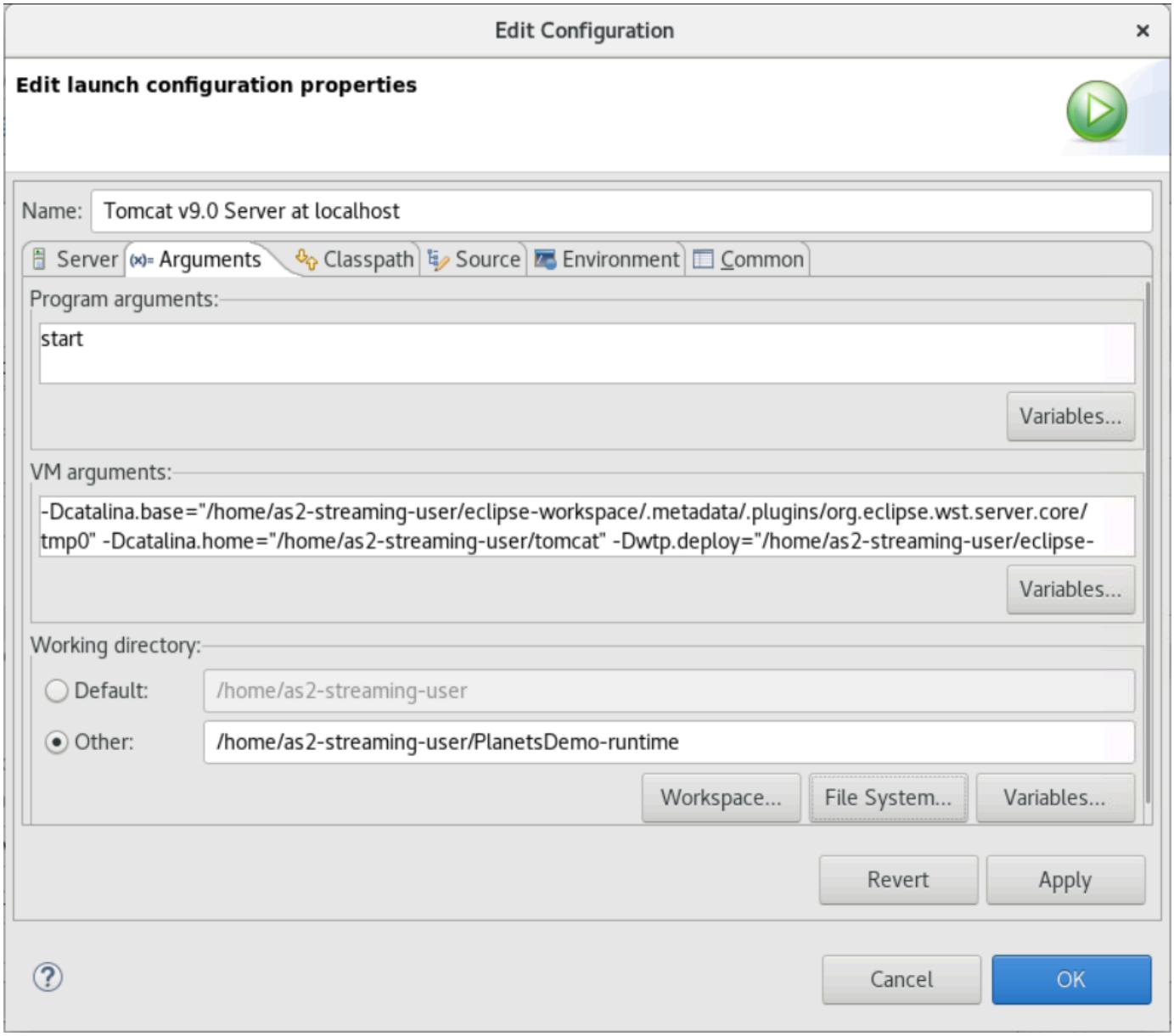
4. 在tomcat v9.0伺服器檢視中開啟伺服器項目。伺服器屬性編輯器隨即出現：



5. 在概觀索引標籤中，將啟動的逾時值增加至 450 秒，將停止增加至 150 秒，如下所示：



6. 選擇開啟啟動組態。精靈隨即出現。在精靈中，導覽至引數資料夾，然後在工作目錄中選擇其他。選擇檔案系統，然後導覽至先前解壓縮的`PlanetsDemo-runtime`資料夾。此資料夾應包含稱為 config 的直接子資料夾。

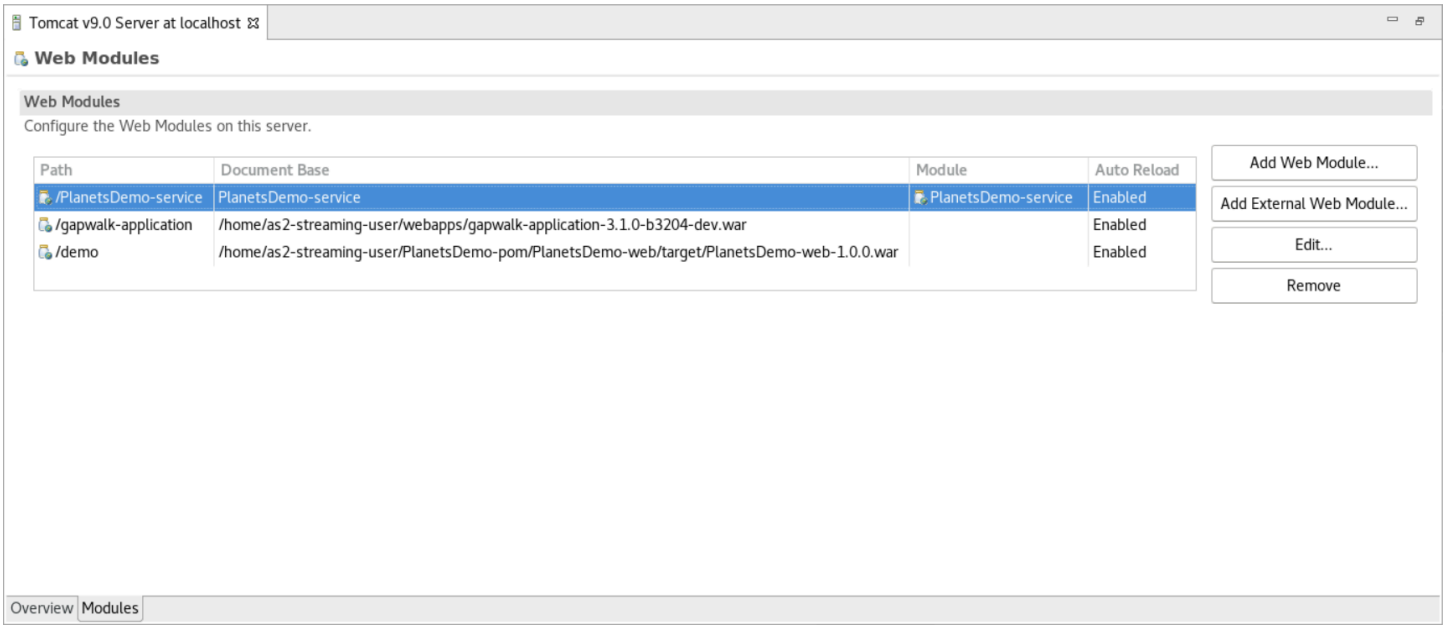


7. 選擇伺服器屬性編輯器的模組索引標籤，並進行下列變更：

   - 選擇新增 Web 模組並新增 `PlanetsDemo-service`。
   - 選擇新增外部 Web 模組。新增 Web 模組對話方塊視窗隨即出現。進行下列變更：
     - 在文件基礎中，選擇瀏覽並導覽至 `~/webapps/gapwalk-application...war`
     - 在路徑中，輸入 `/gapwalk-application`。
   - 選擇確定。
   - 再次選擇新增外部 Web 模組，並進行下列變更：
     - 針對文件基礎，輸入前端 .war 的路徑 （ 在 中`PlanetsDemo-web/target`)

    - 對於路徑，輸入 /demo
- 選擇 OK (確定)。
- 儲存編輯器修改 (Ctrl + S)。

編輯器內容現在應類似以下內容。



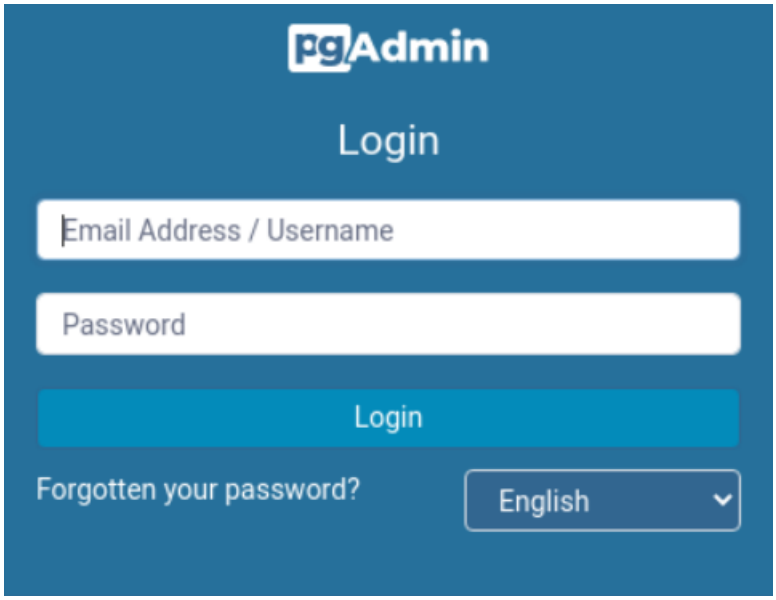
步驟 8：建立 JICS 資料庫

在此步驟中，您會連線到您在 中建立的資料庫步驟 1：建立資料庫。

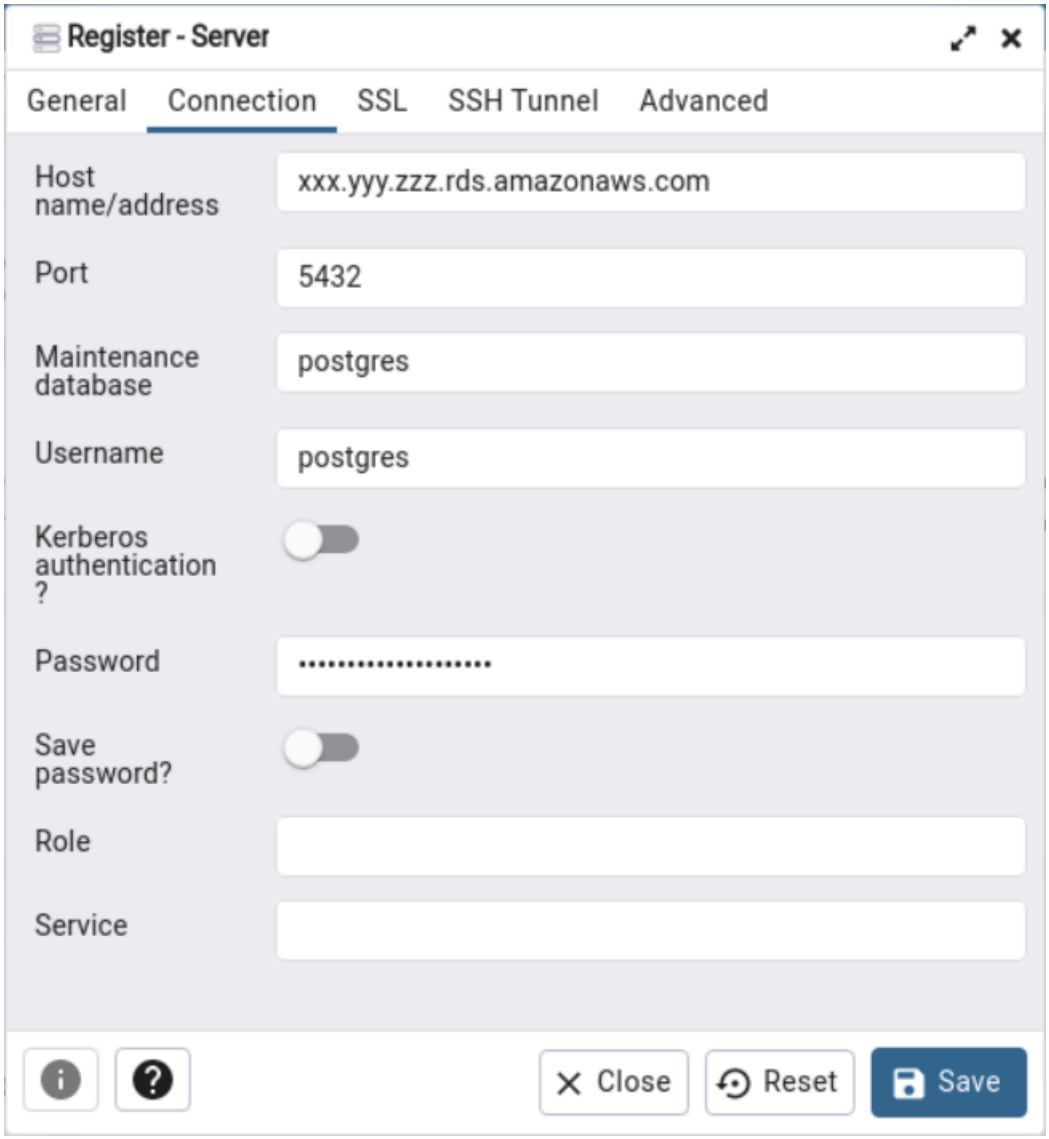
1. 從 AppStream 2.0 執行個體，在終端機中發出下列命令以啟動 pgAdmin：

    ```
    ./pgadmin-start.sh
    ```

2. 選擇電子郵件地址和密碼做為登入識別符。請記下提供的 URL （通常是 http://127.0.0.1:5050)。在執行個體中啟動 Google Chrome，複製 URL 並貼到瀏覽器中，然後使用您的識別符登入。



3. 登入後，請選擇新增伺服器，然後輸入先前建立的資料庫連線資訊，如下所示。

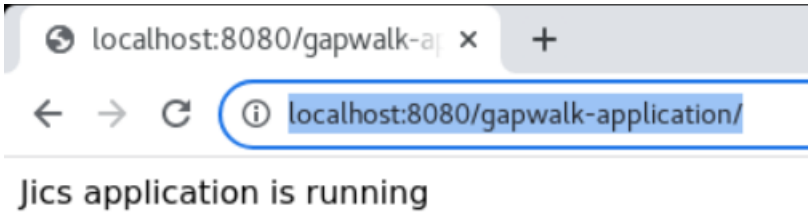


4. 當您連線到資料庫伺服器時，請使用物件 > 建立 > 資料庫，並建立新的資料庫，名稱為 jics。
5. 編輯示範應用程式使用的資料庫連線資訊。此資訊在 中定義`PlanetsDemo-runtime/config/application-main.yml`。搜尋`jicsDs`項目。若要擷取 `username`和 的值`password`，請在 Amazon RDS 主控台中，導覽至 資料庫。在 Configuration (組態) 索引標籤的 Master Credentials ARN (主要憑證 ARN) 中，選擇 Manage in Secrets Manager (在 Secrets Manager 中管理)。然後，在 Secrets Manager 主控台的秘密中，選擇擷取秘密值。

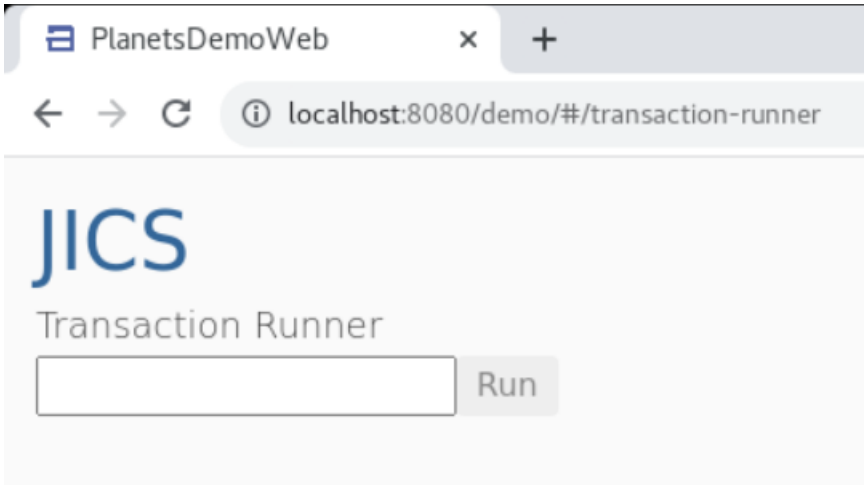
步驟 9：啟動並測試應用程式

在此步驟中，您會啟動 Tomcat 伺服器和示範應用程式，以便進行測試。

1. 若要啟動 Tomcat 伺服器和先前部署的應用程式，請在伺服器檢視中選取伺服器項目，然後選擇啟動。隨即出現主控台，顯示啟動日誌。
2. 在伺服器檢視中檢查伺服器狀態，或在 主控台中等待 【xxx】 毫秒內啟動伺服器訊息。伺服器啟動後，請檢查 gapwalk-application 是否已正確部署。若要這樣做，請在 Google Chrome 瀏覽器中存取 http：//localhost：8080/gapwalk-application URL。您應該會看到以下內容。



3. 從 Google Chrome 存取已部署的應用程式前端，網址為 https：//http://localhost:8080/demo。應該會出現下列交易啟動器頁面。

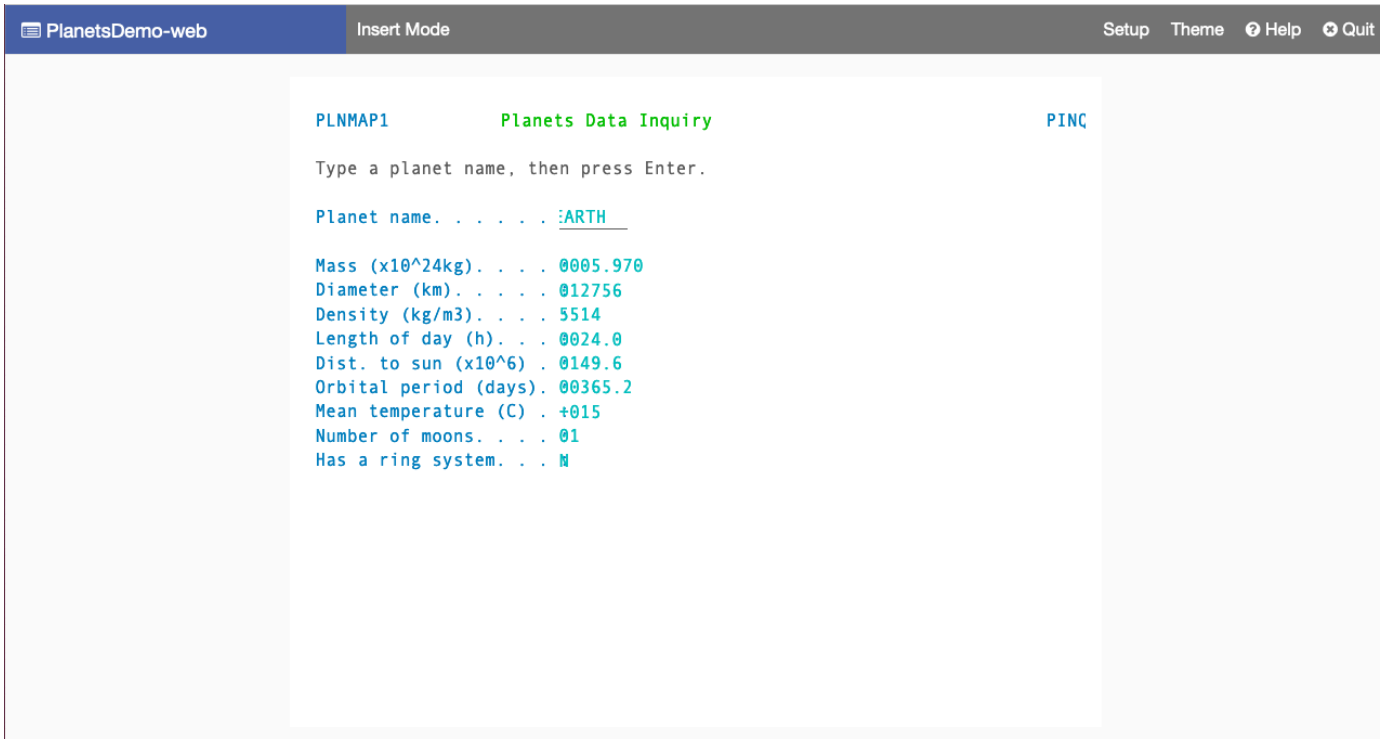


4. 若要啟動應用程式交易，PINQ請在輸入欄位中輸入 ，然後選擇執行 （ 或按 Enter)。

   示範應用程式畫面應會出現。



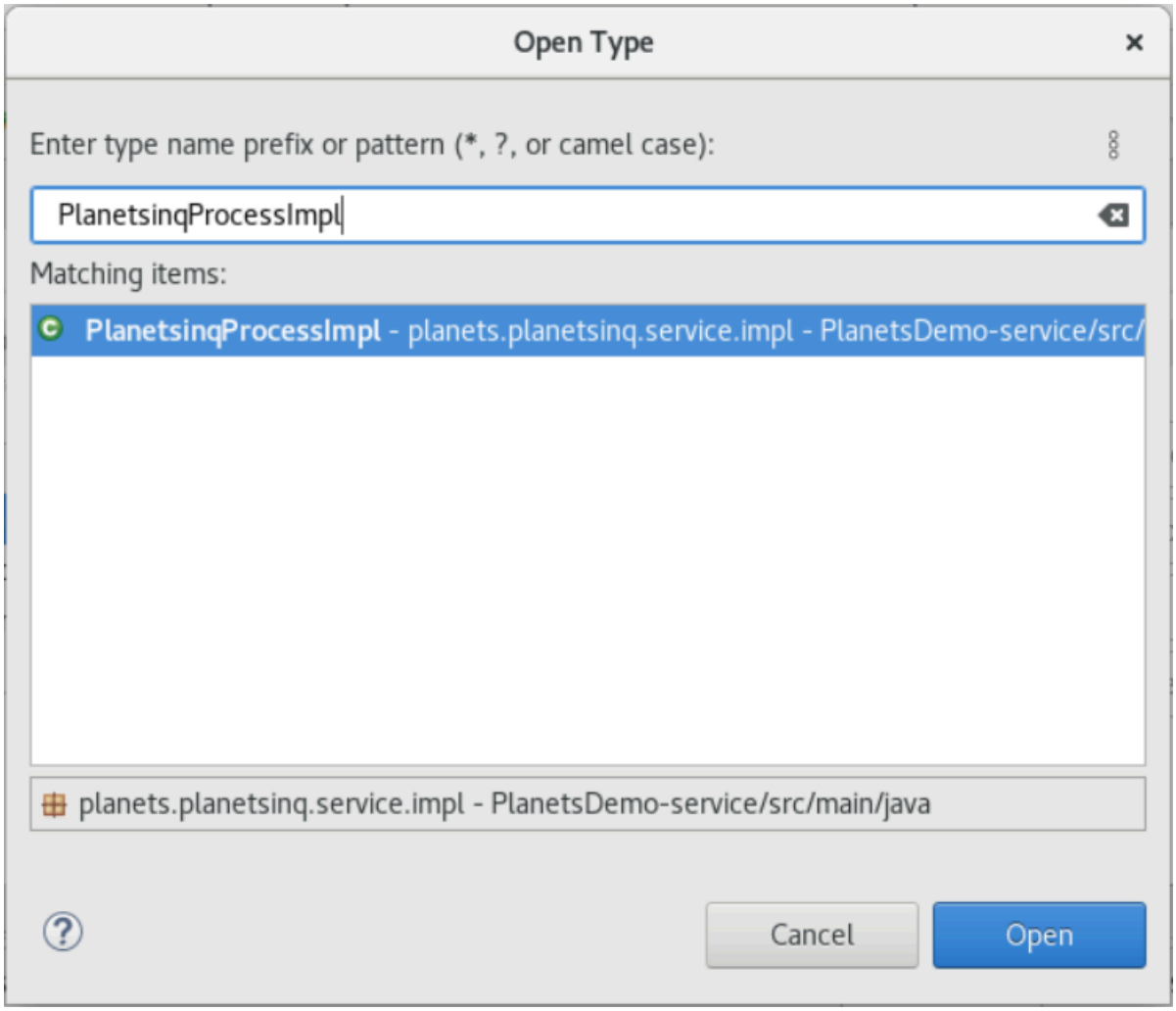
5. 在對應的欄位中輸入行星名稱，然後按 Enter 鍵。



步驟 10：偵錯應用程式

在此步驟中，您會使用標準 Eclipse 偵錯功能進行測試。當您使用現代化應用程式時，即可使用這些功能。

1. 若要開啟主要服務類別，請按 Ctrl + Shift + T。然後輸入 `PlanetsinqProcessImpl`。



2. 導覽至 `searchPlanet`方法，並在該處放置中斷點。
3. 選取伺服器名稱，然後選取在偵錯中重新啟動。
4. 重複上述步驟。也就是說，存取應用程式、輸入行星名稱，然後按下 Enter 鍵。

   Eclipse 會在 `searchPlanet`方法中停止應用程式。現在您可以進行檢查。

清除資源

如果您不再需要為此教學課程建立的資源，請將其刪除，以免產生額外費用。請完成下列步驟：

- 如果 Planets 應用程式仍在執行，請將其停止。
- 刪除您在 中建立的資料庫步驟 1：建立資料庫。如需詳細資訊，請參閱刪除資料庫執行個體。

AWS 藍年常見問答集

一般

1. Blu AWS Age Refactoring 功能的主要目的是什麼？

   重構功能使用現代分散式應用程式，使用現代語言和架構，遵循自動化重構模式，將舊版單體程式碼重構為 Java。此模式涉及自動分析舊版程式碼、了解其功能，並將其轉換為同等的現代程式碼，同時保留商業邏輯。此程序不僅包括程式碼的現代化，還包括使用自動化工具和程序的整個應用程式堆疊、相依性和基礎設施。解決方案旨在加速現代化，同時保持功能對等性和效能。這包括轉換應用程式程式碼和相關聯的資料庫和資料存放區，同時實作雲端最佳實務和設計模式。

2. AWS Blu Age 支援哪些大型主機應用程式？

   AWS Blu Age 目前支援以 COBOL、PL/I、JCL （工作控制語言） 撰寫的 IBM z/OS 大型主機應用程式現代化，並依賴 CICS （客戶資訊控制系統） 交易管理員、BMS （基本映射支援） 畫面、IMS MFS Screens、DB2 資料庫、IMS 資料庫、平面檔案、GDG （產生資料群組） 和 VSAM （虛擬儲存存取方法） 資料檔案。如需詳細資訊，請參閱 AWS Blu Insights。

3. AWS Blu Age 可以現代化哪些大型主機語言？

   AWS Blu Age 將 COBOL 和 PL/I 程式碼轉換為 Java、JCLs 轉換為 Groovy、將畫面 (BMS 或 Citrix) 轉換為 HTML （使用 Sass) 和 JavaScript （角應用程式 – 目前不支援 React)，從而將舊版大型主機應用程式現代化為雲端原生架構。這些技術是為其廣泛採用、強大的生態系統和雲端原生功能而選擇。Angular 提供現代、回應靈敏的使用者介面層，可取代舊版的綠色螢幕界面。它可以建立動態、易於使用的 Web 應用程式，這些應用程式可以在不同的裝置和平台上存取。其元件型架構支援可維護且可擴展的前端開發。轉換會產生遵循現代架構模式和最佳實務的分散式應用程式。

4. AWS Blu Age 如何平衡傳統限制條件與雲端優勢？

   AWS Blu Age 可保留重要的商業邏輯和功能，同時引進雲端原生功能，進而達到平衡。它可確保現代化應用程式維持必要的傳統商業邏輯，同時利用雲端可擴展性、敏捷性和現代操作實務。此方法可協助組織維持業務連續性，同時獲得雲端基礎設施的優勢。

5. 服務導向架構在現代化應用程式中扮演什麼角色？

服務導向架構在將單體應用程式分解為更易於管理的模組化元件方面扮演了基本角色。 AWS Blu Age 會建立服務導向和物件導向的應用程式，以促進更好的可維護性和可擴展性。這種架構方法可讓組織實現更高的業務效率，並為未來採用潛在的微服務做好準備。

6. 重構程序中包含應用程式堆疊的哪些層面？

   重構程序包括完整的軟體堆疊：應用程式程式碼、相依性、資料庫和基礎設施（例如快取、訊息支援等選項）。它涵蓋傳統程式設計語言、資料庫系統、資料檔案和相關基礎設施元件的轉換。此全方位方法可確保應用程式的所有層面都一致地進行現代化，進而產生完全轉換的現代應用程式堆疊。

7. AWS Blu Age 現代化程序是否免除了對現代化 Java 應用程式進行任何測試或品質保證檢查的需求？

   否， AWS Blu Age 現代化程序不會消除對現代化 Java 應用程式進行任何測試或品質保證檢查的需求。

8. Blu AWS Age JAC 代表什麼？

   JAC 代表 JICS 管理主控台

9. 如何存取 AWS Blu Age 工具？

   AWS 透過 AWS Mainframe Modernization (M2) Refactor，即可透過 AWS 主控台存取 Blu Age 工具，並根據您的認證層級進行功能存取。從轉換中心開始，評估原始程式碼的自動 Java 重構。如需詳細指引，請參閱 AWS Blu Insights 文件。現代化之後，您可以使用受管或非受管執行時間選項來部署應用程式。如需這些部署選擇的詳細資訊，請參閱 AWS Mainframe Modernization 文件。

10. 如何調整專案的大小（工作負載和時間軸）？

    如需詳細資訊，請參閱 AWS Blu Insights 估算或與您的客戶經理合作。

11. 維護 Java AWS Blu Age 遷移解決方案是否有特定要求？

    否，維護 Java AWS Blu Age 遷移解決方案沒有特定要求。

12. Blu AWS Age 產生的程式碼的技術規格和相容性為何？

    AWS Blu Age 產生的程式碼具有特定的技術特性和廣泛的相容性。雖然它不支援 JPA，但它使用直接 SQL 執行搭配外部查詢。此程式碼依賴於執行時間特定的程式庫，以進行功能等效性、Web 服務產生和 MQ 實作。產生的程式碼可以匯入到任何 Java IDE 以進行開發、測試、建置和部署，但必須相應地匯入必要的程式庫。雖然 Maven 預設會與用於建置程序的 AWS Mainframe Modernization 服務整合，但在轉換後修改封裝格式，即可使用 Gradle 等替代工具。該平台在開發工具和來源控

制方面提供了靈活性，並提供培訓供開發團隊管理程式碼。如需詳細資訊，請參閱 AWS Blu Age Runtime 高階架構。

AWS Blu Age 執行時間

1. 在哪裡可以找到 AWS 藍年執行期的相關資訊？

   請參閱設定 AWS Blu Age Runtime （非受管） 文件，其中說明非受管執行期，詳細說明設定程序加入、擷取成品、部署等。

2. 哪裡可以找到開發人員的 AWS Blu Age 執行期？

   開發人員的 AWS Blu Age Runtime 可在 L3 認證個人的 Blu Age Toolbox 中取得。

3. 是否將 AWS Blu Age JAR 相依性上傳至用戶端的 Maven 儲存庫以進行本機開發？

   程式庫可以使用可用於設定開發、測試和生產環境的 AMI 匯入 EC2。訓練和啟用將提供給團隊，以維護/增強產生的應用程式程式碼。如需詳細資訊，請參閱 AWS Blu Age Runtime 高階架構。

4. 在分散式 AWS Blu Age Runtime jar 中，「Gapwalk」一詞是指什麼？

   如需 Gapwalk 的資訊，請參閱AWS 藍時執行期成品。

5. 如何請求存取非受管的 AWS Blu Age 執行期？

   遵循加入 AWS 藍卡年齡執行期的指示，請求存取 AWS 支援 中心。

6. Blu AWS Age 重構應用程式的支援執行期為何？

   若要探索現代化應用程式的完整執行時間選項範圍，建議您檢閱 Blu Age 執行時間選項指南。

7. 何時使用 AWS Blu Age 執行期？

   若要支援執行 AWS Blu Age 重構應用程式，必須使用 AWS Blu Age 執行期。在測試重構應用程式的以 AWS 藍本年齡為基礎的重構專案期間，執行時間是必要的。重構專案完成後，在生產環境中維護、測試和執行 AWS Blu Age 重構應用程式也需要執行時間。

8. 如何為 AWS Blu Age Runtime AWS 發佈新版本？

   對於 M2 受管執行期，包括修補程式、次要和主要版本在內的更新可在 AWS 主控台和 中使用 AWS CLI。其中包括作業系統更新、引擎和相依性變更，通常在一般可用性的 30 天內。 AWS 負責支援的元件，並自動將更新套用至 AWS Mainframe Modernization 執行個體。其他環境也一樣，例如自訂執行期、Linux AMI 和內部部署。

9. AWS Blu Age 執行期的新主要和次要版本發佈的頻率為何？

新版本每月發行一次或兩個版本，客戶可以決定何時及如何升級其執行時間執行個體。如需詳細資訊，請參閱AWS 「藍時版本控制」頁面。

10如何 AWS 支援 AWS Blu Age Runtime？

透過 提供支援 AWS 支援，其中透過提出票證解決問題，並適用標準 SLA。如需詳細資訊，請參閱AWS 大型主機現代化元件生命週期。

11AWS Mainframe Modernization AWS Blu Age Runtime 需要什麼？

AWS Blu Age Runtime 包含工具箱程式庫，用於加速現代化、促進雲端整合，以及改善程式碼品質和可維護性。它還透過促進傳統架構和雲端架構之間的轉換來實現更現代化的自動化。執行時間支援使用 Java 慣用詞處理舊版動詞和資料結構記憶體表示。它允許根據物件導向程式設計技術建置現代化應用程式，並能夠重現舊版控制流程。它使用 Amazon Aurora 等關聯式資料庫，將舊版 VSAM 資料集或 IMS 階層式資料庫支援現代化。它為舊版系統公用程式 (IDCAMS、ISBGENER、DFSORT 等） 和舊版交易管理系統 (CICS、IMS) 提供 Java 替換。它有助於雲端與 Amazon ElastiCache 中的快取整合，並支援 AWS 簡訊解決方案 (SQS、Kinesis)。

12AWS Blu Age Runtime 是否支援非 x86 電腦架構？

目前， AWS Blu Age Runtime 僅支援 x86 型電腦架構和運算。 AWS Blu Age Runtime 不支援 ARM 型和 Graviton 型運算。

13客戶如何隨時了解 AWS Blu Age Runtime 版本，包括新版本的通知，以及版本歷史記錄和版本備註的存取權？

新版的 AWS Blu Age Runtime 會上傳至我們的官方發行頁面。我們建議您定期檢查此頁面，最好每 3 個月檢查一次，以取得最新版本和更新。關於版本歷史記錄和版本備註的存取，可用性取決於每個主要版本的end-of-life(EOL) 日期。如需 EOL 日期、版本升級規劃和歷史資訊存取權的詳細資訊，請參閱AWS 「藍期生命週期」。

14Blu AWS Age Runtime 高階架構的主要元件有哪些？

AWS Blu Age Runtime 架構包含兩種主要元件類型。首先是存放在共用資料夾 （ 可存取應用程式伺服器類別載入器） 中的 Java 程式庫 (jar 檔案） ，提供舊版建構和陳述式支援。其次是包含 Spring 型應用程式的 Web 應用程式 (war 檔案） ，這些應用程式為現代化程式提供架構和服務。執行時間也包括：收集所有叫用和跨程式呼叫的程式登錄檔，以及收集所有現代化任務指令碼的指令碼登錄檔。這些元件共同運作，為現代化應用程式提供統一的 REST 型進入點和執行架構。執行期和現代化應用程式會一起部署在應用程式伺服器 （ 例如 Tomcat) 中。

15如何設定存放 AWS Blu Age Runtime 成品的共用資料夾？

AWS Blu Age Runtime 成品 (jar) 必須收集在共用資料夾中，應用程式伺服器類別載入器可存取。對於 tomcat 伺服器，透過修改名為 catalina.properties 的一般組態檔案來進行組態。例如，如果您在 tomcat 資料夾中將共用資料夾建立為名為 "shared" 的資料夾，則需要修改 catalina.properties 中的 common.loader 項目，讓 tomcat 類別載入器可存取共用資料夾，例如：

```
common.loader="${catalina.base}/lib","${catalina.base}/lib/*.jar","${catalina.home}/
lib","${catalina.home}/lib/*.jar","${catalina.home}/shared","${catalina.home}/shared/
*.jar"
```

16.AWS Blu Age 執行期如何處理無狀態和工作階段管理？

AWS Blu Age Runtime 透過多種機制實作無狀態和工作階段管理。對於 HTTP 工作階段，它會針對使用者內容使用 Cookie 型識別與外部快取儲存。工作階段可以存放在各種資料存放區中，包括 Amazon ElastiCache、Redis 叢集或記憶體內映射。無狀態設計可確保大多數非暫時性狀態存放在外部的常見「單一事實來源」中，實現高可用性和水平擴展。此方法結合負載平衡和共用工作階段，允許跨多個節點分發使用者面向的對話方塊。

17.Web 應用程式在 AWS Blu Age Runtime 環境中扮演什麼角色？

AWS Blu Age Runtime 中的 Web 應用程式提供多個關鍵函數。它們提供可重現舊版環境和交易監控的執行架構（例如 JCL 批次、CICS、IMS)。它們透過 提供以 REST 為基礎的進入點`gapwalk-application.war`，用於觸發和控制交易、程式和批次。此外，它們提供 OS 提供的程式和專用「驅動程式」程式的模擬，傳統應用程式依賴這些程式來存取 IMS 資料庫或使用者對話方塊等服務，這些程式可透過MFS 存取。

18.如何在 AWS Blu Age Runtime 中註冊和管理程式？

AWS Blu Age Runtime 中的程式是透過在伺服器啟動期間填入的 ProgramRegistry 系統註冊。每個程式都會實作程式界面，並標示為 Spring 元件。程式會使用其識別符註冊，如果程式有多個識別符，則可能會有多個項目。註冊程序是自動的，並記錄在 Tomcat 日誌中。ProgramRegistry 可讓其他程式和指令碼尋找和呼叫已註冊的程式，維持現代化系統的模組化和互連性。

19.如何管理 AWS Blu Age Runtime 應用程式中的組態？

AWS Blu Age Runtime 中的組態是透過 YAML 檔案使用 Spring Boot 架構功能來管理。使用兩個主要組態檔案：application-main.yml 用於架構組態，以及`application-profile.yml`用戶端特定的選項。系統遵循 Spring 的優先順序邏輯，允許透過各種方式覆寫組態。其他組態可透過 JNDI 為資料庫和命令列參數提供，提供組態管理的彈性。記錄器組態是使用日誌傳回 xml 組態檔案完成。

20.秘密管理員在 AWS Blu Age Runtime 組態中扮演什麼角色？

使用 AWS Blu Age Runtime 中的秘密管理員保護敏感組態資料，例如資料庫登入資料和 Redis 快取密碼。它們允許在 AWS 秘密中儲存關鍵資料，並在 YAML 組態檔案中參考它們。系統支援不同類型的秘密，包括自動填入所有相關欄位的資料庫秘密，以及受密碼保護資源的單一密碼秘密。此方法透過將敏感資料與應用程式組態分開來增強安全性。

21開發人員如何撰寫自己的程式與 AWS Blu Age Runtime 相容？

開發人員可以透過實作 程式界面並遵循特定模式，建立與 AWS Blu Age Runtime 相容的程式。程式必須宣告為 Spring 元件、實作必要的方法，並在 ProgramRegistry 中正確註冊。開發人員需要建立配套內容和組態類別、處理程式識別符，並確保與 Spring 架構適當整合。實作應遵循程式結構和執行的 AWS 藍時執行期慣例。

22AWS Blu Age Runtime 如何處理程式執行錯誤？

AWS Blu Age Runtime 透過多種機制處理程式執行錯誤。對於批次任務，它會擷取任務執行詳細資訊中的執行狀態、結束代碼和詳細錯誤資訊。錯誤處理包括特定的結束代碼（技術錯誤為 -1、服務程式失敗為 -2），以及 Tomcat 日誌中的詳細記錄。系統可設定為在執行時間例外狀況時轉返交易，並提供錯誤通知和復原的選項。錯誤詳細資訊可透過 REST 端點存取，以進行監控和疑難排解。

23批次任務可使用哪些 AWS 藍本存留期執行期監控功能？

AWS Blu Age Runtime 透過各種端點為批次任務提供監控功能。它會追蹤任務執行狀態、開始/結束時間、執行模式和詳細結果。系統提供端點，用於列出觸發的指令碼、擷取任務執行詳細資訊，以及監控目前執行中的任務。指標的端點提供 JVM 統計資料、工作階段計數和詳細的批次執行指標。平台也支援監控資料的分頁和以時間為基礎的篩選。

24如何追蹤和管理 AWS Blu Age Runtime 任務執行狀態？

任務執行狀態會透過完整的狀態系統進行追蹤，其中包括 DONE、TRIGGERED、RUNNING、KILLED 和 FAILED 等狀態。每個任務執行都會收到唯一識別符，用於追蹤和維護詳細的執行資訊，包括開始時間、結束時間、發起人資訊和執行結果。系統提供 REST 端點，用於查詢任務狀態、管理執行中的任務，以及擷取執行歷史記錄。狀態資訊會保留在伺服器記憶體中，可根據資源管理的存留期進行清除。

25AWS Blu Age Runtime 如何處理外部系統互動？

執行時間會透過各種機制處理外部系統互動，包括服務整合的 REST 端點、訊息佇列 (SQS、RabbitMQ、IBM MQ) 的支援，以及資料庫連線選項。它透過專業元件提供傳統系統互動的模擬，支援 SSL/TLS 進行安全通訊，並包含處理外部檔案系統的功能。系統也支援與外部身分驗證提供者整合，並可設定為與各種第三方服務互動。

26AWS 藍年執行期如何處理身分驗證？

AWS Blu Age Runtime 支援多種身分驗證方法，其中 OAuth2 是主要機制。它可以與 Amazon Cognito 或 Keycloak 等身分提供者整合。身分驗證組態是透過名為 application-main.yml 的主要組態檔案進行管理，其中可以定義安全設定、身分提供者和身分驗證方法。系統支援 XSS 保護、CORS、CSRF 等功能，並可針對全域安全性和特定端點安全性進行設定。針對開發，也提供具有預設超級管理員登入資料的本機身分驗證系統。

27.Blu AWS Age Runtime 如何確保高可用性？

AWS Blu Age Runtime 透過多種機制確保高可用性。它透過將非暫時性狀態存放在外部共用儲存體中來實作無狀態，讓多個應用程式執行個體可以一起運作。系統支援負載平衡和共用工作階段，允許將請求分散到多個節點。對於資料儲存，它可以利用高可用性的資料庫和快取系統。架構支援自動容錯移轉，並且可以部署到多個可用區域，以提高可靠性。

28.使用何種元件來重現具有 AWS Blu Age 應用程式的 CICS 分散式交易？

AWS Blu Age Runtime 提供專用端點，允許在全域交易 (XA 支援） 中調用現有的 JICS 交易。基礎的兩個階段遞交支援依賴於 Atomikos 軟體元件。

29.用於定義特定程式行為的類別的 AWS Blu Age 名稱是什麼？

每個程式都繫結至專用組態類別，允許 指定程式特定的行為。如需命名和位置慣例的詳細資訊，請參閱AWS 現代化應用程式的 Blu Age 結構

30.哪個編碼有下列字元序列順序：空格、小寫字元、大寫字母、數字？

屬於 EBCDIC 變體系列的 Charset （例如 CP1047, CP297等）。

31.如何操作 AWS Blu Age 受管執行期？

使用 AWS Management Console AWS CLI、 或 AWS APIs

32.Blu AWS Age Runtime 的定價維度為何？

AWS Mainframe Modernization-core-hours （請參閱 AWS Mainframe Modernization 定價)。

33.將原始資料透過 HTTP 傳遞至程式端點的機制為何？

Base64 編碼字串。

34.使用者如何啟動批次任務執行？

對其中一個專用批次端點使用 HTTP 呼叫 （請參閱批次端點文件頁面)。

35.哪個 AWS Blu Age Runtime 端點是主要 Web 前端應用程式的主要進入點？

```
/transaction
```

36Blu AWS Age JICS 代表什麼？

AWS Blu Age JICS 是用於支援 CICS 資源現代化的執行期元件。資源定義會存放在專用資料存放區中。若要管理它們，請使用 REST API 或 JICS 應用程式主控台。如需詳細資訊，請參閱管理 AWS 藍本中的 JICS 應用程式主控台。

37.有哪些可用的 AWS Blu Age Runtime 快取機制？

AWS Blu Age Runtime 支援多個快取機制，包括 Redis 和 EhCache。Redis 建議用於生產環境，跨多個節點提供共用的持久性快取。EhCache 適用於具有內嵌揮發性本機快取的獨立部署。系統支援各種元件的快取，包括 Blusam 資料、工作階段資訊、JICS 資源和暫時儲存佇列。快取組態可以針對不同的使用案例和效能需求進行自訂。

38.如何估計 AWS 大型主機現代化 AWS 藍本時代執行期部署的價格？

AWS 會根據客戶的需求和目標架構，向客戶提供預估值。

39AWS Mainframe Modernization AWS Blu Age Runtime 價格是多少？

AWS Mainframe Modernization 為 AWS Blu Age 提供兩種定價模式：受管執行期選項，包括執行期、運算資源、內部儲存和自動化，以及僅涵蓋 AWS Blu Age 執行期本身的非受管執行期選項。對於 AWS 部署，兩者都使用pay-as-you-go定價結構。如需up-to-date且詳細的定價資訊，建議您參閱官方 AWS Mainframe Modernization 定價頁面。

40.如果我們需要在支援的執行時間中未列出的基礎設施上部署 AWS Blu Age 重構應用程式，該怎麼辦？

如果您需要在支援的執行時間中未列出的基礎設施上部署 AWS Blu Age 重構應用程式，有幾個選項可用。首先，檢查您的基礎設施是否與現有的部署選項相容，例如 Amazon EKS Anywhere 或其他容器協同運作平台。若是如此，您可能可以使用 AWS Blu Age 執行期 （非受管）。對於不相容的基礎設施，我們建議諮詢 AWS 大型主機專家，以探索自訂解決方案或潛在的調整。您也可以提交產品功能請求 (PFR) 以取得擴展的基礎設施支援。替代帳單選項可能適用於非標準部署。請聯絡您的 AWS 代表，以討論您的特定需求和最適合您環境的方法。

41.如何授權 AWS Blu Age Runtime？ 是否為開放原始碼？

AWS Blu Age Runtime 不是開放原始碼。它是以雲端原生服務形式分佈的 AWS IP。有兩種部署選項：

a. AWS Blu Age Managed，執行時間會部署到專用的 AWS 受管服務，利用預先設定且已準備好部署環境，無需設定或管理。

b. AWS Blu Age 非受管，可根據 Amazon EC2 或 Amazon ECS/ 部署到您自己的自訂 AWS 架構 AWS Fargate，您必須自行佈建和設定。這兩個選項都會產生執行期費用，這些費用包含在提供給您的專案預估中。由於這是具有 支援 存取權的受管服務，因此您不需要原始碼。如需定價的詳細資訊，請參閱AWS 大型主機現代化定價頁面。

42如何對 AWS Blu Age 架構和程式庫進行變更和升級管理？

AWS Blu Age 架構和程式庫會透過定期程式碼產生和部署程序進行更新。這些更新會做為 AWS 大型主機現代化生命週期的一部分進行管理，其中包括版本升級和 AWS Blu Age 團隊或認證合作夥伴的支援。如需版本控制、升級程序和支援時間表的詳細資訊，請參閱 AWS Mainframe Modernization 生命週期文件。

資料

1. 關於舊版資料庫的現代化，現代化應用程式有哪些可用的資料庫選項？

   現代化應用程式可以使用多種現代資料庫選項，包括：PostgreSQL、Amazon Aurora、RDS for PostgreSQL、Oracle 資料庫、MS-SQL 和 IBM Db2。這些選項可提供根據特定需求選擇最適當資料庫系統的彈性，同時利用現代資料庫管理系統和雲端原生功能的優點。

2. IBM Db2 for z/OS到 Postgres DDL 的轉換涵蓋範圍為何？

   完整轉換（包括資料庫限制）。

3. AWS Blu Age 是否支援群組資料產生 (GDG)？

   是，支援批次使用 GDG，並支援相對和絕對產生和自動清除策略。

4. AWS Blu Age 是否支援串連資料集？

   是，支援批次使用串連資料集。執行串連時，數個資料集可以讀取為單一資料集。請注意，Blusam 資料集不能是串連的一部分。

5. SQL 查詢套用了哪些程序？

   根據目標資料庫，在程式碼轉換期間調整。

6. 如果應用程式有多個資料庫，哪些選項適用？

   為每個查詢設定目標資料庫，並在應用程式和 Apache Tomcat 中定義所有資料庫。

7. 是否可以停用 Blusam？

   是，在主要組態檔案中，不需要資料庫（如需詳細資訊，請參閱 Blusam 組態文件頁面)。

8. 哪個 AWS Blu Age API 用於取代 IMS 資料庫等資料庫？

   JHDB (Java 階層式 DataBase) API。

9. 哪些 AWS Blu Age 產品可用來將舊版資料和資料庫遷移至現代關聯式資料庫管理系統 (RDBMS)？

   AWS Blu Age 資料庫現代化工具 (Data Migrator)。

10. 什麼是 AWS Blu Age Data Simplifier，以及它在現代化中解決了哪些問題？

    Data Simplifier 是 AWS Blu Age 的核心程式庫，可解決在 Java 中處理舊版記憶體存取模式的挑戰。它提供的建構模組支援低階記憶體存取、傳統資料類型（例如分區、封裝、英數字元）和混合結構化/原始記憶體存取，這些類型在大型主機應用程式中很常見，但在 Java 中原生不可用。程式庫透過熟悉的 Java 模式公開這些功能，例如 getter/setters 和類別型 APIs，讓 Java 開發人員可以存取這些功能，同時維護舊版功能。

11. AWS Blu Age 如何處理舊版記憶體配置和資料結構？

    AWS Blu Age 透過 Record 界面處理舊版記憶體配置，該界面提供固定大小的位元組陣列抽象。對於 COBOL '01 資料項目' 等結構化資料，它會使用在現代化期間自動產生的 RecordEntity 子類別。這些類別會維護舊版資料的階層結構，每個元素都有父子關係。系統同時支援原始記憶體存取和結構化存取模式，保留舊式系統的彈性，同時提供現代化的程式設計界面。

12. AWS Blu Age 如何處理 VSAM 資料集現代化？

    Blusam 元件提供 VSAM 資料集現代化支援，包括專用 API、端點和管理 Web 應用程式 (BAC：Blusam 管理主控台)。Blusam 依賴關聯式資料庫做為後端 (PostgreSQL，使用 RDS 或 Aurora)。

轉換

1. 我是否可以找到轉換程序的詳細資訊？

   請參閱 AWS Blu Insights 文件。

2. Blu AWS Age 產生的模組名稱是什麼？

   服務、實體、Web 和工具。

3. 為什麼 Java/Spring 被選為 AWS 藍年的目標技術之一？

   Java/Spring 因其廣泛採用、大型人才集區和強大的企業功能而被選為目標技術。Java 生態系統提供廣泛的程式庫、架構和工具，可支援現代應用程式開發。Spring 架構提供企業級功能、雲端原生功能，並遵循產業最佳實務，使其成為現代化應用程式的理想選擇。

4. 包含 AWS Blu Age 產生的模組之父專案的名稱是什麼？

   父專案的名稱尾碼為「-pom」，並且可以使用名為專案的 Transform 屬性在 Transformation Center 中定義。

5. 如果提供， AWS Blu Age 如何管理舊版排程器現代化？

   舊版排程器資產並未透過 AWS Blu Age 進行現代化。評估階段會考慮這些成品，以協助識別可能的遺失成品。

6. 使用 AWS Blu Age 對產生的程式碼進行偵錯的需求是什麼？

   支援 Java 的任何整合式開發環境 (IDE)，例如 Eclipse、JetBrain 或 VisualCode。

部署

1. 有哪些環境可用來部署具有 AWS Blu Age 的現代化應用程式？

   Windows Server、Linux 伺服器和 Docker Linux 容器。

2. AWS Blu Age 重構應用程式是否可以在任何基礎設施上執行？

   雖然 AWS Blu Age 重構應用程式並非設計在任何基礎設施上執行，但它們在部署選項方面提供很大的靈活性。這些應用程式可以部署在各種運算平台上，包括雲端受管服務、無伺服器運算和內部部署基礎設施。 AWS Blu Age 同時提供受管和非受管執行期選項，讓組織根據其特定需求和需求，在全受管便利性和自訂控制之間進行選擇。這種靈活性可讓您輕鬆跨支援的基礎設施移動，使 AWS Blu Age 重構的應用程式高度適應不同的部署環境。如需詳細資訊，請參閱 AWS Blu Age 執行期選項文件。

3. AWS Blu Age 支援哪些 MQ 組態？

   SQS、IBM WebSphere MQ。

4. 使用者可以使用 AWS Mainframe Modernization 非受管執行時間，將 Java 商業應用程式邏輯部署到哪些應用程式伺服器？

   Apache Tomcat，大於或等於 10.1 的版本。

5. 重構應用程式如何與其他 整合， AWS 服務 例如 Amazon Aurora？

   現代化應用程式 AWS 服務 透過支援轉換至雲端原生資料庫解決方案，例如 Amazon Aurora 和 RDS for PostgreSQL 來整合 與 。 AWS Blu Age 可確保現代化應用程式與 之間的整合 AWS 服務，讓組織能夠使用雲端功能。此整合可延伸至 AWS 生態系統內的資料儲存和應用程式服務。除了資料庫儲存之外， AWS Blu Age Runtime 還與各種 整合， AWS 服務 包括 Amazon ElastiCache

for Redis 快取、 AWS Secrets Manager 用於組態管理和用於部署 AWS 的大型主機現代化。它支援 Amazon EC2、Amazon EKS、由 Fargate 管理的 ECS 進行容器部署。系統可以使用 AWS Identity and Access Management 進行身分驗證、使用 Amazon Simple Storage Service 進行儲存，並支援 AWS 服務 透過組態和服務連接器與其他 整合。

6. 重構應用程式如何確保符合可擴展性要求？

   解決方案可將應用程式轉換為可使用 AWS 彈性基礎設施的雲端原生架構，以確保可擴展性。它實作現代設計模式和最佳實務，以實現水平和垂直擴展。服務導向方法允許獨立擴展元件。現代化應用程式可以利用雲端服務的固有可擴展性功能。

7. 原始程式碼重構完成後會發生什麼情況？

   在原始程式碼重構之後，會發生兩個主要步驟。首先，建置重構的應用程式。其次，應用程式會在 AWS Mainframe Modernization AWS Blu Age Runtime 中部署和監控。部署可以在透過自動化管理基礎設施的 AWS受管環境 (AWS 大型主機現代化受管執行期） 中完成，或在客戶管理自己的基礎設施的 AWS 帳戶 (AWS 大型主機現代化 AWS 藍本時代非受管執行期) 中完成。非受管選項可以部署在各種平台上，包括 Amazon EC2、EC2 上的 EC2或 EC2 上的 Fargate、EKSEC2。

8. 如何在自訂 Amazon Linux AMI 上部署和執行以 AWS Blu Age 進行現代化的應用程式，而不使用 AWS Mainframe Modernization (M2) 受管服務？

   這可以透過在 Amazon EC2 上使用 AWS Blu Age Runtime （非受管） 部署應用程式來實現。此程序涉及建立相依於 AWS Blu Age Runtime 程式庫的 Java/Spring 應用程式，並將其部署在自訂 Amazon Linux AMI 上。如需此方法的詳細說明，請參閱在 Amazon EC2 上設定 AWS Blu Age Runtime （非受管）。

9. 是否有可用的 Amazon Machine Image (AMI)？ 是否有可用的 Docker 映像？

   - AMI：否，由於客戶需要自訂和設定其環境，因此沒有可用的 AMI。客戶可以擷取 AWS Blu Age 成品，並根據其需求設定執行個體。
   - Docker 映像：否，沒有可用的立即可用的 Docker 映像，但容器上的設定 AWS Blu Age Runtime 頁面說明如何根據 AWS Blu Age Runtime 二進位檔建置和部署您自己的 Docker 映像，並將其部署到適當的容器管理系統。

10. 客戶是否可以封裝並執行 AWS Blu Age 應用程式做為 Docker 容器？

    M2 受管執行期是不可能的，但它適用於基於 Amazon Linux AMI 的客戶定義環境，以及內部部署或其他雲端供應商。

11. 如果我想要縮小範圍，如何知道執行非受管 AWS Blu Age 所需的 SQS 政策資源 ARN？

若要判斷特定 SQS 政策資源 ARN，以執行非以範圍縮減政策管理的 AWS Blu Age，請洽詢交付團隊或技術客戶經理 (TAM)。他們可以提供帳戶特定的指引。如需 SQS 政策的一般資訊，請參閱 AWS SQS 政策文件。

12.任務排程如何使用批次？

它與 Control-M/Stone 分支或任何其他分散式排程器整合。

使用 Rocket Software （ 先前為 Micro Focus) 來複寫應用程式

本指南涵蓋使用 AWS Mainframe Modernization 解決方案來修改大型主機應用程式的end-to-end程序 AWS。它描述了所有任務，並包含有關在 Amazon EC2 上設定和操作 AWS 大型主機現代化執行期的資訊，從初始設定和分析到建置、測試和部署現代化應用程式 AWS。它還涵蓋進階主題，例如使用傳統資料結構、使用範本和預先定義的專案，以及設定串流工作階段的自動化。

主題

- 設定 Rocket Software （ 先前為 Micro Focus)（ 在 Amazon EC2 上 ）
- 設定 Automation for Rocket Enterprise Analyzer （ 先前稱為 Micro Focus) 和 Rocket Enterprise 開發人員串流工作階段
- 在 Rocket Enterprise Developer （ 先前為 Micro Focus Enterprise Developer) 中將資料集檢視為資料表和資料欄
- 在企業開發人員中使用 Rocket Software （ 先前稱為 Micro Focus) 資料檔案工具編輯資料集
- Rocket Software （ 先前為 Micro Focus) 的教學課程
- AWS Mainframe Modernization 中的可用批次公用程式

設定 Rocket Software （ 先前為 Micro Focus)（ 在 Amazon EC2 上 ）

AWS Mainframe Modernization 提供數個 Amazon Machine Image AMIs)，其中包含 Rocket Software （ 先前稱為 Micro Focus) 授權產品。這些 AMIs 可讓您快速佈建 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體，以支援您控制和管理的 Rocket Software 環境。本主題提供存取和啟動這些 AMIs 所需的步驟。使用這些 AMIs 是完全選用的，不需要完成本使用者指南中的教學課程。

主題

- 設定 Rocket Software （ 先前為 Micro Focus)（ 在 Amazon EC2 上 ） 的先決條件
- 建立 Amazon S3 的 Amazon VPC 端點
- 請求帳戶的允許清單更新
- 建立 AWS Identity and Access Management 角色
- 授予 License Manager 所需的許可

- 訂閱 Amazon Machine Image
- 啟動 AWS Mainframe Modernization Rocket Software（先前稱為 Micro Focus) 執行個體
- 沒有網際網路存取的子網路或 VPC

設定 Rocket Software（先前為 Micro Focus)（在 Amazon EC2 上）的先決條件

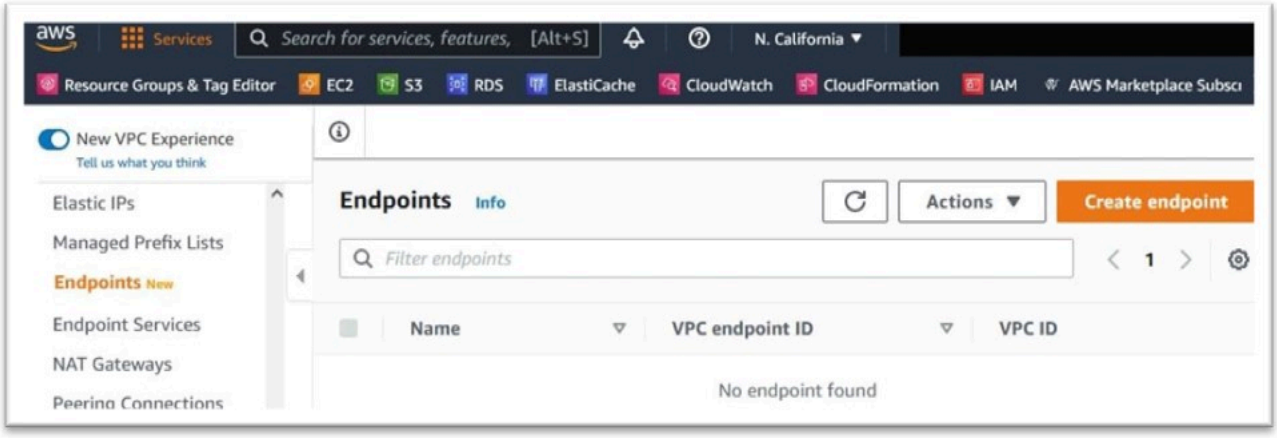
當您設定 Rocket Software（在 Amazon EC2 上）時，請確定您符合下列先決條件。

- 管理員存取將建立 Amazon EC2 執行個體的帳戶。
- 識別要建立 Amazon EC2 執行個體的 AWS 區域 ，並確認 AWS Mainframe Modernization 服務可用。請參閱AWS 依區域的服務。請務必選擇提供服務的區域。
- 識別要建立 Amazon EC2 執行個體的 Amazon Virtual Private Cloud (Amazon VPC)。 Amazon EC2

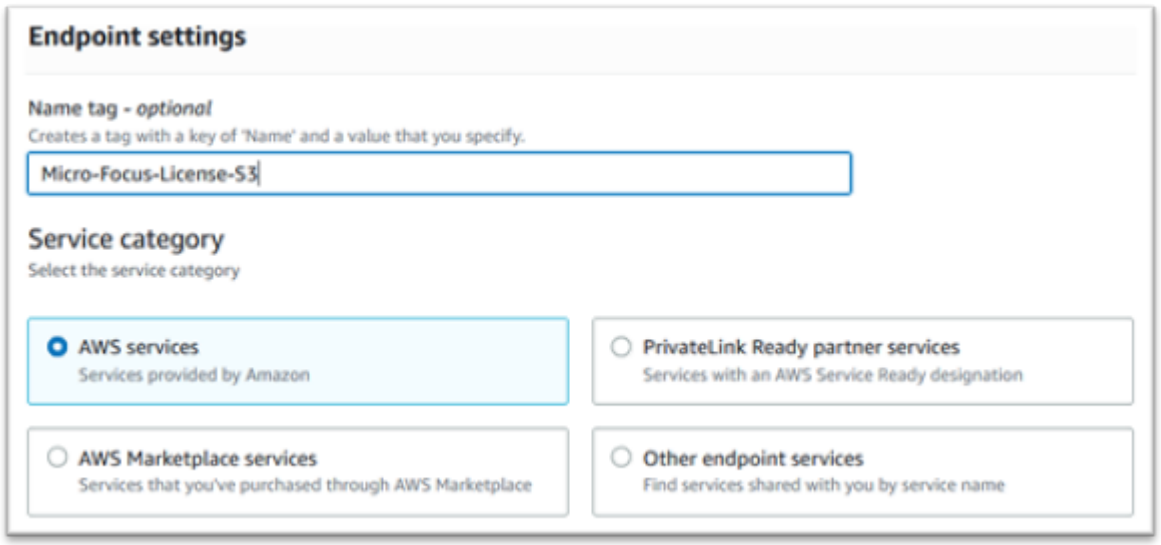
建立 Amazon S3 的 Amazon VPC 端點

在本節中，您會建立 Amazon S3 要使用的 Amazon VPC 端點。稍後設定此端點將協助您設定 VPC 的網際網路存取。

1. 在 中導覽至 Amazon VPC AWS Management Console。
2. 在導覽窗格中選擇端點。
3. 選擇建立端點。



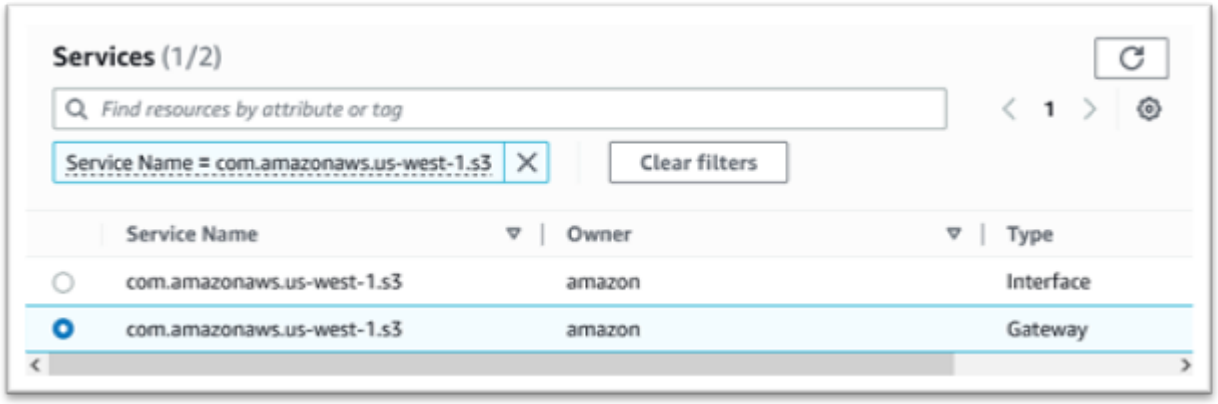
4. 輸入有意義的名稱標籤，例如「Micro-Focus-License-S3」。
5. 選擇 AWS 服務做為服務類別。



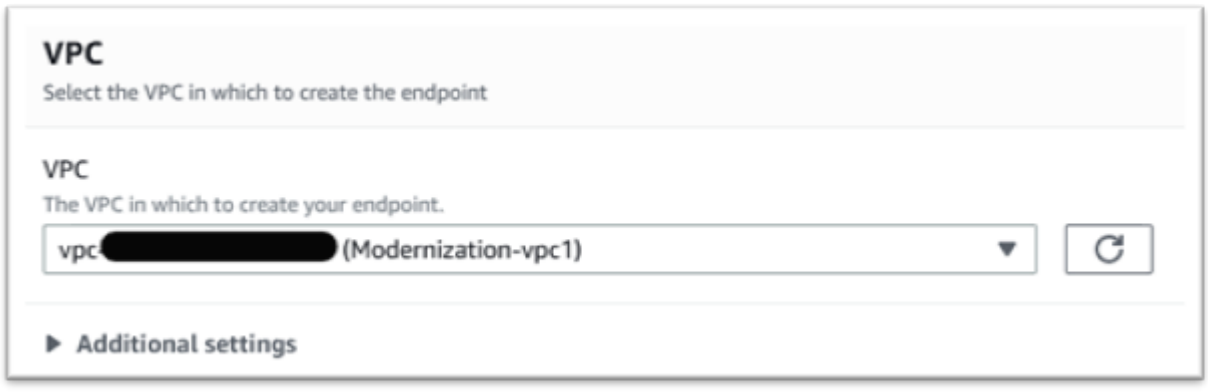
6. 在服務下搜尋 Amazon S3 Gateway 服務：com.amazonaws.【region】.s3。

   us-west-1 為此：com.amazonaws.us-west-1.s3。

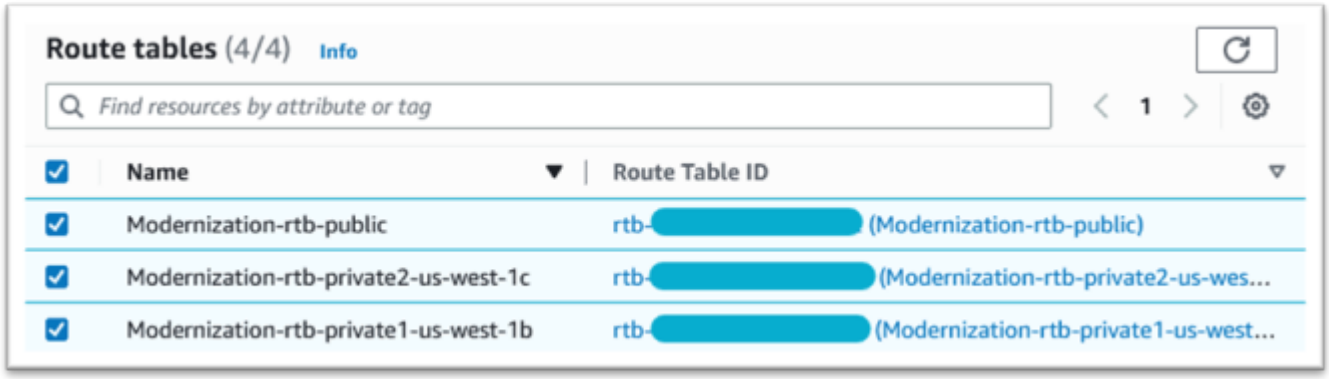
7. 選擇閘道服務。



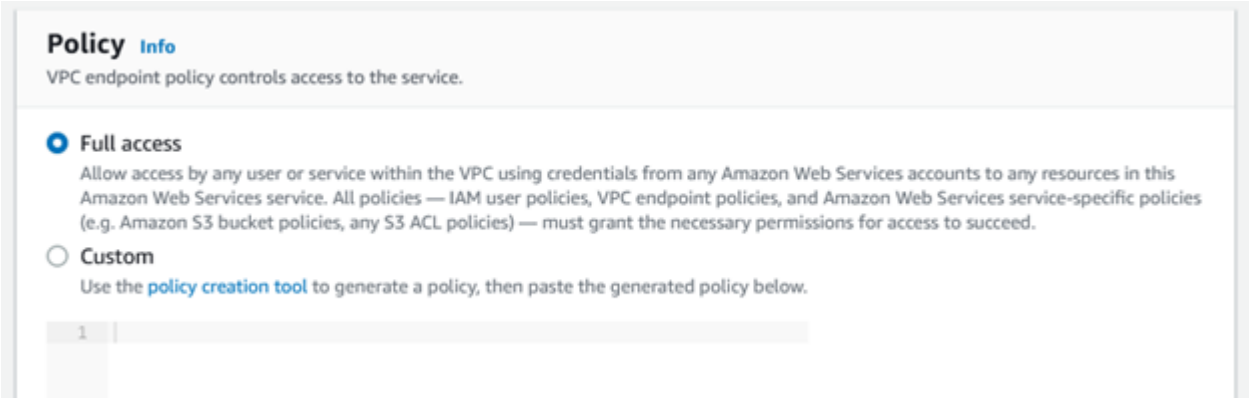
8. 針對 VPC，選擇您將使用的 VPC。



9. 選擇 VPC 的所有路由表。



10. 在政策下，選擇完整存取。



> **Note**
>
> 如果您決定建立自訂政策，請確定它有權存取 Amazon S3 儲存貯體 `s3://aws-supernova-marketplace-<region>-prod`。

11. 選擇建立端點。

請求帳戶的允許清單更新

與您的 AWS 代表合作，讓您的帳戶列入 AWS Mainframe Modernization AMIs的允許清單。請提供下列資訊：

- AWS 帳戶 ID。
- 建立 Amazon VPC 端點 AWS 區域 的 。
- 在 中建立的 Amazon VPC Amazon S3 端點 ID建立 Amazon S3 的 Amazon VPC 端點。這是 com.amazonaws.【region】.s3 Gateway `vpce-xxxxxxxxxxxxxxxxx` 端點的 ID。
- 所有 Rocket Software Enterprise Suite AMI Amazon EC2 執行個體所需的授權數量。

每個 CPU 核心需要一個授權 （對於大多數 Amazon EC2 執行個體，每 2 vCPUs)。

如需詳細資訊，請參閱最佳化 CPU 選項。

未來可以調整請求的號碼 AWS。

> **Note**
>
> 請聯絡您的 AWS 代表，或將代表您開啟允許清單請求的支援票證 AWS 支援 的人員。您無法直接請求，請求可能需要幾天才能完成。

建立 AWS Identity and Access Management 角色

建立 AWS Identity and Access Management 政策和角色以供 AWS Mainframe Modernization Amazon EC2 執行個體使用。透過 IAM 主控台建立角色會建立相同名稱的相關聯執行個體描述檔。將此執行個體描述檔指派給 Amazon EC2 執行個體允許指派 Rocket 軟體授權。如需執行個體描述檔的詳細資訊，請參閱使用 IAM 角色將許可授予在 Amazon EC2 執行個體上執行的應用程式。

建立 IAM 政策

首先建立 IAM 政策，然後連接到角色。

1. 導覽至 AWS Identity and Access Management 中的 AWS Management Console。
2. 選擇政策，然後選擇建立政策。



3. 選擇 JSON 標籤。

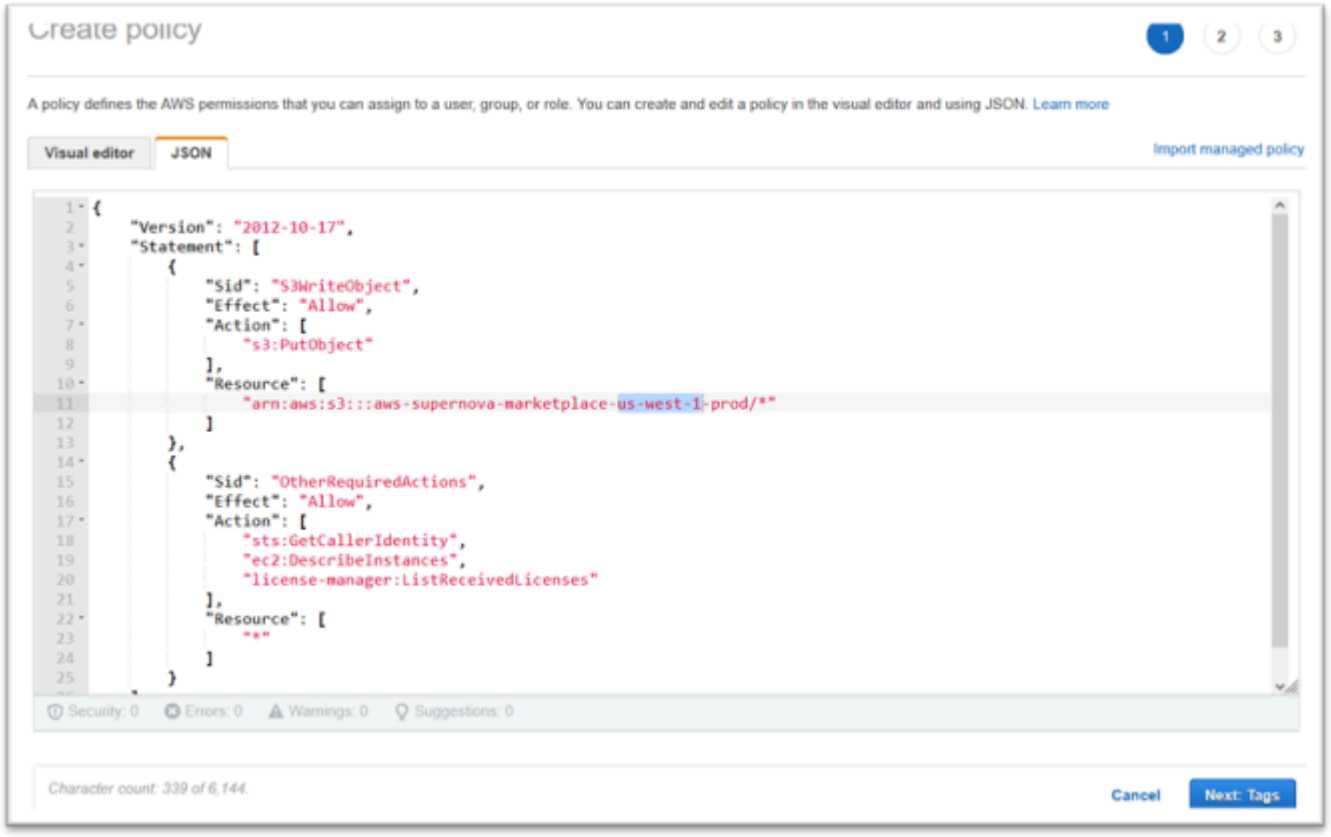


4. 將下列 JSON us-west-1中的 取代為定義 Amazon S3 端點 AWS 區域 的 ，然後將 JSON 複製並貼到政策編輯器中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3WriteObject",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::aws-supernova-marketplace-us-west-1-prod/*"
            ]
        },
        {
            "Sid": "OtherRequiredActions",
            "Effect": "Allow",
            "Action": [
                "sts:GetCallerIdentity",
                "ec2:DescribeInstances",
                "license-manager:ListReceivedLicenses"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
```

```
}
```

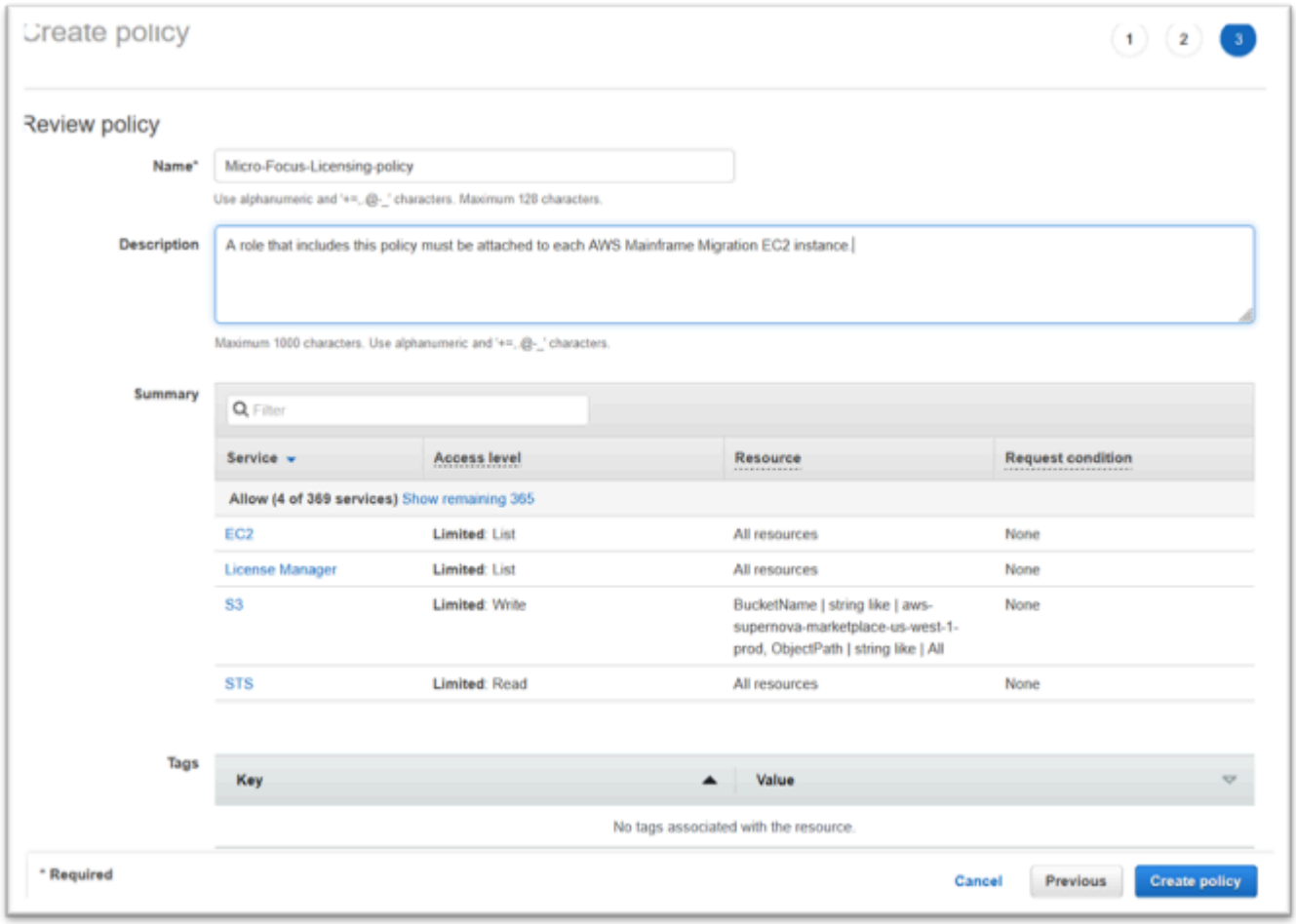
> **Note**
>
> Sid 下的動作`OtherRequiredActions`不支援資源層級許可，且必須在資源元素`*`中指定。



5. 選擇下一步：標籤。



6. 選擇性地輸入任何標籤，然後選擇下一步：檢閱。
7. 輸入政策的名稱，例如「Micro-Focus-Licensing-policy」。選擇性地輸入描述，例如「包含此政策的角色必須連接到每個 AWS Mainframe Modernization Amazon EC2 執行個體。」

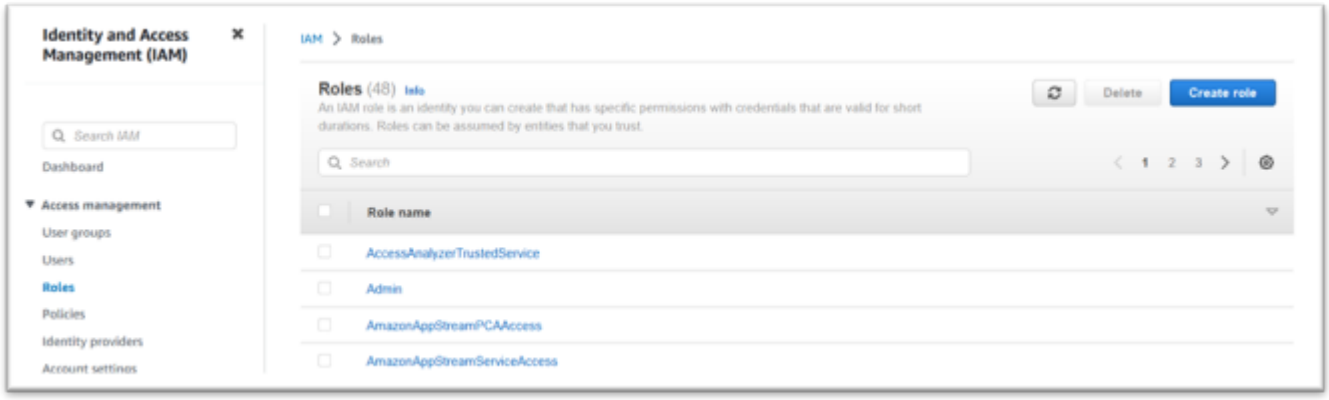


8. 選擇建立政策。

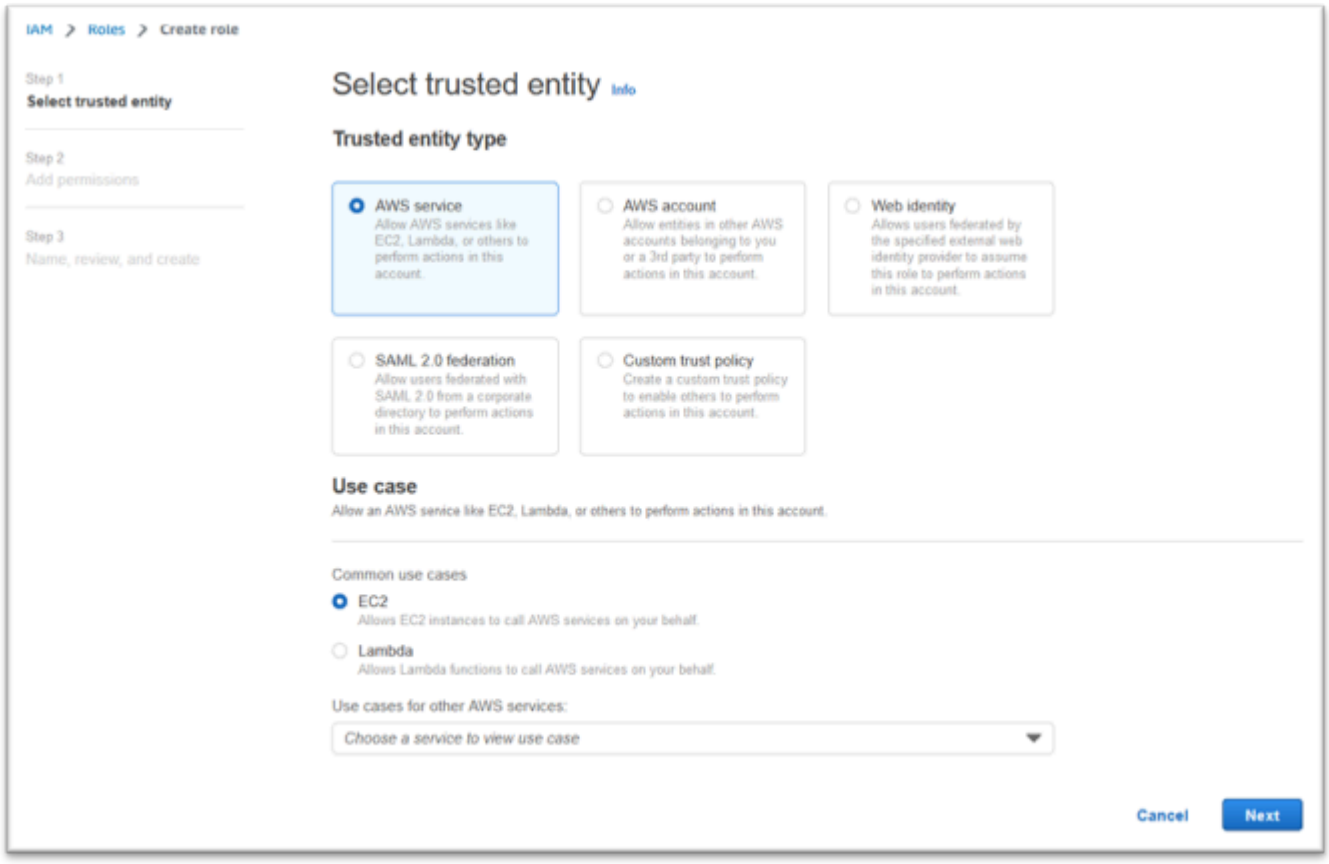
建立 IAM 角色

建立 IAM 政策後，您可以建立 IAM 角色並將其連接至政策。

1. 在 中導覽至 IAM AWS Management Console。
2. 選擇角色，然後選擇建立角色。

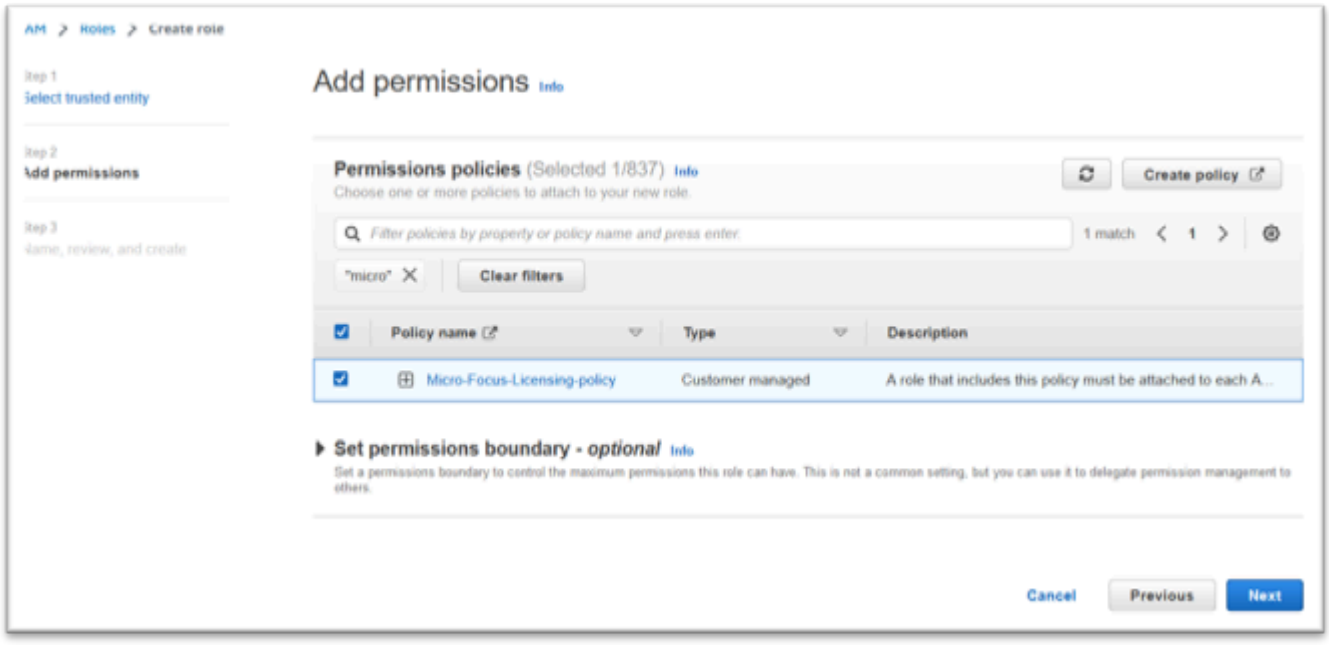


3. 將信任的實體類型保留為AWS 服務，然後選擇 EC2 常用案例。

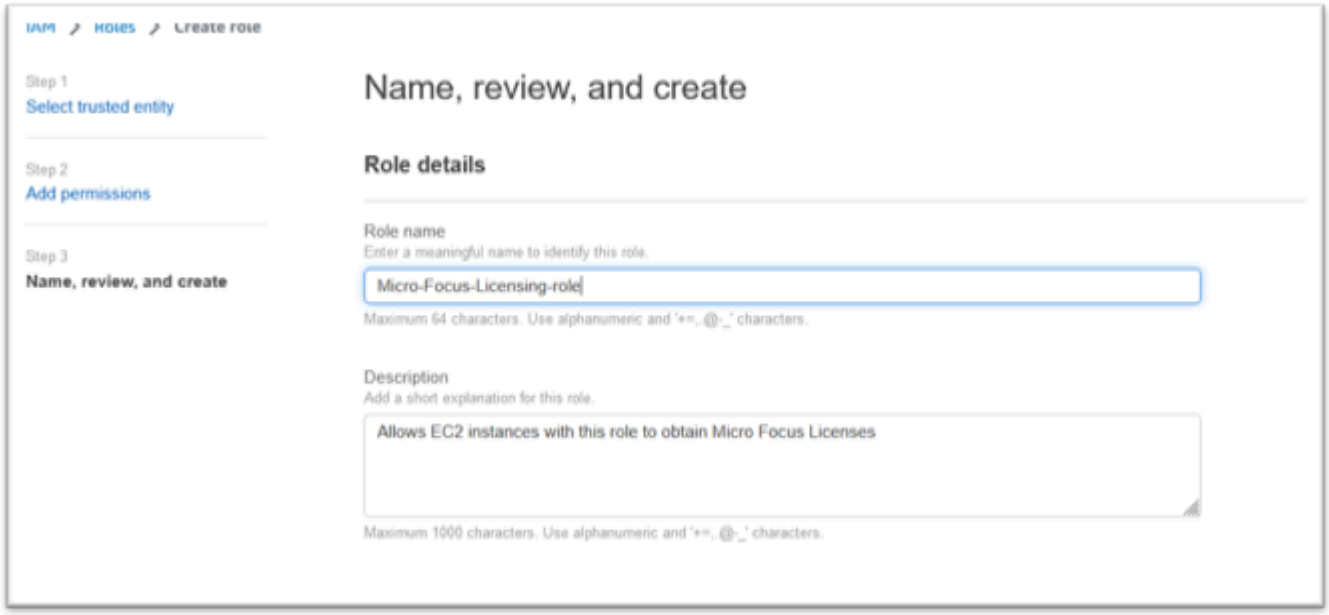


4. 選擇下一步。

5. 在篩選條件中輸入「微」，然後按 Enter 鍵套用篩選條件。
6. 選擇剛建立的政策，例如「Micro-Focus-Licensing-policy」。
7. 選擇下一步。



8. 輸入角色名稱，例如「Micro-Focus-Licensing-role」。
9. 將描述取代為您自己的其中一個，例如「允許使用此角色的 Amazon EC2 執行個體取得 Micro Focus 授權」。

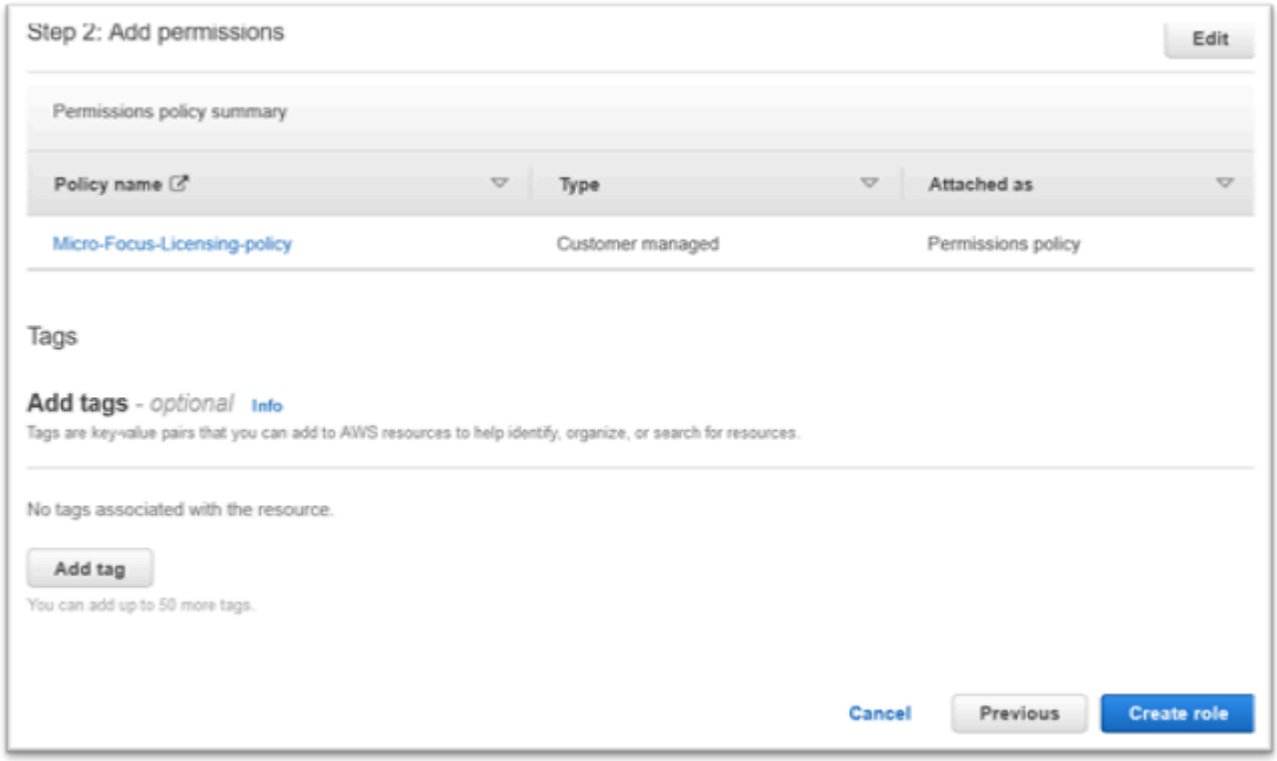


10. 在步驟 1 下：選取信任的實體檢閱 JSON，並確認其具有下列值：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

> **Note**
>
> 效果、動作和委託人的順序並不重要。

11. 確認步驟 2：新增許可顯示您的授權政策。



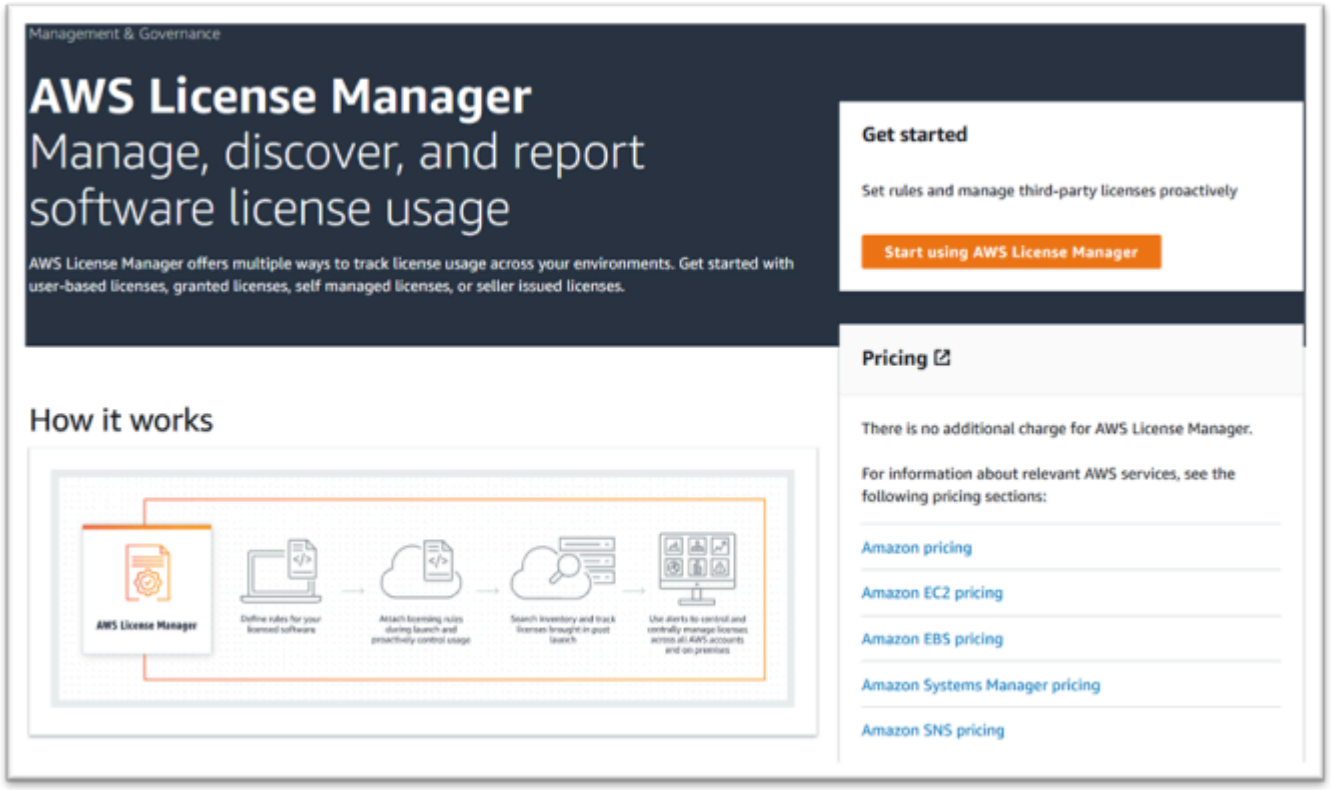
12. 選擇建立角色。

允許清單請求完成後，請繼續執行下列步驟。

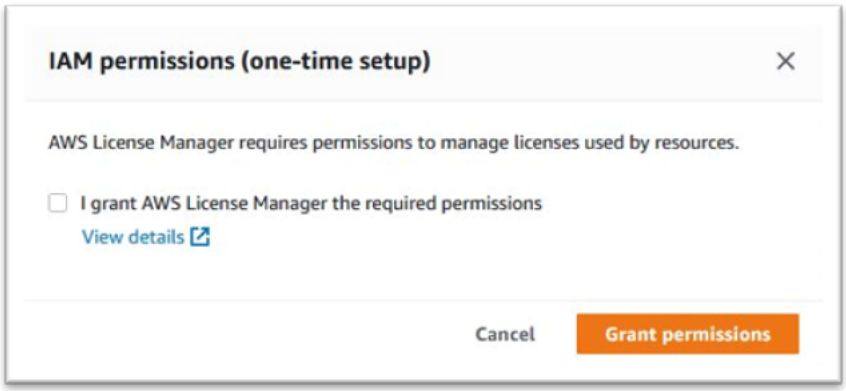
授予 License Manager 所需的許可

您需要授予 許可， AWS License Manager 才能設定 Rocket Software 執行期引擎 （ 在 Amazon EC2 上 ）。

1. 在 AWS License Manager 中導覽至 AWS Management Console。



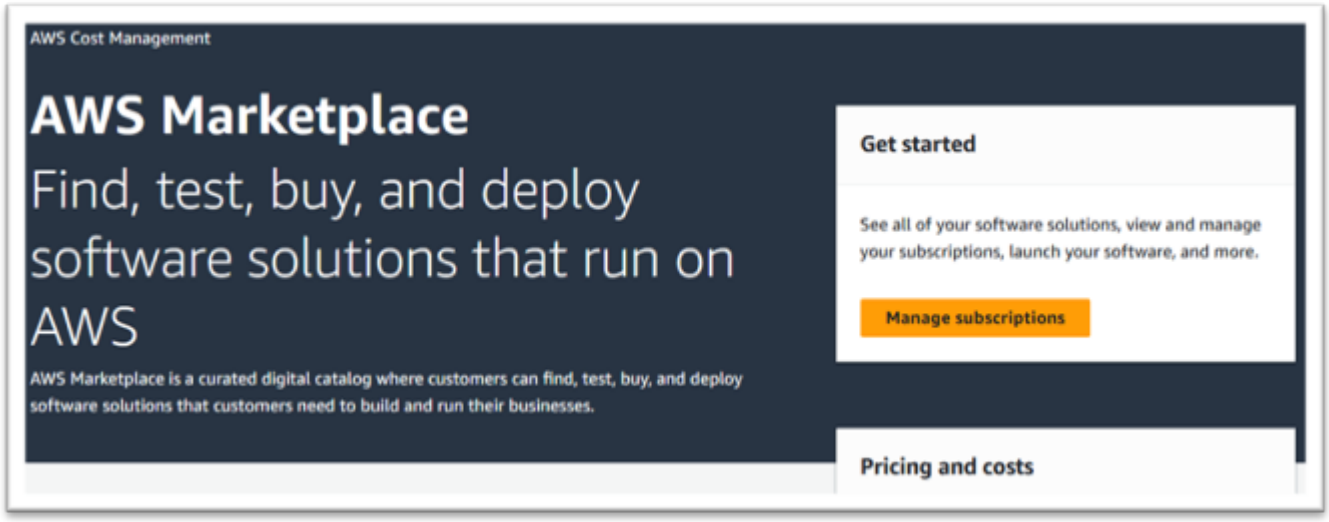
2. 選擇開始使用 AWS License Manager。
3. 如果您看到下列快顯視窗，請檢視詳細資訊，然後選擇核取方塊，然後按授予許可。



訂閱 Amazon Machine Image

訂閱 AWS Marketplace 產品後，您可以從產品的 AMI 啟動執行個體。您也可以在設定 Rocket Software （先前為 Micro Focus) 執行期引擎 （在 Amazon EC2 上） 時管理訂閱的 AMIs。

1. 導覽至 中的 AWS Marketplace 訂閱 AWS Management Console。
2. 選擇 Manage subscriptions (管理訂閱)。



3. 將下列其中一個連結複製並貼到瀏覽器地址列。

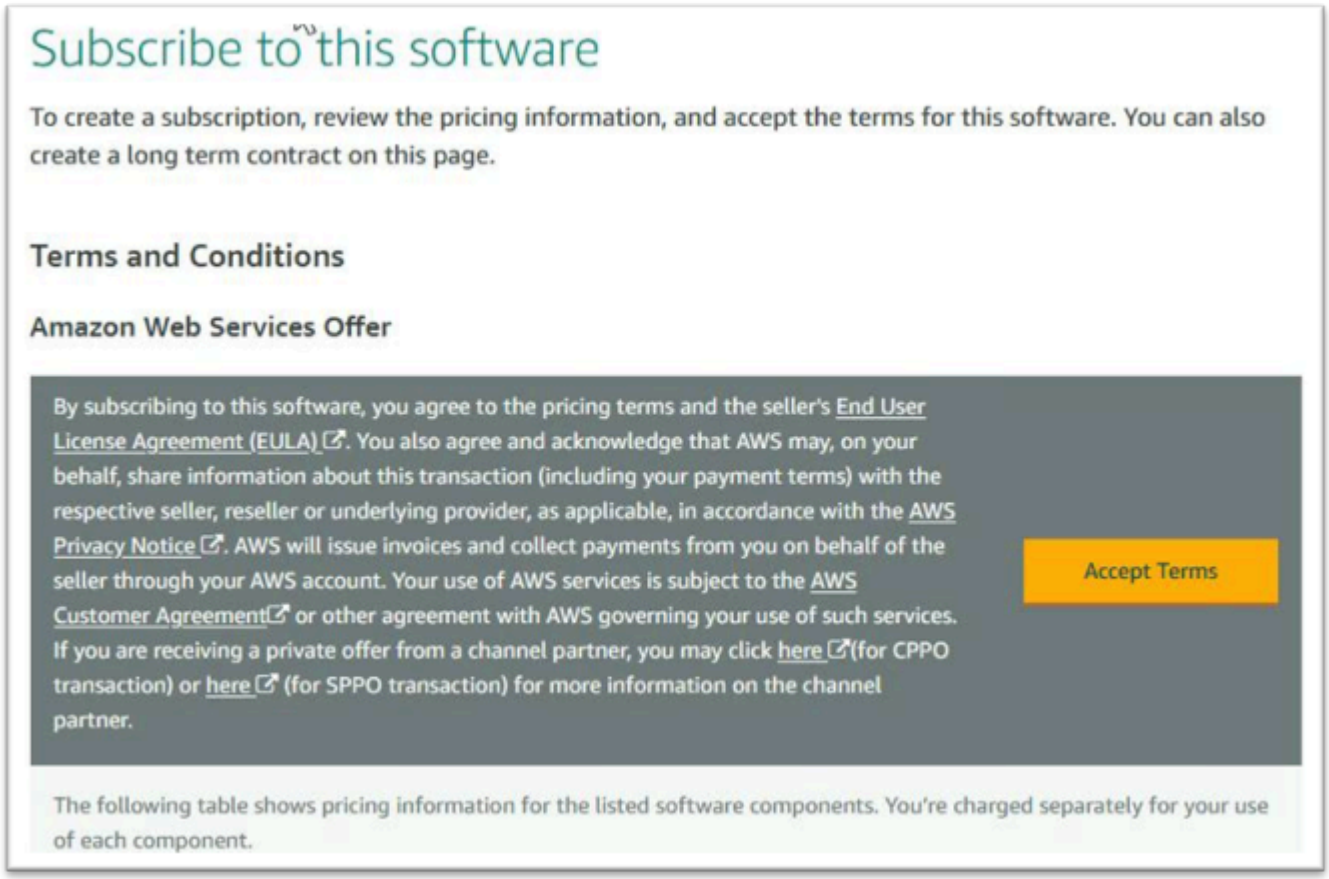
> **Note**
>
> 1. 僅為您獲授權使用的產品之一選擇連結。
> 2. 遵循 請求帳戶的允許清單更新頁面以使用這些連結，確保您的帳戶已列入允許清單。

- 企業伺服器：https：//https://aws.amazon.com/marketplace/pp/prodview-g5emev63l7blc
- 適用於 Windows 的 Enterprise Server：https：//https://aws.amazon.com/marketplace/pp/prodview-lwybsiyikbhc2
- 企業開發人員：https：//https://aws.amazon.com/marketplace/pp/prodview-77qmpr42yzxwk
- 使用 Visual Studio 2022 的企業開發人員： https://aws.amazon.com/marketplace/pp/prodview-m4l3lqiszo6cm
- 企業分析器： https://aws.amazon.com/marketplace/pp/prodview-tttheylcmcihm
- 適用於 Windows 的 Enterprise Build Tools：https：//https://aws.amazon.com/marketplace/pp/prodview-2rw35bbt6uozi
- 企業預存程序：https：//https://aws.amazon.com/marketplace/pp/prodview-zoeyqnsdsj6ha
- 使用 SQL Server 2019 的企業預存程序：https：//https://aws.amazon.com/marketplace/pp/prodview-ynfklquwubnz4

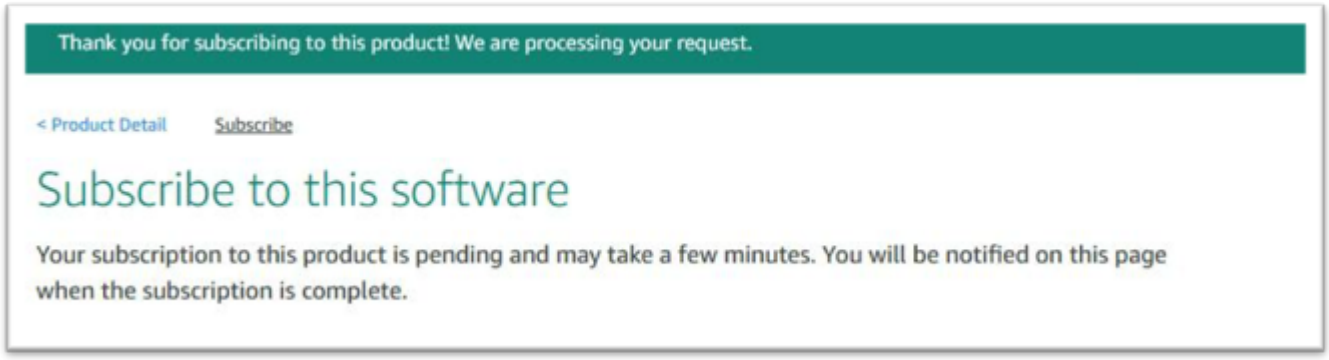
4. 選擇 Continue to Subscribe (繼續以訂閱)。



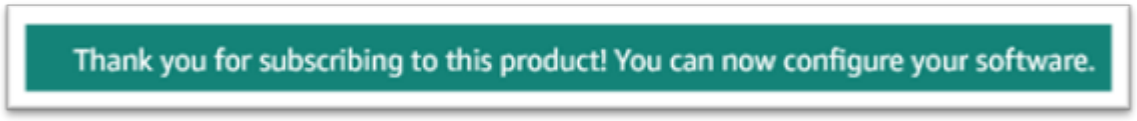
5. 如果條款與條件是可接受的，請選擇接受條款。



6. 訂閱可能需要幾分鐘的時間來處理。

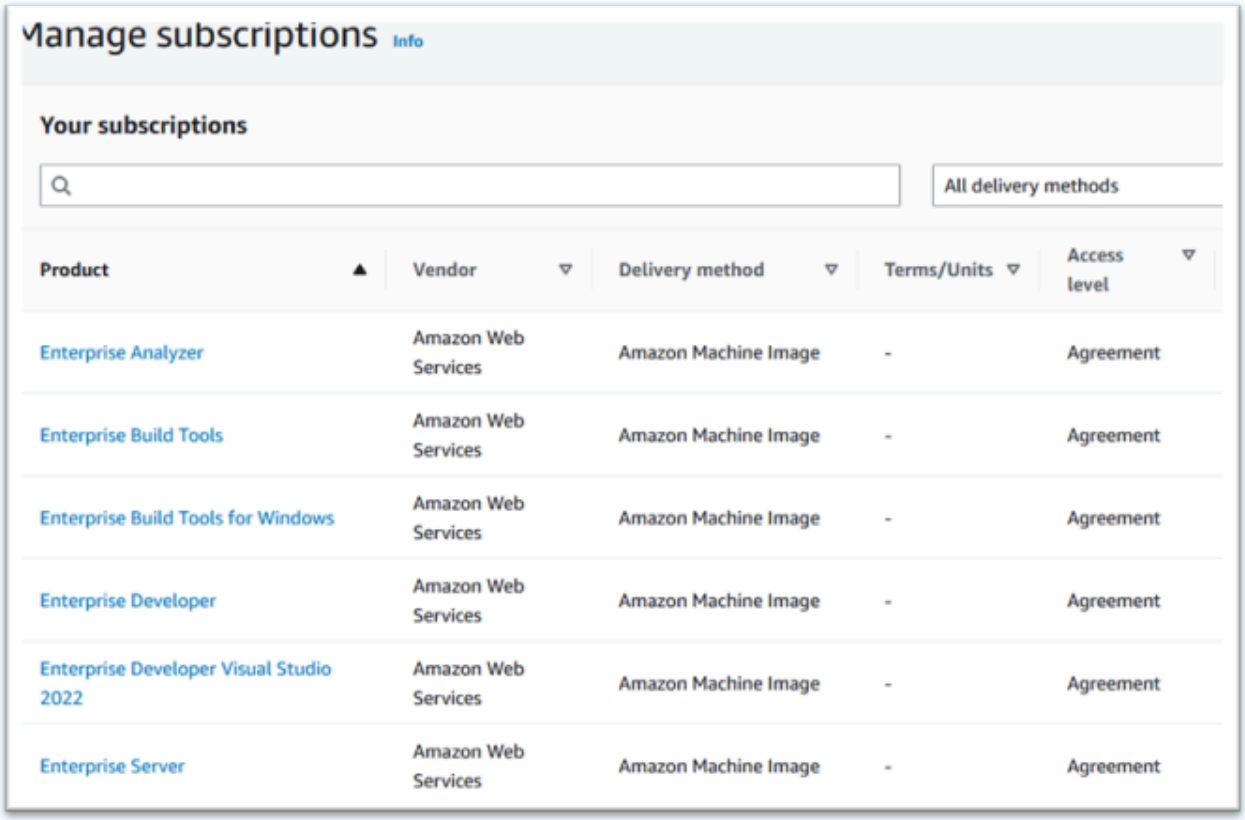


7. 顯示感謝訊息後，複製並貼上步驟 3 的下一個連結，以繼續新增訂閱。



8. 當管理訂閱顯示所有訂閱AMIs 時停止。

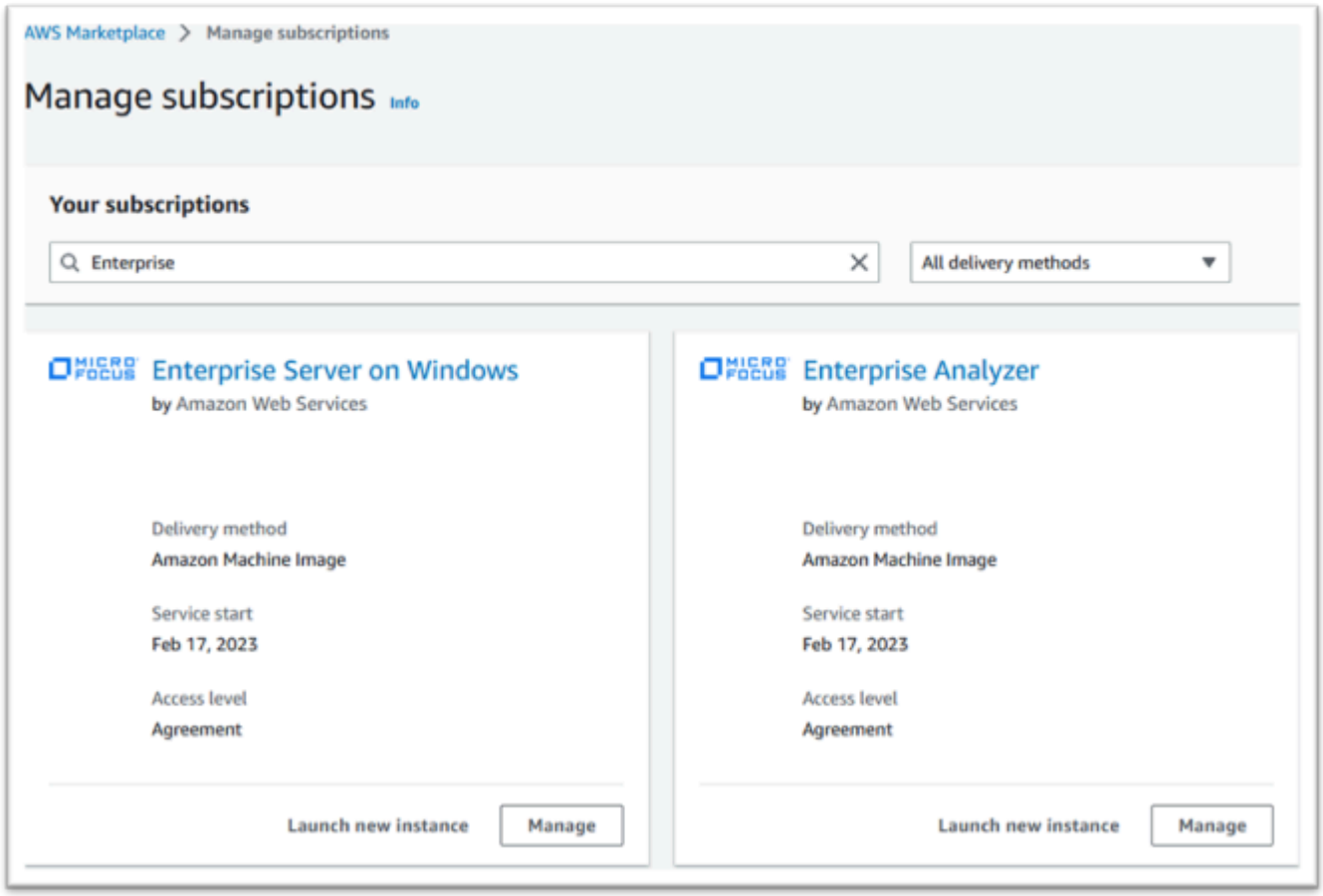
> **Note**
>
> 面板偏好設定（齒輪圖示）設定為將檢視顯示為資料表。



啟動 AWS Mainframe Modernization Rocket Software （ 先前稱為 Micro Focus) 執行個體

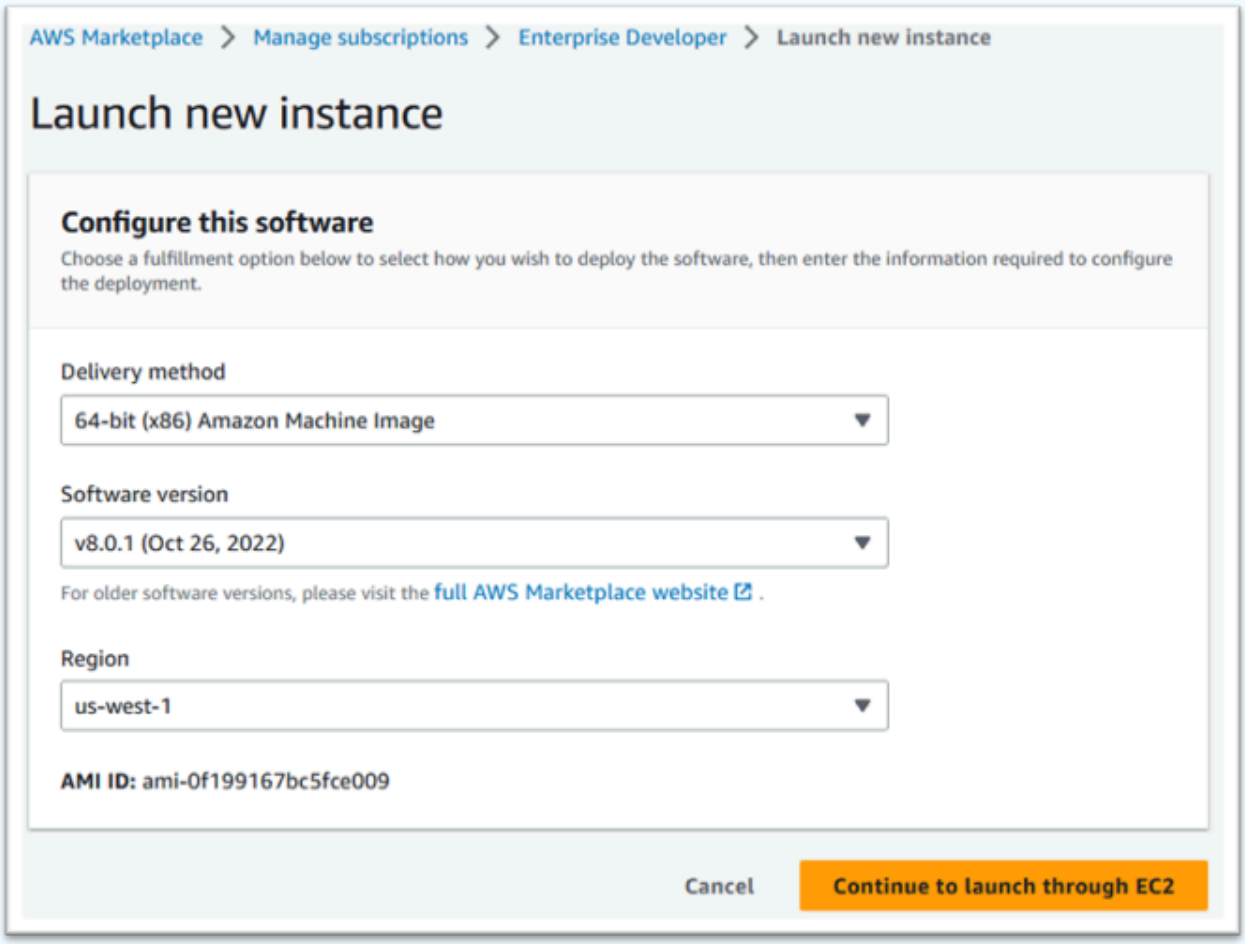
建立端點、IAM 政策、IAM 角色和訂閱 AMIs 之後，您就可以在 中啟動 AWS Mainframe Modernization Rocket Software (Micro Focus) 執行個體 AWS Management Console。

1. 導覽至 中的 AWS Marketplace 訂閱 AWS Management Console。
2. 找到要啟動的 AMI，然後選擇啟動新執行個體。



3. 在啟動新的執行個體對話方塊中，確保已選取允許清單區域。
4. 按下繼續以透過 EC2 啟動。

> **Note**
>
> 下列範例顯示企業開發人員 AMI 的啟動，但所有 AWS Mainframe Modernization AMIs的程序都相同。

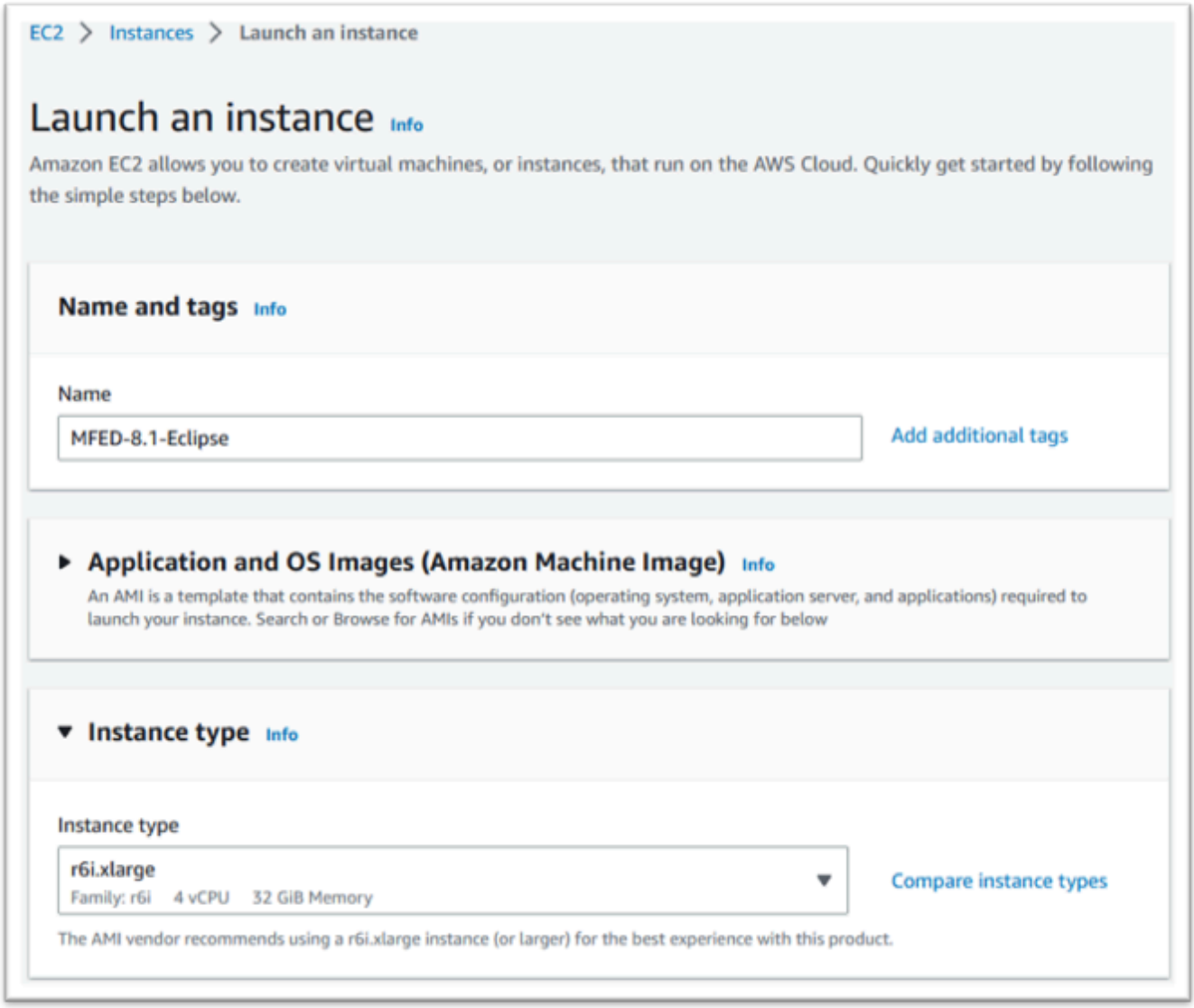


5. 輸入伺服器的名稱。
6. 選擇執行個體類型。

   選取的執行個體類型應取決於專案效能和成本需求。以下是建議的起點：

   - 對於 Enterprise Analyzer，r6i.xlarge
   - 對於企業開發人員，r6i.large
   - 對於獨立 Enterprise Server 執行個體，r6i.xlarge
   - 對於具有向外擴展的 Rocket 軟體效能可用性叢集 (PAC)，r6i.large

   > **Note**
   >
   > 螢幕擷取畫面的應用程式和作業系統映像區段已摺疊。

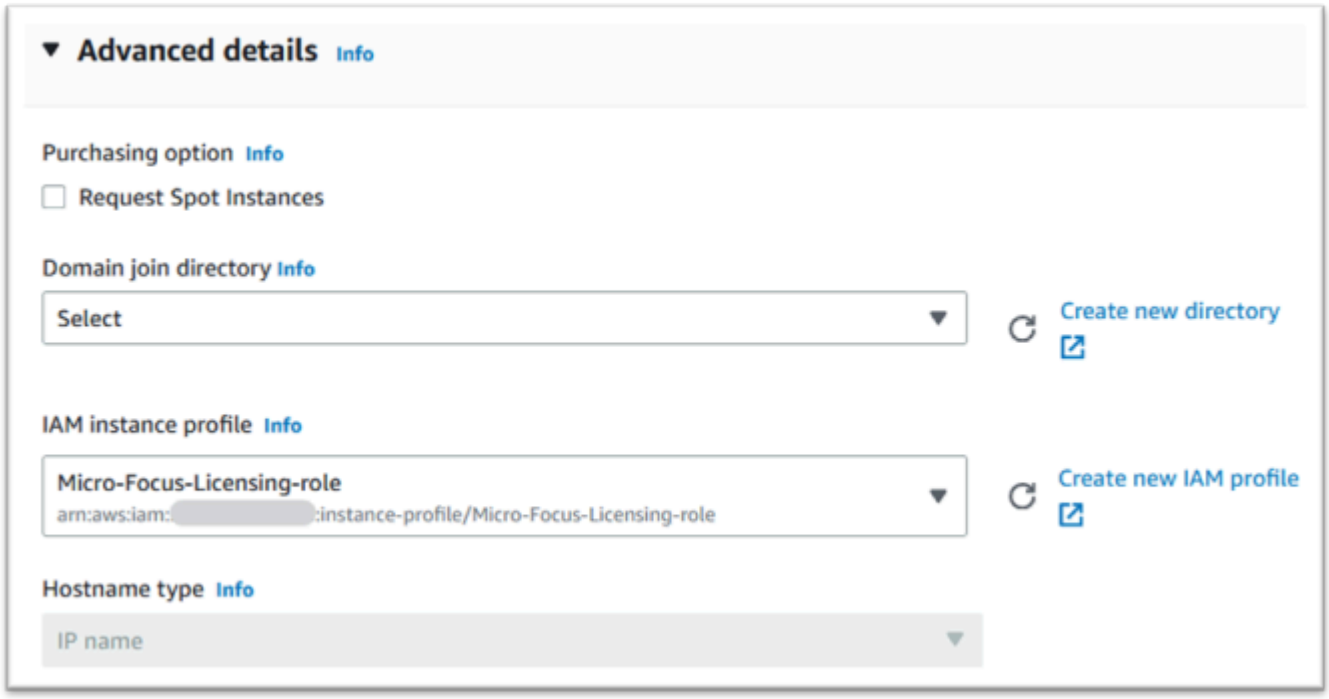


7. 選擇或建立（並儲存）金鑰對（未顯示）。

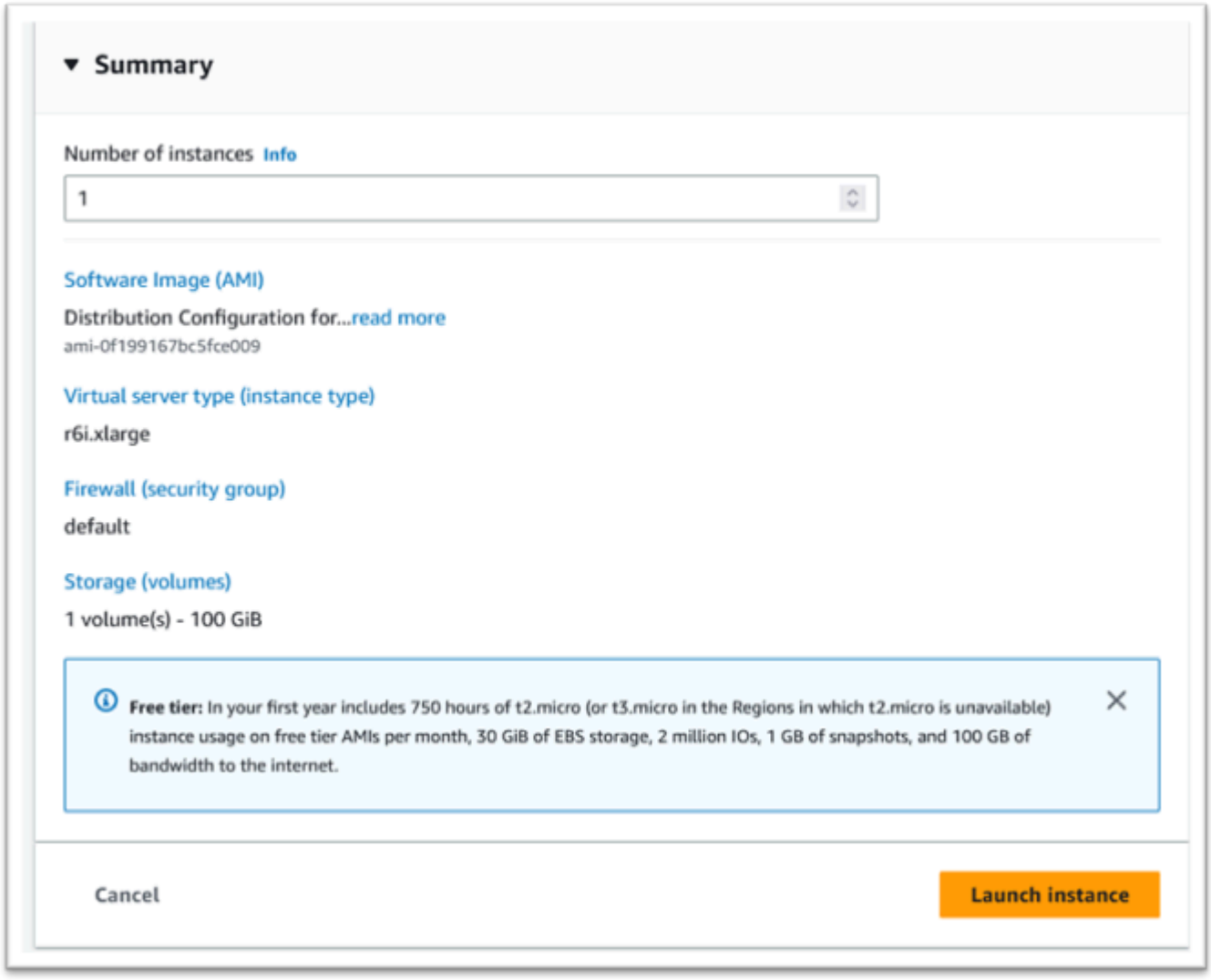
   如需 Linux 執行個體金鑰對的詳細資訊，請參閱 Amazon EC2 金鑰對和 Linux 執行個體。

   如需 Windows 執行個體金鑰對的詳細資訊，請參閱 Amazon EC2 金鑰對和 Windows 執行個體。
8. 編輯網路設定，然後選擇允許清單 VPC 和適當的子網路。
9. 選擇或建立安全群組。如果這是 Enterprise Server EC2 執行個體，通常允許 TCP 流量流向連接埠 86 和 10086 來管理 Rocket 軟體組態。
10. 選擇性地設定 Amazon EC2 執行個體的儲存體。
11. 重要 - 展開進階詳細資訊，並在 IAM 執行個體設定檔下選擇先前建立的授權角色，例如「Micro-Focus-Licensing-role」。

> **Note**
>
> 如果遺漏此步驟，則在建立執行個體之後，您可以從 EC2 執行個體的動作功能表的安全性選項修改 IAM 角色。

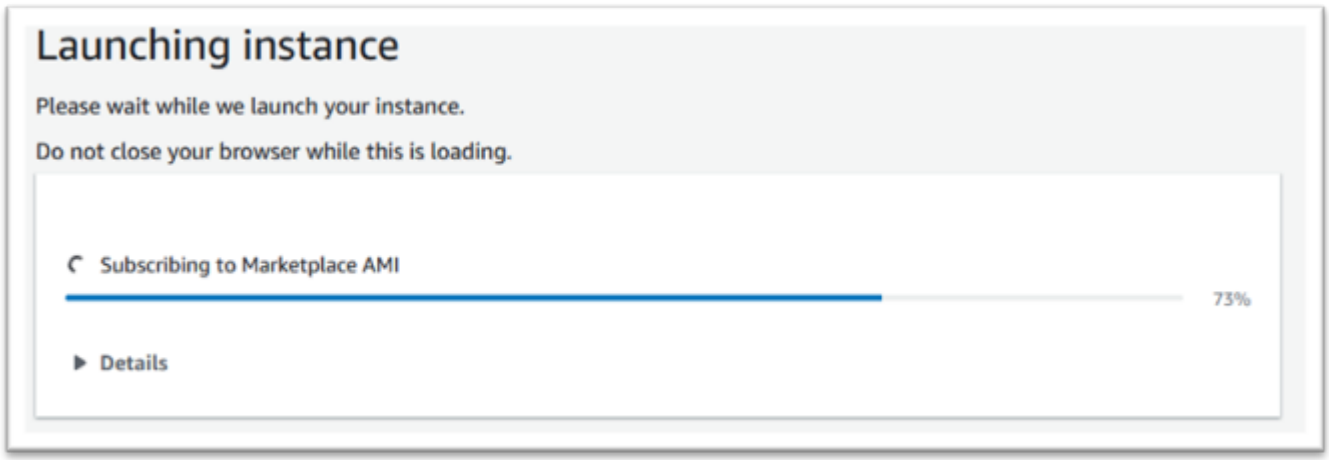


12. 檢閱摘要並推送啟動執行個體。

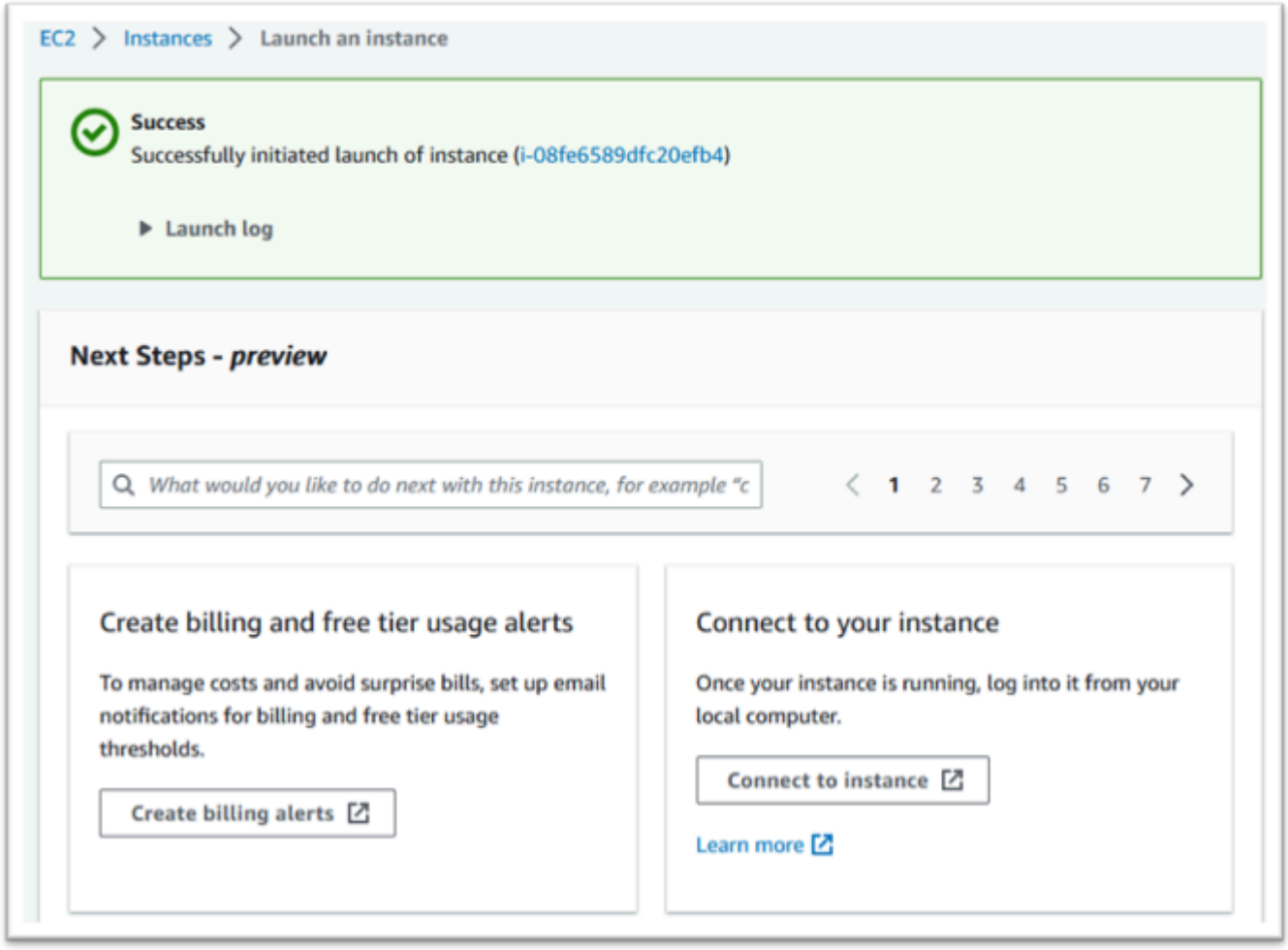


13. 如果選擇無效的虛擬伺服器類型，執行個體啟動將會失敗。

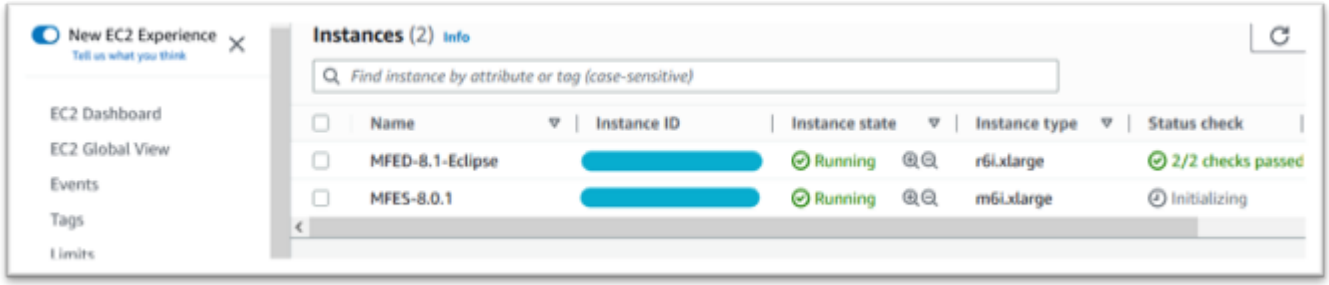
    如果發生這種情況，請選擇編輯執行個體組態並變更執行個體類型。



14. 顯示「成功」訊息後，選擇連線至執行個體以取得連線詳細資訊。



15. 或者，在 中導覽至 EC2 AWS Management Console。
16. 選擇執行個體以查看新執行個體的狀態。



沒有網際網路存取的子網路或 VPC

如果子網路或 VPC 沒有傳出網際網路存取，請進行這些額外的變更。

授權管理員需要存取下列 AWS 服務：

- com.amazonaws.*region*.s3
- com.amazonaws.*region*.ec2
- com.amazonaws.*region*.license-manager
- com.amazonaws.*region*.sts

先前步驟將 com.amazonaws.*region*.s3 服務定義為閘道端點。此端點需要路由表項目，才能用於沒有網際網路存取的任何子網路。

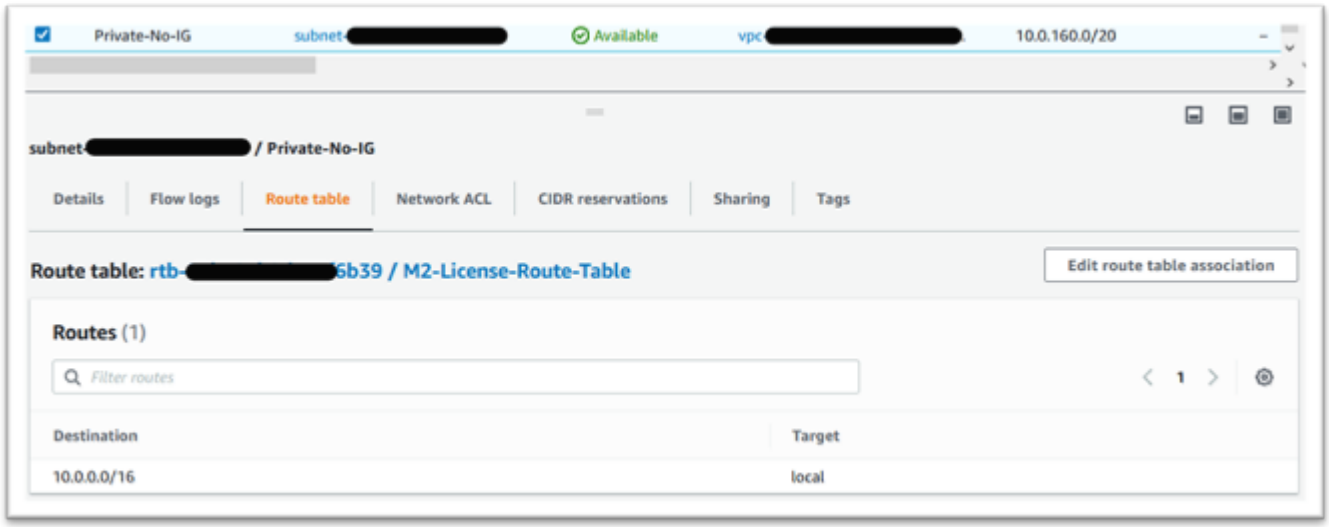
額外的三個服務將定義為介面端點。

主題

- 新增 Amazon S3 端點的路由表項目
- 定義所需的安全群組
- 建立服務端點

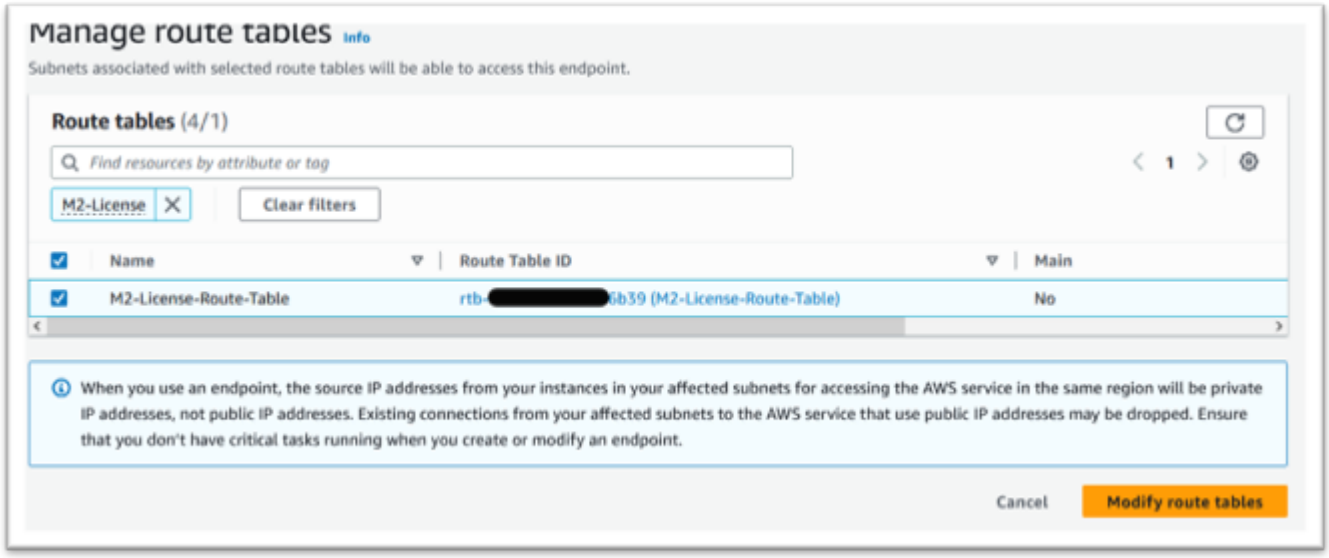
新增 Amazon S3 端點的路由表項目

1. 在 中導覽至 VPC AWS Management Console ，然後選擇子網路。
2. 選擇將建立 Amazon EC2 執行個體的子網路，然後選擇路由表索引標籤。
3. 請注意 Route 資料表 ID 的幾個結尾數字。例如，下圖中的 6b39。



4. 從導覽窗格中選擇端點。

5. 選擇先前建立的端點，然後從端點的路由表索引標籤或從動作下拉式清單中管理路由表。
6. 使用先前識別的數字選擇路由表，然後按修改路由表。

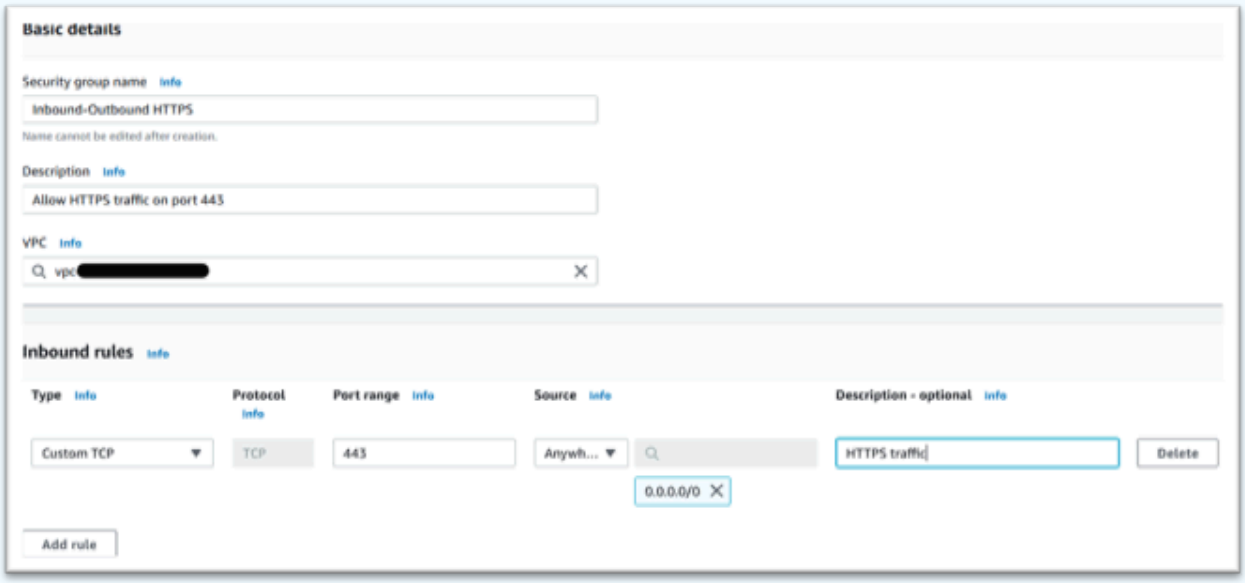


定義所需的安全群組

Amazon EC2 AWS STS和 License Manager 服務會透過連接埠 443 透過 HTTPS 通訊。此通訊是雙向的，需要傳入和傳出規則，以允許執行個體與 服務通訊。

1. 在 中導覽至 Amazon VPC AWS Management Console。
2. 在導覽列中尋找安全群組，然後選擇建立安全群組。
3. 輸入安全群組名稱和描述，例如「傳入/傳出 HTTPS」。
4. 按下 VPC 選擇區域中的 X 以移除預設 VPC，然後選擇包含 S3 端點的 VPC。
5. 新增傳入規則，允許來自任何地方的連接埠 443 上的 TCP 流量。

> **Note**
>
> 透過限制來源，可以進一步限制傳入（和傳出規則）。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的使用安全群組控制 AWS 資源的流量。

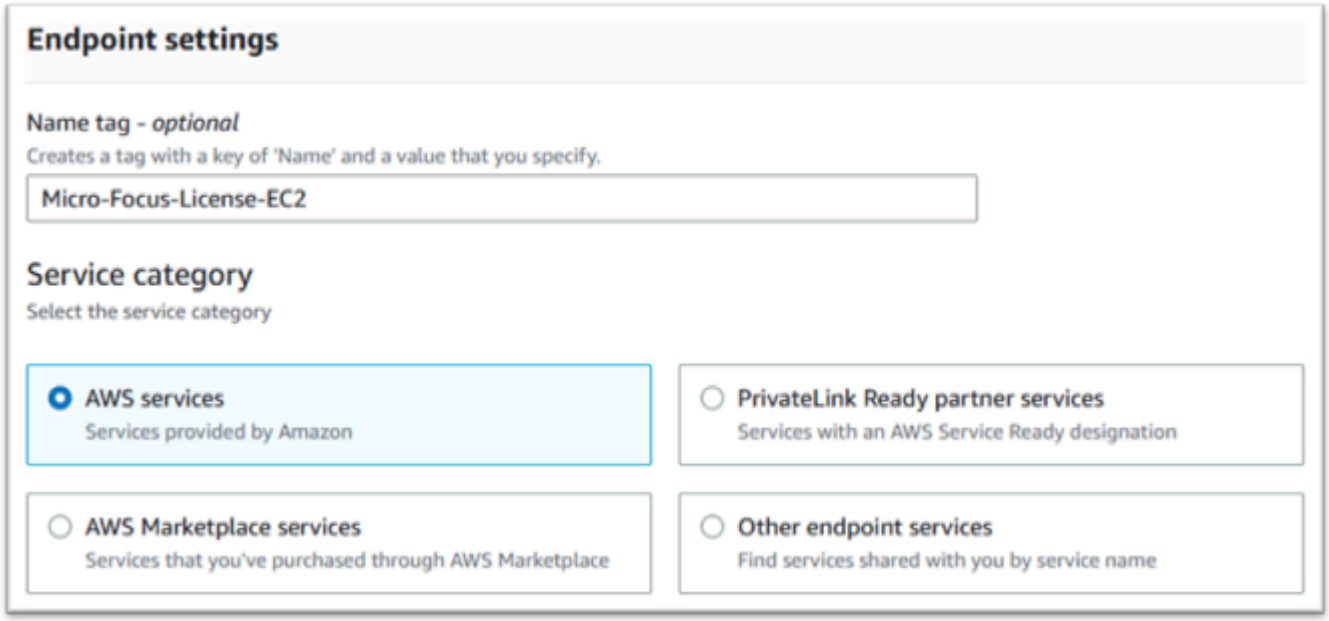


6. 按下建立安全群組。

建立服務端點

重複此程序三次 – 每個服務一次。

1. 在 中導覽至 Amazon VPC， AWS Management Console 然後選擇端點。
2. 按下建立端點。
3. 輸入名稱，例如「Micro-Focus-License-EC2」、「Micro-Focus-License-STS」或「Micro-Focus-License-Manager」。
4. 選擇 AWS Services 服務類別。



5. 在服務下搜尋相符的界面服務，這是以下其中一項：

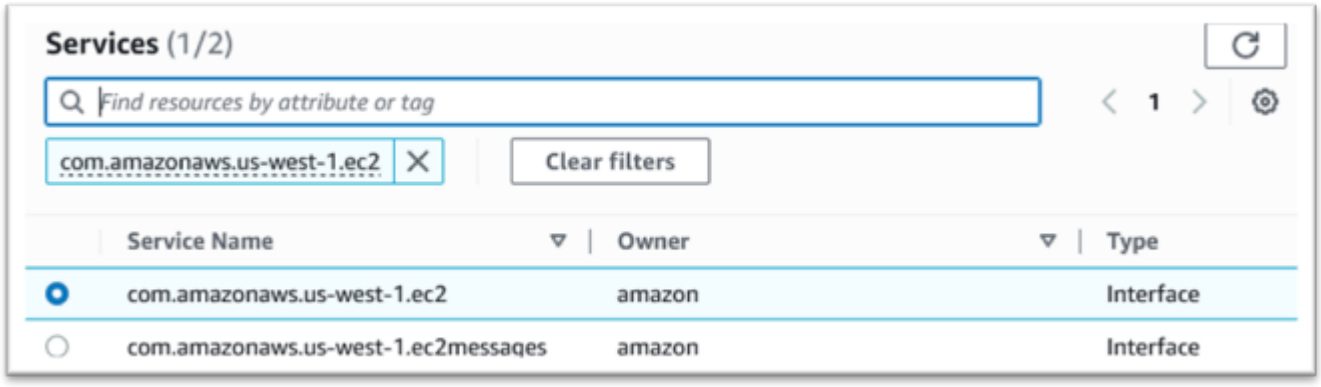
   - 「com.amazonaws.*region*.ec2」
   - 「com.amazonaws.*region*.sts」
   - 「com.amazonaws.*region*.license-manager」

   例如：

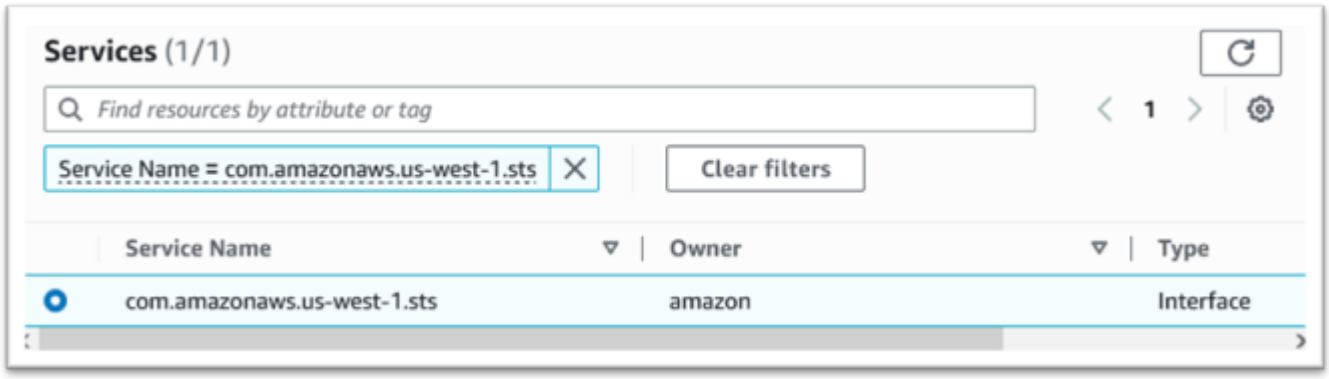
   - 「com.amazonaws.us：//-west-1.ec2」
   - 「com.amazonaws.us：//-west-1.sts」
   - 「com.amazonaws.us：//-west-1.license-manager」

6. 選擇相符的界面服務。

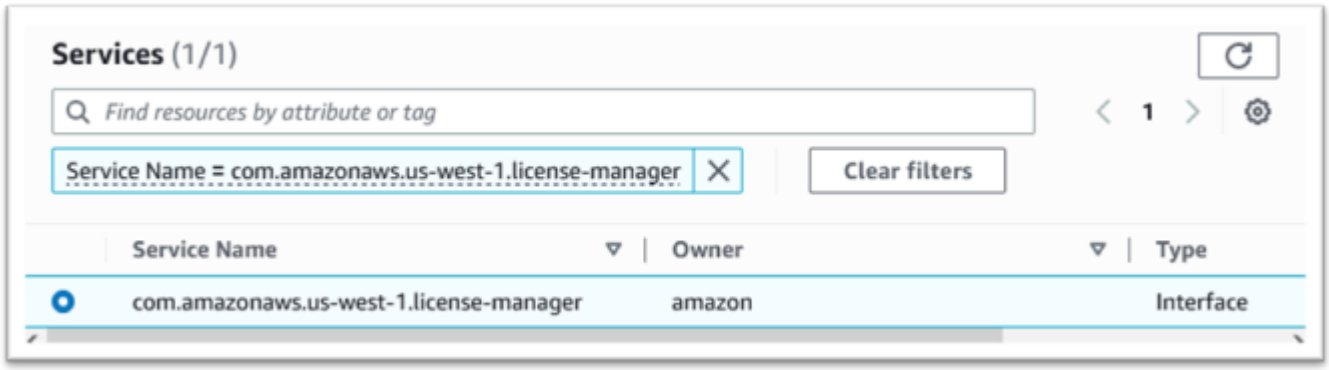
   com.amazonaws.***region***.ec2：



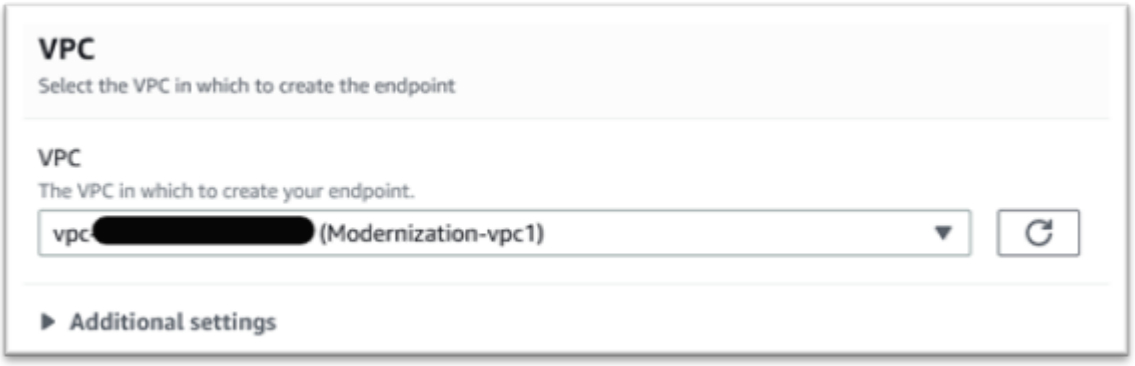
com.amazonaws.***region***.sts：



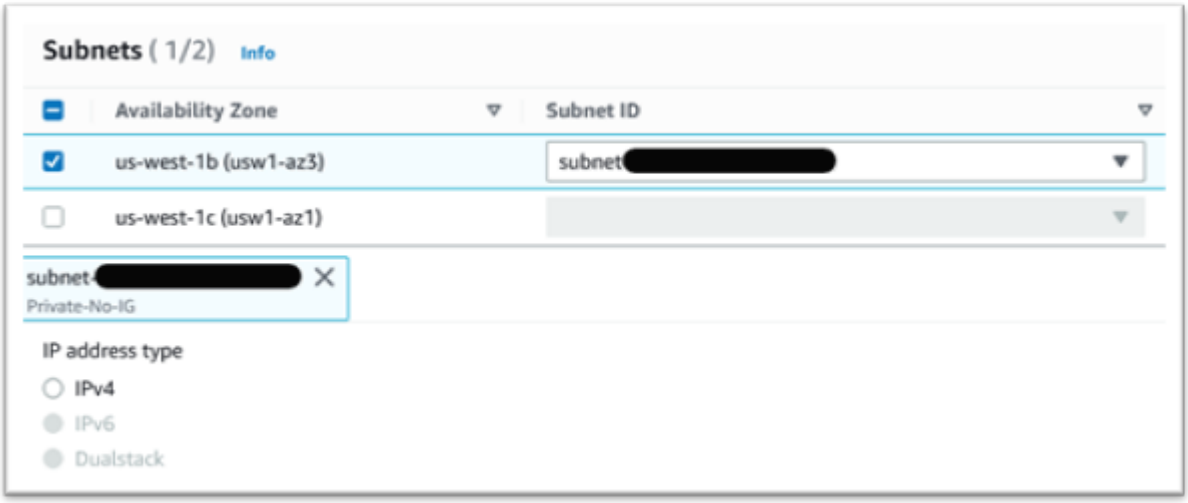
com.amazonaws.***region***.license-manager：



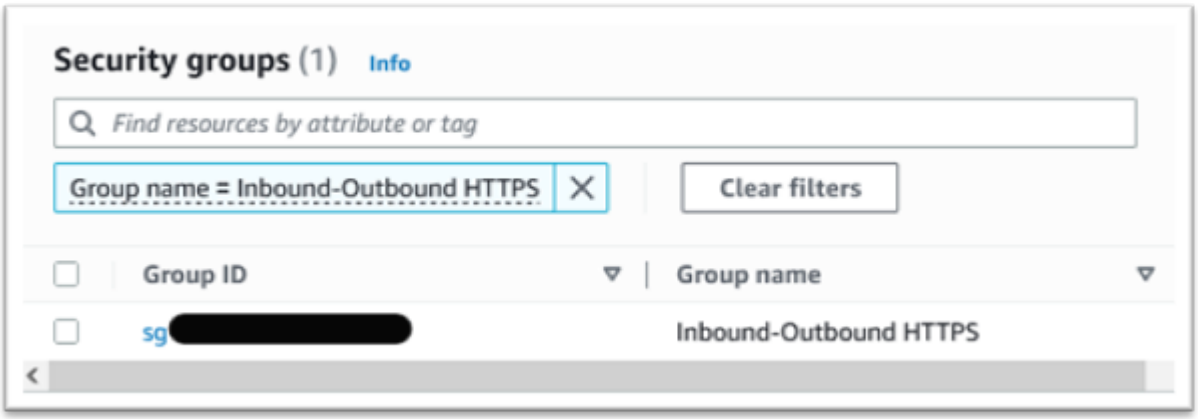
7. 針對 VPC，選擇執行個體的 VPC。



8. 選擇 VPC 的可用區域和子網路。



9. 選擇先前建立的安全群組。



10. 在政策下，選擇完整存取。



11. 選擇建立端點。
12. 對剩餘的界面重複此程序。

設定 Automation for Rocket Enterprise Analyzer （ 先前稱為 Micro Focus) 和 Rocket Enterprise 開發人員串流工作階段

您可以在工作階段開始和結束時自動執行指令碼，以允許客戶內容特定的自動化。如需此 AppStream 2.0 功能的詳細資訊，請參閱《Amazon AppStream 2.0 管理指南》中的使用工作階段指令碼來管理您的 AppStream 2.0 使用者的串流體驗。 Amazon AppStream 2.0

此功能需要您至少擁有以下版本的 Enterprise Analyzer 和 Enterprise Developer 映像：

- `m2-enterprise-analyzer-v8.0.4.R1`
- `m2-enterprise-developer-v8.0.4.R1`

主題

- 在工作階段開始時設定自動化
- 在工作階段結束時設定自動化

在工作階段開始時設定自動化

如果您想要在使用者連線至 AppStream 2.0 時執行自動化指令碼，請建立指令碼並將其命名為 `m2-user-setup.cmd`。將指令碼存放在使用者的 AppStream 2.0 主資料夾中。 AWS Mainframe Modernization 提供的 AppStream 2.0 映像會在該位置尋找具有該名稱的指令碼，並在存在時執行該指令碼。

> **Note**
>
> 指令碼持續時間不能超過 AppStream 2.0 設定的限制，目前為 60 秒。如需詳細資訊，請參閱《Amazon AppStream 2.0 管理指南》中的在串流工作階段開始之前執行指令碼。

在工作階段結束時設定自動化

如果您想要在使用者中斷與 AppStream 2.0 的連線時執行自動化指令碼，請建立指令碼並將其命名為 `m2-user-teardown.cmd`。將指令碼存放在使用者的 AppStream 2.0 主資料夾中。 AWS Mainframe Modernization 提供的 AppStream 2.0 映像會在該位置尋找具有該名稱的指令碼，並在存在時執行該指令碼。

> **Note**
>
> 指令碼持續時間不能超過 AppStream 2.0 設定的限制，目前為 60 秒。如需詳細資訊，請參閱《Amazon AppStream 2.0 管理指南》中的在串流工作階段結束後執行指令碼。

在 Rocket Enterprise Developer （ 先前為 Micro Focus Enterprise Developer) 中將資料集檢視為資料表和資料欄

您可以使用 Rocket Software （ 先前稱為 Micro Focus) 執行時間，存取部署在 AWS Mainframe Modernization 中的大型主機資料集。您可以從 Rocket Enterprise Developer 執行個體將遷移的資料集檢視為資料表和資料欄。以這種方式檢視資料集可讓您：

- 在遷移的資料檔案上執行SQL SELECT操作。
- 在遷移大型主機應用程式外公開資料，而不變更應用程式。
- 輕鬆篩選資料並儲存為 CSV 或其他檔案格式。

> **Note**
>
> 步驟 1 和 2 是一次性活動。針對每個資料集重複步驟 3 和 4，以建立資料庫檢視。

主題

- 先決條件
- 步驟 1：設定連至 Rocket Software 資料存放區的 ODBC 連線 (Amazon RDS 資料庫)
- 步驟 2：建立 MFDBFH.cfg 檔案
- 步驟 3：為您的複製手冊配置建立結構 (STR) 檔案
- 步驟 4：使用 結構 (STR) 檔案建立資料庫檢視
- 步驟 5：將 Rocket Software （ 先前稱為 Micro Focus) 資料集檢視為資料表和資料欄

先決條件

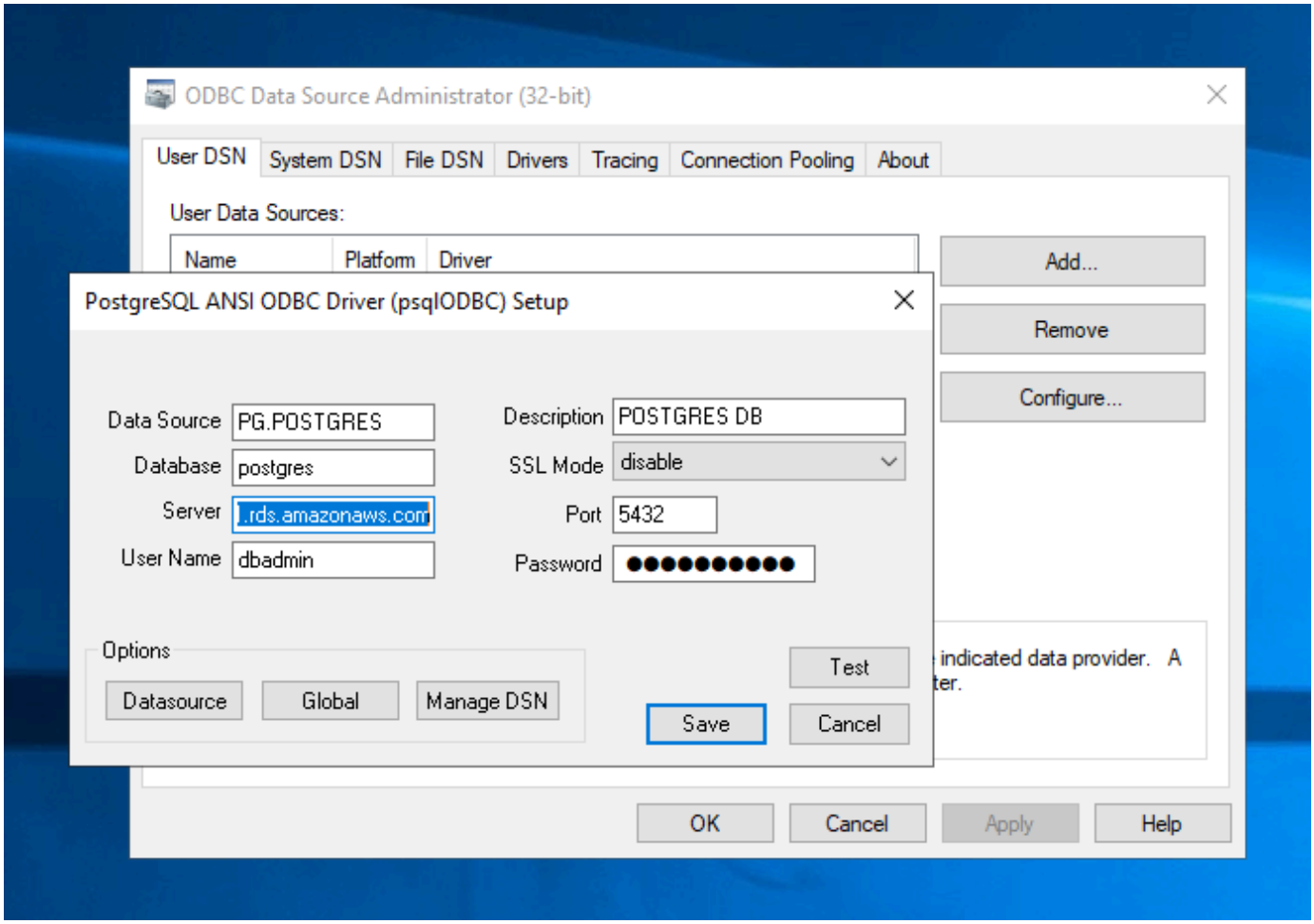
- 您必須能夠透過 AppStream 2.0 存取 Rocket Enterprise Developer Desktop。
- 您必須使用 Rocket Software 執行期引擎，在 AWS Mainframe Modernization 下部署和執行應用程式。
- 您正在 Aurora PostgreSQL 相容版本中存放應用程式資料。

步驟 1：設定連至 Rocket Software 資料存放區的 ODBC 連線 (Amazon RDS 資料庫)

在此步驟中，您會設定資料庫的 ODBC 連線，其中包含要以資料表和資料欄檢視的資料。這是一次性步驟。

1. 使用 AppStream 2.0 串流 URL 登入 Rocket Enterprise Developer Desktop。
2. 開啟 ODBC 資料來源管理員，選擇使用者 DSN，然後選擇新增。
3. 在建立新資料來源中，選擇 PostgreSQL ANSI，然後選擇完成。
4. 透過PG.POSTGRES提供必要的資料庫資訊來建立 的資料來源，如下所示：

```
Data Source : PG.POSTGRES
Database    : postgres
Server      : rds_endpoint.rds.amazonaws.com
Port        : 5432
User Name   : user_name
Password    : user_password
```



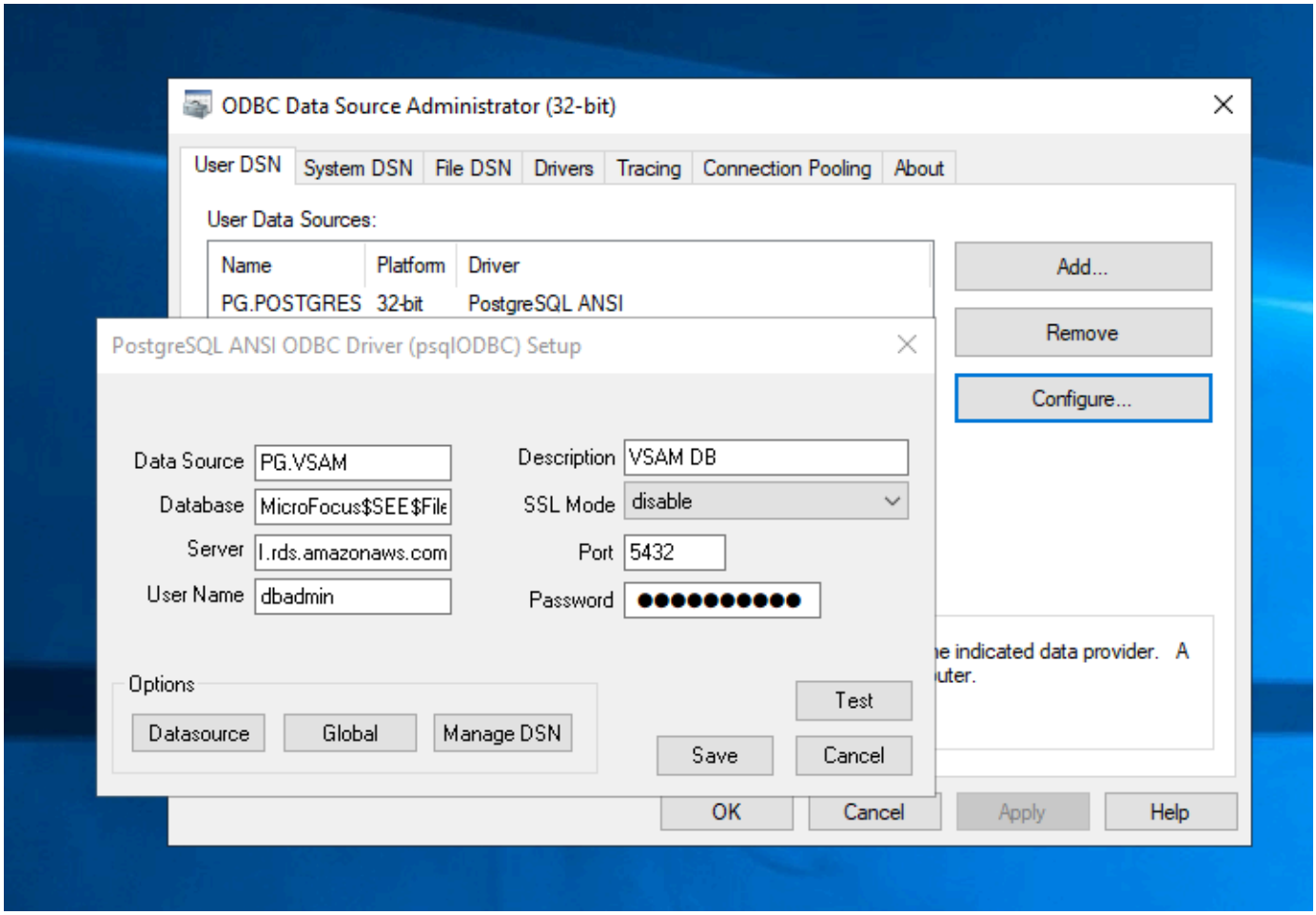
5. 選擇測試，以確保連線正常運作。Connection successful 如果測試成功，您應該會看到訊息。

   如果測試不成功，請檢閱下列資訊。

   - Amazon RDS 故障診斷
   - 如何解決連線至 Amazon RDS 資料庫執行個體時的問題？

6. 儲存資料來源。
7. 建立 的資料來源PG.VSAM、測試連線，以及儲存資料來源。提供下列資料庫資訊：

```
Data Source : PG.VSAM
Database    : MicroFocus$SEE$Files$VSAM
Server      : rds_endpoint.rds.amazonaws.com
Port        : 5432
User Name   : user_name
Password    : user_password
```



步驟 2：建立 MFDBFH.cfg 檔案

在此步驟中，您會建立描述 Micro Focus 資料存放區的組態檔案。這是一次性的組態步驟。

1. 例如，在您的主資料夾中，在 中`D:\PhotonUser\My Files\Home Folder\MFED\cfg\MFDBFH.cfg`，使用下列內容建立 MFDBFH.cfg 檔案。

```
<datastores>
       <server name="ESPACDatabase" type="postgresql" access="odbc">
        <dsn name="PG.POSTGRES" type="database" dbname="postgres"/>
        <dsn name="PG.VSAM" type="datastore" dsname="VSAM"/>
       </server>
      </datastores>
```

2. 執行下列命令來查詢 Micro Focus 資料存放區，以驗證 MFDBFH 組態：

```
*###*
*## Test the connection by running the following commands*
*###*

set MFDBFH_CONFIG="D:\PhotonUser\My Files\Home Folder\MFED\cfg\MFDBFH.cfg"

dbfhdeploy list sql://ESPACDatabase/VSAM?folder=/DATA
```

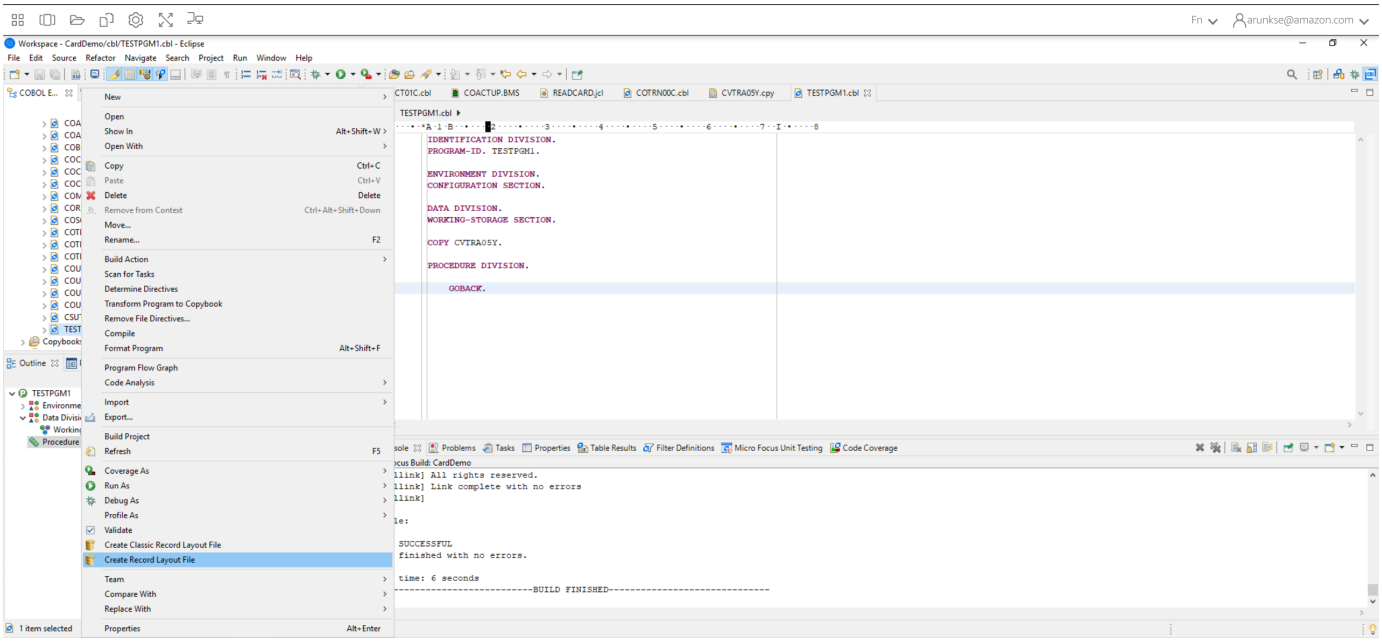
步驟 3：為您的複製手冊配置建立結構 (STR) 檔案

在此步驟中，您會為複製手冊配置建立結構檔案，以便稍後使用它從資料集建立資料庫檢視。

1. 編譯與您複製手冊相關聯的程式。如果沒有程式正在使用複製手冊，請建立簡單的程式，並針對您的複製手冊使用 COPY 陳述式來編譯如下。

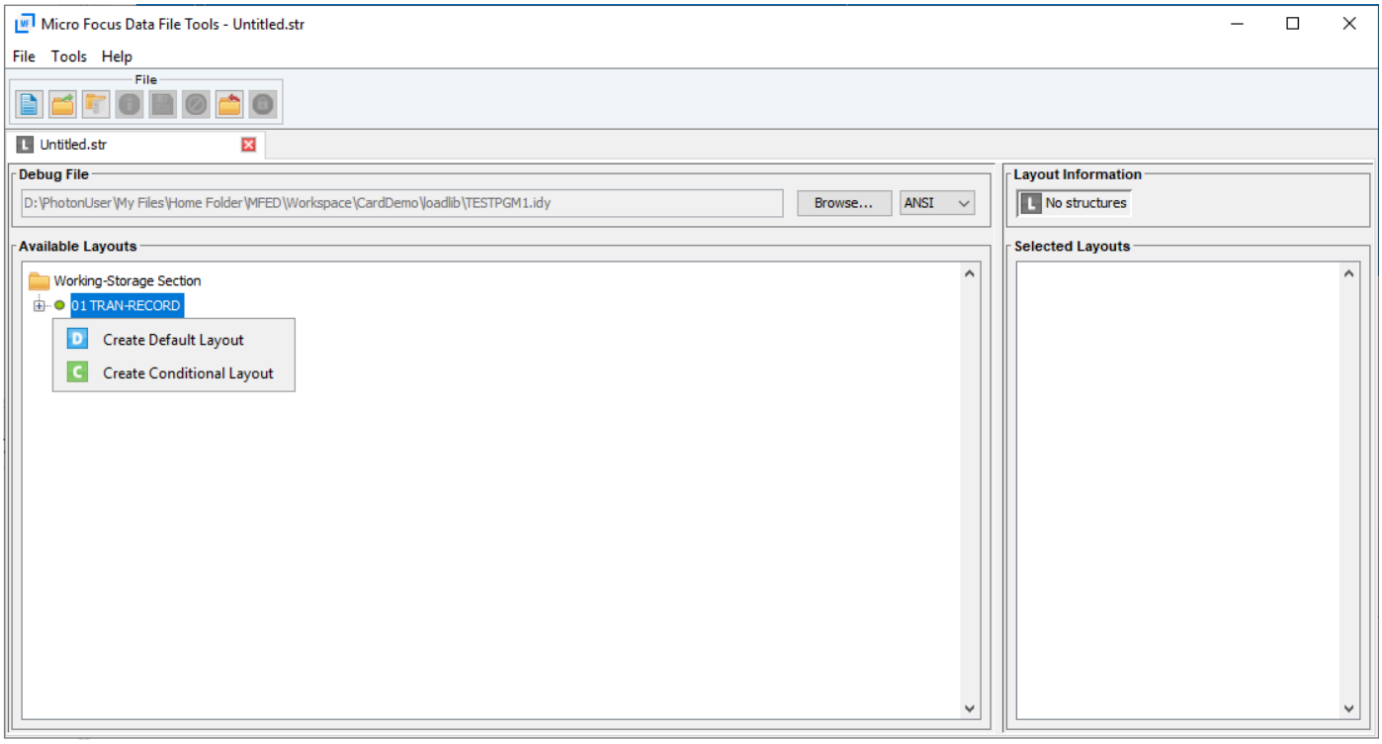
```
IDENTIFICATION DIVISION.
      PROGRAM-ID. TESTPGM1.

      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.

      DATA DIVISION.
      WORKING-STORAGE SECTION.

      COPY CVTRA05Y.

      PROCEDURE DIVISION.

      GOBACK.
```

2. 成功編譯後，在程式上按一下滑鼠右鍵，然後選擇建立記錄配置檔案。這會使用編譯期間產生的 .idy 檔案來開啟 Micro Focus 資料檔案工具。

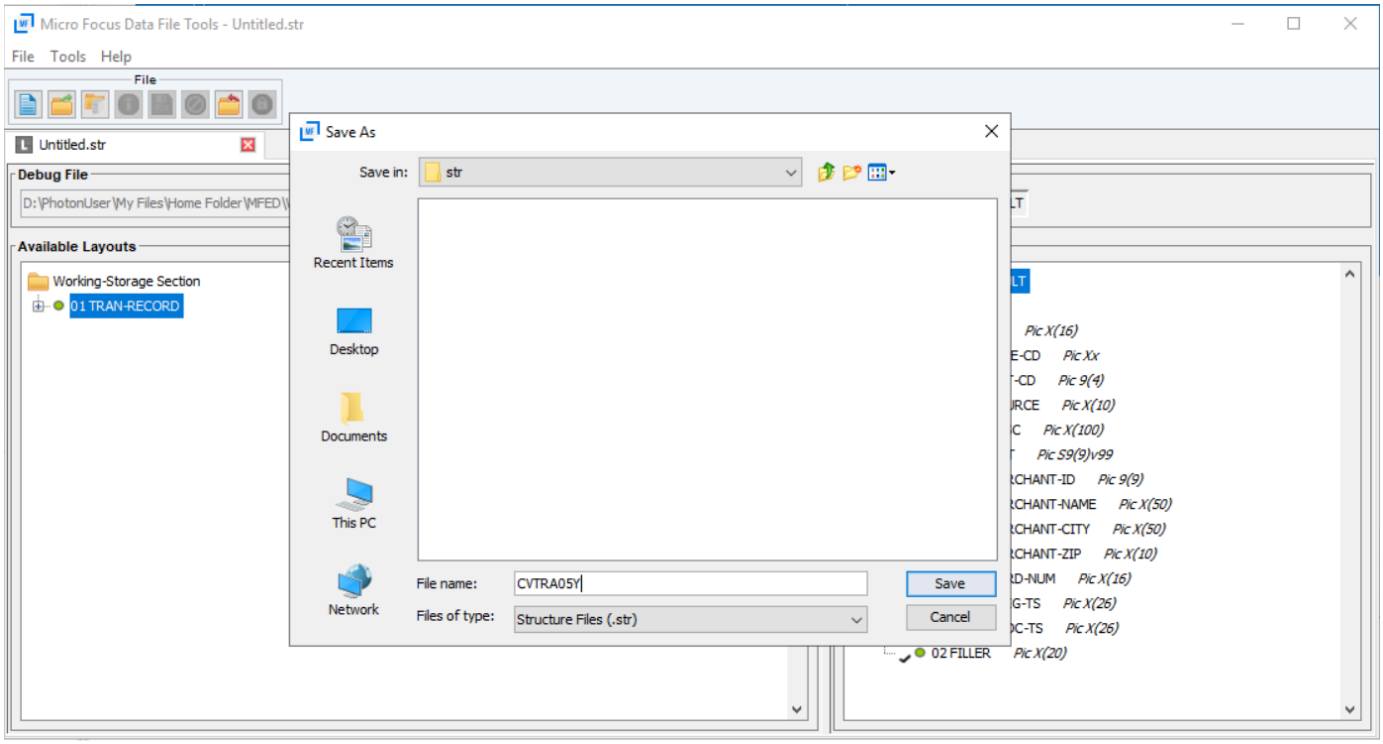


3. 在記錄結構上按一下滑鼠右鍵，然後根據配置選擇建立預設配置（單一結構）或建立條件式配置（多重結構）。

   如需詳細資訊，請參閱 Micro Focus 文件中的建立結構檔案和配置。



4. 建立配置後，從功能表中選擇檔案，然後選擇另存新檔。瀏覽並儲存主資料夾下的檔案，檔案名稱與複製手冊相同。您可以選擇建立名為 的資料夾，`str`並將所有結構檔案儲存在該處。



步驟 4：使用 結構 (STR) 檔案建立資料庫檢視

在此步驟中，您會使用先前建立的結構檔案來建立資料集的資料庫檢視。

- 使用 dbfhview命令為已在 Micro Focus 資料存放區中的資料集建立資料庫檢視，如下列範例所示。

```
##
      ## The below command creates database view for VSAM file
 AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS
      ## using the STR file CVTRA05Y.str
      ##

      dbfhview -create -struct:"D:\PhotonUser\My Files\Home Folder\MFED\str
\CVTRA05Y.str" -name:V_AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT -file:sql://
ESPACDatabase/VSAM/AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT?folder=/DATA

      ##
      ## Output:
      ##

      Micro Focus Database File Handler - View Generation Tool Version 8.0.00
```

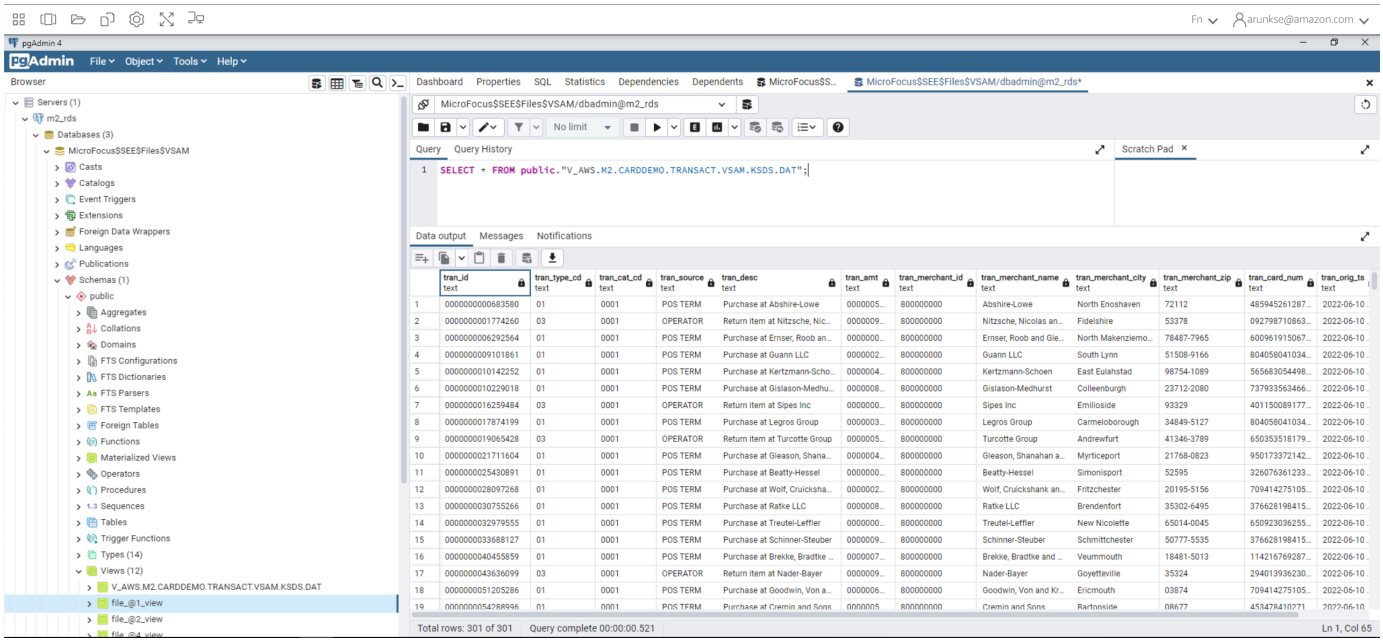
```
      Copyright (C) 1984-2022 Micro Focus. All rights reserved.

      VGN0017I Using structure definition 'TRAN-RECORD-DEFAULT'
      VGN0022I View 'V_AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT' installed in
datastore 'sql://espacdatabase/VSAM'
      VGN0002I The operation completed successfully
```

步驟 5：將 Rocket Software （ 先前稱為 Micro Focus) 資料集檢視為資料表和資料欄

在此步驟中，使用 連線至資料庫pgAdmin，以便您可以執行查詢來檢視資料表和資料欄等資料集。

- MicroFocusSEEFiles$VSAM 使用 pgAdmin 連線到資料庫，並查詢您在步驟 4 中建立的資料庫檢視。

```
SELECT * FROM public."V_AWS.M2.CARDDEMO.TRANSACT.VSAM.KSDS.DAT";
```



在企業開發人員中使用 Rocket Software （ 先前稱為 Micro Focus) 資料檔案工具編輯資料集

您可以使用 Rocket Software 執行時間來檢視和編輯 AWS 大型主機現代化中任何遷移資料集的資料集。
本文件中的步驟將引導您使用資料檔案工具來存取資料集。
這可讓您視需要檢視和編輯遷移的資料集。

主題

- 先決條件
- 啟動 Rocket Software （ 先前稱為 Micro Focus) 資料檔案工具
- 編輯存放在 MFDBFH 資料庫中的 VSAM 資料集
- 編輯存放在 MFDBFH 資料庫中的非 VSAM 資料集
- 編輯存放在檔案系統中的 VSAM 和非 VSAM 資料集 (EFS/FSx)

先決條件

開始之前，您必須使用資料集部署應用程式
使用 Rocket Software 引擎在 AWS Mainframe Modernization 服務下匯入。

若要繼續編輯資料集，您必須完成步驟 1、步驟 2、
和 （ 選用 ） the section called “在企業開發人員中將資料集檢視為資料表”頁面的步驟 3
設定 ODBC 連線和 Micro Focus 資料存放區 （ 即 MFDBFH)。

> **Important**
>
> 本指南假設您使用 Amazon Aurora Postgres 做為 Micro Focus 資料存放區 (MFDBFH)
> 來存放您的應用程式資料。

啟動 Rocket Software （ 先前稱為 Micro Focus) 資料檔案工具

完成先決條件後，您可以透過設定 MFDBFH_CONFIG 環境變數來存取存放在資料庫 () 中的資料集，以啟動 Micro Focus 資料檔案工具MFDBFH。

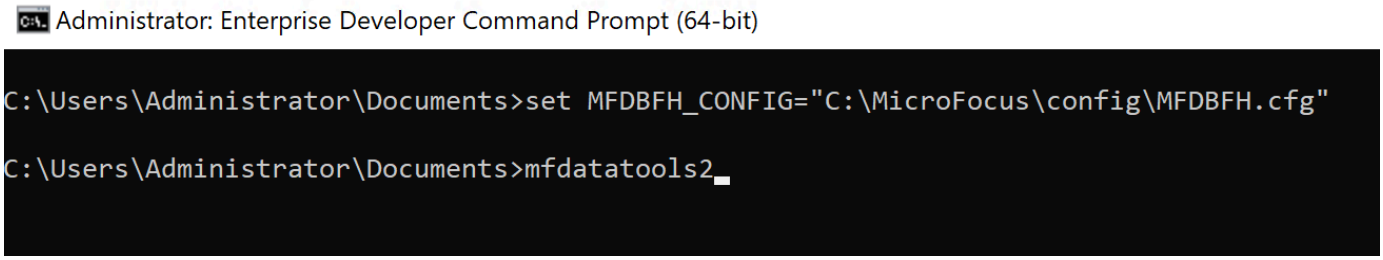
若要執行此作業，

1. 登入 Micro Focus Enterprise Developer 桌面，並從開始功能表啟動企業開發人員命令提示字元 (64 位元）。
2. 使用MFDBCH.cfg檔案的完整路徑設定MFDBFH_CONFIG環境變數。

```
set MFDBFH_CONFIG="C:\MicroFocus\config\MFDBFH.cfg"
```

3. 使用下列命令，從企業開發人員命令列啟動 Micro Focus 資料檔案工具。

```
mfdatatools2
```



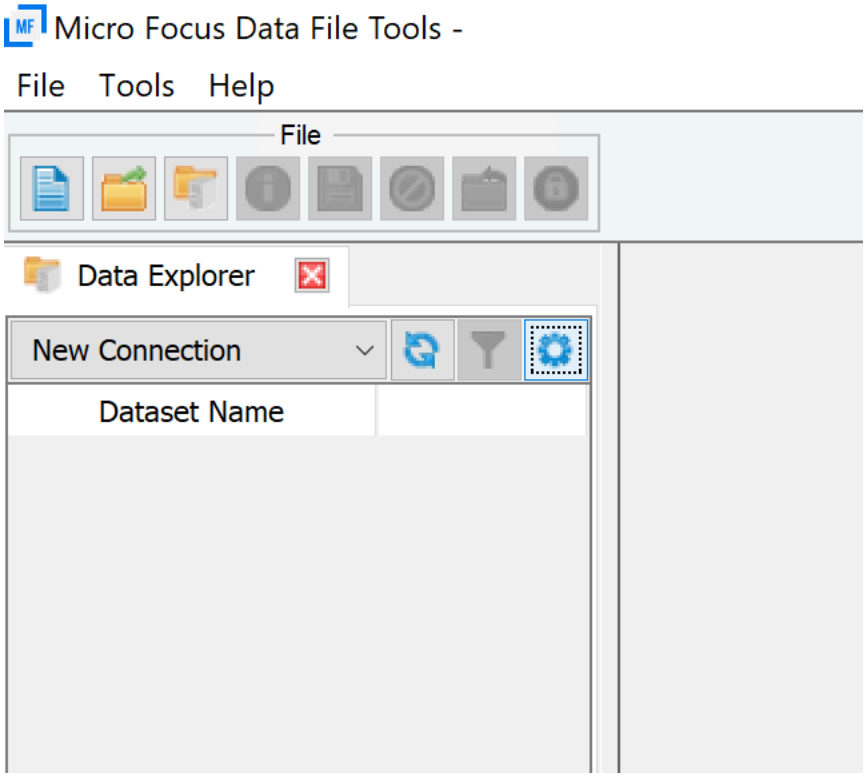
這會在單獨的視窗中開啟 Micro Focus 資料檔案工具。

編輯存放在 MFDBFH 資料庫中的 VSAM 資料集

啟動 Micro Focus 資料檔案工具後，您會開啟存放在 Micro Focus 資料存放區的 VSAM 資料集。

若要執行此作業，

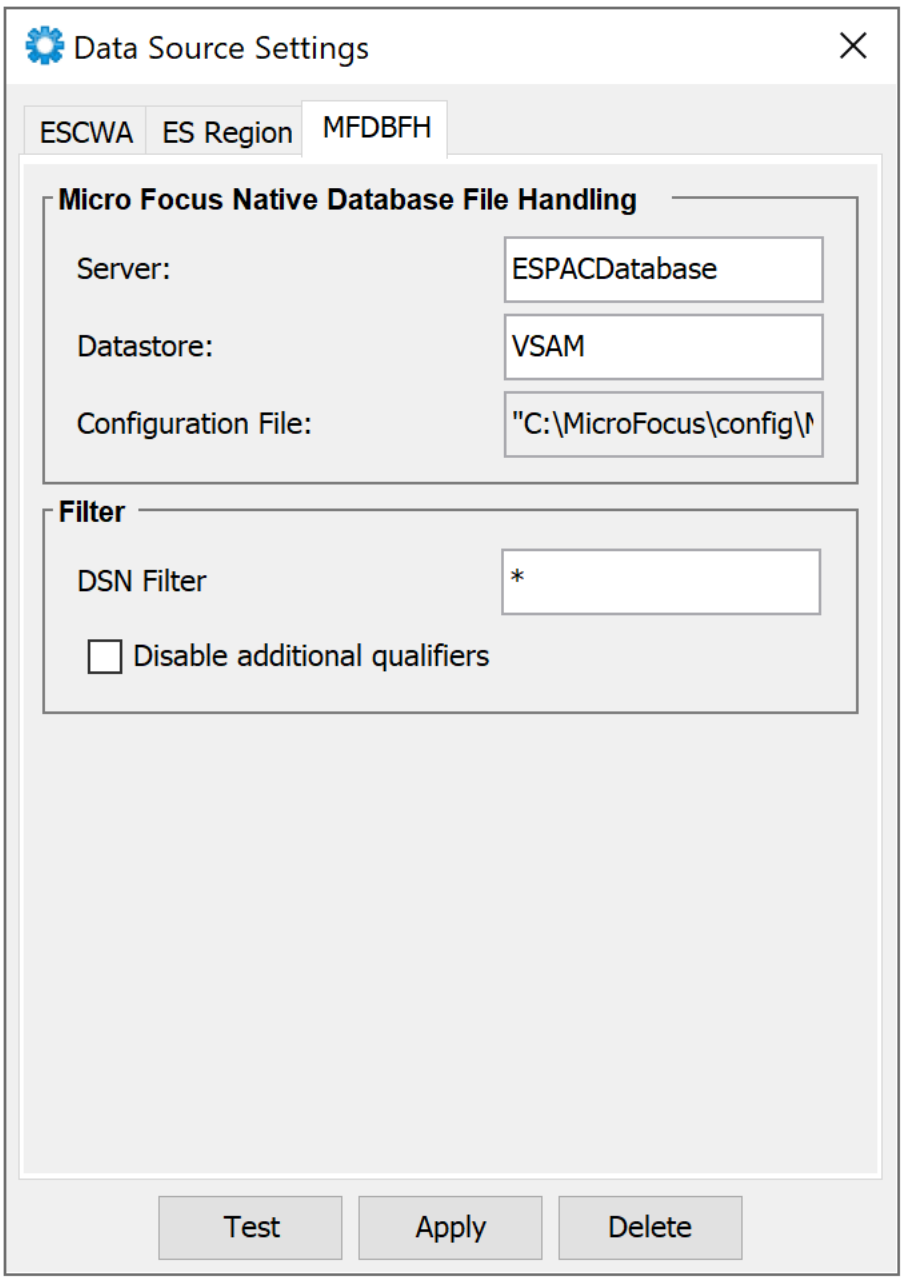
1. 從 Micro Focus Data File Tools 視窗中的檔案選單中，選擇 Data Explorer。
2. 在 Data Explorer 區段中，選擇設定 （齒輪圖示） 以設定新的連線。這會開啟資料來源設定視窗。



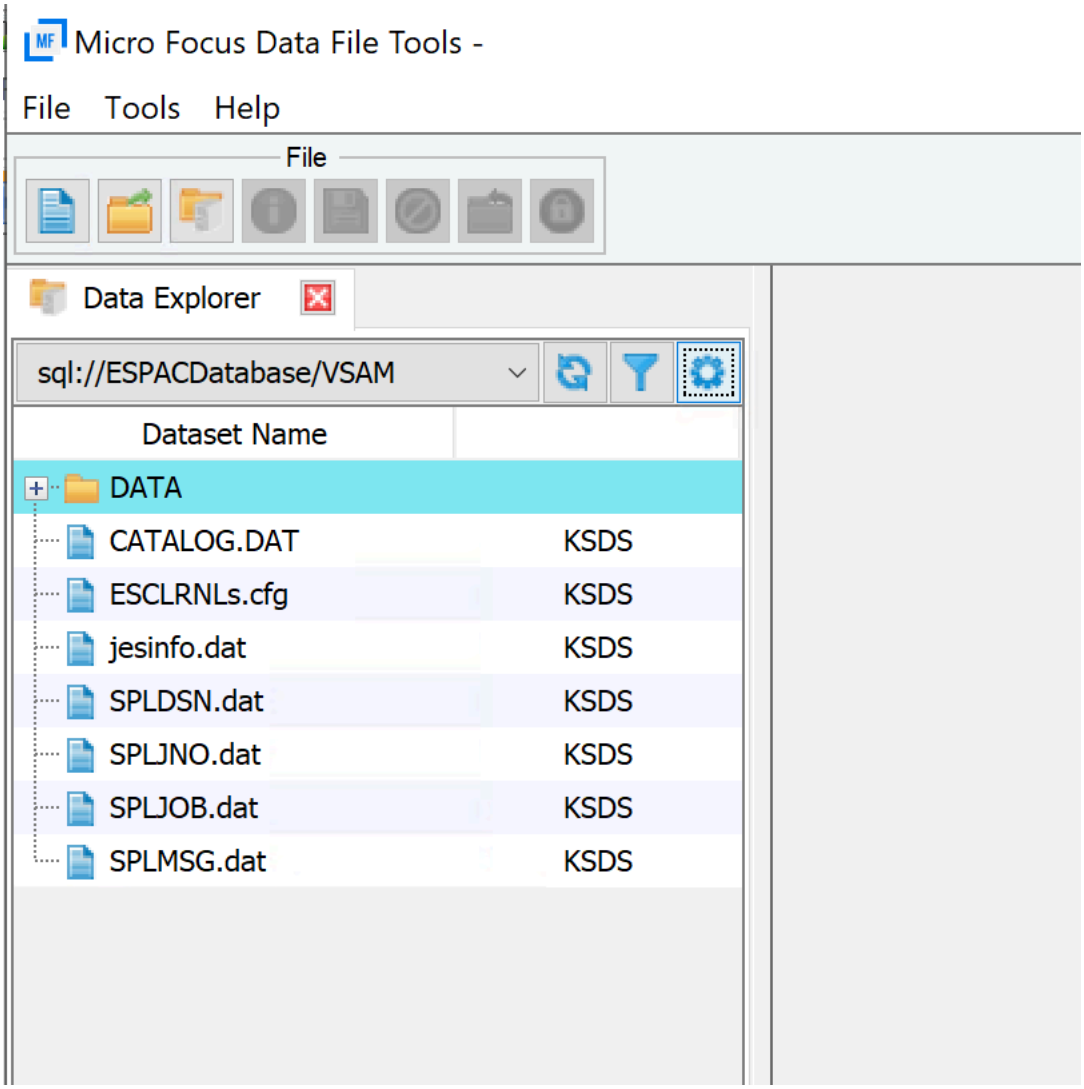
3. 在資料來源設定視窗中，選擇 MFDBFH 索引標籤，然後輸入下列值：

   - 伺服器： `ESPACDatabase`
   - 資料存放區： `VSAM`

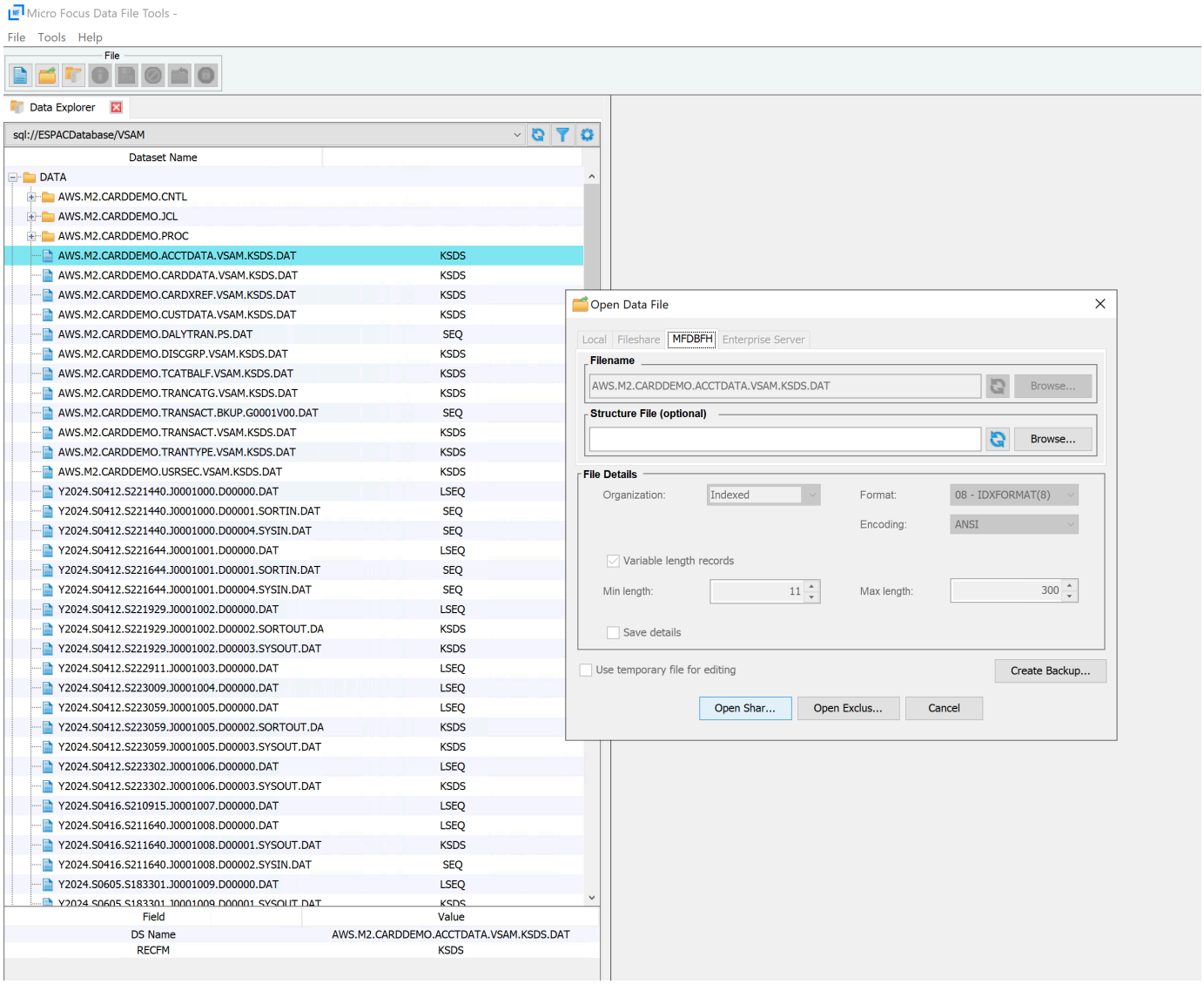
   選擇套用以儲存組態。



Data Explorer 現在會顯示存放在 中的所有資料集MFDBFH。



4. 展開相對路徑DATA，按兩下您要開啟的 VSAM 資料集。
5. 在開啟資料檔案視窗中，選擇開啟共用或開啟獨佔以開啟資料集。



您現在可以檢視或編輯開啟的資料集。

編輯存放在 MFDBFH 資料庫中的非 VSAM 資料集

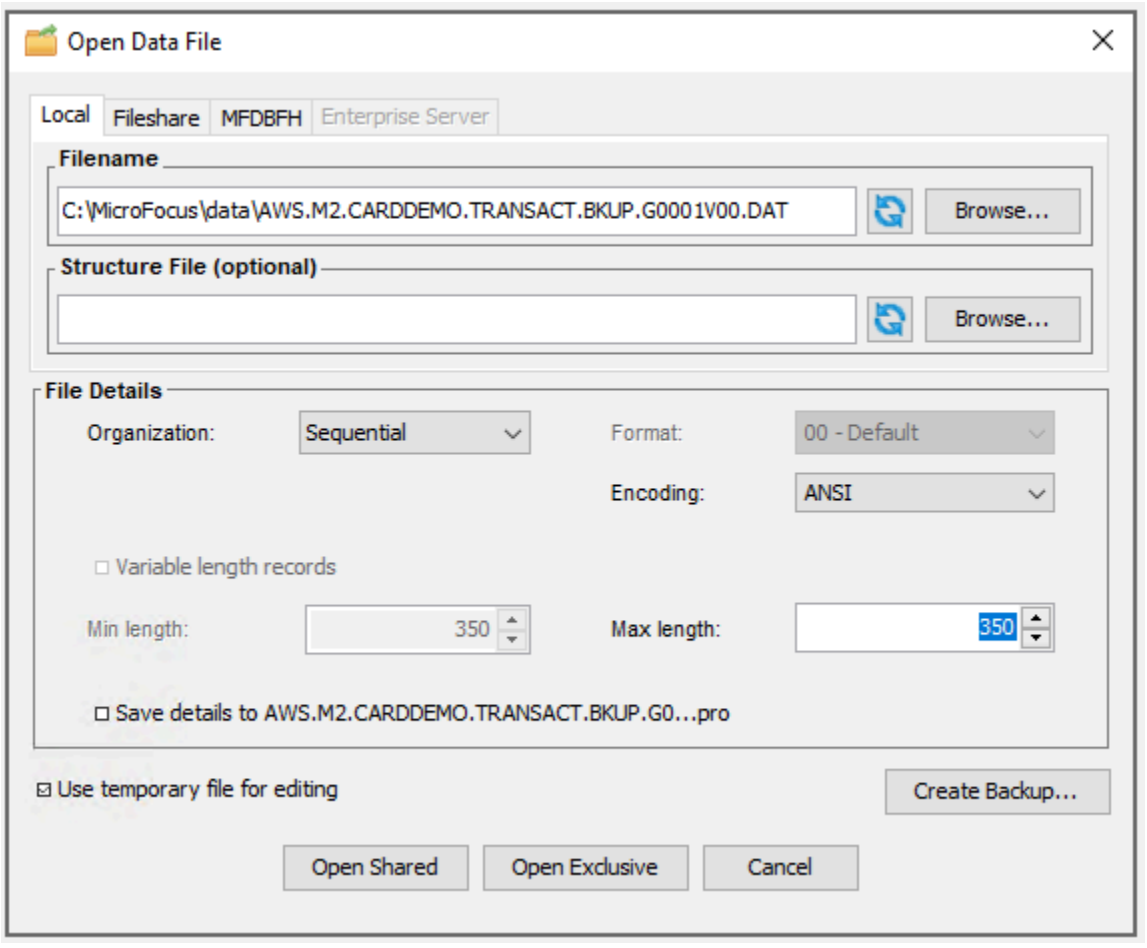
如果您想要編輯非 VSAM 資料集，您可以開啟存放在 Micro Focus 資料存放區中的非 VSAM 資料集。

若要執行此作業，

1. 從企業開發人員命令提示字元 (64 位元） 執行 `dbfhdeploy data extract`命令，將非 VSAM 資料集下載到您的本機檔案系統。

> **Note**
>
> 執行此命令之前，請確定您已使用 MFDBFH.cfg 檔案的完整路徑設定MFDBFH_CONFIG環境變數。
>
> ```
> dbfhdeploy data extract sql://ESPACDatabase/VSAM/
> AWS.M2.CARDDEMO.TRANSACT.BKUP.G0001V00.DAT?folder=/DATA C:\MicroFocus\data
> \AWS.M2.CARDDEMO.TRANSACT.BKUP.G0001V00.DAT
> ```

2. 從開始功能表啟動 Micro Focus 資料檔案工具。
3. 從 Micro Focus Data File Tools 的檔案選單中，選擇開啟，然後選擇資料檔案。
4. 在開啟資料檔案視窗中，瀏覽本機檔案系統中下載的資料集。視需要編輯檔案詳細資訊。然後選擇開啟共用或開啟獨佔以開啟資料集。



您現在可以檢視或編輯開啟的資料集。

編輯或更新的資料集可以使用 中的步驟the section called “匯入應用程式的資料集”或使用 dbfhdeploy Command Line Utility 匯入回 Micro Focus 資料存放區。

編輯存放在檔案系統中的 VSAM 和非 VSAM 資料集 (EFS/FSx)

您也可以開啟存放在檔案系統中的資料集。

若要執行此作業，

1. 在 Enterprise Developer EC2 執行個體的 中掛載EFS/FSx檔案系統。
2. 使用 Micro Focus 資料檔案工具瀏覽，並從檔案系統開啟資料集。

Rocket Software （ 先前為 Micro Focus) 的教學課程

本節中的教學課程可協助您開始為 AWS Mainframe Modernization 服務在 Rocket Software 執行期引擎中設定各種任務。這些教學課程用於設定範例應用程式、CI/CD管道、搭配 Rocket Enterprise Developer 使用 範本，以及設定 Enterprise Analyzer。

主題

- 教學課程：為 BankDemo 範例應用程式設定 Rocket Software （ 先前稱為 Micro Focus) 組建
- 教學課程：設定CI/CD管道以與 Rocket Enterprise Developer （ 先前稱為 Micro Focus Enterprise Developer) 搭配使用
- 教學課程：設定 AppStream 2.0 以搭配 Rocket Enterprise Analyzer 和 Rocket Enterprise Developer 使用
- 教學課程：搭配 Rocket Enterprise Developer （ 先前為 Micro Focus Enterprise Developer) 使用 範本
- 教學課程：在 AppStream 2.0 上設定 Enterprise Analyzer
- 教學課程：在 AppStream 2.0 上設定 Rocket Enterprise Developer

教學課程：為 BankDemo 範例應用程式設定 Rocket Software （ 先前稱為 Micro Focus) 組建

AWS Mainframe Modernization 可讓您為遷移的應用程式設定建置和持續整合/持續交付 (CI/CD) 管道。這些建置和管道會使用 AWS CodeBuild AWS CodeCommit和 AWS CodePipeline 來提供這些功能。CodeBuild 是一種全受管的建置服務，可編譯您的原始程式碼、執行單元測試，並產生準備

好部署的成品。CodeCommit 是一種版本控制服務，可讓您在 AWS 雲端中私下存放和管理 Git 回應。CodePipeline 是一種持續交付服務，可讓您建立模型、視覺化和自動化發行軟體所需的步驟。

本教學課程示範如何使用 從 Amazon S3 AWS CodeBuild 編譯 BankDemo 範例應用程式原始程式碼，然後將編譯後的程式碼匯出回 Amazon S3。

AWS CodeBuild 是一種全受管的持續整合服務，可編譯原始程式碼、執行測試，並產生準備好部署的軟體套件。使用 CodeBuild，您可以使用預先封裝的建置環境，也可以使用您自己的建置工具建立自訂建置環境。此示範案例使用第二個選項。它由使用預先封裝 Docker 映像的 CodeBuild 建置環境組成。

> **Important**
>
> 在開始大型主機現代化專案之前，建議您先了解AWS 大型主機的 Migration Acceleration Program (MAP)，或聯絡AWS 大型主機專家，以了解現代化大型主機應用程式所需的步驟。

主題

- 先決條件
- 步驟 1：與 AWS 帳戶共用建置資產
- 步驟 2：建立 Amazon S3 儲存貯體
- 步驟 3：建立建置規格檔案
- 步驟 4：上傳來源檔案
- 步驟 5：建立 IAM 政策
- 步驟 6：建立 IAM 角色
- 步驟 7：將 IAM 政策連接至 IAM 角色
- 步驟 8：建立 CodeBuild 專案
- 步驟 9：啟動建置
- 步驟 10：下載輸出成品
- 清除資源

先決條件

開始本教學課程之前，請先完成下列先決條件。

- 下載 BankDemo 範例應用程式，並將其解壓縮至資料夾。來源資料夾包含 COBOL 程式、複製手冊和定義。它也包含 JCL 資料夾以供參考，但您不需要建置 JCL。資料夾也包含建置所需的中繼檔案。
- 在 AWS 大型主機現代化主控台中，選擇工具 。在分析、開發和建置資產中，選擇與我的 AWS 帳戶共用資產。

步驟 1：與 AWS 帳戶共用建置資產

在此步驟中，您會確保與 AWS 帳戶共用建置資產，尤其是在使用資產的區域中。

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 在左側導覽中，選擇工具。
3. 在分析、開發和建置資產中，選擇與我的 AWS 帳戶共用資產。

> **Important**
>
> 您需要在每個要進行建置的 AWS 區域中執行此步驟一次。

步驟 2：建立 Amazon S3 儲存貯體

在此步驟中，您會建立兩個 Amazon S3 儲存貯體。第一個是用來保存原始程式碼的輸入儲存貯體，另一個是用來保存建置輸出的輸出儲存貯體。如需詳細資訊，請參閱《Amazon S3 使用者指南》中的建立、設定和使用 Amazon S3 儲存貯體。 Amazon S3

1. 若要建立輸入儲存貯體，請登入 Amazon S3 主控台，然後選擇建立儲存貯體。
2. 在一般組態中，提供儲存貯體的名稱，並指定您要建立儲存貯 AWS 區域 體的 。範例名稱為 `codebuild-regionId-accountId-input-bucket`，其中 `regionId`是儲存貯體 AWS 區域的 ，而 `accountId`是您的 AWS 帳戶 ID。

> **Note**
>
> 如果您要在與美國東部 （維吉尼亞北部） AWS 區域 不同的 中建立儲存貯體，請指定 `LocationConstraint` 參數。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的建立儲存貯體。

3. 保留所有其他設定，然後選擇建立儲存貯體。
4. 重複步驟 1-3 來建立輸出儲存貯體。範例名稱為 codebuild-regionId-accountId-output-bucket，其中 regionId是儲存貯體 AWS 區域 的 ，而 accountId是您的 AWS 帳戶 ID。

   無論您為這些儲存貯體選擇的名稱為何，請務必在本教學課程中使用這些儲存貯體。

步驟 3：建立建置規格檔案

在此步驟中，您會建立建置規格檔案。此檔案提供 YAML 格式的建置命令和相關設定，讓 CodeBuild 執行建置。如需詳細資訊，請參閱AWS CodeBuild 《 使用者指南》中的為 CodeBuild 建置規格參考。

1. 在您解壓縮為先決條件buildspec.yml的目錄中建立名為 的檔案。
2. 將下列內容新增至 檔案並儲存。此檔案不需要變更。

```
version: 0.2
env:
  exported-variables:
    - CODEBUILD_BUILD_ID
    - CODEBUILD_BUILD_ARN
phases:
  install:
    runtime-versions:
      python: 3.7
  pre_build:
    commands:
      - echo Installing source dependencies...
      - ls -lR $CODEBUILD_SRC_DIR/source
  build:
    commands:
      - echo Build started on `date`
      - /start-build.sh -Dbasedir=$CODEBUILD_SRC_DIR/source -Dloaddir=
$CODEBUILD_SRC_DIR/target
  post_build:
    commands:
      - ls -lR $CODEBUILD_SRC_DIR/target
      - echo Build completed on `date`
artifacts:
  files:
    - $CODEBUILD_SRC_DIR/target/**
```

此處的 CODEBUILD_BUILD_ID、$CODEBUILD_SRC_DIR/source、CODEBUILD_BUILD_ARN和 $CODEBUILD_SRC_DIR/target是 CodeBuild 中可用的環境變數。如需詳細資訊，請參閱建置環境中的環境變數。

此時，您的目錄看起來應該像這樣。

```
(root directory name)
    |-- build.xml
    |-- buildspec.yml
    |-- LICENSE.txt
    |-- source
         |... etc.
```

3. 將資料夾的內容壓縮至名為 的檔案BankDemo.zip。在本教學課程中，您無法壓縮 資料夾。反之，請將資料夾的內容壓縮至檔案 BankDemo.zip。

步驟 4：上傳來源檔案

在此步驟中，您將 BankDemo 範例應用程式的原始碼上傳至 Amazon S3 輸入儲存貯體。

1. 登入 Amazon S3 主控台，然後在左側導覽窗格中選擇儲存貯體。然後選擇您先前建立的輸入儲存貯體。
2. 在物件下，選擇上傳。
3. 在檔案和資料夾區段中，選擇新增檔案。
4. 導覽至 並選擇您的 BankDemo.zip 檔案。
5. 選擇上傳。

步驟 5：建立 IAM 政策

在此步驟中，您會建立兩個 IAM 政策。一項政策授予 AWS Mainframe Modernization 存取和使用包含 Rocket Software 組建工具的 Docker 映像的許可。此政策不會為客戶自訂。其他政策會授予 AWS Mainframe Modernization 與輸入和輸出儲存貯體以及 CodeBuild 產生的 Amazon CloudWatch logs互動的許可。

若要了解如何建立 IAM 政策，請參閱《IAM 使用者指南》中的編輯 IAM 政策。

建立存取 Docker 映像的政策

1. 在 IAM 主控台中，複製下列政策文件並將其貼到政策編輯器中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "arn:aws:ecr:*:673918848628:repository/m2-enterprise-build-
tools"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::aws-m2-repo-*-<region>-prod"
        }
    ]
}
```

2. 提供政策的名稱，例如 `m2CodeBuildPolicy`。

建立允許 AWS Mainframe Modernization 與儲存貯體和日誌互動的政策

1. 在 IAM 主控台中，複製下列政策文件並將其貼到政策編輯器中。請務必`regionId`將 AWS 區域和更新`accountId`為 AWS 帳戶。

```
{
```

```
      "Version": "2012-10-17",
      "Statement": [
          {
              "Action": [
                  "logs:CreateLogGroup",
                  "logs:CreateLogStream",
                  "logs:PutLogEvents"
              ],
              "Resource": [
                  "arn:aws:logs:regionId:accountId:log-group:/aws/codebuild/
codebuild-bankdemo-project",
                  "arn:aws:logs:regionId:accountId:log-group:/aws/codebuild/
codebuild-bankdemo-project:*"
              ],
              "Effect": "Allow"
          },
          {
              "Action": [
                  "s3:PutObject",
                  "s3:GetObject",
                  "s3:GetObjectVersion",
                  "s3:GetBucketAcl",
                  "s3:GetBucketLocation",
                  "s3:List*"
              ],
              "Resource": [
                  "arn:aws:s3:::codebuild-regionId-accountId-input-bucket",
                  "arn:aws:s3:::codebuild-regionId-accountId-input-bucket/*",
                  "arn:aws:s3:::codebuild-regionId-accountId-output-bucket",
                  "arn:aws:s3:::codebuild-regionId-accountId-output-bucket/*"
              ],
              "Effect": "Allow"
          }
      ]
  }
```

2. 提供政策的名稱，例如 BankdemoCodeBuildRolePolicy。

步驟 6：建立 IAM 角色

在此步驟中，您會建立新的 IAM 角色，允許 CodeBuild 在您將先前建立的 IAM 政策與此新 IAM 角色建立關聯之後，與您 AWS 資源互動。

如需有關建立服務角色的資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可給 AWS 服務。

1. 登入 IAM 主控台，然後在左側導覽窗格中選擇角色。
2. 選擇建立角色。
3. 在信任的實體類型下，選擇 AWS 服務。
4. 在其他 AWS 服務的使用案例下，選擇 CodeBuild，然後再次選擇 CodeBuild。
5. 選擇下一步。
6. 在 Add permissions (新增許可) 頁面上，選擇 Next (下一步)。您稍後會將政策指派給角色。
7. 在角色詳細資訊下，提供角色的名稱，例如 BankdemoCodeBuildServiceRole。
8. 在選取信任的實體下，確認政策文件看起來如下：

```
{
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": "codebuild.amazonaws.com"
              },
              "Action": "sts:AssumeRole"
            }
          ]
        }
```

9. 選擇建立角色。

步驟 7：將 IAM 政策連接至 IAM 角色

在此步驟中，您將先前建立的兩個 IAM 政策連接至 IAM BankdemoCodeBuildServiceRole 角色。

1. 登入 IAM 主控台，然後在左側導覽窗格中選擇角色。
2. 在角色中，選擇您先前建立的角色，例如 BankdemoCodeBuildServiceRole。
3. 在許可政策中，選擇新增許可，然後選擇連接政策。
4. 在其他許可政策中，選擇您先前建立的政策，例如 m2CodeBuildPolicy和 BankdemoCodeBuildRolePolicy。
5. 選擇連接政策。

步驟 8：建立 CodeBuild 專案

在此步驟中，您會建立 CodeBuild 專案。

1. 登入 CodeBuild 主控台，然後選擇建立建置專案。
2. 在專案組態區段中，提供專案的名稱，例如 `codebuild-bankdemo-project`。
3. 在來源區段中，針對來源提供者選擇 Amazon S3，然後選擇您先前建立的輸入儲存貯體，例如 `codebuild-regionId-accountId-input-bucket`。
4. 在 S3 物件金鑰或 S3 資料夾欄位中，輸入您上傳到 S3 儲存貯體的 zip 檔案名稱。在此情況下，檔案名稱為 `bankdemo.zip`。
5. 在環境區段中，選擇自訂映像。
6. 在環境類型欄位中，選擇 Linux。
7. 在映像登錄檔下，選擇其他登錄檔。
8. 在外部登錄 URL 欄位中，
   - 對於 Rocket 軟體 v9：輸入 `673918848628.dkr.ecr.us-west-1.amazonaws.com/m2-enterprise-build-tools:9.0.7.R1`。如果您將不同的 AWS 區域與 Rocket Software v9 搭配使用，您也可以指定 `673918848628.dkr.ecr.<m2-region>.amazonaws.com/m2-enterprise-build-tools:9.0.7.R1`，其中 <m2-region> 是可使用 AWS 大型主機現代化服務 AWS 的區域（例如 `eu-west-3`）。
   - 對於 Rocket 軟體 v8：輸入 `673918848628.dkr.ecr.us-west-2.amazonaws.com/m2-enterprise-build-tools:8.0.9.R1`
   - 對於 Rocket 軟體 v7：輸入 `673918848628.dkr.ecr.us-west-2.amazonaws.com/m2-enterprise-build-tools:7.0.R10`
9. 在服務角色下，選擇現有服務角色，然後在角色 ARN 欄位中，選擇您先前建立的服務角色；例如，`BankdemoCodeBuildServiceRole`。
10. 在 Buildspec 區段中，選擇使用 buildspec 檔案。
11. 在成品區段的類型下，選擇 Amazon S3，然後選擇輸出儲存貯體，例如 `codebuild-regionId-accountId-output-bucket`。
12. 在名稱欄位中，輸入您要包含建置輸出成品之儲存貯體中的資料夾名稱，例如 `bankdemo-output.zip`。
13. 在成品封裝下，選擇 Zip。
14. 選擇 Create build project (建立建置專案)。

步驟 9：啟動建置

在此步驟中，您會啟動組建。

1. 登入 CodeBuild 主控台。
2. 在左側導覽窗格中，選擇建置專案。
3. 選擇您先前建立的建置專案，例如 `codebuild-bankdemo-project`。
4. 選擇 Start build (開始組建)。

此命令會啟動建置。建置會以非同步方式執行。命令的輸出是包含 屬性 ID 的 JSON。此屬性 idis 是您剛啟動之組建的 CodeBuild 組建 ID 參考。您可以在 CodeBuild 主控台中檢視組建的狀態。您也可以在 主控台中查看有關建置執行的詳細日誌。如需詳細資訊，請參閱AWS CodeBuild 《 使用者指南》中的檢視詳細的建置資訊。

當目前階段為 COMPLETED 時，表示您的建置已成功完成，而且您的編譯成品已在 Amazon S3 上準備就緒。

步驟 10：下載輸出成品

在此步驟中，您會從 Amazon S3 下載輸出成品。Rocket Software 建置工具可以建立數種不同的可執行檔類型。在本教學課程中，它會產生共用物件。

1. 登入 Amazon S3 主控台。
2. 在儲存貯體 role="bold"> 區段中，選擇輸出儲存貯體的名稱，例如 `codebuild-regionId-accountId-output-bucket`。
3. 選擇下載 role="bold">。
4. 解壓縮所下載的 檔案。導覽至目標資料夾以查看建置成品。這些包括 `.so` Linux 共用物件。

清除資源

如果您不再需要為本教學課程建立的資源，請將其刪除以避免額外費用。若要這樣做，請完成下列步驟：

- 刪除您為此教學課程建立的 S3 儲存貯體。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的刪除儲存貯體。
- 刪除您為此教學課程建立的 IAM 政策。如需詳細資訊，請參閱《IAM 使用者指南》中的刪除 IAM 政策。

- 刪除您為此教學課程建立的 IAM 角色。如需詳細資訊，請參閱 IAM 使用者指南中的刪除角色或執行個體描述檔。
- 刪除您為此教學課程建立的 CodeBuild 專案。如需詳細資訊，請參閱AWS CodeBuild 《 使用者指南》中的在 CodeBuild 中刪除組建專案。

教學課程：設定CI/CD管道以與 Rocket Enterprise Developer （ 先前稱為 Micro Focus Enterprise Developer) 搭配使用

本教學課程說明如何在 Rocket Enterprise Developer 中匯入、編輯、編譯和執行 BankDemo 範例應用程式，然後遞交變更以觸發CI/CD管道。

內容

- 先決條件
- 建立CI/CD管道基本基礎設施
- 建立 AWS CodeCommit 儲存庫和CI/CD管道
  - YAML 觸發檔案 config_git.yml 範例
- 建立企業開發人員 AppStream 2.0
- 企業開發人員設定和測試
  - 在企業開發人員中複製 BankDemo CodeCommit 儲存庫
  - 建立 BankDemo 大型主機 COBOL 專案並建置應用程式
  - 建立本機 BankDemo CICS 和批次環境進行測試
  - 從企業開發人員啟動 BANKDEMO 伺服器
  - 啟動 Rumba 3270 終端機
  - 執行 BankDemo 交易
  - 從企業開發人員停止 BANKDEMO 伺服器
- 練習 1：增強 BANKDEMO 應用程式中的貸款計算
  - 將貸款分析規則新增至企業開發人員程式碼分析
  - 步驟 1：執行貸款計算的程式碼分析
  - 步驟 2：修改 CICSMS 映射和 COBOL 程式和測試
  - 步驟 3：在 COBOL 程式中新增總金額計算
  - 步驟 4：遞交變更並執行 CI/CD 管道
- 練習 2：在 BankDemo 應用程式中擷取貸款計算

  - 步驟 1：將貸款計算常式重構為 COBOL 區段
  - 步驟 2：將貸款計算常式擷取至獨立 COBOL 程式
  - 步驟 3：遞交變更並執行CI/CD管道
- 清除資源

先決條件

下載下列檔案。

- basic-infra.yaml
  - 從歐洲（法蘭克福）區域下載。
  - 從美國東部（維吉尼亞北部）區域下載。
- pipeline.yaml
  - 從歐洲（法蘭克福）區域下載。
  - 從美國東部（維吉尼亞北部）區域下載。
- m2-code-sync-function.zip
  - 從歐洲（法蘭克福）區域下載。
  - 從美國東部（維吉尼亞北部）區域下載。
- config_git.yml
  - 從歐洲（法蘭克福）區域下載。
  - 從美國東部（維吉尼亞北部）區域下載。
- BANKDEMO-source.zip
  - 從歐洲（法蘭克福）區域下載。
  - 從美國東部（維吉尼亞北部）區域下載。
- BANKDEMO-exercise.zip
  - 從歐洲（法蘭克福）區域下載。
  - 從美國東部（維吉尼亞北部）區域下載。

每個檔案的用途如下：

`basic-infra.yaml`

此 AWS CloudFormation 範本會建立CI/CD管道所需的基本基礎設施：VPC、Amazon S3 儲存貯體等。

`pipeline.yaml`

Lambda 函數使用此 AWS CloudFormation 範本來啟動管道堆疊。請確定此範本位於可公開存取的 Amazon S3 儲存貯體中。新增此儲存貯體的連結做為`basic-infra.yaml`範本中`PipelineTemplateURL`參數的預設值。

`m2-code-sync-function.zip`

此 Lambda 函數會根據 建立 CodeCommit 儲存庫、目錄結構`config_git.yaml`，並使用 啟動管道堆疊`pipeline.yaml`。請確認此 zip 檔案可在支援 AWS Mainframe Modernization 的所有 中公開存取 AWS 區域 的 Amazon S3 儲存貯體中使用。我們建議您將檔案存放在一個儲存貯體中， AWS 區域 並將它複寫到所有儲存貯體 AWS 區域。使用 儲存貯體的命名慣例搭配識別特定 AWS 區域 （例如`m2-cicd-deployment-source-eu-west-1`，) 的尾碼，並將字首新增`m2-cicd-deployment-source`為參數`DeploymentSourceBucket`的預設值，並使用 AWS CloudFormation 替代函數形成完整儲存貯體，`!Sub {DeploymentSourceBucket}-${AWS::Region}`同時在資源 的`basic-infra.yaml`範本中參考該儲存貯體`SourceSyncLambdaFunction`。

`config_git.yml`

CodeCommit 目錄結構定義。如需詳細資訊，請參閱YAML 觸發檔案 config_git.yml 範例。

`BANKDEMO-source.zip`.

從 CodeCommit 儲存庫建立的 BankDemo 原始程式碼和組態檔案。

`BANKDEMO-exercise.zip`.

從 CodeCommit 儲存庫建立的教學練習的 BankDemo 來源。

建立CI/CD管道基本基礎設施

使用 AWS CloudFormation 範本`basic-infra.yaml`透過 AWS CloudFormation 主控台建立 CI/CD 管道基本基礎設施堆疊。此堆疊會建立 Amazon S3 儲存貯體，您可以在其中上傳應用程式程式碼和資料，以及支援 AWS Lambda 函數，以建立其他必要的資源，例如 AWS CodeCommit 儲存庫和 AWS CodePipeline 管道。

> **Note**
>
> 若要啟動此堆疊，您需要管理 IAM、Amazon S3、Lambda 和 AWS CloudFormation 的許可才能使用 AWS KMS。

1. 登入 AWS Management Console ，並在 https://console.aws.amazon.com/cloudformation 開啟 AWS CloudFormation 主控台。
2. 使用以下選項之一建立新的堆疊：
   - 選擇 Create Stack (建立堆疊)。如果您目前有堆疊正在執行，這是「唯一的」選項。
   - 在堆疊頁面上，選擇建立堆疊。只有當無堆疊執行時才看得到此選項。
3. 在指定範本頁面上：
   - 在準備範本中，選擇範本已就緒。
   - 在指定範本中，選擇 Amazon S3 URL 做為範本來源，然後根據您的 輸入下列其中一個 URLs AWS 區域。
     - `https://m2-us-east-1.s3.us-east-1.amazonaws.com/cicd/mf/basic-infra.yaml`
     - `https://m2-eu-central-1.s3.eu-central-1.amazonaws.com/cicd/mf/basic-infra.yaml`
   - 若要接受您的設定，請選擇下一步。

   建立堆疊頁面隨即開啟。

Specify stack details

Stack name

Stack name

m2-cicd-pipeline-foundation

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

Networking Configuration

Do you want to use an existing VPC in your account?

If you select 'Yes', then you must provide the VPC ID and the Subnet IDs.

No (Create one)

Which VPC ID should be used?

If you selected 'Yes' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

Which private subnet ID should be used?

If you selected 'Yes' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

Which private subnet ID in a different AZ should be used for HA?

If you selected 'Yes' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

Enter the CIDR block that should be used for the new VPC

If you selected 'No (Create one)' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

11.11.0.0/16

CIDR bits for creating subnets. Choose 5 for /27, 6 for /26, 7 for /25, 8 for /24 range

If you selected 'No (Create one)' for UseExistingVPC, this parameter is required. Otherwise, this value will be ignored.

5

Deployment Configuration

Name of the S3 bucket which contains the source files for this stack deployment

Don't change unless you know what you are doing.

m2

Name of the source package file for the infrastructure Lambda function

Don't change unless you know what you are doing.

cicd/mf/m2-code-sync-function.zip

Full URL of the pipeline CloudFormation template file

Don't change unless you know what you are doing.

https://m2-us-east-1.s3.us-east-1.amazonaws.com/cicd/mf/pipeline.yaml

What name prefix to use for the new S3 buckets?

A name prefix for the S3 buckets that will be created by this stack.

mf

Cancel Previous Next

進行下列變更：

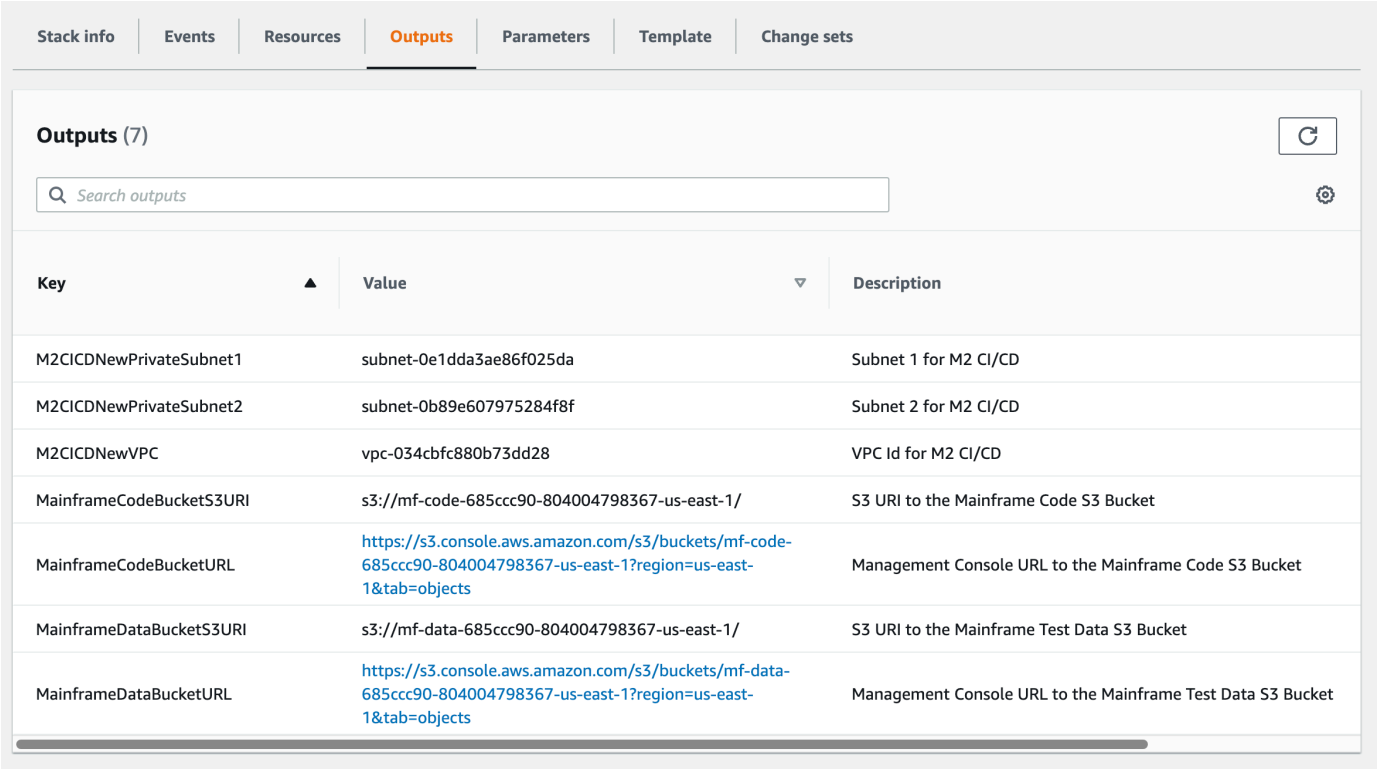
- 為網路組態的堆疊名稱和參數提供適當的值。
- 部署組態中的大多數參數都會預先適當填入，因此您不需要修改它們。根據您的 AWS 區域，將管道 AWS CloudFormation 範本變更為下列其中一個 Amazon S3 URLs。
  - `https://m2-us-east-1.s3.amazonaws.com/cicd/mf/pipeline.yaml`
  - `https://m2-eu-central-1.s3.eu-central-1.amazonaws.com/cicd/mf/pipeline.yaml`
- 選擇下一步。

> **Note**
>
> 除非您已自行修改 AWS CloudFormation 範本，否則請勿變更預設參數值。

4. 在設定堆疊選項中，選擇下一步。
5. 在功能中，選擇我確認 AWS CloudFormation 可能會建立 IAM 資源，以允許 代表您建立 IAM 角色 AWS CloudFormation 的許可。選擇建立堆疊。

> **Note**
>
> 佈建此堆疊可能需要 3 到 5 分鐘。

6. 成功建立堆疊後，導覽至新佈建堆疊的輸出區段。您可以在 Amazon S3 儲存貯體中找到您需要上傳大型主機程式碼和相依檔案的儲存貯體。



建立 AWS CodeCommit 儲存庫和CI/CD管道

在此步驟中，您會呼叫 Lambda 函數來建立 CodeCommit 儲存庫並佈建CI/CD管道堆疊，該函數會呼叫 AWS CloudFormation 來建立管道堆疊。

1. 將 BankDemo 範例應用程式下載到您的本機電腦。
2. `bankdemo.zip` 從本機電腦上傳到 中建立的 Amazon S3 儲存貯體建立CI/CD管道基本基礎設施。
3. 下載 `config_git.yml`。
4. `config_git.yml` 視需要修改 ，如下所示：

   - 新增您自己的目標儲存庫名稱、目標分支和遞交訊息。

   ```
   repository-config:
     target-repository: bankdemo-repo
     target-branch: main
     commit-message: Initial commit for bankdemo-repo main branch
   ```

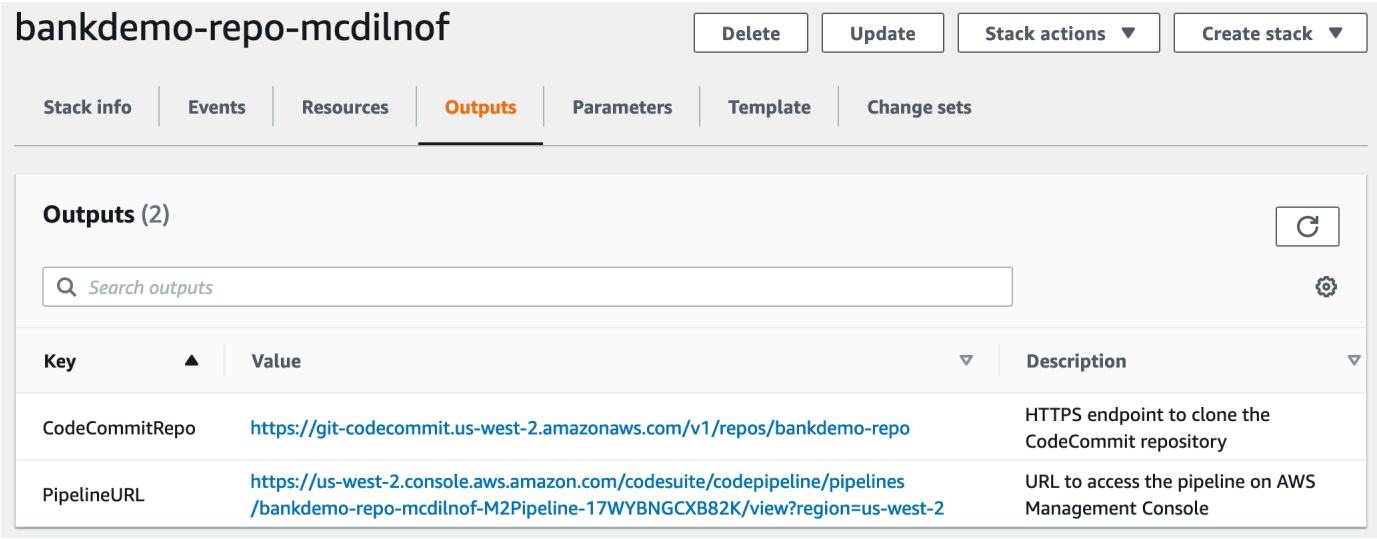
   - 新增您要接收通知的電子郵件地址。

```
pipeline-config:
  # Send pipeline failure notifications to these email addresses
  alert-notifications:
    - myname@mycompany.com
  # Send notifications for manual approval before production deployment to these
 email addresses
  approval-notifications:
    - myname@mycompany.com
```

5. 將包含 CodeCommit 儲存庫資料夾結構定義的`config_git.yml`檔案上傳至 中建立的 Amazon S3 儲存貯體建立CI/CD管道基本基礎設施。這將調用 Lambda 函數，以自動佈建儲存庫和管道。

   這會建立 CodeCommit 儲存庫，其名稱在 `config_git.yml` 檔案中`target-repository`定義的 中提供，例如 `bankdemo-repo`。

   Lambda 函數也會透過 建立CI/CD管道堆疊 AWS CloudFormation。 AWS CloudFormation 堆疊的字首會與提供`target-repository`的名稱相同，後面接著隨機字串 （ 例如 `bankdemo-repo-`*`01234567`*。您可以在 管理主控台中找到 AWS CodeCommit 儲存庫 URL 和 URL，以存取建立的管道。



6. 如果 CodeCommit 儲存庫建立完成，將立即觸發CI/CD管道以執行完整的 CI/CD。
7. 推送檔案後，它將自動觸發管道，該管道將建置、在預備階段部署、執行一些測試並等待手動核准，然後再將其部署在生產環境中。

YAML 觸發檔案 config_git.yml 範例

```
repository-config:
```

```
target-repository: bankdemo-repo
target-branch: main
commit-message: Initial commit for bankdemo-repo main branch
directory-structure:
  - '/':
      files:
      - build.xml
      - '*.yaml'
      - '*.yml'
      - '*.xml'
      - 'LICENSE.txt'
      readme: |
        # Root Folder
        - 'build.xml' : Build configuration for the application
  - tests:
      files:
      - '*.py'
      readme: |
        # Test Folder
        - '*.py' : Test scripts
  - config:
      files:
      - 'BANKDEMO.csd'
      - 'BANKDEMO.json'
      - 'BANKDEMO_ED.json'
      - 'dfhdrdat'
      - 'ESPGSQLXA.dll'
      - 'ESPGSQLXA64.so'
      - 'ESPGSQLXA64_S.so'
      - 'EXTFH.cfg'
      - 'm2-2021-04-28.normal.json'
      - 'MFDBFH.cfg'
      - 'application-definition-template-config.json'
      readme: |
        # Config Folder
        This folder contains the application configuration files.
        - 'BANKDEMO.csd'     : CICS Resource definitions export file
        - 'BANKDEMO.json'    : Enterprise Server configuration
        - 'BANKDEMO_ED.json' : Enterprise Server configuration for ED
        - 'dfhdrdat'         : CICS resource definition file
        - 'ESPGSQLXA.dll'    : XA switch module Windows
        - 'ESPGSQLXA64.so'   : XA switch module Linux
        - 'ESPGSQLXA64_S.so' : XA switch module Linux
        - 'EXTFH.cfg'        : Micro Focus File Handler configuration
```

```
          - 'm2-2021-04-28.normal.json' : M2 request document
          - 'MFDBFH.cfg'       : Micro Focus Database File Handler
          - 'application-definition-template-config.json' : Application definition for
 M2
    - source:
        subdirs:
        - .settings:
            files:
              - '.bms.mfdirset'
              - '.cbl.mfdirset'
        - copybook:
            files:
              - '*.cpy'
              - '*.inc'
            readme: |
              # Copy folder
              This folder contains the source for COBOL copy books, PLI includes, ...
              - .cpy COBOL copybooks
              - .inc PLI includes
#        - ctlcards:
#            files:
#              - '*.ctl'
#              - 'KBNKSRT1.txt'
#            readme: |
#              # Control Card folder
#              This folder contains the source for Batch Control Cards
#              - .ctl Control Cards
        - ims:
            files:
              - '*.dbd'
              - '*.psb'
            readme: |
              # ims folder
              This folder contains the IMS DB source files with the extensions
              - .dbd for IMS DBD source
              - .psb for IMS PSB source
        - jcl:
            files:
              - '*.jcl'
              - '*.ctl'
              - 'KBNKSRT1.txt'
              - '*.prc'
            readme: |
              # jcl folder
```

```
              This folder contains the JCL source files with the extensions
              - .jcl
#        - proclib:
#            files:
#              - '*.prc'
#            readme: |
#              # proclib folder
#              This folder contains the JCL procedures referenced via PROCLIB
 statements in the JCL with extensions
#              - .prc
        - rdbms:
            files:
              - '*.sql'
            readme: |
              # rdbms folder
              This folder contains any DB2 related source files with extensions
              - .sql for any kind of SQL source
        - screens:
            files:
            - '*.bms'
            - '*.mfs'
            readme: |
              # screens folder
              This folder contains the screens source files with the extensions
              - .bms for CICS BMS screens
              - .mfs for IMS MFS screens
            subdirs:
            - .settings:
                files:
                    - '*.bms.mfdirset'
        - cobol:
            files:
              - '*.cbl'
              - '*.pli'
            readme: |
              # source folder
              This folder contains the program source files with the extensions
              - .cbl for COBOL source
              - .pli for PLI source
            subdirs:
            - .settings:
                files:
                   - '*.cbl.mfdirset'
    - tests:
```

```
        files:
        - 'test_script.py'
        readme: |
          # tests Folder
          This folder contains the application test scripts
pipeline-config:
  alert-notifications:
    - myname@mycompany.com
  approval-notifications:
    - myname@mycompany.com
```

建立企業開發人員 AppStream 2.0

若要在 AppStream 2.0 上設定 Rocket Enterprise Developer，請參閱 教學課程：在 AppStream 2.0 上設定 Rocket Enterprise Developer。

若要將 CodeCommit 儲存庫連線至企業開發人員，請使用 `target-repository`中指定的名稱YAML 觸發檔案 config_git.yml 範例。

企業開發人員設定和測試

主題

- 在企業開發人員中複製 BankDemo CodeCommit 儲存庫
- 建立 BankDemo 大型主機 COBOL 專案並建置應用程式
- 建立本機 BankDemo CICS 和批次環境進行測試
- 從企業開發人員啟動 BANKDEMO 伺服器
- 啟動 Rumba 3270 終端機
- 執行 BankDemo 交易
- 從企業開發人員停止 BANKDEMO 伺服器

連線至您在 中建立的企業開發人員 AppStream 2建立企業開發人員 AppStream 2.0.0 執行個體。

1. 從 Windows Start 啟動企業開發人員。選擇 Micro Focus Enterprise Developer，然後選擇 Enterprise Developer for Eclipse。如果您是第一次啟動，可能需要一些時間。
2. 在 Eclipse Launcher 的工作區中：輸入 `C:\Users\<username>\workspace`，然後選擇啟動。

> **Note**
>
> 請務必在重新連線至 AppStream 2.0 執行個體後選擇相同的位置。工作區選擇不是持久性的。

3. 在歡迎中，選擇開啟 COBOL 觀點。這只會在新工作區第一次顯示。

在企業開發人員中複製 BankDemo CodeCommit 儲存庫

1. 選擇視窗/觀點/開放觀點/其他... / Git。
2. 選擇複製 Git 儲存庫。
3. 在複製 Git 儲存庫中，輸入下列資訊：
   - 在位置 URI 中，輸入 CodeCommit 儲存庫的 HTTPS URL。

     > **Note**
     >
     > 在 AWS 管理主控台中複製 CodeCommit 儲存庫的複製 URL HTTPS，並將其貼到這裡。URI 會分割為主機和儲存庫路徑。

   - 驗證使用者和密碼中的使用者 CodeCommit 儲存庫登入資料，然後選擇存放在安全存放區。
4. 在分支選取中，選擇主要分支，然後選擇下一步。
5. 在本機目的地的目錄中，輸入 `C:\Users\<username>\workspace`並選擇完成。

   當 Git 儲存庫檢視中顯示 `BANKDEMO [main]` 時，複製程序即完成。

建立 BankDemo 大型主機 COBOL 專案並建置應用程式

1. 變更為 COBOL 觀點。
2. 在專案中，停用自動建置。
3. 在檔案中，選擇新增，然後選擇大型主機 COBOL 專案。
4. 在新的大型主機 COBOL 專案中，輸入下列資訊：
   - 在專案名稱中，輸入 `BankDemo`。
   - 選擇 Micro Focus 範本 【64 位元】。
   - 選擇 Finish (完成)。

5. 在 COBOL Explorer 中，展開新的 BankDemo 專案。

> **Note**
>
> `[BANKDEMO main]` 方括號中的 表示專案已與本機 BankDemo CodeCommit 儲存庫連線。

6. 如果樹狀檢視未顯示 COBOL 程式、複製手冊、BMS 來源和 JCL 檔案的項目，請從 BankDemo 專案內容功能表中選擇重新整理。
7. 從 BankDemo 內容功能表中，選擇屬性/微型焦點/專案設定/COBOL：
   - 選擇字元集 - ASCII。
   - 選擇套用，然後選擇關閉。
8. 如果建置的 Adobe 和 COBOL 來源未立即開始，請在專案功能表中檢查是否已啟用自動建置選項。

   組建輸出將顯示在主控台檢視中，並應在幾分鐘後完成訊息 `BUILD SUCCESSFUL`和 `Build finished with no errors`。

   現在應編譯 BankDemo 應用程式，並準備好進行本機執行。

建立本機 BankDemo CICS 和批次環境進行測試

1. 在 COBOL Explorer 中，展開 `BANKDEMO / config`。
2. 在編輯器中，開啟 `BANKDEMO_ED.json`。
3. 尋找指向企業開發人員專案的字串`ED_Home=`和變更路徑，如下所示：`D:\\<username>\\workspace\\BANKDEMO`。請注意，在路徑定義中使用雙斜線 (\\)。
4. 儲存並關閉檔案。
5. 選擇 Server Explorer。
6. 從預設內容功能表中，選擇開啟管理頁面。Micro Focus Enterprise Server 管理頁面會在預設瀏覽器中開啟。
7. 僅限 AppStream 2.0 工作階段，請進行下列變更，以便保留本機企業伺服器區域以進行本機測試：
   - 在目錄伺服器/預設中，選擇 PROPERTIES/Configure。
   - 將儲存庫位置取代為 `D:\<username>\My Files\Home Folder\MFDS`。

> **Note**
>
> 每次新的 AppStream 2.0 執行個體連線後，您都必須完成步驟 5 - 8。

8. 在目錄伺服器/預設中，選擇匯入，然後完成下列步驟：

   - 在步驟 1：匯入類型中，選擇 JSON，然後選擇下一步。
   - 在步驟 2：上傳中，按一下以藍色方塊上傳檔案。
   - 在選擇要上傳的檔案中，輸入：
     - 檔案名稱：`D:\<username>\workspace\BANKDEMO\config\BANKDEMO_ED.json`。
     - 選擇 Open (開啟)。
   - 選擇下一步。
   - 在步驟 3：區域清除來自端點的清除連接埠。
   - 選擇下一步。
   - 在步驟 4：匯入中，選擇匯入。
   - 選擇 Finish (完成)。

   清單現在會顯示新的伺服器名稱 BANKDEMO。

從企業開發人員啟動 BANKDEMO 伺服器

1. 選擇企業開發人員。
2. 在 Server Explorer 中，選擇預設，然後從內容功能表中選擇重新整理。

   伺服器清單現在也應該顯示 BANKDEMO。
3. 選擇 BANKDEMO。
4. 從內容功能表中，選擇與專案建立關聯，然後選擇 BANKDEMO。
5. 從內容功能表中，選擇開始。

   主控台檢視應會顯示伺服器啟動的日誌。

   如果`BANKDEMO CASSI5030I PLTPI Phase 2 List(PI) Processing Completed`顯示訊息，表示伺服器已準備好測試 CICS BANKDEMO 應用程式。

啟動 Rumba 3270 終端機

1. 從 Windows Start 啟動 Micro Focus Rumba+ 桌面/Rumba+ 桌面。
2. 在歡迎中，選擇建立新工作階段/大型主機顯示。
3. 在大型主機顯示中，選擇連線/設定。
4. 在工作階段組態中，選擇連線/TN3270。
5. 在主機名稱/地址中，選擇插入並輸入 IP 地址 `127.0.0.1`。
6. 在 Telnet 連接埠中，輸入連接埠 `6000`。
7. 選擇套用。
8. 選擇連線。

   CICS 歡迎畫面會顯示包含第 1 列訊息的畫面：`This is the Micro Focus MFE CICS region BANKDEMO`。
9. 按 CTRL+Shift+Z 清除畫面。

執行 BankDemo 交易

1. 在空白畫面中，輸入 BANK。
2. 在畫面 BANK10 中，在使用者 ID..... : 的輸入欄位中，輸入 `guest`，然後按 Enter。
3. 在畫面 BANK20 中，在計算貸款成本前的輸入欄位中，輸入 `/`（正斜線）並按下 Enter。
4. 在畫面 BANK70 中：
   - 在您想要借用的金額中... : ，輸入 `10000`。
   - 在 中，以 的利率 : 輸入 `5.0`。
   - 在 中，對於幾個月..... : : ，輸入 `10`。
   - 按 Enter。

   應顯示下列結果：

   ```
   Resulting monthly payment.............:   $1023.06
   ```

   這會在企業開發人員中完成 BANKDEMO 應用程式設定。

從企業開發人員停止 BANKDEMO 伺服器

1. 在 Server Explorer 中，選擇預設，然後從內容功能表中選擇重新整理。
2. 選擇 BANKDEMO。
3. 從內容功能表中，選擇停止。

   主控台檢視應顯示伺服器停止的日誌。

   如果`Server: BANKDEMO stopped successfully`顯示訊息，表示伺服器已成功關閉。

練習 1：增強 BANKDEMO 應用程式中的貸款計算

主題

- 將貸款分析規則新增至企業開發人員程式碼分析
- 步驟 1：執行貸款計算的程式碼分析
- 步驟 2：修改 CICSMS 映射和 COBOL 程式和測試
- 步驟 3：在 COBOL 程式中新增總金額計算
- 步驟 4：遞交變更並執行 CI/CD 管道

在此案例中，您將逐步完成對程式碼進行範例變更、部署和測試的程序。

貸款部門希望貸款計算畫面上的新欄位 BANK70 顯示總貸款金額。這需要變更 Adobe 畫面 MBANK70.CBL，新增新的欄位和對應的畫面處理程式 SBANK70P.CBL 與相關的複製手冊。此外，BBANK70P.CBL 中的貸款計算常式需要透過額外的公式延伸。

若要完成此練習，請務必完成下列先決條件。

- 下載 BANKDEMO-exercise.zip 至 `D:\PhotonUser\My Files\Home Folder`。
- 將 zip 檔案解壓縮至 `D:\PhotonUser\My Files\Home Folder\BANKDEMO-exercise`。
- 建立資料夾 `D:\PhotonUser\My Files\Home Folder\AnalysisRules`。
- 將規則檔案`Loan+Calculation+Update.General-1.xml`從 `BANKDEMO-exercise` 資料夾複製到 `D:\PhotonUser\My Files\Home Folder\AnalysisRules`。

> **Note**
>
> *.CBL 和 *.CPY 中的程式碼變更會在本練習的第 1 - 6 欄中以 EXER01 標示。

將貸款分析規則新增至企業開發人員程式碼分析

Rocket Enterprise Analyzer 中定義的分析規則可以從 Enterprise Analyzer 匯出，並匯入企業開發人員，以在企業開發人員專案的來源上執行相同的分析規則。

1. 打開 `Window/Preferences/Micro Focus/COBOL/Code Analysis/Rules`.
2. 選擇編輯...，然後輸入`D:\PhotonUser\My Files\Home Folder\AnalysisRules`包含規則檔案 的資料夾名稱`Loan+Calculation+Update.General-1.xml`。
3. 選擇 Finish (完成)。
4. 選擇套用，然後選擇關閉。
5. 從 BANKDEMO 專案內容功能表中，選擇程式碼分析。

   您應該會看到貸款計算更新的項目。

步驟 1：執行貸款計算的程式碼分析

使用新的分析規則，我們希望識別符合搜尋模式 的 COBOL 程式和程式碼行`*PAYMENT*`，`*LOAN*`以及`*RATE*`表達式、陳述式和變數。這將有助於導覽程式碼並識別必要的程式碼變更。

1. 從 BANKDEMO 專案內容功能表中，選擇程式碼分析/貸款計算更新。

   這將執行搜尋規則，並在稱為程式碼分析的新標籤中列出結果。當右下角的綠色進度列消失時，分析執行即完成。

   程式碼分析索引標籤應會顯示展開的 `BBANK20P.CBL`、 `BBANK70P.CBL`和 清單`SBANK70P.CBL`，每個清單都會列出符合搜尋模式的陳述式、表達式和變數。

   查看 的結果`BBANK20P.CBL`，只有已移動的常值符合搜尋模式。因此，可以忽略此程式。
2. 在索引標籤選單列中，選擇 - 圖示以全部摺疊。
3. 只要按兩下，即可展開`SBANK70P.CBL`並選取任何順序的行，查看這會如何開啟來源，並反白在來源碼中選取的行。您也將識別所有已識別的來源行都已標記。

步驟 2：修改 CICSMS 映射和 COBOL 程式和測試

首先，我們將變更 Adobe 地圖MBANK70.BMS和畫面處理程式SBANK70P.CBL和複製手冊CBANKDAT.CPY，以顯示新欄位。為了避免在本練習中不必要的編碼，修改後的來源模組可在 D:\PhotonUser\My Files\Home Folder\BANKDEMO-exercise\Exercise01 資料夾中使用。一般而言，開發人員會使用程式碼分析結果來導覽和修改來源。如果您有時間和 ，並且想要手動變更，請使用 MBANK70.BMS 和 SBANK70P.CBL （選用）* 中 *手動變更中提供的資訊來執行。

如需快速變更，請複製下列檔案：

1. ..\BANKDEMO-exercise\Exercis01\screens\MBANK70.BMS 至 D:\PhotonUser\workspace\bankdemo\source\screens。
2. .\BANKDEMO-exercise\Exercis01\cobol\SBANK70P.CBL 至 D:\PhotonUser\workspace\bankdemo\source\cobol。
3. ..\BANKDEMO-exercise\Exercis01\copybook\CBANKDAT.CPY 至 D:\PhotonUser\workspace\bankdemo\source\copybook。
4. 若要確保所有受變更影響的程式都已編譯，請選擇專案/清除.../清除所有專案。

如需手動變更 MBANK70.BMS和 SBANK70P.CBL，請完成下列步驟：

- 對於在 PAYMENT 欄位中MBANK70.BMS新增的手動變更，請執行下列動作：
  - 具有與 TXT08 相同屬性和 INITIAL 值「貸款金額總計」的 TXT09 TXT08
  - 具有與 付款相同屬性的總計

測試變更

若要測試變更，請重複下列各節中的步驟：

1. 從企業開發人員啟動 BANKDEMO 伺服器
2. 啟動 Rumba 3270 終端機
3. 執行 BankDemo 交易

   此外，您現在也應該會看到文字 Total Loan Amount.....:。
4. 從企業開發人員停止 BANKDEMO 伺服器

步驟 3：在 COBOL 程式中新增總金額計算

在第二個步驟中，我們將變更BBANK70P.CBL並新增總貸款金額的計算。D:\PhotonUser\My Files\Home Folder\BANKDEMO-exercise\Exercise01 資料夾中提供具有必要變更的預備來源。如果您有時間並想要手動變更，請使用 BBANK70P.CBL （選用）* 中 *手動變更中提供的資訊進行變更。

如需快速變更，請複製下列檔案：

- ..\BANKDEMO-exercise\Exercis01\source\cobol\BBANK70P.CBL 至 D:\PhotonUser\workspace\bankdemo\source\cobol。

若要手動變更 BBANK70P.CBL，請完成下列步驟：

- 使用程式碼分析結果來識別必要的變更。

測試變更

若要測試變更，請重複下列各節中的步驟：

1. 從企業開發人員啟動 BANKDEMO 伺服器
2. 啟動 Rumba 3270 終端機
3. 執行 BankDemo 交易

   此外，您現在也應該會看到文字 Total Loan Amount.....: $10230.60。
4. 從企業開發人員停止 BANKDEMO 伺服器

步驟 4：遞交變更並執行 CI/CD 管道

將變更遞交至中央 CodeCommit 儲存庫，並觸發CI/CD管道來建置、測試和部署變更。

1. 從 BANKDEMO 專案的內容選單中，選擇團隊/承諾。
2. 在 Git 預備索引標籤中，輸入下列遞交訊息：Added Total Amount Calculation。
3. 選擇遞交和推送...。
4. 開啟 CodePipeline 主控台並檢查管道執行的狀態。

> **Note**
>
> 如果您遇到企業開發人員或團隊函數遞交或推送的任何問題，請使用 Git Bash 命令列界面。

練習 2：在 BankDemo 應用程式中擷取貸款計算

主題

- 步驟 1：將貸款計算常式重構為 COBOL 區段
- 步驟 2：將貸款計算常式擷取至獨立 COBOL 程式
- 步驟 3：遞交變更並執行CI/CD管道

在接下來的練習中，您會處理另一個變更請求範例。在此案例中，貸款部門想要將貸款計算常式重複使用為獨立的 WebService。常式應保留在 COBOL 中，並且仍應可從現有的 CICS COBOL 程式 呼叫`BBANK70P.CBL`。

步驟 1：將貸款計算常式重構為 COBOL 區段

在第一個步驟中，我們將貸款計算常式擷取至 COBOL 區段。在下一個步驟中，需要此步驟才能將程式碼擷取到獨立的 COBOL 程式。

1. 在 COBOL 編輯器`BBANK70P.CBL`中開啟 。
2. 在編輯器中，從內容選單中選擇程式碼分析/貸款計算更新。這只會掃描目前來源中分析規則中定義的模式。
3. 在程式碼分析標籤的結果中，尋找第一個算術陳述式 `DIVIDE WS-LOAN-INTEREST BY 12`。
4. 按兩下 陳述式以導覽至編輯器中的來源行。這是貸款計算常式的第一個陳述式。
5. 標記下列程式碼區塊，以便將貸款計算常式擷取至區段。

```
DIVIDE WS-LOAN-INTEREST BY 12
             GIVING WS-LOAN-INTEREST ROUNDED.
           COMPUTE WS-LOAN-MONTHLY-PAYMENT ROUNDED =
             ((WS-LOAN-INTEREST * ((1 + WS-LOAN-INTEREST)
                 ** WS-LOAN-TERM)) /
             (((1 + WS-LOAN-INTEREST) * WS-LOAN-TERM) - 1 ))
                 * WS-LOAN-PRINCIPAL.
```

```
EXER01     COMPUTE WS-LOAN-TOTAL-PAYMENT =
EXER01             (WS-LOAN-MONTHLY-PAYMENT * WS-LOAN-TERM).
```

6. 在編輯器的內容選單中，選擇重構/擷取至區段...。
7. 輸入新區段名稱：LOAN-CALCULATION。
8. 選擇確定。

   標記的程式碼區塊現在已擷取至新LOAN-CALCULATION區段，且程式碼區塊已取代為 PERFROM LOAN-CALCULATION陳述式。

測試變更

若要測試變更，請重複以下各節所述的步驟。

1. 從企業開發人員啟動 BANKDEMO 伺服器
2. 啟動 Rumba 3270 終端機
3. 執行 BankDemo 交易

   此外，您現在也應該會看到文字 `Total Loan Amount.....................: $10230.60`。
4. 從企業開發人員停止 BANKDEMO 伺服器

> **Note**
>
> 如果您想要避免上述步驟將程式碼區塊擷取至區段，您可以將步驟 1 的修改後來源從複製到 `..\BANKDEMO-exercise\Exercis02\Step1\cobol\BBANK70P.CBL D:\PhotonUser\workspace\bankdemo\source\cobol`。

步驟 2：將貸款計算常式擷取至獨立 COBOL 程式

在步驟 2 中， LOAN-CALCULATION區段中的程式碼區塊將擷取至獨立程式，而原始程式碼將取代為程式碼，以呼叫新的子程式。

1. 在編輯器BBANK70P.CBL中開啟 ，並尋找在步驟 1 中建立的新PERFORM LOAN-CALCULATION陳述式。
2. 將游標放在區段名稱內。它會標示為灰色。
3. 從內容功能表中，選取 Refactor->Extract Section/Paragraph to Program...。

4. 在擷取區段/段落至程式中，輸入新的檔案名稱：LOANCALC.CBL。
5. 選擇確定。

   新LOANCALC.CBL程式將在編輯器中開啟。
6. 向下捲動並檢閱要擷取和產生呼叫界面的程式碼。
7. 選取具有 的編輯器BBANK70P.CBL，然後移至 LOAN-CALCULATION SECTION。檢閱正在產生的程式碼，以呼叫新的子程式 LOANCALC.CBL。

> **Note**
>
> CALL 陳述式使用 DFHEIBLK和 DFHCOMMAREA 來LOANCALC呼叫 CICS 控制區塊。因為我們想要將新的LOANCALC.CBL子程式呼叫為非 CICS 程式，所以我們必須透過註解或刪除，從呼叫中移除 DFHEIBLK 和 DFHCOMMAREA 。

測試變更

若要測試變更，請重複以下各節所述的步驟。

1. 從企業開發人員啟動 BANKDEMO 伺服器
2. 啟動 Rumba 3270 終端機
3. 執行 BankDemo 交易

   此外，您現在也應該會看到文字 Total Loan Amount.....: $10230.60。
4. 從企業開發人員停止 BANKDEMO 伺服器

> **Note**
>
> 如果您想要避免上述步驟將程式碼區塊擷取至區段，您可以將步驟 1 的修改後來源從 ..\BANKDEMO-exercise\Exercis02\Step2\cobol\BBANK70P.CBL和 複製到 LOANCALC.CBL D:\PhotonUser\workspace\bankdemo\source\cobol。

步驟 3：遞交變更並執行CI/CD管道

將變更遞交至中央 CodeCommit 儲存庫，並觸發CI/CD管道來建置、測試和部署變更。

1. 在 BANKDEMO 專案的內容選單中，選擇團隊/承諾。
2. 在 Git 預備索引標籤中
   - 在未暫存階段 LOANCALC.CBL 和 LOANCALC.CBL.mfdirset 中新增 。
   - 輸入遞交訊息：`Added Total Amount Calculation`。
3. 選擇遞交和推送...。
4. 開啟 CodePipeline 主控台並檢查管道執行的狀態。

> **Note**
>
> 如果您遇到企業開發人員或團隊函數遞交或推送的任何問題，請使用 Git Bash 命令列界面。

清除資源

如果您不再需要為本教學課程建立的資源，請將其刪除，這樣就不會繼續收取這些資源的費用。請完成下列步驟：

- 刪除 CodePipeline 管道。如需詳細資訊，請參閱AWS CodePipeline 《 使用者指南》中的在 CodePipeline 中刪除管道。
- 刪除 CodeCommit 儲存庫。如需詳細資訊，請參閱AWS CodeCommit 《 使用者指南》中的刪除 CodeCommit 儲存庫。
- 刪除 S3； 儲存貯體。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的刪除儲存貯體。
- 刪除 AWS CloudFormation 堆疊。如需詳細資訊，請參閱AWS CloudFormation 《 使用者指南》中的在AWS CloudFormation 主控台上刪除堆疊。

教學課程：設定 AppStream 2.0 以搭配 Rocket Enterprise Analyzer 和 Rocket Enterprise Developer 使用

AWS Mainframe Modernization 透過 Amazon AppStream 2.0 提供數種工具。AppStream 2.0 是一項全受管、安全的應用程式串流服務，可讓您將桌面應用程式串流給使用者，而無需重寫應用程式。AppStream 2.0 可讓使用者立即存取所需的應用程式，並在他們選擇的裝置上提供回應靈敏、流暢的使用者體驗。使用 AppStream 2.0 託管執行期引擎特定工具，可讓客戶應用程式團隊直接從 Web 瀏覽器使用工具，並與存放在 Amazon S3 儲存貯體或 CodeCommit 儲存庫中的應用程式檔案互動。

如需有關 AppStream 2.0 中瀏覽器支援的資訊，請參閱《Amazon AppStream 2.0 管理指南》中的系統需求和功能支援 (Web 瀏覽器)。如果您在使用 AppStream 2.0 時遇到問題，請參閱《Amazon AppStream 2.0 管理指南》中的對 AppStream 2.0 使用者問題進行故障診斷。 Amazon AppStream 2.0

本文件適用於客戶營運團隊成員。它說明如何設定 Amazon AppStream 2.0 機群和堆疊來託管與 AWS Mainframe Modernization 搭配使用的 Rocket Enterprise Analyzer 和 Rocket Enterprise Developer 工具。Rocket Enterprise Analyzer 通常在評估階段使用，Rocket Enterprise Developer 通常在 AWS 大型主機現代化方法的遷移和現代化階段使用。如果您打算同時使用 Enterprise Analyzer 和 Enterprise Developer，則必須為每個工具建立單獨的機群和堆疊。每個工具都需要自己的機群和堆疊，因為其授權條款不同。

> **Important**
>
> 本教學課程中的步驟是以可下載的 AWS CloudFormation 範本 cfn-m2-appstream-fleet-ea-ed.yml 為基礎。

主題

- 先決條件
- 步驟 1：取得 AppStream 2.0 映像
- 步驟 2：使用 AWS CloudFormation 範本建立堆疊
- 步驟 3：在 AppStream 2.0 中建立使用者
- 步驟 4：登入 AppStream 2.0
- 步驟 5：驗證 Amazon S3 中的儲存貯體 （選用）
- 後續步驟
- 清除資源

先決條件

- 下載範本：cfn-m2-appstream-fleet-ea-ed.yml。
- 取得預設 VPC 和安全群組的 ID。如需預設 VPC 的詳細資訊，請參閱《Amazon VPCs》中的預設 VPC。 如需預設安全群組的詳細資訊，請參閱《Amazon EC2 使用者指南》中的預設和自訂安全群組。
- 請確定您具有下列許可：

- 在 AppStream 2.0 中建立堆疊、機群和使用者。
- AWS CloudFormation 使用範本在 中建立堆疊。
- 建立儲存貯體並將檔案上傳至 Amazon S3 中的儲存貯體。
- 從 IAM 下載登入資料 (`access_key_id` 和 `secret_access_key`)。

步驟 1：取得 AppStream 2.0 映像

在此步驟中，您會與 AWS 帳戶共用適用於 Enterprise Analyzer 和 Enterprise Developer 的 AppStream 2.0 映像。

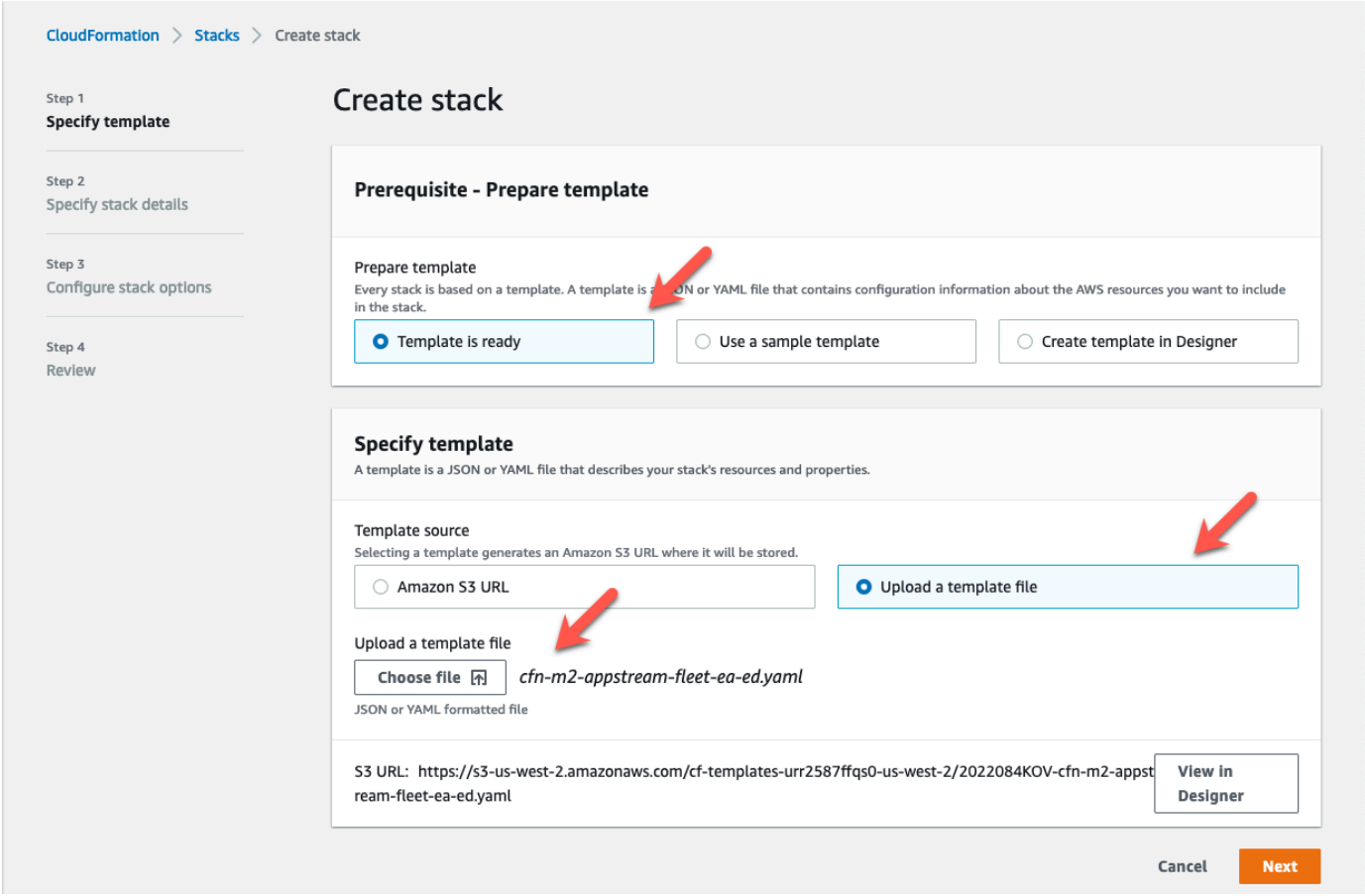
1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 在左側導覽中，選擇工具。
3. 在分析、開發和建置資產中，選擇與我的 AWS 帳戶共用資產。

步驟 2：使用 AWS CloudFormation 範本建立堆疊

在此步驟中，您會使用下載的 AWS CloudFormation 範本來建立 AppStream 2.0 堆疊和機群，以執行 Rocket Enterprise Analyzer。您可以稍後重複此步驟來建立另一個 AppStream 2.0 堆疊和機群，以執行 Rocket Enterprise Developer，因為每個工具都需要在 AppStream 2.0 中自己的機群和堆疊。如需 AWS CloudFormation 堆疊的詳細資訊，請參閱AWS CloudFormation 《 使用者指南》中的使用堆疊。

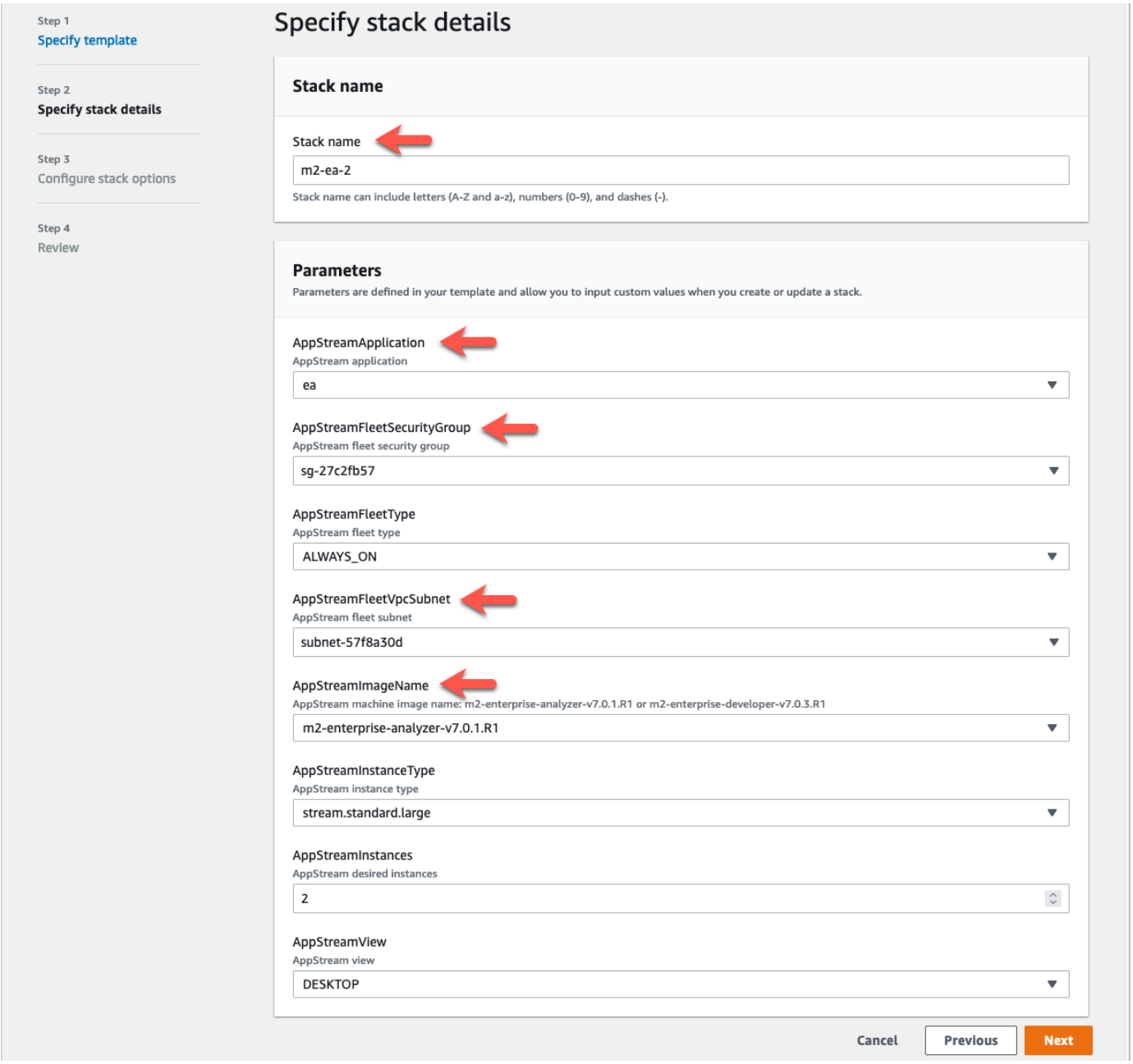
> **Note**
>
> AWS Mainframe Modernization 會針對使用 Enterprise Analyzer 和 Enterprise Developer 的標準 AppStream 2.0 定價增加額外費用。如需詳細資訊，請參閱AWS 大型主機現代化定價。

1. 如有必要，請下載 cfn-m2-appstream-fleet-ea-ed.yml 範本。
2. 開啟 AWS CloudFormation 主控台，然後選擇使用新資源建立堆疊 和 （ 標準 ）。
3. 在先決條件 - 準備範本中，選擇範本已就緒。
4. 在指定範本中，選擇上傳範本檔案。
5. 在上傳範本檔案中，選擇選擇檔案並上傳 cfn-m2-appstream-fleet-ea-ed.yml 範本。
6. 選擇下一步。



7. 在指定堆疊詳細資訊上，輸入下列資訊：

   - 在堆疊名稱中，輸入您選擇的名稱。例如 **`m2-ea`**。
   - 在 AppStreamApplication 中，選擇 ea。
   - 在 AppStreamFleetSecurityGroup 中，選擇預設 VPC 的預設安全群組。
   - 在 AppStreamFleetVpcSubnet 中，選擇預設 VPC 內的子網路。
   - 在 AppStreamImageName 中，選擇以 開頭的影像`m2-enterprise-analyzer`。此映像包含目前支援的 Rocket Enterprise Analyzer 工具版本。
   - 接受其他欄位的預設值，然後選擇下一步。



8. 接受所有預設值，然後再次選擇下一步。
9. 檢閱時，請確定所有參數都是您想要的。
10. 捲動至底部，選擇我確認 AWS CloudFormation 可能會使用自訂名稱建立 IAM 資源，然後選擇建立堆疊。

堆疊和機群需要 20 到 30 分鐘才能建立。您可以選擇重新整理，在 AWS CloudFormation 事件發生時查看事件。

步驟 3：在 AppStream 2.0 中建立使用者

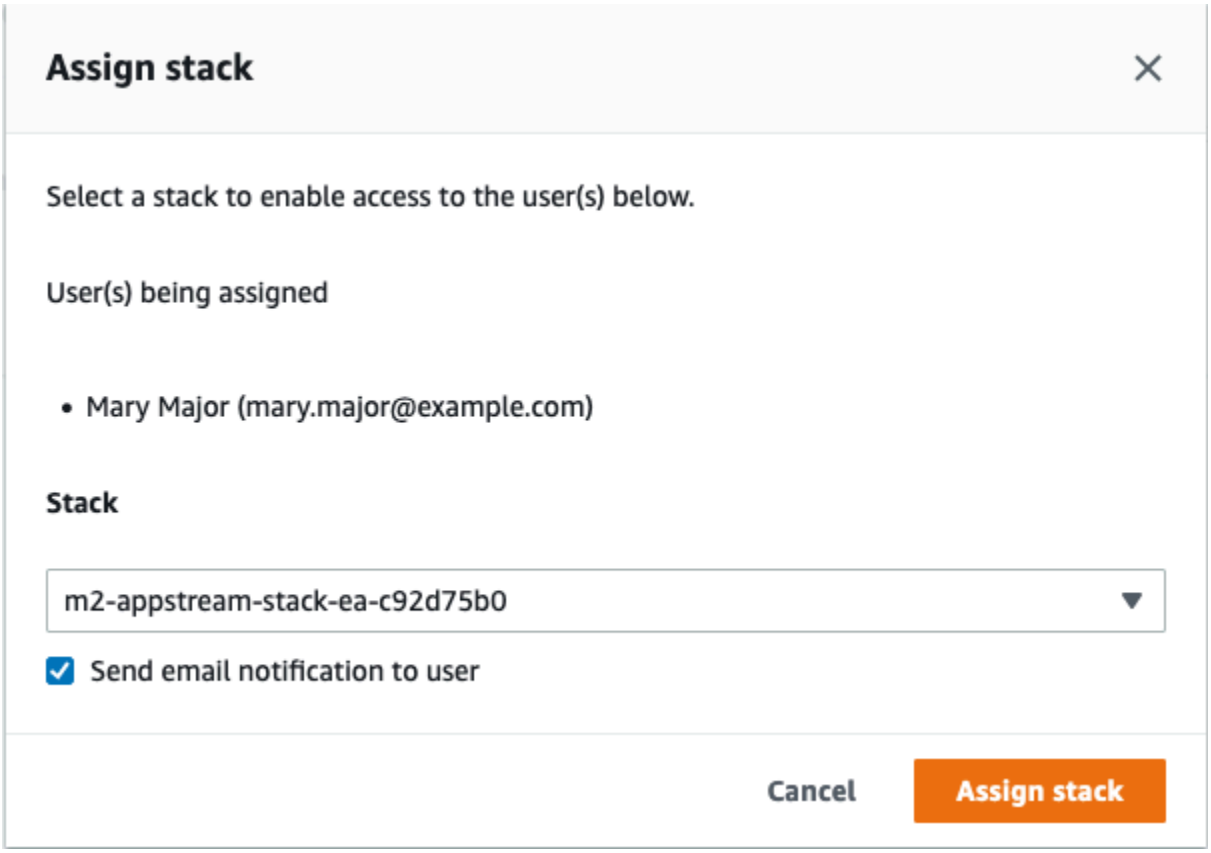
當您等待 AWS CloudFormation 完成堆疊的建立時，您可以在 AppStream 2.0 中建立一或多個使用者。這些使用者將在 AppStream 2.0 中使用 Enterprise Analyzer。您將需要為每個使用者指定電子郵件地址，並確保每個使用者都有足夠的許可在 Amazon S3 中建立儲存貯體、將檔案上傳至儲存貯體，以及連結至儲存貯體以映射其內容。

1. 開啟 AppStream 2.0 主控台。
2. 在左側導覽中，選擇使用者集區。
3. 選擇 Create user (建立使用者)。
4. 提供電子郵件地址，讓使用者可以接收電子郵件邀請，以使用 AppStream 2.0、名字和姓氏，然後選擇建立使用者。
5. 如有必要，請重複 以建立更多使用者。每個使用者的電子郵件地址必須是唯一的。

如需建立 AppStream 2.0 使用者的詳細資訊，請參閱《Amazon AppStream 2.0 管理指南》中的 AppStream 2.0 使用者集區。 Amazon AppStream 2.0

AWS CloudFormation 完成建立堆疊後，您可以將您建立的使用者指派給堆疊，如下所示：

1. 開啟 AppStream 2.0 主控台。
2. 選擇使用者名稱。
3. 選擇動作，然後選擇指派堆疊。
4. 在指派堆疊中，選擇開頭為 的堆疊`m2-appstream-stack-ea`。
5. 選擇 Assign stack (指派堆疊)。



將使用者指派給堆疊會導致 AppStream 2.0 根據您提供的地址傳送電子郵件給使用者。此電子郵件包含 AppStream 2.0 登入頁面的連結。

步驟 4：登入 AppStream 2.0

在此步驟中，您會使用 AppStream 2.0 傳送給您在 中建立之使用者的電子郵件中的連結登入 AppStream 2.0。 步驟 3：在 AppStream 2.0 中建立使用者

1. 使用 AppStream 2.0 傳送的電子郵件中提供的連結登入 AppStream 2.0。
2. 如果出現提示，請變更您的密碼。您看到的 AppStream 2.0 畫面類似如下：



3. 選擇桌面。
4. 在任務列上，選擇搜尋並輸入 **D:** 以導覽至主資料夾。

> **Note**
>
> 如果您略過此步驟，當您嘗試存取主資料夾`Device not ready`時，可能會發生錯誤。

在任何時候，如果您無法登入 AppStream 2.0，您可以使用下列步驟重新啟動 AppStream 2.0 機群並嘗試再次登入。

1. 開啟 AppStream 2.0 主控台。
2. 在左側導覽中，選擇機群。
3. 選擇您嘗試使用的機群。
4. 選擇動作，然後選擇停止。
5. 等待機群停止。
6. 選擇動作，然後選擇開始。

此程序可能需要約 10 分鐘的時間。

步驟 5：驗證 Amazon S3 中的儲存貯體 （選用）

您用來建立堆疊的 AWS CloudFormation 範本所完成的其中一項任務是在 Amazon S3 中建立兩個儲存貯體，這些儲存貯體是跨工作工作階段儲存和還原使用者資料和應用程式設定的必要項目。這些儲存貯體如下所示：

- 名稱以 開頭`appstream2-`。此儲存貯體會將資料映射至 AppStream 2.0 (`D:\PhotonUser\My Files\Home Folder`) 中的主資料夾。

  > **Note**
  >
  > 主資料夾對於指定的電子郵件地址是唯一的，並且會在指定 AWS 帳戶中的所有機群和堆疊之間共用。Home Folder 的名稱是使用者電子郵件地址的 SHA256 雜湊，並根據該雜湊存放在路徑上。

- 名稱以 開頭`appstream-app-settings-`。此儲存貯體包含 AppStream 2.0 的使用者工作階段資訊，並包含瀏覽器我的最愛、IDE 和應用程式連線設定檔，以及 UI 自訂等設定。如需詳細資訊，請參閱《Amazon AppStream 2.0 管理指南》中的應用程式設定持續性的運作方式。

若要驗證儲存貯體是否已建立，請依照下列步驟執行：

1. 開啟 Amazon S3 主控台。
2. 在左側導覽中，選擇儲存貯體。
3. 在依名稱尋找儲存貯體中，輸入 **`appstream`**來篩選清單。

如果您看到儲存貯體，則不需要進一步的動作。請注意，儲存貯體存在。如果您沒有看到儲存貯體，則表示 AWS CloudFormation 範本未完成執行，或發生錯誤。前往 AWS CloudFormation 主控台並檢閱堆疊建立訊息。

後續步驟

現在已設定 AppStream 2.0 基礎設施，您可以設定並開始使用 Enterprise Analyzer。如需詳細資訊，請參閱教學課程：在 AppStream 2.0 上設定 Enterprise Analyzer。您也可以設定企業開發人員。如需詳細資訊，請參閱教學課程：在 AppStream 2.0 上設定 Rocket Enterprise Developer。

清除資源

建立 AppStream 2.0 機群和堆疊中說明了清除已建立堆疊和機群的程序。

刪除 AppStream 2.0 物件後，帳戶管理員也可以在適當時清除應用程式設定和主資料夾的 Amazon S3 儲存貯體。

> **Note**
>
> 指定使用者的主資料夾在所有機群中都是唯一的，因此如果同一帳戶中有其他 AppStream 2.0 堆疊處於作用中狀態，您可能需要保留它。

最後，AppStream 2.0 目前不允許您使用主控台刪除使用者。反之，您必須使用服務 API 搭配 CLI。如需詳細資訊，請參閱《Amazon AppStream 2.0 管理指南》中的使用者集區管理。 Amazon AppStream 2.0

教學課程：搭配 Rocket Enterprise Developer （ 先前為 Micro Focus Enterprise Developer) 使用 範本

本教學課程說明如何搭配 Rocket Enterprise Developer 使用 範本和預先定義的專案。它涵蓋三個使用案例。所有使用案例都使用 BankDemo 範例所提供的範本程式碼。若要下載範例，請選擇 `bankdemo.zip` 。

> **Important**
>
> 如果您使用適用於 Windows 的 Enterprise Developer 版本，則編譯器產生的二進位檔只能在企業開發人員提供的 Enterprise Server 上執行。您無法在以 Linux 為基礎的 AWS Mainframe Modernization 執行時間下執行它們。

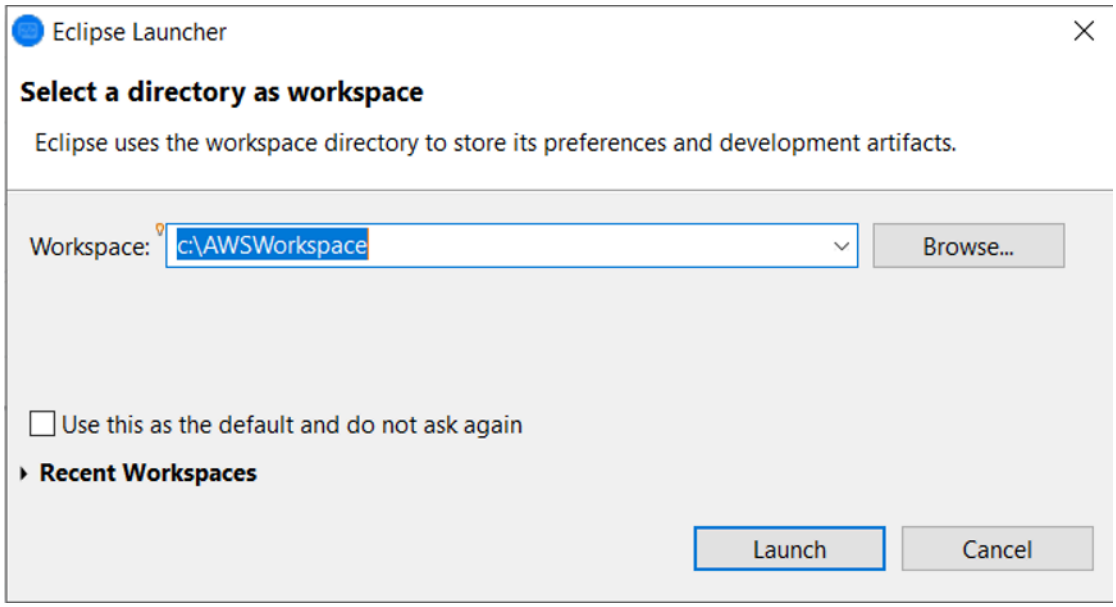
主題

- 使用案例 1 - 使用包含來源元件的 COBOL 專案範本
- 使用案例 2 - 使用不含來源元件的 COBOL 專案範本
- 使用案例 3 - 使用連結至來源資料夾的預先定義 COBOL 專案
- 使用區域定義 JSON 範本

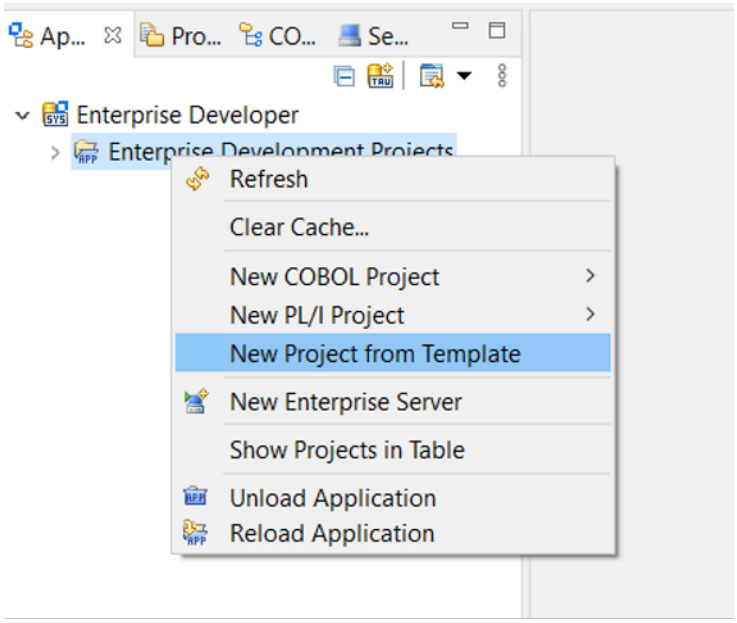
使用案例 1 - 使用包含來源元件的 COBOL 專案範本

此使用案例需要您將來源元件複製到範本目錄結構，作為示範預先設定步驟的一部分。在中，`bankdemo.zip`這已從原始`AWSTemplates.zip`交付變更，以避免有兩個來源複本。

1. 啟動企業開發人員並指定選擇的工作區。

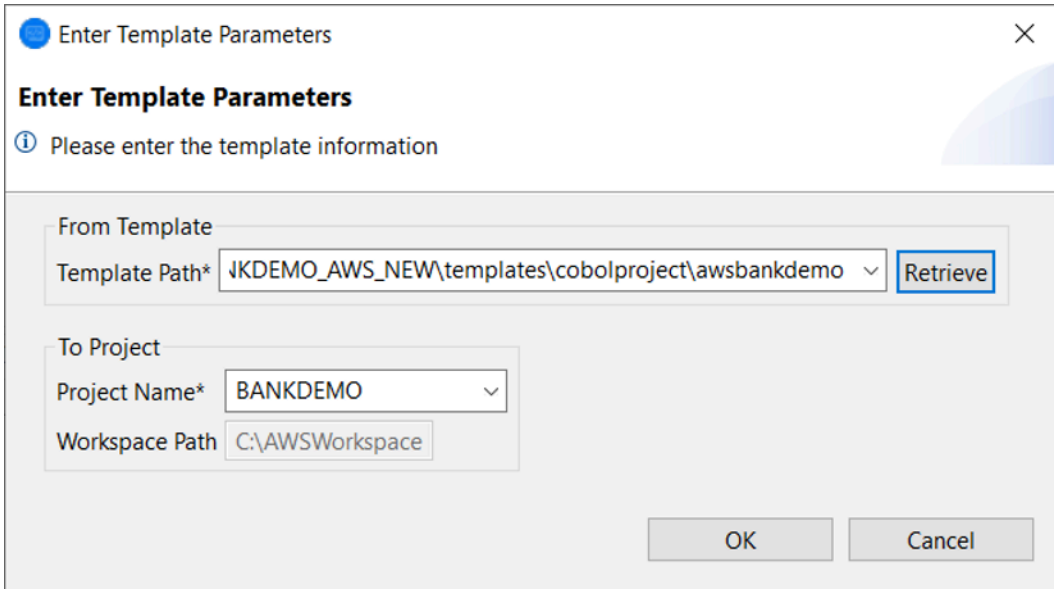


2. 在 Application Explorer 檢視中，從企業開發專案樹狀檢視項目中，從內容功能表中選擇範本中的新專案。

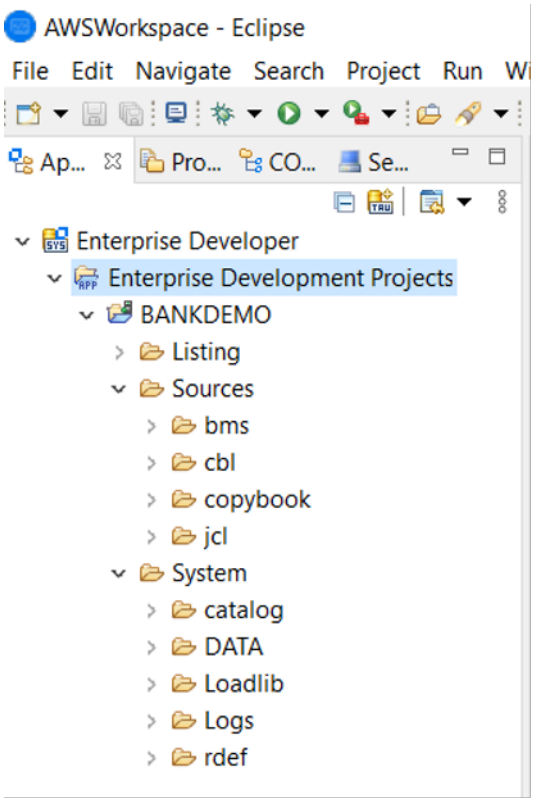


3. 輸入範本參數，如下所示。

> **Note**
>
> 範本路徑會參照 ZIP 的擷取位置。



4. 選擇確定會根據提供的範本建立本機開發 Eclipse 專案，並具有完整的來源和執行環境結構。



System 結構包含一個完整的資源定義檔案，其中包含 BANKDEMO 的必要項目、新增項目的必要目錄，以及對應的 ASCII 資料檔案。

由於來源範本結構包含所有來源項目，因此這些檔案會複製到本機專案，因此會自動建置在企業開發人員中。

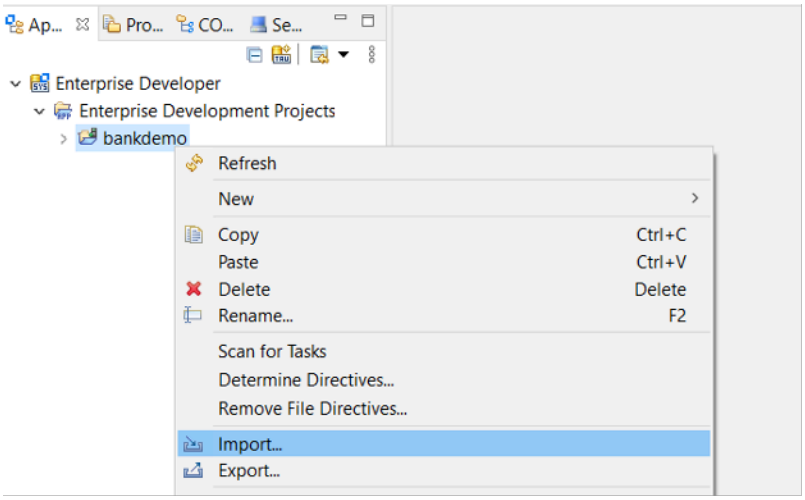
使用案例 2 - 使用不含來源元件的 COBOL 專案範本

步驟 1 到 3 與 相同使用案例 1 - 使用包含來源元件的 COBOL 專案範本。

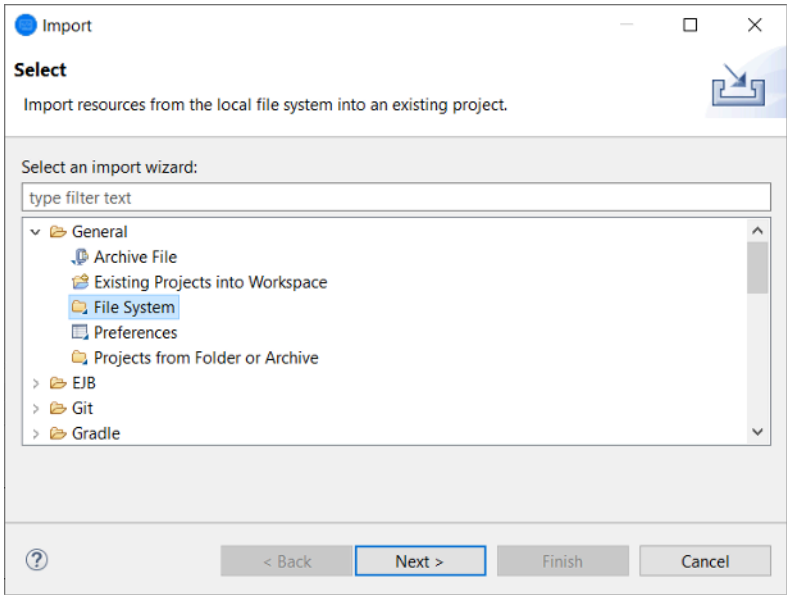
此使用案例中的System結構也包含完整的資源定義檔案，其中包含 BankDemo 所需的項目、新增項目的必要目錄，以及對應的 ASCII 資料檔案。

不過，範本來源結構不包含任何元件。您必須從使用的任何來源儲存庫將這些項目匯入專案。

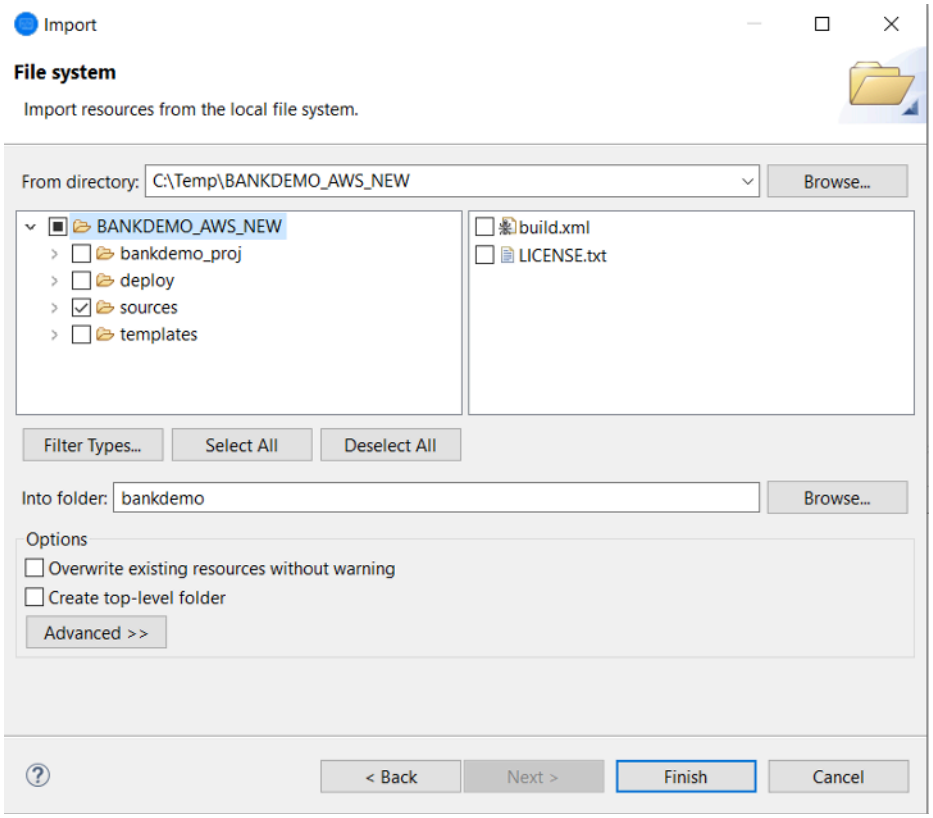
1. 選擇專案名稱。從相關內容功能表中，選擇匯入。



2. 在產生的對話方塊中，於一般區段下，選擇檔案系統，然後選擇下一步。



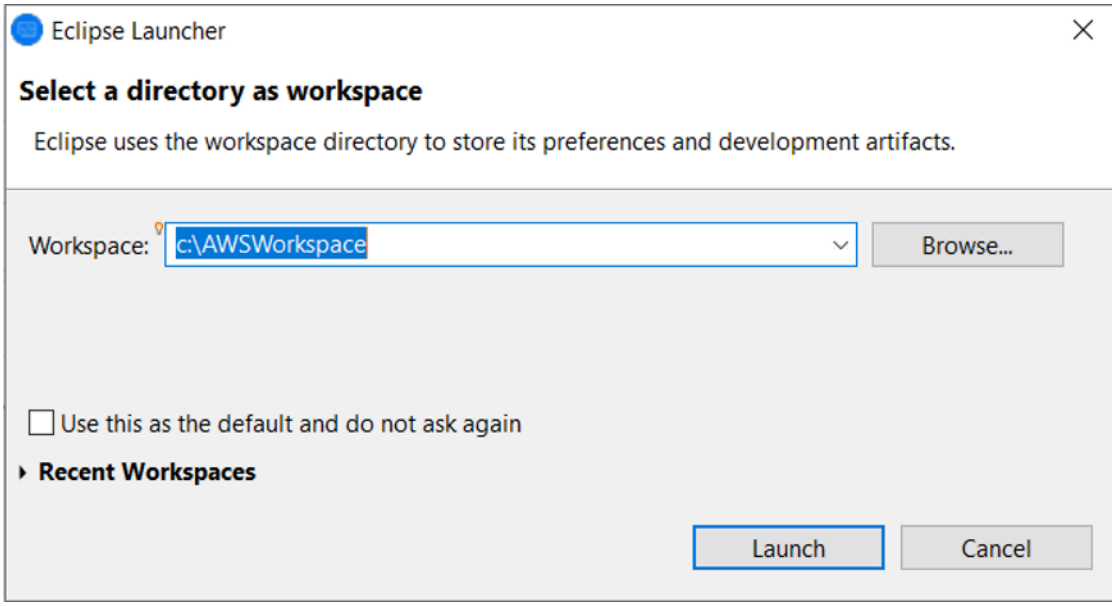
3. 瀏覽檔案系統以指向儲存庫資料夾，填入從目錄欄位。選擇您要匯入的所有資料夾，例如 `sources`。`Into folder` 欄位將預先填入。選擇 Finish (完成)。



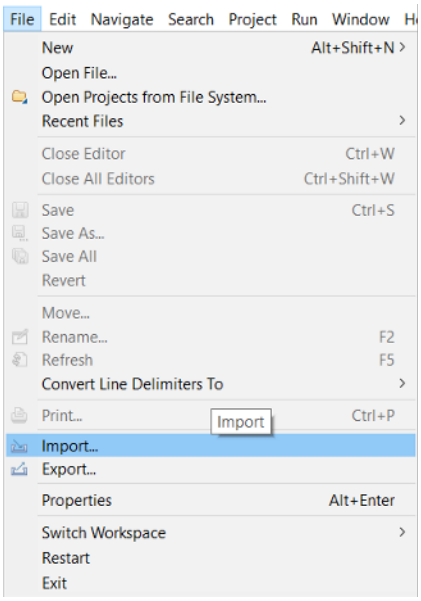
在來源範本結構包含所有來源項目之後，它們會自動建置在企業開發人員中。

使用案例 3 - 使用連結至來源資料夾的預先定義 COBOL 專案

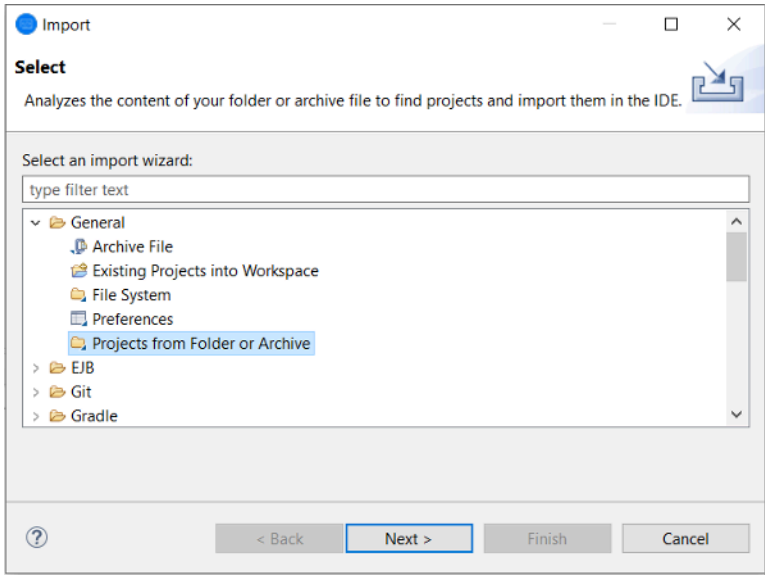
1. 啟動企業開發人員並指定選擇的工作區。



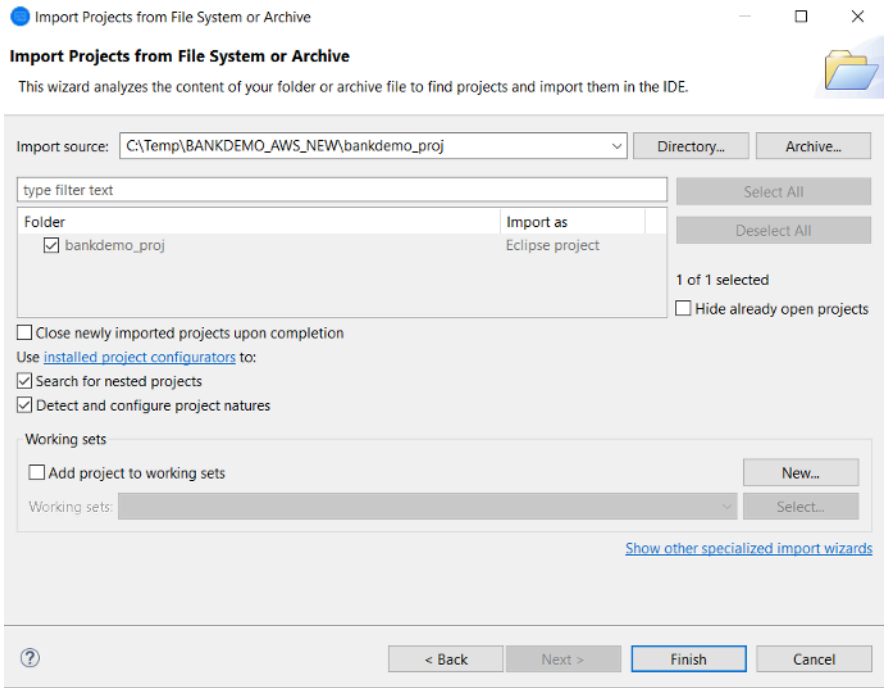
2. 從 File (檔案) 功能表中，選擇 Import (匯入)。



3. 從產生的對話方塊中，在一般下，從資料夾或封存中選擇專案，然後選擇下一步。



4. 填入匯入來源、選擇目錄並瀏覽檔案系統，以選取預先定義的專案資料夾。中包含的專案具有相同儲存庫中來源資料夾的連結。

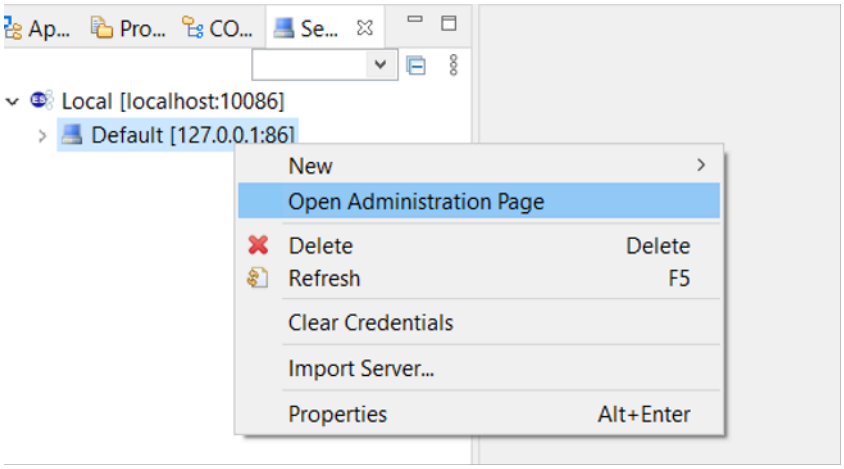


選擇 Finish (完成)。

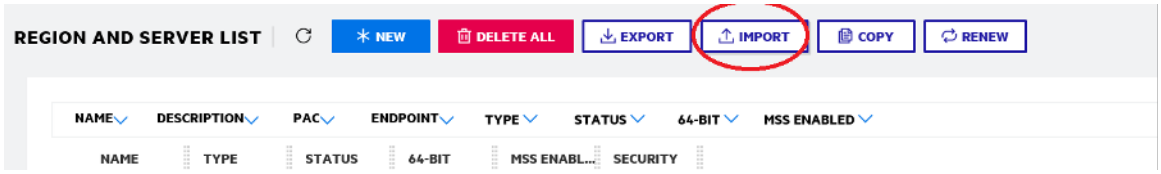
由於專案是由來源資料夾的連結填入，因此會自動建置程式碼。

使用區域定義 JSON 範本

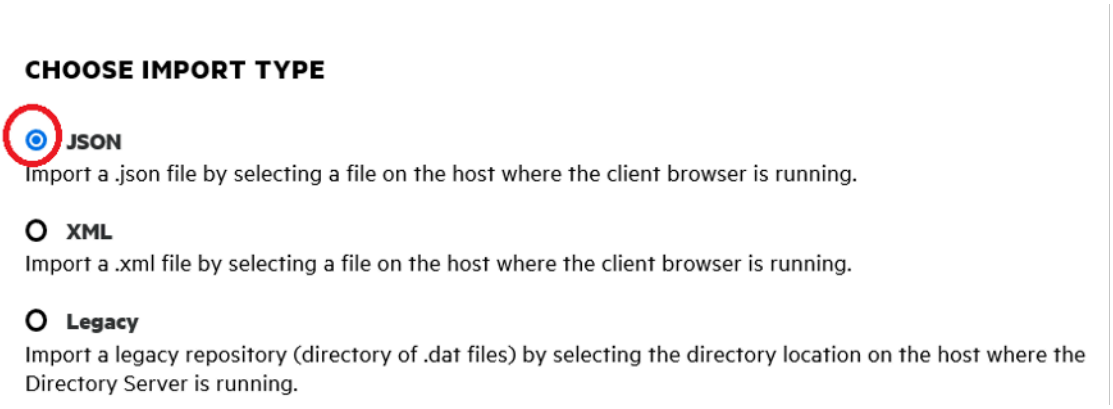
1. 切換到 Server Explorer 檢視。從相關內容功能表中，選擇開啟管理頁面，這會啟動預設瀏覽器。



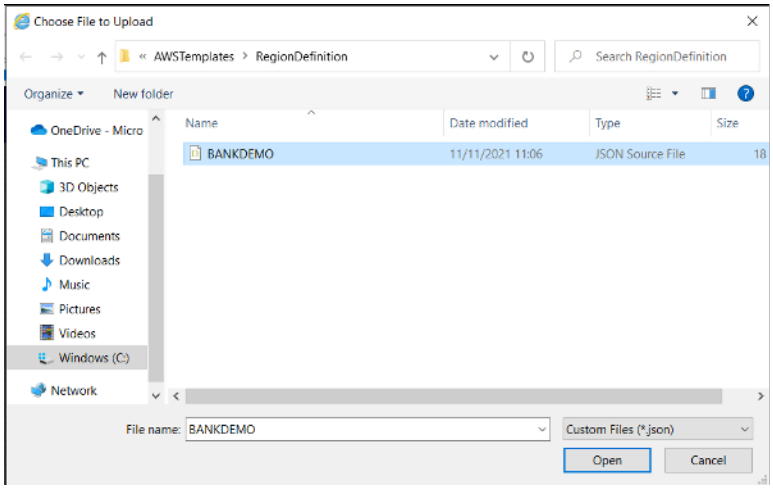
2. 從產生的 Enterprise Server Common Web Administration (ESCWA) 畫面中，選擇匯入 。



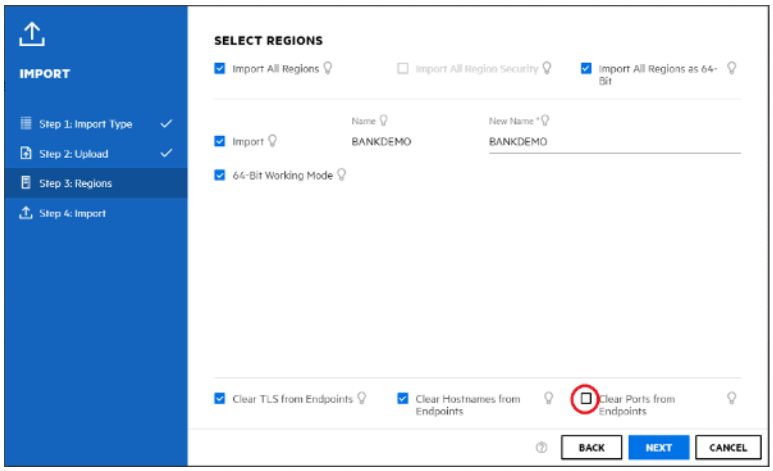
3. 選擇 JSON 匯入類型，然後選擇下一步。



4. 上傳提供的BANKDEMO.JSON檔案。

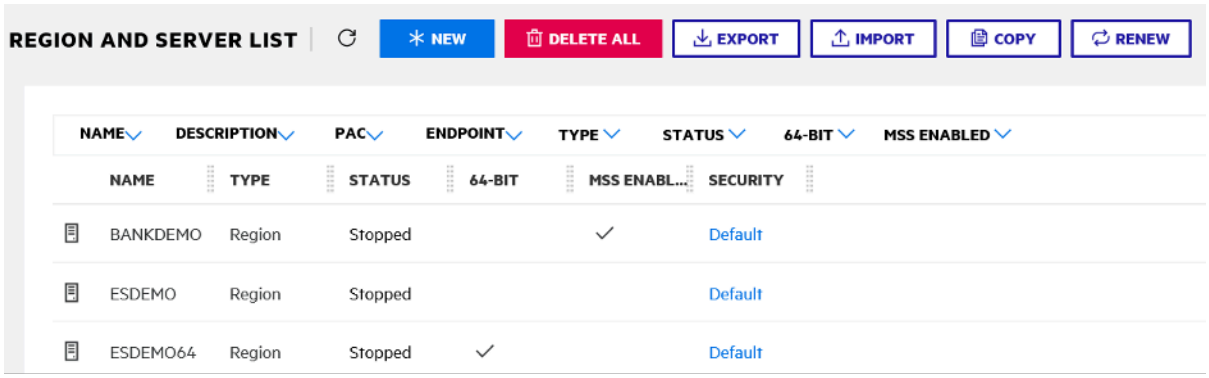


選取後，選擇下一步。



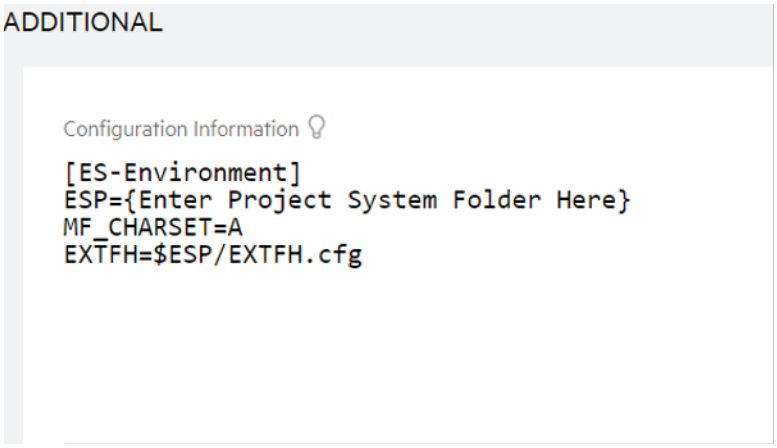
在選取區域面板上，確保未選取從端點清除連接埠選項，然後繼續選擇透過面板下一步，直到顯示執行匯入面板。然後從左側導覽窗格中選擇匯入。

最後按一下完成。然後 BANKDEMO 區域會新增至伺服器清單。



5. 導覽至 BANKDEMO 區域的一般屬性。

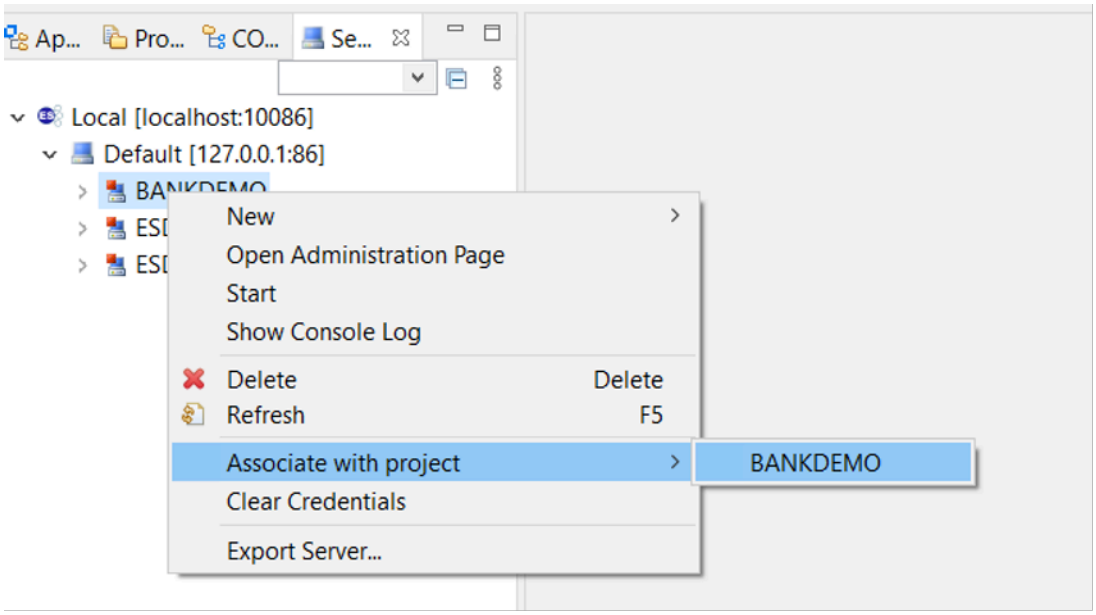
6. 捲動至組態區段。
7. ESP 環境變數需要設定為與先前步驟中建立的 Eclipse 專案相關的`System`資料夾。這應該是 `workspacefolder/projectname/System`。



8. 按一下 Apply (套用)。

   區域現在已完全設定為與 Eclipse COBOL 專案一起執行。

9. 最後，回到企業開發人員，將匯入的區域與專案建立關聯。



企業開發人員環境現在已準備就緒，可搭配完整的 BankDemo 工作版本使用。您可以針對區域編輯、編譯和偵錯程式碼。

> **Important**
>
> 如果您使用適用於 Windows 的 Enterprise Developer 版本，則編譯器產生的二進位檔只能在企業開發人員提供的 Enterprise Server 上執行。您無法在以 Linux 為基礎的 AWS Mainframe Modernization 執行時間下執行它們。

教學課程：在 AppStream 2.0 上設定 Enterprise Analyzer

本教學課程說明如何設定 Rocket Enterprise Analyzer （先前稱為 Micro Focus Enterprise Analyzer) 來分析一或多個大型主機應用程式。Enterprise Analyzer 工具會根據其對應用程式原始碼和系統定義的分析，提供數個報告。

此設定旨在促進團隊協作。安裝會使用 Amazon S3 儲存貯體，與虛擬磁碟共用原始程式碼。這樣做會在 Windows 機器上使用 Rclone)。使用執行 PostgreSQL 的常見 Amazon RDS 執行個體，團隊的任何成員都可以存取所有請求的報告。

團隊成員也可以將虛擬 Amazon S3 支援的磁碟掛載在其個人機器上。 並從工作站更新來源儲存貯體。如果指令碼連接到其他內部部署系統，他們可能會在機器上使用指令碼或任何其他形式的自動化。

設定是以 AWS 大型主機現代化與客戶共用的 AppStream 2.0 Windows 映像為基礎。設定也以建立 AppStream 2.0 機群和堆疊為基礎，如 中所述教學課程：設定 AppStream 2.0 以搭配 Rocket Enterprise Analyzer 和 Rocket Enterprise Developer 使用。

> **Important**
>
> 本教學課程中的步驟假設您使用可 AWS CloudFormation 下載的範本 cfn-m2-appstream-fleet-ea-ed.yml 設定 AppStream 2.0。如需詳細資訊，請參閱教學課程：設定 AppStream 2.0 以搭配 Rocket Enterprise Analyzer 和 Rocket Enterprise Developer 使用。
> 若要執行本教學課程中的步驟，您必須已設定 Enterprise Analyzer 機群和堆疊，且必須正在執行。

如需 Enterprise Analyzer 功能和交付項目的完整說明，請參閱 Rocket Software （先前稱為 Micro Focus) 網站上的 Enterprise Analyzer 文件。

影像內容

除了 Enterprise Analyzer 應用程式本身之外，映像還包含下列工具和程式庫。

第三方工具

- Python
- Rclone
- pgAdmin
- git-scm
- PostgreSQL ODBC 驅動程式

中的程式庫 `C:\Users\Public`

- 企業開發人員的 BankDemo 原始程式碼和專案定義：`m2-bankdemo-template.zip`。
- 大型主機的 MFA 安裝套件：`mfa.zip`。如需詳細資訊，請參閱 Micro Focus Enterprise Developer 文件中的大型主機存取概觀。
- Rclone 的命令和組態檔案 （教學課程中的指示）： `m2-rclone.cmd`和 `m2-rclone.conf`。

主題

- 先決條件
- 步驟 1：設定
- 步驟 2：在 Windows 上建立以 Amazon S3 為基礎的虛擬資料夾
- 步驟 3：建立 Amazon RDS 執行個體的 ODBC 來源
- 後續工作階段
- 工作區連線故障診斷
- 清除資源

先決條件

- 將您要分析之客戶應用程式的原始碼和系統定義上傳至 S3 儲存貯體。系統定義包括 CICS CSD、DB2 物件定義等。您可以在 儲存貯體中建立資料夾結構，以決定您想要如何組織應用程式成品。例如，當您解壓縮 BankDemo 範例時，其具有下列結構：

```
demo
     |--> jcl
     |--> RDEF
     |--> transaction
```

```
    |--> xa
```

- 建立並啟動執行 PostgreSQL 的 Amazon RDS 執行個體。此執行個體將存放 Enterprise Analyzer 產生的資料和結果。您可以與應用程式團隊的所有成員共用此執行個體。此外，在資料庫中建立名為 `m2_ea`（或任何其他適當名稱）的空白結構描述。為授權使用者定義登入資料，允許他們在此結構描述中建立、插入、更新和刪除項目。您可以從 Amazon RDS 主控台或帳戶管理員取得資料庫名稱、其伺服器端點 URL 和 TCP 連接埠。
- 請確定您已設定 的程式設計存取權 AWS 帳戶。如需詳細資訊，請參閱《》中的程式設計存取Amazon Web Services 一般參考。

步驟 1：設定

1. 使用 AppStream 2.0 開始工作階段，其中包含您在 AppStream 2.0 歡迎電子郵件訊息中收到的 URL。
2. 使用您的電子郵件做為您的使用者 ID，並定義您的永久密碼。
3. 選取 Enterprise Analyzer 堆疊。
4. 在 AppStream 2.0 選單頁面上，選擇桌面以到達機群正在串流的 Windows 桌面。

步驟 2：在 Windows 上建立以 Amazon S3 為基礎的虛擬資料夾

> **Note**
>
> 如果您已在 AWS Mainframe Modernization 預覽期間使用 Rclone，則必須更新`m2-rclone.cmd`至位於 的較新版本`C:\Users\Public`。

1. `C:\Users\PhotonUser\My Files\Home Folder` 使用 `m2-rclone.cmd` File Explorer 將 中提供的 `m2-rclone.conf`和 檔案`C:\Users\Public`複製到主資料夾。
2. 使用 AWS 存取金鑰和對應的秘密以及 `m2-rclone.conf` 更新組態參數 AWS 區域。

```
[m2-s3]
type = s3
provider = AWS
access_key_id = YOUR-ACCESS-KEY
secret_access_key = YOUR-SECRET-KEY
region = YOUR-REGION
acl = private
```

```
server_side_encryption = AES256
```

3. 在 m2-rclone.cmd 中，進行下列變更：

   - amzn-s3-demo-bucket 變更為您的 Amazon S3 儲存貯體名稱。例如 m2-s3-mybucket。
   - your-s3-folder-key 變更為 Amazon S3 儲存貯體金鑰。例如 myProject。
   - your-local-folder-path 將 變更為您要從包含它們的 Amazon S3 儲存貯體同步應用程式檔案的目錄路徑。例如 D:\PhotonUser\My Files\Home Folder\m2-new。此同步目錄必須是主資料夾的子目錄，以便 AppStream 2.0 在工作階段開始和結束時正確備份和還原。

```
:loop
timeout /T 10
"C:\Program Files\rclone\rclone.exe" sync m2-s3:amzn-s3-demo-bucket/your-s3-
folder-key "D:\PhotonUser\My Files\Home Folder\your-local-folder-path" --config "D:
\PhotonUser\My Files\Home Folder\m2-rclone.conf"
goto :loop
```

4. 視需要開啟 Windows 命令提示字元，cd 至 C:\Users\PhotonUser\My Files\Home Folder並執行 m2-rclone.cmd。此命令指令碼會執行連續迴圈，將您的 Amazon S3 儲存貯體和金鑰每 10 秒同步至本機資料夾。您可以視需要調整逾時。您應該會在 Windows File Explorer 的 Amazon S3 儲存貯體中看到應用程式的原始碼。

若要將新檔案新增至您正在使用的集，或更新現有的檔案，請將檔案上傳至 Amazon S3 儲存貯體，並在 中定義的下一個反覆運算時同步至您的目錄m2-rclone.cmd。同樣地，如果您想要刪除某些檔案，請從 Amazon S3 儲存貯體中刪除它們。下一個同步操作會從本機目錄刪除它們。

步驟 3：建立 Amazon RDS 執行個體的 ODBC 來源

1. 若要啟動 EA_Admin 工具，請導覽至瀏覽器視窗左上角的應用程式選擇器選單，然後選擇 MF EA_Admin。
2. 從管理員功能表中，選擇 ODBC 資料來源，然後從使用者 DSN 索引標籤選擇新增。
3. 在建立新的資料來源對話方塊中，選擇 PostgreSQL Unicode 驅動程式，然後選擇完成。
4. 在 PostgreSQL Unicode ODBC 驅動程式 (psqlODBC) 設定對話方塊中，定義並記下您想要的資料來源名稱。使用您先前建立之 RDS 執行個體的值完成下列參數：

   描述

   可協助您快速識別此資料庫連線的選用描述。

資料庫

您先前建立的 Amazon RDS 資料庫。

Server

Amazon RDS 端點。

連線埠

Amazon RDS 連接埠。

使用者名稱

如 Amazon RDS 執行個體所定義。

密碼

如 Amazon RDS 執行個體所定義。

5. 選擇測試以驗證與 Amazon RDS 的連線是否成功，然後選擇儲存以儲存您的新使用者 DSN。
6. 等到您看到確認建立適當工作區的訊息，然後選擇確定以使用 ODBC 資料來源完成並關閉 EA_Admin 工具。
7. 再次導覽至應用程式選取器選單，然後選擇 Enterprise Analyzer 以啟動工具。選擇建立新。
8. 在工作區組態視窗中，輸入您的工作區名稱並定義其位置。如果您在此組態下工作，則工作區可以是 Amazon S3 型磁碟，或者如果您偏好的話，可以是主資料夾。
9. 選擇選擇其他資料庫以連線至您的 Amazon RDS 執行個體。
10. 從選項中選擇 Postgre 圖示，然後選擇確定。
11. 針對選項 – 定義連線參數下的 Windows 設定，輸入您建立的資料來源名稱。同時輸入資料庫名稱、結構描述名稱、使用者名稱和密碼。選擇確定。
12. 等待 Enterprise Analyzer 建立所有資料表、索引等，以便儲存結果。此程序可能需要幾分鐘的時間。Enterprise Analyzer 會確認資料庫和工作區何時可供使用。
13. 再次導覽至應用程式選擇器選單，然後選擇 Enterprise Analyzer 以啟動工具。
14. Enterprise Analyzer 啟動視窗會顯示在新的所選工作區位置。選擇確定。
15. 導覽至左側窗格中的儲存庫，選取儲存庫名稱，然後選擇將檔案/資料夾新增至工作區。選取應用程式碼存放所在的資料夾，將其新增至工作區。如果需要，您可以使用先前的 BankDemo 範例程式碼。當 Enterprise Analyzer 提示您驗證這些檔案時，請選擇驗證以啟動初始 Enterprise Analyzer 驗證報告。視您的應用程式大小而定，可能需要幾分鐘的時間才能完成。
16. 展開您的工作區，以查看您已新增至工作區的檔案和資料夾。物件類型和循環複雜性報告也會顯示在圖表檢視器窗格的上象限中。

您現在可以將 Enterprise Analyzer 用於所有必要的任務。

後續工作階段

1. 使用 AppStream 2.0 開始工作階段，其中包含您在 AppStream 2.0 歡迎電子郵件訊息中收到的 URL。
2. 使用您的電子郵件和永久密碼登入。
3. 選取 Enterprise Analyzer 堆疊。
4. 如果您使用此選項來共用工作區檔案，請啟動 `Rclone` 以連線至 Amazon S3 支援的磁碟。
5. 啟動 Enterprise Analyzer 來執行您的任務。

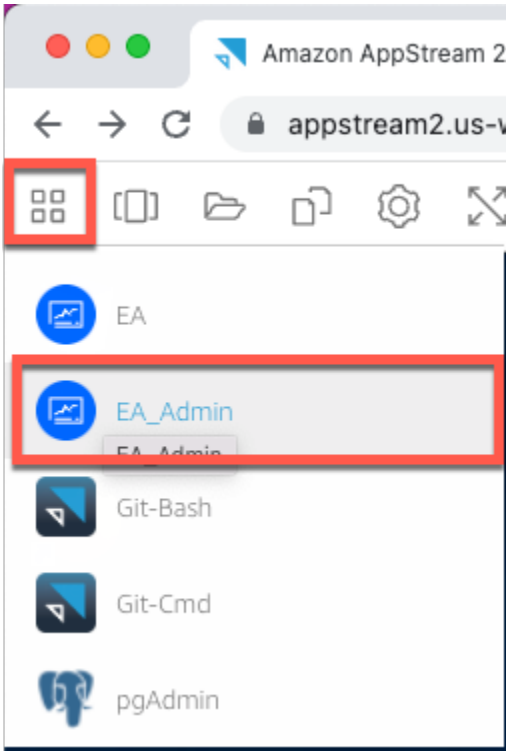
工作區連線故障診斷

當您嘗試重新連線至 Enterprise Analyzer 工作區時，您可能會看到如下錯誤：

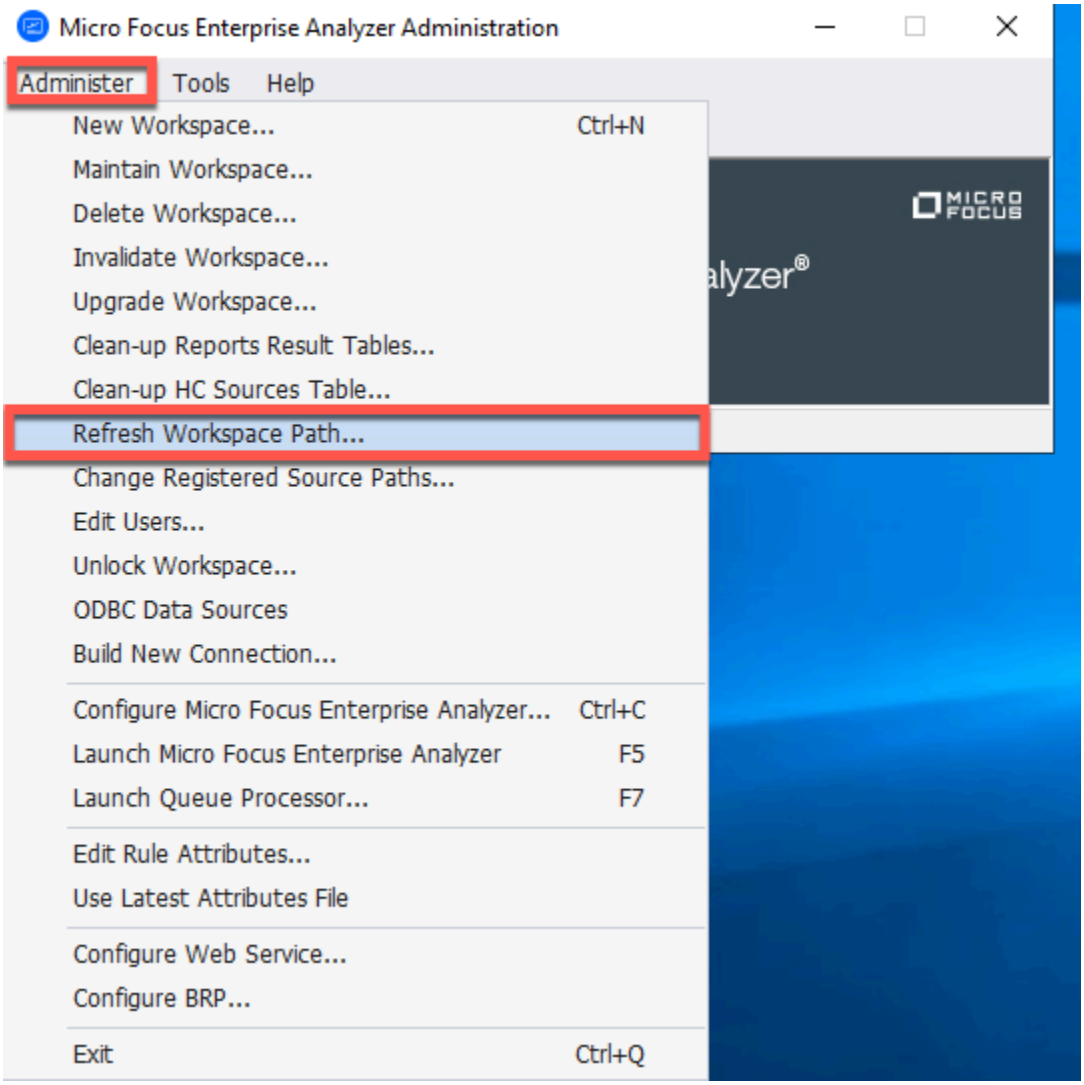
```
Cannot access the workspace directory D:\PhotonUser\My Files\Home Folder\EA_BankDemo.
 The workspace has been created on a non-shared disk of the EC2AMAZ-E6LC33H computer.
 Would you like to correct the workspace directory location?
```

若要解決此問題，請選擇確定以清除訊息，然後完成以下步驟。

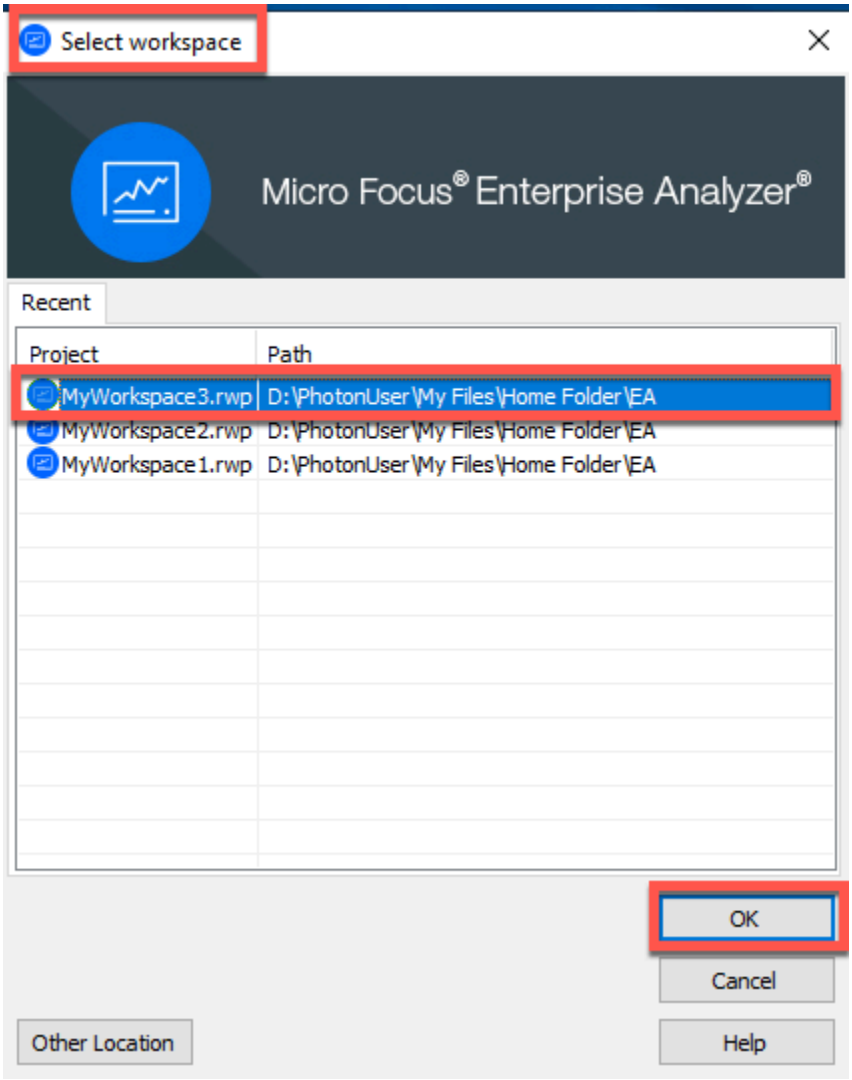
1. 在 AppStream 2.0 中，選擇工具列上的啟動應用程式圖示，然後選擇 EA_Admin 以啟動 Enterprise Analyzer 管理工具。



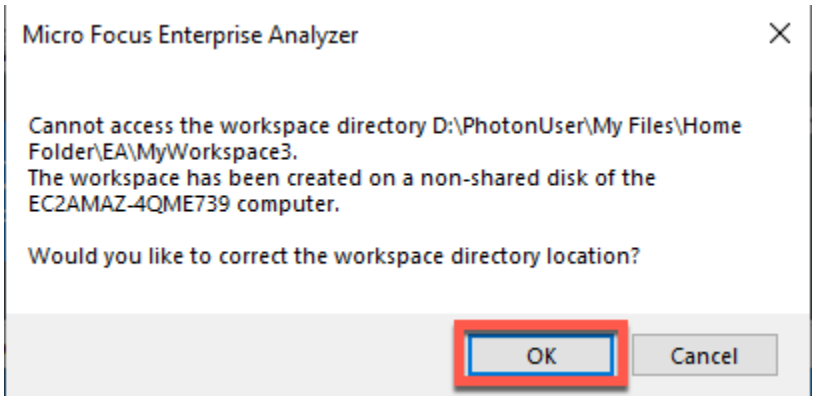
2. 從管理員功能表中，選擇重新整理工作區路徑...。



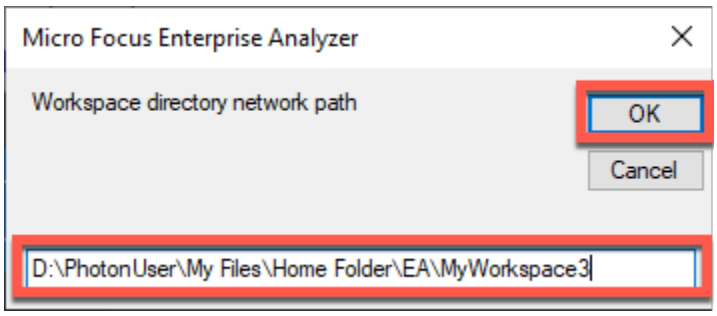
3. 在選取工作區下，選擇您想要的工作區，然後選擇確定。



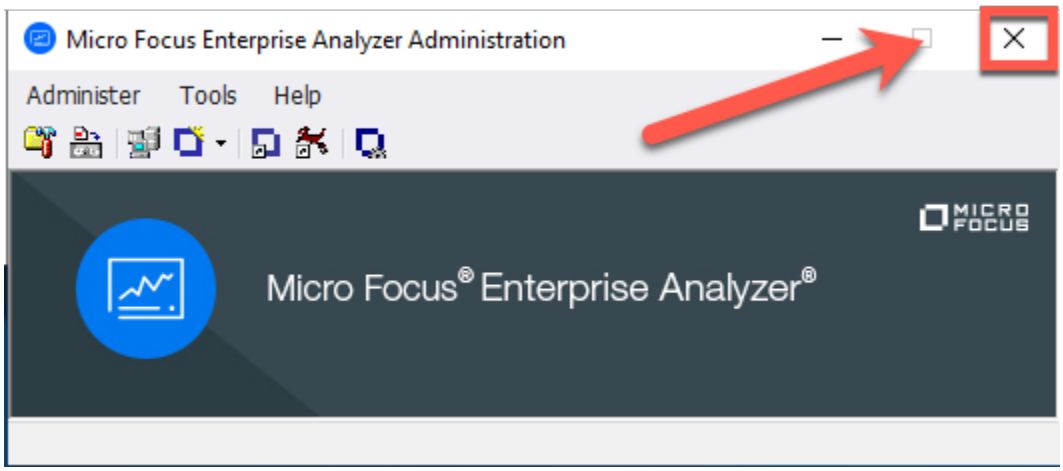
4. 選擇確定以確認錯誤訊息。



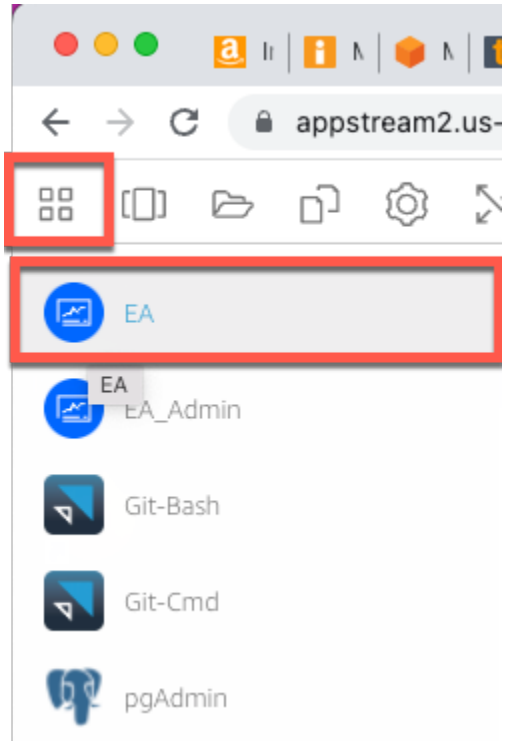
5. 在工作區目錄網路路徑下，輸入工作區的正確路徑，例如 `D:\PhotonUser\My Files\Home Folder\EA\MyWorkspace3`。



6. 關閉 Micro Focus Enterprise Analyzer 管理工具。



7. 在 AppStream 2.0 中，選擇工具列上的啟動應用程式圖示，然後選擇 EA 以啟動 Micro Focus Enterprise Analyzer。



8. 重複步驟 3 - 5。

Micro Focus Enterprise Analyzer 現在應使用現有的工作區開啟。

清除資源

如果您不再需要為此教學課程建立的資源，請將其刪除，以免產生進一步的費用。請完成下列步驟：

- 使用 EA_Admin 工具刪除工作區。
- 刪除您為此教學課程建立的 S3 儲存貯體。如需詳細資訊，請參閱《Amazon S3 使用者指南》中的刪除儲存貯體。
- 刪除您為此教學課程建立的資料庫。如需詳細資訊，請參閱刪除資料庫執行個體。

教學課程：在 AppStream 2.0 上設定 Rocket Enterprise Developer

本教學課程說明如何為一或多個大型主機應用程式設定 Rocket Enterprise Developer（先前稱為 Micro Focus Enterprise Developer)，以便使用企業開發人員功能進行維護、編譯和測試。設定是以 AWS 大型主機現代化與客戶共用的 AppStream 2.0 Windows 映像為基礎，以及建立 AppStream 2.0 機群和堆疊，如 中所述教學課程：設定 AppStream 2.0 以搭配 Rocket Enterprise Analyzer 和 Rocket Enterprise Developer 使用。

> **Important**
>
> 本教學課程中的步驟假設您使用可 AWS CloudFormation 下載的範本 cfn-m2-appstream-fleet-ea-ed.yaml 設定 AppStream 2.0。如需詳細資訊，請參閱教學課程：設定 AppStream 2.0 以搭配 Rocket Enterprise Analyzer 和 Rocket Enterprise Developer 使用。
> 當企業開發人員機群和堆疊啟動並執行時，您必須執行此設定的步驟。

如需企業開發人員 v7 功能和交付項目的完整說明，請參閱 Rocket Software（先前稱為 Micro Focus) 網站上的up-to-date線上文件 (v7.0)。

影像內容

除了企業開發人員本身之外，映像還包含 Rumba (TN3270 模擬器）。它還包含以下工具和程式庫。

第三方工具

- Python
- Rclone
- pgAdmin
- git-scm
- PostgreSQL ODBC 驅動程式

中的程式庫 `C:\Users\Public`

- 企業開發人員的 BankDemo 原始程式碼和專案定義：`m2-bankdemo-template.zip`。
- 大型主機的 MFA 安裝套件：`mfa.zip`。如需詳細資訊，請參閱 Micro Focus Enterprise Developer 文件中的大型主機存取概觀。
- Rclone 的命令和組態檔案 （教學課程中的指示）： `m2-rclone.cmd`和 `m2-rclone.conf`。

如果您需要存取尚未載入 CodeCommit 儲存庫的原始程式碼，但可在 Amazon S3 儲存貯體中使用，例如，若要將原始程式碼初始載入 git，請依照 程序建立虛擬 Windows 磁碟，如 中所述教學課程：在 AppStream 2.0 上設定 Enterprise Analyzer。

主題

- 先決條件
- 步驟 1：由個別企業開發人員使用者設定
- 步驟 2：在 Windows 上建立 Amazon S3 型虛擬資料夾 （選用）
- 步驟 3：複製儲存庫
- 後續工作階段
- 清除資源

先決條件

- 一或多個 CodeCommit 儲存庫，載入要維護的應用程式原始碼。儲存庫設定應符合上述CI/CD管道的要求，以透過結合這兩種工具來建立協同合作。
- 每個使用者都必須擁有 CodeCommit 儲存庫的登入資料，或帳戶管理員根據 AWS CodeCommit 身分驗證和存取控制中的資訊定義的儲存庫。這些登入資料的結構會在 AWS CodeCommit 的身分驗證和存取控制中檢閱，而 CodeCommit 的 IAM 授權完整參考則位於 CodeCommit 許可參考中：管理員可以為具有每個儲存庫角色特定登入資料的不同角色定義不同的 IAM 政策，並將使用者的授權限制在指定儲存庫上必須完成的特定任務集。因此，對於 CodeCommit 儲存庫的每個維護者，帳戶

管理員將產生主要使用者，並透過選取適當的 IAM 政策或 CodeCommit 存取政策，授予此使用者存取所需儲存庫或儲存庫的許可。

步驟 1：由個別企業開發人員使用者設定

1. 取得您的 IAM 登入資料：
    1. 前往 AWS https：//https://console.aws.amazon.com/iam/。
    2. 請遵循《 AWS CodeCommit 使用者指南》中的使用 Git 登入資料設定 HTTPS 使用者的步驟 3 中所述的程序。
    3. 複製 IAM 為您產生的 CodeCommit 特定登入資料，方法是顯示、複製此資訊，然後將此資訊貼到本機電腦上的安全檔案中，或選擇下載登入資料，將此資訊下載為 .CSV 檔案。您需要此資訊才能連線至 CodeCommit。
2. 根據歡迎電子郵件中收到的 URL，使用 AppStream 2.0 啟動工作階段。使用您的電子郵件做為使用者名稱並建立您的密碼。
3. 選取您的企業開發人員堆疊。
4. 在選單頁面上，選擇桌面以連接機群串流的 Windows 桌面。

步驟 2：在 Windows 上建立 Amazon S3 型虛擬資料夾 （選用）

如果需要 Rclone （請參閱上述說明），請在 Windows 上建立以 Amazon S3 為基礎的虛擬資料夾：（如果所有應用程式成品完全來自 CodeCommit 存取，則為選用）。

> **Note**
>
> 如果您已在 AWS Mainframe Modernization 預覽期間使用 Rclone，則必須更新`m2-rclone.cmd`至位於 的較新版本`C:\Users\Public`。

1. `C:\Users\PhotonUser\My Files\Home Folder` 使用 `m2-rclone.cmd` File Explorer 將 中提供的 `m2-rclone.conf`和 檔案`C:\Users\Public`複製到主資料夾。
2. 使用 AWS 存取金鑰和對應的秘密以及 `m2-rclone.conf` 更新組態參數 AWS 區域。

```
[m2-s3]
type = s3
provider = AWS
```

```
access_key_id = YOUR-ACCESS-KEY
secret_access_key = YOUR-SECRET-KEY
region = YOUR-REGION
acl = private
server_side_encryption = AES256
```

3. 在 m2-rclone.cmd 中，進行下列變更：

   - amzn-s3-demo-bucket 變更為您的 Amazon S3 儲存貯體名稱。例如 m2-s3-mybucket。
   - your-s3-folder-key 變更為 Amazon S3 儲存貯體金鑰。例如 myProject。
   - your-local-folder-path 將 變更為您要從包含它們的 Amazon S3 儲存貯體同步應用程式檔案的目錄路徑。例如 D:\PhotonUser\My Files\Home Folder\m2-new。此同步目錄必須是主資料夾的子目錄，以便 AppStream 2.0 在工作階段開始和結束時正確備份和還原。

```
:loop
timeout /T 10
"C:\Program Files\rclone\rclone.exe" sync m2-s3:amzn-s3-demo-bucket/your-s3-
folder-key "D:\PhotonUser\My Files\Home Folder\your-local-folder-path" --config "D:
\PhotonUser\My Files\Home Folder\m2-rclone.conf"
goto :loop
```

4. 視需要開啟 Windows 命令提示字元，cd 至 C:\Users\PhotonUser\My Files\Home Folder並執行 m2-rclone.cmd。此命令指令碼會執行連續迴圈，將您的 Amazon S3 儲存貯體和金鑰每 10 秒同步至本機資料夾。您可以視需要調整逾時。您應該會在 Windows File Explorer 的 Amazon S3 儲存貯體中看到應用程式的原始碼。

若要將新檔案新增至您正在使用的集，或更新現有的檔案，請將檔案上傳至 Amazon S3 儲存貯體，並在 中定義的下一個反覆運算時同步至您的目錄m2-rclone.cmd。同樣地，如果您想要刪除某些檔案，請從 Amazon S3 儲存貯體中刪除它們。下一個同步操作會從本機目錄刪除它們。

步驟 3：複製儲存庫

1. 導覽至瀏覽器視窗左上角的應用程式選擇器選單，然後選取企業開發人員。
2. 選擇 C:\Users\PhotonUser\My Files\Home Folder (也稱為 D: \PhotonUser\My Files\Home Folder) 作為工作區的位置，以完成主資料夾中企業開發人員所需的工作區建立。
3. 在企業開發人員中，前往 Project Explorer 複製您的 CodeCommit 儲存庫，按一下滑鼠右鍵並選擇匯入、匯入...、Git、Git 中的專案。然後，輸入您的 CodeCommit 特定登入憑證，並完成 Eclipse 對話方塊以匯入程式碼。

中的 CodeCommit git 儲存庫現已在您的本機工作區中複製。

您的企業開發人員工作區現在已準備好在應用程式上開始維護工作。特別是，您可以使用與企業開發人員整合的 Enterprise Server (ES) 本機執行個體，以互動方式偵錯和執行您的應用程式，以在本機驗證您的變更。

> **Note**
>
> 本機企業開發人員環境，包括本機企業伺服器執行個體，在 Windows 下執行，而 AWS 大型主機現代化在 Linux 下執行。建議您在將新應用程式遞交至 CodeCommit 後，以及在將新應用程式推展至生產環境之前，在 AWS Mainframe Modernization 提供的 Linux 環境中執行補充測試。

後續工作階段

當您選取 AppStream 2.0 管理下的資料夾時，例如複製 CodeCommit 儲存庫的主資料夾，它將在工作階段之間以透明的方式儲存和還原。下次您需要使用應用程式時，請完成下列步驟：

1. 根據歡迎電子郵件中收到的 URL，使用 AppStream 2.0 啟動工作階段。
2. 使用您的電子郵件和永久密碼登入。
3. 選取企業開發人員堆疊。
4. 使用此選項來共用工作區檔案時，啟動 `Rclone` 以連線至 （請參閱上述） Amazon S3 後端磁碟。
5. 啟動企業開發人員來執行您的工作。

清除資源

如果您不再需要為本教學課程建立的資源，請將其刪除，這樣就不會繼續收取這些資源的費用。請完成下列步驟：

- 刪除您為此教學課程建立的 CodeCommit 儲存庫。如需詳細資訊，請參閱AWS CodeCommit 《使用者指南》中的刪除 CodeCommit 儲存庫。
- 刪除您為此教學課程建立的資料庫。如需詳細資訊，請參閱刪除資料庫執行個體。

AWS Mainframe Modernization 中的可用批次公用程式

大型主機應用程式通常會使用批次公用程式來執行特定功能，例如排序資料、使用 FTP 傳輸檔案、將資料載入資料庫，例如 DB2、從資料庫卸載資料等。

當您將應用程式遷移至 AWS Mainframe Modernization 時，您需要功能上同等的替代公用程式，來執行與大型主機相同的任務。其中一些公用程式可能已做為 AWS Mainframe Modernization 執行期引擎的一部分提供，但我們提供下列替代公用程式：

- M2SFTP - 使用 SFTP 通訊協定啟用安全的檔案傳輸。
- M2WAIT - 等待指定的時間量，然後再繼續批次任務中的下一個步驟。
- TXT2PDF - 將文字檔案轉換為 PDF 格式。
- M2DFUTIL - 在與大型主機 ADRDSSU 公用程式提供的支援類似的資料集上提供備份、還原、刪除和複製函數。
- M2RUNCMD - 可讓您直接從 JCL 執行 Rocket Software （ 先前稱為 Micro Focus) 命令、指令碼和系統呼叫。

我們根據客戶意見回饋開發這些批次公用程式，並設計它們以提供與大型主機公用程式相同的功能。目標是讓您盡可能順利地從大型主機轉換至 AWS 大型主機現代化。

主題

- 二進位位置
- M2SFTP 批次公用程式
- M2WAIT 批次公用程式
- TXT2PDF 批次公用程式
- M2DFUTIL 批次公用程式
- M2RUNCMD 批次公用程式

二進位位置

這些公用程式預先安裝在 Rocket Enterprise Developer (ED) 和 Rocket Software (ES) 產品上。您可以在下列位置找到 ED 和 ES 的所有變體：

- Linux：`/opt/aws/m2/microfocus/utilities/64bit`
- Windows (32 位元 ）： `C:\AWS\M2\MicroFocus\Utilities\32bit`

- Windows (64 位元）： `C:\AWS\M2\MicroFocus\Utilities\64bit`

M2SFTP 批次公用程式

M2SFTP 是一種 JCL 公用程式，旨在使用安全檔案傳輸通訊協定 (SFTP) 在系統之間執行安全的檔案傳輸。此程式使用 Putty SFTP 用戶端 `psftp`來執行實際的檔案傳輸。此程式的運作方式類似於大型主機 FTP 公用程式，並使用使用者和密碼身分驗證。

> **Note**
>
> 不支援公有金鑰身分驗證。

若要將您的大型主機 FTP JCLs 轉換為使用 SFTP，請將 `PGM=FTP`變更為 `PGM=M2SFTP`。

主題

- 支援平台
- 安裝相依項目
- 為 AWS 大型主機現代化受管設定 M2SFTP
- 在 Amazon EC22 上為大型主機現代化執行時間設定 M2SFTP （包括 AppStream 2.0) AWS
- 範例 JCLs
- Putty SFTP (PSFTP) 用戶端命令參考
- 後續步驟

支援平台

您可以在下列任何平台上使用 M2SFTP：

- AWS Mainframe Modernization Rocket Software （先前稱為 Micro Focus) 受管
- Rocket 軟體執行期 （在 Amazon EC2 上）
- Rocket Software Enterprise Developer (ED) 和 Rocket Software Enterprise Server (ES) 產品的所有變體。

安裝相依項目

在 Windows 上安裝 Putty SFTP 用戶端

- 下載 PuTTY SFTP 用戶端並進行安裝。

在 Linux 上安裝 Putty SFTP 用戶端：

- 執行下列命令來安裝 Putty SFTP 用戶端：

```
sudo yum -y install putty
```

為 AWS 大型主機現代化受管設定 M2SFTP

如果您的遷移應用程式在 AWS Mainframe Modernization Managed 上執行，您將需要設定 M2SFTP，如下所示。

- 為 MFFTP 設定適當的 Rocket Enterprise Server 環境變數。以下是幾個範例：
  - `MFFTP_TEMP_DIR`
  - `MFFTP_SENDEOL`
  - `MFFTP_TIME`
  - `MFFTP_ABEND`

  您可以視需要設定任意數量的這些變數。您可以使用 `ENVAR DD`陳述式在 JCL 中設定它們。如需這些變數的詳細資訊，請參閱 Micro Focus 文件中的 MFFTP 控制變數。

若要測試您的組態，請參閱 範例 JCLs。

在 Amazon EC22 上為大型主機現代化執行時間設定 M2SFTP （包括 AppStream 2.0) AWS

如果您遷移的應用程式在 Amazon EC2 上的 AWS Mainframe Modernization 執行時間上執行，請設定 M2SFTP，如下所示。

1. 變更 Micro Focus JES 程式路徑，以包含批次公用程式的二進位位置。如果您需要指定多個路徑，請使用冒號 (:) 來分隔 Linux 上的路徑和 Windows 上的分號 (;)。

   - Linux：`/opt/aws/m2/microfocus/utilities/64bit`
   - Windows (32 位元）： `C:\AWS\M2\MicroFocus\Utilities\32bit`
   - Windows (64 位元）： `C:\AWS\M2\MicroFocus\Utilities\64bit`

2. 為 MFFTP 設定適當的 Rocket Enterprise Server 環境變數。以下是幾個範例：

   - `MFFTP_TEMP_DIR`
   - `MFFTP_SENDEOL`
   - `MFFTP_TIME`
   - MFFTP_ABEND

   您可以視需要設定任意數量的這些變數。您可以使用 ENVAR DD陳述式在 JCL 中設定它們。如需這些變數的詳細資訊，請參閱 Micro Focus 文件中的 MFFTP 控制變數。

若要測試您的組態，請參閱 範例 JCLs。

範例 JCLs

若要測試安裝，您可以使用下列其中一個範例 JCL 檔案。

M2SFTP1.jcl

此 JCL 說明如何呼叫 M2SFTP 將檔案傳送至遠端 SFTP 伺服器。請注意 ENVVAR DD陳述式中設定的環境變數。

```
//M2SFTP1 JOB 'M2SFTP1',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*-------------------------------------------------------------------**
//* Sample SFTP JCL step to send a file to SFTP server*
//*-------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=M2SFTP,
//            PARM='127.0.0.1 (EXIT=99 TIMEOUT 300'
//*
```

```
//SYSFTPD  DD  *
RECFM FB
LRECL 80
SBSENDEOL CRLF
MBSENDEOL CRLF
TRAILINGBLANKS FALSE
/*
//NETRC    DD  *
machine 127.0.0.1 login sftpuser password sftppass
/*
//SYSPRINT DD  SYSOUT=*
//OUTPUT   DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//INPUT    DD  *
type a
locsite notrailingblanks
cd files
put 'AWS.M2.TXT2PDF1.PDF' AWS.M2.TXT2PDF1.pdf
put 'AWS.M2.CARDDEMO.CARDDATA.PS' AWS.M2.CARDDEMO.CARDDATA.PS1.txt
quit
/*
//ENVVAR   DD *
MFFTP_VERBOSE_OUTPUT=ON
MFFTP_KEEP=N
/*
//*
//
```

M2SFTP2.jcl

此 JCL 說明如何呼叫 M2SFTP 以從遠端 SFTP 伺服器接收檔案。請注意 ENVVAR DD陳述式中設定的環境變數。

```
//M2SFTP2 JOB 'M2SFTP2',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*-------------------------------------------------------------------**
//* Sample SFTP JCL step to receive a file from SFTP server*
//*-------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=M2SFTP
```

```
//*
//SYSPRINT DD  SYSOUT=*
//OUTPUT   DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//INPUT    DD  *
open 127.0.0.1
sftpuser
sftppass
cd files
locsite recfm=fb lrecl=150
get AWS.M2.CARDDEMO.CARDDATA.PS.txt +
'AWS.M2.CARDDEMO.CARDDATA.PS2' (replace
quit
/*
//ENVVAR   DD *
MFFTP_VERBOSE_OUTPUT=ON
MFFTP_KEEP=N
/*
//*
//
```

> **Note**
>
> 我們強烈建議將 FTP 登入資料儲存在 NETRC 檔案中，並限制只有授權使用者才能存取。

Putty SFTP (PSFTP) 用戶端命令參考

PSFTP 用戶端不支援所有 FTP 命令。下列清單顯示 PSFTP 支援的所有命令。

Command	描述
!	執行本機命令
再見	完成您的 SFTP 工作階段
cd	變更遠端工作目錄
chmod	變更檔案許可和模式
關閉	完成您的 SFTP 工作階段，但不退出 PSFTP

Command	描述
德爾	刪除遠端伺服器上的檔案
dir	列出遠端檔案
exit	完成您的 SFTP 工作階段
get	將檔案從伺服器下載到您的本機電腦
help	提供協助
lcd	變更本機工作目錄
lpwd	列印本機工作目錄
ls	列出遠端檔案
mget	一次下載多個檔案
mkdir	在遠端伺服器上建立目錄
mput	一次上傳多個檔案
mv	移動或重新命名遠端伺服器上的檔案 (s)
開啟	連線至主機
put	將檔案從本機電腦上傳至伺服器
pwd	列印遠端工作目錄
結束	完成您的 SFTP 工作階段
重新取得	繼續下載檔案
ren	移動或重新命名遠端伺服器上的檔案 (s)
評價	繼續上傳檔案
rm	刪除遠端伺服器上的檔案

Command	描述
rmdir	移除遠端伺服器上的目錄

後續步驟

若要使用 SFTP 將檔案上傳和下載到 Amazon Simple Storage Service，您可以搭配 使用 M2SFTP AWS Transfer Family，如以下部落格文章所述。

- 使用 AWS SFTP 邏輯目錄建置簡單的資料分佈服務
- AWS Transfer for SFTP 使用 啟用密碼身分驗證 AWS Secrets Manager

M2WAIT 批次公用程式

M2WAIT 是一種大型主機公用程式，可讓您指定以秒、分鐘或小時為單位的持續時間，在 JCL 指令碼中引入等待期間。您可以透過傳遞您想要等待的時間做為輸入參數，直接從 JCL 呼叫 M2WAIT。在內部，M2WAIT 程式會呼叫 Rocket Software（先前為 Micro Focus) 提供的模組C$SLEEP，以等待指定的時間。

> **Note**
>
> 您可以使用 Micro Focus 別名來取代 JCL 指令碼中擁有的內容。如需詳細資訊，請參閱 Micro Focus 文件中的 JES 別名。

主題

- 支援平台
- 為 AWS 大型主機現代化受管設定 M2WAIT
- 在 Amazon EC22 上為大型主機現代化執行時間設定 M2WAIT（包括 AppStream 2.0) AWS
- 範例 JCL

支援平台

您可以在下列任何平台上使用 M2WAIT：

- AWS Mainframe Modernization Rocket Software（先前稱為 Micro Focus) 受管

- Rocket 軟體執行期（在 Amazon EC2 上）
- Rocket Software Enterprise Developer (ED) 和 Rocket Software Enterprise Server (ES) 產品的所有變體。

為 AWS 大型主機現代化受管設定 M2WAIT

如果您的遷移應用程式在 AWS Mainframe Modernization Managed 上執行，您將需要設定 M2WAIT，如下所示。

- 透過傳遞輸入參數在 JCL 中使用程式 M2WAIT，如 所示範例 JCL。

在 Amazon EC22 上為大型主機現代化執行時間設定 M2WAIT（包括 AppStream 2.0) AWS

如果您遷移的應用程式在 Amazon EC2 上的 AWS Mainframe Modernization 執行時間上執行，請設定 M2WAIT，如下所示。

1. 變更 Micro Focus JES 程式路徑，以包含批次公用程式的二進位位置。如果您需要指定多個路徑，請使用冒號 (:) 來分隔 Linux 上的路徑和 Windows 上的分號 (;)。
   - Linux：`/opt/aws/m2/microfocus/utilities/64bit`
   - Windows (32 位元）： `C:\AWS\M2\MicroFocus\Utilities\32bit`
   - Windows (64 位元）： `C:\AWS\M2\MicroFocus\Utilities\64bit`
2. 透過傳遞輸入參數，在您的 JCL 中使用程式 M2WAIT，如 所示範例 JCL。

範例 JCL

若要測試安裝，您可以使用 `M2WAIT1.jcl`程式。

此範例 JCL 示範如何呼叫 M2WAIT 並傳遞數個不同的持續時間。

```
//M2WAIT1 JOB 'M2WAIT',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*-------------------------------------------------------------------**
```

```
//* Wait for 12 Seconds*
//*-------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=M2WAIT,PARM='S012'
//SYSOUT DD SYSOUT=*
//*
//*-------------------------------------------------------------------**
//* Wait for 0 Seconds (defaulted to 10 Seconds)*
//*-------------------------------------------------------------------**
//*
//STEP02 EXEC PGM=M2WAIT,PARM='S000'
//SYSOUT DD SYSOUT=*
//*
//*-------------------------------------------------------------------**
//* Wait for 1 Minute*
//*-------------------------------------------------------------------**
//*
//STEP03 EXEC PGM=M2WAIT,PARM='M001'
//SYSOUT DD SYSOUT=*
//*
//
```

TXT2PDF 批次公用程式

TXT2PDF 是一種大型主機公用程式，通常用於將文字檔案轉換為 PDF 檔案。此公用程式使用與 TXT2PDF 相同的原始程式碼 (z/OS 免費軟體）。我們已將其修改為在 AWS Mainframe Modernization Rocket Software （ 先前稱為 Micro Focus) 執行時間環境中執行。

主題

- 支援平台
- 為 AWS 大型主機現代化受管設定 TXT2PDF
- 在 Amazon EC22 上設定適用於大型主機現代化執行時間的 TXT2PDF （ 包括 AppStream 2.0) AWS
- 範例 JCL
- 修改
- 參考

支援平台

您可以在下列任何平台上使用 TXT2PDF：

- AWS 受管大型主機現代化火箭軟體
- Rocket 軟體執行期 （ 在 Amazon EC2 上 ）
- Rocket Enterprise Developer (ED) 和 Rocket Enterprise Server (ES) 產品的所有變體。

為 AWS 大型主機現代化受管設定 TXT2PDF

如果您遷移的應用程式在 AWS Mainframe Modernization Managed 上執行，請設定 TXT2PDF，如下所示。

- 建立稱為 的 REXX EXEC 程式庫`AWS.M2.REXX.EXEC`。下載這些 REXX 模組並將其複製到程式庫。
  - `TXT2PDF.rex` - TXT2PDF z/OS 免費軟體 （ 已修改 ）
  - `TXT2PDFD.rex` - TXT2PDF z/OS 免費軟體 （ 未修改 ）
  - `TXT2PDFX.rex` - TXT2PDF z/OS 免費軟體 （ 已修改 ）
  - `M2GETOS.rex` - 檢查作業系統類型 (Windows 或 Linux)

若要測試您的組態，請參閱 範例 JCL。

在 Amazon EC22 上設定適用於大型主機現代化執行時間的 TXT2PDF （ 包括 AppStream 2.0) AWS

如果您遷移的應用程式在 Amazon EC2 上的 AWS Mainframe Modernization 執行時間上執行，請設定 TXT2PDF，如下所示。

1. 將 Rocket Software 環境變數`MFREXX_CHARSET`設定為適當的值，例如 ASCII 資料的「`A`」。

   > **Important**
   >
   > 輸入錯誤的值可能會導致資料轉換問題 （ 從 EBCDIC 到 ASCII)，導致產生的 PDF 無法讀取或無法操作。我們建議設定 `MFREXX_CHARSET`以符合 `MF_CHARSET`。

2. 變更 Micro Focus JES 程式路徑，以包含批次公用程式的二進位位置。如果您需要指定多個路徑，請使用冒號 (`:`) 來分隔 Linux 上的路徑和 Windows 上的分號 (`;`)。
   - Linux：`/opt/aws/m2/microfocus/utilities/64bit`
   - Windows (32 位元 ）： `C:\AWS\M2\MicroFocus\Utilities\32bit`

- Windows (64 位元）： `C:\AWS\M2\MicroFocus\Utilities\64bit`

3. 建立稱為 的 REXX EXEC 程式庫`AWS.M2.REXX.EXEC`。下載這些 REXX 模組並將其複製到程式庫。

   - `TXT2PDF.rex` - TXT2PDF z/OS 免費軟體 （已修改）
   - `TXT2PDFD.rex` - TXT2PDF z/OS 免費軟體 （未修改）
   - `TXT2PDFX.rex` - TXT2PDF z/OS 免費軟體 （已修改）
   - `M2GETOS.rex` - 檢查作業系統類型 (Windows 或 Linux)

若要測試您的組態，請參閱 範例 JCL。

範例 JCL

若要測試安裝，您可以使用下列其中一個範例 JCL 檔案。

TXT2PDF1.jcl

此範例 JCL 檔案使用 TXT2PDF 轉換的 DD 名稱。

```
//TXT2PDF1 JOB 'TXT2PDF1',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*--------------------------------------------------------------------**
//* PRE DELETE*
//*--------------------------------------------------------------------**
//*
//PREDEL  EXEC PGM=IEFBR14
//*
//DD01     DD DSN=AWS.M2.TXT2PDF1.PDF.VB,
//            DISP=(MOD,DELETE,DELETE)
//*
//DD02     DD DSN=AWS.M2.TXT2PDF1.PDF,
//            DISP=(MOD,DELETE,DELETE)
//*
//*--------------------------------------------------------------------**
//* CALL TXT2PDF TO CONVERT FROM TEXT TO PDF (VB)*
//*--------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=IKJEFT1B
```

```
//*
//SYSEXEC  DD DISP=SHR,DSN=AWS.M2.REXX.EXEC
//*
//INDD     DD *
1THIS IS THE FIRST LINE ON THE PAGE 1
0THIS IS THE THIRD LINE ON THE PAGE 1
-THIS IS THE   6TH LINE ON THE PAGE 1
THIS IS THE   7TH LINE ON THE PAGE 1
+_____________________________________ - OVERSTRIKE 7TH LINE
1THIS IS THE FIRST LINE ON THE PAGE 2
0THIS IS THE THIRD LINE ON THE PAGE 2
-THIS IS THE   6TH LINE ON THE PAGE 2
THIS IS THE   7TH LINE ON THE PAGE 2
+_____________________________________ - OVERSTRIKE 7TH LINE
/*
//*
//OUTDD    DD DSN=AWS.M2.TXT2PDF1.PDF.VB,
//            DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=256,DSORG=PS,RECFM=VB,BLKSIZE=0)
//*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DDNAME=SYSIN
//*
//SYSIN    DD *
%TXT2PDF BROWSE Y IN DD:INDD +
OUT DD:OUTDD +
CC YES
/*
//*
//*--------------------------------------------------------------------**
//* CONVERT PDF (VB) TO PDF (LSEQ - BYTE STREAM)*
//*--------------------------------------------------------------------**
//*
//STEP02 EXEC PGM=VB2LSEQ
//*
//INFILE   DD DSN=AWS.M2.TXT2PDF1.PDF.VB,DISP=SHR
//*
//OUTFILE  DD DSN=AWS.M2.TXT2PDF1.PDF,
//            DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=256,DSORG=PS,RECFM=LSEQ,BLKSIZE=0)
//*
//SYSOUT   DD SYSOUT=*
//*
```

```
//
```

TXT2PDF2.jcl

此範例 JCL 使用 DSN 名稱進行 TXT2PDF 轉換。

```
//TXT2PDF2 JOB 'TXT2PDF2',CLASS=A,MSGCLASS=X,TIME=1440
//*
//* Copyright Amazon.com, Inc. or its affiliates.*
//* All Rights Reserved.*
//*
//*--------------------------------------------------------------------**
//* PRE DELETE*
//*--------------------------------------------------------------------**
//*
//PREDEL  EXEC PGM=IEFBR14
//*
//DD01     DD DSN=AWS.M2.TXT2PDF2.PDF.VB,
//            DISP=(MOD,DELETE,DELETE)
//*
//DD02     DD DSN=AWS.M2.TXT2PDF2.PDF,
//            DISP=(MOD,DELETE,DELETE)
//*
//*--------------------------------------------------------------------**
//* CALL TXT2PDF TO CONVERT FROM TEXT TO PDF (VB)*
//*--------------------------------------------------------------------**
//*
//STEP01 EXEC PGM=IKJEFT1B
//*
//SYSEXEC  DD DISP=SHR,DSN=AWS.M2.REXX.EXEC
//*
//INDD     DD *
1THIS IS THE FIRST LINE ON THE PAGE 1
0THIS IS THE THIRD LINE ON THE PAGE 1
-THIS IS THE   6TH LINE ON THE PAGE 1
THIS IS THE   7TH LINE ON THE PAGE 1
+_____________________________________ - OVERSTRIKE 7TH LINE
1THIS IS THE FIRST LINE ON THE PAGE 2
0THIS IS THE THIRD LINE ON THE PAGE 2
-THIS IS THE   6TH LINE ON THE PAGE 2
THIS IS THE   7TH LINE ON THE PAGE 2
+_____________________________________ - OVERSTRIKE 7TH LINE
/*
//*
```

```
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DDNAME=SYSIN
//*
//SYSIN    DD *
%TXT2PDF BROWSE Y IN DD:INDD +
OUT 'AWS.M2.TXT2PDF2.PDF.VB' +
CC YES
/*
//*
//*-------------------------------------------------------------------**
//* CONVERT PDF (VB) TO PDF (LSEQ - BYTE STREAM)*
//*-------------------------------------------------------------------**
//*
//STEP02 EXEC PGM=VB2LSEQ
//*
//INFILE   DD DSN=AWS.M2.TXT2PDF2.PDF.VB,DISP=SHR
//*
//OUTFILE  DD DSN=AWS.M2.TXT2PDF2.PDF,
//            DISP=(NEW,CATLG,DELETE),
//            DCB=(LRECL=256,DSORG=PS,RECFM=LSEQ,BLKSIZE=0)
//*
//SYSOUT   DD SYSOUT=*
//*
//
```

修改

為了讓 TXT2PDF 程式在 AWS Mainframe Modernization Rocket Software 執行期環境上執行，我們進行了下列變更：

- 變更原始程式碼，以確保與 Rocket Software REXX 執行時間相容
- 變更以確保程式可以在 Windows 和 Linux 作業系統上執行
- 修改以支援 EBCDIC 和 ASCII 執行時間

參考

TXT2PDF 參考和原始程式碼：

- 文字轉 PDF 轉換器
- z/OS 免費軟體 TCP/IP 和郵件工具
- TXT2PDF 使用者指南

M2DFUTIL 批次公用程式

M2DFUTIL 是一種 JCL 公用程式，可在資料集上提供備份、還原、刪除和複製函數，類似於大型主機 ADRDSSU 公用程式提供的支援。此程式會從 ADRDSSU 保留許多 SYSIN 參數，可簡化遷移至此新公用程式的程序。

主題

- 支援平台
- 平台需求
- 計劃的未來支援
- 資產位置
- 在 Amazon EC22 上設定 M2DFUTIL 或 AWS 大型主機現代化執行時間（包括 AppStream 2.0)
- 一般語法
- 範例 JCLs

支援平台

您可以在下列任何平台上使用 M2DFUTIL：

- Windows 上的 Rocket Software （先前稱為 Micro Focus) ES (64 位元和 32 位元）
- Linux 上的 Rocket Software ES (64 位元）

平台需求

M2DFUTIL 取決於呼叫指令碼來執行規則表達式測試。在 Windows 上，您必須安裝 Windows Services for Linux (WSL)，此指令碼才能執行。

計劃的未來支援

目前無法從大型主機 ADRDSSU 公用程式取得，但在未來範圍內的功能包括：

- M2 受管
- VSAM
- 檔案名稱重新命名的 COPY 支援
- RESTORE 的重新命名支援

- 多個 INCLUDE 和 EXCLUDE
- 用於由 DSORG、CREDT、EXPDT 進行子選取的 BY 子句
- 重試佇列失敗的 MWAIT 子句
- DUMP/RESTORE 的 S3 儲存支援

資產位置

此公用程式的載入模組會在 M2DFUTIL.so Linux 和 Windows M2DFUTIL.dll 上呼叫。您可以在下列位置找到此載入模組：

- Linux：/opt/aws/m2/microfocus/utilities/64bit
- Windows (32 位元）： C:\AWS\M2\MicroFocus\Utilities\32bit
- Windows (64 位元）： C:\AWS\M2\MicroFocus\Utilities\64bit

用於規則表達式測試的指令碼稱為 compare.sh。您可以在下列位置找到此指令碼：

- Linux：/opt/aws/m2/microfocus/utilities/scripts
- Windows (32 位元）： C:\AWS\M2\MicroFocus\Utilities\scripts

在 Amazon EC22 上設定 M2DFUTIL 或 AWS 大型主機現代化執行時間 （包括 AppStream 2.0)

使用下列項目設定您的企業伺服器區域：

- 在 【ES-Environment】 中新增下列變數
  - M2DFUTILS_BASE_LOC - DUMP 輸出的預設位置
  - M2DFUTILS_SCRIPTPATH - 記錄在資產位置中的compare.sh指令碼位置
  - M2DFUTILS_VERBOSE - 【VERBOSE 或 NORMAL】。這會控制SYSPRINT 輸出中的詳細資訊層級
- 確認載入模組路徑已新增至JES\Configuration\JES Program Path設定
- 確認公用程式目錄中的指令碼具有執行許可。您可以在 Linux 環境中使用 chmod + x <script name>命令新增執行許可

一般語法

DUMP

提供將檔案從目前目錄位置複製到備份位置的功能。此位置目前必須是檔案系統。

流程

DUMP 將執行下列動作：

1. 建立目標位置目錄。
2. 將目標位置目錄編目為 PDS 成員。
3. 透過處理 INCLUDE 參數來決定要包含的檔案。
4. 處理 EXCLUDE 參數以取消選取包含的檔案。
5. 決定是否要刪除傾印的檔案。
6. 將要處理的檔案排入佇列。
7. 複製檔案。
8. 將複製的檔案編目 DCB 資訊匯出至目標位置的側邊檔案，以協助進行未來的 RESTORE 操作。

語法

```
DUMP
TARGET ( TARGET LOCATION  )    -
INCLUDE ( DSN. )
[ EXCLUDE ( DSN ) ]
[ CANCEL | IGNORE ]
[ DELETE ]
```

必要參數

以下是 DUMP 的必要參數：

- SYSPRINT DD NAME - 包含其他記錄資訊
- TARGET - 目標位置。它可以是：
  - 傾印位置的完整路徑
  - 在 M2DFUTILS_BASE_LOC 變數中定義的位置中建立的子目錄名稱
- INCLUDE - 單一名稱 DSNAME 或有效的大型主機 DSN 搜尋字串
- EXCLUDE - 單一命名 DSNAME 或有效的大型主機 DSN 搜尋字串

選用的參數

- CANCEL - 如果發生任何錯誤，請取消。已處理的檔案將保留
- （預設）IGNORE - 忽略任何錯誤並處理直到結束
- DELETE - 如果沒有發生 ENQ 錯誤，則會刪除檔案，且不會產生目錄

DELETE

提供大量刪除和取消目錄檔案的功能。檔案未備份。

流程

DELETE 將執行下列動作：

1. 透過處理 INCLUDE 參數來決定要包含的檔案。
2. 處理 EXCLUDE 參數以取消選取包含的檔案。
3. 將要處理的檔案排入佇列。將處置設定為 OLD、DELETE、KEEP。

語法

```
DELETE
INCLUDE ( DSN )
[ EXCLUDE ( DSN ) ]
[ CANCEL | IGNORE ]
[ DELETE ]
```

必要參數

以下是 DELETE 的必要參數：

- SYSPRINT DD NAME - 包含其他記錄資訊
- INCLUDE - 單一名稱 DSNAME 或有效的大型主機 DSN 搜尋字串
- EXCLUDE - 單一命名 DSNAME 或有效的大型主機 DSN 搜尋字串

選用的參數

- CANCEL - 如果發生任何錯誤，請取消。處理的檔案將保留
- （預設）IGNORE - 忽略任何錯誤並處理直到結束

RESTORE

提供使用 DUMP 還原先前備份檔案的功能。除非使用 RENAME 來變更還原的 DSNAME，否則檔案會還原至原始目錄位置。

流程

RESTORE 將執行下列動作：

1. 驗證來源位置目錄。
2. 透過處理目錄匯出檔案來決定要包含的檔案。
3. 處理 EXCLUDE 參數以取消選取包含的檔案。
4. 將要處理的檔案排入佇列。
5. 未根據匯出資訊編製目錄的目錄檔案。
6. 如果檔案已編製目錄，且匯出目錄資訊相同，則 RESTORE 會在設定 REPLACE 選項時取代已編製目錄的資料集。

語法

```
RESTORE
SOURCE ( TARGET LOCATION )
INCLUDE ( DSN )
[ EXCLUDE ( DSN ) ]
[ CANCEL | IGNORE ]
[ REPLACE]
```

必要參數

以下是 RESTORE 的必要參數：

- SYSPRINT DD NAME - 包含其他記錄資訊
- SOURCE - 來源位置。它可以是：
  - 傾印位置的完整路徑
  - 在 M2DFUTILS_BASE_LOC 變數中定義的位置中建立的子目錄名稱
- INCLUDE - 單一名稱 DSNAME 或有效的大型主機 DSN 搜尋字串
- EXCLUDE - 單一命名 DSNAME 或有效的大型主機 DSN 搜尋字串

選用的參數

- CANCEL - 如有任何錯誤，請取消。已處理的檔案已保留
- （預設）IGNORE - 忽略任何錯誤並處理直到結束
- REPLACE - 如果還原的檔案已編製目錄，且目錄記錄相同，則取代已編製目錄的檔案

範例 JCLs

DUMP 任務

此任務會建立名為 的子目錄TESTDUMP。這是 M2DFUTILS_BASE_LOC 變數指定的預設備份位置。它會為此名為 的備份建立 PDS 程式庫M2DFUTILS.TESTDUMP。匯出的目錄資料會存放在名為 的備份目錄中的行序列檔案中CATDUMP.DAT。所有選取的檔案都會複製到此備份目錄。

```
//M2DFDMP JOB 'M2DFDMP',CLASS=A,MSGCLASS=X
//STEP001  EXEC PGM=M2DFUTIL
//SYSPRINT DD DSN=TESTDUMP.SYSPRINT,
//        DISP=(NEW,CATLG,DELETE),
//        DCB=(RECFM=LSEQ,LRECL=256)
//SYSIN    DD *
DUMP TARGET(TESTDUMP)               -
     INCLUDE(TEST.FB.FILE*.ABC)     -
 CANCEL
/*
//
```

DELETE 任務

此任務將從目錄中刪除符合 INCLUDE 參數的所有檔案。

```
/M2DFDEL JOB 'M2DFDEL',CLASS=A,MSGCLASS=X
//STEP001  EXEC PGM=M2DFUTIL
//SYSPRINT DD DSN=TESTDEL.SYSPRINT,
//        DISP=(NEW,CATLG,DELETE),
//        DCB=(RECFM=LSEQ,LRECL=256)
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
  DELETE                            -
     INCLUDE(TEST.FB.FILE*.ABC)     -
 CANCEL
/*
```

```
 //
```

RESTORE 任務

此任務將從TESTDUMP備份位置還原符合 INCLUDE 參數的檔案。如果目錄檔案與 CATDUMP 匯出中的檔案相同，並指定 REPLACE 選項，則會取代目錄檔案。

```
//M2DFREST JOB 'M2DFREST',CLASS=A,MSGCLASS=X
//STEP001  EXEC PGM=M2DFUTIL
////SYSPRINT DD DSN=TESTREST.SYSPRINT,
//         DISP=(NEW,CATLG,DELETE),
//         DCB=(RECFM=LSEQ,LRECL=256)
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
RESTORE SOURCE(TESTDUMP)                -
     INCLUDE(TEST.FB.FILE*.ABC)         -
 IGNORE
 REPLACE
/*
//
```

M2RUNCMD 批次公用程式

您可以使用批次公用程式 M2RUNCMD，直接從 JCL 執行 Rocket Software （先前稱為 Micro Focus) 命令、指令碼和系統呼叫，而不是從終端機或命令提示字元執行它們。來自命令的輸出會記錄到批次任務的多工緩衝處理日誌。

主題

- 支援平台
- 在 Amazon EC22 上為大型主機現代化執行時間設定 M2RUNCMD （包括 AppStream 2.0) AWS
- 範例 JCLs

支援平台

您可以在下列平台上使用 M2RUNCMD：

- Rocket 軟體執行期 （在 Amazon EC2 上）
- Rocket Software Enterprise Developer (ED) 和 Rocket Software Enterprise Server (ES) 產品的所有變體。

在 Amazon EC22 上為大型主機現代化執行時間設定 M2RUNCMD （包括 AppStream 2.0) AWS

如果您遷移的應用程式在 Amazon EC2 上的 AWS Mainframe Modernization 執行時間上執行，請設定 M2RUNCMD，如下所示。

- 變更 Micro Focus JES 程式路徑，以包含批次公用程式的二進位位置。如果您必須指定多個路徑，請使用冒號 (:) 來分隔 Linux 上的路徑和 Windows 上的分號 (;)。
  - Linux：`/opt/aws/m2/microfocus/utilities/64bit`
  - Windows (32 位元）： `C:\AWS\M2\MicroFocus\Utilities\32bit`
  - Windows (64 位元）： `C:\AWS\M2\MicroFocus\Utilities\64bit`

範例 JCLs

若要測試安裝，您可以使用下列其中一個範例 JCLs。

RUNSCRL1.jcl

此範例 JCL 會建立指令碼並執行指令碼。第一個步驟會建立名為 `/tmp/TEST_SCRIPT.sh`和 的指令碼，其中包含來自串流SYSUT1中資料的內容。第二個步驟會設定執行許可，並執行第一個步驟中建立的指令碼。您也可以選擇只執行第二個步驟來執行現有的 Rocket 軟體和系統命令。

```
//RUNSCRL1 JOB 'RUN SCRIPT',CLASS=A,MSGCLASS=X,TIME=1440
//*
//*
//*--------------------------------------------------------------------*
//*  CREATE SCRIPT (LINUX)
//*--------------------------------------------------------------------*
//*
//STEP0010 EXEC PGM=IEBGENER
//*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//SYSUT1   DD *
#!/bin/bash

set -x

## ECHO PATH ENVIRONMNET VARIABLE
```

```
echo $PATH

## CLOSE/DISABLE VSAM FILE
casfile -r$ES_SERVER -oc  -ed -dACCTFIL

## OPEN/ENABLE VSAM FILE
casfile -r$ES_SERVER -ooi -ee -dACCTFIL

exit $?
/*
//SYSUT2   DD DSN=&&TEMP,
//            DISP=(NEW,CATLG,DELETE),
//            DCB=(RECFM=LSEQ,LRECL=300,DSORG=PS,BLKSIZE=0)
//*MFE: %PCDSN='/tmp/TEST_SCRIPT.sh'
//*
//*--------------------------------------------------------------------*
//*   RUN SCRIPT (LINUX)                                               *
//*--------------------------------------------------------------------*
//*
//STEP0020 EXEC PGM=RUNCMD
//*
//SYSOUT  DD  SYSOUT=*
//*
//SYSIN   DD *
*RUN SCRIPT
 sh /tmp/TEST_SCRIPT.sh
/*
//
```

SYSOUT

執行中命令或指令碼的輸出會寫入SYSOUT日誌。對於每個執行的命令，它會顯示命令、輸出和傳回程式碼。

```
************ CMD Start ************

CMD_STR: sh /tmp/TEST_SCRIPT.sh

CMD_OUT:


+ echo /opt/microfocus/EnterpriseServer/bin:/sbin:/bin:/usr/sbin:/usr/bin
/opt/microfocus/EnterpriseServer/bin:/sbin:/bin:/usr/sbin:/usr/bin
```

```
+ casfile -rMYDEV -oc -ed -dACCTFIL

-Return Code:   0

Highest return code:    0

+ casfile -rMYDEV -ooi -ee -dACCTFIL

-Return Code:   8

Highest return code:    8

+ exit 8


CMD_RC=8


************  CMD End  ************
```

RUNCMDL1.jcl

此範例 JCL 使用 RUNCMD 來執行多個命令。

```
//RUNCMDL1 JOB 'RUN CMD',CLASS=A,MSGCLASS=X,TIME=1440
//*
//*
//*--------------------------------------------------------------------*
//*   RUN SYSTEM COMMANDS                                              *
//*--------------------------------------------------------------------*
//*
//STEP0001 EXEC PGM=RUNCMD
//*
//SYSOUT  DD  SYSOUT=*
//*
//SYSIN   DD *
*LIST DIRECTORY
 ls
*ECHO PATH ENVIRONMNET VARIABLE
 echo $PATH
/*
//
```

AWS 大型主機現代化中的檔案傳輸

AWS Mainframe Modernization File Transfer 可讓您將大型主機資料集傳輸和轉換為 Amazon S3，以用於大型主機現代化、遷移和擴增使用案例。它簡化了將資料集從大型主機傳輸到 的程序 AWS 雲端。主要功能包括：探索來源大型主機資料集和成品，以及可擴展性和效率，以更快地將資料傳輸到 Amazon S3。檔案傳輸支援各種大型主機資料集類型，例如循序、PDS、GDS、GDG 和 VSAM KSDS。服務會將資料集傳輸到中繼 Amazon S3 儲存貯體，將其轉換為指定的目標碼頁面，然後將它們移動到所需的目標 S3 儲存貯體。

主題

- 什麼是 AWS Mainframe Modernization File Transfer？
- 安裝檔案傳輸代理程式
- 設定檔案傳輸代理程式
- 建立資料傳輸端點以進行檔案傳輸
- 在檔案傳輸中建立傳輸任務
- 教學課程：開始使用 AWS Mainframe Modernization File Transfer
- AWS Mainframe Modernization File Transfer 中支援的來源和目標編碼

什麼是 AWS Mainframe Modernization File Transfer？

使用 AWS Mainframe Modernization File Transfer，您可以使用全受管服務傳輸和轉換資料集和檔案，以加速和簡化 AWS 大型主機現代化服務和 Amazon S3 的現代化、遷移和擴增使用案例。

主題

- AWS Mainframe Modernization File Transfer 的優點
- AWS Mainframe Modernization File Transfer 的運作方式

AWS Mainframe Modernization File Transfer 的優點

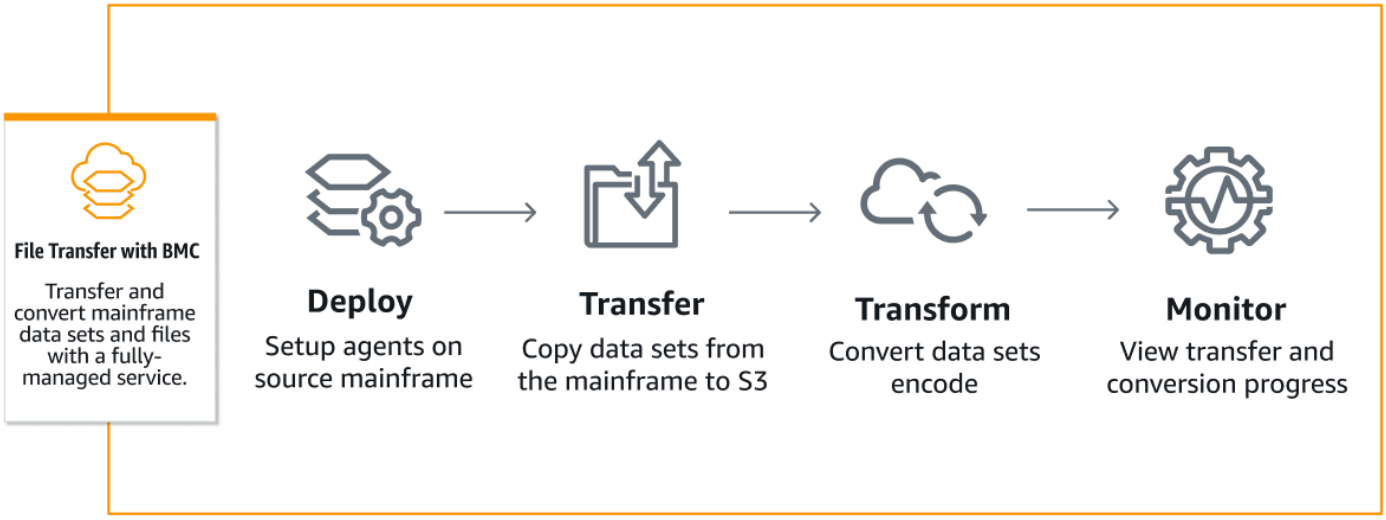
AWS Mainframe Modernization File Transfer 可協助您將資料集從大型主機傳輸到 Amazon S3。一些優點包括：

- 探索來源大型主機資料集和成品
- 自動化傳輸和資料集轉換

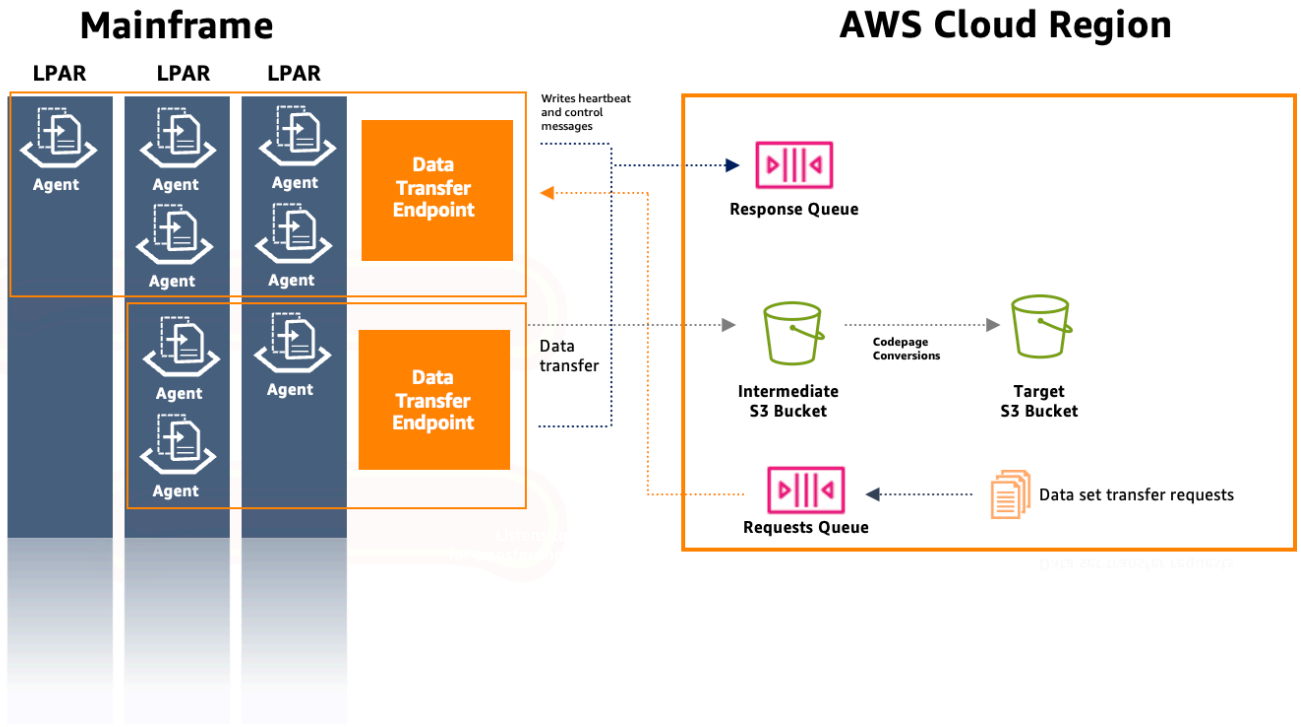
- 可擴展性、效率和速度，以實現更快的資料集傳輸到 AWS

AWS Mainframe Modernization File Transfer 的運作方式

下圖概述 AWS Mainframe Modernization File Transfer 在概念層面的運作方式。



下圖是 AWS Mainframe Modernization File Transfer 功能的架構概觀。



安裝檔案傳輸代理程式

您可以使用本文件做為step-by-step指南，在來源大型主機上安裝 代理程式。

> **Note**
>
> 本指南僅適用於大型主機系統程式設計人員。

主題

- 步驟 1：為 M2-agent程式建立 zFS 資料集
- 步驟 2：將資料集格式化為 zFS
- 步驟 3：掛載 檔案系統
- 步驟 4：驗證掛載
- 步驟 5：輸入 OMVS
- 步驟 6：設定代理程式安裝目錄環境變數
- 步驟 7：設定工作目錄環境變數
- 步驟 8：建立工作目錄
- 步驟 9：複製代理程式 tar 檔案並複製工作目錄
- 步驟 10：完成代理程式安裝

步驟 1：為 M2-agent程式建立 zFS 資料集

使用下列 JCL（工作控制語言）為 M2-agent安裝建立 zFS：

```
DEFINE   EXEC   PGM=IDCAMS
SYSPRINT DD     SYSOUT=A
SYSIN    DD     *
 DEFINE CLUSTER (NAME(yourhlq.M2AGENT.ZFS) -
 VOLUMES(*) -
 LINEAR CYL(1000 200))
```

步驟 2：將資料集格式化為 zFS

建立資料集後，將其格式化為 zFS 檔案系統。

其中一種方法是使用下列 JCL：

```
FORMAT  EXEC PGM=IOEAGFMT,
PARM=('-aggregate yourhlq.M2AGENT.ZFS -size 1200') ,
SYSPRINT DD  SYSOUT=*
```

提交此任務並檢查是否成功完成。

步驟 3：掛載 檔案系統

若要掛載檔案系統，請使用 MOUNT命令。您可以在 ISPF 或 批次命令列中掛載檔案系統。

例如：

```
MOUNT FILESYSTEM('yourhlq.M2AGENT.ZFS') TYPE(ZFS) MODE(RDWR) MOUNTPOINT('/usr/lpp/aws/
m2-agent')
```

您將在步驟 6 中使用此掛載點。

> **Note**
>
> 定義掛載路徑是選用的，您應該為此使用現有的目錄。

步驟 4：驗證掛載

使用 D OMVS,F命令或在 Unix System Service (USS) 中檢查 ，確認檔案系統已正確掛載。

步驟 5：輸入 OMVS

使用下列命令來輸入 OMVS：

```
TSO OMVS
```

步驟 6：設定代理程式安裝目錄環境變數

使用下列命令來設定代理程式安裝目錄環境：

```
export AGENT_DIR=/usr/lpp/aws/m2-agent
```

> **Note**
>
> 掛載點在步驟 3 中定義。

步驟 7：設定工作目錄環境變數

使用下列命令來設定工作目錄環境變數：

```
export WORK_DIR=$AGENT_DIR/tmp
```

步驟 8：建立工作目錄

使用下列命令來設定工作目錄環境：

```
mkdir -p $WORK_DIR
```

步驟 9：複製代理程式 tar 檔案並複製工作目錄

使用 M2 代理程式連結從 AWS 下載代理程式 tar 檔案。

傳輸機制將取決於您的環境，但請確定 tar 檔案是以二進位模式傳輸。

步驟 10：完成代理程式安裝

請依照下列步驟完成代理程式安裝。

1. 使用下列命令，將 m2-agent 版本環境變數設定為目前正在安裝的版本：

   ```
   export M2_AGENT_VERSION=1.0.0
   ```

2. 使用下列命令解壓縮代理程式 tar 套件：

   ```
   tar -xpf m2-agent-$M2_AGENT_VERSION.tar -C $AGENT_DIR
   ```

3. 使用下列命令建立目前代理程式安裝目錄的`current-version`符號連結：

```
ln -s $AGENT_DIR/m2-agent-v$M2_AGENT_VERSION $AGENT_DIR/current-version
```

4. 更新並提交CPY#PDS以建立檔案傳輸代理程式資料集。

> **Note**
>
> JCL 使用 SYS2.AWS.M2 HLQ。

若要建立檔案傳輸代理程式，請更新三個符號變數 HLQ （高階限定詞）VOLSER、 和 AGNTPATH，以供稍後在 JCL 中使用：

```
oedit $AGENT_DIR/current-version/installation/CPY#PDS
```

> **Note**
>
> 此 JCL 專為在大型主機上設定代理程式安裝的特定層面量身打造。它會配置必要的資料集，然後將特定檔案從 Unix 檔案系統複製到這些資料集。

設定檔案傳輸代理程式

安裝檔案傳輸代理程式後，請依照下列步驟設定代理程式。如果您需要安裝新的代理程式，請遵循the section called “安裝檔案傳輸代理程式”頁面上的指示。

主題

- 步驟 1：設定許可和已啟動的任務控制 (STC)
- 步驟 2：建立 Amazon S3 儲存貯體
- 步驟 3：建立用於加密 AWS KMS 的客戶受管金鑰
- 步驟 4：建立大型主機登入資料的 AWS Secrets Manager 秘密
- 步驟 5：建立 IAM 政策
- 步驟 6：建立具有長期存取憑證的 IAM 使用者
- 步驟 7：為要擔任的代理程式建立 IAM 角色
- 步驟 8：客服人員組態

步驟 1：設定許可和已啟動的任務控制 (STC)

1. 根據其指示更新並提交其中一個 SYS2.AWS.M2.SAMPLIB(SEC#RACF)（用於設定 RACF 許可）或 SYS2.AWS.M2.SAMPLIB(SEC#TSS)（用於設定 TSS 許可）。這些成員是由上一個CPY#PDS步驟建立的。

   > **Note**
   >
   > SYS2.AWS.M2 應該以安裝期間選擇的高階限定詞 (HLQ) 取代。

2. 如果預設檔案傳輸代理程式目錄路徑 (/usr/lpp/aws/m2-agent) 已變更，請在 SYS2.AWS.M2.SAMPLIB(M2AGENT) STC JCL 中更新 PWD 匯出。
3. 根據您的站點標準更新 PROC：

   a. 根據您的安裝需求更新 PROC 卡。

   b. 使用 更新 STEPLIBM2 LOADLIB PDSE ALIAS。

   c. 編輯 PWD 以指向代理程式安裝路徑（僅包含此項目）。

   d. JAVA_HOME 視需要更新。

4. 將 SYS2.AWS.M2.SAMPLIB(M2AGENT) JCL 更新並複製到 SYS1.PROCLIB或PROCLIB串連中的其中一個 PROCLIBs。
5. 使用下列命令將 SYS2.AWS.M2.LOADLIB新增至 APF 清單：

   ```
   SETPROG APF ADD DSNAME(SYS2.AWS.M2.LOADLIB) SMS
   ```

6. 將客服人員的群組和擁有者設定為客服人員使用者/群組 (M2USER/M2GROUP)。在 OMVS 中使用下列命令：

   ```
   chown -R M2USER:M2GROUP $AGENT_DIR/current-version
   ```

   > **Note**
   >
   > 使用您在安全定義任務中使用的名稱編輯 M2USER 和 M2GROUP。

步驟 2：建立 Amazon S3 儲存貯體

AWS Mainframe Modernization File Transfer 需要中繼 Amazon S3 儲存貯體做為工作區。我們建議您特別為此建立儲存貯體。

或者，為傳輸的資料集建立新的目標 Amazon S3 儲存貯體。否則，您也可以使用現有的 Amazon S3 儲存貯體。如需建立 Amazon S3 儲存貯體的詳細資訊，請參閱建立儲存貯體。

步驟 3：建立用於加密 AWS KMS 的客戶受管金鑰

在 中建立客戶受管金鑰 AWS KMS

1. 在 開啟 AWS KMS 主控台https://console.aws.amazon.com/kms。
2. 在左側導覽窗格中選擇客戶受管金鑰。
3. 選擇建立金鑰。
4. 在設定金鑰下，選擇金鑰類型做為對稱，選擇金鑰用量做為加密和解密。使用其他預設組態。
5. 選擇下一步。
6. 在新增標籤中，為您的金鑰新增別名和描述。
7. 選擇下一步。
8. 在定義金鑰管理許可下，選擇至少一個管理此金鑰的 IAM 使用者和角色。
9. 選擇下一步。
10. 或者，在定義金鑰管理許可下，選擇至少一個可以使用此金鑰的 IAM 使用者和角色。
11. 選擇下一步。
12. 在編輯金鑰政策區段中，選擇編輯，然後將下列語法新增至金鑰政策。這可讓 AWS Mainframe Modernization 服務讀取和使用這些金鑰進行加密/解密。

> **Important**
>
> 將陳述式新增至現有的陳述式。請勿取代政策中已有的內容。

```
{
    "Sid" : "Enable AWS M2 File Transfer Permissions",
    "Effect" : "Allow",
    "Principal" : {
        "Service" : "m2.amazonaws.com"
```

```
    },
    "Action" : [
        "kms:Encrypt",
        "kms:Decrypt"
    ],
   "Resource" : "*"
},
```

13. 選擇下一步。
14. 在檢閱頁面上，檢查所有詳細資訊，然後選擇完成。

開啟新建立的 KMS 金鑰，複製並儲存客戶受管金鑰的 ARN。稍後將在政策中使用它。

步驟 4：建立大型主機登入資料的 AWS Secrets Manager 秘密

需要大型主機登入資料才能存取要傳輸的資料集，而且這些必須儲存為 AWS Secrets Manager 秘密。

建立 AWS Secrets Manager 秘密

1. 在 開啟 Secrets Manager 主控台https://console.aws.amazon.com/secretsmanager。
2. 選擇儲存新機密。
3. 在選擇秘密類型中，選擇其他類型的秘密。
4. 針對可存取資料集`userId`的大型主機 userId，使用金鑰值。使用密碼欄位`password`的鍵值。
5. 針對加密金鑰，選擇先前建立 AWS 的客戶受管金鑰。
6. 選擇下一步。
7. 在設定秘密頁面上，提供名稱和描述。
8. 在相同頁面上，編輯資源許可，並使用下列資源政策，讓 AWS Mainframe Modernization 服務可以存取它。

```
{
  "Version" : "2012-10-17",
  "Statement" : [ {
    "Effect" : "Allow",
    "Principal" : {
        "Service" : "m2.amazonaws.com"
    },
    "Action" : [ "secretsmanager:GetSecretValue",
                 "secretsmanager:DescribeSecret" ],
    "Resource" : "*"
```

```
  } ]
}
```

9. 選擇儲存以儲存更新的許可。
10. 選擇下一步。
11. 略過設定輪換頁面，然後選擇下一步。
12. 在檢閱頁面上，檢查所有組態，然後選擇儲存以儲存秘密。

> **Important**
>
> `userId` 和 `password`私密金鑰區分大小寫，且必須輸入，如下所示。

步驟 5：建立 IAM 政策

建立具有代理程式所需許可的新政策

1. 在 https://console.aws.amazon.com/iam 開啟 IAM 主控台。
2. 在存取管理下選擇政策。
3. 選擇建立政策。
4. 在指定許可頁面的政策編輯器下，從視覺化編輯器切換至 JSON 編輯器，並以下列範本取代內容：
5.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
         "Sid": "FileTransferAgentSQSReceive",
            "Effect": "Allow",
            "Action": [
                "sqs:DeleteMessage",
                "sqs:ReceiveMessage"
            ],
            "Resource": "arn:aws:sqs:*:111122223333:m2-*-request-queue.fifo"
        },
        {
         "Sid": "FileTransferAgentSQSSend",
            "Effect": "Allow",
            "Action": "sqs:SendMessage",
```

```
            "Resource": "arn:aws:sqs:*:111122223333:m2-*-response-queue.fifo"
        },
        {
         "Sid": "FileTransferWorkingS3",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "<file-transfer-endpoint-intermediate-bucket-arn>/*"
        },
        {
         "Sid": "FileTransferAgentKMSDecrypt",
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "<kms-key-arn>"
        }
    ]
}
```

6. 111122223333 將 request-queue 和 response-queue ARN 中的 取代為您的帳戶。

> **Note**
>
> 這些是萬用字元 ARN，符合在資料傳輸端點初始化期間建立的兩個 Amazon SQS 佇列。建立檔案傳輸端點後，可選擇將這些 ARN 取代為來自 Amazon SQS 的實際值。

7. file-transfer-endpoint-intermediate-bucket-arn 將 取代為先前建立之傳輸儲存貯體的 ARN。將「/*」萬用字元保留在結尾。
8. kms-key-arn 將 取代為先前建立之 AWS KMS 金鑰的 ARN。
9. 選擇下一步。
10. 在檢閱和建立頁面上，新增政策名稱和描述。
11. 選擇建立政策。

步驟 6：建立具有長期存取憑證的 IAM 使用者

建立允許大型主機代理程式連線到您 AWS 帳戶的 IAM 使用者。代理程式會與此使用者連線，然後擔任您定義的角色，其具有使用 Amazon SQS 回應和請求佇列以及將資料集儲存至 Amazon S3 儲存貯體的許可。

建立此 IAM 使用者

1. 導覽至位於 的 IAM 主控台https://console.aws.amazon.com/iam。

2. 在存取管理下選擇使用者。
3. 選擇 Create user (建立使用者)。
4. 在使用者詳細資訊下新增有意義的使用者名稱。例如 `Configure-ft-agent`。
5. 選擇下一步。
6. 在許可選項中，選擇直接連接政策選項，但不連接任何許可政策。這些許可將由要連接的角色管理。
7. 選擇下一步。
8. 檢閱詳細資訊，然後選擇建立使用者。
9. 建立使用者後，請選擇使用者並開啟安全登入資料索引標籤。
10. 在存取金鑰之下選擇建立存取金鑰。
11. 然後，在出現使用案例提示時選擇其他。
12. 選擇下一步。
13. 或者，您可以設定描述標籤，例如 `Access key for configuring file transfer agent`。
14. 選擇 Create access key (建立新的存取金鑰)。
15. 複製並安全地儲存產生的存取金鑰和私密存取金鑰。這些將在稍後使用。

如需建立 IAM 存取金鑰的詳細資訊，請參閱管理 IAM 使用者的存取金鑰。

> **Important**
>
> 在選擇完成之前，儲存存取金鑰和存取金鑰建立精靈最後一頁上顯示的私密存取金鑰。這些金鑰用於設定大型主機代理程式，之後無法擷取。

> **Note**
>
> 儲存用來設定與 IAM 角色信任關係的 IAM 使用者 ARN。

步驟 7：為要擔任的代理程式建立 IAM 角色

為客服人員建立新的 IAM 角色

1. 在 的 IAM 主控台中選擇角色https://console.aws.amazon.com/iam。

2. 選擇建立角色。
3. 在選取信任的實體頁面上，選擇信任的實體類型的自訂信任政策。
4. 將自訂信任政策取代為下列項目，並將 取代<iam-user-arn>為先前建立的使用者 ARN。

```
{
    "Version": "2012-10-17",
    "Statement": [ {
         "Sid": "FileTransferAgent",
         "Effect": "Allow",
         "Principal": {
            "AWS": "<IAM-User-arn>"
         },
         "Action": "sts:AssumeRole"
    } ]
}
```

5. 選擇下一步。
6. 在新增許可中，篩選您先前建立的政策名稱，然後選擇它。
7. 選擇下一步。
8. 命名角色，然後選擇建立角色。

> **Note**
>
> 儲存角色名稱，稍後將用於設定大型主機代理程式。

步驟 8：客服人員組態

設定檔案傳輸代理程式

1. 導覽至 $AGENT_DIR/current-version/config。
2. 編輯代理程式的組態檔案appication.properties，以使用下列命令新增環境組態：

```
oedit $AGENT_DIR/current-version/config/application.properties
```

例如：

```
agent.environments[0].account-id=<AWS_ACCOUNT_ID>
```

```
agent.environments[0].agent-role-name=<AWS_IAM_ROLE_NAME>
agent.environments[0].access-key-id=<AWS_IAM_ROLE_ACCESS_KEY>
agent.environments[0].secret-access-id=<AWS_IAM_ROLE_SECRET_KEY>
agent.environments[0].bucket-name=<AWS_S3_BUCKET_NAME>
agent.environments[0].environment-name=<AWS_REGION>
agent.environments[0].region=<AWS_REGION>
zos.complex-name=<File_Transfer_Endpoint_Name>
```

其中：

- AWS_ACCOUNT_ID 是 AWS 帳戶的 ID。
- AWS_IAM_ROLE_NAME 是在 中建立的 IAM 角色名稱the section called “步驟 7：為要擔任的代理程式建立 IAM 角色”。
- AWS_IAM_ROLE_ACCESS_KEY 是 中建立之 IAM 使用者的存取金鑰the section called “步驟 6：建立具有長期存取憑證的 IAM 使用者”。
- AWS_IAM_ROLE_SECRET_KEY 是在 中建立之 IAM 使用者的存取私密金鑰the section called “步驟 6：建立具有長期存取憑證的 IAM 使用者”。
- AWS_S3_BUCKET_NAME 是使用資料傳輸端點建立的傳輸儲存貯體名稱。
- AWS_REGION 是您設定檔案傳輸代理程式的區域。

> **Note**
>
> 您可以透過定義多個環境，將檔案傳輸代理程式傳輸到 AWS 中的多個區域和帳戶。

- （選用）。 zos.complex-name 是您在建立檔案傳輸端點時建立的複雜名稱。

> **Note**
>
> 只有在您想要自訂與您建立檔案傳輸端點時所定義相同的複雜名稱（預設為 sysplex 名稱）時，才需要此欄位。如需詳細資訊，請參閱the section called “建立資料傳輸端點”。

> **Important**
>
> 只要括號中的索引 [0]- 每個區段都會遞增，就可以有數個這類區段。

您必須重新啟動代理程式，變更才會生效。

需求

1. 新增或移除參數時，必須停止並啟動代理程式。在 CLI 中使用下列命令啟動檔案傳輸代理程式：

```
/S M2AGENT
```

若要停止 M2 代理程式，請在 CLI 中使用下列命令：

```
/P M2AGENT
```

2. 您可以透過定義環境項目，將檔案傳輸代理程式設定為將資料傳輸到 AWS 中的多個區域和帳戶。

> **Note**
>
> 將值取代為您先前建立和設定的參數值。

```
#Region 1
agent.environments[0].account-id=AWS_ACCOUNT_ID
agent.environments[0].agent-role-name=AWS_IAM_ROLE_NAME
agent.environments[0].access-key-id=AWS_IAM_ROLE_ACCESS_KEY
agent.environments[0].secret-access-id=AWS_IAM_ROLE_SECRET_KEY
agent.environments[0].bucket-name=AWS_S3_BUCKET_NAME
agent.environments[0].environment-name=AWS_REGION
agent.environments[0].region=AWS_REGION

#Region 2
agent.environments[1].account-id=AWS_ACCOUNT_ID
agent.environments[1].agent-role-name=AWS_IAM_ROLE_NAME
agent.environments[1].access-key-id=AWS_IAM_ROLE_ACCESS_KEY
agent.environments[1].secret-access-id=AWS_IAM_ROLE_SECRET_KEY
agent.environments[1].bucket-name=AWS_S3_BUCKET_NAME
agent.environments[1].environment-name=AWS_REGION
agent.environments[1].region=AWS_REGION
```

建立資料傳輸端點以進行檔案傳輸

資料傳輸端點可讓您與來源大型主機連線，並支援客服人員的高可用性、可擴展性和簡化管理。個別代理程式安裝在大型主機 LPARs上，並可以分組到資料傳輸端點。發出傳輸資料集的請求時，資料傳輸端點中的一個代理程式會處理該特定傳輸。若要啟動資料傳輸，資料傳輸端點上至少必須有一個代理程式在線上。

此程序假設您已完成 中的步驟，設定 AWS 大型主機現代化並在來源大型主機上設定檔案傳輸代理程式。

建立資料傳輸端點

若要為檔案傳輸建立資料傳輸端點，請遵循 AWS Mainframe Modernization 主控台中的下列步驟。

建立資料傳輸端點

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 在 AWS 區域 選擇器中，選擇您要將檔案從大型主機傳輸到 Amazon S3 儲存貯體的區域。
3. 在資料傳輸端點頁面的檔案傳輸下，選擇建立資料傳輸端點。
4. 在資料傳輸端點先決條件頁面上，閱讀所有指示，確認您已完成來源大型主機上的這些步驟。確認後，選擇下一步。
5. 在設定資料傳輸端點頁面上，為您的資料傳輸端點新增基本資訊。

    1. 在基本資訊區段中，輸入您的資料傳輸端點名稱。

        > **Note**
        >
        > 資料傳輸端點名稱必須符合 Sysplex 名稱，除非您在代理程式組態中指定複雜的名稱。

    2. 選擇性的描述。
    3. 用來加密秘密的 KMS 金鑰。

        > **Note**
        >
        > 您必須為 KMS 新增下列以資源為基礎的政策，讓 AWS Mainframe Modernization 服務可以讀取和使用這些金鑰進行加密/解密：
        >
        > ```
        > {
        > ```

```
  "Sid" : "Enable AWS M2 Permissions",
  "Effect" : "Allow",
  "Principal" : {
      "Service" : "m2.amazonaws.com"
           },
   "Action" : [
       "kms:Encrypt",
       "kms:Decrypt"
             ],
   "Resource" : "*"
}
```

4. 指定中繼資料的 S3 位置，這是中繼 S3 位置，在轉換和傳輸至目標 Amazon S3 儲存貯體之前，會先存放從大型主機傳輸的資料集。

   .

   > **Note**
   >
   > 建議您為傳輸任務建立新的 Amazon S3 儲存貯體。如需詳細資訊，請參閱建立儲存貯體。您也可以選擇瀏覽 S3 選項來瀏覽現有的 Amazon S3 儲存貯體。 S3

5. 輸入必要欄位後，選擇下一步。
6. 在檢閱和建立資料傳輸端點頁面上，檢查您是否已完成先決條件，並檢閱基本資訊。確認後，選擇建立資料傳輸端點。

系統會將您重新導向至資料傳輸端點概觀頁面，您可以在其中查看所有資料傳輸端點的清單。您也可以看到可用的或失敗的資料傳輸端點。

您也可以依名稱搜尋資料傳輸端點，並存取每個可用代理程式的其他資訊。

在檔案傳輸中建立傳輸任務

傳輸任務用於指定要從大型主機傳輸到 Amazon S3 的資料集，並允許您選擇程式碼頁面轉換選項。

這些指示假設您已完成 中的步驟，設定 AWS 大型主機現代化並已建立 the section called “建立資料傳輸端點”。

主題

- 建立傳輸任務
- 檢視傳輸任務

建立傳輸任務

若要在檔案傳輸中建立傳輸任務，請遵循 AWS Mainframe Modernization 主控台中的下列步驟。

建立傳輸任務

> **Important**
> 您必須至少有一個資料傳輸端點，才能建立新的傳輸任務。

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 在 AWS 區域 選擇器中，選擇您要將檔案從大型主機傳輸到 Amazon S3 儲存貯體的區域。
3. 在傳輸任務頁面上，您可以選擇任何資料傳輸端點來建立傳輸任務。
4. 在建立傳輸任務頁面上，設定傳輸任務的屬性。如果您先前尚未建立任何轉移任務，您可以選擇建立轉移任務選項來建立第一個轉移任務。

   - 在此頁面上，輸入傳輸任務的基本資訊，包括傳輸任務名稱、描述和私密金鑰。

   > **Note**
   > - 使用資料傳輸端點所定義的 KMS 金鑰來加密秘密。秘密應包含使用 `userId`和 `password`金鑰存取大型主機上資料集所需的大型主機登入資料。如需詳細資訊，請參閱 AWS Secrets Manager 秘密。
   > - 您必須使用下列資源型政策設定私密金鑰，讓 AWS Mainframe Modernization 服務可以存取秘密金鑰來執行資料傳輸任務。

   ```
   {
     "Version" : "2012-10-17",
     "Statement" : [ {
       "Effect" : "Allow",
       "Principal" : {
           "Service" : "m2.amazonaws.com"
       },
   ```

```
    "Action" : [ "secretsmanager:GetSecretValue",
                 "secretsmanager:DescribeSecret" ],
    "Resource" : "*"
  } ]
}
```

> **Note**
>
> 目前支援的傳輸資料集大小上限為 90 GB。

- 接下來，選取目標 Amazon S3 儲存貯體位置，以從大型主機傳輸目標資料集。
- 將會選取先前選擇的資料傳輸端點。您也可以從可用的端點中選取另一個端點。

5. 選擇下一步。
6. 在新增資料集頁面的傳輸任務組態區段中，您可以選擇以二進位模式設定傳輸任務，或轉換和傳輸資料集。

   - 在二進位模式中傳輸選項可讓您略過程式碼頁面轉換並保留其記錄描述項 Word (RDW) 位元組，以傳輸資料集。
   - 傳輸和轉換資料集選項可讓您透過設定資料集的來源和目標程式碼頁面來傳輸資料集。您可以在頁面上查看檔案傳輸的可用程式碼the section called “支援的來源和目標程式碼頁面”頁面。
7. 在搜尋大型主機中輸入您的查詢，尋找要包含在傳輸任務中的大型主機。選擇檢視資料集。

   下列萬用字元可作為大型主機資料集搜尋條件的一部分：

   - 單一星號 (*) 做為限定詞（介於期間或最後一個期間之後）符合該位置的單一限定詞。
   - 限定詞內的單一星號 (*) 符合該位置中的零個或多個字元。
   - 作為限定詞的雙星號 (**)（介於期間或最後一個期間之後）符合該位置的零個或多個限定詞。
   - 限定詞內的雙星號 (**) 不是有效的查詢。
   - 單一百分比符號 (%) 符合該位置中的任何單一英數字元或國家字元。每個限定詞最多可以使用 8% 的符號。

> **Note**
>
> 我們建議一律以句點結尾您的搜尋條件，後接雙星號 (.**)，然後視需要進一步縮小搜尋範圍。

如需萬用字元規則的詳細資訊，請參閱 IBM 文件中的篩選資料集名稱。

8. 這些資料集會在大型主機資料集區段下載入，您可以在其中搜尋或選擇一或多個資料集，以設定程式碼頁面轉換。這些選擇的資料集會顯示在新增的資料集區段中。如果未載入任何資料集，您需要重新檢視步驟 7。

> **Note**
>
> 您可以從多個搜尋查詢中選取資料集，並將其新增至傳輸任務。

9. 在新增的資料集區段中，您會看到資料集的名稱、類型和磁碟區名稱。

> **Important**
>
> 對於傳輸和轉換資料集選項，您需要為每個所選資料集手動輸入來源碼頁面和目標碼頁面。來源碼頁面是來源資料集格式，而目標碼頁面是目標資料集格式，用於轉換資料集並將其存放在目標 Amazon S3 儲存貯體中。

10. 在新增的資料集區段中確認資料集之後（以及用於傳輸和轉換資料集的來源和目標程式碼頁面選項），請選擇下一步。
11. 在檢閱和建立頁面上，您可以檢閱或編輯傳輸任務的資訊。
12. 然後，選擇建立傳輸任務。

> **Important**
>
> 選擇建立傳輸任務按鈕會開始資料傳輸，這可根據 AWS Mainframe Modernization 定價頁面計費。此計費是根據資料集大小所測量的傳輸資料量 (GB)。

檢視傳輸任務

若要在檔案傳輸中檢視傳輸任務，您必須遵循 AWS Mainframe Modernization 主控台中的下列步驟。

檢視轉移任務

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 在 AWS 區域 選擇器中，選擇您要將檔案從大型主機傳輸到 Amazon S3 儲存貯體的區域。
3. 在傳輸任務頁面上，選取資料傳輸端點以檢視傳輸任務。
4. 對於具有預先存在轉移任務的端點，這些將顯示在轉移任務區段下。您可以選擇從此清單中檢視任何傳輸任務的詳細資訊。

教學課程：開始使用 AWS Mainframe Modernization File Transfer

AWS Mainframe Modernization File Transfer 可讓您針對大型主機現代化、遷移和擴增使用案例傳輸和轉換大型主機資料集。

遵循本教學課程中的步驟，了解 AWS Mainframe Modernization File Transfer 的運作方式。

概觀

檔案傳輸包含下列項目：

1. 要安裝在來源大型主機上的代理程式。
2. 直接從 AWS Mainframe Modernization 管理服務主控台存取資料集探索、傳輸和轉換功能。

身為 使用者，您可以將資料集從大型主機傳輸到 Amazon S3 儲存貯體。

主題

- 步驟 1：將代理程式二進位檔 tar 套件從 轉移到 AWS 大型主機邏輯分割區
- 步驟 2：在來源大型主機上設定檔案傳輸代理程式
- 步驟 3：建立資料傳輸端點
- 步驟 4：建立轉移任務
- 步驟 5：檢視傳輸任務進度

步驟 1：將代理程式二進位檔 tar 套件從 轉移到 AWS 大型主機邏輯分割區

從 M2-agent tar 連結下載 tar 檔案。

步驟 2：在來源大型主機上設定檔案傳輸代理程式

在此步驟中，您會在來源大型主機上設定並啟動 AWS Mainframe Modernization File Transfer 代理程式。代理程式是促進檔案傳輸服務功能與來源大型主機之間的通訊的必要條件。每個大型主機至少需要一個代理程式。可以啟動多個代理程式，以獲得高可用性和增強的可擴展性。

請遵循the section called “設定檔案傳輸代理程式”指南中的指示，在大型主機上完成檔案傳輸代理程式安裝。

步驟 3：建立資料傳輸端點

依照the section called “建立資料傳輸端點”頁面上的步驟建立新的資料傳輸端點。

步驟 4：建立轉移任務

依照the section called “建立傳輸任務”頁面上的步驟來建立和管理傳輸任務。

步驟 5：檢視傳輸任務進度

您可以在 AWS Mainframe Modernization 主控台中檢視傳輸任務的進度。如需詳細資訊，請參閱 the section called “檢視傳輸任務”一節。

AWS Mainframe Modernization File Transfer 中支援的來源和目標編碼

AWS Mainframe Modernization File Transfer 支援各種資料集類型和程式碼頁面轉換選項。

大型主機資料集類型

AWS Mainframe Modernization File Transfer 支援下列大型主機資料集類型：

- 非 VSAM：序列 (PS)、PDS、GDS、GDG
- VSAM 類型：KSDS

支援的程式碼頁面

AWS Mainframe Modernization File Transfer 支援下列程式碼頁面進行資料集轉換（往返）：

"BIG5" ，「BIG5_HKSCS」、 "CESU_8" ，"EUC_JP" ，"EUC_KR" ，"GB18030" ，"GB2312" ，
"GBK" 、 "IBM00858" ，"IBM01140" ，"IBM01141" ，"IBM01142" ，"IBM01143" ，"IBM01144" ，
"IBM01145" ，"IBM01146" ，"IBM01147" ，"IBM01148" ，"IBM01149" ，"IBM037" ，
"IBM1026" ，"IBM1047" ，"IBM273" ，"IBM277" ，"IBM278" ，"IBM280" ，"IBM284" ，
"IBM285" ，"IBM290" ，"IBM297" ，"IBM420" ，"IBM424" ，"IBM437" ，"IBM500" ，
「IBM775」、 "IBM850" ，"IBM852" ，"IBM855" ，"IBM857" ，"IBM860" ，"IBM861" ，
"IBM862" ，"IBM863" ，"IBM864" ，"IBM865" ，"IBM866" ，"IBM868" ，"IBM869" ，
"IBM870" ，"IBM871" ，"IBM918" ，"IBM_THAI" ，"ISO_2022_CN" ，「ISO_2022_JP」，
「ISO_2022_JP_2」、 「ISO_2022_KR」、 「ISO_8859_1」、 「ISO_8859_13」、
「ISO_8859_15」、 「ISO_8859_16」、 「ISO_8859_2」、 「ISO_8859_3」、
「ISO_8859_4」、 「ISO_8859_5」，「ISO_8859_6」、 「ISO_8859_7」、 「ISO_8859_8」、
「ISO_8859_9」，」JIS_X0201" ，「JIS_X0212_1990」、 "KOI8_R" ，"KOI8_U" ，
"SHIFT_JIS" ，"TIS_620" ，"US_ASCII" ，「UTF_16」、 「UTF_16BE」、 「UTF_16LE」、
"UTF_32" ，「UTF_32BE」、 「UTF_32LE」、 "UTF_8" ，「WINDOWS_1250」、
「WINDOWS_1251」、 「WINDOWS_1252」、 「WINDOWS_1253」、」WINDOWS_1254" ，
「WINDOWS_1255」、 「WINDOWS_1256」、 「WINDOWS_1257」、 「WINDOWS_1258」、
「WINDOWS_31J」、 "X_BIG5_HKSCS_2001" ，"X_BIG5_SOLARIS" ，"X_EUCJP_OPEN" ，
"X_EUC_JP_LINUX" ，"X_EUC_TW" ，"X_IBM1006" ，"X_IBM1025" ，"X_IBM1046" ，
"X_IBM1097" ，"X_IBM1098" ，"X_IBM1112" ，"X_IBM1122" ，"X_IBM1123" ，
"X_IBM1124" ，"X_IBM1129" ，"X_IBM1166" ，"X_IBM1364" ，"X_IBM1381" ，"X_IBM1383" ，
"X_IBM29626C" ，"X_IBM300" ，"X_IBM33722" ，"X_IBM737" ，"X_IBM833" ，
"X_IBM834" ，"X_IBM856" ，"X_IBM874" ，"X_IBM875" ，"X_IBM921" ，"X_IBM922" ，
"X_IBM930" ，"X_IBM933" ，"X_IBM935" ，"X_IBM937" ，"X_IBM939" ，"X_IBM942" ，
"X_IBM942C" ，"X_IBM943" ，"X_IBM943C" ，"X_IBM948" ，"X_IBM949" ，"X_IBM949C" ，
"X_IBM950" ，"X_IBM964" ，"X_IBM970" ，"X_ISCII91" ，"X_ISO_2022_CN_CNS" ，
"X_ISO_2022_CN_GB" ，"X_ISO_8859_11" ，"X_JIS0208" ，"X_JISAUTODETECT" ，
"X_JOHAB" ，"X_MACARABIC" ，"X_MACCENTRALEUROPE" ，"X_MACCROATIAN" ，
"X_MACCYRILLIC" ，"X_MACDINGBAT" ，"X_MACGREEK" ，"X_MACHEBREW" ，
"X_MACICE Land" ，"X_MACROMAN" ，"X_MACROMANIA" ，"X_MACSYMBOL" ，
"X_MACTHAI" ，"X_MACTURKISH" ，"X_MACUKRAINE" ，"X_MS932_0213" ，
"X_MS950_HKSCS" ，"X_MS950_HKSCS_XP" ，"X_MSWIN_936" ，"X_PCK" ，
"X_SJIS_0213" ，"X_UTF_16LE_BOM" ，"X_UTF_32BE_BOM" ，"X_UTF_32LE_BOM" ，

"X_WINDOWS_50220" ， "X_WINDOWS_50221" ， "X_WINDOWS_874" ， "X_WINDOWS_949" ， "X_WINDOWS_950" ， "X_WINDOWS_ISO2022j"

AWS Transform 用於現代化大型主機應用程式

AWS Transform for mainframe 可讓您透過自動化分析程式碼庫、規劃轉換、產生技術文件、擷取商業邏輯、分解程式碼、波動規劃，以及執行重構和強化等複雜且耗時的作業，更快速地將舊版 COBOL 大型主機應用程式現代化為 Java 應用程式。它透過利用生成式 AI 和自動化來降低大型主機現代化的成本和複雜性，同時保留您的關鍵任務商業邏輯。 AWS Transform的自然語言界面和目標導向方法可讓您控制轉型，讓您專注於策略優先順序，而自動化則處理現代化旅程的繁重工作。

如需功能和主要功能、高階演練和人類參與輸入和處理的詳細資訊，請參閱AWS Transform 《 使用者指南》中的大型主機應用程式的遷移和現代化。

主要優點

AWS Transform的現代化大型主機應用程式功能有多項優點。其中一些包括：

- 使用生成式 AI 加速大型主機現代化旅程： AWS Transform 協助您在幾個月內將 COBOL 程式碼轉換為現代 Java 程式碼，而不是
  以年為單位的傳統時間表。
- 橋接知識差距：為您的大型主機應用程式 AWS Transform 產生全面的文件、橋接知識差距，並實現更明智的決策。
- 保留關鍵商業邏輯： AWS Transform 會保留舊版系統的關鍵商業邏輯，同時將其重構為現代化的雲端最佳化 Java 應用程式。
- 商業和技術邏輯網域分解： AWS Transform 自動將大型主機程式碼庫分解為不同的商業網域，以減少程式碼庫分析和分解所需的手動工作量和時間。
- Human in the loop (HITL) 功能： AWS Transform 提供自主、目標驅動的方法，讓您掌控大型主機現代化旅程。

大型主機應用程式主控台演練的轉換

在 AWS Transform Web 體驗中，您可以執行大型主機應用程式的轉換，從 COBOL 轉換為 Java。若要了解如何使用此函數，請遵循AWS Transform 《 使用者指南》中大型主機應用程式轉換頁面中的所有步驟。

AWS 使用 Precisely 進行大型主機現代化資料複寫

AWS Mainframe Modernization 提供各種 Amazon Machine Image (AMIs)。這些 AMIs 有助於快速佈建 Amazon EC2 執行個體，為從大型主機系統到 AWS 使用 Precisely 的資料複寫建立量身打造的環境。本指南提供存取和使用這些 AMIs 所需的步驟。

先決條件

- 確定您擁有管理員存取可建立 Amazon EC2 執行個體 AWS 的帳戶。
- 確認 AWS Mainframe Modernization 服務在您計劃建立 Amazon EC2 執行個體的區域中可用。請參閱依區域提供的 AWS 服務清單。
- 識別要建立 Amazon EC2 執行個體的 Amazon Virtual Private Cloud (Amazon VPC)。 Amazon EC2
- 在 Amazon VPC 中建立 Amazon EC2 執行個體時，請確定相關聯的路由表具有網際網路閘道或 NAT 閘道。

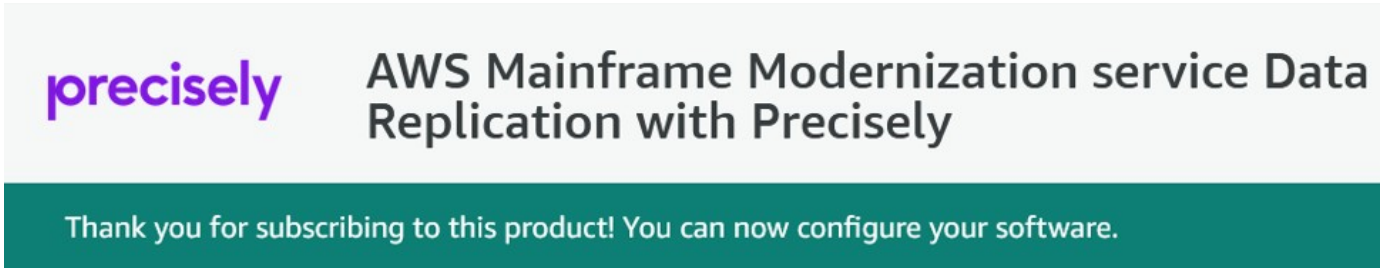
> **Note**
>
> 成功的資料複寫需要 AWS EC2 執行個體具有 AWS Marketplace 的通訊存取權。如果 AWS Marketplace 發生連線問題，複寫程序將會失敗。

訂閱 Amazon Machine Image

當您訂閱 AWS Marketplace 產品時，您可以從產品的 AMI 啟動執行個體。

1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/marketplace 開啟 AWS Marketplace 主控台。
2. 選擇 Manage subscriptions (管理訂閱)。
3. 根據您的使用案例導覽至下列任一連結：
   - IBM z/OS 的資料複寫： https://aws.amazon.com/marketplace/pp/prodview-doe2lroefogia
   - IBM i 的資料複寫： https://aws.amazon.com/marketplace/pp/prodview-iqrkflccxf7ko
4. 選擇 Continue to Subscribe (繼續以訂閱)。
5. 如果條款與條件是可接受的，請選擇接受條款。訂閱可能需要幾分鐘的時間來處理。

6. 等待顯示感謝訊息，如下所示。此訊息確認您已成功訂閱產品。



7. 在左側導覽窗格中，選擇管理訂閱。此檢視會顯示您已訂閱的所有訂閱。

使用 Precisely 啟動 AWS 大型主機現代化資料複寫

1. 開啟 AWS Marketplace 主控台，網址為 https：//https://console.aws.amazon.com/marketplace。
2. 在左側導覽窗格中，選擇管理訂閱。
3. 尋找您要啟動的 AMI，然後選擇啟動新執行個體。
4. 在區域下，選取允許清單的區域。
5. 選擇繼續以透過 EC2 啟動。此動作會帶您前往 Amazon EC2 主控台。
6. 輸入伺服器的名稱。
7. 選取符合您專案效能和成本需求的執行個體類型。執行個體大小的建議起點為 `c5.2xLarge`。
8. 選擇現有的金鑰對，或建立並儲存新的金鑰對。如需有關金鑰對的資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon EC2 金鑰對和 Linux 執行個體。 Amazon EC2
9. 編輯網路設定，然後選擇允許列出的 VPC 和適當的子網路。
10. 選擇現有的安全群組或建立新的安全群組。除了允許 SSH 存取（預設在連接埠 22 上）之外，對於具有精確伺服器 EC2 執行個體的資料複寫，通常允許 TCP 流量傳送到其預設連接埠 2626。
11. 設定 Amazon EC2 執行個體的儲存體。
12. 檢閱摘要，然後選擇啟動執行個體。若要成功啟動，執行個體類型必須是有效的。如果啟動失敗，請選擇編輯執行個體組態，然後選擇不同的執行個體類型。
13. 看到成功訊息後，請選擇連線至執行個體。
14. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
15. 在左側導覽窗格中，於執行個體功能表下，選擇執行個體。
16. 在主窗格中，檢查執行個體的狀態。

建立 IAM 政策

若要成功操作透過 AWS Marketplace 清單部署 AWS 的 Mainframe Modernization EC2 執行個體，您必須設定 IAM 角色和政策。此特別自訂的 IAM 設定不是選用的；它授權您的 Amazon EC2 執行個體與 AWS Marketplace 服務互動。IAM 角色和政策允許 AWS Mainframe Modernization 準確記錄用量資料，這對於精確計費至關重要。未實作此組態可能會導致資料複寫嘗試失敗和操作中斷。

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側的導覽窗格中，選擇 Policies (政策)。
3. 如果這是您第一次選擇政策，則會顯示歡迎使用受管政策頁面。選擇開始使用。
4. 在頁面頂端，選擇 Create policy (建立政策)。
5. 在政策編輯器中，選擇 JSON 選項。
6. 輸入下列 JSON 政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": ["aws-marketplace:MeterUsage"],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

建立 IAM 角色

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，然後選擇建立角色。
3. 在 Trusted entity type (信任的實體類型) 區段中，選擇 AWS service (服務)。
4. 在使用案例區段中的服務或使用案例下，選擇 Amazon EC2。
5. 選擇下一步。
6. 在政策清單中，從依類型篩選下拉式清單中選取客戶受管，然後輸入您建立的政策名稱。選取政策名稱旁的核取方塊。
7. 選擇下一步。

8. 輸入名稱，並選擇性地輸入角色的描述。
9. 檢閱信任政策和許可，然後選擇建立角色。

將 IAM 角色連接至 Amazon EC2 執行個體

1. 前往 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. 選取您的 Amazon EC2 執行個體。
4. 從動作功能表中，選擇安全性，然後選擇修改 IAM 角色。
5. 選取要連接至執行個體的角色，然後選擇更新 IAM 角色。

如需 IBM i AWS 的資料複寫入門的詳細資訊，請參閱資料複寫概觀。

AWS Mainframe Modernization 使用 mLogica 進行程式碼轉換

AWS Mainframe Modernization 使用 mLogica 的程式碼轉換 （程式碼轉換） 是一種 AWS Mainframe Modernization 功能，可自動將z/OS大型主機組件程式碼轉換為 COBOL。您可以使用程式碼轉換來使用 AWS CodeBuild 服務提取組合器映像，以便透過 進行預期的程式碼轉換 AWS 帳戶。

主題

- 什麼是 mLogica 的 Assembler Conversion？
- 了解 Assembler 轉換的程式碼轉換計費
- 程式碼轉換概念
- 了解程式碼轉換的元件和程序
- 教學課程：在 中將程式碼從 Assembler 轉換為 COBOL AWS Mainframe Modernization

什麼是 mLogica 的 Assembler Conversion？

AWS Mainframe Modernization 使用 mLogica 的程式碼轉換 （程式碼轉換） 會自動將z/OS大型主機組件程式碼轉換為 COBOL。服務會在您的 中執行 AWS 帳戶 ，不會在 外部傳輸或存放 Assembler 或 COBOL 原始碼 AWS 帳戶。程式碼轉換可讓您的授權帳戶使用 AWS CodeBuild 服務提取組合器映像，以進行預期的程式碼轉換。

AWS Mainframe Modernization 可讓您為遷移的應用程式設定建置和持續整合/持續交付 (CI/CD) 管道。這些建置和管道使用 AWS CodeBuild 和 Amazon S3 來提供此功能。 AWS CodeBuild 是一種全受管的建置服務，可編譯您的原始程式碼、執行單元測試，並產生準備好部署的成品。Amazon S3 是一種物件儲存服務，可提供業界領先的可擴展性、資料可用性、安全性和效能。

主題

- 程式碼轉換編譯器
- 程式碼轉換架構
- 自動化方法
- 安全
- 其他資源

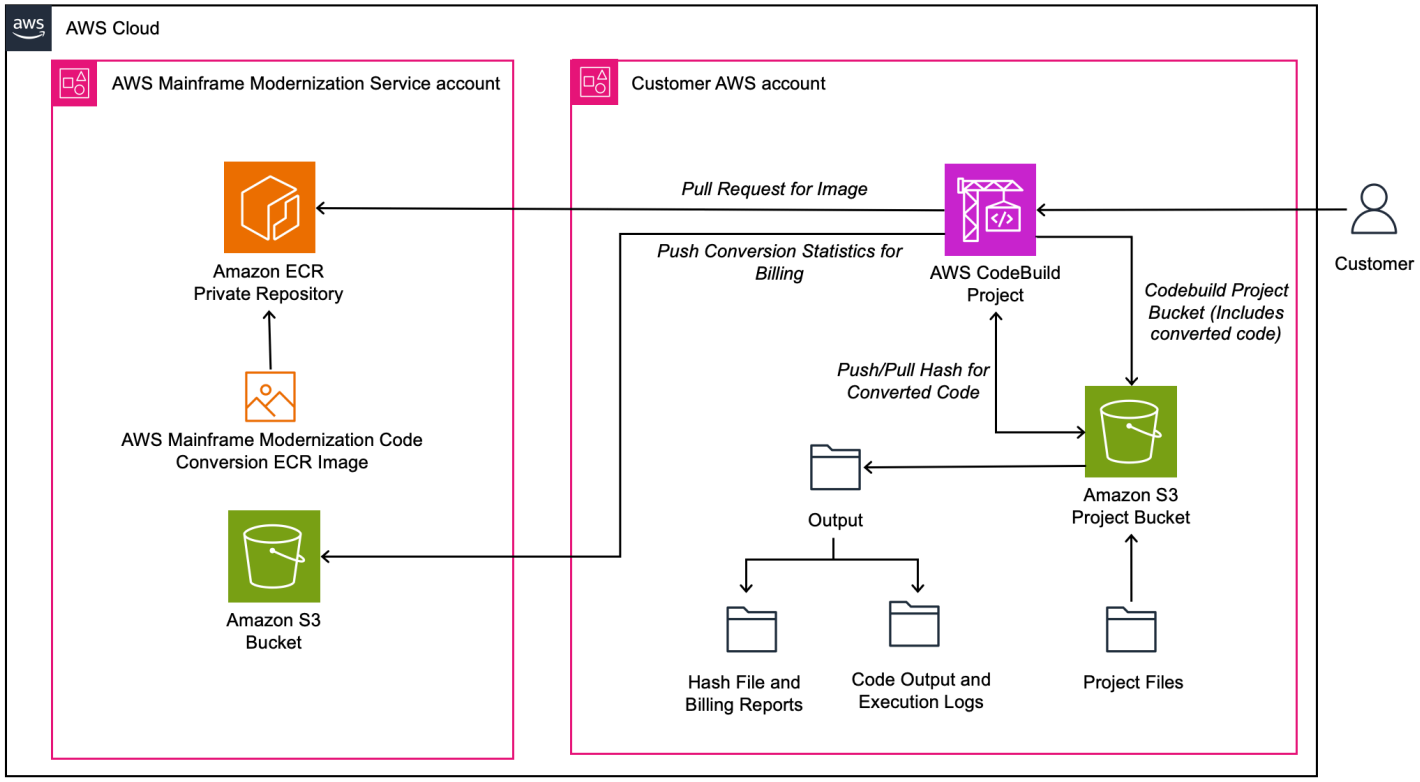
程式碼轉換編譯器

程式碼轉換可以設定為發出適用於編譯的 COBOL，並在具有不同編譯器的多個目標環境中執行。其中一些包括：

- 使用 Rocket Software （ 先前稱為 Micro Focus) 和其他 Rocket Enterprise Server 環境進行 M2 重建
- 使用 NTT DATA Enterprise COBOL (UniKix) 進行 M2 重建
- mLogica LIBER*COBOL
- z/OS 使用 IBM Enterprise COBOL 的大型主機
- Veryant isCOBOL

程式碼轉換架構

以下是程式碼轉換程序的架構圖：



自動化方法

若要搭配 CodeBuild 使用程式碼轉換，需要將 Assembler 程式碼上傳到 Amazon S3 儲存貯體，以便稍後設定轉換參數並叫用 CodeBuild 專案，以執行轉換程序中的每個步驟。目標 COBOL 程式碼會自動儲存在 Amazon S3 儲存貯體的指定路徑中。

安全

AWS Mainframe Modernization 程式碼轉換可啟用轉換，同時將所有來源和目標程式碼保留在您的 中 AWS 帳戶。來源組件程式碼、目標 COBOL 程式碼和組態檔案會存放在 Amazon S3 儲存貯體中。自動化轉換工具會在您 的 CodeBuild 環境中以容器形式執行 AWS 帳戶。程式碼會隨時保留在您的 帳戶中。

若要讓 Conversion 工具存取 Amazon S3 儲存貯體，請將儲存貯體的許可授予 AWS 服務 角色。當您設定 CodeBuild 時，您將設定此服務角色，以便 CodeBuild 可以存取容器映像並存取您的 Amazon S3 儲存貯體。

其他資源

除了 之外the section called “教學課程：將程式碼從 Assembler 轉換為 COBOL”，以下是一些其他資源，您可以了解如何建立 AWS CloudFormation 範本，以及有關將 Assembler 轉換為 COBOL 的其他資訊。

- 從 Assembler 到 COBOL 的自動化程式碼轉換研討會連結：https://catalog.workshops.aws/awsm2ccm-assembler-cobol/en-US。
- 部落格文章：https://aws.amazon.com/blogs/migration-and-modernization/unlocking-new-potential-transform-your-assembler-programs-to-cobol-with-aws-mainframe-modernization/。

了解 Assembler 轉換的程式碼轉換計費

在實際轉換之前，您將參考此頁面以了解程式碼轉換帳單範圍和程序。帳單計算區段說明從 Assembler 轉換為 COBOL 的程序，會依每一行程式碼收費。

程式碼轉換帳單和範圍

組裝器程式碼轉換只會在完成轉換步驟後，在您的 AWS 帳戶 上產生費用 （帳單報告）。費用是根據轉換的程式碼行數而定。如果您執行多個轉換步驟，例如在新增組件程式碼、變更轉換組態或套用新版

本的容器之後，只會使用變更的行和/或新增的行來計算費用。對於在相同程式中轉換同一行程式碼，我們不會向您收取兩次費用。

> **Note**
>
> 變更程式碼行和新程式或重新命名程式中所有程式碼行的模組都會收費。

為了避免多次收費，程式碼轉換會將每個 Assembler 或 Macro 模組的編碼二進位檔案存放在 的專案儲存貯體中<Project_bucket>/*awsm2ccm-do-not-delete*/<AWS_account_number>/Hash。這些編碼檔案不包含任何客戶代碼。

> **Important**
>
> 請勿手動編輯或刪除這些檔案。變更可能會導致轉換相同元件的多個帳單。

AWS Mainframe Modernization 程式碼轉換分析報告（「分析報告」）可為客戶提供預期轉換範圍、結果和帳單的詳細資訊，以確保實際轉換的準確預期。轉換可能會導致某些程式碼行無法轉換、某些程式碼行部分轉換，以及某些程式碼行完全轉換。分析報告會顯示每個類別的程式碼行數。在處理程式、巨集和複製手冊的任何轉換之前，您必須先執行和閱讀分析報告。一旦客戶檢閱分析報告，並且同意報告的範圍、預期成果和預期帳單，客戶就可以繼續執行轉換。

> **Note**
>
> 透過執行 AWS Mainframe Modernization Code Conversion **`Convert`**命令，您確認您已執行並閱讀分析報告，並且同意預期結果和可計費的程式碼行數。

轉換範圍

AWS Mainframe Modernization 程式碼轉換會處理設定之 S3 來源位置中 scrlib 和 macrolib 目錄中所有可用組合器、巨集和複製手冊元件的所有程式碼行。組合器程式中參考的組合器程式、任何巨集和複製手冊都在範圍內。組合器程式未參考的巨集和複製手冊元件會被視為超出範圍且未轉換。在處理期間，轉換器會執行進階演算法，以全面考量每個範圍內元件。這些元件的所有程式碼行都會參與處理，無論它們是否完全轉換、部分轉換或未轉換。 AWS Mainframe Modernization 程式碼轉換會忽略空白行，且不將其計為程式碼行。包含任何其他文字的註解行和行 （ 例如，內嵌在 JCL 中的組合器的 JCL 陳述式 ） 會計為用於計費的程式碼行。

帳單計算

AWS Mainframe Modernization 整個範圍內元件的程式碼轉換費用。這表示它會針對每個範圍內元件內的每行程式碼收費，包括無法轉換、部分轉換和完全轉換的行。 AWS Mainframe Modernization 程式碼轉換會加總提供用於處理的所有元件行（包括組合程式、參考的複本手冊和參考的巨集），並使用程式碼行總數計費。

> **Note**
>
> 未由 Assembler 程式參考的複本和巨集不會被視為範圍內。

例如，假設程式有 1，000 行程式碼：

- 700 行已完全轉換
- 200 行部分轉換
- 100 行未轉換

將處理 1，000 行程式碼，且應計費。

改善轉換

身為客戶，如果您想要提高程式碼行的轉換率，或有其他特定需求，您可以聯絡 AWS 代表以取得其他參與選項，例如校正工作或專業服務協助。

程式碼轉換概念

若要了解程式碼轉換的方式，了解一些重要概念，例如巨集處理、程式碼頁面和 CodeBuild。

主題

- 巨集處理
- 程式碼頁面 (EBCDIC 與 ASCII)
- CodeBuild

巨集處理

Mainframe Assembler 程式碼經常使用巨集封裝功能以重複使用。巨集行為通常根據從 Assembler 程式傳遞的參數，在應用程式執行時間決定。程式碼轉換提供數種機制，可在轉換為 COBOL 之前擴展 Assembler Macros。

程式碼頁面 (EBCDIC 與 ASCII)

Mainframe Assembler 通常包含以十六進位值表示的字元常值，對應於 EBCDIC 字元。程式碼轉換提供可設定的功能，可在為 ASCII 環境發出 COBOL 時，自動管理 ASCII 中的角色常值。

CodeBuild

程式碼轉換可透過 AWS CodeBuild 服務取得。 AWS CodeBuild 是最初設計為 CI/CD 管道一部分的建置自動化工具。在 中 AWS Mainframe Modernization， AWS CodeBuild 用於自動化 MCCAC 轉換工具和其他工具，例如 Rocket Software （ 先前稱為 Micro Focus) COBOL 編譯器。

了解程式碼轉換的元件和程序

AWS Mainframe Modernization 程式碼轉換程序包括各種元件，例如 AWS Mainframe Modernization 容器、S3 專案儲存貯體和日誌檔案位置。

主題

- AWS Mainframe Modernization 容器
- S3 專案儲存貯體
- 日誌檔案位置
- 程序概觀

AWS Mainframe Modernization 容器

AWS Mainframe Modernization 程式碼轉換容器會在 AWS CodeBuild 專案中執行，並提供命令來設定專案目錄和組態檔案、評估 Assembler 程式碼、展開 Assembler 巨集，以及將 Assembler 程式碼轉換為 COBOL。

您可以存取下列 AWS ECR 儲存庫：`381492161314.dkr.ecr.us-east-1.amazonaws.com/aws-mlogica-codebuild-prod`。

若要使用映像，您可以遵循下列兩個選項之一：

- 透過 使用映像時，請使用最新的標籤 AWS CodeBuild。使用映像時，您將使用此路徑：381492161314.dkr.ecr.us-east-1.amazonaws.com/aws-mlogica-codebuild-prod。這表示 AWS CodeBuild 將挑選最後一個推送到儲存庫的映像。
- 列出版本並從中選取。若要這樣做，請透過 CLI 使用以下命令列出儲存庫中的不同版本：

```
aws ecr describe-images \
  --registry-id 381492161314 \
  --repository-name aws-mlogica-codebuild-prod \
  --query 'imageDetails[*].{ImagePushedAt: imagePushedAt, ImageTags: imageTags}' \
  --output json | jq '[.[] | {ImageURI: (.ImageTags[] |
 "381492161314.dkr.ecr.us-east-1.amazonaws.com/aws-mlogica-codebuild-prod:" + .),
 ImagePushedAt: .ImagePushedAt}] | sort_by(.ImagePushedAt) | reverse'
```

這將列出每個映像上具有關聯標籤的所有映像，以及特定映像發佈到儲存庫的時間。根據上述程式碼，您會取得影像清單，其中影像上的標籤代表程式碼轉換公用程式的版本。您可以根據您的需求選取適當的映像。

S3 專案儲存貯體

輸入和輸出程式碼、以擴充巨集更新的程式碼，以及 AWS Mainframe Modernization 程式碼轉換產生的報告，都會存放在您在 中建立的專案儲存貯體中 AWS 帳戶管理。您可以透過授予 AWS 服務角色許可，為 AWS Mainframe Modernization 程式碼轉換提供儲存貯體的存取權。

日誌檔案位置

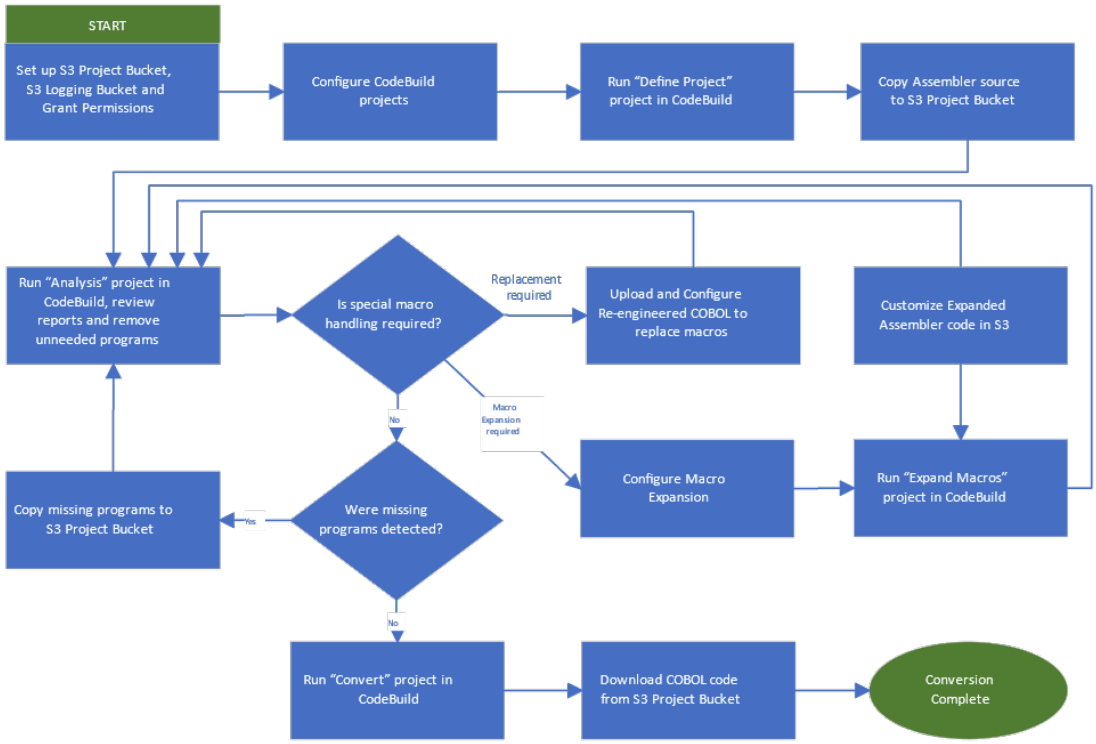
日誌檔案會在每個 CodeBuild 專案執行期間寫入兩個位置：

- 具有每個 CodeBuild 步驟高階結果的日誌檔案會寫入 CodeBuild 中設定的日誌儲存貯體中的日誌檔案。這些檔案會以 gzip 封存形式顯示，其中包含 CodeBuild 架構產生的 GUID 類型檔案名稱 （ 例如 0c03e183-ab40-4fe0-ba77- bc1d87e73b14.gz)。每個封存都包含執行 CodeBuild 專案所產生的日誌。如果 CodeBuild 專案執行失敗，此日誌檔案將包含重要的故障診斷資訊。
- 在元件層級具有詳細執行結果的日誌檔案會寫入具有檔案名稱模式 <Project_Bucket_name>_.log (例如 project- bucket_202406131200.log) 的主要專案儲存貯體路徑中的日誌檔案。這些日誌提供：
  - 記錄輸入和輸出位置的組態摘要。
  - 使用目標檔案名稱處理的每個 Assembler 或 Macro 元件的日誌。
  - 使用檔案位置產生的報告清單。

- 對於轉換執行，提供執行時間複製手冊的清單。

程序概觀

下圖說明將 Assembler 轉換為 COBOL 的程序：



教學課程：在 中將程式碼從 Assembler 轉換為 COBOL AWS Mainframe Modernization

您可以使用本文件做為step-by-step指南，了解如何將大型主機現代化組件程式碼轉換為 COBOL。此外，您也可以參考從 Assembler 到 COBOL 的自動化程式碼轉換研討會，以進一步了解轉換程序。

主題

- 先決條件
- 步驟 1：與 共用建置資產 AWS 帳戶
- 步驟 2：建立 Amazon S3 儲存貯體
- 步驟 3：建立 IAM 政策
- 步驟 4：建立 IAM 角色

- 步驟 5：將 IAM 政策連接至 IAM 角色
- 步驟 6：建立 CodeBuild 專案
  - 步驟 6.1：建立定義專案
  - 步驟 6.2：建立程式碼分析專案
  - 步驟 6.3：建立程式碼轉換專案
- 步驟 7：定義專案並上傳原始碼
- 步驟 8：執行分析並了解報告
- 步驟 9：執行程式碼轉換
- 步驟 10：驗證程式碼轉換
- 步驟 11：下載轉換後的程式碼
- 清除資源

先決條件

請閱讀 了解 Assembler 轉換的程式碼轉換計費一節，了解 Assembler 程式碼轉換如何產生費用（帳單報告）AWS 帳戶管理，以及帳單的運作方式。

步驟 1：與 共用建置資產 AWS 帳戶

在此步驟中，請確定您與 共用建置資產 AWS 帳戶，尤其是在使用資產的區域中。

1. 在 https：//https://console.aws.amazon.com/m2/ 開啟 AWS Mainframe Modernization 主控台。
2. 在左側導覽中，選擇工具。
3. 在 AWS Mainframe Modernization Code Conversion with mLogica 中，選擇與我的共用資產 AWS 帳戶。

> **Important**
>
> 您需要在每個要執行建置的 AWS 區域中執行此步驟一次。

步驟 2：建立 Amazon S3 儲存貯體

在此步驟中，您會建立 Amazon S3 儲存貯體。第一個儲存貯體是 的專案儲存貯體 AWS CodeBuild ，用於保留原始程式碼，然後推送輸出儲存貯體以保留輸出 AWS CodeBuild （轉換程式碼）。如需詳細資訊，請參閱《Amazon S3 使用者指南》中的建立、設定和使用 Amazon S3 儲存貯體。 Amazon S3

1. 若要建立專案儲存貯體，請登入 Amazon S3 主控台，然後選擇建立儲存貯體。
2. 在一般組態中，提供儲存貯體的名稱，並指定您要建立儲存貯 AWS 區域 體的 。範例名稱為 *codebuild-regionId-accountId-bucket*，其中：
   - `regionId` 是 儲存貯 AWS 區域 體的 。
   - `accountId` 是您的 AWS 帳戶 ID。

   > **Note**
   > 如果您要在與美國東部（維吉尼亞北部） AWS 區域 不同的 中建立儲存貯體，請指定 `LocationConstraint` 參數。如需詳細資訊，請參閱《Amazon Simple Storage Service API 參考》中的建立儲存貯體。

3. 保留所有其他設定，然後選擇建立儲存貯體。

無論您為這些儲存貯體選擇的名稱為何，請務必在本教學課程中使用這些儲存貯體。

步驟 3：建立 IAM 政策

在此步驟中，您會建立 IAM 政策。提供的 IAM 政策授予與 Amazon S3、Amazon Elastic Container Registry、CodeBuild 產生的 Amazon CloudWatch logs以及程式碼轉換 Amazon Elastic Compute Cloud 資源 AWS CodeBuild 互動的特定許可。此政策不會為客戶自訂。政策會授予 AWS Mainframe Modernization 互動的許可，並擷取程式碼轉換統計資料，以適當地向客戶收費。

若要了解如何建立 IAM 政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

建立政策

1. 登入 IAM 主控台，然後在左側導覽窗格中選擇政策。
2. 選擇建立政策。
3. 將下列 JSON 政策複製並貼到政策編輯器中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:PutObjectAcl",
                "s3:GetBucketAcl"
            ],
            "Resource": [
                "arn:aws:s3:::codebuild-regionId-accountId-bucket",
                "arn:aws:s3:::codebuild-regionId-accountId-bucket/*",
                "arn:aws:s3:::aws-m2-repo-*" ],
                    "Effect": "Allow"
          },
          {
              "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "logs:*",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DeleteNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeVpcs",
                "ec2:CreateNetworkInterfacePermission"
            ],
            "Resource": "*",
            "Effect": "Allow"
         }
    ]
}
```

4. 您可以選擇將標籤新增至政策。標籤是索引鍵/值對，可協助您組織、追蹤或控制政策的存取。
5. 選擇 Next:Review (下一步：檢閱)。

6. 提供政策的名稱，例如 *CodeBuildAWSM2CCMPolicy*。
7. 您可以選擇性地輸入政策的描述，並檢閱政策摘要以確保其正確。
8. 選擇建立政策。

步驟 4：建立 IAM 角色

在此步驟中，您會建立新的 IAM 角色，在您將先前建立的 IAM 政策與此新 IAM 角色建立關聯之後，允許 CodeBuild 與您 AWS 資源互動。

如需建立服務角色的相關資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可給 AWS 服務。

1. 登入 IAM 主控台，然後在左側導覽窗格中選擇角色。
2. 選擇建立角色。
3. 在信任的實體類型下，選擇 AWS 服務。
4. 在其他 AWS 服務的使用案例下，選擇 CodeBuild，然後再次選擇 CodeBuild。
5. 選擇下一步。
6. 在 Add permissions (新增許可) 頁面上，選擇 Next (下一步)。您稍後會將政策指派給角色。
7. 在角色詳細資訊下，提供角色的名稱，例如 `IAMRoleTaskExecutionRoleForCodeBuild`。
8. 在選取信任的實體下，確認政策文件如下所示：

```
{
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": "codebuild.amazonaws.com"
              },
              "Action": "sts:AssumeRole"
            }
          ]
}
```

9. 選擇建立角色。

步驟 5：將 IAM 政策連接至 IAM 角色

在此步驟中，您將先前建立的 IAM 政策連接至 IAM `IAMRoleTaskExecutionRoleForCodeBuild` 角色。

1. 登入 IAM 主控台，然後在左側導覽窗格中選擇角色。
2. 在角色中，選擇您先前建立的角色，例如 `IAMRoleTaskExecutionRoleForCodeBuild`。
3. 在許可政策中，選擇新增許可，然後選擇連接政策。
4. 在其他許可政策中，選擇您先前建立的政策，例如 *`CodeBuildAWSM2CCMPolicy`*。
5. 選擇連接政策。

步驟 6：建立 CodeBuild 專案

在此步驟中，您會根據上述`buildspec.yml`檔案建立三個不同的 CodeBuild 專案。

步驟 6.1：建立定義專案

建立定義專案

1. 登入 CodeBuild 主控台，然後選擇建立建置專案。
2. 在專案組態區段中，提供專案的名稱，例如 `1-awsm2ccm-define-project`。
3. 在來源區段中，對於來源提供者，保留預設選擇。
4. 在環境區段中，選擇自訂映像。
5. 在環境類型欄位中，選擇 Linux。
6. 在映像登錄檔下，選擇其他登錄檔。
7. 在外部登錄 URL 欄位中，遵循 the section called “AWS Mainframe Modernization 容器”區段。
8. 在服務角色下，選擇現有服務角色，然後在角色 ARN 欄位中，選擇您先前建立的服務角色（例如 `IAMRoleTaskExecutionRoleForCodeBuild`)。
9. 展開其他組態區段，執行下列動作：

    a. VPC：視需要根據您的設定進行設定。

    b. 逾時：設定為 60 分鐘。

    c. 佇列逾時：設定為 480 分鐘。

    d. 加密：選擇適當的加密設定（預設值沒問題）。

    e. 在環境變數區段中，逐一新增下列項目：

- 名稱：PROJECT_BUCKET。值 ：**codebuild-regionId-accountId- bucket**。類型 ：純文字
- 名稱：PROJECT_DIR。值 ：**prj_codebuild_01**。類型 ：純文字
- 名稱：AWSM2CCM_ACTION。值 ：**define_project**。類型 ：純文字
- 名稱：AWSM2CCM_LOGGING_BUCKET。值 ：**s3:// codebuild-regionId-accountId-bucket**。類型 ：純文字

10. 在 Buildspec 區段中，選擇插入建置命令，然後切換到編輯器。
11. 將目前的值取代為：

```
version: 0.2
phases:
    build:
        commands:
            - . /app/awsm2ccm_prod/bin/setup_env.sh
            - run_awsm2ccm.sh $PROJECT_DIR
artifacts:
    files:
        - '**/*'
    discard-paths: no
    base-directory: $PROJECT_DIR
```

其中，PROJECT_DIR 是 CodeBuild 中可用的環境變數。如需詳細資訊，請參閱建置環境中的環境變數。

12. 在成品區段中，執行下列動作：
    - 在類型下，選擇 Amazon S3，然後選擇輸出儲存貯體，例如 codebuild-regionId-accountId-bucket。
    - 對於路徑，請將此欄位保留空白。
    - 針對名稱，輸入 **prj_codebuild_01**。
    - 針對成品封裝，選取無。
    - 對於覆寫成品名稱，請取消勾選此選項。
    - 對於加密，將其保留為預設設定。
13. 針對日誌區段，執行下列動作：
    - CloudWatch 日誌：已停用

- S3 日誌：已啟用

- 儲存貯體： **codebuild-regionId-account-bucket**
- 日誌路徑： **CODEBUILD-LOGS**

14. 選擇 Create build project (建立建置專案)。

步驟 6.2：建立程式碼分析專案

建立程式碼分析專案

1. 登入 CodeBuild 主控台，然後選擇建立建置專案。
2. 在專案組態區段中，提供專案的名稱，例如 `2-awsm2ccm-analysis`。
3. 在來源區段中，針對來源提供者選擇 Amazon S3，然後選擇您先前建立的輸入儲存貯體 （ 例如 `codebuild-regionId-accountId-bucket`)。
4. 在 S3 物件金鑰或 S3 資料夾欄位中，輸入 **prj_codebuild_01**。
5. 在環境區段中，選擇自訂映像。
6. 在環境類型欄位中，選擇 Linux。
7. 在映像登錄檔下，選擇其他登錄檔。
8. 在外部登錄 URL 欄位中，遵循 the section called “AWS Mainframe Modernization 容器”區段。
9. 在服務角色下，選擇現有服務角色，然後在角色 ARN 欄位中，選擇您先前建立的服務角色 （ 例如，`IAMRoleTaskExecutionRoleForCodeBuild`)。
10. 展開其他組態區段，執行下列動作：

    a. VPC：視需要根據您的設定進行設定。

    b. 逾時：設定為 60 分鐘。

    c. 佇列逾時：設定為 480 分鐘。

    d. 加密：選擇適當的加密設定 （ 預設值沒問題 ）。

    e. 在環境變數區段中，逐一新增下列項目：

    - 名稱：PROJECT_BUCKET。值 ：**codebuild-regionId-accountId-bucket**。類型 ：純文字
    - 名稱：PROJECT_DIR。值 ：**prj_codebuild_01**。類型 ：純文字
    - 名稱：AWSM2CCM_ACTION。值 ：**analysis**。類型 ：純文字

- 名稱：AWSM2CCM_LOGGING_BUCKET。值 ：**s3:// codebuild-regionId-accountId-bucket**。類型 ：純文字

11. 在 Buildspec 區段中，選擇插入建置命令，然後切換到編輯器。
12. 將目前的值取代為：

```
version: 0.2
phases:
    build:
        commands:
            - ln -s $CODEBUILD_SRC_DIR $PROJECT_DIR
            - . /app/awsm2ccm_prod/bin/setup_env.sh
            - run_awsm2ccm.sh $PROJECT_DIR
artifacts:
    files:
        - '*.log'
        - '_Converted/*/*'
        - '_Reports/*'
    secondary-artifacts:
        reports:
            files:
                - '_Reports/AWSM2CCM*'
    discard-paths: no
    base-directory: $PROJECT_DIR
```

其中，PROJECT_DIR 是 CodeBuild 中可用的環境變數。如需詳細資訊，請參閱建置環境中的環境變數。

13. 在成品區段中，執行下列動作：
    - 在類型下，選擇 Amazon S3，然後選擇輸出儲存貯體 （例如 codebuild-regionId-accountId-bucket)。
    - 針對路徑，輸入 ARTIFACTS。
    - 針對名稱，輸入 **prj_codebuild_01**。
    - 針對成品封裝，選取無。
    - 對於覆寫成品名稱，請取消勾選此選項。
    - 對於加密，請將其保留為預設設定。
14. 針對日誌區段，執行下列動作：
    - CloudWatch 日誌：已停用

- S3 日誌：已啟用

- 儲存貯體： **codebuild-regionId-account-bucket**
- 日誌路徑： **CODEBUILD-LOGS**

15. 選擇 Create build project (建立建置專案)。

步驟 6.3：建立程式碼轉換專案

建立程式碼轉換專案

1. 登入 CodeBuild 主控台，然後選擇建立建置專案。
2. 在專案組態區段中，提供專案的名稱 （ 例如`3-awsm2ccm-convert`)。
3. 在來源區段中，針對來源提供者選擇 Amazon S3，然後選擇您先前建立的輸入儲存貯體 （ 例如，`codebuild-regionId-accountId-bucket`)。
4. 在 S3 物件金鑰或 S3 資料夾欄位中，輸入 **prj_codebuild_01**。
5. 在環境區段中，選擇自訂映像。
6. 在環境類型欄位中，選擇 Linux。
7. 在映像登錄檔下，選擇其他登錄檔。
8. 在外部登錄 URL 欄位中，遵循 the section called “AWS Mainframe Modernization 容器”區段。
9. 在服務角色下，選擇現有服務角色，然後在角色 ARN 欄位中，選擇您先前建立的服務角色；例如，`IAMRoleTaskExecutionRoleForCodeBuild`。
10. 展開其他組態區段，執行下列動作：

    a. VPC：視需要根據您的設定進行設定。

    b. 逾時：設定為 60 分鐘。

    c. 佇列逾時：設定為 480 分鐘。

    d. 加密：選擇適當的加密設定 （ 預設值沒問題 ）。

    e. 在環境變數區段中，逐一新增下列項目：

    - 名稱：PROJECT_BUCKET。值 ：**codebuild-regionId-accountId-bucket**。類型 ：純文字
    - 名稱：PROJECT_DIR。值 ：**prj_codebuild_01**。類型 ：純文字
    - 名稱 ：AWSM2CCM_ACTION。值 ：**conversion**。類型 ：純文字

- 名稱：AWSM2CCM_LOGGING_BUCKET。值 ：**s3:// codebuild-regionId-accountId-bucket**。類型 ：純文字

11. 在 Buildspec 區段中，選擇插入建置命令，然後切換到編輯器。
12. 將目前的值取代為：

```
version: 0.2
phases:
    build:
        commands:
            - export AWSM2CCM_PUSH_RUNTIME_COPYBOOKS=y
            - ln -s $CODEBUILD_SRC_DIR $PROJECT_DIR
            - . /app/awsm2ccm_prod/bin/setup_env.sh
            - run_awsm2ccm.sh $PROJECT_DIR
artifacts:
    files:
        - '*.log'
        - '_Converted/*/*'
        - '_Reports/*'
    discard-paths: no
    base-directory: $PROJECT_DIR
```

其中 PROJECT_DIR 是 CodeBuild 中可用的環境變數。如需詳細資訊，請參閱建置環境中的環境變數。

13. 在成品區段中，執行下列動作：
    - 在類型下，選擇 Amazon S3，然後選擇輸出儲存貯體 （例如 codebuild-regionId-accountId-bucket)。
    - 針對路徑，輸入 ARTIFACTS。
    - 針對名稱，輸入 **prj_codebuild_01**。
    - 針對成品封裝，選取無。
    - 針對覆寫成品名稱，請取消勾選此選項。
    - 對於加密，將其保留為預設設定。
14. 針對日誌區段，執行下列動作：
    - CloudWatch 日誌：已停用
    - S3 日誌：已啟用

- 儲存貯體： **codebuild-regionId-account-bucket**
- 日誌路徑： **CODEBUILD-LOGS**

15. 選擇 Create build project (建立建置專案)。

步驟 7：定義專案並上傳原始碼

定義專案會設定專案資料夾和組態檔案，並以預設組態初始化。在此步驟中，您會啟動建置。若要執行此作業：

1. 登入 AWS CodeBuild 主控台。
2. 在左側導覽窗格中，選擇建置專案。
3. 選取要建置的先前建立的專案 (`1-awsm2ccm-define-project`)
4. 選擇開始建置，然後立即開始定義專案。一旦建置開始，狀態會變更為進行中。
5. 選擇階段詳細資訊，以查看專案協調的每個步驟的進度 AWS CodeBuild 。
6. 等到所有步驟的狀態變更為成功。
7. 前往 Amazon S3 主控台。
8. 找到並按一下名為 的 Amazon S3 儲存貯體 `codebuild-regionId-accountId-bucket`

   - **CODEBUILD-LOGS/** 資料夾包含執行 AWS CodeBuild 中專案的 AWS CodeBuild 日誌。
   - **prj_codebuild_01/** 包含專案結構的 資料夾。它用於分析、explication_macros 和轉換步驟。您可以選擇 `prj_codebuild_01/`來探索詳細資訊
   - **cobol_reserved.rsw** 為轉換器保留的組態檔案 (COBOL 單字清單）。其會在轉換步驟期間使用。
   - **Macro_Expansion/** 資料夾包含巨集以擴展至 Assembler 程式。它在 expand_macros 步驟期間使用。
   - **macro_settings.json** 組態檔案包含自訂巨集替換。它在 expand_macros 步驟期間使用。
   - **macrolib/** 資料夾包含要轉換的 Assembler 巨集。它會在分析和轉換步驟期間使用。

     1. 選取 `macrolib/`。
     2. 根據預設，一個名為 的 Assembler 巨集`MACR01.mac`會以範例檔案的形式提供。刪除此檔案，因為分析不需要。
     3. 在此目錄中上傳您的巨集。

- **project_settings_aux.json** 組態檔案包含與程式碼頁面相關的設定。其會在轉換步驟期間使用。
- **project_settings.json** 組態檔案包含轉換器的設定。其會在轉換步驟期間使用。
- **srclib/** 資料夾包含要轉換的 Assembler 程式。它會在分析和轉換步驟期間使用。
  1. 選擇 srclib/。
  2. 根據預設，兩個名為 SQtest01.asm和 的 Assembler 程式SQtest02.asm會以範例形式提供。刪除這些檔案，因為分析和轉換不需要這些檔案。
  3. 在此目錄中上傳您的 Assembler 程式。

9. 驗證1-awsm2ccm-define-project步驟的狀態。它應該在最新建置狀態索引標籤下成功。

您已準備好進行下一個步驟：程式碼分析。

步驟 8：執行分析並了解報告

> **Note**
>
> AWS Mainframe Modernization 程式碼轉換分析步驟是免費的。

在此步驟中，您會啟動另一個組建：

1. 在左側導覽窗格中，選擇建置專案。
2. 選擇您在步驟 6.2 中建立的專案來建置：。 2-awsm2ccm-analysis
3. 選擇開始建置，然後立即開始產生分析報告。這將啟動建置並將狀態變更為進行中。
4. 選擇階段詳細資訊，您會在其中看到專案協調的每個步驟的進度 AWS CodeBuild 。等到狀態在所有步驟中變更為成功。
5. 從 AWS Management Console前往 Amazon S3 服務主控台。
6. 找到並按一下 Amazon S3 儲存貯體： codebuild-regionId-accountId-bucket

   a. **ARTIFACTS/** 資料夾包含分析和轉換步驟的輸出。

   b. 選擇 ARTIFACTS/prj_codebuild_01/_Reports/。

   c. 下列報告將可供使用：

      - AWSM2CCM-Analysis-Report-<timestamp>.pdf 是一項執行報告，可提供 AWS Mainframe Modernization 程式碼轉換帳單和範圍、改善轉換、轉換摘要和詳細的轉換統計

資料。它也會摘要專案層級的程式碼計數和可計費程式碼計數，並提供每個元件的指標和參考成員清單。在執行實際轉換之前，執行和檢查此報告至關重要。

- `Conversion_Detailed_Statistics.txt` 針對每個元件中找到的每個指令，提供頻率和預期的轉換結果（顯示為「轉換狀態」）。這可讓您快速識別指示是否明確表示轉換器不支援。可能的轉換狀態結果為：
  - 完全轉換：指示將準確轉換為 COBOL。
  - 部分轉換：支援 指令，但使用不支援的參數或表達式。轉換後可能需要手動調整。
  - 未轉換：轉換器不支援 指令。
  - 預先編譯確認指示：這些指示通常包含在巨集中，並參考可能也稱為大型主機上的條件式組合語言（例如 AIF、AGO) 指示。這些是由前置編譯器處理，由這類指示或指令所驅動，會選取並產生乾淨/靜態的 ASM 程式碼。這些指示取決於要編譯之巨集參數的實際值。因此，相同的巨集可以產生不同的 ASM 程式碼片段，取決於傳遞參數的值。這是因為存在此類預先編譯指示。在這種情況下，請考慮擴展或重新設計巨集。
- `Conversion_Global_Statistics.txt` 提供元件層級的 Conversion 狀態摘要。
- `CrossReference_PgmToCpyMacro.txt` 報告 Macros 上的 Assembler 程式相依性。它提供快速方法來判斷上傳的程式碼中是否有任何巨集遺失。
- `CrossReference_PgmToPgm.txt` 報告其他 Assembler 程式的 Assembler 程式相依性。它提供快速方法來判斷上傳的程式碼中是否有任何 Assembler 程式遺失。

7. 返回 AWS CodeBuild 服務主控台。
8. 驗證 2-awsm2ccm-analysis 步驟的狀態。它應該在最新建置狀態索引標籤下成功。

您已準備好進行下一個步驟：程式碼轉換。

步驟 9：執行程式碼轉換

> **Important**
>
> AWS Mainframe Modernization 程式碼轉換轉換步驟將根據您的用量計費。如需計費的詳細資訊，請參閱the section called “了解程式碼轉換帳單”。

在此步驟中，您將設定轉換程序，然後啟動建置。

1. 從 AWS Management Console前往 Amazon S3 服務。
2. 找到並按一下 Amazon S3 儲存貯體：`codebuild-regionId-accountId-bucket`。

a. 前往 prj_codebuild_01/。

b. 選取 project_settings.json，然後選擇下載。

c. 開啟 project_settings.json 檔案以查看下列 JSON 結構：

```
{
"Source programs directory":"srclib",
"Source copybooks/macros directory":"macrolib",
"Copybook/Macros Conversion":"Called_only",
"Do not regenerate the Copy/Macro if already exists":"false",
"Target Compiler":"IBM",
"Endianess":"Big",
"Converted programs extension":"",
"Converted CICS programs extension":"",
"Converted copies/macros extension":"",
"Trace Level":"STANDARD",
"Trace file open mode":"append",
"Data definition level":5,
"Start picture column":40,
"Generate Sync FILLER with name":"FILL-SYNC",
"Use SYNC clause":"yes",
"Decimal Point Comma":"true",
"Original Source Placement":"RIGHT"
}
```

其中，

- 來源程式目錄： 包含轉換所需的 Assembler 程式。
- 來源複製手冊/巨集目錄： 包含轉換所需的 Assembler Macros 和複製手冊。
- 複製手冊/巨集轉換可以是：
  - 全部：此選項按鈕表示完整轉換將轉換目錄中可用的所有複本手冊/巨集，無論程式是否正在使用。
  - Called_only：此選項按鈕表示完整轉換只會轉換程式實際使用的複製手冊/巨集。
-

> **Important**
>
> 如果 Copy/Macro 已存在，則不需要重新產生。

當這是 true 時，如果工具已轉換（存在於輸出資料夾中），則工具不會再次轉換 copybook/Macro。

- 目標：轉換程式（產生的程式碼）取決於目標 COBOL 編譯器。支援下列選項：
  - IBM 大型主機的「IBM」
  - Micro Focus COBOL 的「MF」
  - Veryant isCOBOL 的「VERYANT」
  - NTT DATA Enterprise COBOL (Unikix) 的「NTT」
- 結尾和位元：程式的轉換（產生的程式碼）取決於目標平台（位元/結尾）。此組合允許選取下列支援的選項：
  - Endianess：Big（適用於 Big-Endian)/Little (Little-Endian)。例如，IBM z/OS 大型主機為 Big-Endian、Windows 為 Little-Endian、Linux 因發行版本而異（例如 EC2 上的 Amazon Linux 2 為 Little-Endian)。
  - 位元：32/64（如果未指定，則預設為 32)。建議設定為 32 位元。
- 轉換的程式副檔名：這是為產生的 COBOL 程式設定副檔名。空白 (“”)：無延伸。對於 Rocket Software（先前稱為 Micro Focus) COBOL 目標，建議 CBL 讓 Rocket Enterprise Developer 正確識別檔案。
- 轉換的 CICS 程式延伸：這是為產生的 CICS COBOL 程式設定副檔名。空白 (“”)：無延伸。對於 Rocket Software COBOL 目標，建議 CBL 讓 Rocket Enterprise Developer 正確辨識檔案。
- 轉換的複本/巨集延伸：這是為產生的 COBOL 複本設定副檔名。空白 (“”)：無延伸。對於 Rocket Software COBOL 目標，建議 CPY 讓 Rocket Enterprise Developer 正確辨識檔案。
- 追蹤層級：追蹤是在轉換期間使用 CodeBuild 記錄的資訊。使用者可以透過選取其中一個提供的選項來選取詳細資訊層級。
  - ERROR = TRACE ERROR：只會顯示轉換錯誤。
  - STANDARD = TRACE STANDARD：顯示轉換錯誤和標準資訊。這是建議的設定。
  - ALL = TRACE ALL：追蹤層級上限
- 追蹤檔案開啟模式：未使用。建議預設設定為附加。
- 資料定義層級：這表示工作儲存和連結區段中定義的子欄位初始層級（在層級「01」之後）。必須是數字。
- 開始圖片欄：這與產生的 COBOL 程式碼格式有關，並指出放置 PIC 子句的欄（在欄位名稱之後）。必須是數字。

- 原始來源置放：這表示註解放置在程式中的位置。它有兩個選項：
  - 右：此選項會將註解或其他資訊放在七十三 (73) 欄之後的正確位置。在 COBOL 中，程式碼會寫入前七十二 (1-72) 欄，而來自七十三 (>= 73) 欄的任何內容都會視為註解。
  - 上述：此選項會將註解放在翻譯內容上方。
- 使用名稱產生同步 FILLER：此選項與二進位欄位記憶體中的對齊有關 (Assembler "H"、"F"、"D" 資料類型，這些資料類型會轉換為 COBOL "COMP" 資料類型）。為了確保適當的對齊界限，轉換期間會新增明確的填充欄位。這是文字型選項，值必須是字串（例如 FILL-SYNC)。
- 使用 SYNC 子句：此選項是指二進位欄位記憶體中的對齊。是 = 轉換為 COBOL 的所有欄位。「COMP」將以「SYNC」子句定義（例如，05 WRKFLD PIC S9(09) COMP SYNC)。
- 小數點逗號：若為 true，DECIMAL-POINT IS COMMA 子句會新增至 "SPECIAL-NAMES" COBOL 段落。

d. 根據您的需求，變更適當的參數，然後儲存 `project_settings.json`。

e. 從 Amazon S3 儲存貯`prj_codebuild_01/`體中移除現有`project_settings.json`檔案，然後上傳新版本。

3. 返回 AWS CodeBuild 服務。
4. 選取要建置您先前建立的專案： `3-awsm2ccm-convert`

a. 選擇開始建置，然後立即開始將 Assembler 程式和巨集轉換為 COBOL 程式和複製手冊。

b. 等待此專案的建置狀態變更為成功。它將在最新建置狀態索引標籤下。

步驟 10：驗證程式碼轉換

1. 從 AWS Management Console，前往 Amazon S3 服務。
2. 找到並按一下 Amazon S3 儲存貯體：`codebuild-regionId-accountId-bucket`。
3. 導覽至 **`awsm2ccm-do-not-delete`** 。AWS Mainframe Modernization Code 轉換會在轉換過程中為每個 Assembler 或 Macro 模組建立編碼的二進位檔案。這些檔案對於防止向客戶重複計費，以及追蹤分析和轉換多少提供的 Assembler 程式碼至關重要。檔案會存放在下列位置：`codebuild-regionId-accountId- bucket/awsm2ccm-do-not-delete/<your_AWS_account_id>/Hash`。編碼的檔案不包含任何 Assembler 程式碼，也無法從這些檔案中擷取客戶程式碼。

> **Important**
>
> 既不手動編輯這些檔案，也不刪除這些檔案。編輯或刪除這些檔案可能會導致相同元件的多個帳單。

將**awsm2ccm-do-not-delete/**資料夾視為系統受管目錄。在對此目錄或其內容進行任何變更支援 之前，請參閱 。

4. 按一下codebuild-regionId-accountId-bucket以返回儲存貯體。
5. 選擇 **ARTIFACTS/prj_codebuild_01/**。_Converted/ 資料夾包含程式碼轉換步驟所產生的 COBOL 輸出。它將具有下列子目錄：
   - copybooks/ 資料夾包含產生的 COBOL 複製手冊。
   - 程式/資料夾包含產生的 COBOL 程式。
   - runtime_lib/ 資料夾包含解決方案提供的其他 COBOL 程式和複製手冊。
6. 如果分析報告和其他報告指出轉換成功，且 AWS CodeBuild 專案3-awsm2ccm-convert標記為成功，請從 _Converted/ 目錄下載 COBOL 程式碼和複製手冊。

步驟 11：下載轉換後的程式碼

在此步驟中，從 _Converted/ 目錄下載 COBOL 程式碼和複製手冊，並在目標 COBOL 環境中編譯它們。

1. 從 AWS Management Console，前往 Amazon S3 服務。
2. 找到並按一下 Amazon S3 儲存貯體：codebuild-regionId-accountId-bucket。
3. 導覽至位置： ARTIFACTS/prj_codebuild_01/_Converted/。
4. 從 _Converted/ 下的所有子目錄下載轉換後的 COBOL 程式碼。您也可以使用下列 CLI 命令一次下載：

```
aws s3 cp s3://codebuild-regionId-accountId-
bucket/ARTIFACTS/prj_codebuild_01/_Converted/ . --recursive
```

5. 在目標 COBOL 環境中分析和編譯轉換後的 COBOL。

清除資源

如果您不再需要為本教學課程建立的資源，請將其刪除以避免額外費用。若要這樣做，請完成下列步驟：

- 刪除您為此教學課程建立的 S3 儲存貯體。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的刪除儲存貯體。
- 刪除您為此教學課程建立的 IAM 政策。如需詳細資訊，請參閱《IAM 使用者指南》中的刪除 IAM 政策。
- 刪除您為此教學課程建立的 IAM 角色。如需詳細資訊，請參閱《IAM 使用者指南》中的刪除角色或執行個體描述檔。
- 刪除您為此教學課程建立的 CodeBuild 專案。如需詳細資訊，請參閱AWS CodeBuild 《 使用者指南》中的在 CodeBuild 中刪除組建專案。

Charon 整合

Charon-SSP 簡介

1987 年，Sun Microsystems 發佈 SPARC V7 處理器，這是一個 32 位元的 RISC 處理器。SPARC V8 於 1990 年推出 - 原始 SPARC V7 的修訂版，其中包含最顯著的硬體分割和乘以指示。SPARC V8 處理器是 SPARCstation 5、10 和 20 等多個伺服器和工作站的基礎。1993 年，SPARC V8 後面接著 64 位元 SPARC V9 處理器。這也成為許多伺服器和工作站的基礎，例如 Enterprise 250 和 450。

由於硬體淘汰和缺少備用或重新組裝的組件，為這些較舊的 SPARC 型工作站和伺服器開發的軟體和系統變得難以維護。為了滿足某些end-of-life SPARC 型系統的持續需求，Stromasys S.A. 開發了 SPARC 模擬器產品的 Charon-SSP 系列。下列產品是指定原生硬體 SPARC 系統的軟體型虛擬機器替換。以下是模擬硬體系列的一般概觀。

Charon-SSP/4M 模擬下列 SPARC 硬體：

- Sun-4m 系列（由 Sun SPARCstation 20 表示）：最初是多處理器 Sun-4 變體，以 SPARCServer 600MP 系列推出的 MBus 處理器模組匯流排為基礎。Sun-4m 架構之後也包含非 MBus 單處理器系統，例如使用 SPARC V8-architecture處理器的 SPAR 底座 5。從 SunOS 4.1.2 和 Solaris 2.1 開始支援到 Solaris 9。SPARCServer 600MP 支援在 Solaris 2.5.1 之後遭到捨棄。

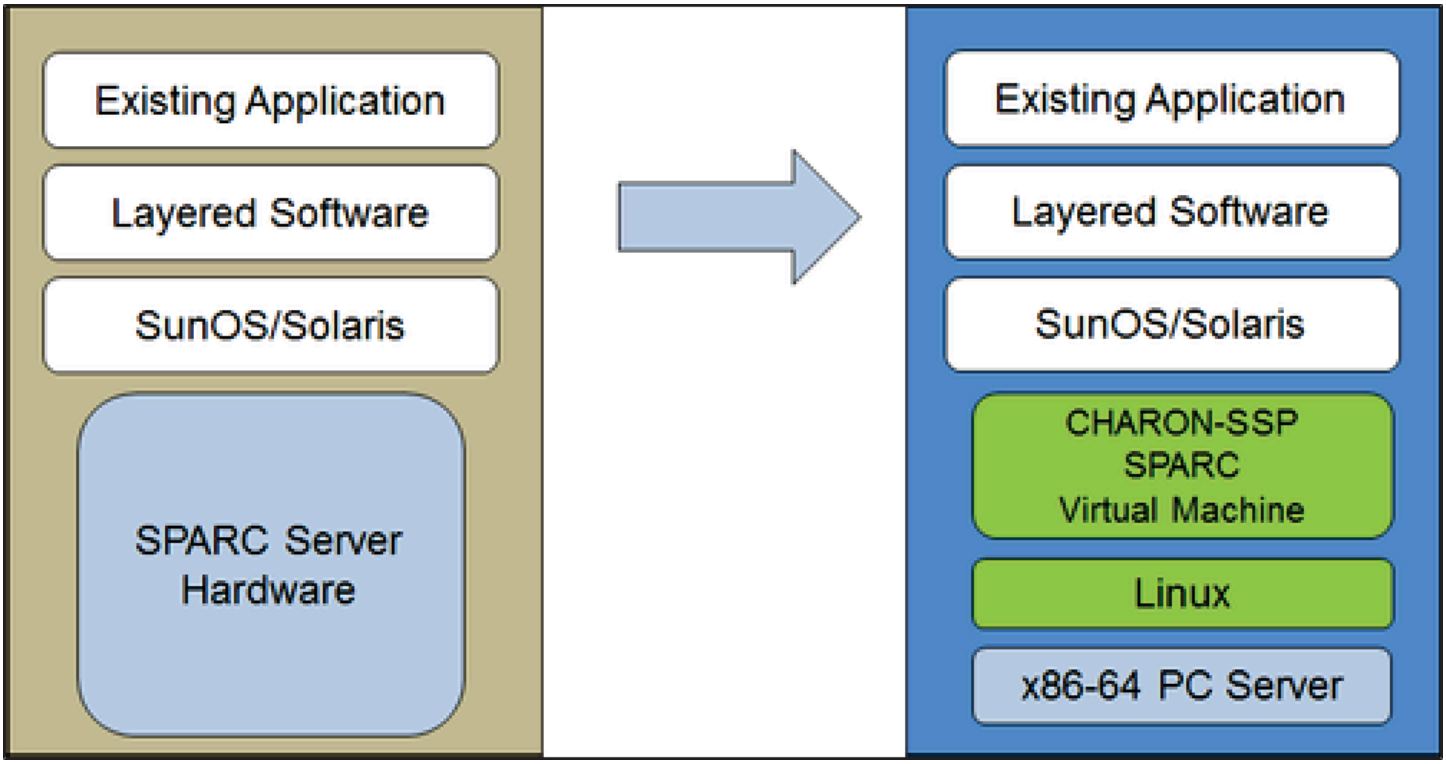
Charon-SSP/4U(+) 模擬下列 SPARC 硬體：

- Sun-4u 系列（由 Sun Enterprise 450 表示）：(U for UltraSPARC) - 此變體推出 Sun Ultra 系列首次使用的 64 位元 SPARC V9 處理器架構和 UPA 處理器互連。從 2.5.1 版開始支援 32 位元版本的 Solaris。Sun-4u 的第一個 64 位元 Solaris 版本是 Solaris 7。UltraSPARC I 支援在 Solaris 9 之後遭到捨棄。Solaris 10 支援從 UltraSP ARC II 到 UltraSPARC IV 的 Sun-4u 實作。

Charon-SSP/4V(+) 模擬下列 SPARC 硬體：

- Sun-4v 系列（由 SPARC T2 和 T4 表示）：此變化將 Hypervisor 處理器虛擬化新增至 Ultra SPARC T1 多核心處理器中推出的 Sun-4u；。從 3/05 HW2 版開始，Solaris 第 10 版支援選取的硬體（大多數模型 - 包括 Charon-SSP 模擬的硬體 - 需要較新版本的 Solaris 10)。也支援數個 Solaris 11 版本。

下圖顯示將實體硬體遷移至模擬器的基本概念。



Charon-SSP 虛擬機器可讓 Sun 和 Oracle SPARC 型電腦的使用者，以幾乎不需要變更原始系統組態的方式取代其原生硬體。這表示您可以繼續執行應用程式和資料，而不需要切換或移植到另一個平台。Charon-SSP 軟體會在商品 Intel 64 位元系統上執行，確保持續保護您的投資。

Charon-SSP/4U+ 支援與 Charon-SSP/4U 相同的虛擬 SPARC 平台，以及與 Charon-SSP/4V 相同的 Charon-SSP/4V+。不過，4U+ 和 4V+ 版本利用 Intel 的 VTx/EPT 和 AMD 的 AMD-v/NPT 硬體輔助現代 CPUs 中的虛擬化技術，以提供更好的虛擬 CPU 效能。Charon-SSP/4U+ 和 Charon-SSP/4V+ 需要支援 VT-x/EPT 或 AMD-v/NPT CPUs，且必須安裝在專用主機系統上。不支援在 VM 中執行這些產品變體 （例如 VMware)。

> **Note**
>
> 如果您計劃在雲端環境中執行 Charon-SSP/4U+ 或 4V+，請聯絡 Stromasys 或 Stromasys VAR 以討論您的需求。

支援的訪客作業系統

Charon-SSP/4M 虛擬機器支援下列訪客作業系統版本：

- SunOS 4.1.3 - 4.1.4

- Solaris 2.3 到 Solaris 9

Charon-SSP/4U(+) 虛擬機器支援下列訪客作業系統版本：

- Solaris 2.5.1 到 Solaris 10

Charon-SSP/4V(+) 虛擬機器支援下列訪客作業系統版本：

- Solaris 10 （從 2007 年 8 月 4 日更新開始） 和 Solaris 11.1 更新至 Solaris 11.4

對於 Charon-SSP/4V(+)，請注意下列事項：

- 對於模擬 SPARC T4，支援的 Solaris 10 版本包括：Oracle Solaris 10 1/13、Oracle Solaris 10 8/11 和 Solaris 10 9/10，或使用 Oracle Solaris 10 8/11 修補程式集的 Solaris 10 10/09。
- 模擬 SPARC T4 模型是在模擬器中執行 Solaris 11.4 的先決條件。
- 不支援 Solaris 核心區域。

Charon-SSP 雲端執行個體先決條件

透過選取執行個體類型或形狀，您可以選取用於雲端中 Charon-SSP 主機執行個體的虛擬硬體。因此，選擇執行個體類型或形狀會決定 Charon-SSP 虛擬主機硬體的硬體特性 （例如，有多少 CPU 核心，以及虛擬 Charon 主機系統有多少記憶體）。

> **Note**
>
> 如果您使用 Charon-SSP 市集映像來啟動執行個體，則會滿足所有 Linux 主機作業系統需求。

最低硬體需求如下所述。

有關調整大小準則的重點：

- 以下關於主機 CPU 核心數量和主機記憶體的大小調整準則，顯示最低需求。必須檢閱每個部署情況，並視需要調整實際的主機大小。例如，如果訪客應用程式產生高 I/O 負載，則必須增加可用於 I/O 的 CPU 核心數量。此外，具有許多模擬 CPUs 的系統通常能夠建立較高的 I/O 負載，因此可能需要增加可用於 I/O 的 CPU 核心數量。在超執行緒環境中，為了獲得最佳效能，CPU 核心 （即實際/

實體 CPUs) 的數量必須足以滿足作用中模擬器的 CPU 需求，因此避免高工作負載執行緒共用一個實體 CPU 核心。

- 模擬 CPUs 的 CPU 核心配置和用於 I/O 處理的 CPU 核心配置取決於組態。如需此項目的詳細資訊，以及用於 I/O 處理的 CPU 核心預設配置，請參閱一般 Charon-SSP 使用者指南中的 CPU 組態。

⚠ 重要一般資訊

- 為了方便將模擬器資料從一個雲端執行個體快速傳輸到另一個雲端執行個體，強烈建議將所有相關模擬器資料存放在單獨的磁碟區上，以便輕鬆地從舊執行個體分離並連接到新執行個體。
- 請務必從頭開始正確調整執行個體的維度（請檢查以下最低需求）。第一次啟動執行個體時，會建立 Charon-SSP AL 的 Charon-SSP 授權。稍後變更至另一個執行個體大小/類型，進而變更 CPU 核心的數量，將使授權失效，進而防止 Charon 執行個體啟動（需要新的執行個體）。如果計劃在 AutoVE 模式下使用 Charon-SSP AL 執行個體，請務必在第一次啟動之前包含 AutoVE 伺服器資訊，否則將使用公有授權伺服器。Charon-SSP VE 的授權是根據授權伺服器上取得的指紋建立。如果授權伺服器直接在模擬器主機上執行，且模擬器主機稍後需要變更 CPU 核心數量，授權將會失效（需要新的授權和可能需要新的執行個體）。

執行個體先決條件

一般 CPU 需求：Charon-SSP 支援以 Amazon EC2 執行個體為基礎的現代 x86-64 架構處理器。

Charon-SSP 的最低需求：

- 主機系統 CPU 核心的最小數量：
  - 主機作業系統至少有一個 CPU 核心，加上：
  - 對於每個模擬 SPARC 系統：
    - 執行個體的每個模擬 CPU 一個 CPU 核心，加上：
    - 至少一個額外的 CPU 核心進行 I/O 處理（如果使用伺服器 JIT 最佳化，則至少有兩個）。如需組態選項，請參閱上述 CPU 組態一節。根據預設，Charon 會將 Charon 主機可見的 CPUs 數量的 1/3（最少 1；四捨五入）指派給 I/O 處理。
- 最低記憶體需求：

  - Linux 主機作業系統的 4GB 或更多 RAM。實際需求可能較高，且取決於 Linux 主機上執行的非模擬器服務需求。先前針對 Linux 主機建議至少 2GB 的 RAM 對許多系統仍然有效，但 Linux 作業系統和應用程式的需求不斷增加，導致新安裝的建議更新。此外：
  - 對於每個模擬 SPARC 系統：
    - 模擬執行個體的已設定記憶體，加上：
    - 2GB 的 RAM （如果使用伺服器 JIT，則為 6GB 的 RAM)，以允許 DIT 最佳化、模擬器需求、執行時間緩衝區、SMP 和圖形模擬。
- 如果在現代 x86-64 CPUs 上啟用超執行緒，兩個執行緒可以在一個實體 CPU 核心上執行，為主機作業系統提供兩個邏輯 CPUs。如果可能，請在 Charon-SSP 主機上停用超執行緒。不過，在 VMware 和雲端環境中，這通常是不可能的，或者不清楚是否使用超執行緒。Charon-SSP 超執行緒選項可讓 Charon-SSP 適應此類環境。如需詳細組態資訊，請參閱上述一般 Charon-SSP 使用者指南中的 CPU 組態一節。P 租用備註：為了獲得最佳效能，Charon-SSP 執行緒不應共用實體 CPU 核心 - 主機系統上應該有足夠的實體核心，以滿足設定的模擬器 (s) 的需求。
- 一或多個網路介面，視客戶需求而定。
- Charon-SSP/4U+ 和 Charon-SSP/4V+ 必須在支援 Intel VT-x/EPT 或 AMD-v/NPT （裸機執行個體） 的實體硬體上執行，因此無法在所有雲端環境中執行。請查看雲端供應商的文件，了解此類硬體的可用性。此外，請注意下列幾點：
  - 只有在使用 Stromasys 支援的 Linux 核心時，才能使用 Charon-SSP/4U+ 和 Charon-SSP/4V+。
  - 如果您需要此類型的模擬 SPARC 硬體，請聯絡 Stromasys 或您的 Stromasys VAR 以詳細討論您的需求。

為 Charon 建立和設定 AWS 雲端執行個體 （新 GUI)

本節反映 2022 年春 AWS Management Console 天的 。如果您仍然使用較舊的主控台，請參閱 Charon-SSP AWS 入門指南的附錄。

一般先決條件

此描述顯示 Linux 執行個體的基本設定 AWS。它不會列出特定的先決條件。不過，根據您的使用案例，請考慮下列先決條件：

- Amazon 帳戶和 AWS Marketplace 訂閱
  - 若要在 中設定 Linux 執行個體 AWS，您需要具有管理員存取權 AWS 的帳戶。
  - 識別您計劃啟動執行個體 AWS 的區域。確保您 AWS 計劃使用的服務可在該區域中使用。請參閱AWS 依區域的服務。

- 識別您計劃在其中啟動執行個體的 VPC 和子網路。
- 如果您的執行個體需要網際網路存取，請確定與您的 VPC 相關聯的路由表具有網際網路閘道。如果您的執行個體需要 VPN 存取您的內部部署網路，請確定 VPN 閘道可用。VPC 及其子網路的確切組態將取決於您的網路設計和應用程式需求。
- 若要訂閱特定 AWS Marketplace 服務，請在 中選擇 AWS Marketplace 訂閱 AWS Management Console ，然後選擇管理訂閱。
- 搜尋您計劃使用的服務並訂閱該服務。訂閱成功後，您會在管理訂閱區段中找到訂閱。您可以從那裡直接啟動新的執行個體。

- 執行個體硬體和軟體先決條件會根據執行個體的計劃使用而有所不同：
  - 選項 1：執行個體用作 Charon 模擬器主機系統：
    - 請參閱《 使用者指南》和/或 Charon 產品入門指南》中的硬體和軟體先決條件章節，以判斷 Linux 執行個體必須滿足的確切硬體和軟體先決條件。您用來啟動執行個體的映像，以及您選擇的執行個體類型，會決定雲端執行個體的軟體和硬體。
    - 執行模擬舊版系統需要 Charon 產品授權。請參閱 Charon 產品文件中的授權資訊，或聯絡您的 Stromasys 代表或 Stromasys VAR 以取得其他資訊。
  - 選項 2：執行個體用作專用 VE 授權伺服器：
    - 如需詳細的先決條件，請參閱 VE License Server 指南。
- 可在 Charon 模擬器產品提供的模擬系統中執行的某些舊版作業系統需要作業系統原始廠商的授權。使用者負責與舊版作業系統相關的任何授權義務，並且必須提供適當的授權。

使用 AWS Management Console 啟動新的執行個體

建立新的執行個體

1. 登入 AWS Management Console ，並在 https：//Amazon EC2 主控台：//https：/https://console.aws.amazon.com/ec2/.microsoft.com。
2. 選擇啟動執行個體。
3. 輸入執行個體的名稱。
4. 選取 AMI。AMI 是用來啟動雲端執行個體的預先封裝映像。它包含作業系統和適用的應用程式軟體。AMI 的選擇取決於您計劃如何使用執行個體：
   - 如果要將執行個體用作 Charon 模擬器主機系統，則可以選擇多個 AMI：
     - 從預先封裝的 Charon Marketplace 映像安裝 Charon 主機系統：其中包含基礎作業系統和預先安裝的 Charon 軟體。

- 請洽詢您的 Stromasys 代表，了解您的雲端供應商市集目前提供哪些選項。
- 根據雲端供應商和 Stromasys 產品發行計劃，有兩種變體：
  - 自動授權 (AL)，可搭配公有、Stromasys 操作的授權伺服器，或搭配私有、客戶操作的 AutoVE 授權伺服器使用
  - 虛擬環境 (VE)，用於私有、客戶操作的 VE 授權伺服器
- 使用傳統 Charon 模擬器安裝搭配適用於 Linux 的 Charon 模擬器安裝 RPM 套件來安裝 Charon 主機系統：
  - 選擇所選 Charon 產品和版本支援的發行版本的 Linux AMI。請參閱 Stromasys 文件網站上的產品使用者指南。
- 如果要將執行個體用作專用 VE 授權伺服器，請參閱授權文件中的 VE 授權伺服器指南，了解 Linux 執行個體的需求。

在您決定需要哪個 AMI 之後，請選取相符的 Linux 或 Charon 產品 AMI。如果您沒有看到所需的 AMI，請選擇瀏覽更多 AMIs。選擇與您計劃使用執行個體的方式相符的 Linux AMI。可為下列其中之一：

- 預先封裝的 Charon VE 市集映像。AMI 的名稱將包含字串 "ve"。
- 自動授權或 AutoVE 的預先封裝 Charon AL 市集映像。
- RPM 產品安裝支援的 Linux 版本。
- VE 授權伺服器支援的 Linux 版本。

5. 選取執行個體類型。Amazon EC2 提供具有不同 CPU、記憶體、儲存體和聯網容量組合的執行個體類型。選取符合您要使用之 Charon 產品需求的執行個體類型。某些市場映像的執行個體類型選擇有限。
6. 選取現有的金鑰對，或建立並儲存新的金鑰對。如果您選取現有的金鑰對，請確定您有相符的私有金鑰。否則，您將無法連線到執行個體。

> **Note**
>
> 如果您的管理系統支援它，對於 RHEL 9.x、Rocky Linux 9.x 和 Oracle Linux 9.x，請使用 SSH 金鑰類型 ECDSA 或 ED25519。這些類型可讓您使用 SSH 通道連線到這些 Charon 主機 Linux 系統，而不需要將 Charon 主機上的預設加密政策設定變更為較不安全的設定。例如，這對 Charon-SSP Manager 很重要。請參閱 Red Hat 文件中的使用全系統密碼編譯政策。

7. 在網路設定區段中選擇編輯。選擇與您的環境對應的設定。

- 指定 VPC。
- 指定現有的子網路或建立新的子網路。
- 啟用或停用公有 IP 地址自動指派給主要介面。只有在執行個體具有單一網路界面時，才能自動指派。
- 指派現有或新的自訂安全群組。安全群組必須至少允許 SSH 存取執行個體。您計劃在執行個體上執行的應用程式所需的任何連接埠也必須被允許。您可以在建立執行個體之後隨時修改安全群組。

8. 在儲存區段中，針對根磁碟區（系統磁碟），選擇適合您環境的大小。Linux 系統的建議最小系統磁碟大小為 30 GiB。若要為虛擬磁碟容器和其他儲存需求提供空間，您可以立即或在啟動執行個體之後新增更多儲存空間。但是系統磁碟大小必須涵蓋 Linux 系統需求，包括您計劃安裝的任何應用程式和公用程式。

> **Note**
>
> 建議您為 Charon 應用程式資料（例如磁碟映像）建立個別的儲存磁碟區。如有必要，您可以稍後將這類磁碟區遷移到另一個執行個體。

9. 展開進階詳細資訊區段，向下捲動，然後選取指定 CPU 選項。下圖顯示三個更可能對 Charon 模擬器環境有用的範例。

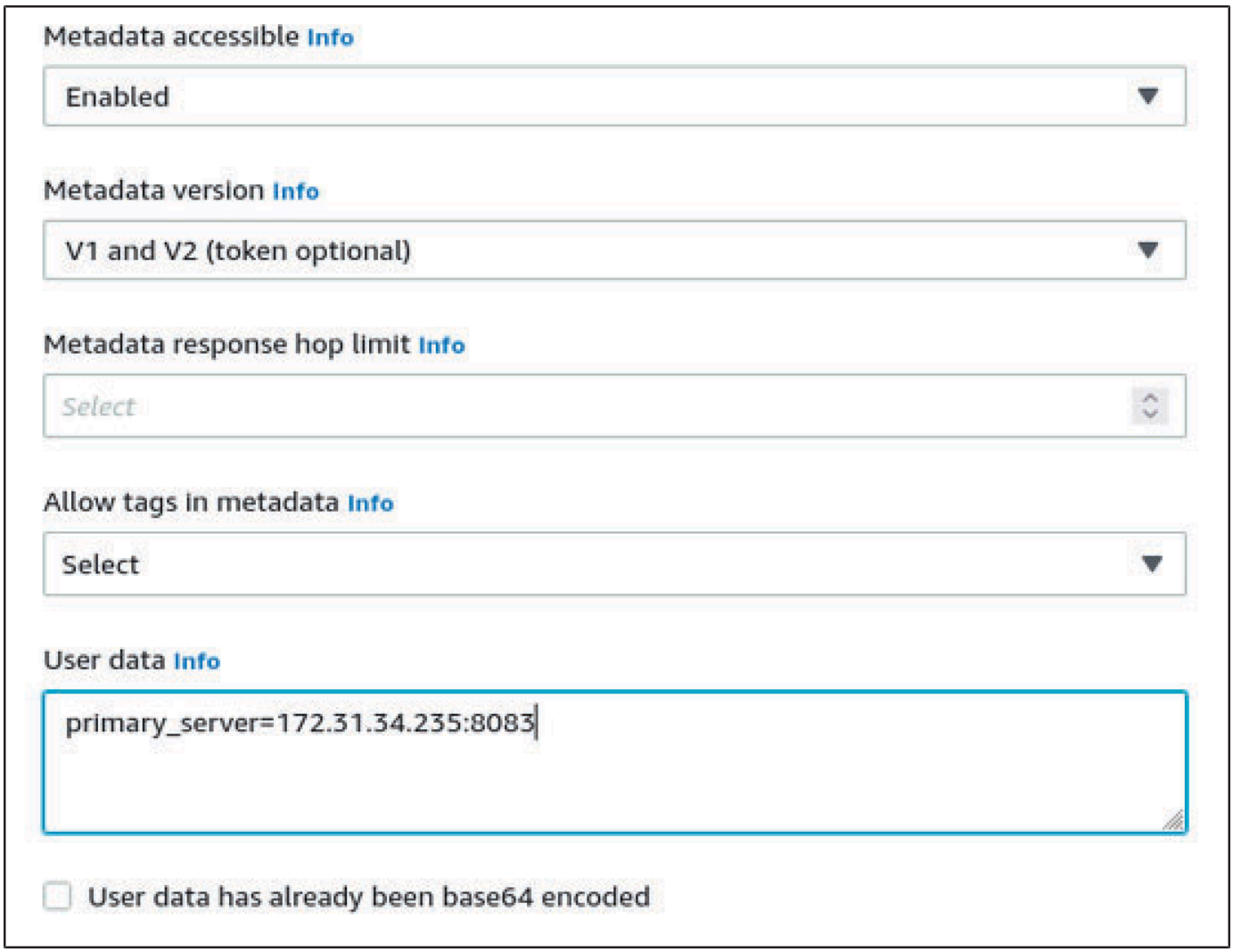


10. 對於 VE 授權伺服器系統，若版本早於 1.1.23您必須將所需的 IAM 角色指派給執行個體。它必須是允許 `ListUsers`動作的角色。若要指派角色，請在展開的進階詳細資訊區段中選取 IAM 執行

個體設定檔下的角色，或選擇建立新的 IAM 設定檔。如需詳細資訊，請參閱 Amazon EC2 的 IAM 角色。

11. 如果您的執行個體是以 Charon AL AWS Marketplace 映像為基礎，且您計劃使用 Stromasys 操作的公有授權伺服器，您必須在啟動執行個體之前，將對應的資訊新增至執行個體組態。

    輸入 AutoVE 授權伺服器的資訊，如下圖所示。



    以下是有效的使用者資料組態選項：

    - **primary_server=**<*ip-address*>[**:**<*port*>]
    - **backup_server=**<*ip-address*>[**:**<*port*>]

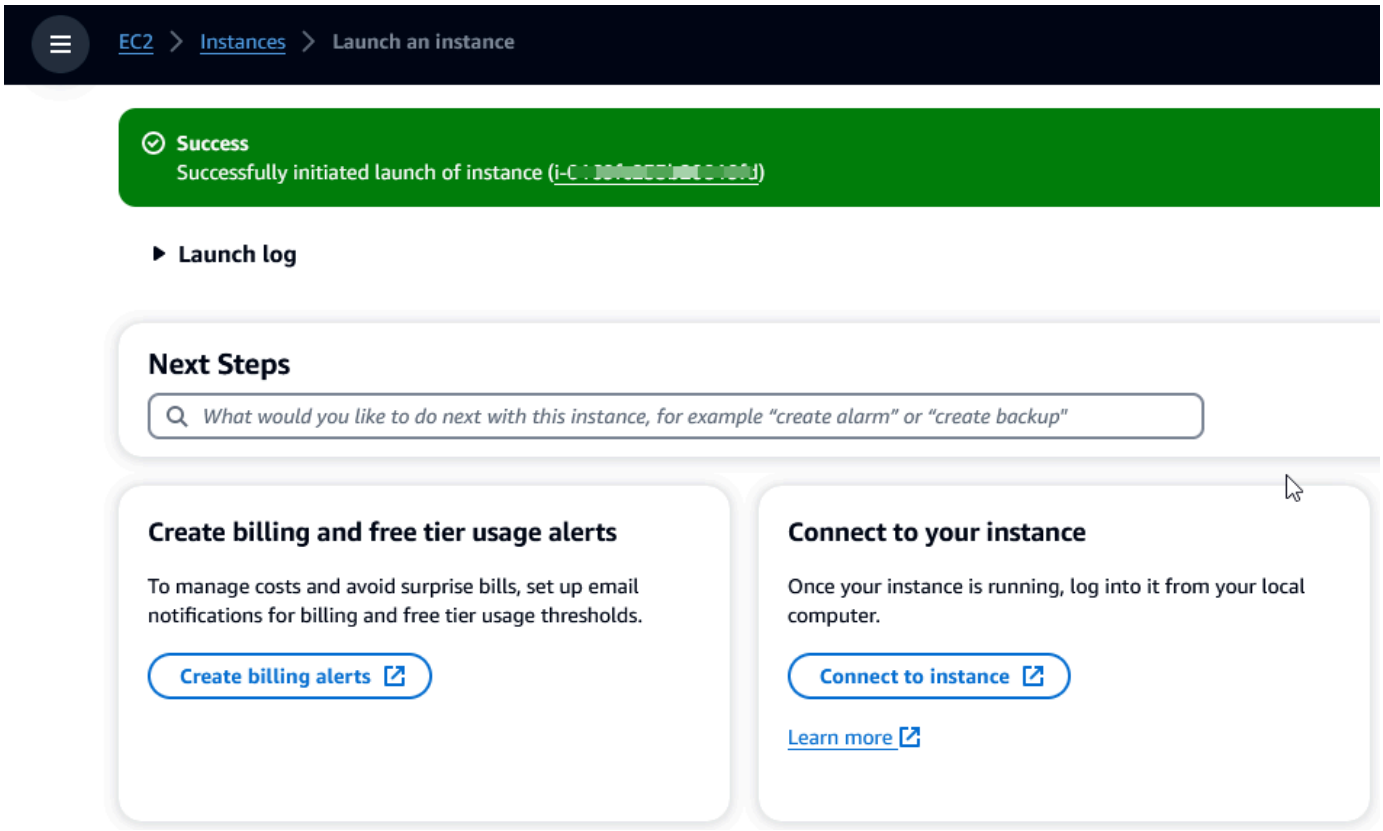
    位置

    - <ip-address> 代表主伺服器和備份伺服器的 IP 地址，如適用。

- <port> 代表用於與授權伺服器通訊的非預設 TCP 連接埠 （ 預設值：TCP/8083)。

> **Note**
>
> 初次啟動時必須至少設定一個授權伺服器，才能啟用 AutoVE 模式。0否則，執行個體會繫結至 Stromasys 操作的其中一個公有授權伺服器。

12. 在摘要區段中，選擇啟動執行個體。一段時間後，您將看到以下成功訊息：



13. 在畫面的右下角，選擇檢視所有執行個體。
14. 若要查看執行個體的詳細資訊，請選取執行個體資料表中代表執行個體之資料列左側的核取方塊。您的執行個體詳細資訊會顯示在畫面的下半部。如需如何連線至執行個體的資訊，請參閱《Amazon EC2 使用者指南》中的連線。

AWS 使用 NTT DATA 進行大型主機現代化轉換

AWS Mainframe Modernization 提供各種 Amazon Machine Image (AMIs)。這些 AMIs 可加速 Amazon EC2 執行個體的快速佈建， AWS 透過 NTT Data 建立量身打造的環境，以便在 中重新託管和重建大型主機應用程式。本指南提供存取和使用這些 AMIs 所需的步驟。

先決條件

- 請確定您擁有可建立 Amazon EC2 執行個體之 AWS 帳戶的管理員存取權。
- 確認 AWS Mainframe Modernization 服務在您計劃建立 Amazon EC2 執行個體的區域中可用。請參閱依區域提供的 AWS 服務清單。
- 識別您要建立 Amazon EC2 執行個體的 Amazon VPC。

訂閱 Amazon Machine Image

當您訂閱 AWS Marketplace 產品時，您可以從產品的 AMI 啟動執行個體。

1. 登入 AWS Management Console 並開啟位於 https：//https://console.aws.amazon.com/marketplace 的 AWS Marketplace 主控台。
2. 選擇 Manage subscriptions (管理訂閱)。
3. 將以下連結複製並貼到瀏覽器地址列：https：//https://aws.amazon.com/marketplace/pp/prodview-eg227ymldsnx2
4. 選擇 Continue to Subscribe (繼續以訂閱)。
5. 如果條款與條件是可接受的，請選擇接受條款。訂閱可能需要幾分鐘的時間來處理。
6. 等待顯示感謝訊息。此訊息確認您已成功訂閱產品。
7. 在左側導覽窗格中，選擇管理訂閱。此檢視會顯示您的所有訂閱。

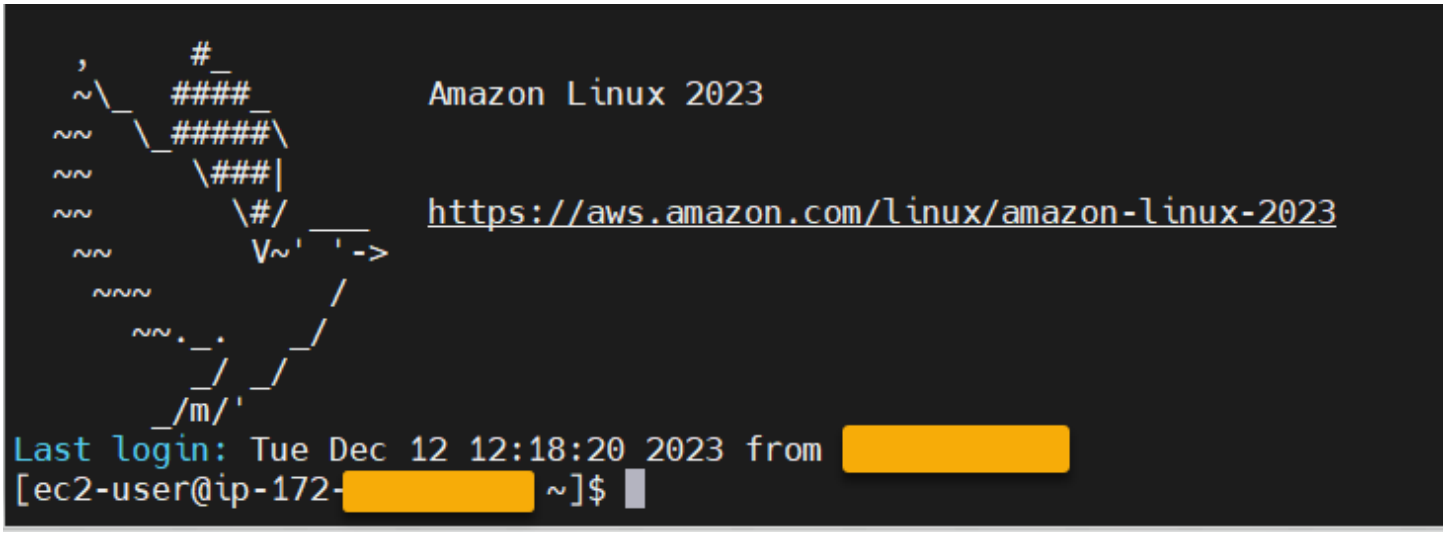
使用 NTT DATA 執行個體啟動 AWS 大型主機現代化平台

1. 在 https：//https://console.aws.amazon.com/marketplace 開啟 AWS Marketplace 主控台。
2. 在左側導覽窗格中，選擇管理訂閱。
3. 尋找您要啟動的 AMI，然後選擇啟動新執行個體。
4. 在區域下，選取允許列出的區域。

5. 選擇繼續以透過 EC2 啟動。此動作會帶您前往 Amazon EC2 主控台。
6. 輸入伺服器的名稱。
7. 選取符合您專案效能和成本需求的執行個體類型。建議的執行個體大小起點為 `c5.2xLarge`。
8. 選擇現有的金鑰對，或建立並儲存新的金鑰對。如需金鑰對的詳細資訊，請參閱《Amazon EC2 使用者指南》中的 Amazon EC2 金鑰對和 Linux 執行個體。 Amazon EC2
9. 編輯網路設定，然後選擇允許列出的 VPC 和適當的子網路。
10. 選擇現有的安全群組或建立新的安全群組。如果這是 Enterprise Server Amazon EC2 執行個體，通常允許連接埠 86 和 10086 的 TCP 流量管理 Rocket Software （ 先前稱為 Micro Focus) 組態。
11. 設定 Amazon EC2 執行個體的儲存體。
12. 檢閱摘要，然後選擇啟動執行個體。若要成功啟動，執行個體類型必須是有效的。如果啟動失敗，請選擇編輯執行個體組態，然後選擇不同的執行個體類型。
13. 看到成功訊息後，請選擇連線至執行個體。
14. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
15. 在左側導覽窗格中的執行個體功能表下，選擇執行個體。
16. 在主窗格中，檢查執行個體的狀態。

NTT 資料入門

在您佈建 Amazon EC2 執行個體之後，SSH 會以使用者名稱 插入其中`ec2-user`。畫面看起來如下圖所示。



在 `/opt/software/` 資料夾下，有一個名為 的資料夾`UniKix_Product_Guides`，如下圖所示。

```
[ec2-user@ip-172-          ~]$ ls -l /opt/software/
total 64
lrwxrwxrwx.  1 root     root        23 Oct 17 19:27 BPE -> /opt/software/BPE17.2.3
drwxr-xr-x.  6 ec2-user ec2-user 16384 Oct  4 16:38 BPE17.2.3
lrwxrwxrwx.  1 ec2-user ec2-user    32 Oct 17 19:27 COBOL -> /opt/software/NTT_DATA_COBOL_6.5
drwxr-xr-x. 11 ec2-user ec2-user 16384 Oct 17 19:27 NTT_DATA_COBOL_6.5
lrwxrwxrwx.  1 ec2-user ec2-user    36 Oct 17 19:28 NTT_DATA_TPE_Agent -> /opt/software/NTT_DATA_TPE_Agent_4.9
drwxr-xr-x.  8 ec2-user ec2-user 16384 Nov  9 01:59 NTT_DATA_TPE_Agent_4.9
lrwxrwxrwx.  1 ec2-user ec2-user    25 Oct 17 19:28 Secure -> /opt/software/Secure6.3.1
drwxr-xr-x.  8 ec2-user ec2-user   156 Oct 17 19:28 Secure6.3.1
lrwxrwxrwx.  1 ec2-user ec2-user    23 Oct 17 19:27 TPE -> /opt/software/TPE17.2.2
drwxr-xr-x. 12 ec2-user ec2-user 16384 Oct  4 16:34 TPE17.2.2
lrwxrwxrwx.  1 ec2-user ec2-user    20 Oct 17 19:28 UCM -> /opt/software/UCM2.1
drwxr-xr-x.  7 ec2-user ec2-user   173 Oct 17 19:28 UCM2.1
drwxr-xr-x.  2 ec2-user ec2-user     6 Dec 12 12:20 UniKix_Product_Guides
drwxr-xr-x.  2 ec2-user ec2-user     6 Oct 17 19:22 bin
drwxr-xr-x.  2 ec2-user ec2-user    34 Nov 10 17:03 license
drwxr-xr-x.  8 root     root        88 Oct 17 19:28 staging
```

`UniKix_Product_Guides` 資料夾包含此 Amazon EC2 執行個體上安裝的下列元件的文件：

- NTT DATA TPE
- NTT DATA BPE
- NTT DATA Enterprise COBOL
- NTT DATA UniKix 安全
- NTT DATA UniKix Central Manager

上一個影像中顯示的`software`資料夾具有上述元件的二進位檔。

成功驗證 Amazon EC2 執行個體後，請依照 NTT 資料文件開始使用 AWS 大型主機現代化 Replatform 搭配 NTT DATA。

教學課程：在 NTT DATA 上部署 CardDemo 應用程式

此頁面會逐步引導您透過 NTT DATA Unikix step-by-step 執行時間，在 AWS 大型主機現代化平台部署 CardDemo 範例應用程式。

CardDemo 範例應用程式是一種簡化的大型主機應用程式，旨在測試和展示大型主機遷移和現代化使用案例的 AWS 合作夥伴技術。

如需此應用程式的詳細資訊，請參閱 CardDemo 的 GitHub 儲存庫。

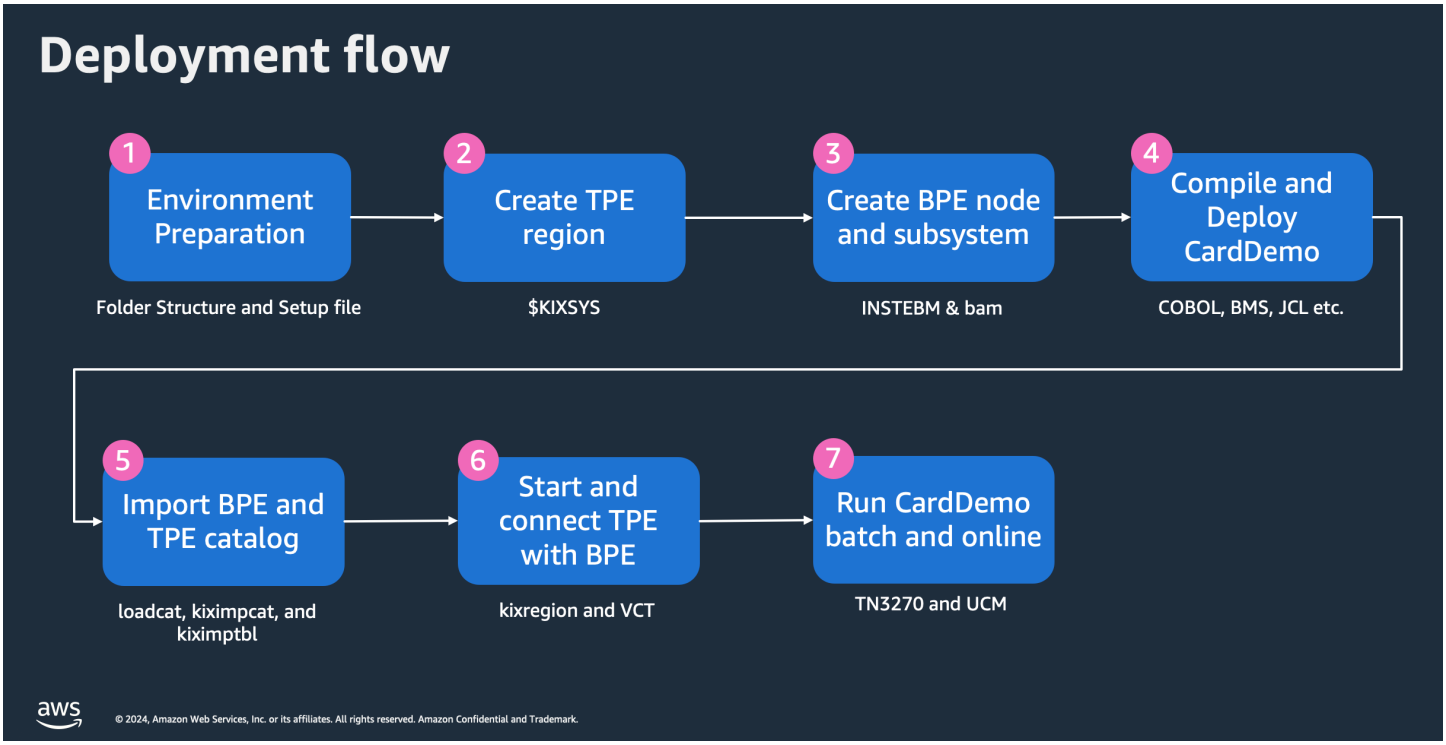
主題

- 部署流程圖

- 先決條件
- 步驟 1：準備環境
- 步驟 2：建立 TPE 區域
- 步驟 3：建立 BPE 節點和子系統
- 步驟 4：編譯和部署 CardDemo 應用程式
- 步驟 5：匯入 BPE 和 TPE 目錄
- 步驟 6：啟動並連接 TPE 與 BPE
- 步驟 7：執行 CardDemo 應用程式
- 故障診斷

部署流程圖

下圖顯示工作流程中在 NTT DATA Unikix 執行時間部署應用程式的每個步驟。



先決條件

- 遵循 中的指示，使用 NTT DATA 進行複寫使用 NTT DATA UniKix Marketplace AMI。
- 將執行個體中繼資料選項 IMDSv2 修改為選用，如 Amazon EC2Amazon EC2 使用者指南中的還原 IMDSv1 的使用所述。

- 從 下載 NTT DATA UniKix 的 CardDemo 執行期元件

  GitHub 儲存庫。
- 以 身分登入 UniKix 執行期 EC2 執行個體ec2-user。
- 使用此連結解壓縮下載的 CardDemo 執行期元件：UniKix_CardDemo_runtime_v1.zip。
  - 解壓縮的目錄應該包含 bin和 migrated_app目錄。
  - 在$HOME目錄下同時移動 bin和 migrated_app 目錄。路徑看起來像 /home/ec2-user。
  - 您的 中應該有下列目錄$HOME：
    - /home/ec2-user/bin
    - /home/ec2-user/migrated_app
  - 使用下列命令在 $HOME/bin 目錄中移動所有檔案：
  - • chmod +x $HOME/bin/*

步驟 1：準備環境

完成先決條件後，第一步是準備要部署 CardDemo 應用程式的環境。

1. 以 身分登入 UniKix 執行期 EC2 執行個體ec2-user。
2. 在 EC2 執行個體中使用下列命令，觀察 AMI 中預先封裝的 UniKix 軟體清單，例如 TPE、BPE 和 COBOL，以及其他來自 NTT DATA UniKix 產品位置的清單：

   ```
   ls -l /opt/software/
   ```
3. 檢查遷移的 CardDemo 應用程式。您將會看到所有原始程式碼，包括 BMS 地圖、COBOL 程式、COBOL 複製手冊和 JCLs。您也可以執行以下操作，找到 BPE 和 TPE 目錄的匯出、CICS 資源定義，以及循序檔案和 VSAM 檔案等遷移資料：

   ```
   ls  $HOME/migrated_app/*/*
   ```
4. 使用下列命令執行create_project指令碼來建立資料夾結構：

   ```
   sh $HOME/bin/create_project
   ```
5. 使用 來源carddemo.env設定檔案來啟用 CardDemo 環境：

   ```
   source $HOME/bin/carddemo.env
   ```

步驟 2：建立 TPE 區域

啟用要部署應用程式的環境後，您需要建立 TPE 區域。

1. 使用 kixregion createRegion命令建立 TPE 區域，該命令需要輸入，例如 $KIXSYS、$JAVA_HOME和 $KIXLICDIR。這些環境變數已在設定檔案中carddemo.env設定。

```
kixregion createRegion $KIXSYS $JAVA_HOME $KIXLICDIR
```

2. 使用 kixregion setAttr命令設定 TPE 區域。

```
kixregion setAttr $KIXSYS server.tx.languages.cobol.enabled true
kixregion setAttr $KIXSYS server.tx.languages.cobol.flavor vcobol
kixregion setAttr $KIXSYS server.tx.languages.cobol.home $VCOBOL
kixregion setAttr $KIXSYS maps.location $PROJECT_ROOT/maps
kixregion setAttr $KIXSYS programs.location $PROJECT_ROOT/loadlib
kixregion setAttr $KIXSYS environment.KIXDATA $KIXDATA
kixregion setAttr $KIXSYS td.jobq.submission.node $EBMHOME
kixregion setAttr $KIXSYS td.jobq.submission.subsys $EBMSYS
```

3. 執行 kixregion createScript命令，產生此 TPE 區域特有的使用者環境檔案。此命令$KIXSYS/bin/userenv會根據 TPE 區域組態建立或更新。

```
kixregion createScript $KIXSYS
```

4. 透過來源使用者環境檔案 () 來啟用 TPE 區域$KIXSYS/bin/userenv。

```
source $KIXSYS/bin/userenv
```

5. 執行 kixinstall2命令來建置 TPE 區域。

```
kixinstall2
```

步驟 3：建立 BPE 節點和子系統

建立 TPE 區域之後，您需要依照下列步驟建立 BPE 節點和子系統。

1. 變更 的擁有權和許可INSTEBM。

```
sudo chown root $INSTEBM
```

```
sudo chmod 4755 $INSTEBM
```

2. 使用 INSTEBM命令建立 BPE 節點。BPE 節點目錄做為輸入參數提供。

```
$INSTEBM $EBMHOME
```

3. 透過從新建立的 BPE 節點取得batchenv檔案來啟用批次環境。

```
source $EBMHOME/batchenv
```

4. 使用 Batch Administration Manager (bam) 在此節點內建立 BPE 子系統。bam 命令會開啟 Batch Administration Manager 介面。

```
bam
```

   a. 使用 BAM 介面啟動 BPE 節點。從主功能表中選擇選項 2，系統環境。

```
                Batch Administration Manager                2024/09/17 21:25:56




            1   Software License Management
            2   System Environments
            3   Applications & Subsystems
            4   Security & Users
            5   Classes & Activities
            6   Problem Determination



            H - Help
            Q - Quit

--------------------------------------------------------------------------------
Type an option and press Return: 2
```

   b. 選擇選項 2，開始/（停止）批次節點以啟動 BPE 節點。

```
                System Environments                          2024/09/17 21:27:03




             1   Report System Status
             2   Start/(Stop) Batch Node
             3   Assign the Console
             4   Change the Date
             5   Redirect Job Output
             6   Inter-Node Communications
             7   Job Accounting
             8   Assign the Initial Job Number
             9   Enable/Disable Duplicate Name Execution Delay : Enabled

             H - Help
             R - Return to Main Menu

--------------------------------------------------------------------------------
Type an option and press Return: 2
```

c. 開始後，按兩次 Return 鍵以返回 BAM 主功能表。

```
                System Environments: Start Batch Node        2024/09/17 21:28:28









              Batch Node Startup Completed.









--------------------------------------------------------------------------------
Press Return to Continue
```

d. 若要建立 BPE 子系統，請選擇選項 3，應用程式和子系統。

```
                Batch Administration Manager                2024/09/17 21:29:03




             1   Software License Management
             2   System Environments
             3   Applications & Subsystems
             4   Security & Users
             5   Classes & Activities
             6   Problem Determination



             H - Help
             Q - Quit

--------------------------------------------------------------------------------
Type an option and press Return: 3
```

e. 然後選擇選項 3，建立子系統。

```
                Applications & Subsystems                   2024/09/17 21:29:57




             1   List All Subsystems
             2   Query a Subsystem
             3   Create a Subsystem
             4   Update a Subsystem
             5   Delete a Subsystem
             6   Change/Show Default Subsystem
             7   Import a Subsystem (from another batch node)
             8   Create a BPESUB Project
             9   Export Subsystem bldsub Config File

             H - Help
             R - Return to Main Menu

--------------------------------------------------------------------------------
Type an option and press Return: 3
```

f. 將子系統名稱輸入為 `sys1`。

```
                Applications & Subsystems: Create          2024/09/17 21:30:22









         No Subsystems are currently defined.









             R - Return to Previous Screen

--------------------------------------------------------------------------------
Enter the Subsystem's name you want to create: sys1
```

g. 選擇選項 3，資料管理。

```
                Applications & Subsystems: Create          2024/09/17 21:30:53



             D   Display Current sys1 Subsystem's Configuration
             S   Set to Default Subsystem Configuration

             1   Application Languages
             2   Database Management System (DBMS)
             3   Data Management
             4   Optional Packages
             5   Date/Time Management
             6   User-Specific Objects
             7   Configuration Options

             C - Create sys1 Subsystem
             H - Help
             R - Return to Main Menu

--------------------------------------------------------------------------------
Type an option and press Return: 3
```

h. 選擇選項 5，因為 CardDemo 應用程式同時涉及序列和 VSAM 檔案。

```
                Applications & Subsystems: Data Management  2024/09/17 21:31:49


              1   No TPE VSAM Data Management        < Default >

              2   TPE VSAM "RELATIVE/INDEXED" Files

              3   TPE VSAM "RELATIVE", COBOL "INDEXED" Files

              4   COBOL "RELATIVE", TPE VSAM "INDEXED" Files

         ->   5   COBOL "RELATIVE/INDEXED/SEQUENTIAL" Files
                              and
                        TPE VSAM "RRDS/KSDS/ESDS" Files


              R - Return to Create Menu


--------------------------------------------------------------------------------
Type an option and press Return:
```

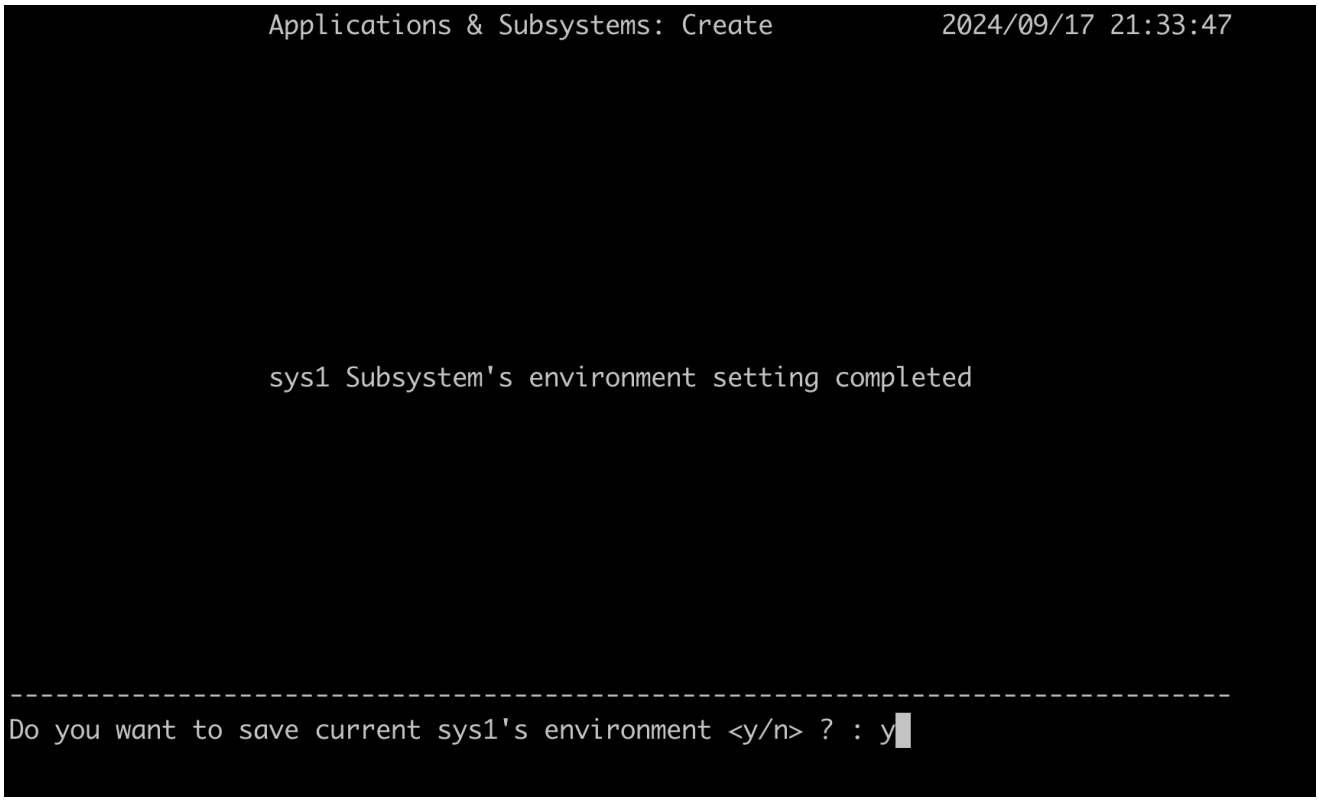
i. (選用)。按 "R" 返回建立選單頁面，檢閱可用的不同組態選項。

j. 在建立頁面上，輸入 "C" 來建立子系統 sys1。

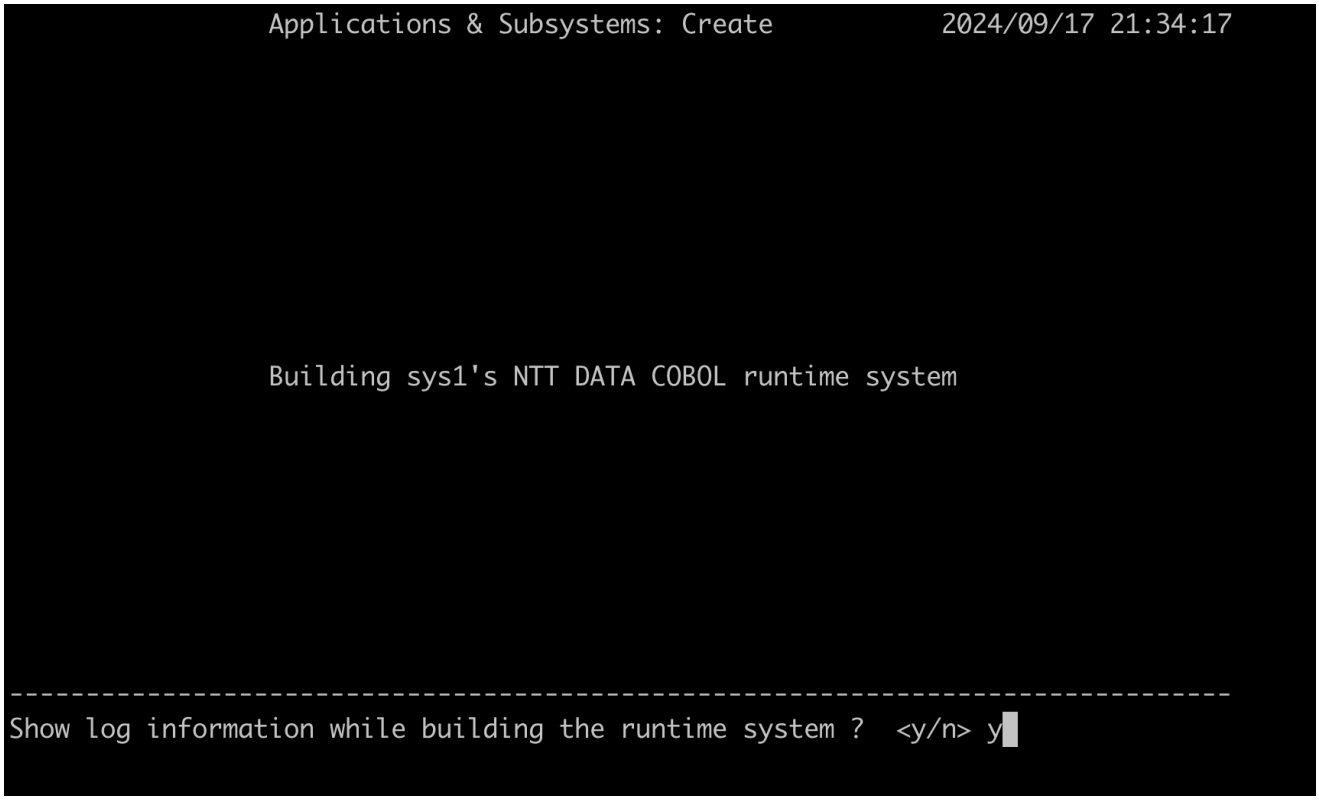
```
                Applications & Subsystems: Create          2024/09/17 21:32:37



            D   Display Current sys1 Subsystem's Configuration
            S   Set to Default Subsystem Configuration

            1   Application Languages
            2   Database Management System (DBMS)
            3   Data Management
            4   Optional Packages
            5   Date/Time Management
            6   User-Specific Objects
            7   Configuration Options

            C - Create sys1 Subsystem
            H - Help
            R - Return to Main Menu

--------------------------------------------------------------------------------
Type an option and press Return: C
```

k. 檢閱設定，然後輸入 "C" 以繼續其餘的環境設定。由於carddemo.env設定檔案中定義的必要環境變數和建議的資料夾結構已就位，這些環境設定會預先填入。

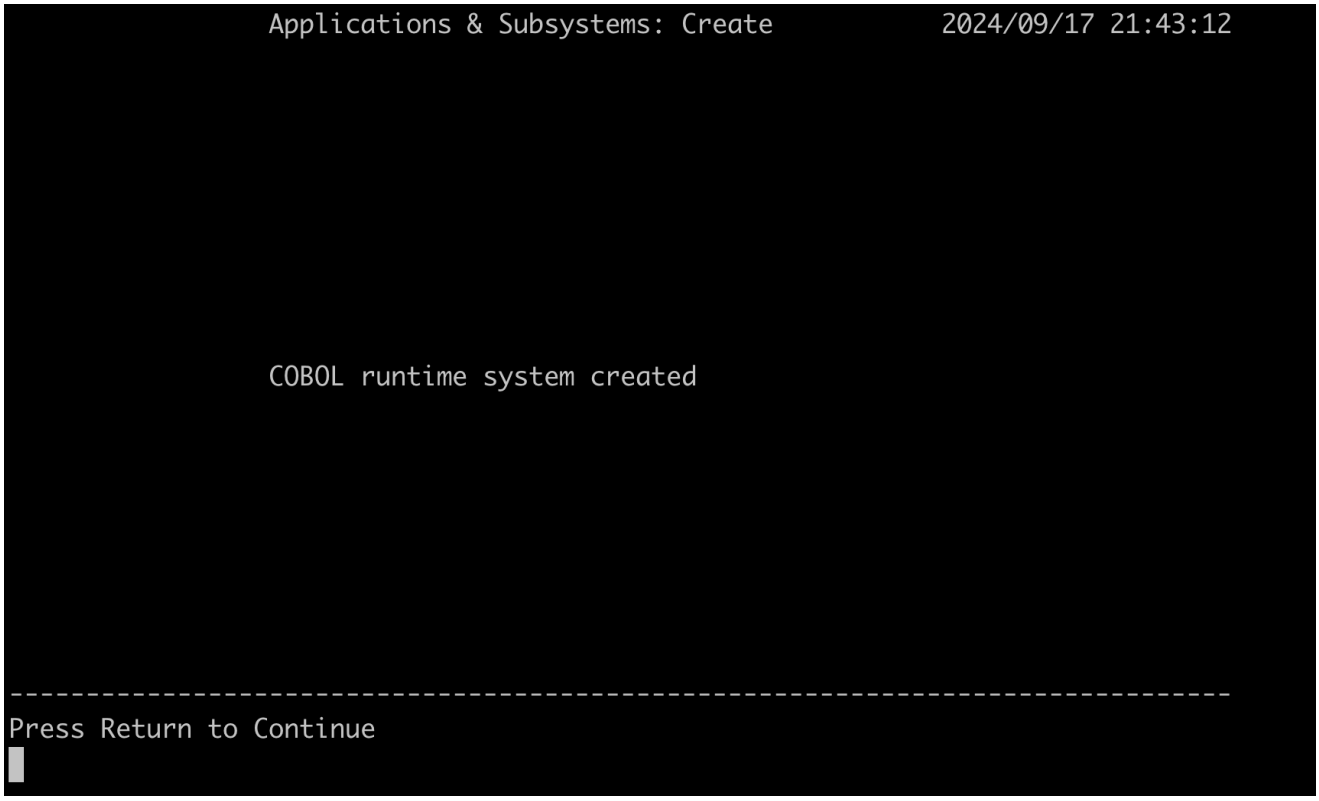
l. 輸入「y」以確認並儲存目前的環境設定。



m. 輸入「y」以在建置子系統時顯示日誌。



n. 按傳回鍵，直到您返回主功能表，然後選取結束選項以結束 BAM 介面。



```
                Applications & Subsystems: Create          2024/09/17 21:43:55


         Subsystem sys1 created. Configuration updated.

         o  To set this Subsystem's environment you must source the node
            batchenv file passing sys1 as the Subsystem argument.
            For example, after exiting BAM, type:

            .  /home/ec2-user/unikixdemo/bpenode/batchenv sys1

         o  To customize the Subsystem's environment select
            "Update a Subsystem".




--------------------------------------------------------------------------------
Press Return to Continue
```

```
                Batch Administration Manager               2024/09/17 21:47:40




             1   Software License Management
             2   System Environments
             3   Applications & Subsystems
             4   Security & Users
             5   Classes & Activities
             6   Problem Determination



             H - Help
             Q - Quit

--------------------------------------------------------------------------------
Type an option and press Return: Q
```

5. 透過使用子系統名稱 來源 來啟用 BPE batchenv子系統sys1。

```
source $EBMHOME/batchenv sys1
```

步驟 4：編譯和部署 CardDemo 應用程式

在此步驟中，您將編譯 COBOL 程式並部署應用程式成品，例如 JCL、程序、資料檔案和 CICS 資源定義。

1. 透過來源carddemo.env設定檔案再次啟用 CardDemo 環境。

```
source $HOME/bin/carddemo.env
```

2. 導覽至 COBOL 來源目錄。

```
cd $MIGAPP_DIR/cbl
```

3. CBACT01C.cbl 使用compile指令碼編譯 Cobol 程式。

```
compile CBACT01C.cbl
```

4. 使用compile.all指令碼編譯所有 Cobol 程式。

```
compile.all
```

5. 導覽至 BMS 映射來源目錄。

```
cd $MIGAPP_DIR/bms
```

6. COACTUP.bms 使用compbms指令碼編譯 BMS 映射。

```
compbms COACTUP.bms
```

7. 使用compbms.all指令碼編譯所有 BMS 映射。

```
compbms.all
```

8. 驗證 COBOL 和 BMS 映射的編譯二進位檔。

```
ls $PROJECT_ROOT/loadlib
ls $PROJECT_ROOT/maps
```

9. 使用deploy_app指令碼部署其他應用程式成品，例如 JCL、程序、資料檔案和 CICS 資源定義。

```
deploy_app
```

10. 導覽至專案 JCL 目錄。

```
cd $PROJECT_ROOT/jcl
```

11. 將 JCL ACCTFILE 轉換為 BPE JCL Macro。使用 mvstrans命令，利用「-v」選項進行 JCL 驗證，並使用「-f」選項來建立巨集。

```
mvstrans ACCTFILE -v
mvstrans ACCTFILE -f
```

12. 將 JCL 程序 REPROC 轉換為 BPE JCL 程序巨集。使用 mvstrans命令搭配 "-p" 選項以及用於驗證的 "-v" 選項，以及用於建立巨集的 "-f" 選項。

```
mvstrans REPROC -v -p
mvstrans REPROC -f -p
```

13. 翻譯所有 JCLs和 JCL 程序。

```
for file in "./jmvs/*"; do mvstrans $file -f; done > jmvs.out
for file in "./mvsp/*"; do mvstrans $file -p -f; done > mvsp.out
```

步驟 5：匯入 BPE 和 TPE 目錄

在此步驟中，您會使用不同的命令匯入 BPE 和 TPE 目錄。

1. 使用 loadcat命令匯入 BPE 目錄。

```
loadcat $MIGAPP_DIR/catlg/bpe/BPECAT*
```

2. 導覽至 $KIXSYS 目錄。

```
cd $KIXSYS
```

3. 使用 kiximpcat命令匯入 TPE 目錄。

```
kiximpcat -c CATALOG -l CATALOG.lst
```

4. 使用 kiximptbl 命令匯入 CICS 資源定義。

```
kiximptbl
```

步驟 6：啟動並連接 TPE 與 BPE

在此步驟中，您需要啟動先前建立的 TPE 區域與 BPE 管理員，並連接這些區域，才能執行範例 CardDemo 應用程式。

1. 針對所有 VSAM 檔案執行 `kixverify`命令，以確保它們已重設，且任何先前開啟的檔案都已關閉。

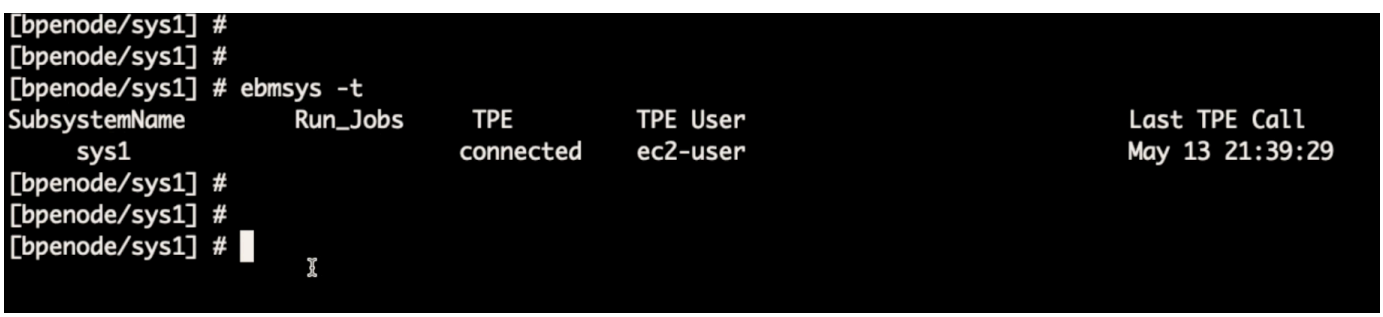
```
kixverify -r ALL
```

2. 啟動 TPE 區域。

```
kixregion start $KIXSYS
```

3. 確保 BPE 和 TPE 都已連接。這很重要，因為 VSAM 檔案由 TPE 擁有，存取 VSAM 的任何批次操作都需要連線至 TPE。

```
ebmsys -t
```



步驟 7：執行 CardDemo 應用程式

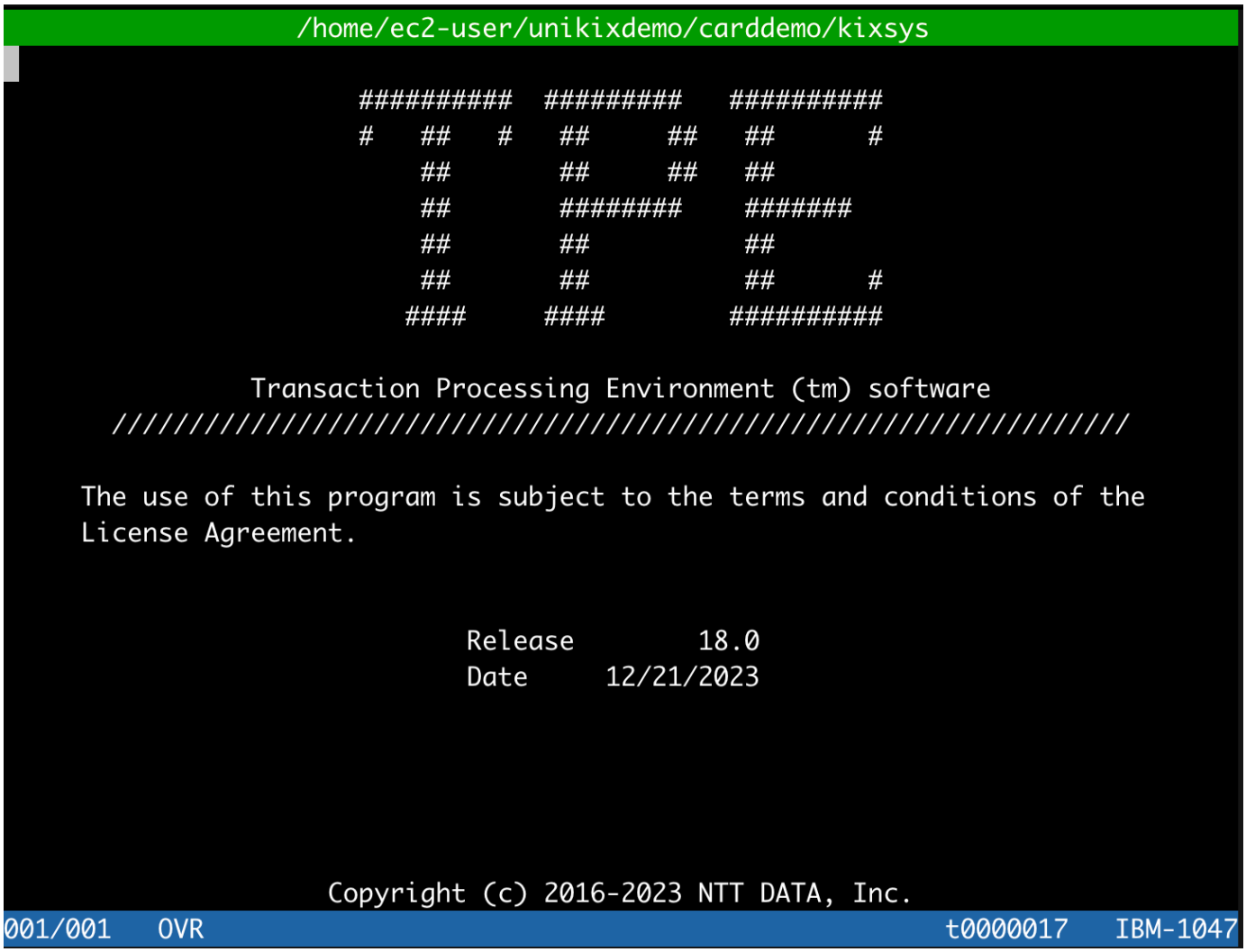
在此步驟中，您會在 TN3270 終端機模擬器中執行 CardDemo 應用程式。

UniKix 執行期 AMI 隨附 TN3270 終端機模擬器，您可以直接從 UniKix EC2 執行個體啟動。

使用 TN3270 終端機模擬器連線至 TPE

- 使用 `kixterm`命令啟動 TN3270 終端機。

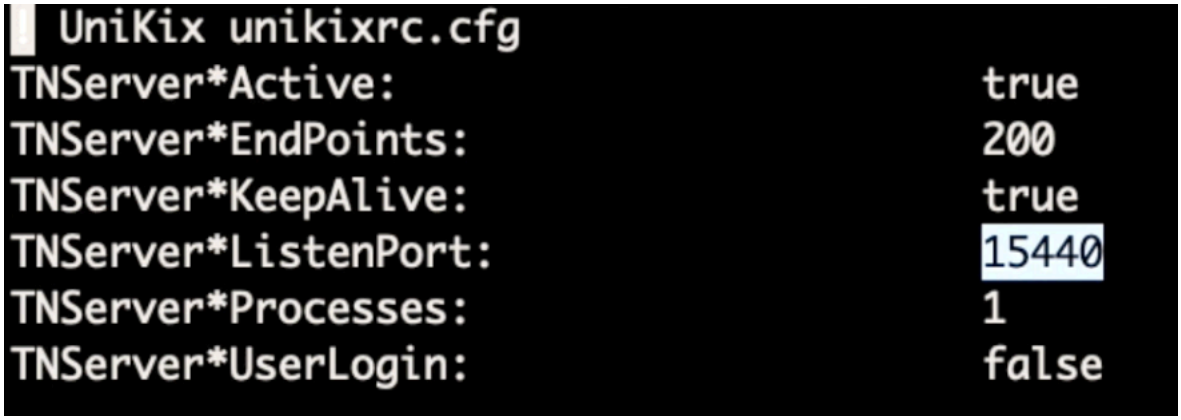
```
kixterm
```



(選用)。如果您想要使用自己的終端機模擬器：

1. 從 Amazon EC2 主控台取得 UniKix 執行時間執行個體的 IP 地址。
2. 取得連接埠號碼，以使用 TN3270 終端機模擬器連線至 TPE 區域。您可以從 unikixrc.cfg 檔案的 TNServer ListenPort 找到此項目。

```
cat $KIXSYS/unikixrc.cfg
```

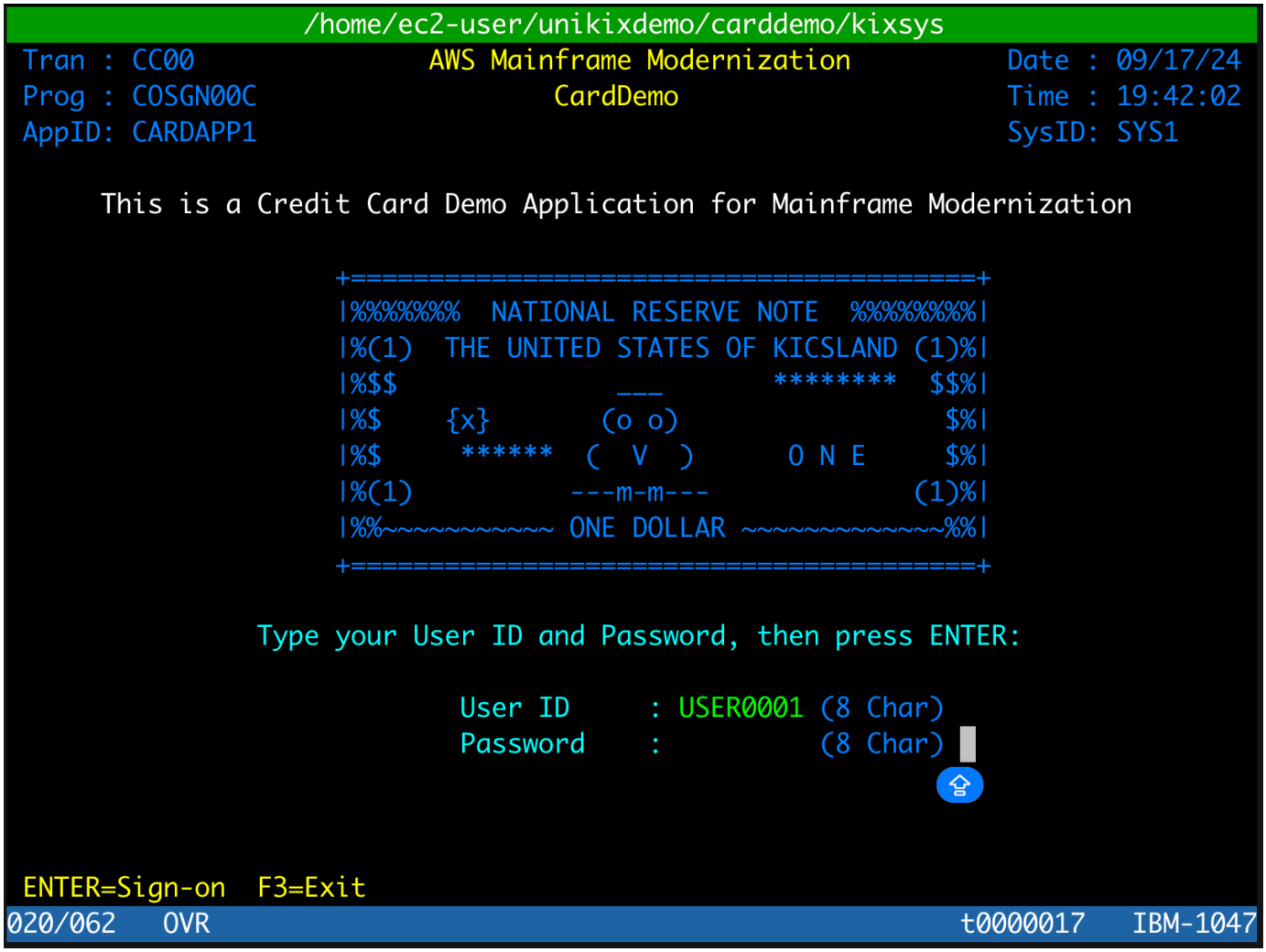


3. 將 TN3270 終端機模擬器設定為使用 UniKix 執行時間執行個體的 IP 地址和連接埠號碼 15440。

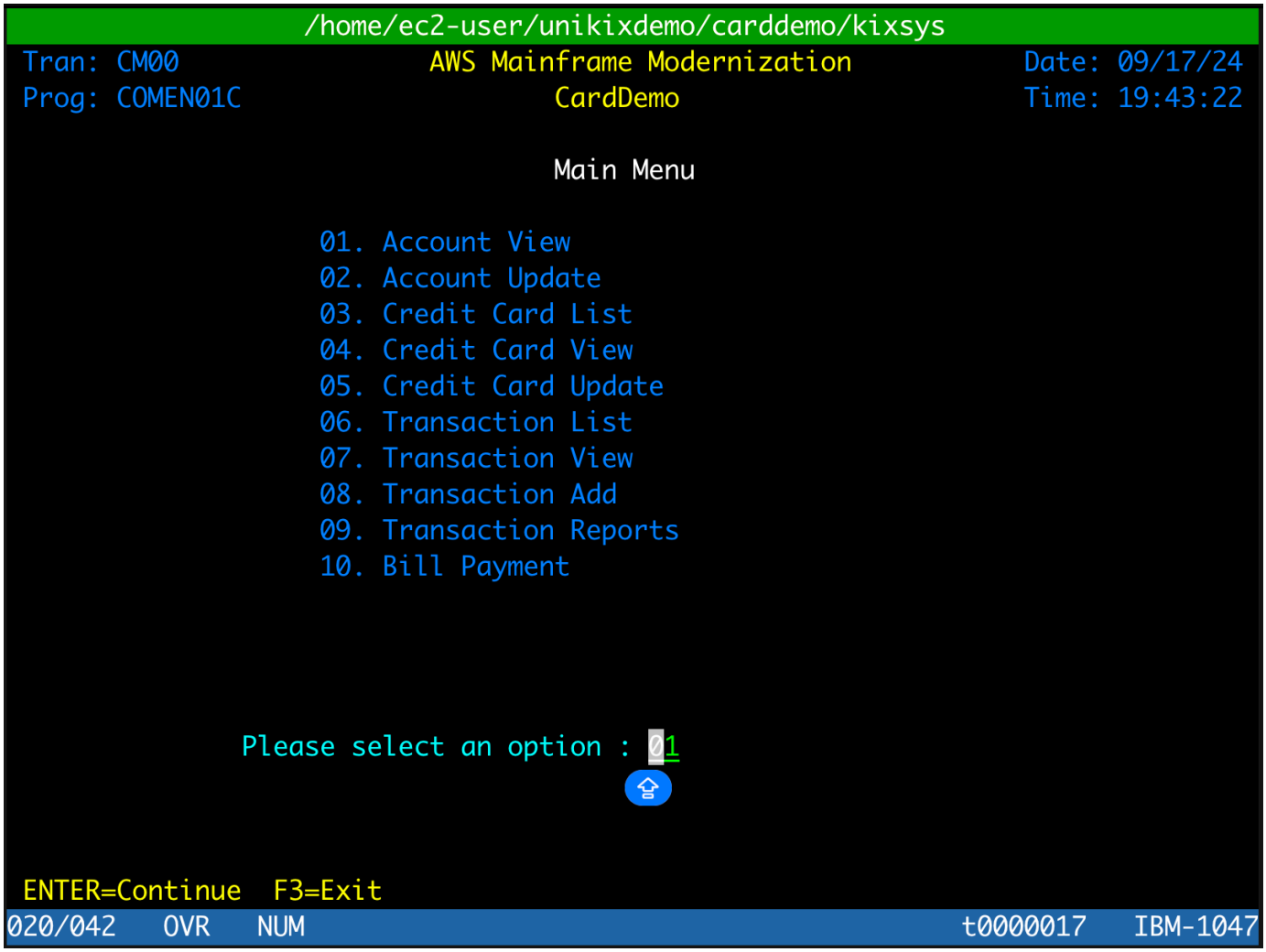
線上交易

本節假設您已使用 `kixterm`命令連線至 TN3270 終端機模擬器。

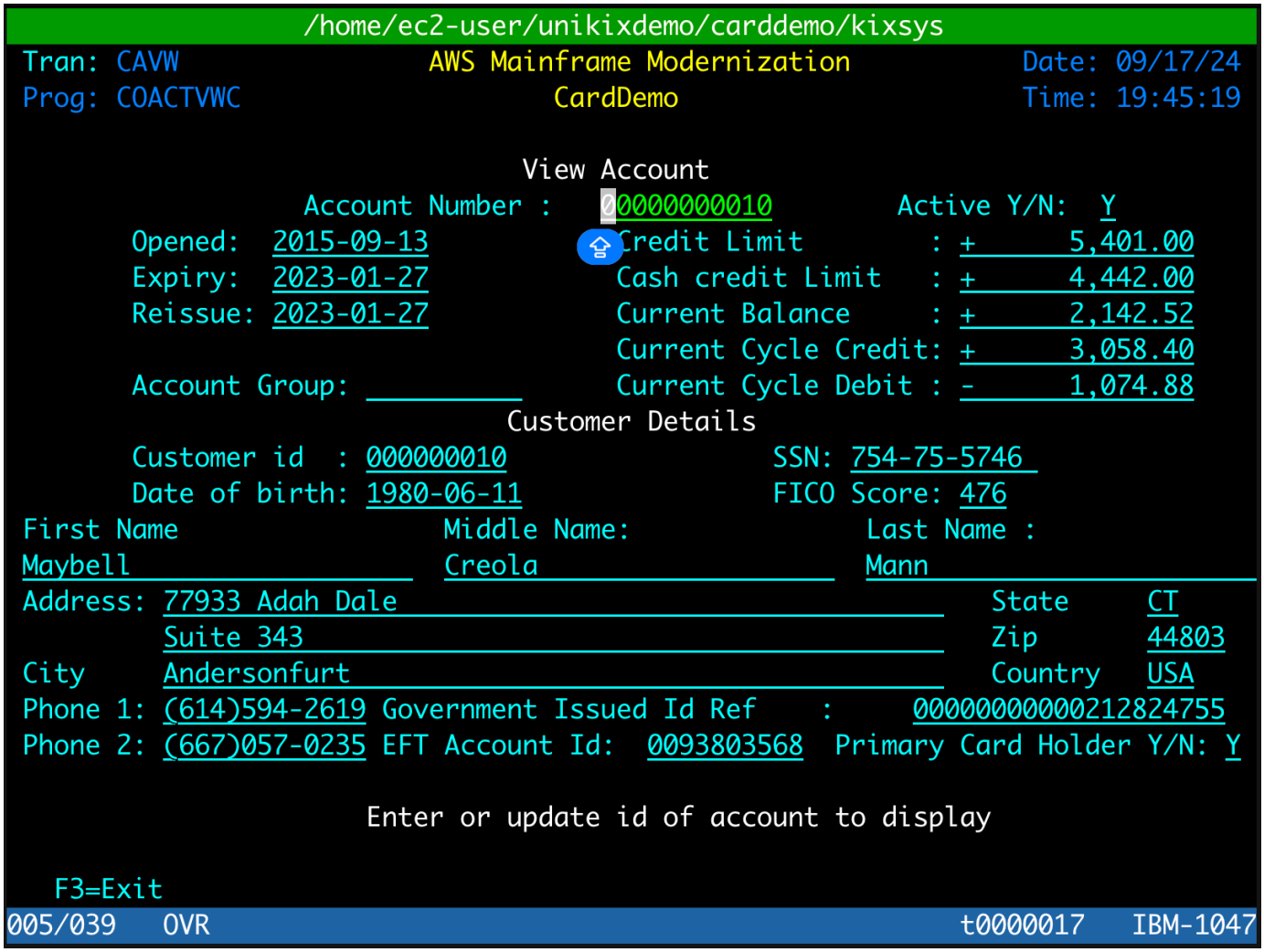
1. 從 TN3270 終端機模擬器連線後，按「Enter」鍵清除 TPE 畫面並輸入初始交易。
2. 在初始交易 CC00 （登入畫面） 上，輸入`USER001`使用者名稱`PASSWORD`和密碼。



3. 從主功能表中選擇選項 “01” 以檢視帳戶。



4. 在檢視帳戶畫面中，輸入帳戶號碼（例如 00000000010)。您應該會看到從遷移資料填入的帳戶資訊。



5. 按兩次「PF03」鍵返回登入畫面，然後按「Ctrl+C」(Windows) 或「Cmd+C」(Macbook) 結束 TN3270 終端機。

批次任務

1. 導覽至 JCL 目錄。

```
cd $MBMSUB
```

2. 提交任務MFCATGL1並觀察任務日誌輸出。

```
BPESUB READCARD
```

3. 或者，您可以從 $SUBSYS_OUTDIR目錄檢視任務日誌。

```
ls -lrt $SUBSYS_OUTDIR/*
```

您現在已成功將 CardDemo 應用程式部署到 NTT DATA UniKix 執行時間，並透過瀏覽幾個 CICS 線上畫面和批次任務來驗證執行中的應用程式。

故障診斷

以下是您在設定 CardDemo 應用程式時可能發現的一些常見錯誤。

錯誤：授權錯誤

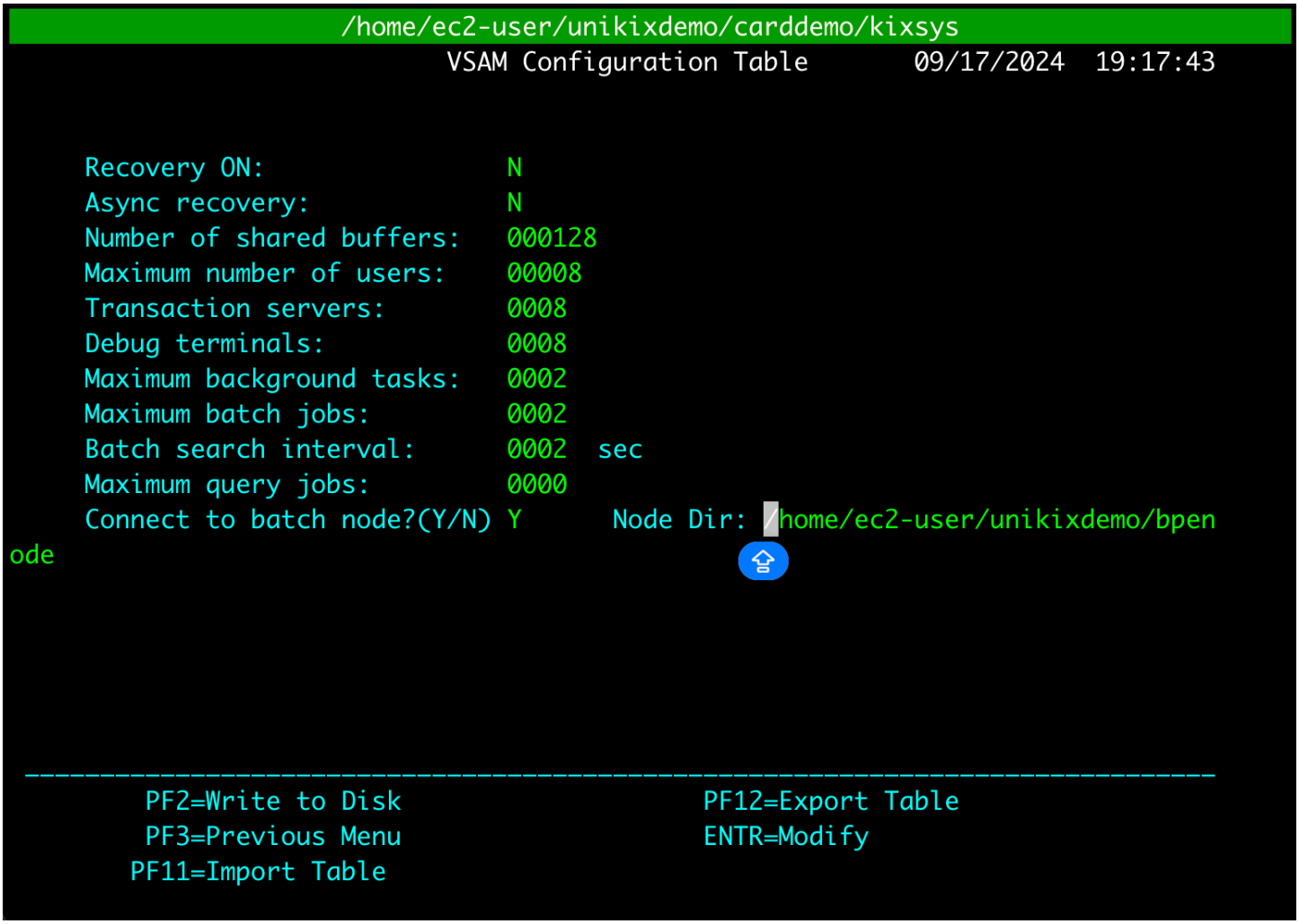
如果您在遵循本教學課程期間收到授權失敗錯誤，可能是您的 ECIMDSv22。 EC2 您可以透過將執行個體中繼資料選項 IMDSv2 修改為選用來解決此問題，如《Amazon EC2 使用者指南》中的還原 IMDSv1 的使用中所述。

錯誤：TPE 未連線至 BPE

如果 TPE 未連線至 BPE，請確定 VSAM 組態表已使用 BPE 節點目錄正確設定。若要存取 VSAM 組態表，請使用下列命令啟動 TN3270 終端機模擬器：

```
kixterm
```

1. 將交易名稱輸入為 CTBL。
2. 在資料表管理員功能表中，選擇標準資料表選項。
3. 在標準資料表畫面上，選擇選項 VSAM 組態資料表。
4. 檢查連線至批次節點？ 是否設定為「Y」且節點目錄是否正確。


/home/ec2-user/unikixdemo/carddemo/kixsys
VSAM Configuration Table 09/17/2024 19:17:43
Recovery ON: N
Async recovery: N
Number of shared buffers: 000128
Maximum number of users: 00008
Transaction servers: 0008
Debug terminals: 0008
Maximum background tasks: 0002
Maximum batch jobs: 0002
Batch search interval: 0002 sec
Maximum query jobs: 0000
Connect to batch node?(Y/N) Y Node Dir: /home/ec2-user/unikixdemo/bpen
ode
PF2=Write to Disk
PF3=Previous Menu
PF11=Import Table
PF12=Export Table
ENTR=Modify

AWS 大型主機現代化的安全性

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

安全性是 AWS 與您之間的共同責任。共同責任模型將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 Cloud 中執行 AWS 服務的基礎設施。 AWS 也為您提供可安全使用的服務。在AWS 合規計劃中，第三方稽核人員會定期測試和驗證我們安全的有效性。若要了解適用於 AWS Mainframe Modernization 的合規計劃，請參閱合規計劃的 AWS 服務範圍。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。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 AWS Mainframe Modernization 時套用共同責任模型。它說明如何設定 AWS Mainframe Modernization 以符合您的安全與合規目標。您也會了解如何使用其他 AWS 服務來協助您監控和保護 AWS 大型主機現代化資源。

AWS Mainframe Modernization 提供自己的 IAM 保護資源 （應用程式、環境、部署等），這是 AWS Mainframe Modernization 管理資源，IAM 政策必須對其允許任何動作。

AWS Mainframe Modernization for replatforming 也受到 IAM 保護。IAM 也會透過標準 IAM 政策，針對衍生自原始大型主機環境的已定義資源的特定動作，授予或拒絕委託人許可。當應用程式在受保護的資源上嘗試此類動作時， AWS 大型主機現代化重建執行期會呼叫 IAM 授權服務。IAM 會根據標準 IAM 政策評估機制傳回允許或拒絕。

目錄

- AWS 大型主機現代化中的資料保護
- AWS 適用於大型主機現代化的 Identity and Access Management
- AWS Mainframe Modernization 的合規驗證
- AWS 大型主機現代化中的彈性
- 中的基礎設施安全 AWS Mainframe Modernization
- AWS Mainframe Modernization 使用 AWS PrivateLink 介面端點存取

AWS 大型主機現代化中的資料保護

AWS 共同責任模型適用於 AWS Mainframe Modernization 中的資料保護。如此模型所述， AWS 負責保護執行所有 的 全球基礎設施 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，我們建議您保護 AWS 帳戶 登入資料，並使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 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。如需有關使用 CloudTrail 追蹤擷取 AWS 活動的資訊，請參閱AWS CloudTrail 《 使用者指南》中的使用 CloudTrail 追蹤。
- 使用 AWS 加密解決方案，以及其中的所有預設安全控制 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 AWS 透過命令列界面或 API 存取 時需要 FIPS 140-3 驗證的密碼編譯模組，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) 140-3。

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 AWS Mainframe Modernization 或使用 AWS 服務 主控台、API AWS CLI或其他 AWS SDKs 時。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。

AWS Mainframe Modernization 收集的資料

AWS Mainframe Modernization 會收集多種類型的資料：

- `Application configuration`：這是您建立用來設定應用程式的 JSON 檔案。它包含您對 AWS 大型主機現代化提供的不同選項的選擇。檔案也包含相依 AWS 資源的資訊，例如存放應用程式成品的 Amazon Simple Storage Service 路徑，或存放資料庫登入 AWS Secrets Manager 資料的 Amazon Resource Name (ARN)。

- `Application executable (binary)`：這是您編譯的二進位檔，而且您想要在 AWS Mainframe Modernization 上部署。
- `Application JCL or scripts`：此原始程式碼會代表您的應用程式管理批次任務或其他處理。
- `User application data`：當您匯入資料集時， AWS Mainframe Modernization 會將它們存放在關聯式資料庫中，讓您的應用程式可以存取它們。
- `Application source code`：透過 Amazon AppStream 2.0， AWS Mainframe Modernization 提供您撰寫和編譯程式碼的開發環境。

AWS Mainframe Modernization 會以原生方式存放此資料 AWS。我們從您收集的資料會存放在 AWS 大型主機現代化管理的 Amazon S3 儲存貯體中。當您部署應用程式時， AWS Mainframe Modernization 會將資料下載到 Amazon Elastic Block Store 支援的 Amazon Elastic Compute Cloud 執行個體。觸發清除時，資料會從 Amazon EBS 磁碟區和 Amazon S3 中移除。Amazon EBS 磁碟區為單一租用戶，這表示一個客戶使用一個執行個體。執行個體永遠不會共用。當您刪除執行時間環境時，也會刪除 Amazon EBS 磁碟區。當您刪除應用程式時，成品和組態會從 Amazon S3 中刪除。

應用程式日誌存放在 Amazon CloudWatch 中。客戶應用程式日誌訊息也會匯出至 CloudWatch。CloudWatch 日誌可能包含客戶敏感資料，例如業務資料或偵錯訊息中的安全資訊）。如需詳細資訊，請參閱使用 Amazon CloudWatch 監控 AWS 大型主機現代化。

此外，如果您選擇將一或多個 Amazon Elastic File System 或 Amazon FSx 檔案系統連接到您的執行時間環境，這些系統內的資料將儲存在其中 AWS。如果您決定停止使用檔案系統，則需要清除該資料。

當您將資料放入 AWS 大型主機現代化用於應用程式部署和資料集匯入的 Amazon S3 儲存貯體時，您可以使用所有可用的 Amazon S3 Amazon S3加密選項來保護資料。此外，如果您將這些檔案系統連接到執行期環境，則可以使用 Amazon EFS 和 Amazon FSx 加密選項。

AWS Mainframe Modernization 服務靜態資料加密

AWS Mainframe Modernization 與 整合 AWS Key Management Service ，可在永久存放資料的所有相依資源上提供透明的伺服器端加密 (SSE)，也就是 Amazon Simple Storage Service、Amazon DynamoDB 和 Amazon Elastic Block Store。 AWS Mainframe Modernization 會為您建立和管理 中的對稱加密 AWS KMS 金鑰 AWS KMS。

依預設加密靜態資料，有助於降低保護敏感資料所涉及的營運開銷和複雜性。同時，它可讓您遷移需要嚴格加密合規性和法規要求的應用程式。

您無法在建立執行時間環境和應用程式時停用此加密層或選取替代加密類型。

您可以將自己的客戶受管金鑰用於 AWS 大型主機現代化應用程式和執行期環境，以加密 Amazon S3 和 Amazon EBS 資源。

對於您的 AWS 大型主機現代化應用程式，您可以使用此金鑰來加密應用程式定義和其他應用程式資源，例如 JCL 檔案，這些檔案儲存在服務帳戶中建立的 Amazon S3 儲存貯體中。如需詳細資訊，請參閱建立應用程式 。

針對 AWS 您的 Mainframe Modernization 執行期環境， AWS Mainframe Modernization 會使用客戶受管金鑰來加密其建立的 Amazon EBS 磁碟區，並將其連接至您的 AWS Mainframe Modernization Amazon EC2 執行個體，該執行個體也位於服務帳戶中。如需詳細資訊，請參閱建立執行時間環境。

> **Note**
>
> DynamoDB 資源一律使用 AWS Mainframe Modernization 服務帳戶中 AWS 受管金鑰 的 進行加密。您無法使用客戶受管金鑰加密 DynamoDB 資源。

AWS Mainframe Modernization 會將您的客戶受管金鑰用於下列任務：

- 重新部署應用程式。
- 取代 AWS 大型主機現代化 Amazon EC2 執行個體。

AWS Mainframe Modernization 不會使用您的客戶受管金鑰來加密為支援 AWS Mainframe Modernization 應用程式而建立的 Amazon Relational Database Service 或 Amazon Aurora 資料庫、Amazon Simple Queue Service 佇列和 Amazon ElastiCache 快取，因為這些快取都不包含客戶資料。

如需更多資訊，請參閱 AWS Key Management Service 開發人員指南中的客戶受管金鑰。

下表摘要說明 AWS Mainframe Modernization 如何加密您的敏感資料。

資料類型	AWS 受管金鑰 加密	客戶受管金鑰加密
`Definition` 包含特定應用程式的定義。	已啟用	已啟用
`EnvironmentSummary`	已啟用	已啟用

資料類型	AWS 受管金鑰 加密	客戶受管金鑰加密
包含執行時間環境的相關資訊。		
`ApplicationSummary` 包含 AWS Mainframe Modernization 應用程式的相關資訊。	已啟用	已啟用
`DeploymentSummary` 包含 AWS 大型主機現代化應用程式部署的相關資訊。	已啟用	已啟用

> **Note**
>
> AWS Mainframe Modernization 會使用 自動啟用靜態加密 AWS 受管金鑰 ，以免費保護您的敏感資料。不過，使用客戶受管金鑰會產生 AWS KMS 費用。如需定價的詳細資訊，請參閱 AWS Key Management Service 定價。

如需 的詳細資訊 AWS KMS，請參閱 AWS Key Management Service。

AWS Mainframe Modernization 如何在 中使用授予 AWS KMS

AWS Mainframe Modernization 需要授予才能使用客戶受管金鑰。

當您建立應用程式或執行時間環境，或使用客戶受管金鑰在 AWS Mainframe Modernization 中部署應用程式時， AWS Mainframe Modernization 會透過傳送 CreateGrant 請求來代表您建立授予 AWS KMS。中的授予 AWS KMS 用於授予 AWS Mainframe Modernization 存取客戶帳戶中 KMS 金鑰的權限。

AWS Mainframe Modernization 需要授予 ，才能將客戶受管金鑰用於下列內部操作：

- 將 DescribeKey 請求傳送至 AWS KMS ，以驗證在建立應用程式、執行時間環境或應用程式部署時輸入的對稱客戶受管金鑰 ID 是否有效。

- 將 GenerateDataKey 請求傳送至 AWS KMS ，以加密連接至託管 AWS 大型主機現代化執行期環境之 Amazon EC2 執行個體的 Amazon EBS 磁碟區。
- 傳送解密請求至 AWS KMS ，以解密 Amazon EBS 上的加密內容。

AWS Mainframe Modernization 使用 AWS KMS 授權來解密存放在 Secrets Manager 中的秘密，並在建立執行期環境、建立或重新部署應用程式，以及建立部署時解密秘密。 AWS Mainframe Modernization 建立的授予支援下列操作：

- 建立或更新執行時間環境授權：
  - 解密
  - 加密
  - ReEncryptFrom
  - ReEncryptTo
  - GenerateDataKey
  - DescribeKey
  - CreateGrant
- 建立或重新部署應用程式授權：
  - GenerateDataKey
- 建立部署授權：
  - 解密

您可以隨時撤銷授予的存取權，或移除服務對客戶受管金鑰的存取權。如果您這麼做， AWS 大型主機現代化將無法存取客戶受管金鑰加密的任何資料，這會影響依賴資料的操作。例如，如果 AWS Mainframe Modernization 嘗試存取由客戶受管金鑰加密的應用程式定義，而不授予該金鑰，則應用程式建立操作會失敗。

AWS Mainframe Modernization 會收集使用者應用程式組態 (JSON 檔案） 和成品 （ 二進位檔和可執行檔）。它也會建立中繼資料，以追蹤用於 AWS 大型主機現代化操作的各種實體，並建立日誌和指標。客戶可見的日誌和指標包括：

- 反映應用程式和執行期引擎的 CloudWatch 日誌 (AWS Blu Age 或 Rocket Software （ 先前稱為 Micro Focus))。
- 操作儀表板的 CloudWatch 指標。

此外，AWS Mainframe Modernization 會收集有關服務的計量、活動報告等用量資料和指標。此資料不可見於客戶。

AWS Mainframe Modernization 會根據資料類型，將此資料存放在不同的位置。您上傳的客戶資料會存放在 Amazon S3 儲存貯體中。服務資料同時儲存在 Amazon S3 和 DynamoDB 中。部署應用程式時，您的資料和服務資料都會下載到 Amazon EBS 磁碟區。如果您選擇將 Amazon EFS 或 Amazon FSx 儲存連接到執行期環境，則存放在這些檔案系統中的資料也會下載到 Amazon EBS 磁碟區。

預設為靜態加密。您無法停用或變更它。目前，您也無法變更其組態。

建立客戶受管金鑰

您可以使用 AWS Management Console 或 AWS KMS APIs 來建立對稱客戶受管金鑰。

建立對稱客戶受管金鑰

請依照《AWS Key Management Service 開發人員指南》中建立對稱客戶受管金鑰的步驟進行。

金鑰政策

金鑰政策會控制客戶受管金鑰的存取權限。每個客戶受管金鑰都必須只有一個金鑰政策，其中包含決定誰可以使用金鑰及其使用方式的陳述式。在建立客戶受管金鑰時，可以指定金鑰政策。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的管理客戶受管金鑰的存取。

若要將客戶受管金鑰與 AWS Mainframe Modernization 資源搭配使用，必須在金鑰政策中允許下列 API 操作：

- `kms:CreateGrant`：新增客戶受管金鑰的授權。授予控制對指定 KMS 金鑰的存取權，以允許存取授予大型主機現代化所需的操作 AWS 。如需使用授權的詳細資訊，請參閱 AWS Key Management Service 開發人員指南。

  這可讓 AWS 大型主機現代化執行下列動作：

  - 呼叫 `GenerateDataKey` 以產生加密的資料金鑰並加以儲存，因為資料金鑰不會立即用來加密。
  - 呼叫 `Decrypt` 以使用儲存的加密資料金鑰來存取加密的資料。
  - 設定淘汰委託人以允許 服務至 `RetireGrant`。
- `kms:DescribeKey` – 提供客戶受管金鑰詳細資訊，以允許 AWS Mainframe Modernization 驗證金鑰。

AWS Mainframe Modernization 需要客戶金鑰政策中的 `kms:CreateGrant`和 `kms:DescribeKey` 許可。 AWS Mainframe Modernization 使用此政策為自己建立授予。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "Enable IAM User Permissions",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::AccountId:role/ExampleRole"
        },
        "Action": [
            "kms:CreateGrant",
            "kms:DescribeKey"
        ],
        "Resource": "*"
    }]
}
```

> **Note**
>
> Principal 上述範例中顯示的 角色是您用於 AWS 大型主機現代化操作的角色，例如 CreateApplication和 CreateEnvironment。

如需有關在政策中指定許可的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》。

如需有關故障診斷金鑰存取的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》。

為 AWS 大型主機現代化指定客戶受管金鑰

您可以為下列資源指定客戶受管金鑰：

- 應用程式
- 環境

建立資源時，您可以輸入 KMS ID 來指定金鑰， AWS 而 Mainframe Modernization 會使用該 ID 來加密資源存放的敏感資料。

- KMS ID — 客戶受管金鑰的金鑰識別符。輸入金鑰 ID、金鑰 ARN、別名名稱或別名 ARN。

您可以使用 AWS Management Console 或 指定客戶受管金鑰 AWS CLI。

若要在 中建立執行期環境時指定您的客戶受管金鑰 AWS Management Console，請參閱 建立 AWS 大型主機現代化執行時間環境。若要在 中建立應用程式時指定您的客戶受管金鑰 AWS Management Console，請參閱 建立 AWS Mainframe Modernization 應用程式。

若要在使用 建立執行期環境時新增客戶受管金鑰 AWS CLI，請指定 kms-key-id 參數，如下所示：

```
aws m2 create-environment –engine-type microfocus –instance-type M2.m5.large
--publicly-accessible –engine-version 7.0.3 –name test
--high-availability-config desiredCapacity=2
--kms-key-id myEnvironmentKey
```

若要在使用 建立應用程式時新增客戶受管金鑰 AWS CLI，請指定 kms-key-id 參數，如下所示：

```
aws m2 create-application –name test-application –description my description
--engine-type microfocus
--definition content="$(jq -c . raw-template.json | jq -R)"
--kms-key-id myApplicationKey
```

AWS Mainframe Modernization 加密內容

加密內容是一組選用的金鑰值對，包含資料的其他相關內容資訊。

AWS KMS 使用加密內容做為額外的已驗證資料，以支援已驗證的加密。當您在加密資料的請求中包含加密內容時， 會將加密內容 AWS KMS 繫結至加密的資料。若要解密資料，您必須在請求中包含相同的加密內容。

AWS Mainframe Modernization 加密內容

AWS Mainframe Modernization 會在與應用程式 （ 建立應用程式和建立部署 ） 相關的所有 AWS KMS 密碼編譯操作中使用相同的加密內容，其中金鑰為 aws:m2:app，而值為應用程式的唯一識別符。

Example

```
"encryptionContextSubset": {
        "aws:m2:app": "a1bc2defabc3defabc4defabcd"
}
```

使用加密內容進行監控

當您使用對稱客戶受管金鑰來加密應用程式或執行期環境時，您也可以使用稽核記錄和日誌中的加密內容來識別客戶受管金鑰的使用方式。

使用加密內容控制對客戶受管金鑰的存取

您也可以在金鑰政策和 IAM 政策中，使用加密內容作為 conditions 來控制對於對稱客戶受管金鑰的存取。您也可以在授予中使用加密內容條件。

AWS Mainframe Modernization 在授予中使用加密內容限制，以控制對您帳戶或區域中客戶受管金鑰的存取。授予條件會要求授予允許的操作使用指定的加密內容。下列範例是 AWS Mainframe Modernization 在建立應用程式時用來加密應用程式成品的授予。

```
//This grant is retired immediately after create application finish
{
   "grantee-principal": m2.us-west-2.amazonaws.com,
   "retiring-principal": m2.us-west-2.amazonaws.com,
   "operations": [
       "GenerateDataKey"
   ]
   "condition": {
        "encryptionContextSubset": {
            “aws:m2:app”: “a1bc2defabc3defabc4defabcd”
   }
}
```

監控您的加密金鑰以進行 AWS 大型主機現代化

當您搭配 AWS 大型主機現代化資源使用 AWS KMS 客戶受管金鑰時，您可以使用 AWS CloudTrail或 Amazon CloudWatch Logs 來追蹤 AWS 大型主機現代化傳送的請求 AWS KMS。

執行時間環境的範例

下列範例是 DescribeKey、GenerateDataKey、 CreateGrant和 AWS CloudTrail 的事件Decrypt，用於監控 AWS Mainframe Modernization 呼叫的 KMS 操作，以存取客戶受管金鑰加密的資料：

DescribeKey

AWS Mainframe Modernization 使用 DescribeKey操作來驗證與執行時間環境相關聯的 AWS KMS 客戶受管金鑰是否存在於帳戶和區域中。

下面的範例事件會記錄 DescribeKey 操作：

```
{
    "eventVersion": "1.08",
```

```
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T19:40:26Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-12-06T20:23:43Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "205.251.233.182",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "keyId": "00dd0db0-0000-0000-ac00-b0c000SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
```

```
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    },
    "sessionCredentialFromConsole": "true"
}
```

CreateGrant

當您使用 AWS KMS 客戶受管金鑰加密執行期環境時， AWS Mainframe Modernization 會代表您傳送多個`CreateGrant`請求，以執行必要的 KMS 操作。 AWS Mainframe Modernization 建立的部分授予在使用後立即淘汰。當您刪除執行時間環境時，其他 會淘汰。

下列範例事件會記錄與建立環境工作流程相關聯之 Lambda 執行角色`CreateGrant`的操作。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T20:11:45Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T20:23:09Z",
```

```
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "operations": [
            "Encrypt",
            "Decrypt",
            "ReEncryptFrom",
            "ReEncryptTo",
            "GenerateDataKey",
            "GenerateDataKey",
            "DescribeKey",
            "CreateGrant"
        ],
        "granteePrincipal": "m2.us-west-2.amazonaws.com",
        "retiringPrincipal": "m2.us-west-2.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

下列範例事件會記錄 Auto Scaling 群組服務連結角色`CreateGrant`的操作。與建立環境工作流程相關聯的 Lambda 執行角色會呼叫`CreateGrant`此操作。它授予執行角色許可，以針對 Auto Scaling 群組的服務連結角色建立子授權。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3YPCLM65MZFUPM4JO:EnvironmentWorkflow-alpha-
CreateEnvironmentLambda7-HfxDj5zz86tr",
        "arn": "arn:aws:sts::111122223333:assumed-role/EnvironmentWorkflow-
alpha-CreateEnvironmentLambdaS-1AU4A8VNQEEKN/EnvironmentWorkflow-alpha-
CreateEnvironmentLambda7-HfxDj5zz86tr",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:iam::111122223333:role/EnvironmentWorkflow-alpha-
CreateEnvironmentLambdaS-1AU4A8VNQEEKN",
                "accountId": "111122223333",
                "userName": "EnvironmentWorkflow-alpha-
CreateEnvironmentLambdaS-1AU4A8VNQEEKN"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T20:22:28Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-12-06T20:23:09Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "54.148.236.160",
    "userAgent": "aws-sdk-java/2.18.21 Linux/4.14.255-276-224.499.amzn2.x86_64
 OpenJDK_64-Bit_Server_VM/11.0.14.1+10-LTS Java/11.0.14.1 vendor/Amazon.com_Inc. md/
internal exec-env/AWS_Lambda_java11 io/sync http/Apache cfg/retry-mode/legacy",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
```

```
        "operations": [
            "Encrypt",
            "Decrypt",
            "ReEncryptFrom",
            "ReEncryptTo",
            "GenerateDataKey",
            "GenerateDataKey",
            "DescribeKey",
            "CreateGrant"
        ],
        "granteePrincipal": "m2.us-west-2.amazonaws.com",
        "retiringPrincipal": "m2.us-west-2.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_256_GCM_SHA384",
        "clientProvidedHostHeader": "kms.us-west-2.amazonaws.com"
    }
}
}
```

GenerateDataKey

當您為執行時間環境資源啟用 AWS KMS 客戶受管金鑰時，Auto Scaling 會建立唯一金鑰來加密與執行時間環境相關聯的 Amazon EBS 磁碟區。它會傳送GenerateDataKey請求至 AWS KMS ，以指定資源 AWS KMS 的客戶受管金鑰。

下面的範例事件會記錄 GenerateDataKey 操作：

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3YPCLM65EEXVIEH7D:AutoScaling",
        "arn": "arn:aws:sts::111122223333:assumed-role/AWSServiceRoleForAutoScaling/
AutoScaling",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:iam::111122223333:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling",
                "accountId": "111122223333",
                "userName": "AWSServiceRoleForAutoScaling"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T20:23:16Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "autoscaling.amazonaws.com"
    },
    "eventTime": "2022-12-06T20:23:18Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "autoscaling.amazonaws.com",
    "userAgent": "autoscaling.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "aws:ebs:id": "vol-080f7a32d290807f3"
```

```
        },
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "numberOfBytes": 64
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Decrypt

當您存取加密的執行期環境時，Amazon EBS 會呼叫 `Decrypt`操作，以使用儲存的加密資料金鑰來存取加密的資料。

下面的範例事件會記錄 `Decrypt` 操作：

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "ebs.amazonaws.com"
    },
    "eventTime": "2022-12-06T20:23:22Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "ebs.amazonaws.com",
    "userAgent": "ebs.amazonaws.com",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
```

```
        "encryptionContext": {
            "aws:ebs:id": "vol-080f7a32d290807f3"
        }
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "sharedEventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventCategory": "Management"
}
```

應用程式的範例

下列範例是 CreateGrant和 AWS CloudTrail 的事件GenerateDataKey，用於監控 AWS Mainframe Modernization 呼叫的 KMS 操作，以存取客戶受管金鑰加密的資料：

CreateGrant

當您使用 AWS KMS 客戶受管金鑰加密應用程式資源時，Lambda 執行角色會代表您傳送CreateGrant請求，以存取您 AWS 帳戶中的 KMS 金鑰。授予可讓 Lambda 執行角色使用您的客戶受管金鑰，將客戶應用程式資源上傳至 Amazon S3。應用程式建立後立即淘汰此授予。

下面的範例事件會記錄 CreateGrant 操作：

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
```

```
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T21:51:45Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T22:47:04Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "constraints": {
            "encryptionContextSubset": {
                "aws:m2:app": "a1bc2defabc3defabc4defabcd"
            }
        },
        "retiringPrincipal": "m2.us-west-2.amazonaws.com",
        "operations": [
            "GenerateDataKey"
        ],
        "granteePrincipal": "m2.us-west-2.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
```

```
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

GenerateDataKey

當您為應用程式資源啟用 AWS KMS 客戶受管金鑰時，Lambda 執行角色會建立一個金鑰，用於加密客戶資料並將其上傳至 Amazon Simple Storage Service。Lambda 執行角色會將GenerateDataKey請求傳送至 AWS KMS ，以指定資源 AWS KMS 的客戶受管金鑰。

下面的範例事件會記錄 GenerateDataKey 操作：

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3YPCLM65CLCEKKC7Z:ApplicationWorkflow-alpha-
CreateApplicationVersion-CstWZUn5R4u6",
        "arn": "arn:aws:sts::111122223333:assumed-role/ApplicationWorkflow-
alpha-CreateApplicationVersion-1IZRBZYDG20B/ApplicationWorkflow-alpha-
CreateApplicationVersion-CstWZUn5R4u6",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:iam::111122223333:role/ApplicationWorkflow-alpha-
CreateApplicationVersion-1IZRBZYDG20B",
                "accountId": "111122223333",
```

```
                "userName": "ApplicationWorkflow-alpha-
CreateApplicationVersion-1IZRBZYDG20B"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T23:28:32Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T23:29:08Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "aws:m2:app": "a1bc2defabc3defabc4defabcd",
            "aws:s3:arn": "arn:aws:s3:::supernova-processedtemplate-111122223333-us-
west-2/111122223333/a1bc2defabc3defabc4defabcd/1/cics-transaction/ZBNKE35.so"
        },
        "keySpec": "AES_256",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
```

```
}
```

部署的範例

下列範例是 CreateGrant和 AWS CloudTrail 的事件Decrypt，用於監控 AWS Mainframe Modernization 呼叫的 KMS 操作，以存取客戶受管金鑰加密的資料：

CreateGrant

當您使用 AWS KMS 客戶受管金鑰來加密部署資源時， AWS Mainframe Modernization 會代表您傳送兩個CreateGrant請求。第一個授權是針對目前的 Lambda 執行角色來呼叫 ListBatchJobScriptFiles，並在部署完成後立即淘汰。第二個授權是針對 Amazon EC2 縮小範圍的執行個體角色，以便 Amazon EC2 可以從 Amazon S3 下載客戶應用程式資源。從執行時間環境刪除應用程式時，會淘汰此授權。

下面的範例事件會記錄 CreateGrant 操作：

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T21:51:45Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T23:40:07Z",
```

```
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
        "operations": [
            "Decrypt"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:m2:app": "a1bc2defabc3defabc4defabcd"
            }
        },
        "granteePrincipal": "m2.us-west-2.amazonaws.com",
        "retiringPrincipal": "m2.us-west-2.amazonaws.com",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Decrypt

當您存取部署時，Amazon EC2 會呼叫 Decrypt操作，以使用存放的加密資料金鑰從 Amazon S3 解密和下載加密的客戶資料。

下面的範例事件會記錄 Decrypt 操作：

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3YPCLM65BSPZ37E6G:m2-hm-bqe367dxtfcpdbzmnhfzranisu",
        "arn": "arn:aws:sts::111122223333:assumed-role/
SupernovaEnvironmentInstanceScopeDownRole/m2-hm-bqe367dxtfcpdbzmnhfzranisu",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:iam::111122223333:role/
SupernovaEnvironmentInstanceScopeDownRole",
                "accountId": "111122223333",
                "userName": "SupernovaEnvironmentInstanceScopeDownRole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-12-06T23:19:29Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "m2.us-west-2.amazonaws.com"
    },
    "eventTime": "2022-12-06T23:40:15Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "m2.us-west-2.amazonaws.com",
    "userAgent": "m2.us-west-2.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "aws:m2:app": "a1bc2defabc3defabc4defabcdm",
            "aws:s3:arn": "arn:aws:s3:::supernova-processedtemplate-111122223333-us-
west-2/111122223333/a1bc2defabc3defabc4defabcdm/1/cics-transaction/BBANK40P.so"
```

```
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

進一步了解

下列資源會提供有關靜態資料加密的詳細資訊。

- 如需 AWS Key Management Service 基本概念的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》。
- 如需有關 AWS Key Management Service的安全最佳實務的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》。

傳輸中加密

對於屬於交易工作負載的互動式應用程式，終端模擬器和 TN3270 通訊協定 AWS 大型主機現代化服務端點之間的資料交換不會在傳輸中加密。如果應用程式需要傳輸中加密，您可能想要實作一些額外的通道機制。

AWS Mainframe Modernization 使用 HTTPS 加密服務 APIs。 AWS Mainframe Modernization 內的所有其他通訊都受到服務 VPC 或安全群組以及 HTTPS 的保護。 AWS Mainframe Modernization 會傳輸應用程式成品、組態和應用程式資料。應用程式成品會從您擁有的 Amazon S3 儲存貯體複製，應用程式資料也是如此。您可以使用 Amazon S3 的連結或在本機上傳檔案來提供應用程式組態。

預設會設定傳輸中的基本加密，但不適用於 TN3270 通訊協定。 AWS Mainframe Modernization 會將 HTTPS 用於 API 端點，這也是預設設定。

AWS 適用於大型主機現代化的 Identity and Access Management

AWS Identity and Access Management (IAM) 是 AWS 服務 ，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可控制誰可以進行身分驗證 （登入） 和授權 （具有許可），以使用 AWS Mainframe Modernization 資源。IAM 是您可以免費使用 AWS 服務 的 。

主題

- 目標對象
- 使用身分驗證
- 使用政策管理存取權
- AWS Mainframe Modernization 如何與 IAM 搭配使用
- AWS Mainframe Modernization 的身分型政策範例
- 針對 AWS 大型主機現代化身分和存取進行故障診斷
- 使用 AWS Mainframe Modernization的服務連結角色

目標對象

使用方式 AWS Identity and Access Management (IAM) 會有所不同，取決於您在 AWS 大型主機現代化中所做的工作。

服務使用者 – 如果您使用 AWS Mainframe Modernization 服務來執行任務，您的管理員會為您提供所需的登入資料和許可。當您使用更多 AWS 大型主機現代化功能來執行工作時，您可能需要額外的許可。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。如果您無法存取 AWS 大型主機現代化中的功能，請參閱 針對 AWS 大型主機現代化身分和存取進行故障診斷。

服務管理員 – 如果您在公司負責 AWS 大型主機現代化資源，您可能擁有 AWS 大型主機現代化的完整存取權。您的任務是判斷服務使用者應存取 AWS 的主要主機現代化功能和資源。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司如何將 IAM 與 AWS 大型主機現代化搭配使用，請參閱 AWS Mainframe Modernization 如何與 IAM 搭配使用。

IAM 管理員 – 如果您是 IAM 管理員，建議您進一步了解如何撰寫政策以管理 AWS Mainframe Modernization 的存取權。若要檢視您可以在 IAM 中使用的 AWS Mainframe Modernization 身分型政策範例，請參閱 AWS Mainframe Modernization 的身分型政策範例。

使用身分驗證

身分驗證是您 AWS 使用身分憑證登入 的方式。您必須以 AWS 帳戶根使用者身分、IAM 使用者身分或擔任 IAM 角色來驗證 （ 登入 AWS)。

您可以使用透過身分來源提供的憑證，以聯合身分 AWS 身分身分登入 。 AWS IAM Identity Center (IAM Identity Center) 使用者、您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。當您使用聯合 AWS 身分存取 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。如需登入的詳細資訊 AWS，請參閱AWS 登入 《 使用者指南》中的如何登入您的 AWS 帳戶 。

如果您以 AWS 程式設計方式存取 ， AWS 會提供軟體開發套件 (SDK) 和命令列界面 (CLI)，以使用您的 憑證以密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，則必須自行簽署請求。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》中的適用於 API 請求的AWS Signature 第 4 版。

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 建議您使用多重驗證 (MFA) 來提高帳戶的安全性。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 IAM 中的AWS 多重要素驗證。

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，該身分可完整存取帳戶中的所有 AWS 服務 和資源。此身分稱為 AWS 帳戶 Theroot 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

最佳實務是， 要求人類使用者，包括需要管理員存取權的使用者，使用臨時 AWS 服務 憑證與身分提供者聯合來存取 。

聯合身分是來自您的企業使用者目錄、Web 身分提供者、 AWS Directory Service、 Identity Center 目錄或任何使用透過身分來源提供的登入資料 AWS 服務 存取的使用者。當聯合身分存取時 AWS 帳戶，它們會擔任 角色，而角色會提供臨時登入資料。

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，也可以連接並同步到您自己的身分來源中的一組使用者 AWS 帳戶 和群組，以便

在所有 和應用程式中使用。如需 IAM Identity Center 的詳細資訊，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 中的身分 AWS 帳戶 ，具有單一人員或應用程式的特定許可。建議您盡可能依賴臨時憑證，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，但角色僅提供臨時憑證。如需更多資訊，請參閱《IAM 使用者指南》中的 IAM 使用者的使用案例。

IAM 角色

IAM 角色是 中具有特定許可 AWS 帳戶 的身分。它類似 IAM 使用者，但不與特定的人員相關聯。若要暫時在 中擔任 IAM 角色 AWS Management Console，您可以從使用者切換至 IAM 角色 （主控台）。您可以透過呼叫 AWS CLI 或 AWS API 操作或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱《IAM 使用者指南》中的擔任角色的方法。

使用臨時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需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 (聯合) 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 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。不過，對於某些 AWS 服務，您可以將政策直接連接到資源（而不是使用角色做為代理）。如需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 中的跨帳戶資源存取。

- 跨服務存取 – 有些 AWS 服務 使用其他 中的功能 AWS 服務。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。
  - 轉送存取工作階段 (FAS) – 當您使用 IAM 使用者或角色在其中執行動作時 AWS，您會被視為委託人。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 使用呼叫 的委託人許可 AWS 服務，結合 AWS 服務 請求向下游服務提出請求。只有在服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
  - 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可權給 AWS 服務。
  - 服務連結角色 – 服務連結角色是一種連結至 的服務角色類型 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 中 AWS 帳戶 ，並由服務擁有。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 執行個體上執行之應用程式的臨時登入資料，以及提出 AWS CLI 或 AWS API 請求。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 執行個體並將其提供給其所有應用程式，您可以建立連接至執行個體的執行個體描述檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得臨時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM 角色來授予許可權給 Amazon EC2 執行個體上執行的應用程式。

使用政策管理存取權

您可以透過建立政策並將其連接到身分或資源 AWS 來控制 AWS 中的存取。政策是 中的物件， AWS 當與身分或資源相關聯時， 會定義其許可。當委託人 （使用者、根使用者或角色工作階段） 發出請求時， 會 AWS 評估這些政策。政策中的許可決定是否允許或拒絕請求。大多數政策會以 JSON 文件 AWS 的形式存放在 中。如需 JSON 政策文件結構和內容的詳細資訊，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。具有該政策的使用者可以從 AWS Management Console AWS CLI、 或 API AWS 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。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的透過客戶管理政策定義自訂 IAM 許可。

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。受管政策是獨立的政策，您可以連接到 中的多個使用者、群組和角色 AWS 帳戶。受管政策包括 AWS 受管政策和客戶受管政策。如需了解如何在受管政策及內嵌政策之間選擇，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。委託人可以包含帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策中使用來自 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。

Amazon S3 AWS WAF和 Amazon VPC 是支援 ACLs的服務範例。如需進一步了解 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政策類型。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。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的 IAM 實體許可界限。

- 服務控制政策 SCPs) – SCPs是 JSON 政策，可指定 中組織或組織單位 (OU) 的最大許可 AWS Organizations。 AWS Organizations 是一種用於分組和集中管理您企業擁有 AWS 帳戶 的多個 的服務。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 會限制成員帳戶中實體的許可，包括每個實體 AWS 帳戶根使用者。如需 Organizations 和 SCP 的詳細資訊，請參閱《AWS Organizations 使用者指南》中的服務控制政策。
- 資源控制政策 (RCP) - RCP 是 JSON 政策，可用來設定您帳戶中資源的可用許可上限，採取這種方式就不需要更新附加至您所擁有的每個資源的 IAM 政策。RCP 會限制成員帳戶中資源的許可，並可能影響身分的有效許可，包括 AWS 帳戶根使用者，無論它們是否屬於您的組織。如需 Organizations 和 RCPs的詳細資訊，包括 AWS 服務 支援 RCPs 清單，請參閱AWS Organizations《 使用者指南》中的資源控制政策 (RCPs)。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過撰寫程式的方式建立角色或聯合使用者的暫時工作階段時，做為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要了解如何 AWS 在涉及多個政策類型時決定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

AWS Mainframe Modernization 如何與 IAM 搭配使用

在您使用 IAM 管理 AWS Mainframe Modernization 的存取權之前，請先了解有哪些 IAM 功能可與 AWS Mainframe Modernization 搭配使用。

可與 AWS 大型主機現代化搭配使用的 IAM 功能

IAM 功能	AWS 大型主機現代化支援
身分型政策	是
資源型政策	否
政策動作	是

IAM 功能	AWS 大型主機現代化支援
政策資源	是
政策條件索引鍵	是
ACL	否
ABAC (政策中的標籤)	是
臨時憑證	是
轉送存取工作階段 (FAS)	是
服務角色	是
服務連結角色	是

若要全面了解 AWS Mainframe Modernization 和其他 AWS 服務如何與大多數 IAM 功能搭配使用，請參閱《AWS IAM 使用者指南》中的與 IAM 搭配使用的 服務。

AWS Mainframe Modernization 的身分型政策

支援身分型政策：是

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。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的透過客戶管理政策定義自訂 IAM 許可。

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。

AWS Mainframe Modernization 的身分型政策範例

若要檢視 AWS Mainframe Modernization 身分型政策的範例，請參閱 AWS Mainframe Modernization 的身分型政策範例。

AWS Mainframe Modernization 中的資源型政策

支援資源型政策：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。委託人可以包含帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

如需啟用跨帳戶存取權，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，做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。當委託人和資源位於不同位置時 AWS 帳戶，信任帳戶中的 IAM 管理員也必須授予委託人實體（使用者或角色）存取資源的許可。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 中的快帳戶資源存取。

AWS Mainframe Modernization 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。政策動作通常具有與相關聯 AWS API 操作相同的名稱。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 AWS 大型主機現代化動作的清單，請參閱《服務授權參考》中的AWS 大型主機現代化定義的動作。

AWS Mainframe Modernization 中的政策動作在動作之前使用下列字首：

```
m2
```

如需在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "m2:StartApplication",
      "m2:StopApplication"
         ]
```

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 `List` 文字的所有動作，請包含以下動作：

```
"Action": "m2:List*"
```

若要檢視 AWS Mainframe Modernization 身分型政策的範例，請參閱 AWS Mainframe Modernization 的身分型政策範例。

AWS Mainframe Modernization 的政策資源

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": "*"
```

您可以使用特定 AWS 大型主機現代化資源ARNs 來識別 IAM 政策適用的資源，以限制對特定大型主機現代化資源的存取。如需有關 ARN 格式的詳細資訊，請參閱《AWS 一般參考》中的 Amazon Resource Name (ARN)。

例如， AWS 大型主機現代化環境具有下列 ARN。

```
"Resource": "arn:aws:m2:regionId:accountId:env/service-generated-unique-identifier"
```

AWS Mainframe Modernization 應用程式具有下列 ARN。

```
"Resource": "arn:aws:m2:regionId:accountId:app/service-generated-unique-identifier"
```

並非所有 AWS 大型主機現代化動作都支援資源層級許可。對於不支援資源層級許可的動作，您必須使用萬用字元 (*)。

下列 AWS Mainframe Modernization 動作不支援資源層級許可。

```
ListApplications
            ListApplicationVersions
            ListBatchJobDefinitions
            ListBatchJobExecutions
```

```
            ListDataSetImportHistory
            ListDataSets
            ListDeployments
            ListEngineVersions
            ListEnvironments
            ListTagsForResource
```

若要查看 AWS 大型主機現代化資源類型及其 ARNs的清單，請參閱《服務授權參考》中的AWS 大型主機現代化定義的資源。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱 AWS Mainframe Modernization 定義的動作。

若要檢視 AWS Mainframe Modernization 身分型政策的範例，請參閱 AWS Mainframe Modernization 的身分型政策範例。

AWS Mainframe Modernization API 許可：動作、資源和條件參考

當您撰寫可連接到 IAM 身分的許可政策（以身分為基礎的政策）時，您可以使用下表做為參考。資料表包含下列項目：

- 每個 AWS 大型主機現代化 API 操作。
- 您可以授予執行動作許可的對應動作。
- 您可以授予許可 AWS 的資源。

您要在政策的 `Action` 欄位中指定動作，並在政策的 `Resource` 欄位中指定資源值。

您可以在 AWS 大型主機現代化政策中使用 AWS 全域條件金鑰來表達條件。如需 AWS 金鑰的完整清單，請參閱《IAM 使用者指南》中的可用全域條件金鑰。

> **Note**
>
> 若要指定動作，請使用後接 API 操作名稱的 `m2:` 字首 (例如，`m2:CreateApplication`)。

AWS Mainframe Modernization API 和動作的必要許可

AWS 大型主機現代化 API 操作	所需許可 (API 動作)	資源
CancelBatchJobExecution		應用程式

AWS 大型主機現代化 API 操作	所需許可 (API 動作)	資源
CreateApplication	iam:PassRole kms:DescribeKey kms:CreateGrant s3:GetObject s3:ListBucket	應用程式
CreateDataSetImportTask	s3:GetObject	應用程式
CreateDataSetExportTask	kms:DescribeKey s3:PutObject	應用程式

AWS 大型主機現代化 API 操作	所需許可 (API 動作)	資源
CreateDeployment	elasticloadbalancing:AddTags elasticloadbalancing:CreateListener elasticloadbalancing:CreateTargetGroup elasticloadbalancing:RegisterTargets elasticloadbalancing:DeleteListener elasticloadbalancing:DeleteTargetGroup elasticloadbalancing:DeregisterTargets elasticloadbalancing:DeleteLoadBalancer logs:CreateLogDelivery logs:GetLogDelivery logs:UpdateLogDelivery logs:DeleteLogDelivery logs:ListLogDeliveries logs:PutResourcePolicy logs:DescribeResourcePolicies	應用程式

AWS 大型主機現代化 API 操作	所需許可 (API 動作)	資源
	`logs:DescribeLogGroups`	

AWS 大型主機現代化 API 操作	所需許可 (API 動作)	資源
CreateEnvironment	ec2:CreateNetworkInterface ec2:CreateNetworkInterfacePermission ec2:DescribeNetworkInterfaces ec2:DescribeSecurityGroups ec2:DescribeSubnets ec2:DescribeVpcAttribute ec2:DescribeVpcs ec2:ModifyNetworkInterfaceAttribute elasticfilesystem:DescribeMountTargets elasticloadbalancing:AddTags elasticloadbalancing:CreateLoadBalancer elasticloadbalancing:DeleteLoadBalancer kms:DescribeKey kms:CreateGrant fsx:DescribeFileSystems	環境

AWS 大型主機現代化 API 操作	所需許可 (API 動作)	資源
	`iam:CreateServiceLinkedRole`	

DeleteApplication	`elasticloadbalancing:DeleteListener`
	`elasticloadbalancing:DeleteTargetGroup`
	`logs:DeleteLogDelivery`
		應用程式

DeleteApplicationFromEnvironment	`elasticloadbalancing:DeleteListener` `elasticloadbalancing:DeleteTargetGroup`	應用程式 環境
DeleteEnvironment	`elasticloadbalancing:DeleteLoadBalancer`	環境
GetApplication		應用程式
GetApplicationVersion		應用程式
GetBatchJobExecution		應用程式
GetDataSetDetails		應用程式
GetDataSetImportTask		應用程式
GetDataSetExportTask		應用程式
GetDeployment		應用程式
GetEnvironment		環境

AWS 大型主機現代化 API 操作	所需許可 (API 動作)	資源
ListApplications		*
ListApplicationVersions		*
ListBatchJobDefinitions		*
ListBatchJobExecutions		*
ListDataSetImportHistory		*
ListDataSetExportHistory		*
ListDataSets		*
ListDeployments		*
ListEngineVersions		*
ListEnvironments		*
ListTagsForResource		*
StartApplication		應用程式
StartBatchJob		應用程式
StopApplication		應用程式
TagResource		*
UntagResource		*

AWS 大型主機現代化 API 操作	所需許可 (API 動作)	資源
UpdateApplication	s3:GetObject s3:ListBucket	應用程式
UpdateEnvironment	kms:DescribeKey	環境

AWS Mainframe Modernization 的政策條件索引鍵

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， 會使用邏輯OR操作 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的條件金鑰。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的AWS 全域條件內容索引鍵。

下列條件索引鍵專屬於 AWS Mainframe Modernization

```
m2:EngineType
            m2:InstanceType
```

若要查看 AWS 大型主機現代化條件金鑰的清單，請參閱《服務授權參考》中的AWS 大型主機現代化條件金鑰。若要了解您可以使用條件金鑰的動作和資源，請參閱 AWS Mainframe Modernization 定義的動作。

若要檢視 AWS Mainframe Modernization 身分型政策的範例，請參閱 AWS Mainframe Modernization 的身分型政策範例。

AWS Mainframe Modernization 中的存取控制清單 (ACLs)

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。

具有 AWS 大型主機現代化功能的屬性型存取控制 (ABAC)

支援 ABAC (政策中的標籤)：是

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。您可以將標籤連接至 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。

如需根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，在政策的條件元素中，提供標籤資訊。

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》中的使用 ABAC 授權定義許可。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)。

將暫時登入資料與 AWS 大型主機現代化搭配使用

支援臨時憑證：是

當您使用臨時登入資料登入時，有些 AWS 服務 無法運作。如需詳細資訊，包括哪些 AWS 服務 使用臨時登入資料，請參閱《AWS 服務 IAM 使用者指南》中的 使用 IAM。

如果您 AWS Management Console 使用使用者名稱和密碼以外的任何方法登入 ，則會使用臨時登入資料。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取 時，該程序會自動建立臨時登入資料。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》中的從使用者切換至 IAM 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。

轉送 AWS 大型主機現代化存取工作階段

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，您會被視為委託人。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 使用呼叫 的委託人許可 AWS 服務，結合 AWS 服務 請求向下游服務提出請求。只有在服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時，才會提出 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。

> **Important**
>
> 這些字符可讓 AWS Mainframe Modernization 在沒有您明確同意的情況下存取客戶資料；例如， AWS Mainframe Modernization 會在未取得客戶明確許可的情況下，從 Amazon S3 儲存貯體部署具有相關聯業務資料的應用程式成品。您可能需要相應地更新任何合規文件。

AWS Mainframe Modernization 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。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可權給 AWS 服務。

AWS Mainframe Modernization 支援活動掛鉤的服務角色（交易/任務結束或完成等）。

> **Warning**
>
> 變更服務角色的許可可能會中斷 AWS 大型主機現代化功能。只有在 AWS Mainframe Modernization 提供指引時，才能編輯服務角色。

在 AWS 大型主機現代化中選擇 IAM 角色

如果您先前已建立在 Amazon EC2 上執行的應用程式可擔任的 IAM 角色，您可以在建立啟動範本或啟動組態時選擇此角色。 AWS Mainframe Modernization 會為您提供可供選擇的角色清單。建立這些角色時，請務必將最低權限 IAM 政策建立關聯，以限制應用程式所需之特定 API 呼叫的存取。如需詳細

資訊，請參閱《Amazon EC2 Auto Scaling 使用者指南》中的在 Amazon EC2 執行個體上執行的應用程式 IAM 角色。

AWS Mainframe Modernization 的服務連結角色

支援服務連結角色：是

服務連結角色是連結至 的一種服務角色 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 中 AWS 帳戶 ，並由服務擁有。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 AWS Mainframe Modernization 服務連結角色的詳細資訊，請參閱 使用 AWS Mainframe Modernization的服務連結角色。

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。

AWS Mainframe Modernization 的身分型政策範例

根據預設，使用者和角色沒有建立或修改 AWS Mainframe Modernization 資源的許可。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 來執行任務。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。

如需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 (主控台)。

如需有關 AWS 大型主機現代化所定義之動作和資源類型的詳細資訊，包括每種資源類型的 ARNs 格式，請參閱《服務授權參考》中的AWS 大型主機現代化的動作、資源和條件金鑰。

主題

- 政策最佳實務
- 使用 AWS Mainframe Modernization 主控台
- 允許使用者檢視他們自己的許可

政策最佳實務

身分型政策會判斷您帳戶中的某個人員是否可以建立、存取或刪除 AWS Mainframe Modernization 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 受管政策並邁向最低權限許可 – 若要開始將許可授予使用者和工作負載，請使用將許可授予許多常見使用案例的 AWS 受管政策。它們可在您的 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，以進一步減少許可。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。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 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。如果透過特定 例如 使用服務動作 AWS 服務，您也可以使用條件來授予其存取權 AWS CloudFormation。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM Access Analyzer 驗證政策。
- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或 中的根使用者 AWS 帳戶，請開啟 MFA 以提高安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_configure-api-require.html中的透過 MFA 的安全 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 AWS Mainframe Modernization 主控台

若要存取 AWS Mainframe Modernization 主控台，您必須擁有一組最低許可。這些許可必須允許您列出和檢視 中 AWS 大型主機現代化資源的詳細資訊 AWS 帳戶。如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

對於僅呼叫 AWS CLI 或 AWS API 的使用者，您不需要允許最低主控台許可。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。

為了確保使用者和角色仍然可以使用 AWS Mainframe Modernization 主控台，請將 AWS Mainframe Modernization `ConsoleAccess`或 `ReadOnly` AWS 受管政策連接至實體。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。此政策包含在主控台或使用 或 AWS CLI AWS API 以程式設計方式完成此動作的許可。

```
{
    "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": "*"
        }
    ]
}
```

針對 AWS 大型主機現代化身分和存取進行故障診斷

使用以下資訊來協助您診斷和修正使用 AWS Mainframe Modernization 和 IAM 時可能遇到的常見問題。

主題

- 我未獲得執行 iam:PassRole 的授權
- 我想要允許 以外的人員 AWS 帳戶 存取我的 AWS 大型主機現代化資源

我未獲得執行 iam:PassRole 的授權

如果您收到錯誤，告知您無權執行 `iam:PassRole`動作，您的政策必須更新，以允許您將角色傳遞至 AWS Mainframe Modernization。

有些 AWS 服務 可讓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 `marymajor` 使用者嘗試使用主控台在 AWS Mainframe Modernization 中執行動作時，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 管理員。您的管理員提供您的簽署憑證。

我想要允許 以外的人員 AWS 帳戶 存取我的 AWS 大型主機現代化資源

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：

- 若要了解 AWS Mainframe Modernization 是否支援這些功能，請參閱 AWS Mainframe Modernization 如何與 IAM 搭配使用。

- 若要了解如何提供您擁有 AWS 帳戶 的資源存取權，請參閱《IAM 使用者指南》中的在您擁有 AWS 帳戶 的另一個 中為 IAM 使用者提供存取權。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《IAM 使用者指南》中的將存取權提供給第三方 AWS 帳戶 擁有。
- 如需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)。
- 如需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 中的跨帳戶資源存取。

使用 AWS Mainframe Modernization的服務連結角色

AWS Mainframe Modernization use AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 的唯一 IAM 角色類型 AWS Mainframe Modernization。服務連結角色由 預先定義， AWS Mainframe Modernization 並包含服務代表您呼叫其他 AWS 服務所需的所有許可。

服務連結角色可讓您更 AWS Mainframe Modernization 輕鬆地設定，因為您不必手動新增必要的許可。 AWS Mainframe Modernization 會定義其服務連結角色的許可，除非另有定義，否則只能 AWS Mainframe Modernization 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。

您必須先刪除服務連結角色的相關資源，才能將其刪除。這可保護您的 AWS Mainframe Modernization 資源，因為您不會不小心移除存取資源的許可。

如需有關支援服務連結角色的其他 服務的資訊，請參閱AWS 服務連結角色欄中適用於 IAM 的服務，並尋找具有是的服務。 選擇具有連結的是，以檢視該服務的服務連結角色文件。

AWS Mainframe Modernization的服務連結角色許可

AWS Mainframe Modernization 使用名為 AWSServiceRoleForAWSM2 的服務連結角色 – 設定網路以連線至您的 VPC 並存取檔案系統等資源。

AWSServiceRoleForAWSM2 服務連結角色信任下列服務擔任該角色：

- `m2.amazonaws.com`

名為 AWSM2ServicePolicy 的角色許可政策允許 對指定的資源 AWS Mainframe Modernization 完成下列動作：

- 建立、刪除、描述和連接許可到 AWS Mainframe Modernization 環境的 Amazon EC2 網路介面，以建立與客戶 VPC 的連線。
- 從 Elastic Load Balancing 註冊或取消註冊項目，這是客戶連線到 AWS Mainframe Modernization 環境的方式。
- 如果使用，請描述 Amazon EFS 或 Amazon FSx 檔案系統。
- 從執行時間環境將指標傳送至客戶的 CloudWatch。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "ec2:DescribeSubnets",
    "ec2:CreateNetworkInterface",
    "ec2:DeleteNetworkInterface",
    "ec2:DescribeNetworkInterfaces",
    "ec2:CreateNetworkInterfacePermission",
    "ec2:ModifyNetworkInterfaceAttribute"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "elasticfilesystem:DescribeMountTargets"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "elasticloadbalancing:RegisterTargets",
    "elasticloadbalancing:DeregisterTargets"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "fsx:DescribeFileSystems"
```

```
    ],
    "Resource": "*"
   },
   {
    "Effect": "Allow",
    "Action": [
     "cloudwatch:PutMetricData"
    ],
    "Resource": "*",
    "Condition": {
     "StringEquals": {
      "cloudwatch:namespace": [
       "AWS/M2"
      ]
     }
    }
   }
  ]
}
```

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 AWS Mainframe Modernization建立服務連結角色

您不需要手動建立一個服務連結角色。當您在 AWS Management Console、 AWS CLI或 AWS API 中建立執行期環境時， AWS Mainframe Modernization 會為您建立服務連結角色。

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。當您建立執行期環境時， 會再次為您 AWS Mainframe Modernization 建立服務連結角色。

為 AWS Mainframe Modernization編輯服務連結角色

AWS Mainframe Modernization 不允許您編輯 AWSServiceRoleForAWSM2 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。如需詳細資訊，請參閱《IAM 使用者指南》中的編輯服務連結角色。

為 AWS Mainframe Modernization刪除服務連結角色

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。

> **Note**
>
> 如果 AWS Mainframe Modernization 服務在您嘗試刪除資源時使用角色，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。

刪除 AWSServiceRoleForAWSM2 使用 AWS Mainframe Modernization 的資源

- 刪除 中的執行時間環境 AWS Mainframe Modernization。刪除環境本身之前，請務必從環境刪除應用程式。

使用 IAM 手動刪除服務連結角色

使用 IAM 主控台 AWS CLI、 或 AWS API 來刪除 AWSServiceRoleForAWSM2 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。

AWS Mainframe Modernization 服務連結角色的支援區域

AWS Mainframe Modernization 支援在提供服務的所有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。

AWS Mainframe Modernization 的合規驗證

在多個合規計畫中，第三方稽核人員會評估 AWS 大型主機現代化的安全性和 AWS 合規性。這些計劃包括 SOC、PCI、FedRAMP、HIPAA 等等。

如需特定合規計劃範圍內 AWS 的服務清單，請參閱合規計劃範圍內的 AWS 服務。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，請參閱在 中下載報告 AWS Artifact。

您使用 AWS Mainframe Modernization 時的合規責任取決於資料的敏感度、您公司的合規目標，以及適用的法律和法規。 AWS 提供下列資源來協助合規：

- 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。
- HIPAA 安全與合規架構白皮書 – 此白皮書說明公司如何使用 AWS 來建立符合 HIPAA 規範的應用程式。

- AWS 合規資源 – 此工作手冊和指南集合可能適用於您的產業和據點。
- 《 AWS Config 開發人員指南》中的使用規則評估資源 AWS Config：評估資源組態符合內部實務、產業準則和法規的程度。
- AWS Security Hub – AWS 此服務提供 內安全狀態的完整檢視 AWS ，協助您檢查是否符合安全產業標準和最佳實務。

AWS 大型主機現代化中的彈性

AWS 全球基礎設施是以 AWS 區域和可用區域為基礎建置。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 AWS 區域和可用區域的詳細資訊，請參閱 AWS 全球基礎設施。

中的基礎設施安全 AWS Mainframe Modernization

作為受管服務， AWS Mainframe Modernization 受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及如何 AWS 保護基礎設施的資訊，請參閱AWS 雲端安全。若要使用基礎設施安全最佳實務來設計您的 AWS 環境，請參閱安全支柱 AWS Well-Architected Framework 中的基礎設施保護。

您可以使用 AWS 發佈的 API 呼叫， AWS Mainframe Modernization 透過網路存取 。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。

AWS Mainframe Modernization 使用 AWS PrivateLink 介面端點存取

您可以使用 在 VPC 和 之間 AWS PrivateLink 建立私有連線 AWS Mainframe Modernization。您可以 AWS Mainframe Modernization 像在 VPC 中一樣存取 ，無需使用網際網路閘道、NAT 裝置、VPN

連接或 AWS Direct Connect 連線。VPC 中的執行個體不需要公有 IP 地址即可存取 AWS Mainframe Modernization。

您可以建立由 AWS PrivateLink提供支援的介面端點來建立此私有連線。我們會在您為介面端點啟用的每個子網中建立端點網路介面。這些是請求者管理的網路介面，可作為目的地為 AWS Mainframe Modernization之流量的進入點。

如需詳細資訊，請參閱《 AWS PrivateLink 指南》中的AWS 服務 透過 存取 AWS PrivateLink 。

的考量事項 AWS Mainframe Modernization

在您設定介面端點之前 AWS Mainframe Modernization，請檢閱《 AWS PrivateLink 指南》中的考量事項。

AWS Mainframe Modernization 支援透過界面端點呼叫其所有 API 動作。

建立 的介面端點 AWS Mainframe Modernization

您可以使用 Amazon VPC AWS Mainframe Modernization 主控台或 AWS Command Line Interface () 建立 的介面端點AWS CLI。如需詳細資訊，請參閱《AWS PrivateLink 指南》中的建立介面端點。

AWS Mainframe Modernization 使用下列服務名稱建立 的介面端點：

```
com.amazonaws.region.m2
```

如果您為介面端點啟用私有 DNS，您可以使用 AWS Mainframe Modernization 其預設的區域 DNS 名稱向 提出 API 請求。例如 `m2.us-east-1.amazonaws.com`。

為您的介面端點建立端點政策

端點政策為 IAM 資源，您可將其連接至介面端點。預設端點政策允許 AWS Mainframe Modernization 透過介面端點完整存取 。若要控制允許 AWS Mainframe Modernization 從您的 VPC 存取 ，請將自訂端點政策連接至介面端點。

端點政策會指定以下資訊：

- 可執行動作的委託人 (AWS 帳戶、使用者和 IAM 角色) 。
- 可執行的動作。
- 可供執行動作的資源。

如需詳細資訊，請參閱「AWS PrivateLink 指南」中的使用端點政策控制對服務的存取。

範例： AWS Mainframe Modernization 動作的 VPC 端點政策

以下是自訂端點政策的範例。將此政策附加至介面端點後，此政策會針對所有資源上的所有主體，授予列出的 AWS Mainframe Modernization 動作的存取權限。

```
//Example of an endpoint policy where access is granted to the
//listed AWS Mainframe Modernization actions for all principals on all resources
{"Statement": [
      {"Principal": "*",
         "Effect": "Allow",
         "Action": [
            "m2:ListApplications",
            "m2:ListEnvironments",
            "m2:ListDeployments"
         ],
         "Resource":"*"
      }
   ]
}

//Example of an endpoint policy where access is denied to all the
//AWS Mainframe Modernization CREATE actions for all principals on all resources
{"Statement": [
      {"Principal": "*",
         "Effect": "Deny",
         "Action": [
            "m2:Create*"
         ],
         "Resource":"*"
      }
   ]
}
```

監控 AWS 大型主機現代化

監控是維護 AWS 大型主機現代化和其他 AWS 解決方案可靠性、可用性和效能的重要部分。AWS 提供下列監控工具來監看 AWS Mainframe Modernization、在發生錯誤時回報，以及適時採取自動動作：

- Amazon CloudWatch AWS 會即時監控您的 AWS 資源和您在 上執行的應用程式。您可以收集和追蹤指標、建立自訂儀板表，以及設定警示，在特定指標達到您指定的閾值時通知您或採取動作。例如，您可以讓 CloudWatch 追蹤 CPU 使用量或其他 Amazon EC2 執行個體指標，並在需要時自動啟動新的執行個體。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。
- Amazon CloudWatch Logs 可讓您監控、存放和存取來自 Amazon EC2 執行個體、CloudTrail 及其他來源的日誌檔案。CloudWatch Logs 可監控日誌檔案中的資訊，並在達到特定閾值時通知您。您也可以將日誌資料存檔在高耐用性的儲存空間。如需詳細資訊，請參閱 Amazon CloudWatch Logs 使用者指南。
- AWS CloudTrail 會擷取由您的帳戶發出或代表 AWS 您的帳戶發出的 API 呼叫和相關事件，並將日誌檔案交付至您指定的 Amazon S3 儲存貯體。您可以識別呼叫的使用者和帳戶 AWS、進行呼叫的來源 IP 地址，以及呼叫的時間。如需詳細資訊，請參閱「AWS CloudTrail 使用者指南」。

使用 Amazon CloudWatch 監控 AWS 大型主機現代化

您可以使用 CloudWatch 監控 AWS 大型主機現代化，這會收集原始資料，並將其處理為可讀且近乎即時的指標。這些統計資料會保留 15 個月，以便您存取歷史資訊，並更清楚 Web 應用程式或服務的執行效能。您也可以設定留意特定閾值的警示，當滿足這些閾值時傳送通知或採取動作。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。

下表列出 AWS Mainframe Modernization 的指標和維度。這些指標的命名空間為 AWS/M2。

執行時間環境指標

指標	描述
CPUUtilization	環境中執行個體的 CPU 使用率。 維度： environmentId 單位：百分比

指標	描述
	有效統計：平均、最小值、最大值
InboundNetworkThroughput	環境中執行個體的傳入網路輸送量。 維度： environmentId 單位：位元組/秒 有效統計：平均、最小值、最大值
MemoryUtilization	環境中執行個體的記憶體使用率。 維度： environmentId 單位：百分比 有效統計：平均、最小值、最大值
OutboundNetworkThroughput	環境中執行個體的傳出網路輸送量。 維度： environmentId 單位：位元組/秒 有效統計：平均、最小值、最大值

應用程式指標

指標	描述
BatchJobCompletedCount	在時間間隔內完成的任務數量。 此指標適用於 Rocket Software （先前為 Micro Focus) 和 AWS Blu Age 3.7.0 及更新版本。 維度：applicationId 單位：計數

指標	描述
	有效的統計資訊：總和
BatchJobFailedCount	時間間隔內失敗的任務數量。 此指標適用於 Rocket Software 和 AWS Blu Age 3.7.0 及更新版本。 維度：applicationId 單位：計數 有效的統計資訊：總和
JvmMemoryFree	Java 虛擬機器目前未使用的可用記憶體數量。 此指標僅適用於 AWS Blu Age 執行期引擎。適用於 AWS Blu Age 3.7.0 及更新版本。 維度：applicationId 單位：位元組 有效統計：平均、最小值、最大值
JvmMemoryMax	Java 虛擬機器允許的記憶體數量上限。 此指標僅適用於 AWS Blu Age 執行期引擎。適用於 AWS Blu Age 3.7.0 及更新版本。 維度：applicationId 單位：位元組 有效統計：平均、最小值、最大值

指標	描述
JvmMemoryUsed	Java 虛擬機器主動使用的記憶體數量。 此指標僅適用於 AWS Blu Age 執行期引擎。適用於 AWS Blu Age 3.7.0 及更新版本。 維度：applicationId 單位：位元組 有效統計：平均、最小值、最大值
ProcessesActiveCount	正在處理請求的並行服務執行程序的作用中數量。 此指標僅適用於 Rocket Software 執行期引擎。 維度：applicationId 單位：計數 有效的統計資訊：總和
SessionCount	應用程式 HTTP 工作階段的數量。 此指標僅適用於 AWS Blu Age 執行期引擎。適用於 AWS Blu Age 3.7.0 及更新版本。 維度：applicationId 單位：計數 有效統計：平均、最小值、最大值

指標	描述
SharedMemoryFree	可供企業伺服器儲存執行交易和任務所需的所有資訊的記憶體。 此指標僅適用於 Rocket Software 執行期引擎。 維度：applicationId 單位：KB 有效統計：平均、最小值、最大值
SharedMemoryTotal	分配給企業伺服器的共用記憶體總量，以存放執行交易和任務所需的所有資訊。 此指標僅適用於 Rocket Software 執行期引擎。 維度：applicationId 單位：KB 有效統計：平均、最小值、最大值
ThreadActiveCount	正在處理請求的引擎執行緒數量。 此指標僅適用於 AWS Blu Age 執行期引擎。適用於 AWS Blu Age 3.7.0 及更新版本。 維度：applicationId 單位：計數 有效統計：平均、最小值、最大值

指標	描述
TransactionCompletedCount	在時間間隔內遞交的交易數量。 此指標適用於 Rocket Software 和 AWS Blu Age 3.7.0 及更新版本。 維度：applicationId 單位：計數 有效的統計資訊：總和
TransactionFailedCount	時間間隔內失敗的交易數量。 此指標適用於 Rocket Software 和 AWS Blu Age 3.7.0 及更新版本。 維度：applicationId 單位：計數 有效的統計資訊：總和
TransactionResponseTime	從使用者傳送請求到應用程式指出請求已完成的時間。 此指標適用於 Rocket Software 和 AWS Blu Age 3.7.0 及更新版本。 維度：applicationId 單位：毫秒 有效統計：平均、最小值、最大值

維度

維度	描述
applicationId	此維度會依 ID 篩選已識別應用程式的指標。
environmentId	此維度會依 ID 篩選已識別環境的指標。

使用 記錄 AWS 大型主機現代化 API 呼叫 AWS CloudTrail

AWS Mainframe Modernization 已與 服務整合 AWS CloudTrail，此服務提供由使用者、角色或 AWS Mainframe Modernization 中的 AWS 服務所採取之動作的記錄。CloudTrail 會將 AWS 大型主機現代化的所有 API 呼叫擷取為事件。擷取的呼叫包括來自 AWS Mainframe Modernization 主控台的呼叫，以及對 AWS Mainframe Modernization API 操作的程式碼呼叫。如果您建立線索，則可以將 CloudTrail 事件持續交付至 Amazon S3 儲存貯體，包括 AWS 大型主機現代化的事件。即使您未設定追蹤，依然可以透過 CloudTrail 主控台中的事件歷史記錄檢視最新事件。您可以使用 CloudTrail 所收集的資訊，判斷對 AWS Mainframe Modernization 提出的請求、提出請求的 IP 地址、提出請求的人員、提出請求的時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱「AWS CloudTrail 使用者指南」。

AWS CloudTrail 中的大型主機現代化資訊

當您建立 AWS 帳戶時，會在您的帳戶上啟用 CloudTrail。當活動在 AWS Mainframe Modernization 中發生時，該活動會記錄在 CloudTrail 事件中，以及事件歷史記錄中的其他服務 AWS 事件。您可以在 AWS 帳戶中檢視、搜尋和下載最近的事件。如需詳細資訊，請參閱《使用 CloudTrail 事件歷史記錄檢視事件》https://docs.aws.amazon.com/awscloudtrail/latest/userguide/view-cloudtrail-events.html。

若要持續記錄您 AWS 帳戶中的事件，包括 AWS 大型主機現代化的事件，請建立追蹤。線索能讓 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。根據預設，當您在主控台建立追蹤記錄時，追蹤記錄會套用到所有 AWS 區域。線索會記錄 AWS 分割區中所有區域的事件，並將日誌檔案交付至您指定的 Amazon S3 儲存貯體。此外，您可以設定其他 AWS 服務，以進一步分析和處理 CloudTrail 日誌中所收集的事件資料。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務和整合
- 設定 CloudTrail 的 Amazon SNS 通知

- 從多個區域接收 CloudTrail 日誌檔案
- 從多個帳戶接收 CloudTrail 日誌檔案

CloudTrail 會記錄所有 AWS 大型主機現代化動作，並記錄在AWS 大型主機現代化 API 參考中。例如，對 CreateApplication、CreateEnvironment 和 CreateDeployment 動作發出的呼叫會在 CloudTrail 記錄檔案中產生專案。

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：

- 該請求是使用根使用者還是使用者憑證提出。
- 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 userIdentity 元素。

了解 AWS Mainframe Modernization 日誌檔案項目

追蹤是一種組態，能讓事件以日誌檔案的形式交付到您指定的 Amazon S3 儲存貯體。CloudTrail 日誌檔案包含一或多個日誌專案。一個事件為任何來源提出的單一請求，並包含請求動作、請求的日期和時間、請求參數等資訊。CloudTrail 日誌檔並非依公有 API 呼叫的堆疊追蹤排序，因此不會以任何特定順序出現。

以下範例顯示的是展示 CreateApplication 動作的 CloudTrail 日誌項目。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAII6WZTHGYAEXAMPLE",
        "arn": "arn:aws:sts::444455556666:assumed-role/Admin/Mary_Major",
        "accountId": "444455556666",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAII6WZTHGYAEXAMPLE",
                "arn": "arn:aws:iam::444455556666:role/Admin",
                "accountId": "444455556666",
                "userName": "Admin"
```

```
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-06-01T20:38:22Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-06-01T20:40:39Z",
    "eventSource": "m2.amazonaws.com",
    "eventName": "CreateApplication",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.196.65",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:91.0) Gecko/20100101
 Firefox/91.0",
    "requestParameters": {
        "clientToken": "1abc23de-f45g-6789-h01i-jkl2m3456789",
        "name": "MyApp",
        "description": "",
        "engineType": "microfocus",
        "definition": {
            "content": "{}"
        },
        "tags": {}
    },
    "responseElements": {
        "applicationVersion": 1,
        "Access-Control-Expose-Headers": "x-amzn-RequestId,x-amzn-ErrorType,x-amzn-
ErrorMessage,Date",
        "applicationArn": "arn:aws:m2:us-east-1:444455556666:app/
lsfhmwhw7fffrosff2lncwqcua",
        "applicationId": "lsfhmwhw7fffrosff2lncwqcua"
    },
    "requestID": "36982d38-fcde-4bfe-a89a-7bd78d43c926",
    "eventID": "d7f0fc36-46ae-4157-9a79-c79f385fda98",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "444455556666",
    "eventCategory": "Management"
}
```

AWS 大型主機現代化中的故障診斷

使用本節中的資訊，協助您使用 AWS Blu Age 和 Rocket Software 引擎，對 AWS 大型主機現代化應用程式和執行時間環境中的常見錯誤進行故障診斷。

主題

- 故障診斷錯誤：等待資料集名稱解除鎖定時逾時
- 故障診斷錯誤：無法存取應用程式 URL
- 故障診斷： AWS Blu Insights 不會從主控台開啟
- 故障診斷錯誤：環境運作狀態不佳
- 針對 Rocket Software （ 先前稱為 Micro Focus) 的授權問題進行故障診斷

故障診斷錯誤：等待資料集名稱解除鎖定時逾時

此頁面說明當您看到環境中的另一個應用程式在共用資料集上保持鎖定時，如何解決錯誤。

- Engine： AWS Blu 存留期
- 元件：Blusam

如果您在 AWS 大型主機現代化應用程式的 Amazon CloudWatch logs中看到此錯誤，使用 AWS Blu Age 引擎並在具有高可用性模式的環境中執行，則表示另一個應用程式正在鎖定共用資料集。一般而言，如果其他應用程式當機或失敗，且 不會釋放鎖定，就會發生這種情況。

尋找失敗的應用程式，並檢查其是否使用錯誤訊息中提到的相同資料集。檢查應用程式是否在具有高可用性模式的執行期環境中執行。引發逾時例外狀況的應用程式無法繼續，並會顯示`Failed`狀態。

常見原因

應用程式`example-app-1`會嘗試鎖定寫入操作`example-record-1`的記錄。此操作會在擁有`example-dataset-1`的資料集 上建立鎖定`example-record-1`，並在`example-record-1`本身建立鎖定。現在另一個應用程式 `example-app-2`嘗試鎖定相同的記錄 `example-record-1`。資料集和記錄已經鎖定，因此 會`example-app-2`等待鎖定釋放。如果當`example-app-1`機，資料集上的保留鎖定`example-dataset-1`仍然存在，這會導致 `example-app-2` 取消其寫入嘗試並引發逾時例外狀況。此死結情況可防止所有應用程式到達 `example-dataset-1`。

Resolution

若要立即解決這種情況，您可以強制釋放鎖定。若要防止未來發生類似情況，您可以設定兩個參數來控制 Blusam 自動修復機制。

強制鎖定釋放

Blusam 鎖定管理員使用 Amazon ElastiCache (Redis OSS) 在應用程式之間提供共用鎖定。若要在 ElastiCache 中釋出鎖定，請使用 Redis CLI 公用程式。您無法刪除個別記錄鎖定。您必須從擁有的資料集移除所有鎖定。請完成下列步驟：

1. 使用下列命令連線至 ElastiCache：

   ```
   redis-cli -h hostname -p port
   ```

   您可以在 ElastiCache 主控台中找到 ElastiCache 的詳細資訊，網址為 https：//https://console.aws.amazon.com/elasticache/。
2. 輸入您的密碼。
3. 輸入您要執行的命令，如下所示：

Command	用途
KEYS *	取得所有現有的金鑰。
KEYS *YOUR_DATASET_NAME*	取得資料集鎖定金鑰。
DEL *THE_RETURNED_KEY*	刪除資料集鎖定。
FLUSHDB	清理整個 Redis。 **Warning** Redis 快取中的所有資料都會遺失。如果 Redis 用於其他用途，例如處理 http 工作階段，您可能不想使用 FLUSHDB。

設定 Blusam 自動修復機制

Blusam locks Manager 包含自動修復機制，可防止資料集或記錄發生死結。您可以在應用程式定義 (`application-main.yml`) 中調整下列參數，以設定自動修復機制：

- `locksDeadTime`： 是指應用程式可以保留鎖定的時間上限。當這段時間過後，鎖定會宣告過期並立即釋放。`locksDeadTime` 值以毫秒為單位，預設值為 1000。
- `locksCheck`：定義用於檢查鎖定的 Blusam 鎖定管理員策略。ElastiCache 中的所有 Blusam 鎖定都會加上時間戳記，並具有過期時間。`locksCheck` 參數值會判斷是否移除過期的鎖定。
  - `off`：隨時都不會執行檢查。可能會發生死鎖。(不建議使用)
  - `reboot`：當在 AWS Mainframe Modernization 執行時間環境中執行 AWS 的 Mainframe Modernization 應用程式執行個體啟動或重新啟動時，會執行 檢查。所有過期的鎖定都會立即釋出。（預設）
  - `timeout`：當在 AWS Mainframe Modernization 執行時間環境中執行 AWS 的 Mainframe Modernization 應用程式執行個體啟動或重新啟動，或在嘗試鎖定資料集期間逾時過期時，會執行檢查。過期的鎖定會立即釋放。

如需 AWS Blu Age 應用程式之應用程式定義的詳細資訊，請參閱 AWS Blu Age 應用程式定義範例。

Blusam 鎖定管理員

在使用高可用性模式 AWS 的大型主機現代化執行期環境環境中， AWS Blu Age 應用程式可能會部署多次。對於處理 Blusam 資料集的應用程式，可能會發生並行存取問題。Blusam 鎖定管理員透過使用 ElastiCache 提供應用程式之間的共用鎖定，確保資料完整性並管理對記錄和資料集的讀取和寫入存取。此機制允許多個應用程式同時讀取記錄，並確保一次只有一個應用程式寫入記錄。

寫入鎖定

若要更新或刪除特定記錄，應用程式必須先鎖定擁有記錄的資料集，然後鎖定記錄本身。當記錄鎖定時，資料集鎖定會釋出，且相同資料集的其他記錄可供使用。當更新或刪除操作完成時，保留的記錄鎖定會釋出。如果定義的應用程式政策允許等待發行，則一次只能有一個應用程式可以更新記錄，這會封鎖其他應用程式讀取或寫入，直到鎖定解除為止。

讀取鎖定

只要記錄或資料集上沒有寫入鎖定，多個應用程式就可以同時讀取相同的記錄。若要鎖定寫入操作的記錄，必須釋放所有讀取鎖定。

> **Note**
>
> Blusam locks Manager 會使用相同的鎖定機制，處理指定應用程式中多個執行緒的存取。

故障診斷錯誤：無法存取應用程式 URL

此頁面說明如何在無法存取執行 AWS 中 Mainframe Modernization 應用程式的 URL 時解決錯誤。

- Engine： AWS Blu Age and Rocket Software （ 先前稱為 Micro Focus)
- 元件：應用程式

如果您無法存取您建立並部署至 AWS Mainframe Modernization 執行期環境之執行 AWS 中 Mainframe Modernization 應用程式的 URL，您可能需要在與執行期環境相關聯的安全群組上設定傳入規則。

常見原因

當您建立執行時間環境時，您提供的安全群組，包括預設安全群組，必須設定傳入規則，以允許從 VPC 外部對已部署應用程式的流量，如果您想要允許此類型的存取。

Resolution

檢查與執行時間環境相關聯的 Amazon VPC 安全群組是否允許適當應用程式連接埠上的環境流量。若要檢查安全群組規則，請完成下列步驟：

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 在左側導覽中，選擇環境。
3. 選擇託管您要連線之應用程式的執行時間環境。
4. 選擇組態。
5. 在安全與網路中，選擇安全群組。此連結會在 Amazon VPC 主控台中開啟安全群組的詳細資訊。
6. 如有必要，請選擇編輯傳入規則，如果尚未存在，請選擇新增下列規則：

   Type

   自訂 TCP

連線埠

8196 或符合應用程式定義中指定接聽程式屬性的連接埠。如需詳細資訊，請參閱步驟 2：建立應用程式定義。

來源

您呼叫應用程式的 IP 地址。您可以從下拉式清單中選擇 myIP。如果您仍然遇到逾時問題，請嘗試選擇 Anywhere IPV4 或 Anywhere IPV6。在安全群組上新增傳入規則後，請務必停止應用程式並再次啟動。

如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的使用安全群組規則。

故障診斷： AWS Blu Insights 不會從主控台開啟

此頁面說明如何解決不從 AWS 大型主機現代化主控台開啟的 Blu Insights 頁面。

- Engine： AWS Blu Age
- 元件：Blu Insights

當您嘗試從 AWS Mainframe Modernization 主控台存取 Blu Insights 時，不會開啟，且新標籤會立即關閉。

常見原因

您用來存取 Blu Insights 的角色沒有足夠的許可。

Resolution

將 IAM 政策連接至角色，以允許其存取 Blu Insights。請確定政策至少包含下列許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "m2:GetSignedBluinsightsUrl"
            ],
```

```
            "Resource": "*"
            }
    ]
}
```

請務必account使用正確的 region和 取代 AWS 區域 和 AWS 帳戶。

故障診斷錯誤：環境運作狀態不佳

此頁面說明當您收到通知指出其中一個 AWS 大型主機現代化環境運作狀態不佳時，如何解決錯誤。

- Engine： AWS Blu Age and Rocket Software （ 先前稱為 Micro Focus)
- 元件：環境

如果您收到通知，指出您的其中一個 AWS 大型主機現代化環境運作狀態不佳，則這適用於您。您會收到以下其中一個來源的通知：

- 運作狀態不佳的環境狀態會顯示在 AWS Mainframe Modernization 主控台中。
- 有關 環境狀態不良的電子郵件通知 AWS Health。
- 您可以在您的 帳戶運作狀態下，於 AWS Health 儀表板中看到 AWS 來自 Mainframe Modernization 的相關事件。

常見原因

當您 AWS 帳戶中與 AWS Mainframe Modernization 環境相關聯的資源無法存取時，就會發生錯誤。此問題的常見原因是正在修改或刪除與環境相關的資源。

Resolution

如需特定指引，請使用 AWS Health 電子郵件中提供的錯誤代碼，或透過AWS 您的 Mainframe Modernization 主控台提供。

錯誤碼：

- 儲存體無法連線

此錯誤表示環境的連接儲存體 (Amazon Elastic File System 或 Amazon FSx 檔案系統） 無法正確掛載。若要檢查運作狀態不佳環境的詳細資訊，請完成下列步驟：

1. 開啟 AWS Mainframe Modernization 主控台，網址為 https：//https://console.aws.amazon.com/m2/。
2. 選取運作狀態不佳的環境，然後選擇組態。
3. 選擇連接儲存以檢視與此環境相關聯的儲存資源。
4. 檢查與網路相關的組態，例如與儲存相關聯的安全群組、子網路和 Amazon VPC。如果這些組態不正確，請嘗試將其還原以解決此問題。

> **Note**
>
> 如果已刪除儲存體，則無法復原環境。在這種情況下，您應該考慮刪除運作狀態不佳的環境。

針對 Rocket Software （ 先前稱為 Micro Focus) 的授權問題進行故障診斷

此頁面說明如何解決 Rocket 軟體執行期引擎的授權問題

- 引擎：Rocket 軟體
- 元件：Amazon EC2

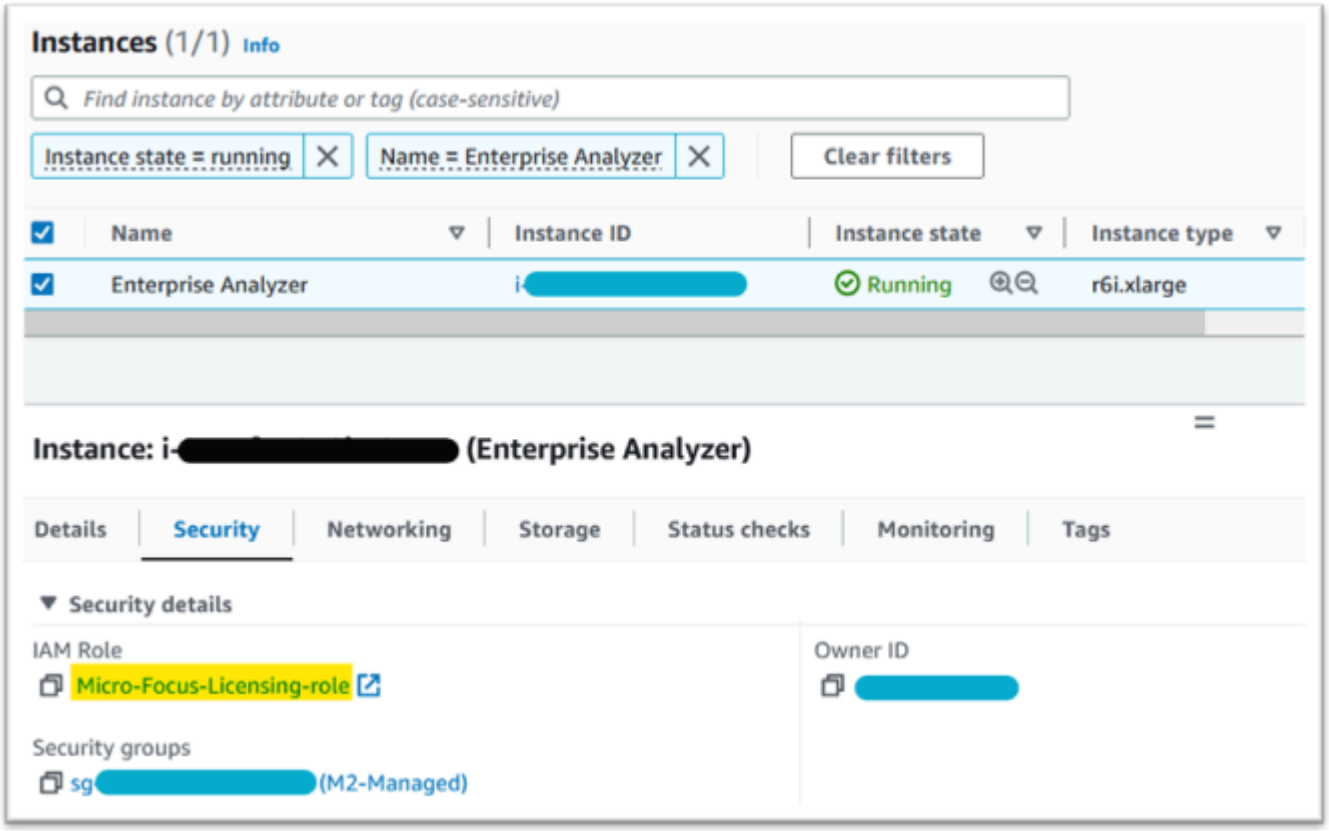
如果您在存取或使用 AMIs時遇到問題，下列資訊可能可協助您。

主題

- 確認 Amazon EC2 執行個體具有 IAM 授權角色
- 使用連線能力分析器
- 執行 license-daemon
- 作業系統修補後，Linux 上的 Enterprise Server 或 Enterprise Build Tools 的授權問題

確認 Amazon EC2 執行個體具有 IAM 授權角色

這可以在 Amazon EC2 執行個體詳細資訊的安全索引標籤上檢查。這可以使用動作下拉式選單的安全選項進行變更。



使用連線能力分析器

在 AWS Network Manager 主控台頁面上尋找 Reachability Analyzer。

在從 AMI 建立的 Amazon EC2 執行個體與 Amazon S3 VPC 端點之間建立和分析路徑。

如果 Amazon EC2 執行個體沒有網際網路存取，請重複對全部 4 個端點的路徑分析。

如需 Reachability Analyzer 的詳細資訊，請參閱 Reachability Analyzer 指南中的 Reachability Analyzer 入門。

執行 license-daemon

在 Windows Enterprise Developer 上，從命令提示字元使用下列命令：

```
“C:\Program Files (x86)\Micro Focus\Enterprise Developer\AdoptOpenJDK\bin\java” -jar
 "C:\Program Files (x86)\Micro Focus\Licensing\aws-license-daemon.jar"
```

並檢查輸出。忽略 SLF4J 訊息並尋找第一個例外狀況。

在 Enterprise Analyzer 上，從命令提示字元使用以下命令：

```
"C:\Program Files (x86)\Micro Focus\AdoptOpenJDK\bin\java" -jar "C:\Program Files
 (x86)\Micro Focus\Licensing\aws-license-daemon.jar"
```

並檢查輸出。忽略 SLF4J 訊息並尋找第一個例外狀況。

在 Linux 執行時：

```
java -jar /var/microfocuslicensing/bin/aws-license-daemon.jar
```

忽略 SLF4J 訊息並尋找第一個例外狀況。

例如，如果 Amazon S3 資源不可用，則例外狀況如下：

```
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.

Exception in thread "main" software.amazon.awssdk.services.s3.model.S3Exception: Access
 Denied (Service: S3, Status Code: 403, Request ID: P6
```

例外狀況訊息指出無法使用的資源。將組態值與本主題中顯示的值進行比較。

作業系統修補後，Linux 上的 Enterprise Server 或 Enterprise Build Tools 的授權問題

如果您在作業系統修補後在 Linux 上遇到 Enterprise Server 或 Enterprise Build Tools 的授權問題，請下載並執行修補程式指令碼來更新授權協助程式。若要這樣做，請在命令提示中使用下列命令：

```
sudo curl https://d148y999krizvm.cloudfront.net/patch/v8/linux/patch.sh -o /var/
microfocuslicensing/bin/patch.sh
sudo chmod +x /var/microfocuslicensing/bin/patch.sh
sudo /var/microfocuslicensing/bin/patch.sh
sudo ./startmfcesd.sh
```

> **Note**
>
> 即使下載路徑適用於第 8 版，此修補程式指令碼也適用於第 9 版。

AWS Mainframe Modernization 使用者指南的文件歷史記錄

下表說明 AWS Mainframe Modernization 的文件版本。

變更	描述	日期
AWS Blu Age 版本備註 4.7.0 和 4.8.0	4.7.0 版的 AWS Blu Age Runtime 和 AWS Blu Age Transformation Engines 包含 Spring Boot 3.4.2 和 Angular 19 的核心相依性升級，可增強安全性和效能，同時現代化前端應用程式架構。4.8.0 版包含 IBM IMS 資料庫轉換支援等重要功能，以及改善 AS400 現代化資料密集型程式效能的新記憶體內快取功能。	2025 年 5 月 18 日
AWS Blu Age 版本備註 4.6.0	此版本的 AWS Blu Age Runtime 和 AWS Blu Age 轉換引擎著重於 zOS 和 AS400 的新功能和改進。	2025 年 1 月 24 日
AWS Blu Age 版本備註 4.5.0	此版本的 AWS Blu Age Runtime 和 AWS Blu Age 轉換引擎著重於 JCL 支援的重要功能、AS/400 現代化應用程式的繫結目錄和啟用群組支援，以及更新的相依性。	2024 年 12 月 20 日
AWS Blu Age 常見問答集	透過此完整的FAQs清單，了解 AWS Blu Age Refactoring 功能。	2024 年 12 月 20 日
適用於大型主機的 Amazon Q Developer 轉換	您可以了解 Amazon Q Developer 的大型主機轉換功能，這些功能可讓您將舊版	2024 年 12 月 2 日

	COBOL 大型主機應用程式現代化為 Java 應用程式。	
AWS Blu Age 版本備註 4.4.0	此版本的 AWS Blu Age Runtime 和轉型引擎著重於升級關鍵相依性和支援的技術，同時提升多種功能的效能。	2024 年 11 月 13 日
在容器上部署 AWS Blu Age 執行期	您可以了解如何在容器上設定 AWS Blu Age Runtime，以在 Amazon ECS （由 Amazon EC2 或 管理 AWS Fargate) 和 Amazon EC2 管理的 Amazon EKS 上部署。	2024 年 10 月 28 日
已更新 LDAP 使用者登入資料	您現在可以使用 AWS 主控台或 AWS CLI （或 SDK) 建立和管理 LDAP 使用者憑證以進行身分驗證和授權。	2024 年 10 月 21 日
設定 Rocket Software 受管應用程式	您現在可以使用 Rocket Software 執行期引擎來設定應用程式，以自訂其他屬性，包括整合。	2024 年 10 月 21 日
使用 NTT DATA Unikix 執行時間進行複寫	了解如何使用 Amazon Machine Image (AMIs) 建立量身打造的環境， AWS 以使用 NTT DATA 在 中重新託管和重建大型主機應用程式。	2024 年 9 月 25 日
AWS 大型主機現代化應用程式測試 IAM	您可以了解如何使用可用的 IAM 功能和政策來管理 AWS 大型主機現代化應用程式測試的存取權。	2024 年 9 月 20 日

AWS Blu Age 版本	AWS Mainframe Modernization 使用者指南的本節會擷取所有 AWS Blu Age 版本控制詳細資訊、 AWS Blu Age 版本備註、 AWS Blu Age 升級指示，以及整個 AWS Blu Age 生命週期。	2024 年 9 月 16 日
AWS 大型主機現代化元件生命週期	此頁面會擷取每個 AWS 大型主機現代化元件的生命週期，包括其版本升級、整體發行計劃，以及終止支援和淘汰計劃。	2024 年 9 月 5 日
使用 mLogica 進行組裝器轉換	AWS 使用 mLogica 進行大型主機現代化程式碼轉換是一種 AWS 大型主機現代化功能，可自動將 z/OS 大型主機組合程式碼轉換為 COBOL。	2024 年 7 月 22 日
應用程式測試 GA 版本	Application Testing 的一般可用性文件。 AWS Mainframe Modernization Application Testing 為您的遷移專案提供自動化的功能等效性測試。此版本包含資料保護頁面、主控台工作流程，以及自預覽以來其他文件頁面的更新。	2024 年 6 月 12 日
已更新 Rocket Softwaretutorial 的受管執行期	本教學課程說明如何使用 Rocket Software 執行期引擎，在 AWS 大型主機現代化受管執行期環境中部署和執行 CardDemo 範例應用程式。	2024 年 2 月 5 日

AWS Blu Age 執行期和現代化工具 3.9.0 版的版本備註。	此版本的 AWS Blu Age Runtime 和 Modernization Tools 專注於跨產品的多個橫向增強功能，努力提高高可用性架構的效能，以及將任務執行提升到下一個層級的新功能。	2023 年 12 月 18 日
在大型主機和 之間傳輸檔案 AWS	發佈新功能，以將檔案從來源大型主機傳輸到 AWS。	2023 年 11 月 27 日
管理應用程式的交易	發佈新功能，以顯示和編輯 AWS 大型主機現代化應用程式的交易。	2023 年 10 月 16 日
AWS Blu Age Runtime and Modernization Tools 3.6.0 版的版本備註。	此版本的 AWS Blu Age Runtime 和 Modernization Tools 為 zOS 和 AS400 舊版遷移提供了新功能，主要用於擴展 CICS 支援機制、補充 JCL 功能、最佳化並行和大量功能的效能，以及新增multi-data-source功能。	2023 年 8 月 4 日
您現在可以在應用程式停止時部署應用程式的新版本。	先前，若要部署應用程式的新版本，您必須刪除已部署的版本。現在，您可以停止已部署的版本並部署新版本。	2023 年 7 月 26 日
AWS 封裝的 Blu Age 執行時間，讓 Amazon EC2 部署更輕鬆	AWS Mainframe Modernization with AWS Blu Age 執行期現在可在 Amazon EC2 執行個體上更靈活地設定完整堆疊和部署 AWS 帳戶。	2023 年 7 月 6 日
單一登入 AWS Blu Age Blu Insights。	AWS Blu Age Blu Insights 可透過單一登入 AWS Management Console 從 取得。	2023 年 3 月 31 日

GA 版本	AWS Mainframe Modernization 使用者指南的 GA 版本。	2022 年 6 月 8 日
初始版本	AWS Mainframe Modernization 使用者指南的初始版本（公開預覽版）。	2021 年 11 月 30 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。