使用者指南

AWS OpsWorks

API 版本 2013-02-18

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

AWS OpsWorks: 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、任從何貶低或使 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，這些擁有者可能隸屬於 Amazon，或與 Amazon 有合作關係，或由 Amazon 贊助。

Table of Contents

什麼是 AWS OpsWorks？ 1
AWS OpsWorks 服務 1
AWS OpsWorks for Puppet Enterprise 4
區域支援 Puppet Enterprise 的 OpsWorks 5
生命週期結束FAQs 6
現有客戶將受到此生命週期結束的影響？ 6
如果我不採取任何動作，我的伺服器會發生什麼情況？ 6
是否 AWS OpsWorks for Puppet Enterprise 接受新客戶？ 6
生命週期結束會 AWS 區域 同時影響所有 嗎？ 7
提供哪些層級的技術支援 AWS OpsWorks for Puppet Enterprise？ 7
我是 OpsWorks for Puppet Enterprise 的目前客戶，我需要在先前未使用此服務的帳戶中啟動伺服器。我是否可以這麼做？ 7
是否有任何新的功能版本 AWS OpsWorks for Puppet Enterprise？ 7
開始使用 7
先決條件 8
建立 Puppet 主伺服器 11
尋找組態 23
新增要管理的節點 27
登入 Puppet Enterprise 主控台 29
選用：使用 CodeCommit 33
在 CloudFormation 中建立 Puppet 主伺服器 39
先決條件 40
在 AWS CloudFormation中建立 Puppet Enterprise 主伺服器 40
更新伺服器以使用自訂網域 46
先決條件 46
限制 46
更新伺服器以使用自訂網域 47
另請參閱 50
處理標籤 50
標籤在 中的運作方式 AWS OpsWorks for Puppet Enterprise 51
在 OpsWorks for Puppet Enterprise 中新增和管理標籤 （主控台） 52
在 OpsWorks for Puppet Enterprise (CLI) 中新增和管理標籤 55
另請參閱 59
備份與還原伺服器 60

備份 Puppet Enterprise Server 的 OpsWorks 60
還原 Puppet Enterprise Server 的 OpsWorks 64
系統維護 65
設定系統維護 66
隨需啟動系統維護 68
在維護之後還原自訂組態和檔案 68
自動新增節點 69
步驟 1：建立 IAM 角色以用作您的執行個體描述檔 69
步驟 2：使用自動關聯指令碼建立執行個體 70
移除節點 71
另請參閱 72
刪除 Puppet 主伺服器 72
步驟 1：取消與受管節點的關聯 73
步驟 2：刪除伺服器 73
另請參閱 73
將 Puppet 伺服器遷移至 Amazon EC2 74
步驟 1：聯絡 Puppet 購買授權 74
步驟 2：取得 OpsWorks for Puppet Enterprise 伺服器的詳細資訊 74
步驟 3：備份您的 OpsWorks for Puppet Enterprise 伺服器 75
步驟 4：啟動新的 EC2 執行個體 76
步驟 5：在新的 EC2 執行個體上安裝 Puppet Enterprise 77
步驟 6：在新的 EC2 執行個體上還原備份 78
步驟 7：設定您的 Puppet 授權 78
步驟 8：遷移節點 78
步驟 9：刪除 Puppet Enterprise 伺服器的 OpsWorks 81
使用 AWS CloudTrail 81
CloudTrail 中 Puppet Enterprise 資訊的 OpsWorks 82
了解 Puppet Enterprise Log File 項目的 OpsWorks 83
故障診斷 85
一般疑難排解秘訣 85
針對特定錯誤進行故障診斷 85
其他協助及支援 90
AWS OpsWorks for Chef Automate 91
的區域支援 AWS OpsWorks for Chef Automate 93
生命週期結束FAQs 94
現有使用者將如何受到此生命週期結束的影響？ 94

如果我不採取任何動作，我的伺服器會發生什麼情況？ 94
我可以轉換到哪些替代方案？ 95
服務是否仍然接受新客戶？ 95
生命週期結束會 AWS 區域 同時影響所有 嗎？ 95
提供何種程度的技術支援？ 95
我是 OpsWorks for Chef Automate 的目前客戶，我需要在先前未使用此服務的帳戶中啟動伺服器。我是否可以這麼做？ 95
明年是否會有任何主要功能版本？ 95
升級至 Chef Automate 2 96
升級至 Chef Automate 2 的先決條件 96
關於升級程序 96
升級至 Chef Automate 2 (主控台) 97
升級至 Chef Automate 2 (CLI) 97
將 AWS OpsWorks for Chef Automate 伺服器復原至 Chef Automate 1 (CLI) 98
另請參閱 99
開始使用 99
先決條件 100
建立 Chef Automate 伺服器 102
完成組態並上傳技術指南 114
新增要管理的節點 122
登入 Chef Automate 儀表板 128
在 CloudFormation 中建立 Chef Automate 伺服器 131
先決條件 132
在 AWS CloudFormation中建立 Chef Automate 伺服器 133
更新伺服器以使用自訂網域 139
先決條件 139
限制 46
更新伺服器以使用自訂網域 47
另請參閱 50
重新產生入門套件 143
使用 重新產生 AWS OpsWorks for Chef Automate 入門套件 AWS CLI 144
處理標籤 145
標籤在 中的運作方式 AWS OpsWorks for Chef Automate 146
在 AWS OpsWorks for Chef Automate （主控台） 中新增和管理標籤 147
在 AWS OpsWorks for Chef Automate (CLI) 中新增和管理標籤 149
另請參閱 154

備份與還原伺服器 154
備份 AWS OpsWorks for Chef Automate 伺服器 155
還原 AWS OpsWorks for Chef Automate 伺服器 157
系統維護 158
確保節點信任 AWS OpsWorks 憑證授權機構 159
設定系統維護 160
隨需啟動系統維護 162
在維護之後還原自訂組態和檔案 162
合規掃描 163
使用 Chef Automate 2.0 中的合規 164
使用 Chef Automate 1.x 中的合規 170
合規更新 176
社群和自訂合規描述檔 176
另請參閱 177
移除節點 177
相關主題 178
刪除 Chef Automate 伺服器 178
步驟 1：取消與受管節點的關聯 179
步驟 2：刪除伺服器 179
重設 Chef 登入資料 179
使用 AWS CloudTrail 181
AWS OpsWorks for Chef Automate CloudTrail 中的資訊 181
了解 AWS OpsWorks for Chef Automate 日誌檔案項目 182
疑難排解 184
一般疑難排解秘訣 184
對特定錯誤進行故障診斷 185
其他協助及支援 191
AWS OpsWorks 組態管理 (CM) 中的安全性 192
資料保護 192
與 整合 AWS Secrets Manager 194
資料加密 194
靜態加密 194
傳輸中加密 194
金鑰管理 195
身分和存取權管理 195
目標對象 195

使用身分來驗證 196
使用政策管理存取權 198
CM AWS OpsWorks 如何與 IAM 搭配使用 200
身分型政策範例 205
故障診斷 208
AWS 受管政策 209
中的跨服務混淆代理預防 AWS OpsWorks CM 217
網際網路流量隱私權 220
記錄和監控 220
合規驗證 220
恢復能力 221
基礎設施安全性 222
組態與漏洞分析 222
安全最佳實務 223
AWS OpsWorks 堆疊 224
堆疊 227
層 227
配方和生命週期 (LifeCycle) 事件 227
執行個體 228
應用程式 229
自訂您的 Stack 229
資源管理 230
安全與許可 230
監控和記錄 230
CLI、軟體開發套件和 AWS CloudFormation 範本 231
生命週期結束FAQs 231
現有客戶將受到此生命週期結束的影響？ 232
是否 AWS OpsWorks Stacks 接受新客戶？ 232
我應該將現有堆疊遷移到哪裡？ 232
如何在生命週期結束後保留現有的 Amazon EC2 執行個體？ 233
生命週期結束會 AWS 區域 同時影響所有 嗎？ 233
提供哪些層級的技術支援 AWS OpsWorks Stacks？ 233
是否有任何新的功能版本 AWS OpsWorks Stacks？ 233
將您的應用程式遷移至 Systems Manager Application Manager 233
指令碼的運作方式 234
先決條件 234

限制 ... 235
開始使用 ... 235
常見問答集 ... 248
故障診斷 ... 258
使用 AWS OpsWorks Stacks 分離就地工具 ... 259
程序的運作方式 ... 260
限制 ... 261
開始使用 ... 262
開始使用 ... 269
區域支援 ... 270
入門：範例 ... 271
入門：Linux ... 290
入門：Windows ... 319
入門：技術指南 ... 349
最佳實務 ... 380
根設備儲存 ... 380
最佳化應用程式伺服器的數目 ... 382
管理許可 ... 385
管理和部署應用程式與技術指南 ... 387
本機封裝技術指南依存性 ... 395
堆疊 ... 399
從 EC2-Classic 遷移堆疊 ... 400
建立新的堆疊 ... 402
在 VPC 中執行堆疊 ... 409
更新堆疊 ... 420
複製堆疊 ... 421
執行堆疊命令 ... 422
使用自訂 JSON ... 425
刪除堆疊 ... 427
層 ... 431
OpsWorks Layer 基本概念 ... 432
Elastic Load Balancing Layer ... 446
Amazon RDS 服務層 ... 450
ECS 叢集層 ... 455
自訂 Layer ... 461
個別 layer 套件安裝 ... 462

執行個體 463
使用 AWS OpsWorks Stacks 執行個體 464
使用在 AWS OpsWorks Stacks 之外建立的運算資源 518
編輯執行個體組態 560
刪除 AWS OpsWorks Stacks 執行個體 562
使用 SSH 登入 563
使用 RDP 登入 566
應用程式 570
新增應用程式 571
部署應用程式 577
編輯應用程式 581
連線至資料庫 582
使用 環境變數 583
傳遞資料到應用程式 585
使用 Git 儲存庫 SSH 金鑰 588
使用自訂網域 589
使用 SSL 591
技術指南和配方 598
技術指南儲存庫 599
Chef 版本 602
Ruby 版本 618
安裝自訂技術指南 619
更新自訂技術指南 622
執行配方 625
資源管理 632
向堆疊註冊資源 633
連接與移動資源 638
分離資源 644
取消註冊資源 646
標籤 649
在堆疊層級設定標籤 650
在 Layer 層級設定標籤 651
使用 管理標籤 AWS CLI 653
標籤限制 654
監控 654
使用 Amazon CloudWatch 655

使用 AWS CloudTrail 665
使用 Amazon CloudWatch Logs 668
使用 Amazon CloudWatch Events 672
安全與許可 673
管理使用者許可 675
允許 AWS OpsWorks Stacks 代表您 696
預防混淆代理人 701
指定在 EC2 執行個體上執行之應用程式的許可 704
管理 SSH 存取 708
管理安全性更新 715
使用安全群組 716
Chef 12 Linux 719
概觀 719
移至 Chef 12 720
支援的作業系統 721
支援的執行個體類型 721
詳細資訊 722
移至資料包 722
先前的 Chef 版本 724
適用於 Linux 的 Chef 11.10 和較舊版本 724
搭配其他 AWS 服務使用 AWS OpsWorks Stacks 1110
使用後端資料存放區 1111
ElastiCache Redis 1118
使用 Amazon S3 儲存貯體 1131
AWS CodePipeline 搭配 Stacks AWS OpsWorks 使用 1144
使用 AWS OpsWorks Stacks CLI 1202
建立 執行個體 1204
部署應用程式 1207
列出應用程式 1208
清單命令 1209
列出部署 1210
列出彈性 IP 地址 1211
列出執行個體 1212
列出堆疊 1214
列出 Layer 1215
執行配方 1219

安裝相依性 1220
更新堆疊組態 1221
偵錯和故障診斷指南 1222
除錯配方 1223
常見的除錯和故障診斷問題 1238
AWS OpsWorks Stacks 代理程式 CLI 1247
agent_report 1249
get_json 1250
instance_report 1254
list_commands 1255
run_command 1256
show_log 1257
stack_state 1258
AWS OpsWorks Stacks Data Bag 參考 1260
應用程式資料包 (aws_opsworks_app) 1264
命令資料包 (aws_opsworks_command) 1267
Amazon ECS 叢集資料包 (aws_opsworks_ecs_cluster) 1269
Elastic Load Balancing Data Bag (aws_opsworks_elastic_load_balancer) 1270
執行個體資料包 (aws_opsworks_instance) 1271
Layer 資料包 (aws_opsworks_layer) 1276
Amazon RDS Data Bag (aws_opsworks_rds_db_instance) 1278
堆疊資料包 (aws_opsworks_stack) 1280
使用者資料包 (aws_opsworks_user) 1282
OpsWorks 代理程式變更 1283
Chef 12 代理版本 1283
Chef 11.10 代理版本 1286
資源 1292
參考指南、工具和支援資源 1292
AWS 軟體開發套件 1293
開放原始碼軟體 1293
AWS OpsWorks 文件歷史記錄 1294
舊版更新 1300
..... mcccciii

什麼是 AWS OpsWorks？

> **Important**
>
> AWS OpsWorks 服務已達到生命週期結束，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks 是一種組態管理服務，可協助您使用 Puppet 或 Chef. AWS OpsWorks Stacks 在雲端企業中設定和操作應用程式，並 AWS OpsWorks for Chef Automate 可讓您使用 Chef 技術指南和解決方案進行組態管理，而 OpsWorks for Puppet Enterprise 可讓您在其中設定 Puppet Enterprise 主伺服器 AWS。Puppet 提供一組工具，用於強制執行所需的基礎設施狀態，並自動執行隨需任務。

AWS OpsWorks 服務

AWS OpsWorks for Puppet Enterprise

OpsWorks for Puppet Enterprise 可讓您建立 AWS受管 Puppet 主伺服器。Puppet 主伺服器管理基礎設施中的節點、存放有關這些節點的事實，並做為 Puppet 模組的中央儲存庫使用。Puppet 模組是可重複使用且可共享的 Puppet 程式碼單位，包含如何設定基礎設施的說明。您可以從 Puppet Forge 下載社群模組，或使用 Puppet Development Kit 建立您自己的自訂模組，然後使用 Puppet Code Manager 管理其部署。

OpsWorks for Puppet Enterprise 提供全受管 Puppet 主控端，一套自動化工具，可讓您檢查、交付、操作和未來驗證您的應用程式，以及存取使用者介面，讓您檢視節點和 Puppet 活動的相關資訊。OpsWorks for Puppet Enterprise 可讓您使用 Puppet 自動設定、部署和管理節點的方式，無論是 Amazon EC2 執行個體還是內部部署裝置。OpsWorks for Puppet Enterprise 主控端透過處理軟體和作業系統組態、套件安裝、資料庫設定、變更管理、政策強制執行、監控和品質保證等任務，提供全堆疊自動化。

由於 OpsWorks for Puppet Enterprise 會管理 Puppet Enterprise 軟體，因此您的伺服器可以在您選擇的時間自動備份、一律執行與 Puppet 相容的最新版本，以及一律套用最新的安全更新。您可以使用 Amazon EC2 Auto Scaling 群組自動將新的 Amazon EC2 節點與您的伺服器建立關聯。

AWS OpsWorks for Chef Automate

AWS OpsWorks for Chef Automate 可讓您建立包含 Chef Automate AWS進階功能的受管 Chef 伺服器，並使用 Chef DK 和其他 Chef 工具來管理它們。Chef 伺服器會管理環境中的節點、存放這些節點的資訊，並做為 Chef 技術指南的中央儲存庫使用。技術指南包含在您使用 Chef 所管理的每個節點上，由 Chef Infra 用戶端 `chef-client` 代理程式執行的配方。您可以使用 Chef 工具和 `knife` Test Kitchen 等工具來管理 AWS OpsWorks for Chef Automate 服務中 Chef 伺服器上的節點和技術指南。

Chef Automate 是包含的伺服器軟體套件，可提供自動化工作流程以進行持續部署和合規檢查。AWS OpsWorks for Chef Automate 會使用單一 Amazon Elastic Compute Cloud 執行個體安裝和管理 Chef Automate、Chef Infra 和 Chef InSpec。透過 AWS OpsWorks for Chef Automate，您可以使用社群撰寫或自訂的 Chef 技術指南，而無需進行 AWS OpsWorks特定變更。

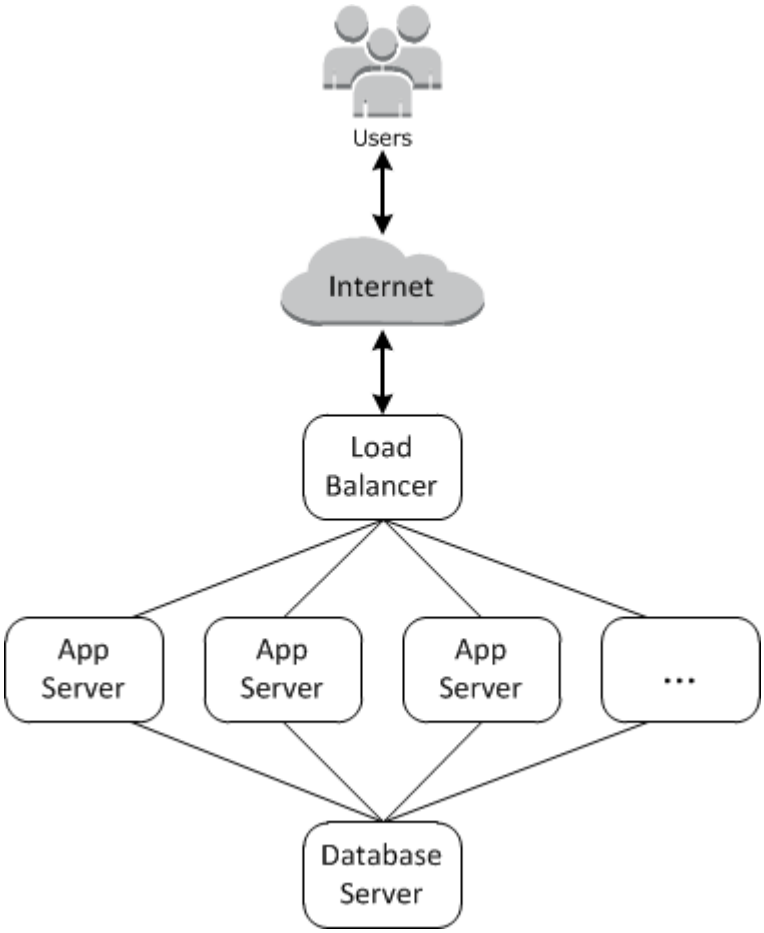
由於 會在單一執行個體上 AWS OpsWorks for Chef Automate 管理 Chef Automate 元件，因此您的伺服器可以在您選擇的時間自動備份、一律執行最新的 Chef 次要版本，以及一律套用最新的安全更新。您可以使用 Amazon EC2 Auto Scaling 群組自動將新的 Amazon EC2 節點與您的伺服器建立關聯。

AWS OpsWorks 堆疊

雲端型運算通常涉及一組 AWS 資源，例如 EC2 執行個體和 Amazon Relational Database Service (RDS) 執行個體。例如，Web 應用程式通常需要應用程式伺服器、資料庫伺服器、負載平衡器，以及其他資源。此執行個體群組通常稱為堆疊。

AWS OpsWorks Stacks 是原始服務，提供簡單且彈性的方式來建立和管理堆疊和應用程式。 AWS OpsWorks Stacks 可讓您部署和監控堆疊中的應用程式。您可以建立堆疊，以協助您管理稱為 layer 之專門群組中的雲端資源。一個 layer 代表一組用於特定用途的 EC2 執行個體 (例如供應用程式運作或託管資料庫伺服器)。Layer 會根據 Chef recipes 處理像是在執行個體上安裝套件、部署應用程式、執行指令碼等任務。

與 不同 AWS OpsWorks for Chef Automate， AWS OpsWorks Stacks 不需要或建立 Chef 伺服器； AWS OpsWorks Stacks 會為您執行 Chef 伺服器的一些工作。 AWS OpsWorks Stacks 會監控執行個體運作狀態，並在必要時使用 Auto Healing 和 Auto Scaling 為您佈建新的執行個體。簡易的應用程式伺服器堆疊看起來如下表所示。


Users
Internet
Load Balancer
App Server
App Server
App Server
...
Database Server

AWS OpsWorks for Puppet Enterprise

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

OpsWorks for Puppet Enterprise 可讓您在幾分鐘內啟動 Puppet Enterprise 主控端，並讓 AWS OpsWorks 處理其操作、備份、還原和軟體升級。OpsWorks for Puppet Enterprise 可讓您專注於核心組態管理任務，而不是管理 Puppet 主伺服器。透過使用 OpsWorks for Puppet Enterprise，您可以使用相同的組態來管理內部部署和雲端基礎設施，協助您在混合環境中有效率地擴展操作。Puppet Enterprise 主控台、 AWS Management Console以及 AWS CLI可簡化 Puppet 主伺服器的管理作業。

Puppet 主伺服器可以將特定節點的組態目錄提供給 `puppet-agent` 軟體，並做為 Puppet 模組的集中儲存庫，以管理您環境中的節點組態。OpsWorks for Puppet Enterprise 中的 Puppet 主伺服器會部署`puppet-agent`到您的受管節點，並提供 Puppet Enterprise 的進階功能。

OpsWorks for Puppet Enterprise 主執行個體會在 Amazon Elastic Compute Cloud 執行個體上執行。OpsWorks for Puppet Enterprise 伺服器設定為執行最新版本的 Amazon Linux (Amazon Linux 2)，以及最新版本的 Puppet Enterprise Master，2019.8.5 版。如需 Puppet Enterprise 2019.8.5 中變更的詳細資訊，請參閱 Puppet Enterprise 版本備註。

當有新版本的 Puppet 軟體可用時，系統維護的設計會在其通過 AWS 測試之後，自動更新伺服器上的 Puppet Enterprise 版本。AWS 會執行廣泛的測試，驗證 Puppet 升級已生產就緒，而不會中斷現有的客戶環境。

您可以連接任何執行受支援作業系統且具有 OpsWorks for Puppet Enterprise 主控端網路存取權的現場部署電腦或 EC2 執行個體。Puppet 主伺服器會在您要管理的節點上安裝 `puppet` 代理程式軟體。

主題

- 區域支援 Puppet Enterprise 的 OpsWorks
- AWS OpsWorks for Puppet Enterprise 生命週期結束FAQs
- OpsWorks for Puppet Enterprise 入門
- 使用 建立 AWS OpsWorks for Puppet Enterprise 主伺服器 AWS CloudFormation

- 更新 Puppet Enterprise Server 的 OpsWorks 以使用自訂網域
- 在 AWS OpsWorks for Puppet Enterprise 資源上使用標籤
- 備份和還原 Puppet Enterprise Server 的 OpsWorks
- OpsWorks for Puppet Enterprise 中的系統維護
- 在適用於 Puppet Enterprise 的 OpsWorks 中自動新增節點
- 取消節點與 Puppet Enterprise Server OpsWorks 的關聯
- 刪除 Puppet Enterprise Server 的 OpsWorks
- 如何將 OpsWorks for Puppet Enterprise 伺服器遷移至 Amazon Elastic Compute Cloud (Amazon EC2)
- 使用 記錄 Puppet Enterprise API 呼叫的 OpsWorks AWS CloudTrail
- 故障診斷 Puppet Enterprise 的 OpsWorks

區域支援 Puppet Enterprise 的 OpsWorks

下列區域端點支援 OpsWorks for Puppet Enterprise 主控端。OpsWorks for Puppet Enterprise 會在與 Puppet 主伺服器相同的區域端點中建立與您的 Puppet 主伺服器相關聯的資源，例如執行個體設定檔、使用者和服務角色。您的 Puppet 主伺服器必須在 VPC 中。您可以使用您建立的 VPC、既有的 VPC 或預設的 VPC。

- 美國東部 (俄亥俄) 區域
- 美國東部 (維吉尼亞北部) 區域
- 美國西部 (加利佛尼亞北部) 區域
- 美國西部 (奧勒岡) 區域
- 亞太 (東京) 區域
- 亞太區域 (新加坡) 區域
- 亞太區域 (雪梨) 區域
- 歐洲 (法蘭克福) 區域
- 歐洲 (愛爾蘭) 區域

AWS OpsWorks for Puppet Enterprise 生命週期結束FAQs

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

主題

- 現有客戶將受到此生命週期結束的影響？
- 如果我不採取任何動作，我的伺服器會發生什麼情況？
- 是否 AWS OpsWorks for Puppet Enterprise 接受新客戶？
- 生命週期結束會 AWS 區域 同時影響所有 嗎？
- 提供哪些層級的技術支援 AWS OpsWorks for Puppet Enterprise？
- 我是 OpsWorks for Puppet Enterprise 的目前客戶，我需要在先前未使用此服務的帳戶中啟動伺服器。我是否可以這麼做？
- 是否有任何新的功能版本 AWS OpsWorks for Puppet Enterprise？

現有客戶將受到此生命週期結束的影響？

現有客戶在 2024 年 3 月 31 日之前，OpsWorks for Puppet Enterprise 的生命週期結束日期不會受到影響。在生命週期終止日期之後，客戶將無法再使用 OpsWorks 主控台或 API 管理其伺服器。

如果我不採取任何動作，我的伺服器會發生什麼情況？

從 2024 年 3 月 31 日起，您將無法再使用 OpsWorks 主控台或 API 管理您的伺服器。屆時，我們將停止為您的伺服器執行任何持續管理功能，例如備份或維護。為了限制對客戶的影響，我們會讓執行 的 EC2 執行個體備份 Puppet Enterprise 伺服器，但他們的授權將不再有效，因為 OpsWorks for Puppet Enterprise 服務協議不再涵蓋 （或計費） 用量。如果您想要繼續使用 Puppet Enterprise 管理基礎設施，請參閱如何將 OpsWorks for Puppet Enterprise 伺服器遷移至 Amazon Elastic Compute Cloud (Amazon EC2)。

是否 AWS OpsWorks for Puppet Enterprise 接受新客戶？

否。 AWS OpsWorks for Puppet Enterprise 不再接受新客戶。

生命週期結束會 AWS 區域 同時影響所有 嗎？

是。API 和主控台將達到生命週期結束，截至 2024 年 3 月 31 日，所有 區域都將無法使用。如需 AWS OpsWorks for Puppet Enterprise 可用 AWS 區域 的清單，請參閱AWS 區域服務清單。

提供哪些層級的技術支援 AWS OpsWorks for Puppet Enterprise？

AWS 將持續為 AWS OpsWorks for Puppet Enterprise 該客戶提供相同層級的支援，直到生命週期結束日為止。如果您有問題或疑慮，可以透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

我是 OpsWorks for Puppet Enterprise 的目前客戶，我需要在先前未使用此服務的帳戶中啟動伺服器。我是否可以這麼做？

一般而言不會，除非有特殊情況需要這樣做。如果您有特殊情況，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊，並提供詳細資訊和理由，我們將審核您的請求。

是否有任何新的功能版本 AWS OpsWorks for Puppet Enterprise？

否。由於服務即將生命週期結束，我們將不會發行任何新功能。不過，我們將繼續如預期進行安全性改善和管理伺服器，直到生命週期結束為止。

OpsWorks for Puppet Enterprise 入門

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

OpsWorks for Puppet Enterprise 可讓您在其中執行 Puppet Enterprise 伺服器 AWS。只要約 15 分鐘就能佈建 Puppet Enterprise 主伺服器。

從 2021 年 5 月 3 日開始，OpsWorks for Puppet Enterprise 會在 中存放一些 Puppet Enterprise 伺服器屬性 AWS Secrets Manager。如需詳細資訊，請參閱與 整合 AWS Secrets Manager。

下列逐步解說可協助您在 OpsWorks for Puppet Enterprise 中建立第一個 Puppet 主伺服器。

先決條件

開始之前，您必須完成下列先決條件。

主題

- 安裝 Puppet 開發套件
- 安裝 Puppet Enterprise 用戶端工具
- 設定 Git 控制儲存庫
- 設定 VPC
- 設定 EC2 金鑰對 (選用)
- 使用自訂網域的先決條件 (選用)

安裝 Puppet 開發套件

1. 從 Puppet 網站，下載符合您本機電腦作業系統的 Puppet 開發套件。
2. 安裝 Puppet 開發套件。
3. 將 Puppet 開發套件新增至您本機電腦的 PATH 變數。

   - 在 Linux 或 macOS 作業系統上，您可以透過在 Bash shell 中執行下列命令，將 Puppet 開發套件新增至您的 PATH 變數。

     ```
     echo 'export PATH=/opt/puppetlabs/pdk/bin/pdk:$PATH' >> ~/.bash_profile && source
      ~/.bash_profile
     ```

   - 在 Windows 作業系統上，您可以在 PowerShell 工作階段PATH中使用下列 .NET Framework 命令，或在可從系統屬性存取的環境變數對話方塊中，將 Puppet 開發套件新增至您的變數。您可能需要以管理員身分執行 PowerShell 工作階段，才能執行下列命令。

     ```
     [Environment]::SetEnvironmentVariable("Path","new path value","Machine")
     ```

安裝 Puppet Enterprise 用戶端工具

Puppet Enterprise (PE) 用戶端工具是一組命令列工具，可讓您從工作站存取 Puppet Enterprise 服務。這些工具可以安裝在許多不同的作業系統上，也可以安裝在您要使用 Puppet 管理的節點上。如需工具支援之作業系統及其安裝方式的資訊，請參閱 Puppet Enterprise 文件中的 Installing PE client tools。

設定 Git 控制儲存庫

在您啟動 Puppet 主伺服器之前，您必須在 Git 中設定控制儲存庫，以存放及變更管理您的 Puppet 模組和類別。在啟動 Puppet Enterprise 主伺服器的步驟中，需要 Git 儲存庫的 URL 及 HTTPS 或用來存取儲存庫的 SSH 帳戶資訊。如需如何設定您的 Puppet Enterprise 主伺服器將使用之控制儲存庫的詳細資訊，請參閱 Setting up a control repository。您也可以在 `control-repo`GitHub 上 Puppet 範例儲存庫的讀我檔案中，找到控制儲存庫設定說明。控制儲存庫的結構類似如下。

```
### LICENSE
### Puppetfile
### README.md
### environment.conf
### hieradata
#   ### common.yaml
#   ### nodes
#       ### example-node.yaml
### manifests
#   ### site.pp
### scripts
#   ### code_manager_config_version.rb
#   ### config_version.rb
#   ### config_version.sh
### site
    ### profile
    #   ### manifests
    #       ### base.pp
    #       ### example.pp
    ### role
        ### manifests
            ### database_server.pp
            ### example.pp
            ### webserver.pp
```

使用 CodeCommit 設定儲存庫

您可以使用 CodeCommit 建立新的儲存庫。如需如何使用 CodeCommit 建立控制儲存庫的詳細資訊，請參閱本指南the section called “選用：使用 CodeCommit”中的 。如需如何在 CodeCommit 上開始使用 Git 的詳細資訊，請參閱 AWS CodeCommit 入門。若要為您的儲存庫授權 OpsWorks for Puppet Enterprise 伺服器，請將`AWSCodeCommitReadOnly`政策連接至您的 IAM 執行個體描述檔角色。

設定 VPC

您的 OpsWorks for Puppet Enterprise 主伺服器必須在 Amazon Virtual Private Cloud 中操作。您可以將其新增至現有的 VPC、使用預設 VPC，或建立新的 VPC 來包含伺服器。如需有關 Amazon VPC 以及如何建立新的 VPC 的資訊，請參閱 Amazon VPC 入門指南。

如果您建立自己的 VPC 或使用現有的 VPC，VPC 應有下列設定或屬性。

- VPC 至少應有一個子網路。

  如果您的 OpsWorks for Puppet Enterprise 主伺服器將可公開存取，請將子網路設為公有，並啟用自動指派公有 IP。
- 應啟用 DNS resolution (DNS 解析)。
- 在子網路上，啟用 Auto-assign public IP (自動指派公有 IP)。

設定 EC2 金鑰對 (選用)

SSH 連線對於 Puppet 伺服器的典型管理並非必要或建議；您可以使用 AWS Management Console 和 AWS CLI 命令在 Puppet 伺服器上執行許多管理任務。

如果您遺失或想要變更 Puppet Enterprise Web 型主控台的登入密碼，則需要 EC2 金鑰對，才能使用 SSH 連線至您的伺服器。您可以使用現有的金鑰對，或建立新的金鑰對。如需如何建立新的 EC2 金鑰對的詳細資訊，請參閱 Amazon EC2 金鑰對。

如果您不需要 EC2 金鑰對，則可以建立 Puppet Enterprise 主伺服器。

使用自訂網域的先決條件 (選用)

您可以在自有網域上設定 Puppet Enterprise 主伺服器，指定自訂網域中的公有端點做為伺服器的端點。如本節所述，當您使用自訂網域時，下列所有項目都是必要的。

主題

- 設定自訂網域
- 取得憑證
- 取得私密金鑰

設定自訂網域

若要在自有的自訂網域上執行 Puppet Enterprise 主伺服器，您需要一部伺服器的公有端點，例如 `https://aws.my-company.com`。如先前章節所述，如果您指定自訂網域，您也必須提供憑證和私密金鑰。

若要在建立伺服器之後存取伺服器，請在慣用的 DNS 服務中新增 CNAME DNS 記錄。此記錄必須將自訂網域指向由 Puppet 主伺服器建立程序所產生的端點 (伺服器 `Endpoint` 屬性的值)。如果伺服器使用自訂網域，您將無法使用產生的 `Endpoint` 值來存取伺服器。

取得憑證

若要在自有的自訂網域上設定 Puppet 主伺服器，您需要一個 PEM 格式的 HTTPS 憑證。這可以是單一、自我簽署的憑證或憑證鏈。當您完成 Create a Puppet Enterprise Master (建立 Puppet Enterprise 主伺服器) 工作流程時，如果您指定此憑證，則還必須提供自訂網域和私密金鑰。

以下是憑證值的需求：

- 您可以提供自我簽署、自訂憑證或完整的憑證鏈。
- 憑證必須是有效的 X509 憑證，或是 PEM 格式的憑證鏈。
- 憑證在上傳時必須有效。您不能在憑證有效期間開始 (憑證的 `NotBefore` 日期) 之前或憑證有效期到期 (憑證的 `NotAfter` 日期) 之後使用憑證。
- 憑證的一般名稱或主體別名 (SAN) (如果存在) 必須符合自訂網域值。
- 憑證必須符合 Custom private key (自訂私密金鑰) 欄位的值。

取得私密金鑰

若要在自有的自訂網域上設定 Puppet 主伺服器，您需要使用 PEM 格式的私密金鑰以利用 HTTPS 連接到伺服器。私密金鑰不得加密，不能受密碼或密碼短語保護。如果您指定自訂私密金鑰，您還必須提供自訂網域和憑證。

建立 Puppet Enterprise 主伺服器

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

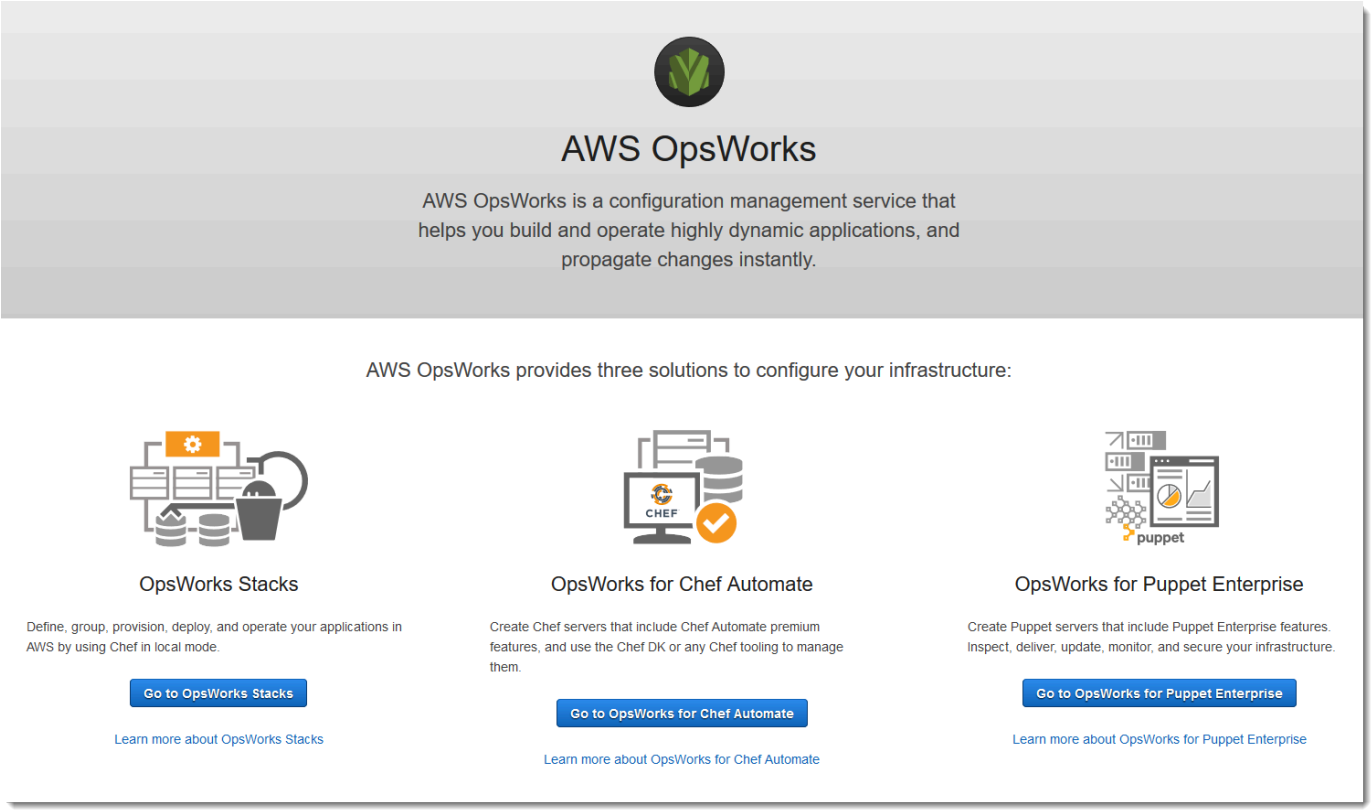
您可以使用 OpsWorks for Puppet Enterprise 主控台或 來建立 Puppet 主伺服器 AWS CLI。

主題

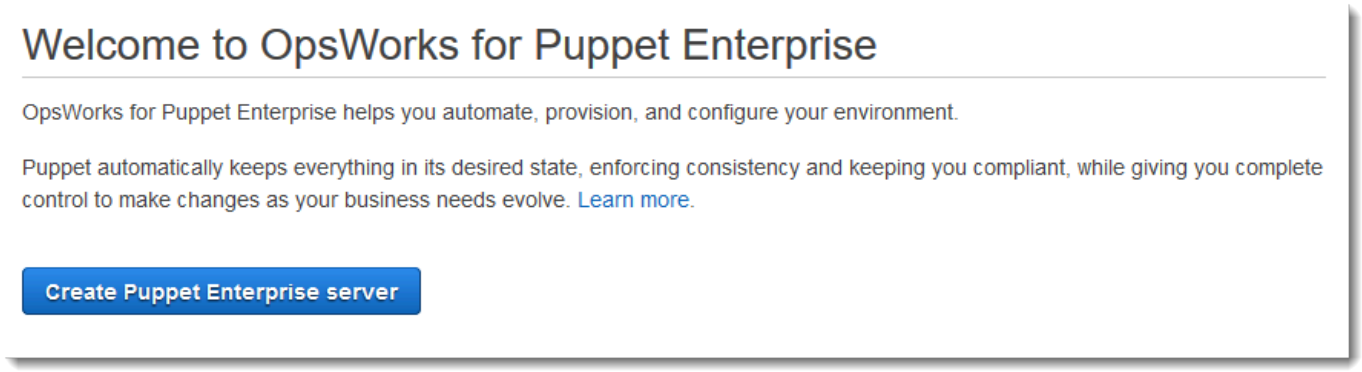
- 使用 建立 Puppet Enterprise Master AWS Management Console
- 使用 建立 Puppet Enterprise Master AWS CLI

使用 建立 Puppet Enterprise Master AWS Management Console

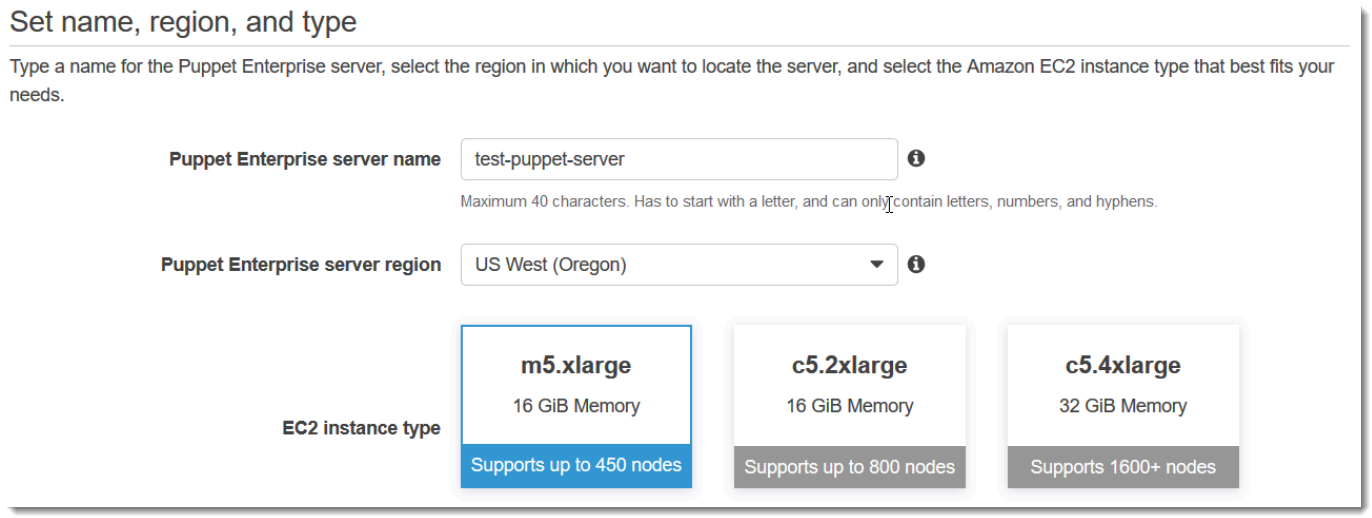
1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/opsworks/ 開啟 AWS OpsWorks 主控台。
2. 在 AWS OpsWorks 首頁上，選擇 Go to OpsWorks for Puppet Enterprise。



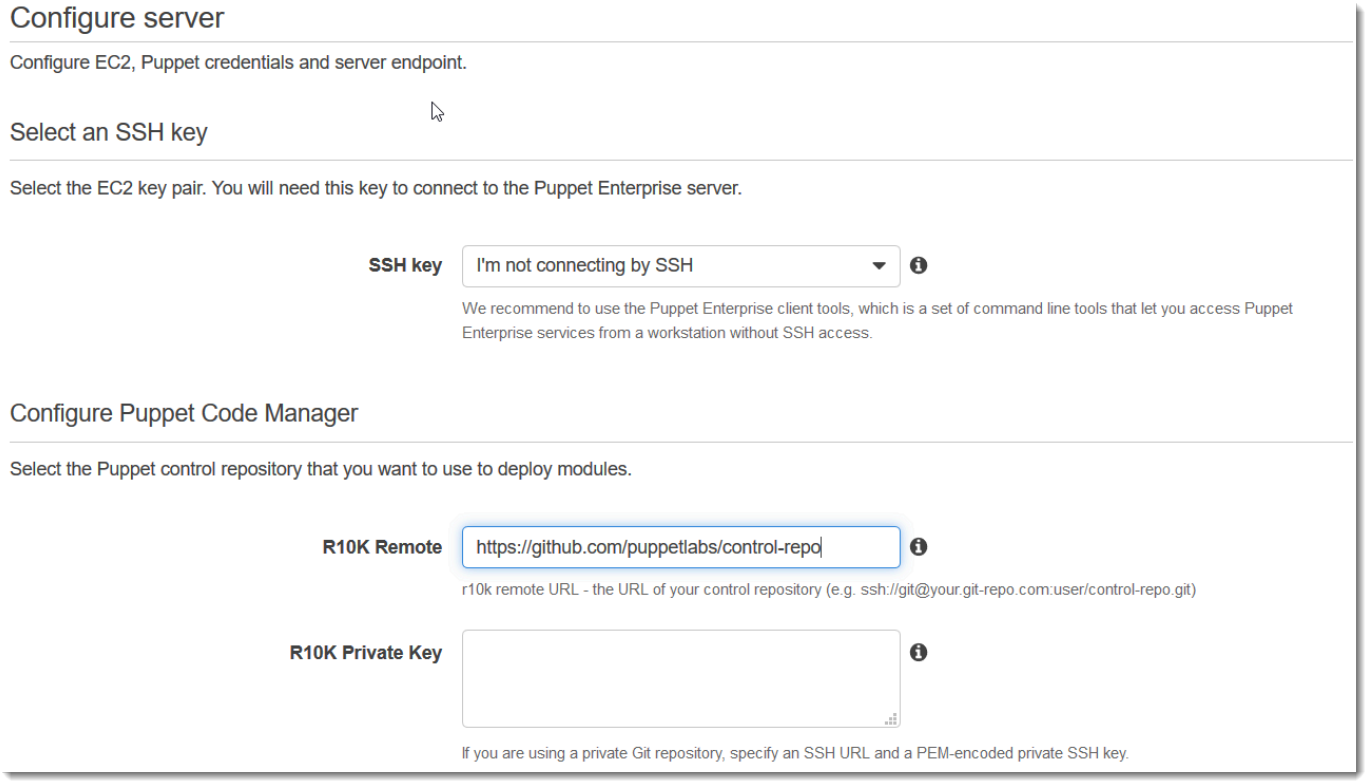
3. 在 OpsWorks for Puppet Enterprise 首頁上，選擇建立 Puppet Enterprise 伺服器。



4. 在 Set name, region, and type (設定名稱、區域和類型) 頁面上，指定您的伺服器名稱。Puppet 主伺服器名稱最多可包含 40 個字元、必須以字母開頭，而且只能包含英數字元和破折號。選取支援的區域，然後選擇支援您想要管理之節點數目的執行個體類型。如有需要，您可以在建立伺服器之後變更執行個體類型。在此演練中，我們會在美國西部 （奧勒岡） 區域中建立 m5.xlarge 執行個體類型。選擇下一步。



5. 在 Configure server (設定伺服器) 頁面上，保留 SSH key (SSH 金鑰) 下拉式清單中的預設選項，除非您想要指定金鑰對名稱。在設定 Puppet Code Manager 區域的 r10k 遠端欄位中，指定 Git 遠端的有效 SSH 或 HTTPS URL。在 r10k 私有金鑰欄位中，貼上 AWS OpsWorks 可用於存取 r10k 遠端儲存庫的 SSH 私有金鑰。當您建立私有儲存庫時這是由 Git 所提供，但如果您使用 HTTPS 身分驗證來存取控制儲存庫則非必要。選擇下一步。

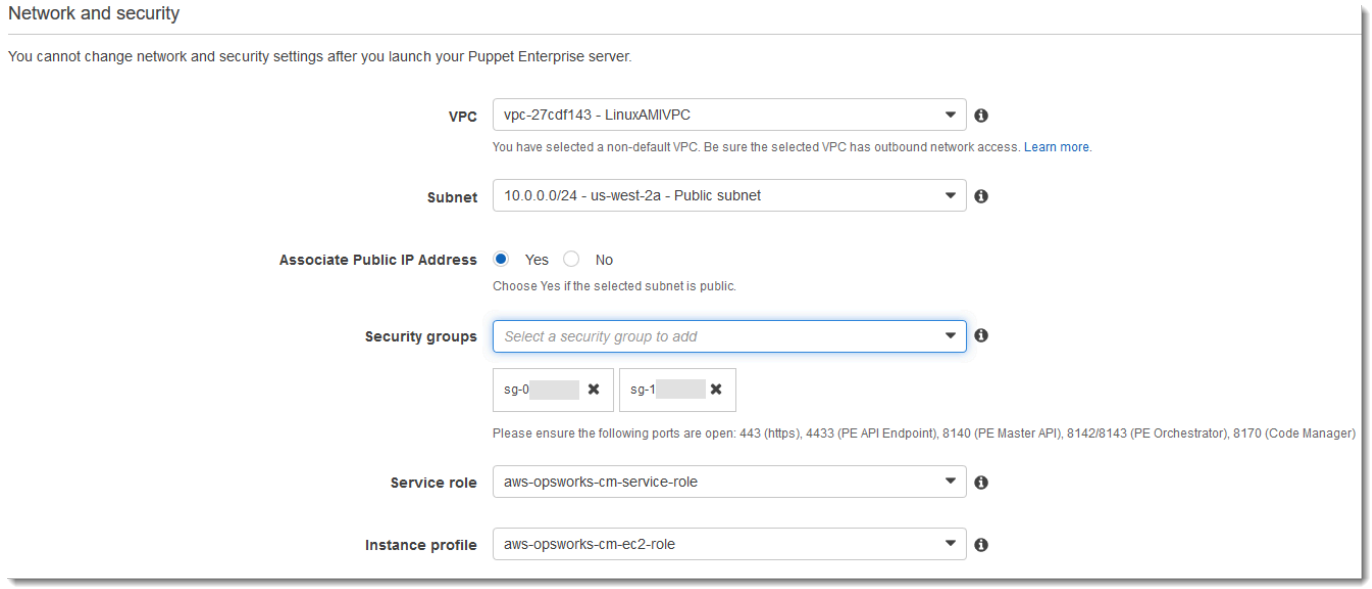


6. 保留 Specify server endpoint (指定伺服器端點) 中的預設值 Use an automatically-generated endpoint (使用自動產生的端點)，然後選擇 Next (下一步)，除非您希望伺服器位於您自己的自訂網域中。若要設定自訂網域，請繼續下一個步驟。
7. 若要使用自訂網域，請在 Specify server endpoint (指定伺服器端點) 下拉式清單中選擇 Use a custom domain (使用自訂網域)。

   a. 在 Fully qualified domain name (FQDN) (完整網域名稱 (FQDN)) 中指定 FQDN。您必須擁有想要使用的網域名稱。

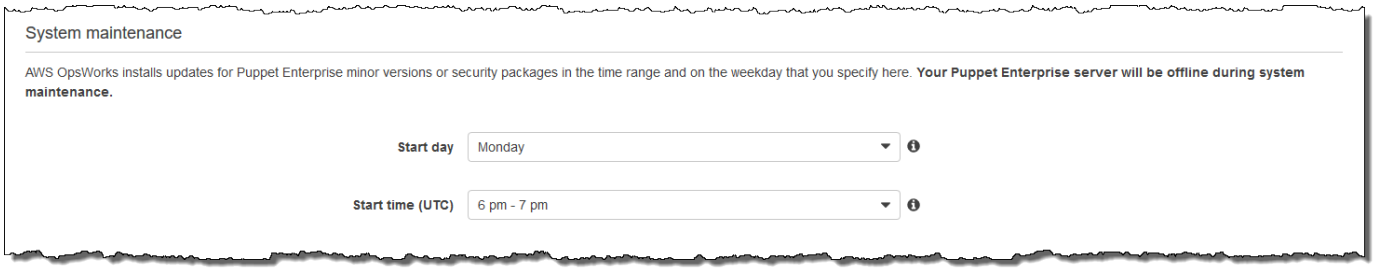
   b. 在 SSL certificate (SSL 憑證) 中貼上整個 PEM 格式憑證，開頭為 `-----BEGIN CERTIFICATE-----` 且結尾為 `-----END CERTIFICATE-----`。SSL 憑證主體必須符合您在上個步驟輸入的 FQDN。在憑證之前和之後移除任何額外的資料行。

   c. 在 SSL private key (SSL 私密金鑰) 中貼上整個 RSA 私密金鑰，開頭為 `-----BEGIN RSA PRIVATE KEY-----` 且結尾為 `-----END RSA PRIVATE KEY-----`。SSL 私密金鑰必須符合您在上個步驟的 SSL 憑證中輸入的公開金鑰。在私密金鑰之前和之後刪除任何額外的資料行。選擇下一步。
8. 在設定進階設定頁面的網路和安全區域中，選擇 VPC、子網路和一或多個安全群組。如果您還沒有要使用的安全群組、服務角色和執行個體描述檔， AWS OpsWorks 可以為您產生。您的伺服器可以是多個安全群組的成員。離開此頁面之後，您將無法變更 Puppet 主伺服器的網路與安全設定。

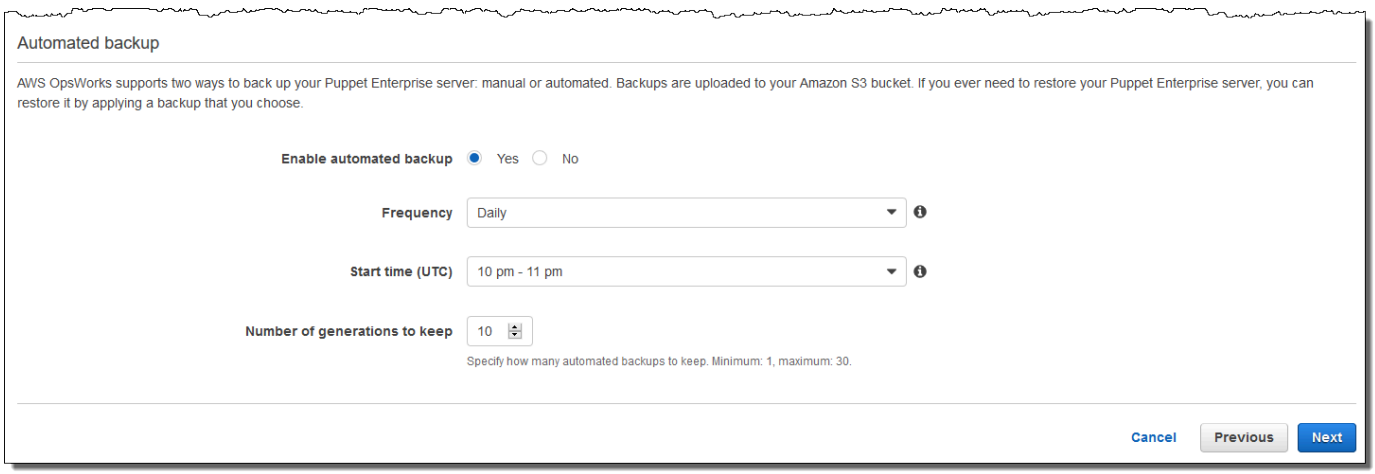


9. 在 System maintenance (系統維護) 區段中，將日和小時設為您希望開始系統維護的時間。因為您應預期伺服器在系統維護期間離線，請選擇一般工作時間內伺服器需求較低的時間。

   需要維護時段。您可以使用 AWS Management Console AWS CLI或 APIs，在稍後變更開始日期和時間。



10. 設定備份。預設會啟用自動備份。設定自動備份開始的偏好頻率和小時數，並設定要存放在 Amazon Simple Storage Service 中的備份產生次數。最多可保留 30 個備份；達到上限時，OpsWorks for Puppet Enterprise 會刪除最舊的備份，為新的備份騰出空間。



11. (選用) 在 Tags (標籤) 中，將標籤新增至伺服器和相關資源，例如 EC2 執行個體、彈性 IP 地址、安全群組、S3 儲存貯體和備份。如需標記 OpsWorks for Puppet Enterprise 伺服器的詳細資訊，請參閱 在 AWS OpsWorks for Puppet Enterprise 資源上使用標籤。
12. 當您完成設定進階設定時，選擇 Next (下一步)。
13. 在 Review (檢閱) 頁面上，檢視您的選擇。當您準備好建立伺服器時，選擇 Launch (啟動)。

    當您等待 AWS OpsWorks 建立 Puppet 主伺服器時，請前往 使用入門套件設定 Puppet 主伺服器並下載入門套件和 Puppet Enterprise 主控台登入資料。不要等到您的伺服器上線，再下載這些項目。

    當伺服器建立完成時，您可以在 OpsWorks for Puppet Enterprise 首頁上使用 Puppet 主伺服器，狀態為線上。伺服器上線之後，伺服器的網域會提供 Puppet Enterprise 主控台，URL 格式如下：`https://`*`your_server_name`*`-`*`randomID`*`.`*`region`*`.opsworks-cm.io`。

使用 建立 Puppet Enterprise Master AWS CLI

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

透過執行 AWS CLI 命令建立 OpsWorks for Puppet Enterprise 主伺服器與在主控台中建立伺服器不同。在 主控台中，如果您未指定要使用的現有服務角色和安全群組， 會為您 AWS OpsWorks 建立服務角色和安全群組。在 中 AWS CLI， AWS OpsWorks 如果您未指定安全群組，但它不會自動建立

服務角色，則必須在`create-server`命令中提供服務角色 ARN。在 主控台中，當 AWS OpsWorks 正在建立 Puppet 主伺服器時，您可以下載 Puppet Enterprise 主控台的入門套件和登入憑證。由於您在使用 建立 OpsWorks for Puppet Enterprise 主控端時無法執行此操作 AWS CLI，因此在新的 OpsWorks for Puppet Enterprise 主控端上線之後，您可以使用 JSON 處理公用程式從`create-server`命令結果取得登入憑證和入門套件。

如果您的本機電腦尚未執行 AWS CLI， AWS CLI 請遵循 AWS Command Line Interface 使用者指南中的安裝說明下載並安裝 。本節不會說明您可以搭配 `create-server` 命令使用的所有參數。如需 `create-server` 參數的詳細資訊，請參閱「參考」`create-server`中的 AWS CLI 。

1. 請務必完成先決條件。若要建立您的 Puppet 主伺服器，您需要子網路 ID，因此您必須具備 VPC。
2. 建立服務角色和執行個體描述檔。
3. 尋找並複製您帳戶中服務角色ARNs。

   ```
   aws iam list-roles --path-prefix "/service-role/" --no-paginate
   ```

   在 `list-roles` 命令的結果中，尋找類似如下的服務角色 ARN 項目。請記下服務角色 ARN。您需要這些值，才能建立您的 Puppet Enterprise 主伺服器。

   ```
   {
       "AssumeRolePolicyDocument": {
           "Version": "2012-10-17",
           "Statement": [
               {
                   "Action": "sts:AssumeRole",
                   "Effect": "Allow",
                   "Principal": {
                       "Service": "ec2.amazonaws.com"
                   }
               }
           ]
       },
       "RoleId": "AROZZZZZZZZZZQG6R22HC",
       "CreateDate": "2018-01-05T20:42:20Z",
       "RoleName": "aws-opsworks-cm-ec2-role",
       "Path": "/service-role/",
       "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-ec2-role"
   },
   {
   ```

```
    "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "opsworks-cm.amazonaws.com"
                }
            }
        ]
    },
    "RoleId": "AROZZZZZZZZZZZZZZZ6QE",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-service-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-service-
role"
}
```

4. 在您的帳戶中，找出並複製執行個體描述檔的 ARN。

```
aws iam list-instance-profiles --no-paginate
```

在 `list-instance-profiles` 命令的結果中，尋找類似如下的執行個體描述檔 ARN 項目。請記下執行個體描述檔 ARN。您需要這些值，才能建立您的 Puppet Enterprise 主伺服器。

```
{
    "Path": "/",
    "InstanceProfileName": "aws-opsworks-cm-ec2-role",
    "InstanceProfileId": "EXAMPLEDC6UR3LTUW7VHK",
    "Arn": "arn:aws:iam::123456789012:instance-profile/aws-opsworks-cm-ec2-role",
    "CreateDate": "2017-01-05T20:42:20Z",
    "Roles": [
        {
            "Path": "/service-role/",
            "RoleName": "aws-opsworks-cm-ec2-role",
            "RoleId": "EXAMPLEE4STNUQG6R22HC",
            "Arn": "arn:aws:iam::123456789012:role/service-role/aws-opsworks-cm-
ec2-role",
            "CreateDate": "2017-01-05T20:42:20Z",
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
```

```
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            }
        }
    ]
},
```

5. 執行 `create-server`命令來建立 OpsWorks for Puppet Enterprise 主伺服器。

   - `--engine` 值為 `Puppet`、 `--engine-model` 為 `Monolithic`，而且`--engine-version`可以是 `2019`或 `2017`。
   - 伺服器名稱在每個區域中 AWS 的帳戶內必須是唯一的。伺服器名稱開頭必須是字母，後面允許字母、數字或連字號 (-)，最多可包含 40 個字元。
   - 使用您在步驟 3 和 4 中複製的執行個體描述檔 ARN 和服務角色 ARN。
   - 有效的執行個體類型為 `m5.xlarge`、`c5.2xlarge` 或 `c5.4xlarge`。如需這些執行個體類型規格的詳細資訊，請參閱《Amazon EC2 使用者指南》中的執行個體類型。
   - `--engine-attributes` 參數為選用；如果您未指定 Puppet 管理員密碼，伺服器建立程序會為您產生密碼。如果您新增 `--engine-attributes`，請指定 `PUPPET_ADMIN_PASSWORD`，這是用於登入 Puppet Enterprise 主控台網頁的管理員密碼。此密碼必須使用 8 到 32 個 ASCII 字元。
   - SSH 金鑰對為選用，但如果您需要重設主控台管理員密碼，它可協助您連線至 Puppet 主伺服器。如需建立 SSH 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。 Amazon EC2
   - 若要使用自訂網域，請將下列參數新增至您的命令。否則，Puppet 主伺服器建立程序會自動為您產生端點。需要所有三個參數才能設定自訂網域。如需使用這些參數的其他需求的相關資訊，請參閱《CM API AWS OpsWorks 參考》中的 CreateServer。
     - `--custom-domain` - 伺服器的選用公有端點，例如 `https://aws.my-company.com`。
     - `--custom-certificate` - PEM 格式的 HTTPS 憑證。此值可以是單一、自我簽署的憑證或憑證鏈。

  - `--custom-private-key` - PEM 格式的私密金鑰，以便利用 HTTPS 連線至伺服器。私密金鑰不得加密，不能受密碼或密碼短語保護。
- 需要每週系統維護。有效值必須以下列格式指定：`DDD:HH:MM`。指定的時間是以國際標準時間 (UTC) 表示。如果您未指定 `--preferred-maintenance-window` 的值，預設值為星期二、星期三或星期五的隨機一小時時段。
- `--preferred-backup-window` 的有效值必須以下列其中一種格式指定：`HH:MM` 表示每日備份，或 `DDD:HH:MM` 表示每週備份。指定的時間是以 UTC 表示。預設值為隨機的每日開始時間。若要退出自動備份，請改為新增參數 `--disable-automated-backup`。
- 針對 `--security-group-ids`，輸入一或多個安全群組 ID，並以空格分隔。
- 針對 `--subnet-ids`，輸入子網路 ID。

```
aws opsworks-cm create-server --engine "Puppet" --engine-model "Monolithic"
 --engine-version "2019" --server-name "server_name" --instance-profile-arn
 "instance_profile_ARN" --instance-type "instance_type" --engine-attributes
 '{"PUPPET_ADMIN_PASSWORD":"ASCII_password"}' --key-pair "key_pair_name" --
preferred-maintenance-window "ddd:hh:mm" --preferred-backup-window "ddd:hh:mm"
 --security-group-ids security_group_id1 security_group_id2 --service-role-arn
 "service_role_ARN" --subnet-ids subnet_ID
```

以下是範例。

```
aws opsworks-cm create-server --engine "Puppet" --engine-model
 "Monolithic" --engine-version "2019" --server-name "puppet-02" --
instance-profile-arn "arn:aws:iam::111122223333:instance-profile/aws-
opsworks-cm-ec2-role" --instance-type "m5.xlarge" --engine-attributes
 '{"PUPPET_ADMIN_PASSWORD":"zZZzDj2DLYXSZFRv1d"}' --key-pair "amazon-test"
 --preferred-maintenance-window "Mon:08:00" --preferred-backup-window
 "Sun:02:00" --security-group-ids sg-b00000001 sg-b0000008 --service-role-arn
 "arn:aws:iam::111122223333:role/service-role/aws-opsworks-cm-service-role" --
subnet-ids subnet-383daa71
```

下列範例會建立使用自訂網域的 Puppet 主伺服器。

```
aws opsworks-cm create-server \
    --engine "Puppet" \
    --engine-model "Monolithic" \
    --engine-version "2019" \
    --server-name "puppet-02" \
```

```
    --instance-profile-arn "arn:aws:iam::111122223333:instance-profile/aws-
opsworks-cm-ec2-role" \
    --instance-type "m5.xlarge" \
    --engine-attributes '{"PUPPET_ADMIN_PASSWORD":"zZZzDj2DLYXSZFRv1d"}' \
    --custom-domain "my-puppet-master.my-corp.com" \
    --custom-certificate "-----BEGIN CERTIFICATE----- EXAMPLEqEXAMPLE== -----END
 CERTIFICATE-----" \
    --custom-private-key "-----BEGIN RSA PRIVATE KEY----- EXAMPLEqEXAMPLE= -----END
 RSA PRIVATE KEY-----" \
    --key-pair "amazon-test"
    --preferred-maintenance-window "Mon:08:00" \
    --preferred-backup-window "Sun:02:00" \
    --security-group-ids sg-b00000001 sg-b0000008 \
    --service-role-arn "arn:aws:iam::111122223333:role/service-role/aws-opsworks-
cm-service-role" \
    --subnet-ids subnet-383daa71
```

下列範例會建立可新增兩個標籤的 Puppet 主伺服器：`Stage: Production` 和 `Department: Marketing`。如需在 OpsWorks for Puppet Enterprise 伺服器上新增和管理標籤的詳細資訊，請參閱本指南[在 AWS OpsWorks for Puppet Enterprise 資源上使用標籤]中的 。

```
aws opsworks-cm create-server \
    --engine "Puppet" \
    --engine-model "Monolithic" \
    --engine-version "2019" \
    --server-name "puppet-02" \
    --instance-profile-arn "arn:aws:iam::111122223333:instance-profile/aws-
opsworks-cm-ec2-role" \
    --instance-type "m5.xlarge" \
    --engine-attributes '{"PUPPET_ADMIN_PASSWORD":"zZZzDj2DLYXSZFRv1d"}' \
    --key-pair "amazon-test"
    --preferred-maintenance-window "Mon:08:00" \
    --preferred-backup-window "Sun:02:00" \
    --security-group-ids sg-b00000001 sg-b0000008 \
    --service-role-arn "arn:aws:iam::111122223333:role/service-role/aws-opsworks-
cm-service-role" \
    --subnet-ids subnet-383daa71 \
    --tags [{\"Key\":\"Stage\",\"Value\":\"Production\"},{\"Key\":\"Department\",
\"Value\":\"Marketing\"}]
```

6. OpsWorks for Puppet Enterprise 約需 15 分鐘才能建立新的伺服器。請勿關閉 `create-server` 命令的輸出或關閉您的 shell 工作階段，因為輸出可能包含不會再顯示的重要資訊。若要從 `create-server` 結果取得密碼和入門套件，請前往下一個步驟。

   如果您在伺服器上使用自訂網域，請在 `create-server` 命令的輸出中複製 `Endpoint` 屬性的值。以下是範例。

   ```
   "Endpoint": "puppet-07-exampleexample.opsworks-cm.us-east-1.amazonaws.com"
   ```

7. 如果您選擇讓 OpsWorks for Puppet Enterprise 為您產生密碼，您可以使用 jq 等 JSON 處理器，從`create-server`結果中擷取可用格式的密碼。安裝 jq 之後，您可以執行下列命令來解壓縮 Puppet 管理員密碼和入門套件。如果您未在步驟 3 中提供自己的密碼，請務必將解壓縮的管理員密碼儲存在方便但安全的位置。

   ```
   #Get the Puppet password:
   cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
    "PUPPET_ADMIN_PASSWORD") | .Value'

   #Get the Puppet Starter Kit:
   cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
    "PUPPET_STARTER_KIT") | .Value' | base64 -D > starterkit.zip
   ```

   > **Note**
   >
   > 您無法在 AWS Management Console中重新產生新的 Puppet 主伺服器入門套件。當您使用 建立 Puppet 主伺服器時 AWS CLI，請執行上述`jq`命令，將`create-server`結果中的 base64 編碼入門套件儲存為 ZIP 檔案。

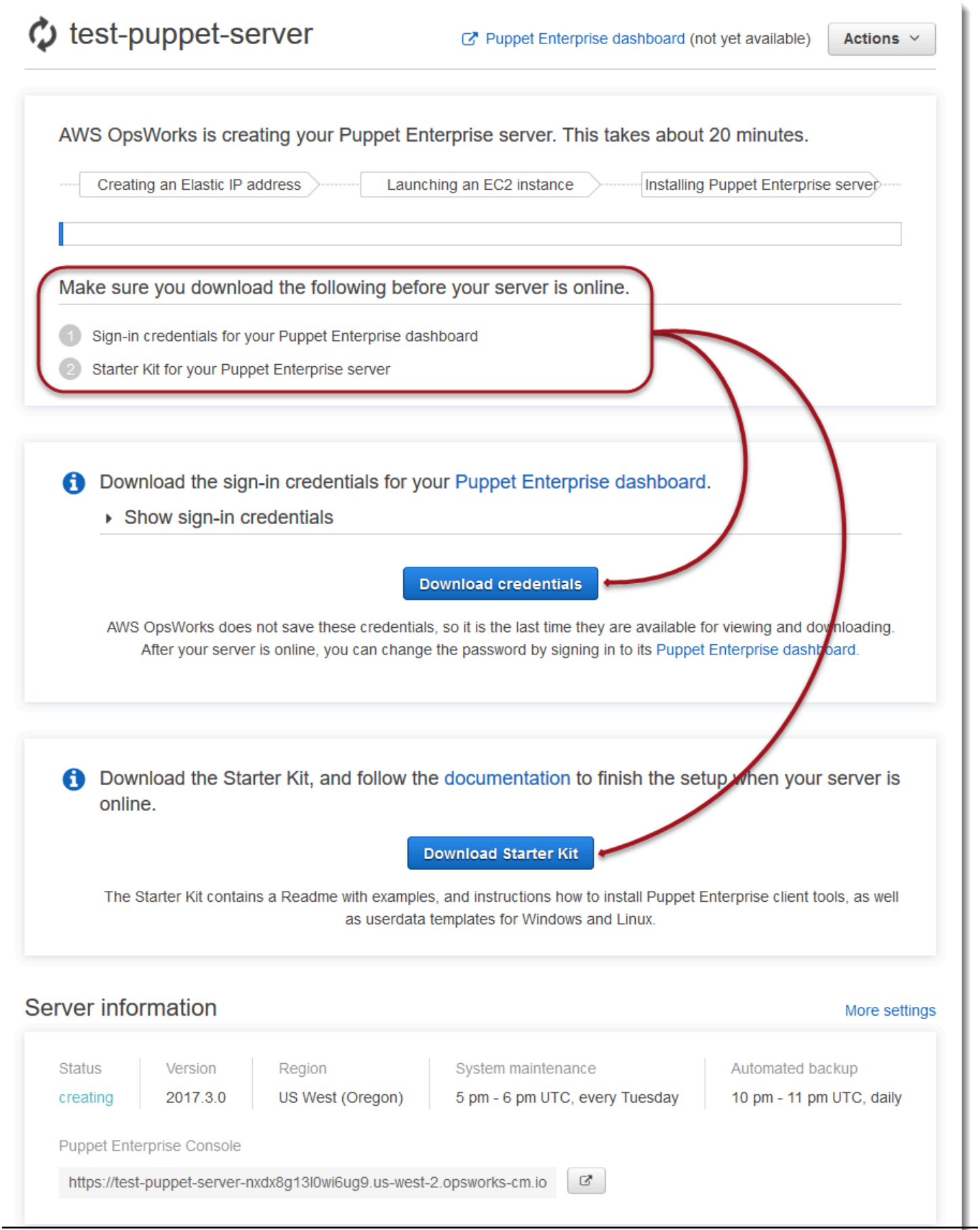
8. 如果您不是使用自訂網域，請繼續下一個步驟。如果您搭配 伺服器使用自訂網域，請在企業的 DNS 管理工具中建立 CNAME 項目，將自訂網域指向您在步驟 6 中複製的 OpsWorks for Puppet Enterprise 端點。在完成此步驟之前，您無法連線或登入具有自訂網域的伺服器。

9. 繼續進行下一節：the section called “尋找組態”。

使用入門套件設定 Puppet 主伺服器

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

當 Puppet 主伺服器建立仍在進行中時，伺服器的屬性頁面會在 OpsWorks for Puppet Enterprise 主控台中開啟。當您第一次使用新的 Puppet 主伺服器時，Properties (屬性) 頁面會提示您下載兩個必要項目。請下載這些項目後再連線至您的 Puppet 伺服器；在新的伺服器上線之後，將無法使用下載按鈕。


test-puppet-server
Puppet Enterprise dashboard (not yet available)
Actions
AWS OpsWorks is creating your Puppet Enterprise server. This takes about 20 minutes.
Creating an Elastic IP address
Launching an EC2 instance
Installing Puppet Enterprise server
Make sure you download the following before your server is online.
1 Sign-in credentials for your Puppet Enterprise dashboard
2 Starter Kit for your Puppet Enterprise server
Download the sign-in credentials for your Puppet Enterprise dashboard.
Show sign-in credentials
Download credentials
AWS OpsWorks does not save these credentials, so it is the last time they are available for viewing and downloading.
After your server is online, you can change the password by signing in to its Puppet Enterprise dashboard.
Download the Starter Kit, and follow the documentation to finish the setup when your server is online.
Download Starter Kit
The Starter Kit contains a Readme with examples, and instructions how to install Puppet Enterprise client tools, as well as userdata templates for Windows and Linux.
Server information
More settings
Status
creating
Version
2017.3.0
Region
US West (Oregon)
System maintenance
5 pm - 6 pm UTC, every Tuesday
Automated backup
10 pm - 11 pm UTC, daily
Puppet Enterprise Console
https://test-puppet-server-nxdx8g13l0wi6ug9.us-west-2.opsworks-cm.io

- Sign-in credentials for the Puppet master. (Puppet 主伺服器的登入資料。) 您將使用這些登入資料來登入 Puppet Enterprise 主控台，您可以在其中執行大多數節點管理。 AWS OpsWorks 不會儲存這些登入資料；這是最後一次可供檢視和下載。如有需要，您可以在登入之後變更這些登入資料隨附的密碼。
- Starter Kit. (入門套件。) 入門套件包含內附說明如何完成設定之資訊和範例的讀我檔案，以及 Puppet Enterprise 主控台的管理員登入資料。每次下載入門套件，都會產生新的登入資料，且舊的登入資料會失效。

先決條件

1. 當伺服器建立仍在進行時，下載 Puppet 主伺服器的登入資料，並將其儲存在安全但方便的位置。
2. 下載入門套件，並將入門套件 .zip 檔案解壓縮至您的工作空間目錄。請不要共享您的登入資料。如果其他使用者將要管理 Puppet 主伺服器，請稍後將其新增為 Puppet Enterprise 主控台的管理員。如需如何將使用者新增至 Puppet 主伺服器的詳細資訊，請參閱 Puppet Enterprise 文件中的 Creating and managing users and user roles。

安裝 Puppet 主伺服器憑證

若要使用您的 Puppet 主伺服器並新增要管理的節點，您將需要安裝其憑證。執行下列 AWS CLI 命令來安裝它。您無法在 中執行此任務 AWS Management Console。

```
aws --region region opsworks-cm describe-servers --server-name server_name --query
 "Servers[0].EngineAttributes[?Name=='PUPPET_API_CA_CERT'].Value" --output text
 > .config/ssl/cert/ca.pem
```

產生短期字符

若要使用 Puppet API，您必須為自己建立短期字符。若是使用 Puppet Enterprise 主控台，則不需要此步驟。請執行下列命令來產生字符。

預設字符存留期為五分鐘，但您可以變更此預設值。

```
puppet-access login --config-file .config/puppetlabs/client-tools/puppet-access.conf --
lifetime 8h
```

> **Note**
>
> 由於預設字符存留期為五分鐘，上述範例命令會新增 `--lifetime` 參數將字符存留期延長為更長的期間。您可以將字符存留期設為高達 10 年 (`10y`) 的期間。如需如何變更預設字符存留期的詳細資訊，請參閱 Puppet Enterprise 文件中的 Change the token's default lifetime。

設定入門套件 Apache 範例

下載並解壓縮入門套件後，您可以使用隨附的範例`control-repo-example`資料夾中的範例分支，在受管節點上設定 Apache Web 伺服器。

此入門套件包含兩個 `control-repo` 資料夾：`control-repo` 和 `control-repo-example`。`control-repo` 資料夾包含 `production` 分支，與您在 Puppet GitHub 儲存庫中看到的分支並無不同。`control-repo-example` 資料夾也有`production`分支，其中包含使用測試網站設定 Apache 伺服器的範例程式碼。

1. 將 `control-repo-example` `production` 分支推送至您的 Git 遠端 (您 Puppet 主伺服器的 `r10k_remote` URL)。在您的入門套件根目錄中，執行下列程式碼，並以您的 *`URL ##`* r10kRemoteUrl`r10k_remote`。

   ```
   cd  control-repo-example
   git remote add origin r10kRemoteUrl
   git push origin production
   ```

   Puppet 的 Code Manager 使用 Git 分支做為環境。根據預設，所有節點都會在生產環境中。

   > **Important**
   >
   > 請勿推送至 `master` 分支。`master` 分支會預留給 Puppet 主伺服器。

2. 將 `control-repo-example` 分支中的程式碼部署至您的 Puppet 主伺服器。這可讓 Puppet 主伺服器從您的 Git 儲存庫下載 Puppet 程式碼 (`r10k_remote`)。在您的入門套件根目錄中，執行下列程式碼。

   ```
   puppet-code deploy --all --wait --config-file .config/puppet-code.conf
   ```

如需如何將範例 Apache 組態套用至您在 Amazon EC2 中建立的受管節點的詳細資訊，請參閱本指南步驟 2：使用自動關聯指令碼建立執行個體中的 。

為 Puppet 主伺服器新增要管理的節點

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

主題

- 執行 associateNode() API 呼叫
- 新增現場部署節點的考量
- 詳細資訊

新增節點的建議方法是使用 AWS OpsWorks `associateNode()` API。Puppet Enterprise 主伺服器託管一個儲存庫，可供您用來在要管理的節點上安裝 Puppet 代理程式軟體，而不論節點是在現場部署實體電腦或虛擬機器上。做為啟動程序的一部分，某些作業系統的 Puppet 代理程式軟體會安裝在 OpsWorks for Puppet Enterprise 伺服器上。下表顯示啟動時 OpsWorks for Puppet Enterprise 伺服器上可用的作業系統代理程式。

預先安裝的作業系統代理程式

支援的作業系統	版本
Ubuntu	16.04、18.04、20.04
Red Hat Enterprise Linux (RHEL)	6、7、8
Windows	所有 Puppet 支援之 Windows 版本的 64 位元版本

您可以針對其他作業系統將 `puppet-agent` 新增至您的伺服器。請注意，系統維護將會刪除您在啟動之後已新增至伺服器的代理程式。雖然已在執行遭刪除代理程式的大多數現有連接節點會繼續簽入，但執行 Debian 作業系統的節點可能會停止報告。建議您`puppet-agent`在執行作業系統的節點上手

動安裝 ，其中代理程式軟體未預先安裝在您的 OpsWorks for Puppet Enterprise 伺服器上。如需如何讓具有其他作業系統的節點可以在您伺服器上使用 puppet-agent 的詳細資訊，請參閱 Puppet Enterprise 文件中的 Installing agents。

如需如何透過填入 EC2 執行個體使用者資料，自動將節點與您的 Puppet 主伺服器建立關聯的資訊，請參閱在適用於 Puppet Enterprise 的 OpsWorks 中自動新增節點。

執行 `associateNode()` API 呼叫

透過安裝 新增節點之後puppet-agent，節點會將憑證簽署請求 (CSRs) 傳送至 OpsWorks for Puppet Enterprise 伺服器。您可以在 Puppet 主控台中檢視 CSR；如需節點 CSR 的詳細資訊，請參閱 Puppet Enterprise 文件中的 Managing certificate signing requests。執行 OpsWorks for Puppet Enterprise associateNode() API 呼叫會處理節點 CSRs，並將節點與您的伺服器建立關聯。以下是如何在 中使用此 API 呼叫 AWS CLI 來關聯單一節點的範例。您將需要節點所傳送之 PEM 格式的 CSR，這可透過 Puppet 主控台取得。

```
aws opsworks-cm associate-node --server-name "test-puppet-
server" --node-name "node or instance ID" --engine-attributes
 "Name=PUPPET_NODE_CSR,Value='PEM_formatted_CSR_from_the_node'
```

如需如何使用 associateNode() 自動新增節點的詳細資訊，請參閱在適用於 Puppet Enterprise 的 OpsWorks 中自動新增節點。

新增現場部署節點的考量

在現場部署電腦或虛擬機器puppet-agent上安裝 之後，您可以使用兩種方式之一，將現場部署節點與 OpsWorks for Puppet Enterprise 主節點建立關聯。

- 如果節點支援 AWS 開發套件、AWS CLI 或 AWS Tools for PowerShell 的安裝，您可以使用建議的方法來關聯節點，也就是執行 associateNode() API 呼叫。您第一次建立 OpsWorks for Puppet Enterprise 主節點時下載的入門套件，示範如何使用標籤將角色指派給節點。您可以透過在 CSR 中指定信任的事實，在建立節點與 Puppet 主伺服器關聯同時套用標籤。例如，入門套件隨附的示範控制儲存庫設定為使用標籤 pp_role 將角色指派給 Amazon EC2 執行個體。如需如何將標籤新增至 CSR 做為信任事實的詳細資訊，請參閱 Puppet 平台文件中的 Extension requests (permanent certificate data)。
- 如果節點無法執行 AWS 管理或開發工具，您仍然可以向 OpsWorks for Puppet Enterprise 主伺服器註冊節點，就像向任何未受管 Puppet Enterprise 主伺服器註冊節點一樣。如本主題所述，安裝 puppet-agent 會將 CSR 傳送至 OpsWorks for Puppet Enterprise 主伺服器。獲得授權的 Puppet

使用者可以手動簽署 CSR，或透過編輯存放在 Puppet 主伺服器上的 `autosign.conf` 檔案來設定 CSR 的自動簽署。如需設定自動簽署和編輯 `autosign.conf` 的詳細資訊，請參閱 Puppet 平台文件中的 SSL configuration: autosigning certificate requests。

若要將現場部署節點與一個 Puppet 主伺服器連結，允許 Puppet 主伺服器以接受所有 CSR，請在 Puppet Enterprise 主控台執行下列動作。控制此行為的參數為`puppet_enterprise::profile::master::allow_unauthenticated_ca`。

> **Important**
>
> 啟用 Puppet 主伺服器接受自簽 CSR，或不建議所有 CSRs 用於安全考量。根據預設，允許未經授權的 CSR 讓全世界都能使用 Puppet 主伺服器。設定上傳憑證要求須根據預設啟用，會讓您的 Puppet 主程式容易受到阻斷服務攻擊 (DoS)。

1. 登入 Puppet Enterprise 主控台
2. 選擇 Configure (設定)，選擇 Classification (分類)，選擇 PE Master，然後選擇 Configuration (組態) 標籤。
3. 在 Classification (分類) 標籤，找到 puppet_enterprise::profile::master 類別。
4. 將 allow_unauthenticated_ca 參數的值設為 true。
5. 儲存您的變更。您的變更會在下一次 Puppet run 期間套用。您可以允許變更在 30 分鐘內生效 (並新增現場部署節點)，或者您可以在 PE 主控台的 Run (執行) 區段手動初始化 Puppet run。

詳細資訊

請造訪 Learn Puppet 教學課程網站，進一步了解如何使用 OpsWorks for Puppet Enterprise 伺服器和 Puppet Enterprise 主控台功能。

登入 Puppet Enterprise 主控台

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

從 Puppet 主伺服器的 Properties (屬性) 頁面下載登入資料，且伺服器在線上之後，登入 Puppet Enterprise 主控台。在本演練中，我們已指示您指定包含您模組的控制儲存庫，並新增至少一個要管理的節點。這可讓您在主控台中看到有關代理程式和節點的資訊。

當您嘗試連線至 Puppet Enterprise 主控台網頁時，憑證警告會顯示在瀏覽器中，直到您在用來管理 Puppet 伺服器的用戶端電腦上安裝 AWS OpsWorks特定的 CA 簽署 SSL 憑證為止。如果您在繼續前往儀表板網頁之前不想看到警告，請安裝 SSL 憑證後再登入。

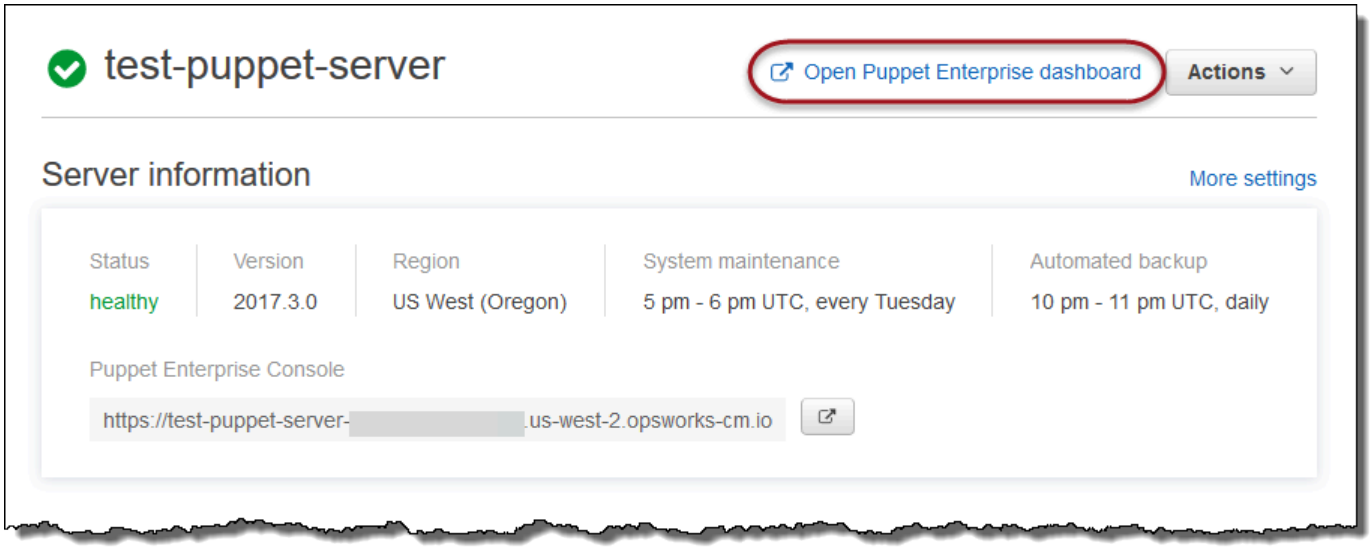
安裝 AWS OpsWorks SSL 憑證

- 選擇符合您系統的憑證。

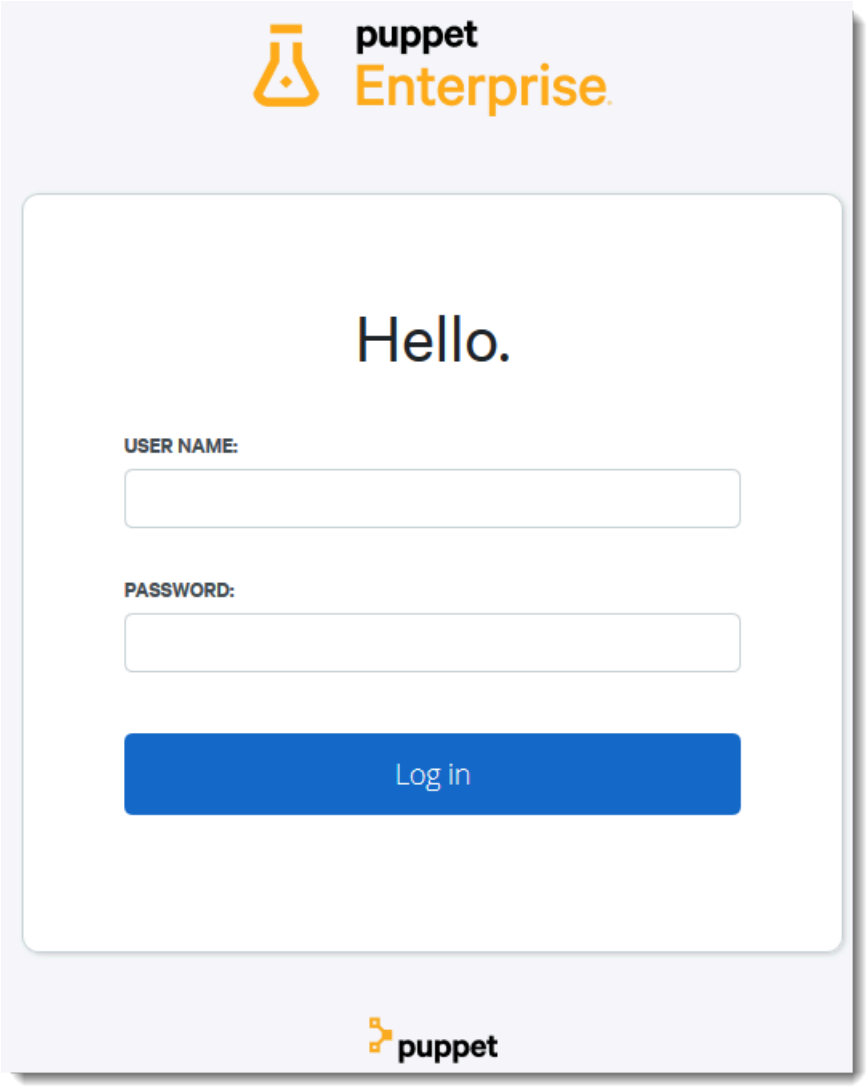
安裝用戶端 SSL 憑證之後，您可以登入 Puppet Enterprise 主控台，而不會看到警告訊息。

登入 Puppet Enterprise 主控台

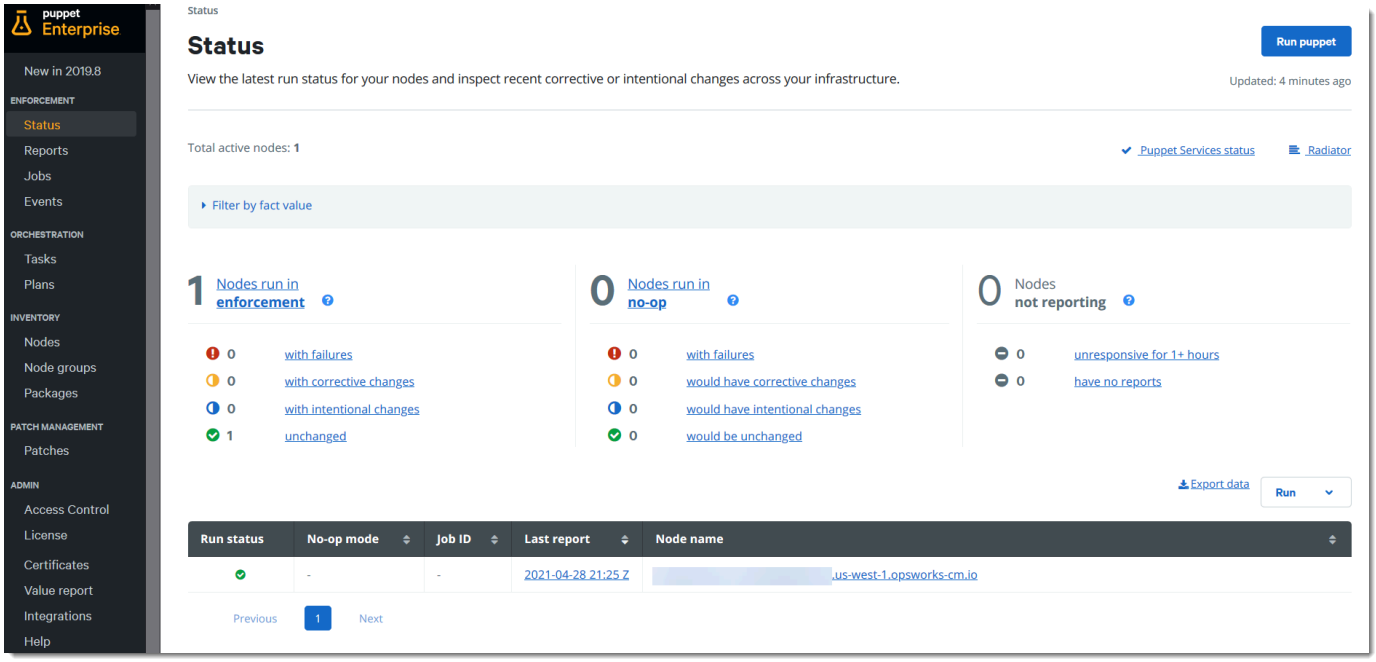
1. 解壓縮並開啟您在先決條件中下載的 Puppet Enterprise 登入資料。您需要這些登入資料才能登入。
2. 在 中 AWS Management Console，開啟 Puppet 伺服器的屬性頁面。
3. 在 Properties (屬性) 頁面右上方，選擇 Open Puppet Enterprise console (開啟 Puppet Enterprise 主控台)。



4. 使用步驟 1 中的登入資料登入。



5. 在 Puppet Enterprise 主控台中，您可以檢視有關您要管理的節點、模組執行進度和事件、節點的合規層級等詳細資訊。如需 Puppet Enterprise 主控台功能及其使用方式的詳細資訊，請參閱 Puppet Enterprise 文件中的管理節點。



群組和分類節點

在您透過對節點套用類別來指定所需的節點組態之前，請根據您企業中的節點角色或其常見特性來群組節點。群組和分類節點包含下列高階任務。您也可以使用 PE 主控台來完成這些任務。如需如何群組及分類節點的詳細資訊，請參閱 Puppet Enterprise 文件中的 Grouping and classifying nodes。

1. 建立節點群組。
2. 手動新增節點，或透過套用您建立的規則來自動新增。
3. 將類別指派給節點群組。

重設管理員和使用者密碼

如需有關如何變更您用來登入 Puppet Enterprise 主控台的密碼的資訊，請參閱 Puppet Enterprise 文件中的重設主控台管理員密碼。

根據預設，在嘗試登入十次之後，使用者會遭到鎖定而無法登入 Puppet 主控台。如需如何在鎖定時重設使用者密碼的詳細資訊，請參閱 Puppet Enterprise 文件中的 Password endpoints。

選用：使用 AWS CodeCommit 做為 Puppet r10k 遠端控制儲存庫

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

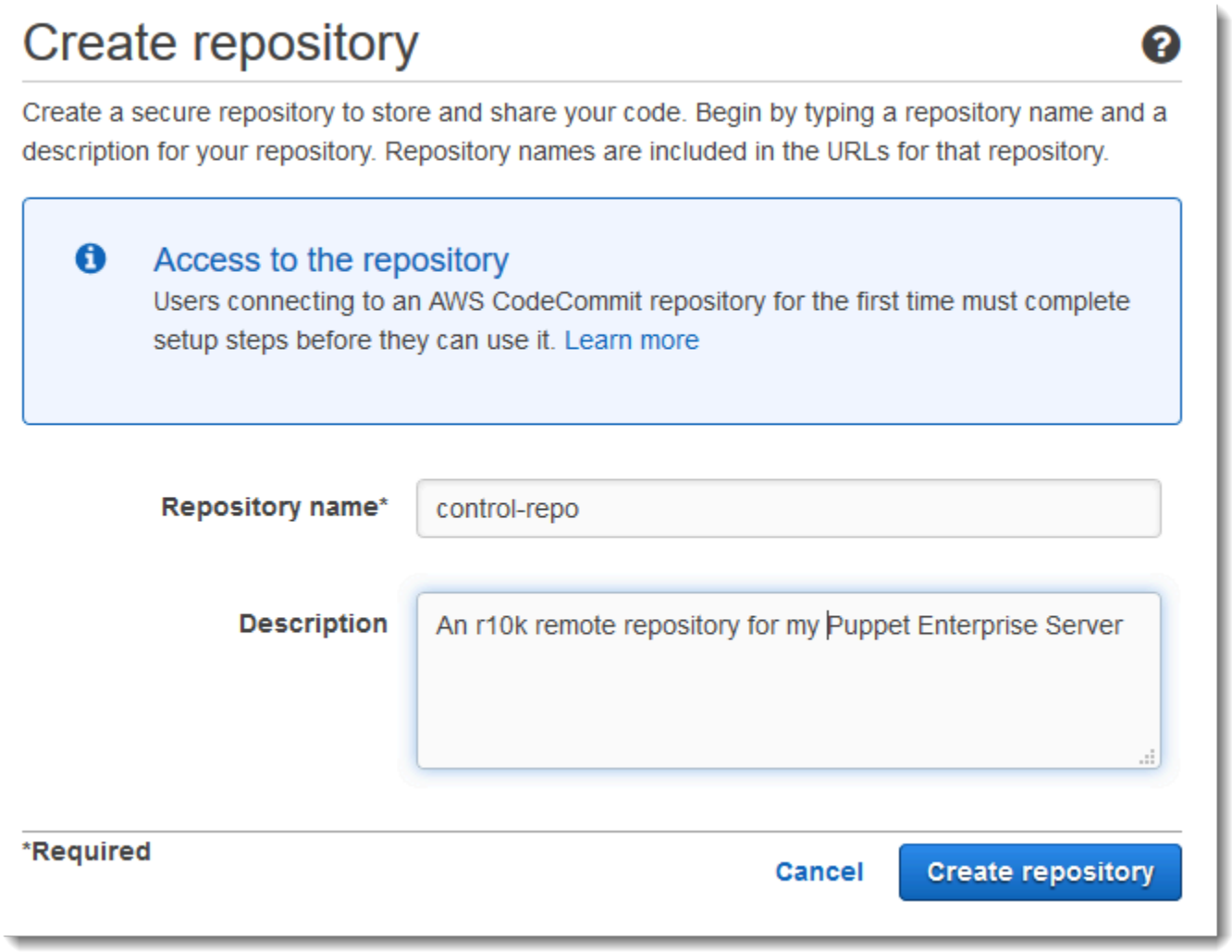
您可以使用 建立新的儲存庫 AWS CodeCommit，並將其用作 r10k 遠端控制儲存庫。若要完成本節中的步驟，並使用 CodeCommit 儲存庫，您需要具有 AWSCodeCommitReadOnly 受管政策所提供許可的使用者。

主題

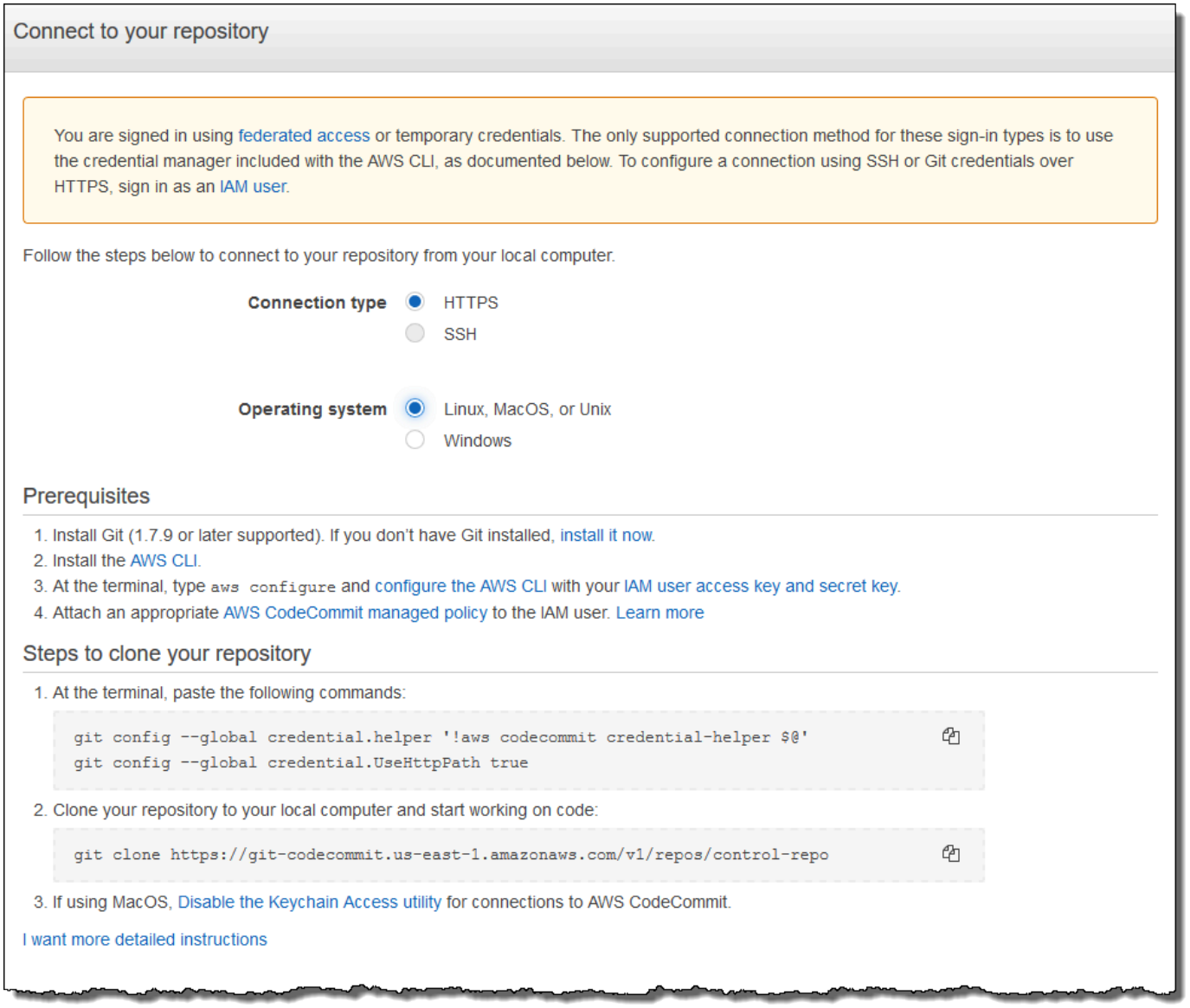
- 步驟 1：使用 CodeCommit 做為具有 HTTPS 連線類型的儲存庫
- 步驟 2：（選用）使用 CodeCommit 做為具有 SSH 連線類型的儲存庫

步驟 1：使用 CodeCommit 做為具有 HTTPS 連線類型的儲存庫

1. 在 CodeCommit 主控台中，建立新的儲存庫。



2. 選擇略過以略過設定 Amazon SNS 主題。
3. 在 Code (程式碼) 頁面上，選擇 Connect to your repository (連線至您的儲存庫)。
4. 在 Connect to your repository (連線至您的儲存庫) 頁面上，選擇 HTTPS 做為 Connection type (連線類型)，然後選擇您的作業系統。



在 Steps to clone your repository (複製您儲存庫的步驟) 區域中，您的 `git clone` URL 應類似如下：`https://git-codecommit.region.amazonaws.com/v1/repos/control-repo`。將此 URL 複製到方便用於 Puppet 伺服器設定的位置。

5. 關閉連線至您的儲存庫頁面，並返回 OpsWorks for Puppet Enterprise 伺服器設定。
6. 在 Puppet 主設定精靈的設定登入資料頁面的 r10k 遠端字串方塊中，貼上您在步驟 4 中複製的 URL。將 r10k private key (r10k 私有金鑰) 方塊保留空白。完成建立並啟動您的 Puppet 主伺服器。
7. 在 IAM 主控台中，將 AWSCodeCommitReadOnly 政策連接至 Puppet 主伺服器的執行個體描述檔角色。如需如何將許可政策新增至 IAM 角色的詳細資訊，請參閱《IAM 使用者指南》中的新增 IAM 身分許可 （主控台）。

8. 請遵循AWS CodeCommit 《 使用者指南》中的使用 Git 登入資料為 HTTPS 使用者設定 中的步驟，將現有`control-repo`內容推送至新的 CodeCommit 儲存庫。
9. 現在，您可以遵循the section called “尋找組態”中的說明繼續進行，並使用入門套件將程式碼部署至您的 Puppet 主伺服器。下列是範例命令。

```
puppet-code deploy --all --wait --config-file .config/puppet-code.conf
```

步驟 2：（選用）使用 CodeCommit 做為具有 SSH 連線類型的儲存庫

您可以設定 AWS CodeCommit r10k 遠端控制儲存庫來使用 SSH 金鑰對身分驗證。在您開始此程序之前，必須完成下列先決條件。

- 您必須已使用 HTTPS 控制儲存庫啟動 OpsWorks for Puppet Enterprise 伺服器，如上節所述：the section called “步驟 1：使用 CodeCommit 做為具有 HTTPS 連線類型的儲存庫”。必須先完成此操作，您才能將所需組態上傳到 Puppet 主程式。
- 請確定您有一個已連接 AWSCodeCommitReadOnly 受管政策的使用者。如需如何建立使用者的詳細資訊，請參閱《IAM 使用者指南》中的在 AWS 帳戶中建立 IAM 使用者。
- 建立 SSH 金鑰並與 使用者建立關聯。遵循《 AWS CodeCommit 使用者指南`ssh-keygen`》中的步驟 3：在 Linux、macOS 或 Unix 上設定登入資料中建立公有/私有金鑰對的指示。

1. 在 AWS CLI 工作階段中，執行下列命令，將私有金鑰檔案內容上傳至 AWS Systems Manager 參數存放區。您的 OpsWorks for Puppet Enterprise 伺服器會查詢此參數，以取得所需的憑證檔案。將 *`private_key_file`* 取代為 SSH 私有金鑰檔案的路徑。

```
aws ssm put-parameter --name puppet_user_pk --type String --value
 "`cat private_key_file`"
```

2. 將 Systems Manager 參數存放區許可新增至您的 Puppet 主伺服器。
   a. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
   b. 在左側導覽窗格中，選擇 Roles (角色)。
   c. 請選擇 aws-opsworks-cm-ec2-role。
   d. 在 Permissions (許可) 標籤上，選擇 Attach policies (連接政策)。
   e. 在 Search (搜尋) 列中，輸入 **`AmazonSSMManagedInstanceCore`**。
   f. 在搜尋結果中，選擇 AmazonSSMManagedInstanceCore。

g. 選擇連接政策。

3. 建立組態檔案資訊清單。如果您使用的是入門套件提供的 control-repo-example 儲存庫，請在範例儲存庫顯示的位置建立以下檔案。否則，請根據您自己的控制儲存庫結構來建立。將 *IAM_USER_SSH_KEY* 值取代為您在此程序的先決條件中建立的 SSH 金鑰 ID。

```
control-repo-example/site/profile/manifests/codecommit.pp
```

```
class profile::codecommit {
  $configfile = @(CONFIGFILE)
      Host git-codecommit.*.amazonaws.com
      User IAM_USER_SSH_KEY
      IdentityFile /etc/puppetlabs/puppetserver/ssh/codecommit.rsa
      StrictHostKeyChecking=no
      | CONFIGFILE

  # Replace REGION with the correct region for your server.
  $command = @(COMMAND)
      aws ssm get-parameters \
      --region REGION \
      --names puppet_user_pk \
      --query "Parameters[0].Value" \
      --output text >| /etc/puppetlabs/puppetserver/ssh/codecommit.rsa
      | COMMAND

  $dirs = [
            '/opt/puppetlabs/server/data/puppetserver/.ssh',
            '/etc/puppetlabs/puppetserver/ssh',
          ]

  file { $dirs:
    ensure => 'directory',
    group  => 'pe-puppet',
    owner  => 'pe-puppet',
    mode   => '0750',
  }

  file { 'ssh-config':
    path    => '/opt/puppetlabs/server/data/puppetserver/.ssh/config',
    require => File[$dirs],
    content => $configfile,
    group   => 'pe-puppet',
    owner   => 'pe-puppet',
```

```
    mode    => '0600',
  }

  exec { 'download-codecommit-certificate':
    command => $command,
    require => File[$dirs],
    creates => '/etc/puppetlabs/puppetserver/ssh/codecommit.rsa',
    path    => '/bin',
    cwd     => '/etc/puppetlabs',
  }

  file { 'private-key-permissions':
    subscribe => Exec['download-codecommit-certificate'],
    path      => '/etc/puppetlabs/puppetserver/ssh/codecommit.rsa',
    group     => 'pe-puppet',
    owner     => 'pe-puppet',
    mode      => '0600',
  }
}
```

4. 將您的控制儲存庫推送至 CodeCommit。執行下列命令將新的資訊清單檔案推送到您的儲存庫。

```
git add ./site/profile/manifests/codecommit.pp
git commit -m 'Configuring for SSH connection to CodeCommit'
git push origin production
```

5. 部署資訊清單檔案。執行下列命令，將更新的組態部署到您的 OpsWorks for Puppet Enterprise 伺服器。將 *STARTER_KIT_DIRECTORY* 取代為 Puppet 組態檔的路徑。

```
cd STARTER_KIT_DIRECTORY

puppet-access login --config-file .config/puppetlabs/client-tools/puppet-
access.conf

puppet-code deploy --all --wait \
--config-file .config/puppet-code.conf \
--token-file .config/puppetlabs/token
```

6. 更新 OpsWorks for Puppet Enterprise 伺服器的分類。在預設情況下，Puppet 代理程式每 30 分鐘會在節點上執行 (包括主節點)。若不想等待，您可以在 Puppet 主伺服器上手動執行代理程式。執行代理程式會挑選新的資訊清單檔案。

   a. 登入 Puppet Enterprise 主控台

b. 選擇 Classification (分類)。

c. 展開 PE Infrastructure (PE 基礎設施)。

d. 選擇 PE Master。

e. 在組態索引標籤上，在新增類別**`profile::codecommit`**中輸入 。

新類別 `profile::codecommit` 可能不會在執行 `puppet-code deploy` 之後立即出現。若未顯示，請選擇此頁面的 Refresh (重新整理)。

f. 選擇 Add class (新增類別)，然後選擇 Commit 1 change (遞交 1 變更)。

g. 在 OpsWorks for Puppet Enterprise 伺服器上手動執行 Puppet 代理程式。選擇 Nodes (節點)，在清單中選擇您的伺服器，選擇 Run Puppet (執行 Puppet)，然後選擇 Run (執行)。

7. 在 Puppet Enterprise 主控台中，變更儲存庫 URL 以使用 SSH，而非 HTTPS。您在這些步驟中執行的組態會在 OpsWorks for Puppet Enterprise 備份和還原程序期間儲存，因此您不需要在維護活動之後手動變更儲存庫組態。

a. 選擇 Classification (分類)。

b. 展開 PE Infrastructure (PE 基礎設施)。

c. 選擇 PE Master。

d. 在 Configuration (組態) 索引標籤上，找到 `puppet_enterprise::profile::master` 分類。

e. 選擇 `r10k_remote` 參數旁的編輯。

f. 將 HTTPS URL 取代為您儲存庫的 SSH URL，然後選擇 Commit 1 change (遞交 1 變更)。

g. 在 OpsWorks for Puppet Enterprise 伺服器上手動執行 Puppet 代理程式。選擇 Nodes (節點)，在清單中選擇您的伺服器，選擇 Run Puppet (執行 Puppet)，然後選擇 Run (執行)。

使用 建立 AWS OpsWorks for Puppet Enterprise 主伺服器 AWS CloudFormation

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks for Puppet Enterprise 可讓您在 中執行 Puppet Enterprise 伺服器 AWS。只要約 15 分鐘就能佈建 Puppet Enterprise 主伺服器。

從 2021 年 5 月 3 日開始，OpsWorks for Puppet Enterprise 會在 中存放一些 Puppet Enterprise 伺服器屬性 AWS Secrets Manager。如需詳細資訊，請參閱與 整合 AWS Secrets Manager。

下列逐步解說可協助您在 OpsWorks for Puppet Enterprise 中建立 Puppet 主伺服器，方法是在其中建立堆疊 AWS CloudFormation。

主題

- 先決條件
- 在 AWS CloudFormation中建立 Puppet Enterprise 主伺服器

先決條件

建立新的 Puppet 主伺服器之前，請在 OpsWorks for Puppet Enterprise 外部建立您需要的資源，以存取和管理 Puppet 主伺服器。如需詳細資訊，請參閱本指南中「入門」一節的先決條件。

如果您要建立使用自訂網域的伺服器，您需要自訂網域、憑證和私密金鑰。您必須為 AWS CloudFormation 範本中這三個參數指定值。如需 `CustomDomain`、 `CustomCertificate`和 `CustomPrivateKey` 參數需求的詳細資訊，請參閱《AWS OpsWorks CM API 參考》中的 CreateServer。

檢閱 AWS CloudFormation 使用者指南範本參考中的 OpsWorks-CM 區段，以了解您用來建立伺服器的 AWS CloudFormation 範本中支援和必要的值。

在 AWS CloudFormation中建立 Puppet Enterprise 主伺服器

本節說明如何使用 AWS CloudFormation 範本來建置堆疊，以建立 OpsWorks for Puppet Enterprise 主伺服器。您可以使用 AWS CloudFormation 主控台或 來執行此操作 AWS CLI。您可以使用範例 AWS CloudFormation 範本來建置 OpsWorks for Puppet Enterprise 伺服器堆疊。請務必使用您自己的伺服器名稱、IAM 角色、執行個體描述檔、伺服器描述、備份保留計數、維護選項和選用標籤來更新範例範本。如果您的伺服器將使用自訂網域，您必須在 AWS CloudFormation 範本中指定 `CustomDomain`、`CustomCertificate` 和 `CustomPrivateKey` 參數的值。如需有關這些選項的詳細資訊，請參閱本指南「入門」一節的the section called “使用 建立 Puppet Enterprise Master AWS Management Console”。

主題

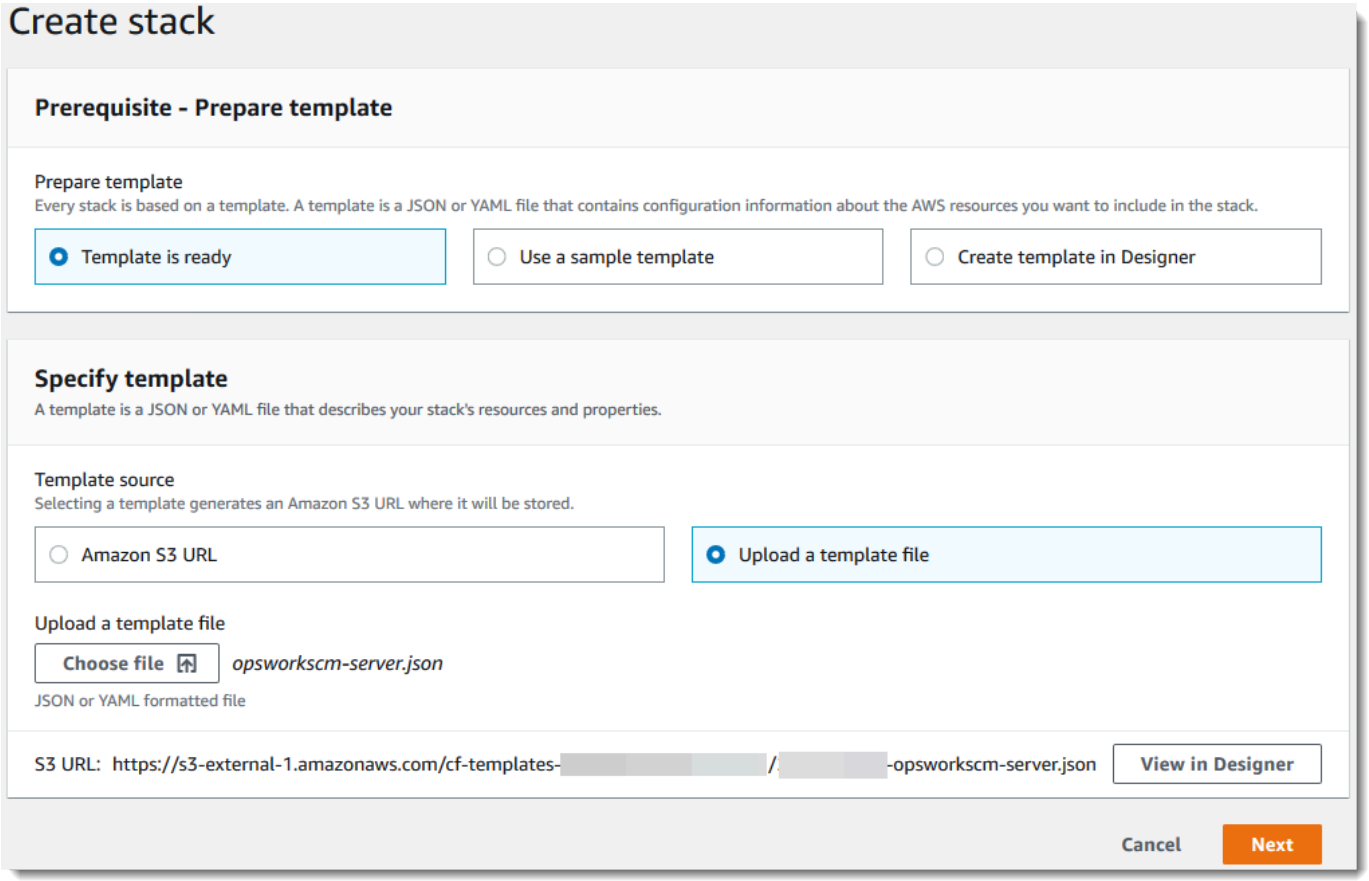
- 使用 AWS CloudFormation （ 主控台 ） 建立 Puppet Enterprise Master

- 使用 AWS CloudFormation (CLI) 建立 Puppet Enterprise Master

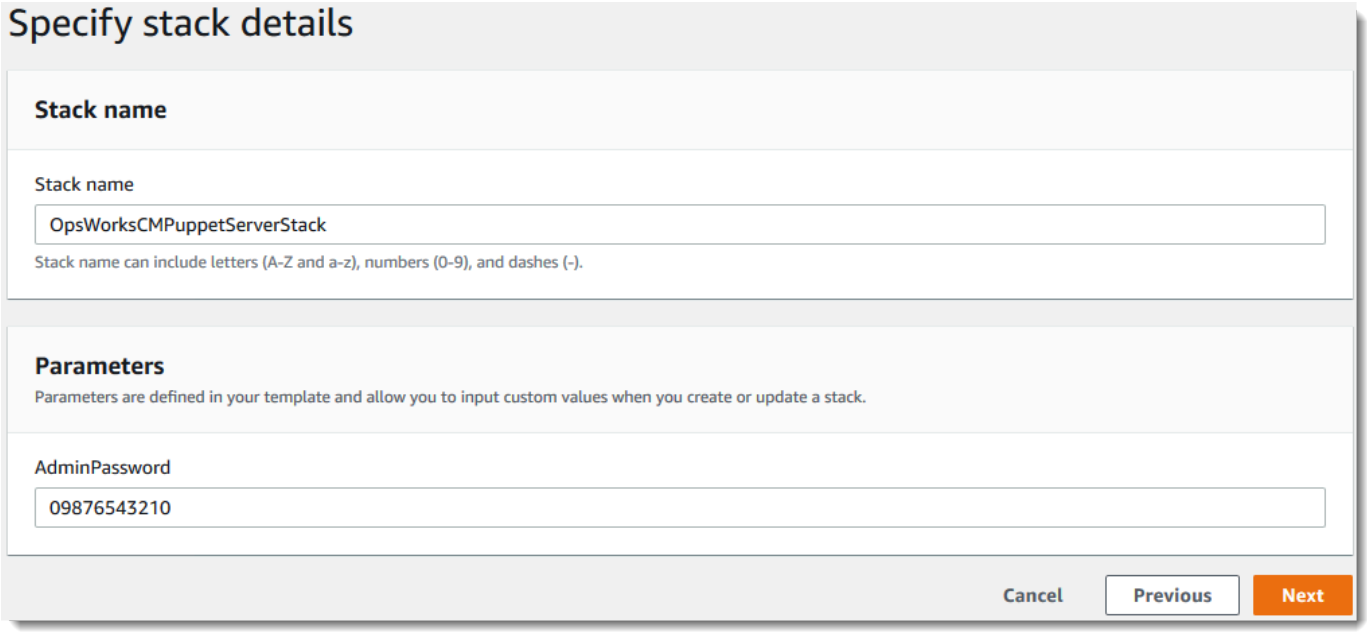
使用 AWS CloudFormation （主控台）建立 Puppet Enterprise Master

1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/cloudformation 開啟 AWS CloudFormation 主控台。
2. 在 AWS CloudFormation 首頁上，選擇建立堆疊。
3. 在 Prerequisite - Prepare template (先決條件 - 準備範本)中，如果您使用的是 範例 AWS CloudFormation 範本，請選擇 Template is ready (範本備妥)。
4. 在 Specify template (指定範本) 中，請選擇範本的來源。在此演練中，選擇上傳範本檔案，然後上傳建立 Puppet Enterprise 伺服器的 AWS CloudFormation 範本。瀏覽您的範本檔案，然後選擇 Next (下一步)。

   AWS CloudFormation 範本可以是 YAML 或 JSON 格式。範例 AWS CloudFormation 範本可供您使用；請務必將範例值取代為您自己的值。您可以使用 AWS CloudFormation 範本設計工具來建立新的範本或驗證現有的範本。如需如何執行此操作的詳細資訊，請參閱AWS CloudFormation 《使用者指南》中的AWS CloudFormation 設計工具界面概觀。



5. 在 Specify stack details (指定堆疊詳細資訊) 頁面上，輸入堆疊的名稱。這個名稱不會與您的伺服器名稱相同，這只是堆疊名稱。在 Parameters (參數) 區域中，輸入用於登入 Puppet Enterprise 網頁的管理員密碼。此密碼必須使用 8 到 32 個 ASCII 字元。選擇下一步。



6. 在選項頁面上，您可以將標籤新增至您使用堆疊建立的伺服器，如果您尚未指定要在範本中使用的 IAM 角色，請選擇用於建立資源的 IAM 角色。指定選項完成後，選擇 Next (下一步)。如需復原觸發等進階選項的詳細資訊，請參閱AWS CloudFormation 《 使用者指南》中的設定 AWS CloudFormation 堆疊選項。

7. 在 Review (檢閱) 頁面上，檢視您的選擇。當您準備好建立伺服器堆疊時，請選擇 Create (建立)。

   當您等待 AWS CloudFormation 建立堆疊時，請檢視堆疊建立狀態。如果堆疊建立失敗，則檢閱顯示在主控台中的錯誤訊息，以協助您解決問題。如需 AWS CloudFormation 針對堆疊中的錯誤進行故障診斷的詳細資訊，請參閱AWS CloudFormation 《 使用者指南》中的故障診斷錯誤。

   當伺服器建立完成時，您可以在 OpsWorks for Puppet Enterprise 首頁上使用 Puppet 主伺服器，狀態為線上。伺服器上線之後，伺服器的網域會提供 Puppet Enterprise 主控台，URL 格式如下：`https://`*`your_server_name`*`-`*`randomID`*`.`*`region`*`.opsworks-cm.io`。

> **Note**
>
> 如果您為伺服器指定自訂網域、憑證和私有金鑰，請在企業的 DNS 管理工具中建立 CNAME 項目，將您的自訂網域映射到 OpsWorks for Puppet Enterprise 為伺服器自動產生的端點。在將產生的端點對應至您的自訂網域值之前，您無法管理伺服器或連線至該伺服器的 Puppet Enterprise 管理網站。

若要取得產生的端點值，請在伺服器上線後執行下列 AWS CLI 命令：

```
aws opsworks describe-servers --server-name server_name
```

使用 AWS CloudFormation (CLI) 建立 Puppet Enterprise Master

如果您的本機電腦尚未執行 AWS CLI，AWS CLI 請依照《AWS Command Line Interface 使用者指南》中的安裝說明下載並安裝 。本節不會說明您可以搭配 create-stack 命令使用的所有參數。如需 create-stack 參數的詳細資訊，請參閱「參考」create-stack中的 AWS CLI 。

1. 請務必完成先決條件建立 OpsWorks for Puppet Enterprise 主控端的 。
2. 建立服務角色和執行個體描述檔。

   尋找並複製您帳戶中服務角色ARNs。

   ```
   aws iam list-roles --path-prefix "/service-role/" --no-paginate
   ```

   在 `list-roles` 命令的結果中，尋找類似如下的服務角色和執行個體描述檔項目。記下服務角色和執行個體描述檔的 ARNs，並將其新增至您用來建立 Puppet 主伺服器堆疊的 AWS CloudFormation 範本。

   ```
   {
       "AssumeRolePolicyDocument": {
           "Version": "2012-10-17",
           "Statement": [
               {
                   "Action": "sts:AssumeRole",
                   "Effect": "Allow",
                   "Principal": {
                       "Service": "ec2.amazonaws.com"
                   }
               }
           ]
       },
       "RoleId": "AROZZZZZZZZZZQG6R22HC",
       "CreateDate": "2018-01-05T20:42:20Z",
       "RoleName": "aws-opsworks-cm-ec2-role",
       "Path": "/service-role/",
       "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-ec2-role"
   ```

```
},
{
    "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "opsworks-cm.amazonaws.com"
                }
            }
        ]
    },
    "RoleId": "AROZZZZZZZZZZZZZZZ6QE",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-service-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-service-
role"
}
```

3. 再次執行 create-stack命令，以建立 OpsWorks for Puppet Enterprise 主伺服器。

   - 將 *stack_name* 更換成您的堆疊名稱。這是 AWS CloudFormation 堆疊的名稱，而不是您的 Puppet 主伺服器。Puppet 主伺服器名稱就是 AWS CloudFormation 範本內 `ServerName` 的值。
   - 將 *template* 更換成您的範本檔案路徑，並視情況，將副檔名*yaml # json* 改成 `.yaml` 或 `.json`。
   - `--parameters` 的值會對應到 EngineAttributes，其來源是 CreateServer API。處理 Puppet 時，以下是將用來建立伺服器的使用者提供引擎屬性。r10k 引擎屬性可將您的 Puppet 主伺服器連接到程式碼儲存庫，以供管理伺服器的環境資訊設定。如需 r10k 引擎屬性的詳細資訊，請參閱 Puppet Enterprise 文件中的搭配 r10k 管理程式碼。
     - `PUPPET_ADMIN_PASSWORD`，用於登入 Puppet Enterprise 主控台網頁的管理員密碼。密碼長度必須介於 8 和 32 個 ASCII 字元，且至少包含一個大寫字母、一個小寫字母、一個數字和一個特殊字元。
     - `PUPPET_R10K_REMOTE`，控制儲存庫的 URL (例如，ssh://git@your.git-repo.com:user/control-repo.git)。指定 r10k 遠端開啟 TCP 連接埠 8170。
     - `PUPPET_R10K_PRIVATE_KEY`。 如果您使用的是私有 Git 儲存庫，請新增 PUPPET_R10K_PRIVATE_KEY 來指定 SSH URL 和 PEM 編碼的私有 SSH 金鑰。

```
aws cloudformation create-stack --stack-name stack_name
 --template-body file://template.yaml or json --parameters
 ParameterKey=AdminPassword,ParameterValue="password"
```

以下是範例。

```
aws cloudformation create-stack --stack-name "OpsWorksCMPuppetServerStack"
 --template-body file://opsworkscm-puppet-server.json --parameters
 ParameterKey=AdminPassword,ParameterValue="09876543210Ab#"
```

下列範例會在 AWS CloudFormation 範本中未提供 r10k 引擎屬性時，將它們指定為參數。包含 r10k 引擎屬性的範例範本，puppet-server-param-attributes.yaml，已包含在範例 AWS CloudFormation 範本中。

```
aws cloudformation create-stack --stack-name MyPuppetStack --
template-body file://puppet-server-param-attributes.yaml --parameters
 ParameterKey=AdminPassword,ParameterValue="superSecret1%3"
 ParameterKey=R10KRemote,ParameterValue="https://www.yourRemote.com"
 ParameterKey=R10KKey,ParameterValue="$(cat puppet-r10k.pem)"
```

下面範例會在 AWS CloudFormation 範本中指定 r10k 引擎屬性及其值；此命令只需要指向範本檔案。經指定做為 --template-body、puppet-server-in-file-attributes.yaml 值的範本，則包含於範例 AWS CloudFormation 範本中。

```
aws cloudformation create-stack --stack-name MyPuppetStack --template-body file://
puppet-server-in-file-attributes.yaml
```

4. (選用) 若要取得堆疊建立狀態，請執行下列命令。

```
aws cloudformation describe-stacks --stack-name stack_name
```

5. 當堆疊建立完成後，請前往下一節the section called “尋找組態”。如果堆疊建立失敗，則檢閱顯示在主控台中的錯誤訊息，以協助您解決問題。如需 AWS CloudFormation 針對堆疊中的錯誤進行故障診斷的詳細資訊，請參閱AWS CloudFormation 《 使用者指南》中的故障診斷錯誤。

更新 Puppet Enterprise Server 的 OpsWorks 以使用自訂網域

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節說明如何使用伺服器備份來建立新的伺服器，以更新現有的 OpsWorks for Puppet Enterprise 伺服器來使用自訂網域和憑證。基本上，您要複製現有的 OpsWorks for Puppet Enterprise 2.0 伺服器，方法是從備份建立新伺服器，然後將新伺服器設定為使用自訂網域、憑證和私有金鑰。

主題

- 先決條件
- 限制
- 更新伺服器以使用自訂網域
- 另請參閱

先決條件

以下是更新現有 OpsWorks for Puppet Enterprise 伺服器以使用自訂網域和憑證的要求。

- 您要更新 （或複製） 的伺服器必須執行 Puppet Enterprise 2019.8.5。
- 決定要使用哪個備份來建立新伺服器。您必須至少有一個可用的備份，以供您要更新的伺服器使用。如需 OpsWorks for Puppet Enterprise 備份的詳細資訊，請參閱 備份 Puppet Enterprise Server 的 OpsWorks 。
- 準備好您要用來建立做為備份來源的現有伺服器的服務角色和執行個體描述檔 ARN。
- 請確定您正在執行最新版本的 AWS CLI。如需更新 AWS CLI 工具的詳細資訊，請參閱《AWS 命令列界面使用者指南》中的安裝 AWS CLI 。

限制

當您透過從備份建立新伺服器來更新現有伺服器時，新伺服器不能與現有 OpsWorks for Puppet Enterprise 伺服器完全相同。

- 您只能使用 AWS CLI 或其中一個 AWS SDKs來完成此程序。您無法使用 AWS Management Console從備份建立新伺服器。
- 新伺服器不能使用與帳戶內及 AWS 區域內現有伺服器相同的名稱。名稱必須與您做為備份來源使用的現有伺服器不同。
- 連接到現有伺服器的節點不會由新伺服器管理。您必須執行下列其中一項作業。
  - 連接不同的節點，因為節點不能由多個 Puppet 主伺服器管理。
  - 將節點從現有伺服器 (備份的來源) 移轉至新伺服器和新的自訂網域端點。如需如何移轉節點的詳細資訊，請參閱 Puppet Enterprise 文件。

更新伺服器以使用自訂網域

若要更新現有的 Puppet 主伺服器，您可以製作它的副本，方法是執行 `create-server` 命令，並新增參數以指定備份、自訂網域、自訂憑證和自訂私密金鑰。

1. 如果您沒有可在 `create-server` 命令中指定的服務角色或執行個體描述檔 ARN，請遵循 使用 建立 Chef Automate 伺服器 AWS CLI 中的步驟 1-5 來建立可使用的服務角色和執行個體設定檔。
2. 如果您還沒有這樣做，請找到現有的 Puppet 主伺服器 (您想以此伺服器為基礎建立使用自訂網域的新伺服器) 的備份。執行下列命令，以顯示您的帳戶和區域中所有 OpsWorks for Puppet Enterprise 備份的相關資訊。請記下您要使用的備份 ID。

   ```
   aws opsworks-cm --region region name describe-backups
   ```

3. 執行 `create-server`命令來建立 OpsWorks for Puppet Enterprise 伺服器。

   - `--engine` 值為 `Puppet`、 `--engine-model` 為 `Monolithic`、 `--engine-version`為 `2019`或 `2017`。
   - 伺服器名稱在 AWS 您的帳戶中在每個區域中必須是唯一的。伺服器名稱開頭必須是字母，後面允許字母、數字或連字號 (-)，最多可包含 40 個字元。
   - 使用您在步驟 3 和 4 中複製的執行個體描述檔 ARN 和服務角色 ARN。
   - 有效的執行個體類型為 `c4.large`、`c4.xlarge` 或 `c4.2xlarge`。如需這些執行個體類型規格的詳細資訊，請參閱《Amazon EC2 使用者指南》中的執行個體類型。
   - `--engine-attributes` 參數為選用；如果您未指定 Puppet 管理員密碼，伺服器建立程序會為您產生密碼。如果您新增 `--engine-attributes`，請指定 `PUPPET_ADMIN_PASSWORD`，這是用於登入 Puppet Enterprise 主控台網頁的管理員密碼。此密碼必須使用 8 到 32 個 ASCII 字元。

- SSH 金鑰對為選用，但如果您需要重設主控台管理員密碼，它可協助您連線至 Puppet 主伺服器。如需建立 SSH 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。 Amazon EC2
- 若要使用自訂網域，請將下列參數新增至您的命令。否則，Puppet 主伺服器建立程序會自動為您產生端點。需要所有三個參數才能設定自訂網域。如需使用這些參數的其他需求的相關資訊，請參閱 CM API AWS OpsWorks 參考中的 CreateServer。
  - `--custom-domain` - 伺服器的選用公有端點，例如 `https://aws.my-company.com`。
  - `--custom-certificate` - PEM 格式的 HTTPS 憑證。此值可以是單一、自我簽署的憑證或憑證鏈。
  - `--custom-private-key` - PEM 格式的私密金鑰，以便利用 HTTPS 連線至伺服器。私密金鑰不得加密，不能受密碼或密碼短語保護。
- 需要每週系統維護。有效值必須以下列格式指定：`DDD:HH:MM`。指定的時間是以國際標準時間 (UTC) 表示。如果您未指定 `--preferred-maintenance-window` 的值，預設值為星期二、星期三或星期五的隨機一小時時段。
- `--preferred-backup-window` 的有效值必須以下列其中一種格式指定：`HH:MM` 表示每日備份，或 `DDD:HH:MM` 表示每週備份。指定的時間是以 UTC 表示。預設值為隨機的每日開始時間。若要退出自動備份，請改為新增參數 `--disable-automated-backup`。
- 針對 `--security-group-ids`，輸入一或多個安全群組 ID，並以空格分隔。
- 針對 `--subnet-ids`，輸入子網路 ID。

```
aws opsworks-cm create-server --engine "Puppet" --engine-model "Monolithic"
 --engine-version "2019" --server-name "server_name" --instance-profile-arn
 "instance_profile_ARN" --instance-type "instance_type" --engine-attributes
 '{"PUPPET_ADMIN_PASSWORD":"ASCII_password"}' --key-pair "key_pair_name" --
preferred-maintenance-window "ddd:hh:mm" --preferred-backup-window "ddd:hh:mm"
 --security-group-ids security_group_id1 security_group_id2 --service-role-arn
 "service_role_ARN" --subnet-ids subnet_ID
```

下列範例會建立使用自訂網域的 Puppet 主伺服器。

```
aws opsworks-cm create-server \
    --engine "Puppet" \
    --engine-model "Monolithic" \
    --engine-version "2019" \
    --server-name "puppet-02" \
```

```
    --instance-profile-arn "arn:aws:iam::1019881987024:instance-profile/aws-
opsworks-cm-ec2-role" \
    --instance-type "c4.large" \
    --engine-attributes '{"PUPPET_ADMIN_PASSWORD":"zZZzDj2DLYXSZFRv1d"}' \
    --custom-domain "my-puppet-master.my-corp.com" \
    --custom-certificate "-----BEGIN CERTIFICATE----- EXAMPLEqEXAMPLE== -----END
 CERTIFICATE-----" \
    --custom-private-key "-----BEGIN RSA PRIVATE KEY----- EXAMPLEqEXAMPLE= -----END
 RSA PRIVATE KEY-----" \
    --key-pair "amazon-test"
    --preferred-maintenance-window "Mon:08:00" \
    --preferred-backup-window "Sun:02:00" \
    --security-group-ids sg-b00000001 sg-b0000008 \
    --service-role-arn "arn:aws:iam::044726508045:role/service-role/aws-opsworks-
cm-service-role" \
    --subnet-ids subnet-383daa71
```

4. OpsWorks for Puppet Enterprise 約需 15 分鐘才能建立新的伺服器。在 `create-server` 命令的輸出中，複製 `Endpoint` 屬性的值。以下是範例。

   ```
   "Endpoint": "puppet-2019-exampleexample.opsworks-cm.us-east-1.amazonaws.com"
   ```

   請勿關閉 `create-server` 命令的輸出或關閉您的 shell 工作階段，因為輸出可能包含不會再顯示的重要資訊。若要從 `create-server` 結果取得密碼和入門套件，請前往下一個步驟。

5. 如果您選擇讓 OpsWorks for Puppet Enterprise 為您產生密碼，您可以使用 jq 等 JSON 處理器，從`create-server`結果中擷取可用格式的密碼。安裝 jq 之後，您可以執行下列命令來解壓縮 Puppet 管理員密碼和入門套件。如果您未在步驟 3 中提供自己的密碼，請務必將解壓縮的管理員密碼儲存在方便但安全的位置。

   ```
   #Get the Puppet password:
   cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
    "PUPPET_ADMIN_PASSWORD") | .Value'

   #Get the Puppet Starter Kit:
   cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
    "PUPPET_STARTER_KIT") | .Value' | base64 -D > starterkit.zip
   ```

> **Note**
>
> 您無法在 AWS Management Console中重新產生新的 Puppet 主伺服器入門套件。當您使用 建立 Puppet 主伺服器時 AWS CLI，請執行上述`jq`命令，將`create-server`結果中的 base64 編碼入門套件儲存為 ZIP 檔案。

6. 或者，如果您沒有從`create-server`命令結果中擷取入門套件，您可以從 OpsWorks for Puppet Enterprise 主控台中的伺服器屬性頁面下載新的入門套件。
7. 如果您不是使用自訂網域，請繼續下一個步驟。如果您將自訂網域與伺服器搭配使用，請在企業的 DNS 管理工具中建立 CNAME 項目，將自訂網域指向您在步驟 4 中複製的 OpsWorks for Puppet Enterprise 端點。在完成此步驟之前，您無法連線或登入具有自訂網域的伺服器。
8. 完成伺服器建立程序之後，繼續使用入門套件設定 Puppet 主伺服器。

另請參閱

- 使用 建立 Puppet Enterprise Master AWS CLI
- 備份和還原 Puppet Enterprise Server 的 OpsWorks
- CM API AWS OpsWorks 參考中的 CreateServer
- 《AWS CLI 命令參考》中的 create-server 一節

在 AWS OpsWorks for Puppet Enterprise 資源上使用標籤

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

標籤是單字或片語，會以中繼資料形式用於識別和組織您的 AWS 資源。在 OpsWorks for Puppet Enterprise 中，資源最多可有 50 個使用者套用的標籤。每個標籤皆包含一個索引鍵與一個選用值。您可以在 OpsWorks for Puppet Enterprise 中將標籤套用至下列資源：

- Puppet Enterprise 伺服器的 OpsWorks

- 適用於 Puppet Enterprise 伺服器的 OpsWorks 備份

AWS 資源上的標籤可協助您追蹤成本、控制資源的存取、將任務自動化的群組資源，或依用途或生命週期階段組織資源。如需標籤優點的詳細資訊，請參閱AWS 帳單與成本管理 《 使用者指南》中的 AWS 答案中的 AWS 標記策略和使用成本分配標籤。

若要使用標籤來控制 OpsWorks for Puppet Enterprise 伺服器或備份的存取，請在 AWS Identity and Access Management (IAM) 中建立或編輯政策陳述式。如需詳細資訊，請參閱AWS Identity and Access Management 《 使用者指南》中的使用資源標籤控制 AWS 資源的存取。

當您將標籤套用到 OpsWorks for Puppet Enterprise 主伺服器時，標籤也會套用到主伺服器的備份、存放備份的 Amazon S3 儲存貯體、主伺服器的 Amazon EC2 執行個體、存放於 的主伺服器的秘密 AWS Secrets Manager，以及主伺服器的彈性 IP 地址。標籤不會傳播到 AWS OpsWorks 用於建立 Puppet 主伺服器的 AWS CloudFormation 堆疊。

主題

- 標籤在 中的運作方式 AWS OpsWorks for Puppet Enterprise
- 在 OpsWorks for Puppet Enterprise 中新增和管理標籤 （ 主控台 ）
- 在 OpsWorks for Puppet Enterprise (CLI) 中新增和管理標籤
- 另請參閱

標籤在 中的運作方式 AWS OpsWorks for Puppet Enterprise

在此版本中，您可以使用 AWS OpsWorks CM API 或 AWS Management Console來新增和管理標籤。 AWS OpsWorks CM 也會嘗試將新增至伺服器的標籤新增至與伺服器相關聯的 AWS 資源，包括 EC2 執行個體、Secrets Manager 中的秘密、彈性 IP 地址、安全群組、S3 儲存貯體和備份。

下表提供如何在 OpsWorks for Puppet Enterprise 中新增和管理標籤的概觀。

動作	執行方式
將標籤新增至您手動建立的新 OpsWorks for Puppet Enterprise 伺服器或備份。	• 選擇 Create Puppet Enterprise server (建立 Puppet Enterprise 伺服器)，然後在 Configure advanced settings (配置進階設定) 頁面上新增標籤。

動作	執行方式
	• 在現有伺服器的備份頁面上選擇建立備份，然後在建立 Puppet Enterprise 伺服器備份頁面上新增標籤。 • 將 Tags 參數新增到 CreateServer 或 CreateBackup 命令。
檢視資源上的標籤。	• 在伺服器的詳細資訊頁面上，選擇導覽窗格中的 Tags (標籤)。 • 在伺服器的 Backups (備份) 頁面上，選取備份，然後選擇 Edit backup (編輯備份)。 • 執行 ListTagsForResource 命令。
將標籤新增至現有的 OpsWorks for Puppet Enterprise 伺服器或備份，無論備份是手動還是自動建立。	• 在伺服器的詳細資訊頁面上，選擇導覽窗格中的 Tags (標籤)，然後選擇 Edit (編輯)。 • 在伺服器的 Backups (備份) 頁面上，選取備份，然後選擇 Edit backup (編輯備份)。 • 執行 TagResource 命令。
從資源刪除標籤。	• 在伺服器的詳細資訊頁面上，選擇導覽窗格中的 Tags (標籤)，然後選擇 Edit (編輯)。選擇您要刪除的標籤旁的 X。 • 在伺服器的 Backups (備份) 頁面上，選取備份，然後選擇 Edit backup (編輯備份)。選擇您要刪除的標籤旁的 X。 • 執行 UntagResource 命令。

`DescribeServers` 和 `DescribeBackups` 回應不包含標籤資訊。若要顯示標籤，請使用 `ListTagsForResource` API。

在 OpsWorks for Puppet Enterprise 中新增和管理標籤（主控台）

本節中的程序將在 AWS Management Console中執行。

新增標籤時，索引鍵不能為空。此索引鍵最多可包含 127 個字元，並且只能包含 Unicode 字母、數字或分隔符號，或是下列特殊字元：`+ - = . _ : / @`。標籤值是選擇性的。您可以新增具有索引鍵但

沒有值的標籤。值最多可包含 255 個字元，並且只能包含 Unicode 字母、數字或分隔符號，或是下列特殊字元：+ - = . _ : / @

主題

- 將標籤新增至 Puppet Enterprise Server 的新 OpsWorks （ 主控台 ）
- 新增標籤至新的備份 (主控台)
- 新增或檢視現有伺服器上的標籤 (主控台)
- 新增或檢視現有備份上的標籤 (主控台)
- 從伺服器刪除標籤 (主控台)
- 從備份刪除標籤 (主控台)

將標籤新增至 Puppet Enterprise Server 的新 OpsWorks （ 主控台 ）

1. 請務必完成建立 OpsWorks for Puppet Enterprise 主控端的任何先決條件。
2. 依照 使用 建立 Puppet Enterprise Master AWS Management Console 中的步驟 1-8 執行。
3. 指定自動備份設定後，請在設定進階設定頁面的標籤區域中新增標籤。您最多可新增 50 個標籤。完成標籤新增作業時，選擇 Next (下一步)。
4. 繼續進行 使用 建立 Puppet Enterprise Master AWS Management Console 的步驟 11，並檢閱您為新伺服器選擇的設定。

新增標籤至新的備份 (主控台)

1. 在 OpsWorks for Puppet Enterprise 首頁上，選擇現有的 Puppet 主伺服器。
2. 在伺服器的詳細資訊頁面中，選擇導覽窗格中的 Backups (備份)。
3. 在 Backups (備份) 頁面上，選擇 Create backup (建立備份)。
4. 新增標籤。完成新增標籤作業時，選擇 Create (建立)。

新增或檢視現有伺服器上的標籤 (主控台)

1. 在 OpsWorks for Puppet Enterprise 首頁上，選擇現有的 Puppet 主伺服器以開啟其詳細資訊頁面。
2. 在導覽窗格中選擇 Tags (標籤)，或在詳細資訊頁面底部選擇 View all tags (檢視所有標籤)。
3. 在 Tags (標籤) 頁面上，選擇 Edit (編輯)。

4. 新增或編輯伺服器上的標籤。完成時，請選擇 Save (儲存)。

> **Note**
>
> 請注意，變更 Puppet 主伺服器上的標籤也會變更與該伺服器相關聯的資源 (例如 EC2 執行個體、彈性 IP 地址、安全群組、S3 儲存貯體和備份) 上的標籤。

新增或檢視現有備份上的標籤 (主控台)

1. 在 OpsWorks for Puppet Enterprise 首頁上，選擇現有的 Puppet 主伺服器以開啟其詳細資訊頁面。
2. 在導覽窗格中選擇 Backups (備份)，或在詳細資訊頁面的 Recent backups (最近備份) 區域中，選擇 View all backups (檢視所有備份)。
3. 在 Backups (備份) 頁面上，選擇要管理的備份，然後選擇 Edit backup (編輯備份)。
4. 新增或編輯備份上的標籤。完成後，選擇 Update (更新)。

從伺服器刪除標籤 (主控台)

1. 在 OpsWorks for Puppet Enterprise 首頁上，選擇現有的 Puppet 主伺服器以開啟其詳細資訊頁面。
2. 在導覽窗格中選擇 Tags (標籤)，或在詳細資訊頁面底部選擇 View all tags (檢視所有標籤)。
3. 在 Tags (標籤) 頁面上，選擇 Edit (編輯)。
4. 選擇標籤旁邊的 X 以刪除標籤。完成時，請選擇 Save (儲存)。

> **Note**
>
> 請注意，變更 Puppet 主伺服器上的標籤也會變更與該伺服器相關聯的資源 (例如 EC2 執行個體、彈性 IP 地址、安全群組、S3 儲存貯體和備份) 上的標籤。

從備份刪除標籤 (主控台)

1. 在 OpsWorks for Puppet Enterprise 首頁上，選擇現有的 Puppet 主伺服器以開啟其詳細資訊頁面。

2. 在導覽窗格中選擇 Backups (備份)，或在詳細資訊頁面的 Recent backups (最近備份) 區域中，選擇 View all backups (檢視所有備份)。
3. 在 Backups (備份) 頁面上，選擇要管理的備份，然後選擇 Edit backup (編輯備份)。
4. 選擇標籤旁邊的 X 以刪除標籤。完成後，選擇 Update (更新)。

在 OpsWorks for Puppet Enterprise (CLI) 中新增和管理標籤

本節中的程序將在 AWS CLI中執行。開始處理標籤 AWS CLI 之前，請務必執行最新版本的 。如需安裝或更新 的詳細資訊 AWS CLI，請參閱AWS Command Line Interface 《 使用者指南》中的安裝 AWS CLI。

新增標籤時，索引鍵不能為空。此索引鍵最多可包含 127 個字元，並且只能包含 Unicode 字母、數字或分隔符號，或是下列特殊字元：`+ - = . _ : / @`。標籤值是選擇性的。您可以新增具有索引鍵但沒有值的標籤。值最多可包含 255 個字元，並且只包含 Unicode 字母、數字或分隔符號，或是下列特殊字元：`+ - = . _ : / @`

主題

- 將標籤新增至 Puppet Enterprise Server (CLI) 的新 OpsWorks
- 新增標籤至新的備份 (CLI)
- 新增標籤至現有伺服器或備份 (CLI)
- 列出資源標籤 (CLI)
- 從資源刪除標籤 (CLI)

將標籤新增至 Puppet Enterprise Server (CLI) 的新 OpsWorks

您可以在建立 OpsWorks for Puppet Enterprise 伺服器時 AWS CLI ，使用 新增標籤。此程序並未完整說明如何建立伺服器。如需如何使用 建立 OpsWorks for Puppet Enterprise 伺服器的詳細資訊 AWS CLI，請參閱本指南使用 建立 Puppet Enterprise Master AWS CLI中的 。您最多可以為每個伺服器新增 50 個標籤。

1. 請務必完成建立 OpsWorks for Puppet Enterprise 伺服器的任何先決條件。
2. 完成 使用 建立 Puppet Enterprise Master AWS CLI 的步驟 1-4。
3. 在步驟 5 時，執行 `create-server` 命令時請將 `--tags` 參數新增至命令，如下列範例所示。

```
aws opsworks-cm create-server ... --tags Key=Key1,Value=Value1
 Key=Key2,Value=Value2
```

以下示範僅顯示 `create-server` 命令的標籤部分。

```
aws opsworks-cm create-server ... --tags Key=Stage,Value=Production
 Key=Department,Value=Marketing
```

4. 完成使用 建立 Puppet Enterprise Master AWS CLI 中剩餘的步驟。若要確認您的標籤是否已新增至新的伺服器，請遵循本主題 列出資源標籤 (CLI) 中的步驟。

新增標籤至新的備份 (CLI)

當您建立新的 OpsWorks for Puppet Enterprise 伺服器手動備份時，您可以使用 AWS CLI 新增標籤。此程序並未完整說明如何建立手動備份。如需如何建立手動備份的詳細資訊，請參閱 中的「在 中執行手動備份 AWS CLI」備份 Puppet Enterprise Server 的 OpsWorks 。您最多可以為備份新增 50 個標籤。如果伺服器有標籤，則新的備份會自動加上伺服器的標籤。

根據預設，當您建立新的 OpsWorks for Puppet Enterprise 伺服器時，會啟用自動備份。您可以依本主題中新增標籤至現有伺服器或備份 (CLI)所述，透過執行 `tag-resource` 命令，將標籤新增至自動備份。

- 若要在建立備份時將標籤新增至手動備份，請執行下列命令。僅展示命令的標籤部分。如需完整`create-backup`命令的範例，請參閱「若要在 AWS CLI「的「中執行手動備份備份 Puppet Enterprise Server 的 OpsWorks 。

```
aws opsworks-cm create-backup ... --tags Key=Key1,Value=Value1
 Key=Key2,Value=Value2
```

以下示範僅顯示 `create-backup` 命令的標籤部分。

```
aws opsworks-cm create-backup ... --tags Key=Stage,Value=Production
 Key=Department,Value=Marketing
```

新增標籤至現有伺服器或備份 (CLI)

您可以執行 `tag-resource`命令，將標籤新增至現有的 OpsWorks for Puppet Enterprise 伺服器或備份 （無論是自動或手動建立備份）。指定要在其中新增標籤之目標資源的 Amazon 資源編號 (ARN)。

1. 若要取得標籤將套用其中之資源的 ARN：

   - 用於伺服器時，請執行 `describe-servers --server-name` *`server_name`*。此命令執行後結果是顯示伺服器 ARN。
   - 用於備份時，請執行 `describe-backups --backup-id` *`backup_ID`*。此命令執行後結果是顯示備份 ARN。您也可以執行 `describe-backups --server-name` *`server_name`* 以顯示特定 OpsWorks for Puppet Enterprise 伺服器的所有備份的相關資訊。

   以下示範僅顯示 `describe-servers --server-name opsworks-cm-test` 命令結果的 `ServerArn`。此 `ServerArn` 值會新增至 `tag-resource` 命令，以便將標籤新增至伺服器。

   ```
   {
       "Servers": [
           {
               ...
               "ServerArn": "arn:aws:opsworks-cm:us-west-2:123456789012:server/
   opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
           }
       ]
   }
   ```

2. 使用您在步驟 1 中傳回的 ARN 執行 `tag-resource` 命令。

   ```
   aws opsworks-cm tag-resource --resource-arn "server_or_backup_ARN" --tags
    Key=Key1,Value=Value1 Key=Key2,Value=Value2
   ```

   以下是範例。

   ```
   aws opsworks-cm tag-resource --resource-arn "arn:aws:opsworks-cm:us-
   west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
    --tags Key=Stage,Value=Production Key=Department,Value=Marketing
   ```

3. 若要確認標籤是否已成功新增，請繼續進行下一個程序 列出資源標籤 (CLI)。

列出資源標籤 (CLI)

您可以執行 `list-tags-for-resource`命令，以顯示連接到 OpsWorks for Puppet Enterprise 伺服器或備份的標籤。指定要檢視其中標籤之目標資源的 ARN。

1. 若要取得將要列出其中標籤之資源的 ARN：
   - 用於伺服器時，請執行 `describe-servers --server-name` *`server_name`*。此命令執行後結果是顯示伺服器 ARN。
   - 用於備份時，請執行 `describe-backups --backup-id` *`backup_ID`*。此命令執行後結果是顯示備份 ARN。您也可以執行 `describe-backups --server-name` *`server_name`* 以顯示特定 OpsWorks for Puppet Enterprise 伺服器的所有備份的相關資訊。
2. 使用您在步驟 1 中傳回的 ARN 執行 `list-tags-for-resource` 命令。

```
aws opsworks-cm list-tags-for-resource --resource-arn "server_or_backup_ARN"
```

以下是範例。

```
aws opsworks-cm tag-resource --resource-arn "arn:aws:opsworks-cm:us-
west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
```

如果資源上有標籤，此命令結果將會傳回如下。

```
{
    "Tags": [
        {
            "Key": "Stage",
            "Value": "Production"
        },
        {
            "Key": "Department",
            "Value": "Marketing"
        }
    ]
}
```

從資源刪除標籤 (CLI)

您可以執行 `untag-resource`命令，從 OpsWorks for Puppet Enterprise 伺服器或備份刪除標籤。如果資源遭到刪除，資源上的標籤也會遭到刪除。指定要從其中移除標籤之目標資源的 Amazon 資源編號 (ARN)。

1. 若要取得將要移除其中標籤之資源的 ARN：
   - 用於伺服器時，請執行 `describe-servers --server-name` *`server_name`*。此命令執行後結果是顯示伺服器 ARN。
   - 用於備份時，請執行 `describe-backups --backup-id` *`backup_ID`*。此命令執行後結果是顯示備份 ARN。您也可以執行 `describe-backups --server-name` *`server_name`* 以顯示特定 OpsWorks for Puppet Enterprise 伺服器的所有備份的相關資訊。
2. 使用您在步驟 1 中傳回的 ARN 執行 `untag-resource` 命令。僅指定您要刪除的標籤。

   ```
   aws opsworks-cm untag-resource --resource-arn "server_or_backup_ARN" --tags
    Key=Key1,Value=Value1 Key=Key2,Value=Value2
   ```

   在此範例中，`untag-resource` 命令只會移除索引鍵為 `Stage`、和值為 `Production` 的標籤。

   ```
   aws opsworks-cm untag-resource --resource-arn "arn:aws:opsworks-cm:us-
   west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
    --tags Key=Stage,Value=Production
   ```

3. 若要確認已成功刪除標籤，請遵循本主題 列出資源標籤 (CLI) 中的步驟。

另請參閱

- 使用 建立 Puppet Enterprise Master AWS CLI
- 備份 Puppet Enterprise Server 的 OpsWorks
- AWS 加標籤策略
- AWS Identity and Access Management 使用者指南中的使用資源標籤控制 AWS 對資源的存取
- 《AWS 帳單與成本管理 使用者指南》中的使用成本分配標籤
- AWS OpsWorks CM API 參考中的 CreateBackup
- AWS OpsWorks CM API 參考中的 CreateServer
- AWS OpsWorks CM API 參考中的 TagResource

- AWS OpsWorks CM API 參考中的 ListTagsForResource
- AWS OpsWorks CM API 參考中的 UntagResource

備份和還原 Puppet Enterprise Server 的 OpsWorks

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節說明如何備份和還原 OpsWorks for Puppet Enterprise 伺服器。

主題

- 備份 Puppet Enterprise Server 的 OpsWorks
- 從 備份還原 Puppet Enterprise Server 的 OpsWorks

備份 Puppet Enterprise Server 的 OpsWorks

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以定義每日或每週定期 OpsWorks for Puppet Enterprise 伺服器備份，並讓服務代表您將備份存放在 Amazon Simple Storage Service (Amazon S3) 中。或者，您可以隨需手動備份。

由於備份存放在 Amazon S3 中，因此會產生額外費用。您可以定義備份保留期，最長為 30 代。您可以使用 AWS 支援管道提交服務請求，以變更該限制。傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

您可以將標籤新增至 OpsWorks for Puppet Enterprise 主控端的備份。如果您已將標籤新增至 OpsWorks for Puppet Enterprise 主伺服器，則 Puppet 主伺服器的自動備份會繼承這些標籤。如需有

關如何新增和管理備份標籤的詳細資訊，請參閱本指南中的 在 AWS OpsWorks for Puppet Enterprise 資源上使用標籤。

主題

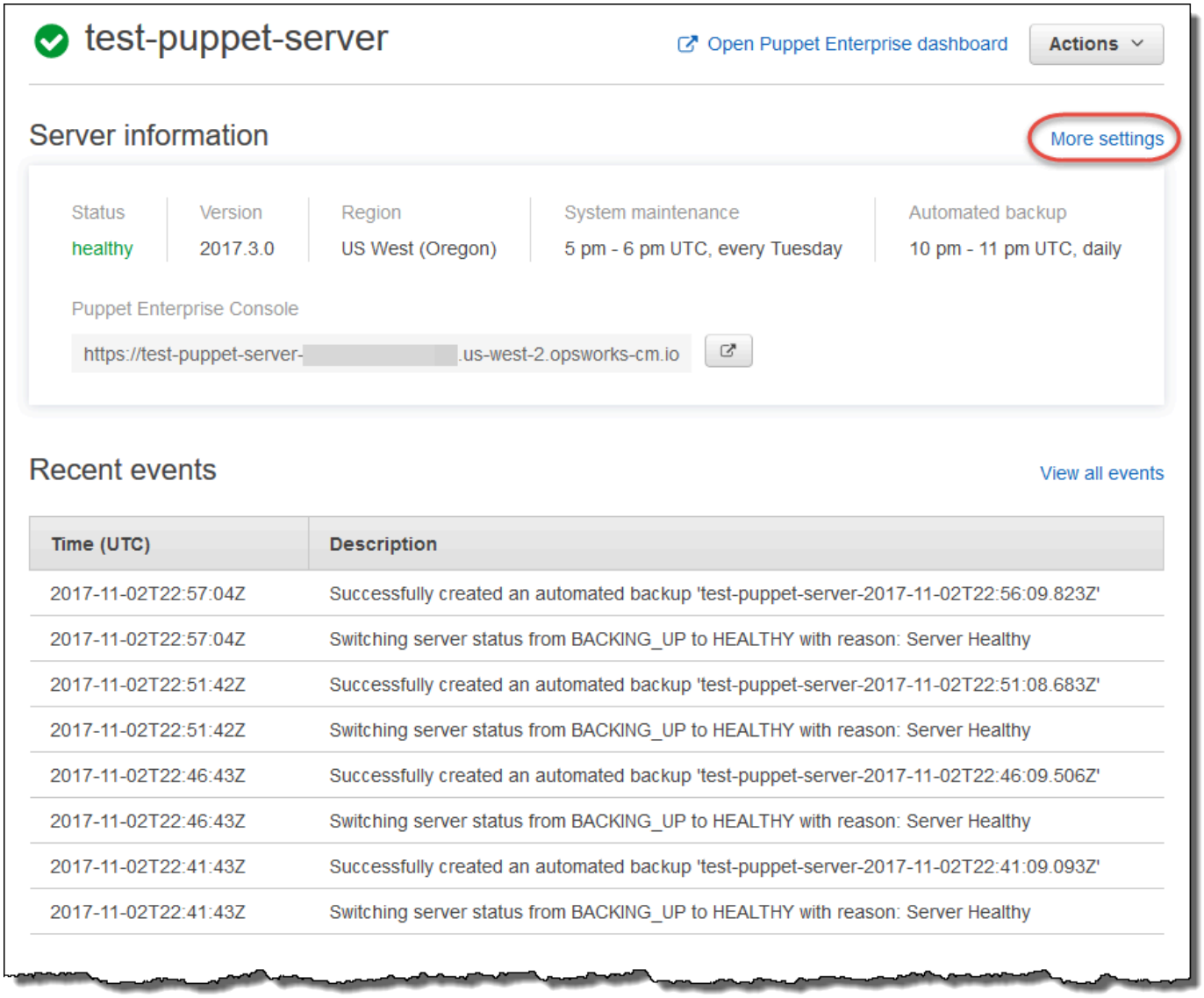
- 自動備份
- 手動備份
- 刪除備份

自動備份

當您設定 OpsWorks for Puppet Enterprise 伺服器時，您可以選擇自動或手動備份。OpsWorks for Puppet Enterprise 會在設定精靈之設定進階設定頁面的自動備份區段中選擇的小時內和當天啟動自動備份。當您的伺服器處於線上狀態之後，您即可在伺服器的屬性頁面上執行下列步驟，以變更備份設定。

變更自動備份設定

1. 在伺服器的屬性頁面中，選擇 More settings (更多設定)。



2. 若要關閉自動備份，請針對 Enable automated backups (啟用自動備份) 選項選擇 No (否)。儲存變更；您不需要繼續進行下一個步驟。
3. 在 Automated Backup (自動備份) 區段中，變更頻率、開始時間或要保留的版本。儲存您的變更。

手動備份

您可以隨時在 中啟動手動備份 AWS Management Console，或執行 AWS CLI create-backup 命令。手動備份不包含在最多 30 代的所儲存自動備份中。最多會儲存 10 個手動備份，而且必須從 Amazon S3 手動刪除。

在 中執行手動備份 AWS Management Console

1. 在 Puppet Enterprise servers (Puppet Enterprise 伺服器) 頁面中，選擇您要備份的伺服器。
2. 在伺服器屬性頁面的左側導覽窗格中，選擇 Backups (備份)。
3. 選擇 Create backup (建立備份)。
4. 當頁面的備份 Status (狀態) 欄中顯示綠色核取記號時，手動備份即已完成。

在 中執行手動備份 AWS CLI

您可以在建立新的 OpsWorks for Puppet Enterprise 伺服器手動備份時新增標籤。如需如何在建立手動備份時新增標籤的詳細資訊，請參閱新增標籤至新的備份 (CLI) 。

- 若要啟動手動備份，請執行下列 AWS CLI 命令。

```
aws opsworks-cm --region region name create-backup --server-name "Puppet server
 name" --description "optional descriptive string"
```

刪除備份

永久刪除備份時，即會將該備份從存放備份的 S3 儲存貯體中刪除。

在 中刪除備份 AWS Management Console

1. 在 Puppet Enterprise servers (Puppet Enterprise 伺服器) 頁面中，選擇您要備份的伺服器。
2. 在伺服器屬性頁面的左側導覽窗格中，選擇 Backups (備份)。
3. 選擇您要刪除的備份，然後選擇 Delete backup (刪除備份)。您一次只能選取一個備份。
4. 當系統提示您確認刪除時，請勾選 Delete the backup, which is stored in an S3 bucket (刪除存放在 S3 儲存貯體中的備份) 核取方塊，然後選擇 Yes, Delete (是，刪除)。

在 中刪除備份 AWS CLI

- 若要刪除備份，請執行下列 AWS CLI 命令，以您要刪除的備份 `--backup-id` ID 取代 的值。備份 ID 的格式為 *ServerName-yyyyMMddHHmmssSSS*。例如 **puppet-server-20171218132604388**。

```
aws opsworks-cm --region region name delete-backup --backup-id ServerName-
yyyyMMddHHmmssSSS
```

從 備份還原 Puppet Enterprise Server 的 OpsWorks

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

瀏覽可用的備份後，您可以輕鬆選擇還原 OpsWorks for Puppet Enterprise 伺服器的時間點。伺服器備份包含組態管理軟體持久性資料，例如模組、類別、節點關聯、資料庫資訊 (包括報告、資料等)。執行伺服器就地還原 （即將現有的 OpsWorks for Puppet Enterprise 伺服器還原至新的 EC2 執行個體） 會重新註冊您在用來還原伺服器之備份時註冊的節點，並在還原成功時將流量切換到新執行個體，且還原的 OpsWorks for Puppet Enterprise 伺服器狀態為 `Healthy`。還原至新建立的 OpsWorks for Puppet Enterprise 伺服器不會維護節點連線。還原伺服器時並不會更新 Puppet 軟體的版本；其會套用您所選備份中可用的相同 Puppet 版本和組態管理資料。

還原伺服器通常比建立新伺服器需要更多時間；時間取決於您選擇的備份大小。還原完成後，舊的 EC2 執行個體會保留在 `Running` 或 `Stopped` 狀態，但只是暫時狀態。這個狀態最終將會結束。

在此版本中，您可以使用 AWS CLI 還原 OpsWorks for Puppet Enterprise 中的 Puppet 主伺服器。

> **Note**
>
> 您也可以執行 restore-server 命令，以變更目前的執行個體類型；或者，還原或設定您的 SSH 金鑰 (如果遺失或受損的話)。

從備份還原伺服器

1. 在 中 AWS CLI，執行下列命令，以傳回可用備份及其 IDs的清單。請記下您要使用的備份 ID。備份 ID 的格式為 *myServerName-yyyyMMddHHmmssSSS*。

   ```
   aws opsworks-cm --region region name describe-backups
   ```

2. 執行下列命令。

```
aws opsworks-cm --region region name restore-server --backup-id "myServerName-
yyyyMMddHHmmssSSS" --instance-type "Type of instance" --key-pair "name of your EC2
 key pair" --server-name "name of Puppet master"
```

以下是範例。

```
aws opsworks-cm --region us-west-2 restore-server --backup-id
 "MyPuppetServer-20161120122143125" --server-name "MyPuppetServer"
```

3. 等待還原完成。

OpsWorks for Puppet Enterprise 中的系統維護

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

強制性系統維護可確保 AWS Puppet 伺服器的最新版本，包括安全性更新，一律在 OpsWorks for Puppet Enterprise 伺服器上執行。每週至少需要進行一次系統維護。如有需要 AWS CLI，您可以使用設定每日自動維護。除了排定的系統維護之外，您也可以使用 隨需 AWS CLI 執行系統維護。

當有新版本的 Puppet 軟體可用時，系統維護的設計會在其通過 AWS 測試之後，自動更新伺服器上 Puppet Server 的版本。AWS 會執行廣泛的測試，以驗證 Puppet 升級是否可供生產使用，而且不會中斷現有的客戶環境，因此 Puppet 軟體版本與現有 OpsWorks for Puppet Enterprise 伺服器的應用程式可用性之間可能會有延遲。若要隨需更新 Puppet 軟體的可用版本，請參閱本主題中的隨需啟動系統維護。

系統維護會從作為維護程序一部分執行的備份啟動新的執行個體，這有助於降低因定期維護而降級或受損的 Amazon EC2 執行個體所造成的風險。

> **Important**
>
> 系統維護會刪除您已新增至 OpsWorks for Puppet Enterprise 伺服器的任何檔案或自訂組態。如需如何修復組態或檔案遺失的詳細資訊，請參閱本主題中的在維護之後還原自訂組態和檔案。

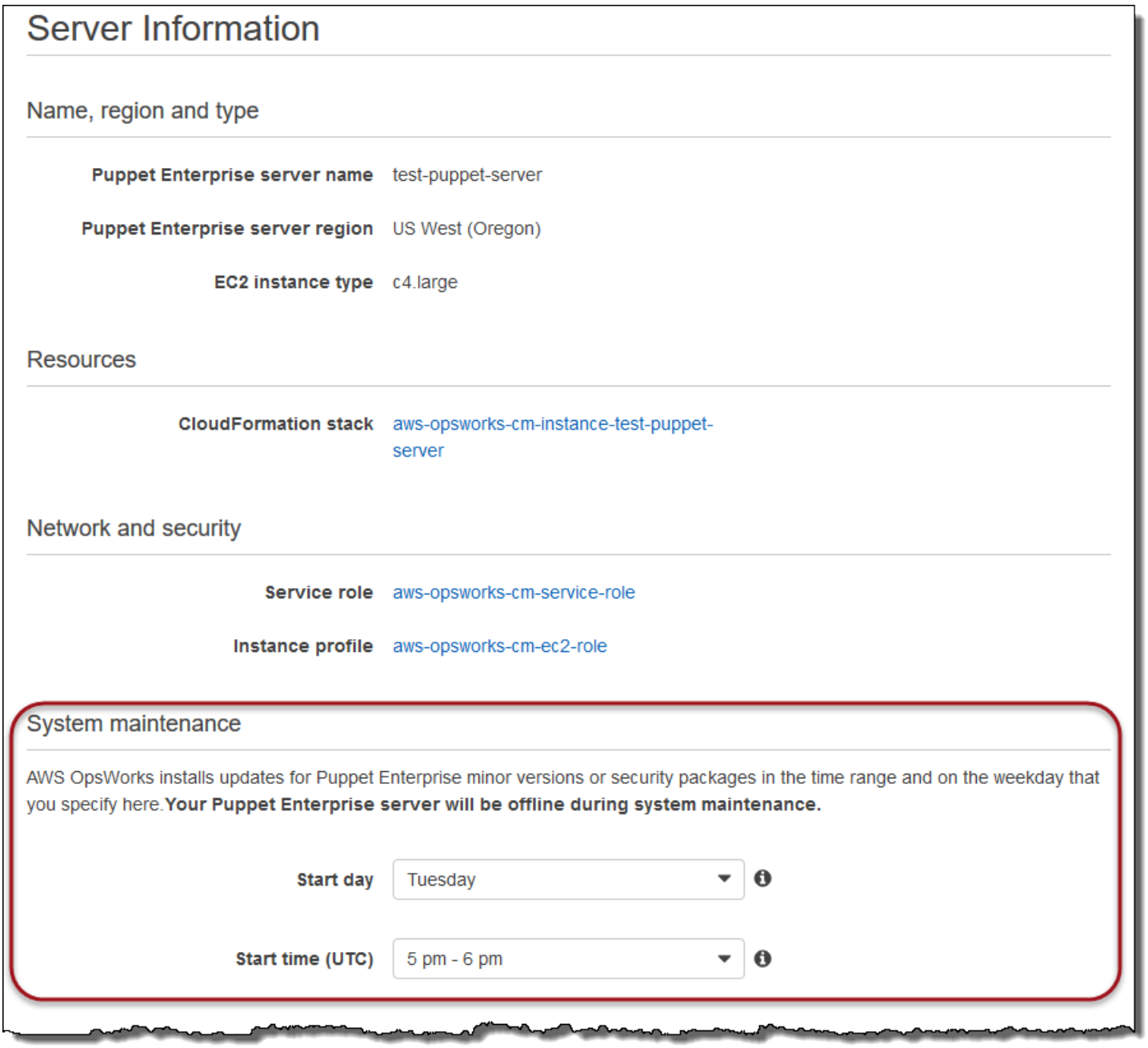
主題

- 設定系統維護
- 隨需啟動系統維護
- 在維護之後還原自訂組態和檔案

設定系統維護

當您建立新的 OpsWorks for Puppet Enterprise 伺服器時，您可以在國際標準時間 (UTC) 中設定工作日和時間，以啟動系統維護。維護會在您指定的小時啟動。因為您應預期伺服器在系統維護期間離線，請選擇一般工作時間內伺服器需求較低的時間。當維護正在進行中時，伺服器狀態將為 `UNDER_MAINTENANCE`。

您也可以變更現有 OpsWorks for Puppet Enterprise 伺服器上的系統維護設定，方法是變更伺服器設定頁面之系統維護區域中的設定，如下列螢幕擷取畫面所示。



在 System maintenance (系統維護) 區段中，將日和小時設為您希望開始系統維護的時間。

使用 設定系統維護 AWS CLI

您也可以使用 AWS CLI設定系統維護的自動開始時間。 AWS CLI 可視需要省略三字元工作日字首，以設定每日自動維護。

在 `create-server` 命令中，在指定建立伺服器執行個體的必要項目 (例如執行個體類型、執行個體描述檔 ARN 和伺服器角色 ARN) 之後，將 `--preferred-maintenance-window` 參數新增到您的命

令。在下列 `create-server` 範例中，`--preferred-maintenance-window` 設為 `Mon:08:00`，表示您已將維護設定在每週一 UTC 上午 8:00 開始 data at rest。

```
aws opsworks-cm create-server --engine "Puppet" --engine-model "Monolithic"
 --engine-version "2017" --server-name "puppet-06" --instance-profile-arn
 "arn:aws:iam::1119001987000:instance-profile/aws-opsworks-cm-ec2-role"
 --instance-type "c4.large" --key-pair "amazon-test" --service-role-arn
 "arn:aws:iam::044726508045:role/aws-opsworks-cm-service-role" --preferred-maintenance-
window "Mon:08:00"
```

在 `update-server` 命令中，您可以單獨更新 `--preferred-maintenance-window` 的值 (若需要的話)。在下列範例中，維護時段已設為週五 UTC 下午 6:15。data at rest。

```
aws opsworks-cm update-server --server-name "puppet-06" --preferred-maintenance-window
 "Fri:18:15"
```

若要將維護時段的開始時間變更為每天 UTC 下午 6:15，請省略三個字元的工作日字首，如以下範例所示。

```
aws opsworks-cm update-server --server-name "puppet-06" --preferred-maintenance-window
 "18:15"
```

如需使用 設定偏好系統維護時段的詳細資訊 AWS CLI，請參閱 create-server 和 update-server。

隨需啟動系統維護

若要隨需開始系統維護，在您設定的每週或每日自動維護之外，請執行下列 AWS CLI 命令。您無法在 AWS Management Console中啟動隨需維護。

```
aws opsworks-cm start-maintenance --server-name server_name
```

如需此命令的詳細資訊，請參閱 start-maintenance。

在維護之後還原自訂組態和檔案

系統維護可以刪除或變更您已新增至 OpsWorks for Puppet Enterprise 伺服器的自訂檔案或組態。

如果在維護執行後，您的 Puppet 主伺服器缺少您使用 或 SSH 新增的檔案RunCommand或設定，您可以使用 Amazon Machine Image (AMI) 來啟動新的 Amazon EC2 執行個體。可用的 AMI 是由伺服器的維護前組態建置而成的。

新的執行個體狀態會和維護前的 Puppet 主伺服器相同，因此會包含您遺失的檔案和設定。

> **Important**
>
> 您無法使用新的執行個體還原您的伺服器。執行個體無法做為 Puppet 主伺服器執行。您只能使用執行個體復原您的檔案和組態設定。

若要從 AMI 啟動 EC2 執行個體，請在 Amazon EC2 主控台中開啟啟動精靈，選擇我的 AMIs，然後選擇具有伺服器名稱的 AMI。請依照任何其他執行個體啟動的作法，遵循 Amazon EC2 精靈步驟。

在適用於 Puppet Enterprise 的 OpsWorks 中自動新增節點

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本主題說明如何自動將 Amazon Elastic Compute Cloud (Amazon EC2) 節點新增至 OpsWorks for Puppet Enterprise 伺服器。在為 Puppet 主伺服器新增要管理的節點中，您學習到如何使用 `associate-node` 命令一次新增一個節點至您的 Puppet Enterprise 伺服器。本主題的程式碼示範如何使用無人執行的方法自動新增節點。建議新節點的全自動（或自動）關聯方法是設定 Amazon EC2 使用者資料。根據預設，OpsWorks for Puppet Enterprise 伺服器已針對 Ubuntu、Amazon Linux 和 RHEL 節點作業系統`puppet-agent`提供 。

如需如何取消節點關聯的詳細資訊，請參閱本指南取消節點與 Puppet Enterprise Server OpsWorks 的關聯中的 和 OpsWorks for Puppet Enterprise API 文件`disassociate-node`中的 。

步驟 1：建立 IAM 角色以用作您的執行個體描述檔

建立 AWS Identity and Access Management (IAM) 角色以用作您的 EC2 執行個體描述檔，並將下列政策連接至 IAM 角色。此政策可讓 `opsworks-cm` API 在節點註冊期間與 EC2 執行個體通訊。如需執行個體描述檔的詳細資訊，請參閱 Amazon EC2 文件中的使用執行個體描述檔。如需如何建立 IAM 角色的資訊，請參閱 Amazon EC2 文件中的在主控台中建立 IAM 角色。 Amazon EC2

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "opsworks-cm:AssociateNode",
                "opsworks-cm:DescribeNodeAssociationStatus",
                "opsworks-cm:DescribeServers",
                "ec2:DescribeTags"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

步驟 2：使用自動關聯指令碼建立執行個體

若要建立 EC2 執行個體，您可以將入門套件中包含的使用者資料指令碼複製到 EC2 執行個體指示、Amazon EC2 Auto Scaling 群組啟動組態或 AWS CloudFormation 範本的 `userdata`區段。該指令碼僅支援執行 Ubuntu 和 Amazon Linux 作業系統的 EC2 執行個體。如需將指令碼新增至使用者資料的詳細資訊，請參閱 Amazon EC2 文件中的啟動時在 Linux 執行個體上執行命令。建立新節點的最簡單方法是使用 Amazon EC2 執行個體啟動精靈。本演練使用 中所述的 Apache Web 伺服器範例模組設定OpsWorks for Puppet Enterprise 入門。

1. 入門套件中的使用者資料指令碼會執行 `opsworks-cm` API `associate-node` 命令，將新節點與您的 Puppet 主節點建立關聯。在此版本中，它也會為您在節點 AWS CLI 上安裝目前版本的 ，以防它尚未執行up-to-date版本。將此指令碼以 `userdata.sh` 的名稱儲存到方便的位置。

   根據預設，新註冊節點的名稱為執行個體 ID。
2. 遵循 EC2 文件中啟動執行個體內的程序，並進行此處修改。在 EC2 執行個體啟動精靈中，選擇 Amazon Linux AMI。
3. 在 Configure Instance Details (設定執行個體詳細資訊) 頁面上，選取 myPuppetinstanceprofile，即您於步驟 1：建立 IAM 角色以用作您的執行個體描述檔所建立的角色，做為您的 IAM 角色。
4. 在 Advanced Details (進階詳細資訊) 區域內，上傳您在步驟 1 建立的 `userdata.sh` 指令碼。
5. 您無須在 Add Storage (新增儲存體) 頁面上進行任何變更。前往 Add Tags (新增標籤)。

   透過將標籤套用至您的 EC2 執行個體，您可以自訂 `userdata.sh` 的行為。在此範例中，透過新增下列標籤將角色 `apache_webserver` 套用到您的節點：**`pp_role`**，其值為 **`apache_webserver`**。

設定節點上 `pp_role` 的值會設定持續存放在節點代理程式憑證中的資料值，啟用節點的受信任分類。如需詳細資訊，請參閱 Puppet 平台文件中的 Extension requests (permanent certificate data)。

6. 在設定安全群組頁面上，選擇新增規則，然後選擇在此範例中為 Apache Web 伺服器開啟連接埠 8080 的 HTTP 類型。
7. 選擇 Review and Launch (檢閱及啟動)，然後選擇 Launch (啟動)。當您的新節點啟動時，它會套用您在 中設定之範例模組的 Apache 組態設定入門套件 Apache 範例。
8. 當您開啟連結至新節點公有 DNS 的網頁時，您應該會看到由 Puppet 受管 Apache Web 伺服器託管的網站。

取消節點與 Puppet Enterprise Server OpsWorks 的關聯

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節說明如何取消受管節點與 OpsWorks for Puppet Enterprise 伺服器的管理的關聯或移除受管節點。此操作會在命令列或 Puppet Enterprise 主控台中執行；您無法在 OpsWorks for Puppet Enterprise 管理主控台中取消節點的關聯。目前，OpsWorks for Puppet Enterprise API 不允許批次移除多個節點。本節中的命令會一次取消一個節點的關聯。

我們建議您在刪除伺服器前先從 Puppet 主伺服器取消與節點的關聯，使得節點可在不嘗試重新連線到伺服器的情況下繼續運作。若要執行此操作，請執行 `disassociate-node` AWS CLI 命令。若要從 PE 完全移除節點，您必須取消與節點的關聯，然後撤銷其憑證，使節點不再繼續嘗試與 Puppet 主伺服器確認。當您不再想要使用 Puppet 主伺服器管理他們時，您也應從節點解除安裝 `puppet-agent`。

取消與節點的關聯

1. 在 中 AWS CLI，執行下列命令來取消節點的關聯。*`Node_name`* 是您要取消關聯的節點名稱；對於 Amazon EC2 執行個體，這是執行個體 ID。*`Server_name`* 是您要取消節點關聯的 Puppet 主伺服器名稱。兩個都是必要參數。`--region` 參數並非必要項目，除非您希望從不是位於您預設區域內的 Puppet 主伺服器取消與節點的關聯。

```
aws opsworks-cm --region Region_name disassociate-node --node-name Node_name --
server-name Server_name
```

下列是範例命令。

```
aws opsworks-cm --region us-west-2 disassociate-node --node-name
 i-0010zzz00d66zzz90 --server-name opsworkstest
```

2. 等待回應訊息指出取消關聯已完成。

如需如何刪除 OpsWorks for Puppet Enterprise 伺服器的詳細資訊，請參閱刪除 Puppet Enterprise Server 的 OpsWorks 。

另請參閱

- Puppet Enterprise 文件中的 Remove nodes

刪除 Puppet Enterprise Server 的 OpsWorks

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節說明如何刪除 OpsWorks for Puppet Enterprise 伺服器。刪除伺服器也會刪除其事件、日誌和任何存放在伺服器上的模組。支援的資源 (Amazon Elastic Compute Cloud 執行個體、Amazon Elastic Block Store 磁碟區等） 也會一併刪除，以及所有自動備份。

雖然刪除伺服器不會刪除節點，但節點將不再由刪除的伺服器管理，並會持續嘗試重新連線。因此，我們建議您在刪除 Puppet 主伺服器前取消與受管節點的關聯。在此版本中，您可以透過執行 AWS CLI 命令來取消節點的關聯。

步驟 1：取消與受管節點的關聯

在刪除伺服器前先從 Puppet 主伺服器取消與節點的關聯，使節點可在不嘗試重新連線到伺服器的情況下繼續運作。若要執行此操作，請執行 `disassociate-node` AWS CLI 命令。

取消與節點的關聯

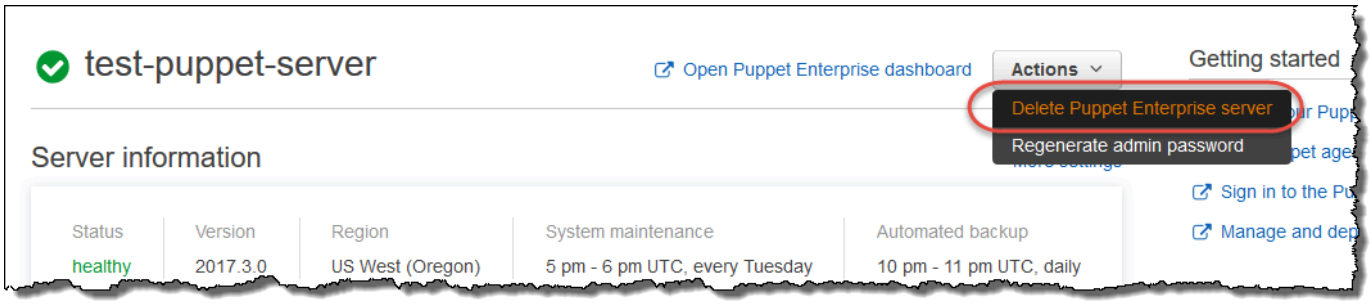
1. 在 中 AWS CLI，執行下列命令來取消節點的關聯。*`Server_name`* 為您要將節點取消關聯的來源 Puppet 主伺服器名稱。`--node-name` 的值可為執行個體 ID。

   ```
   aws opsworks-cm --region Region_name disassociate-node --node-name Node_name --
   server-name Server_name
   ```

2. 等待回應訊息指出取消關聯已完成。

步驟 2：刪除伺服器

1. 在儀表板上的伺服器磚上，展開 Actions (動作) 選單。



2. 選擇 Delete Puppet Enterprise server (刪除 Puppet Enterprise 伺服器)。
3. 當系統提示您確認刪除時，填滿核取方塊以刪除關聯角色和資源，然後選擇 Yes, Delete (是，刪除)。

另請參閱

- 取消節點與 Puppet Enterprise Server OpsWorks 的關聯

如何將 OpsWorks for Puppet Enterprise 伺服器遷移至 Amazon Elastic Compute Cloud (Amazon EC2)

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

下列指示說明如何將現有的 Puppet Enterprise 伺服器遷移至 Amazon EC2，以防您想要繼續使用 Puppet Enterprise 來滿足 OpsWorks 以外的組態管理需求。

主題

- 步驟 1：聯絡 Puppet 購買授權
- 步驟 2：取得 OpsWorks for Puppet Enterprise 伺服器的詳細資訊
- 步驟 3：備份您的 OpsWorks for Puppet Enterprise 伺服器
- 步驟 4：啟動新的 EC2 執行個體
- 步驟 5：在新的 EC2 執行個體上安裝 Puppet Enterprise
- 步驟 6：在新的 EC2 執行個體上還原備份
- 步驟 7：設定您的 Puppet 授權
- 步驟 8：遷移節點
- 步驟 9：刪除 Puppet Enterprise 伺服器的 OpsWorks

步驟 1：聯絡 Puppet 購買授權

當您將伺服器遷移至 EC2 時，新執行個體不會隨附 Puppet 授權。若要購買授權金鑰，請遵循 Puppet 網站的指示。

步驟 2：取得 OpsWorks for Puppet Enterprise 伺服器的詳細資訊

尋找並儲存 OpsWorks for Puppet Enterprise 伺服器的值。

1. 登入 AWS Management Console ，並在 https://console.aws.amazon.com/s3/：// 開啟 Amazon S3 主控台。

複製 OpsWorks for Puppet Enterprise 伺服器現有 Amazon S3 儲存貯體的名稱。儲存貯體名稱的格式如下： `aws-opsworks-cm-`*`server-name`*`-`*`random-string`*

2. 執行 aws opsworks-cm describe-servers命令以取得 OpsWorks for Puppet Enterprise 伺服器的組態。

```
aws opsworks-cm describe-servers \
   --server-name server-name \
   --region region
```

儲存Endpoint回應中 InstanceType、KeyPair、InstanceProfileArn、、 SubnetIds SecurityGroupIds和 的值。

3. 使用 SSH 連線到現有的 OpsWorks for Puppet Enterprise 伺服器。您可以在 EC2 主控台中使用 Session Manager，而不是 SSH。

執行下列命令。

```
rpm -qa | grep opsworks-cm-puppet-enterprise | cut -d '-' -f 5
```

回應提供 Puppet Enterprise 版本（例如 2019.8.10)。儲存此值。

您將使用 SSH 或 Session Manager 進行下一個步驟。

步驟 3：備份您的 OpsWorks for Puppet Enterprise 伺服器

1. 執行下列命令以建立本機備份。

```
mkdir /tmp/puppet-backup/
sudo /opt/puppetlabs/bin/puppet-backup create --dir=/tmp/puppet-backup/
```

2. 執行下列命令來存放備份的名稱。

```
ls /tmp/puppet-backup/
PUPPET_BACKUP=$(ls /tmp/puppet-backup/)
```

3. 執行下列命令，將備份上傳至 S3 儲存貯體。將 *S3-Bucket*體取代為 中步驟 1 的值步驟 2：取得 OpsWorks for Puppet Enterprise 伺服器的詳細資訊。

```
aws s3 cp /tmp/puppet-backup/PUPPET_BACKUP s3://S3_Bucket/tmp/puppet-backup/
```

存放 PUPPET_BACKUP和 S3_BUCKET值。您會將這些值匯入新的 EC2 執行個體。

您可以結束 SSH 或 Session Manager 工作階段。

步驟 4：啟動新的 EC2 執行個體

使用與 OpsWorks for Puppet Enterprise 伺服器相同的組態，從位於 https://console.aws.amazon.com/ec2/ 的 EC2 主控台啟動新的 EC2 EC2執行個體。

參數名稱	Value
作業系統	Amazon Linux 2
執行個體類型	來自 的步驟 2 `InstanceType` 的值步驟 2：取得 OpsWorks for Puppet Enterprise 伺服器的詳細資訊。
金鑰對名稱	來自 的步驟 2 `KeyPair`的值步驟 2：取得 OpsWorks for Puppet Enterprise 伺服器的詳細資訊。
VPC	`SubnetIds` 來自步驟 2 的 VPC步驟 2：取得 OpsWorks for Puppet Enterprise 伺服器的詳細資訊。
子網路	`SubnetIds` 來自 的步驟 2步驟 2：取得 OpsWorks for Puppet Enterprise 伺服器的詳細資訊。
選取現有的安全群組 -> 一般安全群組	`SecurityGroupIds` 來自 的步驟 2步驟 2：取得 OpsWorks for Puppet Enterprise 伺服器的詳細資訊。
儲存	至少 120 GB。
IAM 執行個體設定檔	`InstanceProfileArn` 來自 的步驟 2步驟 2：取得 OpsWorks for Puppet Enterprise 伺服器的詳細資訊。

如果您想要建立彈性 IP 並將其連接至新執行個體，請複製新執行個體的執行個體 ID，並完成 中的步驟（選用）步驟 4.1：建立和連接彈性 IP。

（選用）步驟 4.1：建立和連接彈性 IP

透過彈性 IP 位址，您可以快速地將地址重新映射至帳戶中的另一個執行個體，以遮罩執行個體或軟體的故障。

建立和關聯彈性 IP 地址

1. 登入 AWS Management Console 並開啟 Amazon EC2 主控台，網址為 https：//https://console.aws.amazon.com/ec2/。
2. 選擇彈性 IPs。
3. 選擇 Allocate Elastic IP address (配置彈性 IP 位址)。
4. 在配置彈性 IP 地址頁面中，選擇配置。這會建立公有 IPv4 地址。
5. 複製配置的 IPv4 地址。
6. 在動作中，選擇關聯彈性 IP 地址。
7. 針對執行個體，輸入新執行個體的執行個體 ID。
8. 選擇關聯。

步驟 5：在新的 EC2 執行個體上安裝 Puppet Enterprise

使用 SSH 連線到新的 EC2 執行個體。您可以在 EC2 主控台中使用 Session Manager，而不是 SSH。

```
# switch to sudo user
sudo -i

# Setup environment variables
PUPPET_ENTERPRISE_VERSION=Puppet Enterprise version from step 2.3
hostname Public IPv4 DNS or Custom Domain if available

# Install Puppet Enterprise
curl -JLO https://pm.puppetlabs.com/puppet-enterprise/$PUPPET_ENTERPRISE_VERSION/
puppet-enterprise-$PUPPET_ENTERPRISE_VERSION-el-7-x86_64.tar.gz
tar -xf puppet-enterprise-$PUPPET_ENTERPRISE_VERSION-el-7-x86_64.tar.gz

./puppet-enterprise-$PUPPET_ENTERPRISE_VERSION-el-7-x86_64/puppet-enterprise-installer
```

您可以在下一個步驟中保持 SSH 或 Session Manager 工作階段開啟。

步驟 6：在新的 EC2 執行個體上還原備份

```
# Setup environment variables
S3_BUCKET=S3 bucket name from step 2.1
PUPPET_BACKUP=Puppet backup file name from step 3.2

# download backup
aws s3 cp s3://$S3_BUCKET/tmp/puppet-backup/$PUPPET_BACKUP

# Prepare Puppet Enterprise backup to remove OpsWorks metadata
mkdir output
tar -xf $PUPPET_BACKUP -C output/
cd output/
rm -f opt/puppetlabs/facter/facts.d/opsworks.json
tar -cf ../$PUPPET_BACKUP *
cd ..
rm -rf output/

# Restore from backup
PATH=$PATH:/opt/puppetlabs/puppet/bin/
puppet-backup restore $PUPPET_BACKUP
puppet agent -t
```

您可以在執行個體的 https：//Public IPv4 存取還原 EC2 執行個體的 Puppet 主控台。 *IPv4* 您可以在 EC2 主控台的執行個體詳細資訊頁面上找到公有 IPv4 DNS。登入登入資料與您用來存取 OpsWorks for Puppet Enterprise 伺服器的登入資料相同。

您可以在下一個步驟中保持 SSH 或 Session Manager 工作階段開啟。

步驟 7：設定您的 Puppet 授權

請遵循 Puppet 網站上的步驟來設定您的授權。

您可以在下一個步驟中保持 SSH 或 Session Manager 工作階段開啟。

步驟 8：遷移節點

OpsWorks for Puppet Enterprise 伺服器支援兩種類型的網域：

- BYODC （自帶網域和憑證）
- OpsWorks 端點

步驟 8.1：針對 BYODC（自帶網域和憑證）

對於這些節點，您只需要將 DNS 供應商中的自訂網域指向新 EC2 執行個體的公有 IPv4 DNS 或公有 IPv4 地址。

步驟 8.2：適用於 OpsWorks 端點

對於 OpsWorks 端點，Puppet 文件建議解除安裝節點上的 Puppet 代理程式，然後使用新還原的 Puppet Enterprise 伺服器安裝 Puppet 代理程式。

> **Note**
>
> 雖然 Puppet 沒有自動移動代理程式節點的程序，但 Puppet 社群成員在 Puppet Forge 網站上發佈了幾個模組，以完成自動節點遷移。這些模組包含`pe_migrate`模組，以及由不同作者執行的第二個遷移模組。Puppet Forge 網站上的模組不受 Puppet 或 OpsWorks 支援，除非在 Forge 模組中明確註明。建議您謹慎使用這些模組，並在廣泛使用之前進行測試。

下列各節提供在 Linux 執行個體上解除安裝和重新安裝 Puppet 代理程式的步驟。

主題

- 步驟 8.2.1：從 Puppet 伺服器複製解除安裝程式
- 步驟 8.2.2：下載解除安裝程式並在節點上執行
- 步驟 8.2.3：在節點上重新安裝 Puppet 代理程式

步驟 8.2.1：從 Puppet 伺服器複製解除安裝程式

解除安裝代理程式之前，請確定節點的 IAM 執行個體描述檔提供 S3 ReadOnly 許可。

執行下列命令，將解除安裝程式從 Puppet 伺服器複製到 S3 儲存貯體。

```
aws s3 cp \
   /opt/puppetlabs/bin/puppet-enterprise-uninstaller \
   s3://$S3_BUCKET/tmp/puppet-enterprise-uninstaller
```

執行 命令後，您可以登出 Puppet 伺服器的 SSH 或 Session Manager 工作階段。

步驟 8.2.2：下載解除安裝程式並在節點上執行

使用 SSH 連線到節點。如果節點是 EC2 執行個體，您可以在 EC2 主控台中使用 Session Manager，而不是 SSH。

```
sudo -i

S3_BUCKET=aws-opsworks-cm-abcdefg-uuhtyn6messn
aws s3 cp s3://$S3_BUCKET/tmp/puppet-enterprise-uninstaller /opt/puppetlabs/bin/
chmod 700 /opt/puppetlabs/bin/puppet-enterprise-uninstaller
/opt/puppetlabs/bin/puppet-enterprise-uninstaller
```

您可以在下一個步驟中保持 SSH 或 Session Manager 工作階段開啟。

步驟 8.2.3：在節點上重新安裝 Puppet 代理程式

請完成下列步驟，以在節點上重新安裝 Puppet 代理程式。

主題

- 步驟 8.2.3.1：使用正確的組態安裝 Puppet 代理程式
- 步驟 8.2.3.2：接受 Puppet 主控台中的憑證
- 步驟 8.2.3.3：檢查 Puppet Enterprise 伺服器中的節點

步驟 8.2.3.1：使用正確的組態安裝 Puppet 代理程式

執行下列命令來安裝 Puppet 代理程式。

```
curl -k https://Public_IPv4_DNS:8140/packages/current/install.bash | bash
```

您可以在步驟 8.2.2.3 中保持 SSH 或 Session Manager 工作階段開啟。

步驟 8.2.3.2：接受 Puppet 主控台中的憑證

1. 前往 的 Puppet 伺服器主控台`https://Public_IPv4_DNS`。
2. 選擇憑證，然後選擇未簽署的憑證。
3. 選擇接受以簽署 Puppet 代理程式的憑證。

步驟 8.2.3.3：檢查 Puppet Enterprise 伺服器中的節點

在節點上執行下列命令，以檢查它是否進入伺服器。

```
puppet agent -t
```

節點現在應該會顯示在 Puppet 伺服器主控台中。

步驟 9：刪除 Puppet Enterprise 伺服器的 OpsWorks

您可以使用 OpsWorks 主控台或刪除 AWS CLI OpsWorks for Puppet Enterprise 伺服器。

使用 OpsWorks 主控台刪除您的伺服器

1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/opsworks/ 開啟 AWS OpsWorks 主控台。
2. 從導覽窗格中選擇 Puppet Enterprise 伺服器。
3. 在 Puppet Enterprise 伺服器頁面上，選擇您要刪除的伺服器。
4. 在動作中，選擇刪除 Puppet Enterprise 伺服器。

使用 刪除您的伺服器 AWS CLI

執行下列命令。

```
aws opsworks-cm delete-server \
   --server-name server-name \
   --region region
```

使用 記錄 Puppet Enterprise API 呼叫的 OpsWorks AWS CloudTrail

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

OpsWorks for Puppet Enterprise 已與 整合 AWS CloudTrail，此服務提供使用者、角色或 OpsWorks for Puppet Enterprise AWS 服務所採取動作的記錄。CloudTrail 會將 OpsWorks for Puppet Enterprise 的所有 API 呼叫擷取為事件，包括 OpsWorks for Puppet Enterprise 主控台的呼叫，以及 OpsWorks

for Puppet Enterprise APIs程式碼呼叫。如果您建立線索，您可以啟用 CloudTrail 事件持續交付至 Amazon S3 儲存貯體，包括 OpsWorks for Puppet Enterprise 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 收集的資訊，判斷對 OpsWorks for Puppet Enterprise 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」。

CloudTrail 中 Puppet Enterprise 資訊的 OpsWorks

建立 AWS 帳戶時，會在您的帳戶上啟用 CloudTrail。當活動在 OpsWorks for Puppet Enterprise 中發生時，該活動會記錄於 CloudTrail 事件，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 OpsWorks for Puppet Enterprise 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，當您在主控台建立權杖時，權杖會套用到所有區域。追蹤會記錄 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 CloudTrail 日誌中收集的事件資料並對其採取行動。如需詳細資訊，請參閱：

- 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，以及從多個帳戶接收 CloudTrail 日誌檔案

CloudTrail 會記錄所有 OpsWorks for Puppet Enterprise 動作，並記錄在 OpsWorks for Puppet Enterprise API 參考中。例如，呼叫 CreateServer、CreateBackup 和 DescribeServers 動作會在 CloudTrail 日誌檔案中產生項目。

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：

- 該請求是否使用根或 IAM 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 Puppet Enterprise Log File 項目的 OpsWorks

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 日誌檔案並非依公有 API 呼叫追蹤記錄的堆疊排序，因此不會以任何特定順序出現。

下列範例顯示 OpsWorks for Puppet Enterprise `CreateServer`動作的 CloudTrail 日誌項目。

```
{"eventVersion":"1.05",
"userIdentity":{
    "type":"AssumedRole",
    "principalId":"ID number:OpsWorksCMUser",
    "arn":"arn:aws:sts::831000000000:assumed-role/Admin/OpsWorksCMUser",
    "accountId":"831000000000","accessKeyId":"ID number",
    "sessionContext":{
        "attributes":{
            "mfaAuthenticated":"false",
            "creationDate":"2017-01-05T22:03:47Z"
            },
        "sessionIssuer":{
            "type":"Role",
            "principalId":"ID number",
            "arn":"arn:aws:iam::831000000000:role/Admin",
            "accountId":"831000000000",
            "userName":"Admin"
            }
        }
    },
"eventTime":"2017-01-05T22:18:23Z",
"eventSource":"opsworks-cm.amazonaws.com",
"eventName":"CreateServer",
"awsRegion":"us-west-2",
"sourceIPAddress":"101.25.190.51",
"userAgent":"console.amazonaws.com",
"requestParameters":{
    "serverName":"test-puppet-server",
    "engineModel":"Single",
    "engine":"Puppet",
    "instanceProfileArn":"arn:aws:iam::831000000000:instance-profile/aws-opsworks-cm-
ec2-role",
    "backupRetentionCount":3,"serviceRoleArn":"arn:aws:iam::831000000000:role/service-
role/aws-opsworks-cm-service-role",
```

```
    "engineVersion":"12",
    "preferredMaintenanceWindow":"Fri:21:00",
    "instanceType":"t2.medium",
    "subnetIds":["subnet-1e111f11"],
    "preferredBackupWindow":"Wed:08:00"
    },
"responseElements":{
    "server":{
        "endpoint":"test-puppet-server-xxxx8u4390xo6pd9.us-west-2.opsworks-cm.io",
        "createdAt":"Jan 5, 2017 10:18:22 PM",
        "serviceRoleArn":"arn:aws:iam::831000000000:role/service-role/aws-opsworks-cm-
service-role",
        "preferredBackupWindow":"Wed:08:00",
        "status":"CREATING",
        "subnetIds":["subnet-1e111f11"],
        "engine":"Puppet",
        "instanceType":"t2.medium",
        "serverName":"test-puppet-server",
        "serverArn":"arn:aws:opsworks-cm:us-west-2:831000000000:server/test-puppet-
server/8ezz7f6z-e91f-4z10-89z5-8c6219zzz09f",
        "engineModel":"Single",
        "backupRetentionCount":3,
        "engineAttributes":[
            {"name":"PUPPET_ADMIN_PASSWORD","value":"*** Redacted ***"},
            {"name":"PUPPET_API_CA_CERT","value":"*** Redacted ***"},
            ],
        "engineVersion":"12.11.1",
        "instanceProfileArn":"arn:aws:iam::831000000000:instance-profile/aws-opsworks-
cm-ec2-role",
        "preferredMaintenanceWindow":"Fri:21:00"
        }
    },
"requestID":"de7z64z9-d394-12ug-8081-7zz0386fbcb6",
"eventID":"8z7z18dz-6z90-47bz-87cf-e8346428zzz3",
"eventType":"AwsApiCall",
"recipientAccountId":"831000000000"
}
```

故障診斷 Puppet Enterprise 的 OpsWorks

> **Important**
>
> AWS OpsWorks for Puppet Enterprise 此服務已於 2024 年 3 月 31 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本主題包含一些常見的 OpsWorks for Puppet Enterprise 問題，以及針對這些問題的建議解決方案。

主題

- 一般疑難排解秘訣
- 針對特定錯誤進行故障診斷
- 其他協助及支援

一般疑難排解秘訣

若您無法建立或使用 Puppet 主伺服器，您可以檢視錯誤訊息或日誌，來協助您故障診斷問題。以下任務說明在您故障診斷 Puppet 主伺服器問題時一般開始著手的位置。如需特定錯誤和解決方案的資訊，請參閱本主題的針對特定錯誤進行故障診斷一節。

- 如果 Puppet 主控端無法啟動，請使用 OpsWorks for Puppet Enterprise 主控台檢視錯誤訊息。在 Puppet 主伺服器屬性頁面上，與啟動和執行伺服器相關的錯誤訊息會顯示在頁面的頂端。錯誤可能來自 OpsWorks for Puppet Enterprise AWS CloudFormation或 Amazon EC2，這些服務用於建立 Puppet 主伺服器。在屬性頁面上，您也可以檢視在執行中伺服器上發生的事件，其中也可能包含故障事件訊息。
- 為協助解決 EC2 問題，請使用 SSH 連線到您伺服器的執行個體並檢視日誌。EC2 執行個體日誌存放在 `/var/log/aws/opsworks-cm` 目錄。這些日誌會在 OpsWorks for Puppet Enterprise 啟動 Puppet 主控端時擷取命令輸出。

針對特定錯誤進行故障診斷

主題

- 伺服器處於連線中斷狀態

- 伺服器建立失敗，並顯示 "requested configuration is currently not supported" 訊息
- 無法建立伺服器的 Amazon EC2 執行個體
- 服務角色錯誤導致伺服器無法建立
- 超過彈性 IP 地址限制
- 自動節點關聯失敗
- 系統維護失敗

伺服器處於連線中斷狀態

問題：伺服器的狀態顯示為連線中斷。

原因：當 外部的實體對 OpsWorks for Puppet Enterprise 伺服器或其支援資源 AWS OpsWorks 進行變更時，這最常發生。 AWS OpsWorks 無法連線到連線中斷狀態的 Puppet Enterprise 伺服器，以處理維護任務，例如建立備份、套用作業系統修補程式或更新 Puppet。因此，您的伺服器可能缺少重要的更新、容易發生安全問題，或者無法如預期般運作。

解決方案：嘗試下列步驟來還原伺服器的連線。

1. 請確定您的服務角色具有所有必要的許可。
    a. 在伺服器的設定頁面上，在網路和安全性中，選擇伺服器正在使用之服務角色的連結。這會開啟要在 IAM 主控台中檢視的服務角色。
    b. 在許可索引標籤上，確認 `AWSOpsWorksCMServiceRole` 位於許可政策清單中。如果未列出，請手動將 `AWSOpsWorksCMServiceRole` 受管政策新增至角色。
    c. 在信任關係索引標籤上，確認服務角色具有信任政策，信任`opsworks-cm.amazonaws.com`服務代表您擔任角色。如需如何搭配角色使用信任政策的詳細資訊，請參閱修改角色（主控台）或 AWS 安全部落格文章，如何搭配 IAM 角色使用信任政策。
2. 請確定您的執行個體設定檔具有所有必要的許可。
    a. 在伺服器的設定頁面上，在網路和安全性中，選擇伺服器正在使用的執行個體描述檔連結。這會開啟執行個體設定檔，以便在 IAM 主控台中檢視。
    b. 在許可索引標籤上，確認 `AmazonEC2RoleforSSM`和 `AWSOpsWorksCMInstanceProfileRole` 都位於許可政策清單中。如果其中之一或兩者未列出，請手動將這些受管政策新增至角色。

c. 在信任關係索引標籤上，確認服務角色具有信任政策，信任`ec2.amazonaws.com`服務代表您擔任角色。如需如何將信任政策與角色搭配使用的詳細資訊，請參閱修改角色（主控台）或 AWS 安全部落格文章：如何將信任政策與 IAM 角色搭配使用。

3. 在 Amazon EC2 主控台中，請確定您位於與 OpsWorks for Puppet Enterprise 伺服器相同的區域，然後重新啟動伺服器正在使用的 EC2 執行個體。

   a. 選擇名為 `aws-opsworks-cm-instance-`*`server-name`* 的 EC2 執行個體。

   b. 在執行個體狀態功能表中，選擇重新啟動執行個體。

   c. 您的伺服器最多需要 15 分鐘才能重新啟動並完全上線。

4. 在 OpsWorks for Puppet Enterprise 主控台的伺服器詳細資訊頁面上，確認伺服器狀態現在運作狀態良好。

如果執行上述步驟後，伺服器狀態仍然遺失連線，請嘗試下列其中一項。

- 透過建立新的伺服器並刪除原始伺服器來取代伺服器。如果目前伺服器上的資料對您很重要，請從最近的備份還原伺服器，並確認資料是最新的，然後再刪除原始、無回應的伺服器。
- 聯絡 AWS 支援。

伺服器建立失敗，並顯示 "requested configuration is currently not supported" 訊息

問題：您嘗試建立 Puppet Enterprise 伺服器，但伺服器建立失敗並顯示與下列內容類似的錯誤訊息："The requested configuration is currently not supported. Please check the documentation for supported configurations."

原因：Puppet 主伺服器可能指定了不支援的執行個體類型。若您選擇在擁有非預設租用的 VPC (例如一個專用執行個體) 中建立 Puppet 伺服器，所有指定 VPC 中的執行個體也都必須是專用或主控租用。因為有些執行個體類型 (例如 t2) 僅能使用預設租用，指定 VPC 便可能無法支援 Puppet 主伺服器執行個體類型，因此導致伺服器建立失敗。

解決方案：若您選擇具有非預設租用的 VPC，請使用支援專用租用的 m4 執行個體類型。

無法建立伺服器的 Amazon EC2 執行個體

問題：伺服器建立失敗，並顯示與下列內容相似的錯誤訊息："The following resource(s) failed to create: [EC2Instance]. Failed to receive 1 resource signal(s) within the specified duration."

原因：這最有可能是因為 EC2 執行個體沒有網路存取。

解決方案：確保執行個體具有傳出網際網路存取，且 AWS 服務代理程式能夠發出命令。確認您的 VPC (使用單一公有子網路的 VPC) 已啟用 DNS resolution (DNS 解析)，並且您的子網路也已啟用 Auto-assign Public IP (自動指派公有 IP) 設定。

服務角色錯誤導致伺服器無法建立

問題：伺服器建立失敗，並顯示下列錯誤訊息："Not authorized to perform sts:AssumeRole."

原因：這可能會在您使用的服務角色缺少適當的許可，無法建立新伺服器時發生。

解決方案：開啟 OpsWorks for Puppet Enterprise 主控台；使用主控台產生新的服務角色和執行個體描述檔角色。若您偏好使用您自己的服務角色，請將 AWSOpsWorksCMServiceRole 政策連接到角色。確認 opsworks-cm.amazonaws.com 已列在角色信任關係中的服務之間。確認與 Puppet 主伺服器相關聯的服務角色已連接 AWSOpsWorksCMServiceRole 受管政策。

超過彈性 IP 地址限制

問題：伺服器建立失敗，並顯示下列錯誤訊息："The following resource(s) failed to create: [EIP, EC2Instance]. Resource creation cancelled, the maximum number of addresses has been reached."

原因：當您的帳戶使用的彈性 IP (EIP) 地址數超過最大值時，便可能發生此情況。預設 EIP 地址限制為 5 個。

解決方案：您可以釋出現有的 EIP 地址或刪除您的帳戶未主動使用的 EIP 地址，也可以聯絡 AWS 客戶支援，以增加與您的帳戶相關聯的 EIP 地址限制。

自動節點關聯失敗

問題：新 Amazon EC2 節點的全自動或自動關聯失敗。應新增至 Puppet 主伺服器的節點並未在 Puppet Enterprise 儀表板上出現。

原因：這可能會在您沒有將 IAM 角色設為允許 `opsworks-cm` API 呼叫，以和新的 EC2 執行個體通訊的執行個體描述檔時發生。

解決方案：將政策連接到您的 EC2 執行個體描述檔，以允許 `AssociateNode` 和 `DescribeNodeAssociationStatus` API 呼叫使用 EC2，如[在適用於 Puppet Enterprise 的 OpsWorks 中自動新增節點](#)中所述。

系統維護失敗

AWS OpsWorks CM 會執行每週系統維護，以確保 AWS Puppet 伺服器的最新版本，包括安全性更新，一律在 OpsWorks for Puppet Enterprise 伺服器上執行。如果系統維護因任何原因失敗， 會 AWS

OpsWorks CM 通知您失敗。如需系統維護的詳細資訊，請參閱OpsWorks for Puppet Enterprise 中的系統維護。

本節說明失敗的可能原因，並建議解決方案。

主題

- 服務角色或執行個體描述檔錯誤導致系統維護無法進行

服務角色或執行個體描述檔錯誤導致系統維護無法進行

問題：系統維護失敗，並出現錯誤訊息，指出「未獲授權執行 sts：AssumeRole」，或有關許可的類似錯誤訊息。

原因：當您正在使用的服務角色或執行個體描述檔缺少足夠的許可，在伺服器上執行系統維護時，可能會發生這種情況。

解決方案：確保您的服務角色和執行個體描述檔具有所有必要的許可。

1. 請確定您的服務角色具有所有必要的許可。

    a. 在伺服器的設定頁面上，於網路和安全性中，選擇伺服器正在使用之服務角色的連結。這會開啟要在 IAM 主控台中檢視的服務角色。

    b. 在許可索引標籤上，確認 AWSOpsWorksCMServiceRole 已連接至服務角色。如果 AWSOpsWorksCMServiceRole 未列出，請將此政策新增至角色。

    c. 確認 opsworks-cm.amazonaws.com 已列在角色信任關係中的服務之間。如需如何將信任政策與角色搭配使用的詳細資訊，請參閱修改角色（主控台）或 AWS 安全部落格文章：如何將信任政策與 IAM 角色搭配使用。

2. 請確定您的執行個體設定檔具有所有必要的許可。

    a. 在伺服器的設定頁面上，在網路和安全性中，選擇伺服器正在使用的執行個體描述檔連結。這會開啟執行個體設定檔，以便在 IAM 主控台中檢視。

    b. 在許可索引標籤上，確認 AmazonEC2RoleforSSM和 AWSOpsWorksCMInstanceProfileRole 都位於許可政策清單中。如果其中之一或兩者未列出，請手動將這些受管政策新增至角色。

    c. 在信任關係索引標籤上，確認服務角色具有信任政策，信任ec2.amazonaws.com服務代表您擔任角色。如需如何將信任政策與角色搭配使用的詳細資訊，請參閱修改角色（主控台）或 AWS 安全部落格文章，如何將信任政策與 IAM 角色搭配使用。

其他協助及支援

若您在本主題中沒有看到您的特定問題，或是您已嘗試本主題中的建議，但仍然發生問題，請造訪 AWS OpsWorks 論壇。

您也可以前往 AWS Support 中心。 AWS 支援中心是建立和管理 AWS 支援案例的中樞。 AWS 支援中心也包含其他實用資源的連結，例如論壇、技術FAQs、服務運作狀態和 AWS Trusted Advisor。

AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或替代解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks for Chef Automate 可讓您在 AWS 中執行 Chef Automate 伺服器。您可以在幾分鐘內佈建 Chef 伺服器，並讓 AWS OpsWorks for Chef Automate 處理其操作、備份、還原和軟體升級。AWS OpsWorks for Chef Automate 可讓您專注於核心組態管理任務，而不是管理 Chef 伺服器。

Chef Automate 伺服器透過指示 `chef-client` 要在節點上執行哪些 Chef 配方，以在您的環境中管理節點組態、存放節點的資訊，並成為您 Chef 技術指南的中央儲存庫。 AWS OpsWorks for Chef Automate 提供的 Chef 伺服器中包含 Chef Automate 的頂級功能： Chef Infra 和 Chef InSpec。

AWS OpsWorks for Chef Automate 伺服器在 Amazon Elastic Compute Cloud instance. AWS OpsWorks for Chef Automate server 上執行，設定為執行最新版本的 Amazon Linux (Amazon Linux 2)。如需此版本 Chef Automate 的相關變更資訊，請參閱 Chef Automate 版本備註。下表說明安裝在 AWS OpsWorks for Chef Automate 伺服器上的 Chef 元件。

元件名稱	描述	安裝在 AWS OpsWorks for Chef Automate 伺服器上的版本
Chef Automate	Chef Automate 是企業伺服器軟體套件，可透過以 Web 為基礎的管理主控台，提供持續部署的自動化工作流程和受管節點的相關見解。Chef Automate 提供基礎設施自動化 (Chef Infrastructure)、安全和合規資訊及強制實施 (Chef InSpec)，以及自動化部署 (Chef Habitat)。	2.0

元件名稱	描述	安裝在 AWS OpsWorks for Chef Automate 伺服器上的版本
	如需 Chef Automate 的詳細資訊，請參閱 Chef 網站上的 Chef Automate。	
Chef Infra	舊名為 Chef Server 的 Chef Infra 伺服器，使用 Chef Infra 用戶端 (`chef-client`) 代理程式，對受管節點持續套用組態，維護其所需的狀態。 如需 Infra 的詳細資訊，請參閱Chef 網站上的 Chef Infra。	12.x
Chef InSpec	Chef InSpec 描述可在軟體工程師、營運和安全工程師之間共用的安全和合規規則。合規、安全性和其他政策要求，共同形成了 `chef-client` 代理程式可以針對受管節點執行之自動化測試的框架，因此能確保強制執行標準維持一致。 如需 InSpec 的詳細資訊，請參閱 Chef 網站上的 Chef InSpec。	3.9.0

與 AWS OpsWorks for Chef Automate 伺服器相關聯節點上的 `chef-client` 最低支援版本為 13.x。我們建議執行至少 14.10.9，或最新、穩定的`chef-client`版本。

當有新的 Chef 軟體次要版本可用時，系統維護的設計會在其通過 AWS 測試之後，自動更新伺服器上 Chef Automate 和 Chef Server 的次要版本。AWS 會執行廣泛的測試，驗證 Chef 升級已生產就緒，而不會中斷現有的客戶環境，因此在 Chef 軟體發行後和可供現有 OpsWorks for Chef Automate 伺服器套用之間，可能會有延遲。系統維護也會將您的伺服器升級至最新版的 Amazon Linux。

您可以連接任何執行受支援作業系統且具有 AWS OpsWorks for Chef Automate 伺服器網路存取權的現場部署電腦或 EC2 執行個體。如需您要管理之節點的支援作業系統清單，請參閱 Chef 網站。在您要使用 Chef 伺服器管理的節點上安裝 `chef-client` 代理程式軟體。

主題

- 的區域支援 AWS OpsWorks for Chef Automate
- AWS OpsWorks 適用於 Chef Automate 生命週期結束FAQs
- 將 AWS OpsWorks for Chef Automate 伺服器升級至 Chef Automate 2
- 入門 AWS OpsWorks for Chef Automate
- 使用 建立 AWS OpsWorks for Chef Automate 伺服器 AWS CloudFormation
- 更新 AWS OpsWorks for Chef Automate 伺服器以使用自訂網域
- 重新產生 AWS OpsWorks for Chef Automate 伺服器的入門套件
- 在 AWS OpsWorks for Chef Automate 資源上使用標籤
- 備份和還原 AWS OpsWorks for Chef Automate 伺服器
- 中的系統維護 AWS OpsWorks for Chef Automate
- 中的合規掃描 AWS OpsWorks for Chef Automate
- 取消節點與 AWS OpsWorks for Chef Automate 伺服器的關聯
- 刪除 AWS OpsWorks for Chef Automate 伺服器
- 重設 Chef Automate 儀表板登入資料
- 使用 記錄 AWS OpsWorks for Chef Automate API 呼叫 AWS CloudTrail
- 故障診斷 AWS OpsWorks for Chef Automate

的區域支援 AWS OpsWorks for Chef Automate

下列區域端點支援 AWS OpsWorks for Chef Automate server. AWS OpsWorks for Chef Automate created 與您的 Chef 伺服器相關聯的資源，例如執行個體描述檔、使用者和服務角色，位於與 Chef 伺服器相同的區域端點。您的 Chef 伺服器必須在 VPC 中。您可以使用您建立的 VPC、既有的 VPC 或預設的 VPC。

- 美國東部 (俄亥俄) 區域
- 美國東部 (維吉尼亞北部) 區域
- 美國西部 (加利佛尼亞北部) 區域
- 美國西部 (奧勒岡) 區域

- 亞太 (東京) 區域
- 亞太區域 (新加坡) 區域
- 亞太區域 (雪梨) 區域
- 歐洲 (法蘭克福) 區域
- 歐洲 (愛爾蘭) 區域

AWS OpsWorks 適用於 Chef Automate 生命週期結束FAQs

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或替代解決方案。

主題

- 現有使用者將如何受到此生命週期結束的影響？
- 如果我不採取任何動作，我的伺服器會發生什麼情況？
- 我可以轉換到哪些替代方案？
- 服務是否仍然接受新客戶？
- 生命週期結束會 AWS 區域 同時影響所有 嗎？
- 提供何種程度的技術支援？
- 我是 OpsWorks for Chef Automate 的目前客戶，我需要在先前未使用此服務的帳戶中啟動伺服器。我是否可以這麼做？
- 明年是否會有任何主要功能版本？

現有使用者將如何受到此生命週期結束的影響？

現有客戶在 2024 年 5 月 5 日之前，OpsWorks for Chef Automate 的生命週期結束日期不會受到影響。在生命週期終止日期之後，客戶將無法再使用 OpsWorks 主控台或 API 管理其伺服器。

如果我不採取任何動作，我的伺服器會發生什麼情況？

從 2024 年 5 月 5 日起，您將無法再使用 OpsWorks 主控台或 API 管理您的伺服器。屆時，我們將停止為您的伺服器執行任何持續管理功能，例如備份或維護。為了限制對客戶的影響，我們會保留正

在備份 Chef Automate 伺服器的任何執行中的 EC2 執行個體，但他們的授權將不再有效，因為根據 OpsWorks for Chef Automate 服務協議，用量不再涵蓋（或計費）。您需要聯絡 Chef 以取得新的授權。當您聯絡 Chef 時，請務必告訴他們您是 Chef Automate 的現有 OpsWorks 客戶，而且您正在從 OpsWorks 轉換。

我可以轉換到哪些替代方案？

AWS 和 Progress Chef 建議您遷移到其新的 Chef SaaS 產品，以便您可以繼續從全受管 Chef Automate 服務中受益。若要開始使用 Chef SaaS，您可以聯絡 Chef 取得有關如何設定 Chef SaaS 帳戶以及轉換資料和節點的文件。

如果 Chef SaaS 無法滿足您的需求，因為您偏好在您控制的 AWS 帳戶中的 EC2 執行個體上執行 Chef Automate，Chef 提供多種選項，包括AWS Marketplace 自攜授權 (BYOL) 模型和 EC2 上的自我託管。您可以聯絡 Progress Chef，以取得如何執行此類轉換的詳細資訊。

服務是否仍然接受新客戶？

否。 AWS OpsWorks Chef Automate 不再接受新客戶。

生命週期結束會 AWS 區域 同時影響所有 嗎？

是。API 和主控台將達到生命週期結束，從 2024 年 5 月 5 日起完全無法使用 AWS 區域。如需 Chef Automate AWS OpsWorks 可用 AWS 區域 位置的相關資訊，請參閱AWS 區域服務清單。

提供何種程度的技術支援？

AWS 將繼續提供客戶今天直到生命週期結束為止對 OpsWorks for Chef Automate 的相同層級支援。如果您有問題或疑慮，可以透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。如需轉換支援，我們建議客戶聯絡 Progress Chef。

我是 OpsWorks for Chef Automate 的目前客戶，我需要在先前未使用此服務的帳戶中啟動伺服器。我是否可以這麼做？

一般而言不會，除非有特殊情況需要這樣做。如果您有特殊情況，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊，並提供詳細資訊和理由，我們將審核您的請求。

明年是否會有任何主要功能版本？

否。由於服務即將到期，我們將不會發行任何新功能。不過，我們將繼續如預期進行安全性改善和管理伺服器，直到生命週期結束為止。

將 AWS OpsWorks for Chef Automate 伺服器升級至 Chef Automate 2

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或替代解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

升級至 Chef Automate 2 的先決條件

在開始之前，請務必了解 Chef Automate 2 新增的新功能以及 Chef Automate 2 不支援的功能。如需 Chef Automate 2 中新功能和不支援功能的相關資訊，請參閱 Chef 網站上的 Chef Automate 2 文件。

執行 Chef Automate 1 的伺服器必須在 2019 年 11 月 1 日之後至少有一次成功的維護執行，才符合升級資格。

如同 AWS OpsWorks for Chef Automate 伺服器上的任何維護操作，伺服器在升級期間會離線。在升級程序期間，您應該規劃最多三個小時的停機時間。

您需要此伺服器的登入資料以用於 Chef Automate 儀表板網站。升級完成後，您應該登入到 Chef Automate 儀表板，並確認您的節點和組態資訊沒有變更。

> **Important**
>
> 當您準備好將 AWS OpsWorks for Chef Automate 伺服器升級至 Chef Automate 2 時，請只使用此處的指示進行升級。由於 會 AWS OpsWorks for Chef Automate 自動化許多升級程序，例如備份建立，因此請勿遵循 Chef 網站上的升級指示。

關於升級程序

在升級程序期間，您的伺服器會在開始升級之前和完成升級之後進行備份。會建立下列備份：

- 仍在執行 Chef Automate 1 (版本 12.17.33) 之伺服器的備份。
- 升級完成後的伺服器備份，此伺服器執行 Chef Automate 2 (版本 2019-08)。

升級程序會在執行 Chef Automate 1 時終止伺服器正在使用的 Amazon EC2 執行個體。並建立一個新的執行個體來執行 Chef Automate 2 伺服器。

升級至 Chef Automate 2 (主控台)

1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/opsworks/ 開啟 AWS OpsWorks 主控台。
2. 在左側導覽窗格中，選擇 AWS OpsWorks for Chef Automate。
3. 選擇伺服器以檢視其屬性頁面。頁面頂端的藍色橫幅應該會指出伺服器是否符合升級至 Chef Automate 2 的資格。

   > **Note**
   >
   > 執行 Chef Automate 1 的伺服器必須在 2019 年 11 月 1 日之後至少有一次成功的維護執行，才符合升級資格。

4. 如果伺服器符合升級資格，請選擇 Start upgrade (開始升級)。
5. 升級最多需要三個小時。在升級程序期間，屬性頁面會將伺服器狀態顯示為 Under maintenance (維護中)。
6. 升級完成時，屬性頁面會顯示下列兩則訊息：Successfully upgraded to Automate 2 (成功升級至 Automate 2)、Maintenance completed successfully (維護成功完成)。伺服器狀態應為 HEALTHY (狀態良好)。
7. 使用您現有的登入資料登入 Chef Automate 儀表板，並確認您的節點是否正確回報。

升級至 Chef Automate 2 (CLI)

1. （選用）如果您不確定哪些 AWS OpsWorks for Chef Automate 伺服器符合升級資格，請執行下列命令。如果您想要列出與預設 AWS 區域不同的 AWS 區域中的 AWS OpsWorks for Chef Automate 伺服器，請務必新增 `--region` 參數。

   ```
   aws opsworks-cm describe-servers
   ```

   在結果中，尋找屬性 `CHEF_MAJOR_UPGRADE_AVAILABLE` 是否有 `true` 的值。這表示該伺服器符合升級至 Chef Automate 2 的資格。請記下符合升級資格的 AWS OpsWorks for Chef Automate 伺服器名稱。

2. 執行下列命令，以 AWS OpsWorks for Chef Automate 伺服器的名稱取代 *server_name*。若要升級至 Chef Automate 2 而不是執行例行系統維護，請加上 CHEF_MAJOR_UPGRADE 引擎屬性，如命令所示。如果目標伺服器不在您的預設 AWS 區域中，請加上--region 參數。每個命令只能升級一部伺服器。

```
aws opsworks-cm start-maintenance --server-name server_name --engine-attributes
 Name=CHEF_MAJOR_UPGRADE,Value=true --region region
```

如果 因任何原因 AWS OpsWorks for Chef Automate 無法升級伺服器，此命令會導致驗證例外狀況。

3. 升級最多需要三個小時。您可以執行下列命令來定期檢查升級狀態。

```
aws opsworks-cm describe-servers --server-name server_name
```

在結果中，尋找 Status 值。Status 為 UNDER_MAINTENANCE 表示升級仍在進行中。成功升級會傳回類似下列的訊息。

```
2019/10/24 00:27:56 UTC          Successfully upgraded to Automate 2.
2019/10/23 23:50:38 UTC          Upgrading Chef server from Automate 1 to Automate
 2
```

如果升級失敗， AWS OpsWorks for Chef Automate 會自動將伺服器復原至 Chef Automate 1。

如果升級成功，但伺服器無法如同升級之前那樣運作 (例如，假設受管節點無法回報)，您可手動復原伺服器。如需手動復原資訊，請參閱[將 AWS OpsWorks for Chef Automate 伺服器復原至 Chef Automate 1 (CLI)]。

將 AWS OpsWorks for Chef Automate 伺服器復原至 Chef Automate 1 (CLI)

如果升級程序失敗， AWS OpsWorks for Chef Automate 會自動將您的伺服器轉返 Chef Automate 1。如果升級成功，但伺服器的運作與升級之前不同，您可以使用 手動將 AWS OpsWorks for Chef Automate 伺服器轉返 Chef Automate 1 AWS CLI。

1. 執行下列命令，顯示在您嘗試升級之前，在伺服器上執行之最後一次備份的 BackupId。如果您的伺服器位於與預設 AWS 區域不同的 AWS 區域，請加上 --region 參數。

```
aws opsworks-cm describe-backups server_name
```

備份 ID 的格式為 *ServerName-yyyyMMddHHmmssSSS*。在結果中尋找以下 Chef Automate 1 屬性。

```
"Engine": "Chef"
"EngineVersion": "12.17.33"
```

2. 執行下列命令，使用您在步驟 1 中傳回的備份 ID 作為 `--backup-id` 值。

```
aws opsworks-cm restore-server --server-name server_name --backup-id ServerName-yyyyMMddHHmmssSSS
```

視您儲存在伺服器上的資料量而定，還原伺服器需要 20 分鐘到 3 小時的時間。在還原作業期間，您的伺服器的狀態為 `RESTORING`。此狀態會顯示在 中的伺服器屬性頁面上 AWS Management Console，並在describe-servers命令的結果中傳回。

3. 還原完成後，主控台會顯示 Restore completed successfully (還原成功完成) 訊息。您的 AWS OpsWorks for Chef Automate 伺服器已上線，且與您開始升級程序之前相同。

另請參閱

- 中的系統維護 AWS OpsWorks for Chef Automate
- 從備份還原 AWS OpsWorks for Chef Automate 伺服器
- AWS OpsWorks API 參考中的 DescribeServers
- AWS OpsWorks API 參考中的 StartMaintenance

入門 AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或其他解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks for Chef Automate 可讓您在 中執行 Chef Automate 伺服器 AWS。只要約 15 分鐘就能佈建 Chef 伺服器。

從 2021 年 5 月 3 日開始， 會在其中 AWS OpsWorks for Chef Automate 存放一些 Chef Automate 伺服器屬性 AWS Secrets Manager。如需詳細資訊，請參閱與 整合 AWS Secrets Manager。

下列逐步解說可協助您在其中建立第一個 Chef 伺服器 AWS OpsWorks for Chef Automate。

先決條件

開始之前，您必須完成下列先決條件。

主題

- 設定 VPC
- 使用自訂網域的先決條件 (選用)
- 設定 EC2 金鑰對 (選用)

設定 VPC

您的 AWS OpsWorks for Chef Automate 伺服器必須在 Amazon Virtual Private Cloud 中操作。您可以將其新增至現有的 VPC、使用預設 VPC，或建立新的 VPC 來包含伺服器。如需有關 Amazon VPC 以及如何建立新的 VPC 的資訊，請參閱 Amazon VPC 入門指南。

如果您建立自己的 VPC 或使用現有的 VPC，VPC 應有下列設定或屬性。

- VPC 至少應有一個子網路。

  如果您的 AWS OpsWorks for Chef Automate 伺服器將可公開存取，請將子網路設為公有，並啟用自動指派公有 IP。
- 應啟用 DNS resolution (DNS 解析)。
- 在子網路上，啟用 Auto-assign public IP (自動指派公有 IP)。

使用自訂網域的先決條件 (選用)

您可以在自己的網域上設定 Chef Automate 伺服器，在自訂網域中指定公有端點做為伺服器的端點。如本節所述，當您使用自訂網域時，下列所有項目都是必要的。

主題

- 設定自訂網域
- 取得憑證
- 取得私密金鑰

設定自訂網域

要在自己的自訂網域上執行 Chef Automate 伺服器，您將需要伺服器的公有端點，例如 `https://aws.my-company.com`。如先前章節所述，如果您指定自訂網域，您也必須提供憑證和私密金鑰。

若要在建立伺服器之後存取伺服器，請在慣用的 DNS 服務中新增 CNAME DNS 記錄。此記錄必須將自訂網域指向由 Chef 自動化伺服器建立程序所產生的端點 (伺服器 `Endpoint` 屬性的值)。如果伺服器使用自訂網域，您將無法使用產生的 `Endpoint` 值來存取伺服器。

取得憑證

若要在自己的自訂網域上設定 Chef 自動化伺服器，您需要一個 PEM 格式的 HTTPS 憑證。這可以是單一、自我簽署的憑證或憑證鏈。當您完成 Create Chef Automate server (建立 Chef 自動化伺服器) 工作流程時，如果您指定此憑證，則還必須提供自訂網域和私密金鑰。

以下是憑證值的需求：

- 您可以提供自我簽署、自訂憑證或完整的憑證鏈。
- 憑證必須是有效的 X509 憑證，或是 PEM 格式的憑證鏈。
- 憑證在上傳時必須有效。您不能在憑證有效期間開始 (憑證的 `NotBefore` 日期) 之前或憑證有效期到期 (憑證的 `NotAfter` 日期) 之後使用憑證。
- 憑證的一般名稱或主體別名 (SAN) (如果存在) 必須符合自訂網域值。
- 憑證必須符合 Custom private key (自訂私密金鑰) 欄位的值。

取得私密金鑰

若要在自己的自訂網域上設定 Chef Automate 伺服器，您需要使用 PEM 格式的私密金鑰以利用 HTTPS 連接到伺服器。私密金鑰不得加密，不能受密碼或密碼短語保護。如果您指定自訂私密金鑰，您還必須提供自訂網域和憑證。

設定 EC2 金鑰對 (選用)

Chef 伺服器的一般管理不需要或不建議使用 SSH 連線；您可以使用 `knife` 命令在 Chef 伺服器上執行大多數管理任務。

如果您遺失或想要變更 Chef Automate 儀表板的登入密碼，則需要 EC2 金鑰對，才能使用 SSH 連線至您的伺服器。您可以使用現有的金鑰對，或建立新的金鑰對。如需如何建立新的 EC2 金鑰對的詳細資訊，請參閱 Amazon EC2 金鑰對。

如果您不需要 EC2 金鑰對，則可以建立 Chef 伺服器。

建立 Chef Automate 伺服器

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或其他解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

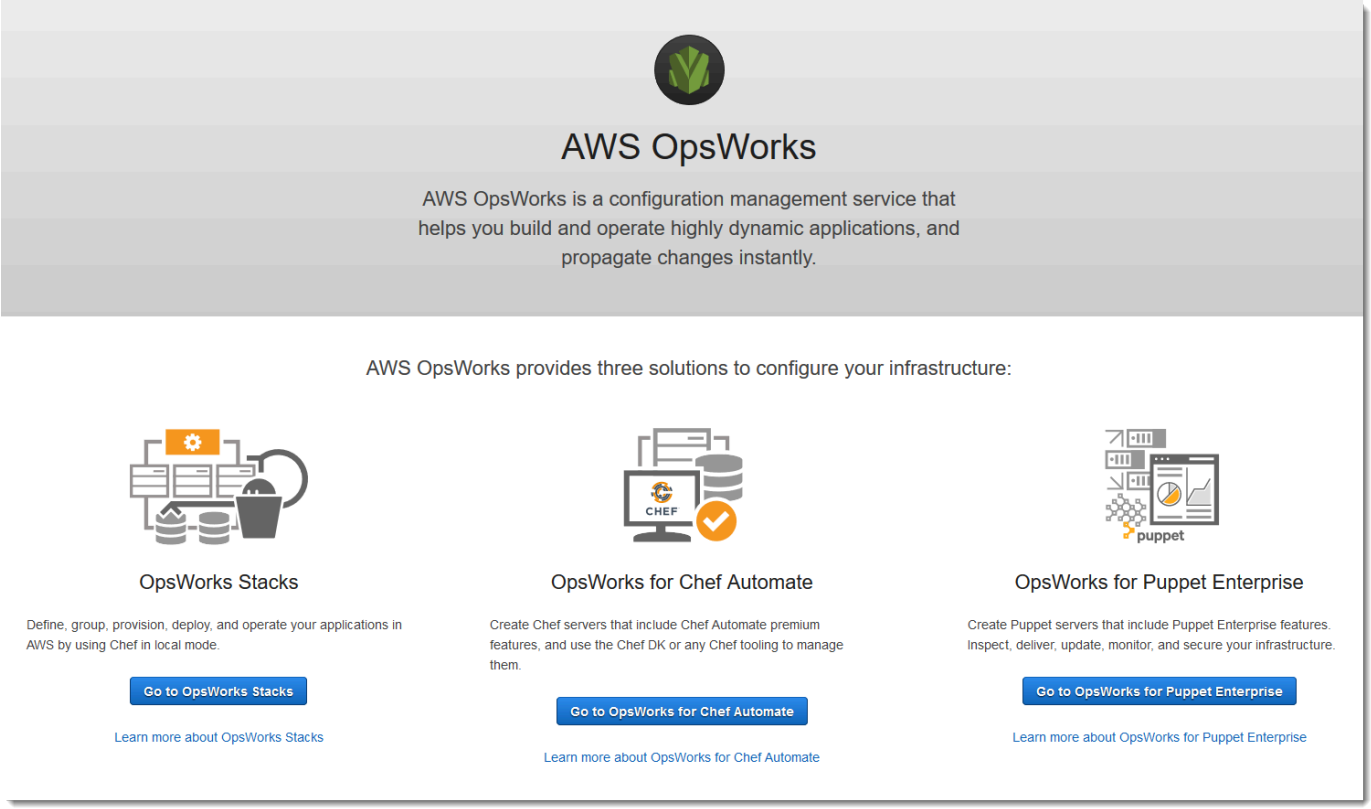
您可以使用 AWS OpsWorks for Chef Automate 主控台或 來建立 Chef 伺服器 AWS CLI。

主題

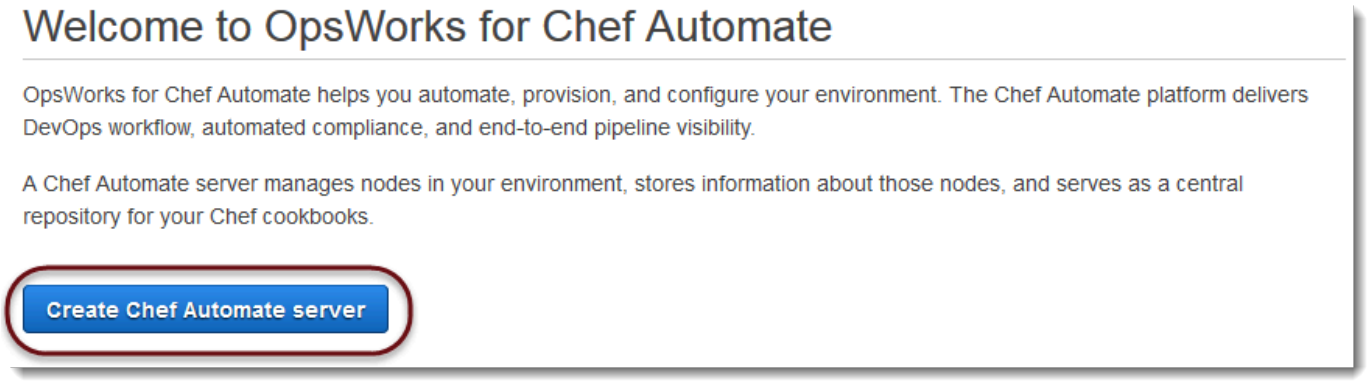
- 在 中建立 Chef Automate 伺服器 AWS Management Console
- 使用 建立 Chef Automate 伺服器 AWS CLI

在 中建立 Chef Automate 伺服器 AWS Management Console

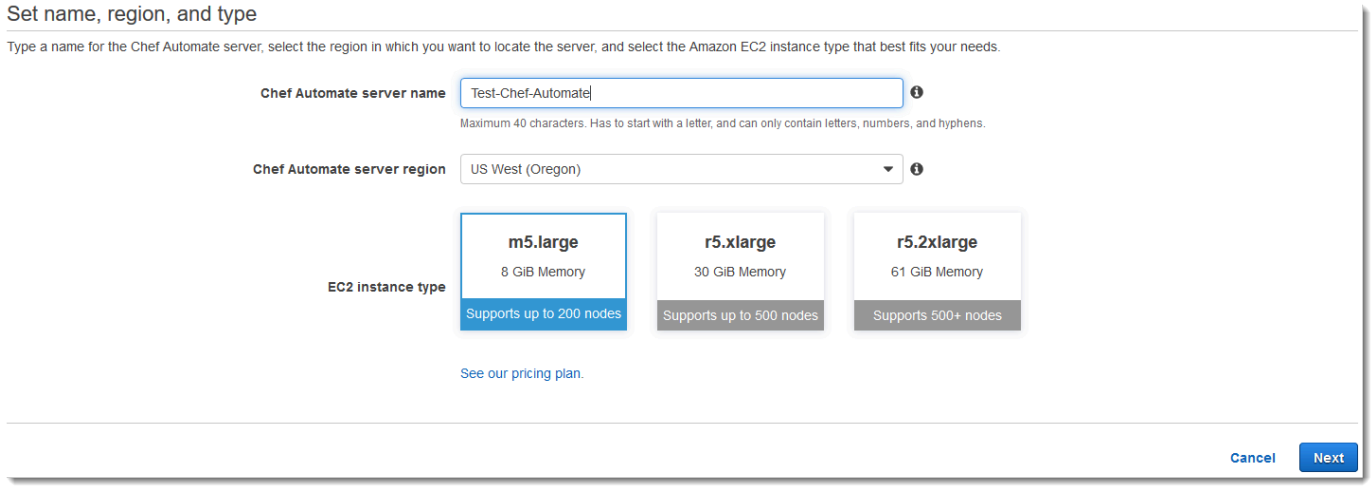
1. 登入 AWS Management Console 並在 https：//https://console.aws.amazon.com/opsworks/ 開啟 AWS OpsWorks 主控台。
2. 在 AWS OpsWorks 首頁上，選擇 Go to OpsWorks for Chef Automate。



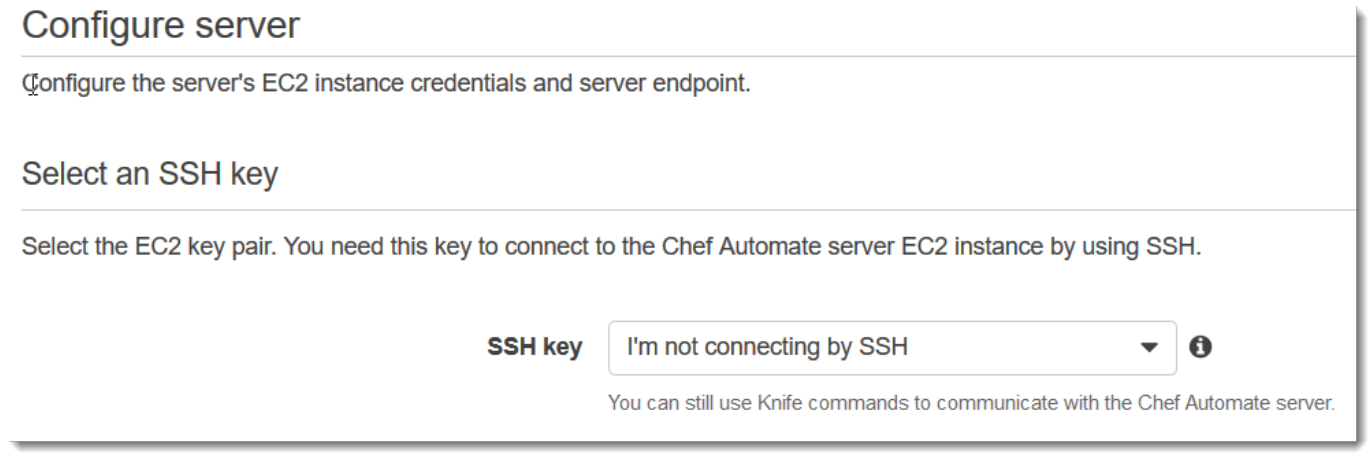
3. 在 AWS OpsWorks for Chef Automate 首頁上，選擇建立 Chef Automate 伺服器。



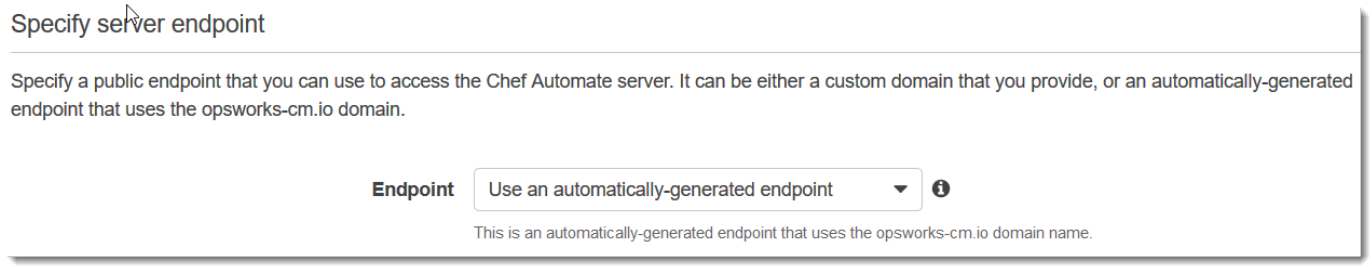
4. 在 Set name, region, and type (設定名稱、區域和類型) 頁面上，指定您的伺服器名稱。Chef 伺服器名稱最多可包含 40 個字元，而且只能包含英數字元和破折號。選取支援的區域，然後選擇支援您想要管理之節點數目的執行個體類型。如有需要，您可以在建立伺服器之後變更執行個體類型。在此演練中，我們會在美國西部（奧勒岡）區域中建立 m5.large 執行個體類型。選擇下一步。



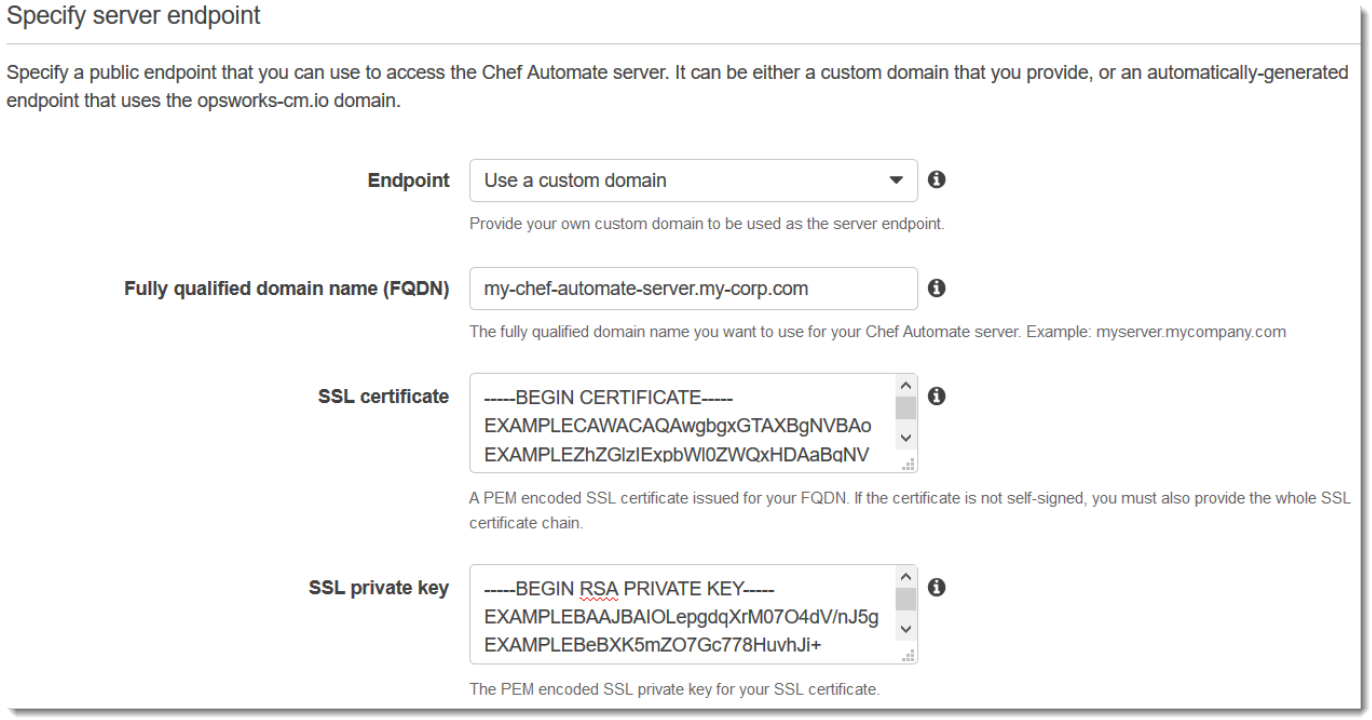
5. 在 Configure server (設定伺服器) 頁面上，保留 SSH key (SSH 金鑰) 下拉式清單中的預設選項，除非您想要指定金鑰對名稱。



6. 保留 Specify server endpoint (指定伺服器端點) 中的預設值 Use an automatically-generated endpoint (使用自動產生的端點)，然後選擇 Next (下一步)，除非您希望伺服器位於您自己的自訂網域中。若要設定自訂網域，請繼續下一個步驟。



7. 若要使用自訂網域，請在 Specify server endpoint (指定伺服器端點) 下拉式清單中選擇 Use a custom domain (使用自訂網域)。



a. 在 Fully qualified domain name (FQDN) (完整網域名稱 (FQDN)) 中指定 FQDN。您必須擁有想要使用的網域名稱。

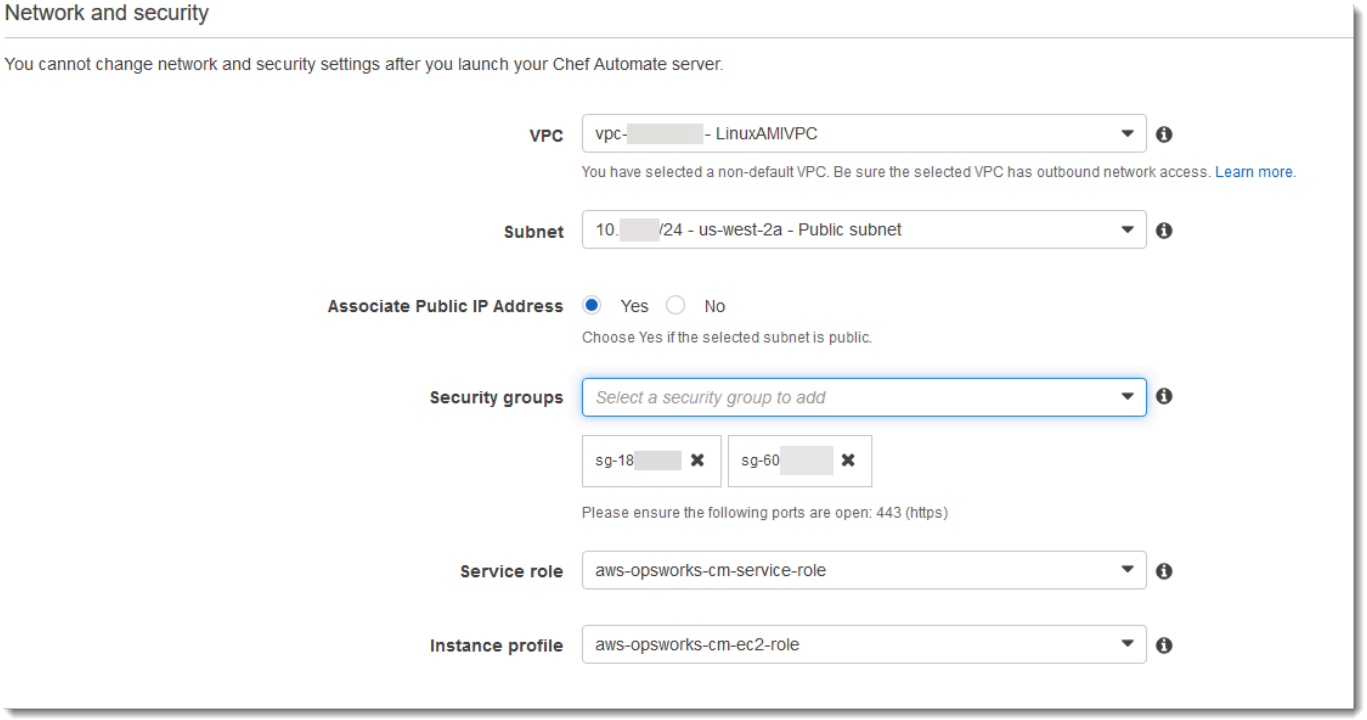
b. 在 SSL certificate (SSL 憑證) 中貼上整個 PEM 格式憑證，開頭為 `-----BEGIN CERTIFICATE-----` 且結尾為 `-----END CERTIFICATE-----`。SSL 憑證主體必須符合您在上個步驟輸入的 FQDN。

c. 在 SSL private key (SSL 私密金鑰) 中貼上整個 RSA 私密金鑰，開頭為 `-----BEGIN RSA PRIVATE KEY-----` 且結尾為 `-----END RSA PRIVATE KEY-----`。SSL 私密金鑰必須符合您在上個步驟的 SSL 憑證中輸入的公開金鑰。選擇下一步。

8. 在 Configure Advanced Settings (設定進階設定) 頁面的 Network and Security (網路與安全) 區域中，選擇 VPC、子網路，以及一或多個安全群組。下列是您的 VPC 的需求：

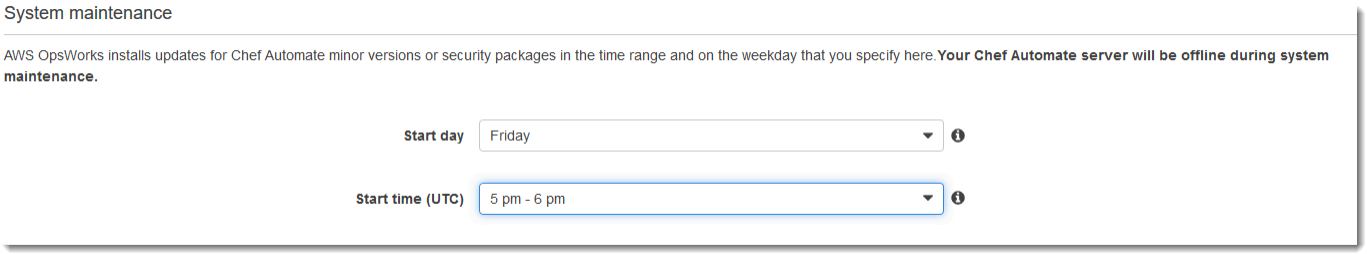
   - VPC 必須至少有一個公有子網路。
   - 必須啟用 DNS 解析。
   - 必須在公有子網路上啟用 Auto-assign public IP (自動指派公有 IP)。

   AWS OpsWorks 如果您還沒有要使用的安全群組、服務角色和執行個體描述檔， 可以為您產生。您的伺服器可以是多個安全群組的成員。離開此頁面之後，您將無法變更 Chef 伺服器的網路與安全設定。

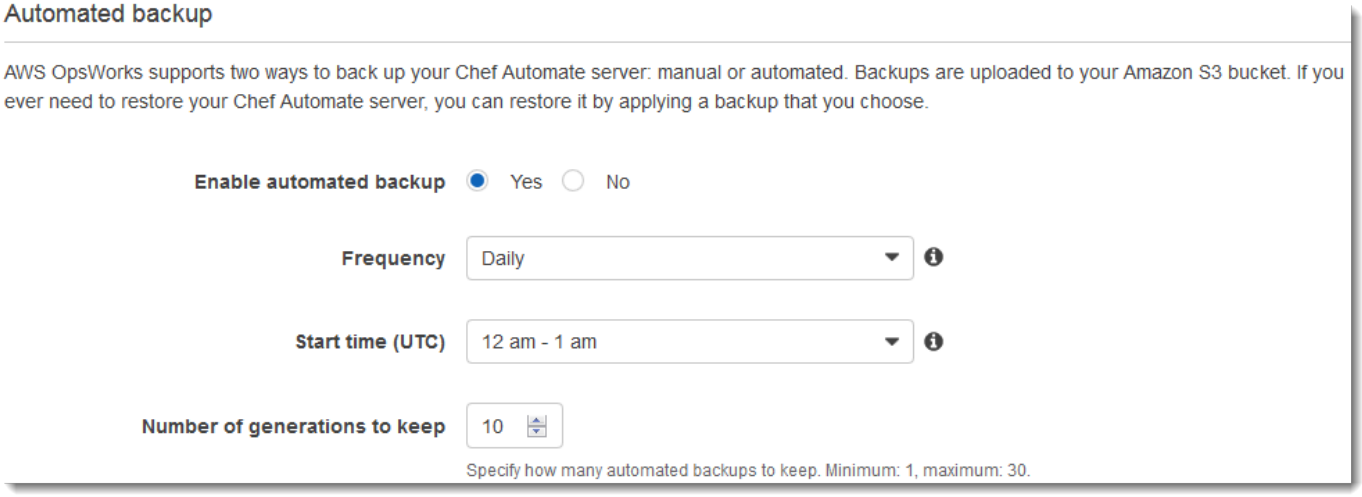


9. 在 System maintenance (系統維護) 區段中，將日和小時設為您希望開始系統維護的時間。因為您應預期伺服器在系統維護期間離線，請選擇一般工作時間內伺服器需求較低的時間。連線的節點會進入 `pending-server` 狀態，直到維護完成為止。

   需要維護時段。您可以使用 AWS Management Console AWS CLI或 APIs，在稍後變更開始日期和時間。



10. 設定備份。預設會啟用自動備份。設定自動備份開始的偏好頻率和小時數，並設定要存放在 Amazon Simple Storage Service 中的備份產生次數。最多保留 30 個備份；達到上限時， 會 AWS OpsWorks for Chef Automate 刪除最舊的備份，為新的備份騰出空間。



11. (選用) 在 Tags (標籤) 中，將標籤新增至伺服器和相關資源，例如 EC2 執行個體、彈性 IP 地址、安全群組、S3 儲存貯體和備份。如需標記 AWS OpsWorks for Chef Automate 伺服器的詳細資訊，請參閱 在 AWS OpsWorks for Chef Automate 資源上使用標籤。
12. 當您完成設定進階設定時，選擇 Next (下一步)。
13. 在 Review (檢閱) 頁面上，檢視您的選擇。當您準備好建立伺服器時，選擇 Launch (啟動)。

    當您等待 AWS OpsWorks 建立 Chef 伺服器時，請前往 使用入門套件設定 Chef 伺服器並下載入門套件和 Chef Automate 儀表板登入資料。不要等到您的伺服器上線，再下載這些項目。

    伺服器建立完成之後， AWS OpsWorks for Chef Automate 首頁即會顯示您的 Chef 伺服器，且狀態為 online (線上)。伺服器上線之後，伺服器的網域會提供 Chef Automate 儀表板，URL 格式如下：`https://`*`your_server_name`*`-random.region.opsworks-cm.io`。

使用 建立 Chef Automate 伺服器 AWS CLI

透過執行 AWS CLI 命令建立 AWS OpsWorks for Chef Automate 伺服器與在 主控台中建立伺服器不同。在 主控台中，如果您未指定要使用的現有服務角色和安全群組， 會為您 AWS OpsWorks 建立服務角色和安全群組。在 中 AWS CLI，如果您未指定 AWS OpsWorks 安全群組，但它不會自動建立服務角色，則必須在`create-server`命令中提供服務角色 ARN。在主控台中，當 AWS OpsWorks 正在建立 Chef Automate 伺服器時，您可以下載 Chef Automate 入門套件和 Chef Automate 儀表板的登入憑證。由於您無法在使用 AWS OpsWorks for Chef Automate 建立伺服器時執行此操作 AWS CLI，因此在新伺服器 AWS OpsWorks for Chef Automate 上線後，您可以使用 JSON 處理公用程式從 `create-server`命令的結果取得登入憑證和入門套件。或者，您可以在新的 AWS OpsWorks for Chef Automate 伺服器上線之後，在主控台中產生一組新的登入資料和新的入門套件。

如果您的本機電腦尚未執行 AWS CLI， AWS CLI 請依照《AWS Command Line Interface 使用者指南》中的安裝說明下載並安裝 。本節不會說明您可以搭配 `create-server` 命令使用的所有參數。如需 `create-server` 參數的詳細資訊，請參閱「參考」`create-server`中的 AWS CLI 。

1. 請務必完成這些先決條件，特別是設定 VPC，或確定您具備想要使用的現有 VPC。若要建立您的 Chef Automate 伺服器，您需要子網路 ID。
2. (選用) 使用 OpenSSL 產生 Chef 樞紐金鑰，並將金鑰儲存至您本機電腦上安全方便的檔案。如果您未在 `create-server` 命令中提供樞紐金鑰，則會在伺服器建立程序中自動產生。如果您想要略過此步驟，您可以改為從 `create-server` 命令的結果取得 Chef Automate 樞紐金鑰。如果您選擇使用以下命令產生樞紐金鑰，請務必包含 `-pubout` 參數，因為 Chef Automate 樞紐金鑰值是 RSA 金鑰對的公有部分。如需詳細資訊，請參閱步驟 6。

   ```
   umask 077
   openssl genrsa -out "pivotal" 2048
   openssl rsa -in "pivotal" -pubout
   ```

3. 建立服務角色和執行個體描述檔。
4. 尋找並複製您帳戶中服務角色ARNs。

   ```
   aws iam list-roles --path-prefix "/service-role/" --no-paginate
   ```

   在 `list-roles` 命令的結果中，尋找類似如下的服務角色 ARN 項目。請記下服務角色 ARN。您需要這些值，才能建立您的 Chef Automate 伺服器。

   ```
   {
       "AssumeRolePolicyDocument": {
           "Version": "2012-10-17",
           "Statement": [
               {
                   "Action": "sts:AssumeRole",
                   "Effect": "Allow",
                   "Principal": {
                       "Service": "ec2.amazonaws.com"
                   }
               }
           ]
       },
       "RoleId": "AROZZZZZZZZZZQG6R22HC",
       "CreateDate": "2018-01-05T20:42:20Z",
       "RoleName": "aws-opsworks-cm-ec2-role",
   ```

```
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-ec2-role"
},
{
    "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "opsworks-cm.amazonaws.com"
                }
            }
        ]
    },
    "RoleId": "AROZZZZZZZZZZZZZZZ6QE",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-service-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-service-
role"
}
```

5. 在您的帳戶中，找出並複製執行個體描述檔的 ARN。

```
aws iam list-instance-profiles --no-paginate
```

在 `list-instance-profiles` 命令的結果中，尋找類似如下的執行個體描述檔 ARN 項目。請記下執行個體描述檔 ARN。您需要這些值，才能建立您的 Chef Automate 伺服器。

```
{
    "Path": "/",
    "InstanceProfileName": "aws-opsworks-cm-ec2-role",
    "InstanceProfileId": "EXAMPLEDC6UR3LTUW7VHK",
    "Arn": "arn:aws:iam::123456789012:instance-profile/aws-opsworks-cm-ec2-role",
    "CreateDate": "2017-01-05T20:42:20Z",
    "Roles": [
        {
            "Path": "/service-role/",
            "RoleName": "aws-opsworks-cm-ec2-role",
            "RoleId": "EXAMPLEE4STNUQG6R22HC",
```

```
            "Arn": "arn:aws:iam::123456789012:role/service-role/aws-opsworks-cm-
ec2-role",
            "CreateDate": "2017-01-05T20:42:20Z",
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            }
        }
    ]
},
```

6. 執行 `create-server`命令來建立 AWS OpsWorks for Chef Automate 伺服器。

   - `--engine` 值為 `ChefAutomate`、`--engine-model` 為 `Single`，而 `--engine-version` 為 12。
   - 伺服器名稱在每個區域中 AWS 的帳戶內必須是唯一的。伺服器名稱開頭必須是字母，後面允許字母、數字或連字號 (-)，最多可包含 40 個字元。
   - 使用您在步驟 4 和 5 中複製的執行個體描述檔 ARN 和服務角色 ARN。
   - 有效的執行個體類型為 `m5.large`、`r5.xlarge` 或 `r5.2xlarge`。如需這些執行個體類型規格的詳細資訊，請參閱《Amazon EC2 使用者指南》中的執行個體類型。
   - `--engine-attributes` 參數為選用；如果您未指定一或兩個值，伺服器建立程序會為您產生值。如果您新增 `--engine-attributes`，請指定您在步驟 2 中產生的 `CHEF_AUTOMATE_PIVOTAL_KEY` 值、`CHEF_AUTOMATE_ADMIN_PASSWORD` 或兩者。

     如果您未設定 `CHEF_AUTOMATE_ADMIN_PASSWORD` 的值，這時系統會產生密碼，並在 `create-server` 回應中傳回。您也可以在主控台中再次下載入門套件，這會重新產生此密碼。密碼長度最少 八個字元，最多 32 個字元。密碼可包含字母、數字和特殊字元 (!/@#$%^+=_)。密碼必須包含至少一個小寫字母、一個大寫字母、一個數字和一個特殊字元。
   - SSH 金鑰對為選用，但如果您需要重設 Chef Automate 儀表板管理員密碼，它可協助您連線至 Chef Automate 伺服器。如需建立 SSH 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。 Amazon EC2

- 若要使用自訂網域，請將下列參數新增至您的命令。否則，Chef Automate 伺服器建立程序會自動為您產生端點。需要所有三個參數才能設定自訂網域。如需使用這些參數的其他需求的相關資訊，請參閱《CM API AWS OpsWorks 參考》中的 CreateServer。
  - `--custom-domain` - 伺服器的選用公有端點，例如 `https://aws.my-company.com`。
  - `--custom-certificate` - PEM 格式的 HTTPS 憑證。此值可以是單一、自我簽署的憑證或憑證鏈。
  - `--custom-private-key` - PEM 格式的私密金鑰，以便利用 HTTPS 連線至伺服器。私密金鑰不得加密，不能受密碼或密碼短語保護。
- 需要每週系統維護。有效值必須以下列格式指定：`DDD:HH:MM`。指定的時間是以國際標準時間 (UTC) 表示。如果您未指定 `--preferred-maintenance-window` 的值，預設值為星期二、星期三或星期五的隨機一小時時段。
- `--preferred-backup-window` 的有效值必須以下列其中一種格式指定：`HH:MM` 表示每日備份，或 `DDD:HH:MM` 表示每週備份。指定的時間是以 UTC 表示。預設值為隨機的每日開始時間。若要退出自動備份，請改為新增參數 `--disable-automated-backup`。
- 針對 `--security-group-ids`，輸入一或多個安全群組 ID，並以空格分隔。
- 針對 `--subnet-ids`，輸入子網路 ID。

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single"
 --engine-version "12" --server-name "server_name" --instance-profile-arn
 "instance_profile_ARN" --instance-type "instance_type" --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"pivotal_key","CHEF_AUTOMATE_ADMIN_PASSWORD":"password"}'
 --key-pair "key_pair_name" --preferred-maintenance-window
 "ddd:hh:mm" --preferred-backup-window "ddd:hh:mm" --security-group-
ids security_group_id1 security_group_id2 --service-role-arn "service_role_ARN" --
subnet-ids subnet_ID
```

以下是範例。

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-
model "Single" --engine-version "12" --server-name "automate-06" --
instance-profile-arn "arn:aws:iam::12345678912:instance-profile/aws-
opsworks-cm-ec2-role" --instance-type "m5.large" --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"MZZE...Wobg","CHEF_AUTOMATE_ADMIN_PASSWORD":"zZZzDj2DLYXSZF
 --key-pair "amazon-test" --preferred-maintenance-window "Mon:08:00" --preferred-
backup-window "Sun:02:00" --security-group-ids sg-b00000001 sg-b0000008 --service-
```

```
role-arn "arn:aws:iam::12345678912:role/service-role/aws-opsworks-cm-service-role"
 --subnet-ids subnet-300aaa00
```

以下範例建立一個使用自訂網域的 Chef Automate 伺服器。

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single" --
engine-version "12" \
    --server-name "my-custom-domain-server" \
    --instance-profile-arn "arn:aws:iam::12345678912:instance-profile/aws-opsworks-
cm-ec2-role" \
    --instance-type "m5.large" \
    --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"MZZE...Wobg","CHEF_AUTOMATE_ADMIN_PASSWORD":"zZZzDj2DLYXSZF
 \
    --custom-domain "my-chef-automate-server.my-corp.com" \
    --custom-certificate "-----BEGIN CERTIFICATE----- EXAMPLEqEXAMPLE== -----END
 CERTIFICATE-----" \
    --custom-private-key "-----BEGIN RSA PRIVATE KEY----- EXAMPLEqEXAMPLE= -----END
 RSA PRIVATE KEY-----" \
    --key-pair "amazon-test" \
    --preferred-maintenance-window "Mon:08:00" \
    --preferred-backup-window "Sun:02:00" \
    --security-group-ids sg-b00000001 sg-b0000008 \
    --service-role-arn "arn:aws:iam::12345678912:role/service-role/aws-opsworks-cm-
service-role" \
    --subnet-ids subnet-300aaa00
```

以下範例建立的 Chef Automate 伺服器會新增下面二個標籤：`Stage: Production` 和 `Department: Marketing`。如需在 AWS OpsWorks for Chef Automate 伺服器上新增和管理標籤的詳細資訊，請參閱本指南在 AWS OpsWorks for Chef Automate 資源上使用標籤中的 。

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single" --
engine-version "12" \
    --server-name "my-test-chef-server" \
    --instance-profile-arn "arn:aws:iam::12345678912:instance-profile/aws-opsworks-
cm-ec2-role" \
    --instance-type "m5.large" \
    --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"MZZE...Wobg","CHEF_AUTOMATE_ADMIN_PASSWORD":"zZZzDj2DLYXSZF
 \
    --key-pair "amazon-test" \
    --preferred-maintenance-window "Mon:08:00" \
```

```
    --preferred-backup-window "Sun:02:00" \
    --security-group-ids sg-b00000001 sg-b0000008 \
    --service-role-arn "arn:aws:iam::12345678912:role/service-role/aws-opsworks-cm-
service-role" \
    --subnet-ids subnet-300aaa00 \
    --tags [{\"Key\":\"Stage\",\"Value\":\"Production\"},{\"Key\":\"Department\",
\"Value\":\"Marketing\"}]
```

7. AWS OpsWorks for Chef Automate 建立新伺服器大約需要 15 分鐘。請勿關閉 `create-server` 命令的輸出或關閉您的 shell 工作階段，因為輸出可能包含不會再顯示的重要資訊。若要從 `create-server` 結果取得密碼和入門套件，請前往下一個步驟。

   如果您在伺服器上使用自訂網域，請在 `create-server` 命令的輸出中複製 `Endpoint` 屬性的值。以下是範例。

   ```
   "Endpoint": "automate-07-exampleexample.opsworks-cm.us-east-1.amazonaws.com"
   ```

8. 如果您選擇讓 為您 AWS OpsWorks for Chef Automate 產生金鑰和密碼，您可以使用 jq 等 JSON 處理器，從`create-server`結果中擷取可用格式的金鑰和密碼。安裝 jq 之後，您可以執行下列命令來解壓縮樞紐金鑰、Chef Automate 儀表板管理員密碼和入門套件。如果您未在步驟 4 中提供自己的樞紐金鑰和密碼，請務必將解壓縮的樞紐金鑰和管理員密碼儲存在方便但安全的位置。

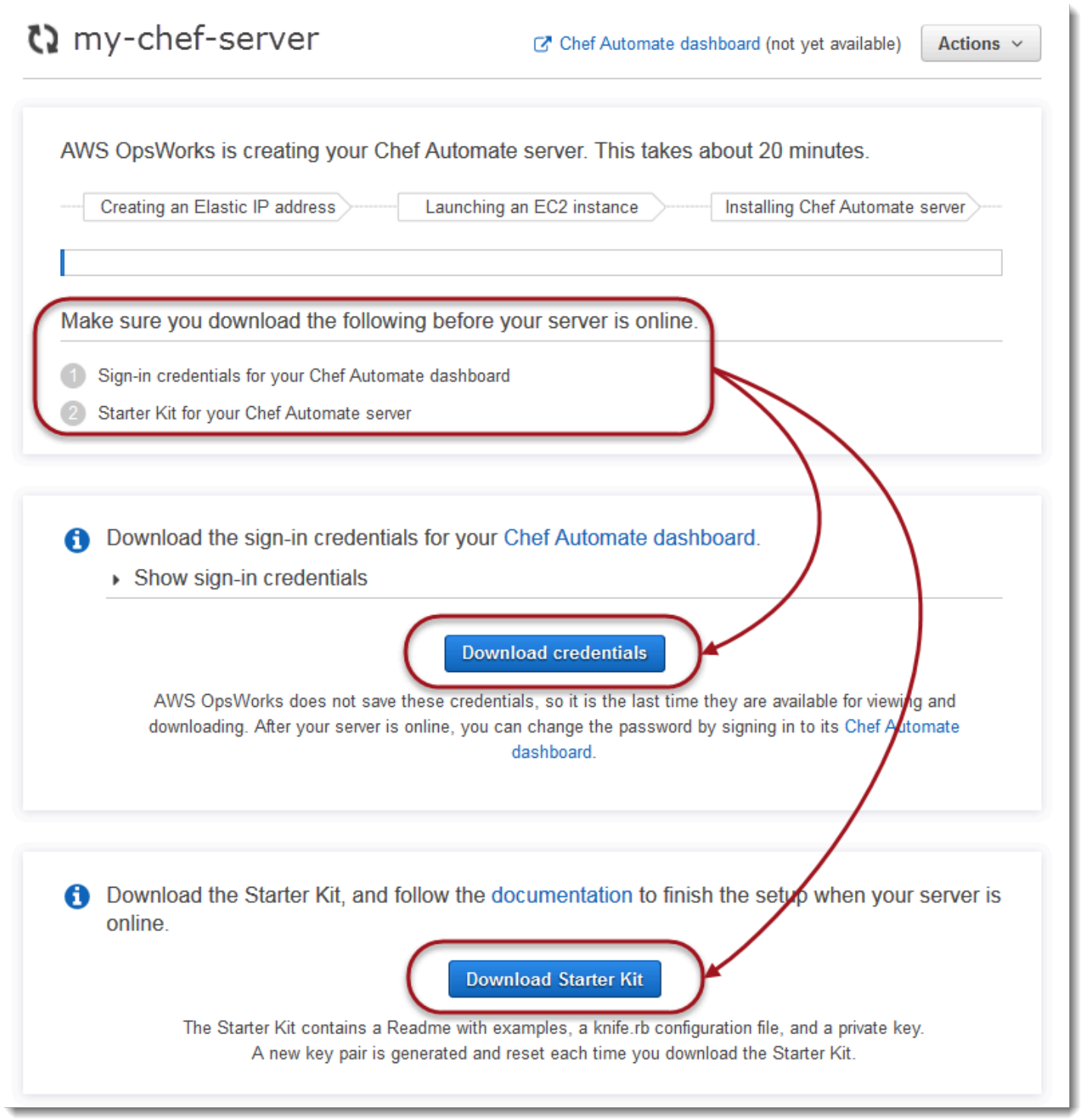
   ```
   #Get the Chef password:
   cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
    "CHEF_AUTOMATE_ADMIN_PASSWORD") | .Value'

   #Get the Chef Pivotal Key:
   cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
    "CHEF_AUTOMATE_PIVOTAL_KEY") | .Value'

   #Get the Chef Starter Kit:
   cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
    "CHEF_STARTER_KIT") | .Value' | base64 -D > starterkit.zip
   ```

9. 或者，如果您沒有從`create-server`命令結果中擷取入門套件，您可以從 AWS OpsWorks for Chef Automate 主控台的伺服器屬性頁面下載新的入門套件。下載新的入門套件會重設 Chef Automate 儀表板管理員密碼。
10. 如果您不是使用自訂網域，請繼續下一個步驟。如果您搭配 伺服器使用自訂網域，請在企業的 DNS 管理工具中建立 CNAME 項目，將自訂網域指向您在步驟 7 中複製的 AWS OpsWorks for Chef Automate 端點。在完成此步驟之前，您無法連線或登入具有自訂網域的伺服器。
11. 完成伺服器建立程序之後，繼續the section called “完成組態並上傳技術指南”。

使用入門套件設定 Chef 伺服器

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或其他解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

當 Chef 伺服器建立仍在進行時，在 AWS OpsWorks for Chef Automate 主控台中開啟其 Properties (屬性) 頁面。當您第一次使用新的 Chef 伺服器時，Properties (屬性) 頁面會提示您下載兩個必要項目。請下載這些項目後再連線至您的 Chef 伺服器；在新的伺服器上線之後，將無法使用下載按鈕。



- Sign-in credentials for the Chef server. (Chef 伺服器的登入資料。) 您將使用這些登入資料來登入 Chef Automate 儀表板，您可以在其中使用 Chef Automate 進階功能，例如工作流程和合規掃描。AWS OpsWorks 不會儲存這些登入資料；這是最後一次可供檢視和下載。如有需要，您可以在登入之後變更這些登入資料隨附的密碼。

- Starter Kit. (入門套件。) 入門套件包含內附範例的讀我檔案、knife.rb 組態檔案，以及主要或樞紐使用者的私有金鑰。每次下載入門套件，都會產生新的金鑰對，並重設舊的金鑰。

除了只能與新伺服器搭配使用的登入資料之外，入門套件 .zip 檔案還包含可與任何 AWS OpsWorks for Chef Automate 伺服器搭配使用的 Chef 儲存庫簡單範例。在 Chef 儲存庫中，您可以存放技術指南、角色、組態檔案，以及可透過 Chef 管理您節點的其他成品。我們建議您將此儲存庫存放在版本控制系統中 (例如 Git)，並將其視為來源碼。如需說明如何設定 Git 中追蹤之 Chef 儲存庫的資訊和範例，請參閱 Chef 文件中的 About the chef-repo。

先決條件

1. 當伺服器建立仍在進行時，下載 Chef 伺服器的登入資料，並將其儲存在安全但方便的位置。
2. 下載入門套件，並將入門套件 .zip 檔案解壓縮至您的工作空間目錄。請勿共享入門套件私有金鑰。如果其他使用者將要管理 Chef 伺服器，請稍後將其新增為 Chef Automate 儀表板的管理員。
3. 在您用來管理 Chef 伺服器和節點的電腦上下載並安裝 Chef Workstation （先前稱為 Chef Development Kit 或 Chef DK)。knife 公用程式是 Chef Workstation 的一部分。如需說明，請參閱 Chef 網站上的安裝 Chef 工作站。

探索入門套件內容

入門套件包含下列內容。

- cookbooks/ - 用於您所建立技術指南的目錄。cookbooks/ 資料夾包含opsworks-webserver技術指南，這是包裝函式技術指南，相依於 Chef Supermarket 網站上的nginx技術指南。如果cookbooks/目錄中沒有技術指南相依性，則 Policyfile.rb 預設為 Chef 超級市場作為次要來源。
- Policyfile.rb – 以 Ruby 為基礎的政策檔案，其定義了將成為您節點政策的技術指南、相依性和屬性。
- userdata.sh 和 userdata.ps1 - 您可以使用使用者資料檔案，在您的 Chef Automate 伺服器啟動之後自動關聯節點。userdata.sh 是用於引導以 Linux 為基礎的節點，而 userdata.ps1 是用於以 Windows 為基礎的節點。
- Berksfile - 如果您偏好使用 Berkshelf 和 berks 命令來上傳技術指南及其相依性，則您可已使用此檔案。在本逐步教學中，我們會使用 Policyfile.rb 和 Chef 命令來上傳技術指南、相依性和屬性。
- README.md 是 Markdown 型的檔案，其描述如何使用入門套件來首次設定您的 Chef Automate 伺服器。

- `.chef` 是一種隱藏目錄，其中包含一個 knife 組態檔案 (`knife.rb`) 和私密驗證金鑰檔案 (.pem)。
  - `.chef/knife.rb` - knife 組態檔案 (`knife.rb`)。已設定 `knife.rb` 檔案，讓 Chef `knife`的工具操作對 AWS OpsWorks for Chef Automate 伺服器執行。
  - `.chef/ca_certs/opsworks-cm-ca-2020-root.pem` - 憑證授權機構 (CA) 簽署的 SSL 私有金鑰，由 AWS OpsWorks所提供。此金鑰可讓伺服器向您伺服器管理之節點上的 Chef Infra 用戶端代理程式表明身分。

設定您的 Chef 儲存庫

Chef 儲存庫包含數個目錄。入門套件中的每個目錄包含讀我檔案，用於描述目錄的用途，以及如何使用它透過 Chef 來管理您的系統。您可以用兩種方式在 Chef 伺服器上安裝技術指南：執行 `knife` 命令或執行 Chef 命令，將政策檔案 (`Policyfile.rb`) 上傳到負責下載並安裝指定技術指南的伺服器。此逐步解說使用 Chef 命令和 `Policyfile.rb`，在您的伺服器上安裝技術指南。

1. 在您的本機電腦上建立用來存放技術指南的目錄，例如 `chef-repo`。將技術指南、角色和其他檔案新增至此儲存庫後，建議您將其上傳或存放在安全且版本控制的系統中，例如 CodeCommit、Git 或 Amazon S3。
2. 在 `chef-repo` 目錄中，建立以下目錄：

   - `cookbooks/` - 存放技術指南。
   - `roles/` - 以 `.rb` 或 `.json` 格式存放角色。
   - `environments/` - 以 `.rb` 或 `.json` 格式存放環境。

使用 Policyfile.rb 取得遠端來源的技術指南

在本節中，您要編輯 `Policyfile.rb` 來指定技術指南，然後執行 Chef 命令，將檔案上傳到您的伺服器和技術指南。

1. 在您的入門套件中查看 `Policyfile.rb`。在預設情況下， `Policyfile.rb` 包含 `opsworks-webserver` 包裝函式技術指南，實際內容會依 Chef Supermarket 網站所提供 `nginx` 技術指南而定。`nginx` 技術指南會在受管節點上安裝和設定 Web 伺服器。這時也會指定必要的 `chef-client` 技術指南，其將在受管節點上安裝 Chef Infra 用戶端代理程式。

   `Policyfile.rb` 也會指向選用的 Chef Audit 技術指南，其可供您設定節點上的合規掃描。如需為受管節點設定合規掃描和取得合規結果的詳細資訊，請參閱 中的合規掃描 AWS OpsWorks for Chef Automate。如果您不想要現在設定合規掃描和稽核，請刪除 `run_list` 區段中的 `'audit'`，而且不要在檔案結尾指定 `audit` 技術指南屬性。

```
# Policyfile.rb - Describe how you want Chef to build your system.
#
# For more information about the Policyfile feature, visit

# https://docs.chef.io/policyfile.html


# A name that describes what the system you're building with Chef does.

name 'opsworks-demo-webserver'

# The cookbooks directory is the preferred source for external cookbooks

default_source :chef_repo, "cookbooks/" do |s|

  s.preferred_for "nginx", "windows", "chef-client", "yum-epel", "seven_zip",
                  "build-essential", "mingw", "ohai", "audit", "logrotate", "cron"

end
# Alternative source
default_source :supermarket


# run_list: chef-client runs these recipes in the order specified.

run_list  'chef-client',
          'opsworks-webserver',
          'audit'
# add 'ssh-hardening' to your runlist to fix compliance issues detected by the ssh-
baseline profile

# Specify a custom source for a single cookbook:

cookbook 'opsworks-webserver', path: 'cookbooks/opsworks-webserver'


# Policyfile defined attributes

# Define audit cookbook attributes
default["opsworks-demo"]["audit"]["reporter"] = "chef-server-automate"
default["opsworks-demo"]["audit"]["profiles"] = [
  {
    "name": "DevSec SSH Baseline",
```

```
    "compliance": "admin/ssh-baseline"
  }
]
```

下面介紹當您現在只想要設定 nginx Web 伺服器時，所出現不含 audit 技術指南和屬性的 Policyfile.rb。

```
# Policyfile.rb - Describe how you want Chef to build your system.
#
# For more information on the Policyfile feature, visit
# https://docs.chef.io/policyfile.html

# A name that describes what the system you're building with Chef does.
name 'opsworks-demo-webserver'

# Where to find external cookbooks:
default_source :supermarket

# run_list: chef-client will run these recipes in the order specified.
run_list  'chef-client',
          'opsworks-webserver'

# Specify a custom source for a single cookbook:
cookbook 'opsworks-webserver', path: 'cookbooks/opsworks-webserver'
```

如果您變更 Policyfile.rb, 請務必儲存檔案。

2. 下載並安裝 Policyfile.rb 中所定義的技術指南。

```
chef install
```

所有技術指南的版本都是經由技術指南的 metadata.rb 檔案所控制。每次變更技術指南時，您都必須在技術指南的 metadata.rb 中提高其版本。

3. 如果您選擇了設定合規掃描，並在政策檔案中保留 audit 說明書資訊，則請將政策 opsworks-demo 推送到您的伺服器。

```
chef push opsworks-demo
```

4. 如果您已完成步驟 3，請驗證您的政策安裝情況。執行下列命令。

```
chef show-policy
```

結果應類似以下內容：

```
opsworks-demo-webserver
=======================
* opsworks-demo:  ec0fe46314
```

5. 您現在可以將節點新增或引導到您的 Chef Automate 伺服器。您可以遵循在 中自動新增節點 AWS OpsWorks for Chef Automate中的步驟將節點的關聯自動化，或遵循個別新增節點中的步驟來逐一新增節點。

(替代方法) 使用 Berkshelf 取得遠端來源的技術指南

Berkshelf 是用於管理技術指南及其相依性的一種工具。如果您在將技術指南安裝到本機儲存空間時，偏好使用 Berkshelf 而不是 `Policyfile.rb` 進行安裝，請使用本節所明程序，而不要用先前章節內容。您可以指定要搭配您的 Chef 伺服器使用的技術指南和版本，之後再予以上傳。此入門套件包含名為 `Berksfile` 的檔案，其中列出您的技術指南。

1. 若要開始使用，請將 `chef-client` 技術指南新增到已包含的 Berksfile。`chef-client` 技術指南會為每部連接到 Chef Automate 伺服器的節點設定 Chef Infra 代理程式軟體。若要進一步了解此技術指南，請參閱 Chef Supermarket 中的 Chef Client Cookbook。
2. 使用文字編輯器，將另一個安裝Web 伺服器應用程式的其他技術指南附加到 Berksfile；例如，安裝 Apache Web 伺服器的 `apache2` 技術指南。您的 Berksfile 應該如下所示。

```
source 'https://supermarket.chef.io'
cookbook 'chef-client'
cookbook 'apache2'
```

3. 在您的本機電腦上下載並安裝技術指南。

```
berks install
```

4. 將技術指南上傳至 Chef 伺服器。

   在 Linux 上，執行：

```
SSL_CERT_FILE='.chef/ca_certs/opsworks-cm-ca-2020-root.pem' berks upload
```

在 Windows 上，在 PowerShell 工作階段中執行下列 Chef Workstation 命令。執行命令之前，請務必將 PowerShell 中的執行政策設為 `RemoteSigned`。新增 `chef shell-init` 讓 Chef Workstation 公用程式命令可供 PowerShell 使用。

```
$env:SSL_CERT_FILE="ca_certs\opsworks-cm-ca-2020-root.pem"
chef shell-init berks upload
Remove-Item Env:\SSL_CERT_FILE
```

5. 顯示 Chef Automate 伺服器上目前可用的技術指南清單，驗證技術指南的安裝情況。您可以執行下列 `knife` 命令，即可進行新增：

   您已準備好使用 AWS OpsWorks for Chef Automate 伺服器新增要管理的節點。

   ```
   knife cookbook list
   ```

(選用) 設定 **`knife`** 以使用自訂網域

如果您的 Chef Automate 伺服器使用自訂網域，您可能需要新增 (指派您伺服器的憑證鏈的) 根 CA 的 PEM 憑證，或者如果憑證是自我簽署的，則需要新增伺服器 PEM 憑證。`ca_certs` 是 `chef/` 的一個子目錄，其中包含 Chef `knife` 公用程式信任的證書授權機構 (CA)。

如果您未使用自訂網域，或您的自訂憑證是由作業系統信任的根 CA 簽署的，則可以略過本節。否則，請設定 `knife` 信任您的 Chef Automate 伺服器 SSL 憑證，如以下步驟所述。

1. 執行下列命令。

   ```
   knife ssl check
   ```

   如果結果類似於以下內容，請略過此程序的其餘部分，然後前往 為 Chef 伺服器新增要管理的節點。

   ```
   Connecting to host my-chef-automate-server.my-corp.com:443
            Successfully verified certificates from 'my-chef-automate-server.my-
   corp.com'
   ```

   如果出現類似下列的錯誤訊息，請繼續下一個步驟。

   ```
   Connecting to host my-chef-automate-server.my-corp.com:443
   ```

```
          ERROR: The SSL certificate of my-chef-automate-server.my-corp.com could
 not be verified.
          ...
```

2. 執行 `knife ssl fetch` 以信任 AWS OpsWorks for Chef Automate 伺服器的憑證。或者，您可以手動將伺服器的根 CA PEM 格式憑證複製到 `knife ssl check` 輸出中的 `trusted_certs_dir` 值的目錄。根據預設，此目錄位於入門套件中的 `.chef/ca_certs/`。輸出應類似以下內容：

```
WARNING: Certificates from my-chef-automate-server.my-corp.com will be fetched and
 placed in your trusted_cert
          directory (/Users/username/starterkit/.chef/../.chef/ca_certs).

          Knife has no means to verify these are the correct certificates. You
 should
          verify the authenticity of these certificates after downloading.

          Adding certificate for my-chef-automate-server in /Users/users/
starterkit/.chef/../.chef/ca_certs/servv-aqtswxu20swzkjgz.crt
          Adding certificate for MyCorp_Root_CA in /Users/users/
starterkit/.chef/../.chef/ca_certs/MyCorp_Root_CA.crt
```

3. 再次執行 `knife ssl check`。輸出應類似以下內容：

```
Connecting to host my-chef-automate-server.my-corp.com:443
          Successfully verified certificates from 'my-chef-automate-server.my-
corp.com'
```

您已準備好使用 `knife` 搭配您的 Chef Automate 伺服器。

為 Chef 伺服器新增要管理的節點

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或其他解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

chef-client 代理程式會在與伺服器相關聯的實體或虛擬電腦 (稱為節點) 上執行 Chef 配方。您可以將現場部署電腦或執行個體連線至 Chef 伺服器來管理，只要節點執行支援的作業系統。向 Chef 伺服器註冊節點會在這些節點上安裝 chef-client 代理程式軟體。

您可以使用下列方法來新增節點：

- 執行可新增或引導 EC2 執行個體的knife命令來個別新增備註，以便 Chef 伺服器可以對其進行管理。如需更多資訊，請參閱個別新增節點。
- 使用指令碼自動新增節點，以執行節點與 Chef 伺服器的全自動關聯。在入門套件中的程式碼會示範如何使用無人執行的方法自動新增節點。如需詳細資訊，請參閱 在 中自動新增節點 AWS OpsWorks for Chef Automate。

主題

- 個別新增節點
- 在 中自動新增節點 AWS OpsWorks for Chef Automate

個別新增節點

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或其他解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節說明如何執行新增或引導 EC2 執行個體的knife命令，以便 Chef 伺服器可以管理它。

與 AWS OpsWorks for Chef Automate 伺服器相關聯節點上的 chef-client 最低支援版本為 13.x。我們建議您執行最新且穩定的chef-client版本。

主題

- (選用) 指定 Chef Automate 伺服器根 CA 的 URL
- 支援的作業系統
- 新增 Knife 節點

(選用) 指定 Chef Automate 伺服器根 CA 的 URL

如果您的伺服器使用自訂網域和憑證，您可能需要使用公開 URL 來編輯 userdata 指令碼中的 ROOT_CA_URL 變數，以用於取得伺服器的根 CA PEM 格式憑證。下列 AWS CLI 命令會將您的根 CA 上傳至 Amazon S3 儲存貯體，並產生預先簽章的 URL，供您使用一小時。

1. 將根 CA PEM 格式的憑證上傳到 S3。

```
aws s3 cp ROOT_CA_PEM_FILE_PATH s3://bucket_name/
```

2. 產生您可以使用一小時 (在此範例中為 3600 秒) 的預先簽章的 URL 以下載根 CA。

```
aws s3 presign s3://bucket_name/ROOT_CA_PEM_FILE_NAME --expires-in 3600
```

3. 使用預先簽章的 URL 的值編輯 userdata 指令碼中的 ROOT_CA_URL 變數。

支援的作業系統

如需節點目前的支援作業系統清單，請參閱 Chef 網站。

新增 Knife 節點

knife-ec2 外掛程式包含在 Chef Workstation 中。如果您比較熟悉 knife-ec2，則可以使用它來取代 knife bootstrap，以便佈建及引導新的 EC2 執行個體。否則，啟動新的 EC2 執行個體，然後依照本節中的步驟進行。

新增要管理的節點

1. 執行下列 knife bootstrap 命令。此命令會將 EC2 執行個體引導至您 Chef 伺服器將管理的節點。請注意，您會指示 Chef 伺服器執行 the section called “使用 Policyfile.rb 取得遠端來源的技術指南” 中所安裝 nginx 技術指南的配方。如需執行 knife bootstrap 命令新增節點的詳細資訊，請參閱 Chef 文件中的 Bootstrap a Node。

   下表顯示此步驟之 knife 命令中節點作業系統的有效使用者名稱。如果 root 和 ec2-user 都無法運作，請聯絡您的 AMI 供應商。如需連線至 Linux 執行個體的詳細資訊，請參閱 AWS 文件中的使用 SSH 連線至您的 Linux 執行個體。

節點作業系統中使用者名稱的有效值

作業系統	有效的使用者名稱
Amazon Linux	`ec2-user`
Red Hat Enterprise Linux 5	`root` 或 `ec2-user`
Ubuntu	`ubuntu`
Fedora	`fedora` 或 `ec2-user`
SUSE Linux	`root` 或 `ec2-user`

```
knife bootstrap INSTANCE_IP_ADDRESS -N INSTANCE_NAME -x USER_NAME --sudo --run-list
 "recipe[nginx]"
```

2. 執行下列命令，並以您剛新增的執行個體名稱取代 *INSTANCE_NAME*，確認已新增節點。

```
knife client show INSTANCE_NAME
knife node show INSTANCE_NAME
```

在 中自動新增節點 AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或替代解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本主題說明如何自動將 Amazon Elastic Compute Cloud (Amazon EC2) 節點新增至 Chef 伺服器。在入門套件中的程式碼會示範如何使用無人執行的方法自動新增節點。建議設定 Chef 用戶端技術指南，來使用無人執行 (或自動) 的方法建立新節點的關聯。您可以使用入門套件中的 `userdata` 指令碼，並搭配您要套用到您的節點的技術指南變更 `userdata` 指令碼的 `run_list` 區段，或者 `Policyfile.rb`。在您執行 `chef-client` 代理程式前，請先將 Chef 用戶端技術指南安裝到 Chef

伺服器，然後使用如 HTTPD 角色在服務模式中安裝 chef-client 代理程式，如以下範例命令所示。

```
chef-client -r "chef-client,role[httpd]"
```

為了能與 Chef 伺服器通訊，chef-client 代理程式軟體必須具有用戶端節點的公有金鑰存取權。您可以在 Amazon EC2 中產生公有/私有金鑰對，然後使用節點名稱將公有金鑰傳遞給 AWS OpsWorks associate-node API 呼叫。本入門主題中所提供的指令碼包含在入門套件中，其中為您收集了您的組織名稱、伺服器名稱和伺服器端點。這可確保節點與 Chef 伺服器相關聯，且於配對私有金鑰後，執行於該節點上的 chef-client 代理程式軟體可與伺服器通訊。

與 AWS OpsWorks for Chef Automate 伺服器相關聯節點上的 chef-client 最低支援版本為 13.x。我們建議您執行最新且穩定的chef-client版本。

如需如何取消節點關聯的詳細資訊，請參閱本指南取消節點與 AWS OpsWorks for Chef Automate 伺服器的關聯中的 和 AWS OpsWorks for Chef Automate API 文件disassociate-node中的 。

主題

- 支援的作業系統
- 步驟 1：建立 IAM 角色以用作您的執行個體描述檔
- 步驟 2：安裝 Chef 用戶端技術指南
- 步驟 3：使用無人執行的關聯指令碼建立執行個體
- 其他可自動重複執行 chef-client 的方法
- 相關主題

支援的作業系統

如需節點目前的支援作業系統清單，請參閱 Chef 網站。

步驟 1：建立 IAM 角色以用作您的執行個體描述檔

建立 AWS Identity and Access Management (IAM) 角色以用作您的 EC2 執行個體描述檔，並將下列政策連接至 IAM 角色。此政策可讓 AWS OpsWorks for Chef Automate (opsworks-cm) API 在節點註冊期間與 EC2 執行個體通訊。如需執行個體描述檔的詳細資訊，請參閱 Amazon EC2 文件中的使用執行個體描述檔。如需如何建立 IAM 角色的資訊，請參閱 Amazon EC2 文件中的在主控台中建立 IAM 角色。 Amazon EC2

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "opsworks-cm:AssociateNode",
                "opsworks-cm:DescribeNodeAssociationStatus",
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

步驟 2：安裝 Chef 用戶端技術指南

如果您尚未這麼做，請遵循中的步驟，(替代方法) 使用 Berkshelf 取得遠端來源的技術指南以確保您的 Berksfile 或 `Policyfile.rb` 檔案參考 Chef Client 技術指南並安裝技術指南。

步驟 3：使用無人執行的關聯指令碼建立執行個體

1. 若要建立 EC2 執行個體，您可以將`userdata`指令碼從入門套件複製到 EC2 執行個體說明、Amazon EC2 Auto Scaling 群組啟動組態或 AWS CloudFormation 範本的 `userdata`區段。如需將指令碼新增至使用者資料的詳細資訊，請參閱 Amazon EC2 文件中的啟動時在 Linux 執行個體上執行命令。

   此指令碼會執行 `opsworks-cm` API `associate-node` 命令，將新節點與 Chef 伺服器建立關聯。

   根據預設，新註冊的節點名稱為執行個體 ID，但您可以修改 `userdata` 指令碼中的 `NODE_NAME` 變數值來變更該名稱。由於目前無法在 Chef 主控台 UI 上變更組織名稱，請將 `CHEF_AUTOMATE_ORGANIZATION` 設定保留為 `default`。
2. 遵循 EC2 文件中啟動執行個體內的程序，並進行此處修改。在 EC2 執行個體啟動精靈中，選擇 Amazon Linux AMI。
3. 在 Configure Instance Details (設定執行個體詳細資訊) 頁面上，選取您在步驟 1：建立 IAM 角色以用作您的執行個體描述檔中建立的角色做為您的 IAM 角色。
4. 在 Advanced Details (進階詳細資訊) 區域內，上傳您先前在此程序中建立的 `userdata.sh` 指令碼。
5. 您無須在 Add Storage (新增儲存體) 頁面上進行任何變更。前往 Add Tags (新增標籤)。

6. 在 Configure Security Group (設定安全群組) 頁面上，選擇 Add Rule (新增規則)，然後選擇類型 HTTP 以為此範例中的 Apache Web 伺服器開啟連接埠號碼 443 和 80。
7. 選擇 Review and Launch (檢閱及啟動)，然後選擇 Launch (啟動)。當新節點啟動時，就會套用您在 RUN_LIST 參數中所指定配方指定的設定。
8. 選用：如果您已將 nginx 技術指南新增至執行清單中，當您開啟連結至新節點公有 DNS 的網頁時，應該會看到 nginx Web 伺服器代管的網站。

其他可自動重複執行 **chef-client** 的方法

雖然較難達成且不建議，但您可以僅作為獨立執行個體使用者資料的一部分執行本主題中的指令碼、使用 AWS CloudFormation 範本將其新增至新的執行個體使用者資料、設定cron任務定期執行指令碼，或在服務chef-client中執行。不過，建議您使用 Chef 用戶端技術指南方法，原因是其他自動化技術存在某些缺點：

如需您可以為 chef-client 提供的完整參數清單，請參閱 Chef 文件。

相關主題

下列 AWS 部落格文章提供有關使用 Auto Scaling 群組或在多個帳戶中自動將節點與 Chef Automate 伺服器建立關聯的詳細資訊。

- 使用 AWS OpsWorks for Chef Automate 搭配 Auto Scaling 管理 EC2 執行個體
- OpsWorks for Chef Automate – 自動引導不同帳戶中的節點

登入 Chef Automate 儀表板

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或其他解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

從 Chef 伺服器的 Properties (屬性) 頁面下載登入資料，且伺服器在線上之後，登入 Chef Automate 儀表板。在本演練中，我們已指示您先上傳技術指南，並新增至少一個要管理的節點。這可讓您在儀表板中看到有關技術指南和節點的資訊。

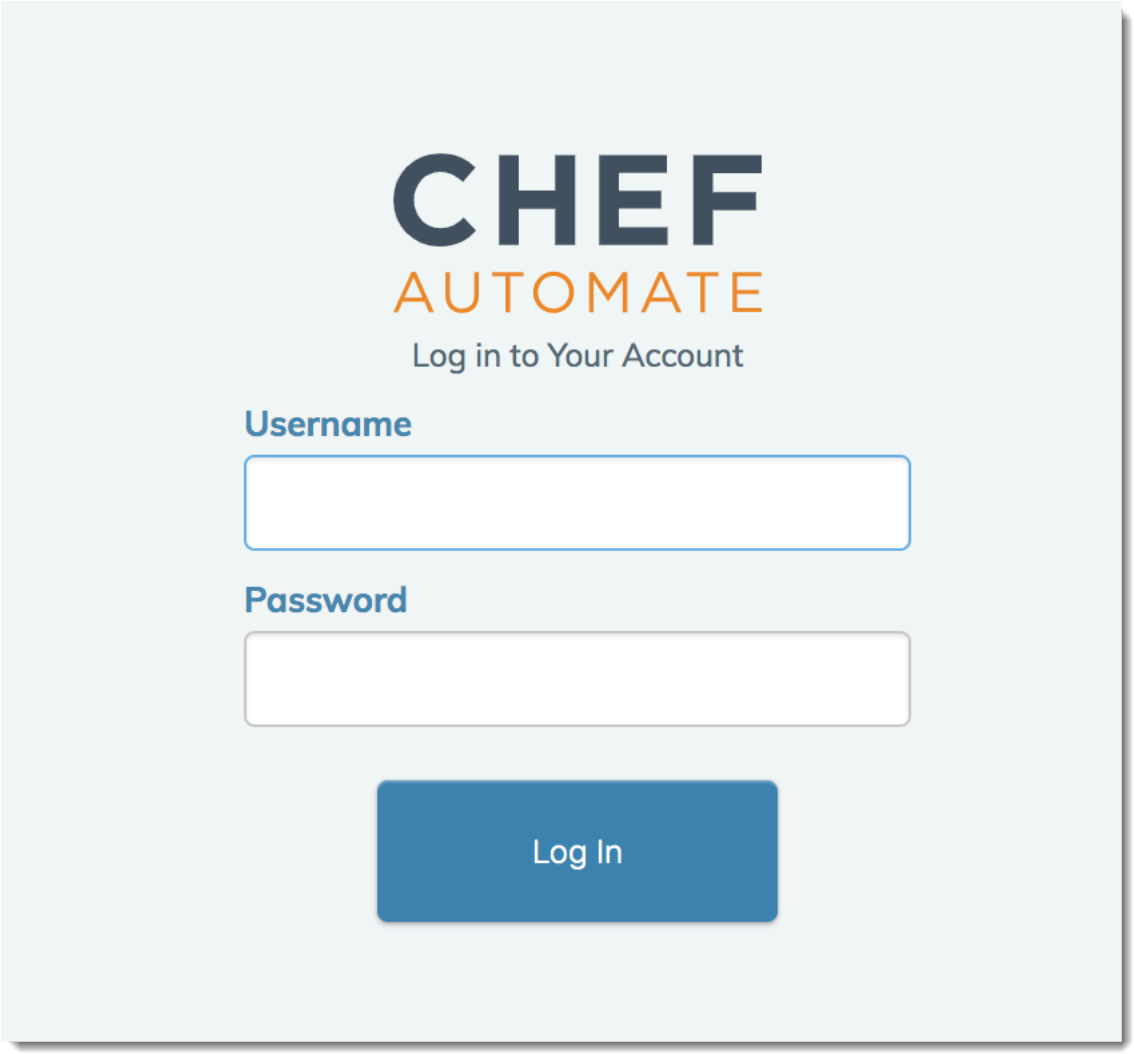
當您嘗試連線至儀表板網頁時，憑證警告會顯示在瀏覽器中，直到您在您用來管理 Chef 伺服器的用戶端電腦上安裝 AWS OpsWorks特定的 CA 簽署 SSL 憑證為止。如果您在繼續前往儀表板網頁之前不想看到警告，請安裝 SSL 憑證後再登入。

安裝用戶端 SSL 憑證之後，您可以登入 Chef Automate 儀表板，而且不會看到警告訊息。

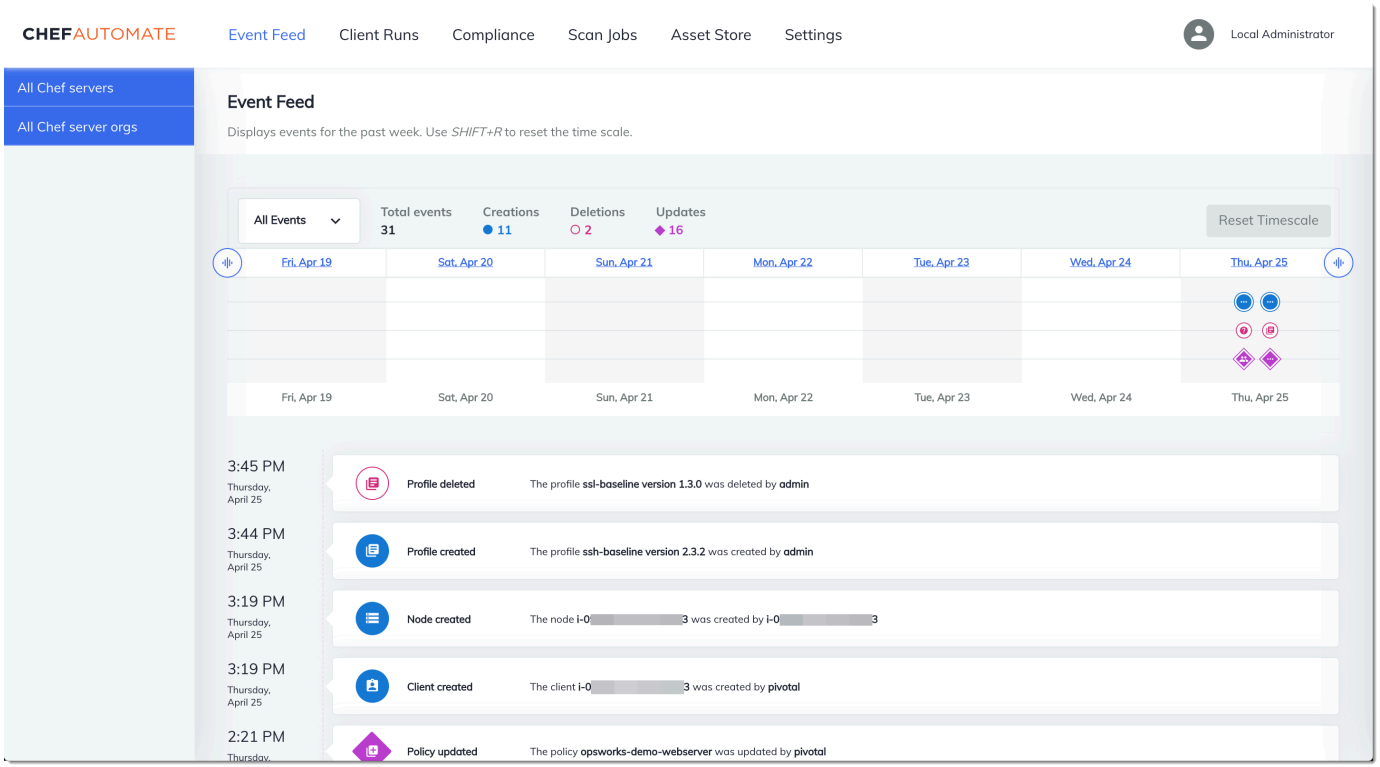
> **Note**
>
> Ubuntu 和 Linux Mint 作業系統上的 Google Chrome 使用者可能無法登入。我們建議您使用 Mozilla Firefox 或其他瀏覽器登入，並使用這些作業系統上的 Chef Automate 儀表板。找不到在 Windows 或 MacOS 上使用 Google Chrome 的問題。

登入 Chef Automate 儀表板

1. 解壓縮並開啟您在先決條件中下載的 Chef Automate 登入資料。您需要這些登入資料才能登入。
2. 開啟您 Chef 伺服器的 Properties (屬性) 頁面。
3. 在 Properties (屬性) 頁面右上方，選擇 Open Chef Automate dashboard (開啟 Chef Automate 儀表板)。
4. 使用步驟 1 中的登入資料登入。



5. 在 Chef Automate 儀表板中，您可以檢視有關您已引導的節點、技術指南執行進度和事件、節點的合規層級等詳細資訊。如需 Chef Automate 儀表板之功能及其使用方式的詳細資訊，請參閱 Chef Automate 文件。



Note

如需如何變更您用來登入 Chef Automate 儀表板之密碼的資訊，請參閱重設 Chef Automate 儀表板登入資料。

使用 建立 AWS OpsWorks for Chef Automate 伺服器 AWS CloudFormation

Important

AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或替代解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks for Chef Automate 可讓您在 中執行 Chef Automate 伺服器 AWS。只要約 15 分鐘就能佈建 Chef Automate 伺服器。

從 2021 年 5 月 3 日開始， 會在 中 AWS OpsWorks for Chef Automate 存放一些 Chef Automate 伺服器屬性 AWS Secrets Manager。如需詳細資訊，請參閱與 整合 AWS Secrets Manager。

下列逐步解說可協助您在 中建立堆疊， AWS OpsWorks for Chef Automate 以在 中建立伺服器 AWS CloudFormation。

主題

- 先決條件
- 在 AWS CloudFormation中建立 Chef Automate 伺服器

先決條件

在建立新的 Chef Automate 伺服器之前，請先建立 AWS OpsWorks for Chef Automate 之外的資源，即您將用來存取和管理 Chef 伺服器的資源。如需詳細資訊，請參閱本指南中「入門」一節的先決條件。

檢閱 AWS CloudFormation 使用者指南範本參考中的 OpsWorks-CM 區段，以了解您用來建立伺服器的 AWS CloudFormation 範本中支援和必要的值。

如果您要建立使用自訂網域的伺服器，您需要自訂網域、憑證和私密金鑰。您必須為 AWS CloudFormation 範本中這三個參數指定值。如需 `CustomDomain`、 `CustomCertificate`和 `CustomPrivateKey` 參數需求的詳細資訊，請參閱《AWS OpsWorks CM API 參考》中的 CreateServer。

建立 `CHEF_AUTOMATE_ADMIN_PASSWORD` 引擎屬性值的密碼值。密碼長度最少 八個字元，最多 32 個字元。密碼可包含字母、數字和特殊字元 (`!/@#$%^+=_`)。密碼必須包含至少一個小寫字母、一個大寫字母、一個數字和一個特殊字元。您可以在 AWS CloudFormation 範本中指定此密碼，或在建立堆疊時將此密碼指定為 `CHEF_AUTOMATE_ADMIN_PASSWORD` 參數的值。

開始建立 Chef Automate 伺服器之前，請先產生 base64 編碼的 RSA 金鑰對 AWS CloudFormation。金鑰對的公有金鑰，就是 `CHEF_AUTOMATE_PIVOTAL_KEY` 的值，Chef 特定 EngineAttributes，來源是 CreateServer API。此金鑰在 AWS CloudFormation 主控台中提供為參數的值，或在 的 create-stack命令中提供 AWS CLI。若要產生這個金鑰，我們建議採用以下方法。

- 在 Linux 電腦上，您可以執行以下 OpenSSL 命令來產生此金鑰。

```
openssl genrsa -out pivotal_key_file_name.pem 2048
```

然後，將金鑰對的 RSA 公有金鑰部分匯出到檔案。公有金鑰會變成 CHEF_AUTOMATE_PIVOTAL_KEY 的值。

```
openssl rsa -in pivotal_key_file_name.pem -pubout -out public.pem -outform PEM
```

- 您可以在以 Windows 為基礎的電腦，使用 PuTTYgen 公用程式來產生 base64 編碼的 RSA 金鑰對。如需詳細資訊，請參閱 SSH.com 上的 PuTTYgen – 適用於 Windows 上 PuTTY 的金鑰產生器。

在 AWS CloudFormation中建立 Chef Automate 伺服器

本節說明如何使用 AWS CloudFormation 範本來建置建立 AWS OpsWorks for Chef Automate 伺服器的堆疊。您可以使用 AWS CloudFormation 主控台或 來執行此操作 AWS CLI。您可以使用範例 AWS CloudFormation 範本來建置 AWS OpsWorks for Chef Automate 伺服器堆疊。請務必使用您自己的伺服器名稱、IAM 角色、執行個體描述檔、伺服器描述、備份保留計數、維護選項和選用標籤來更新範例範本。如果您的伺服器將使用自訂網域，您必須在 AWS CloudFormation 範本中指定 CustomDomain、CustomCertificate和 CustomPrivateKey 參數的值。您可以在 AWS CloudFormation 範本中指定 CHEF_AUTOMATE_ADMIN_PASSWORD和CHEF_AUTOMATE_PIVOTAL_KEY引擎屬性及其值，或僅提供屬性，然後在 AWS CloudFormation 建立堆疊精靈或create-stack命令中指定屬性的值。如需有關這些屬性的詳細資訊，請參閱本指南「入門」一節的the section called “在 中建立 Chef Automate 伺服器 AWS Management Console”。

主題

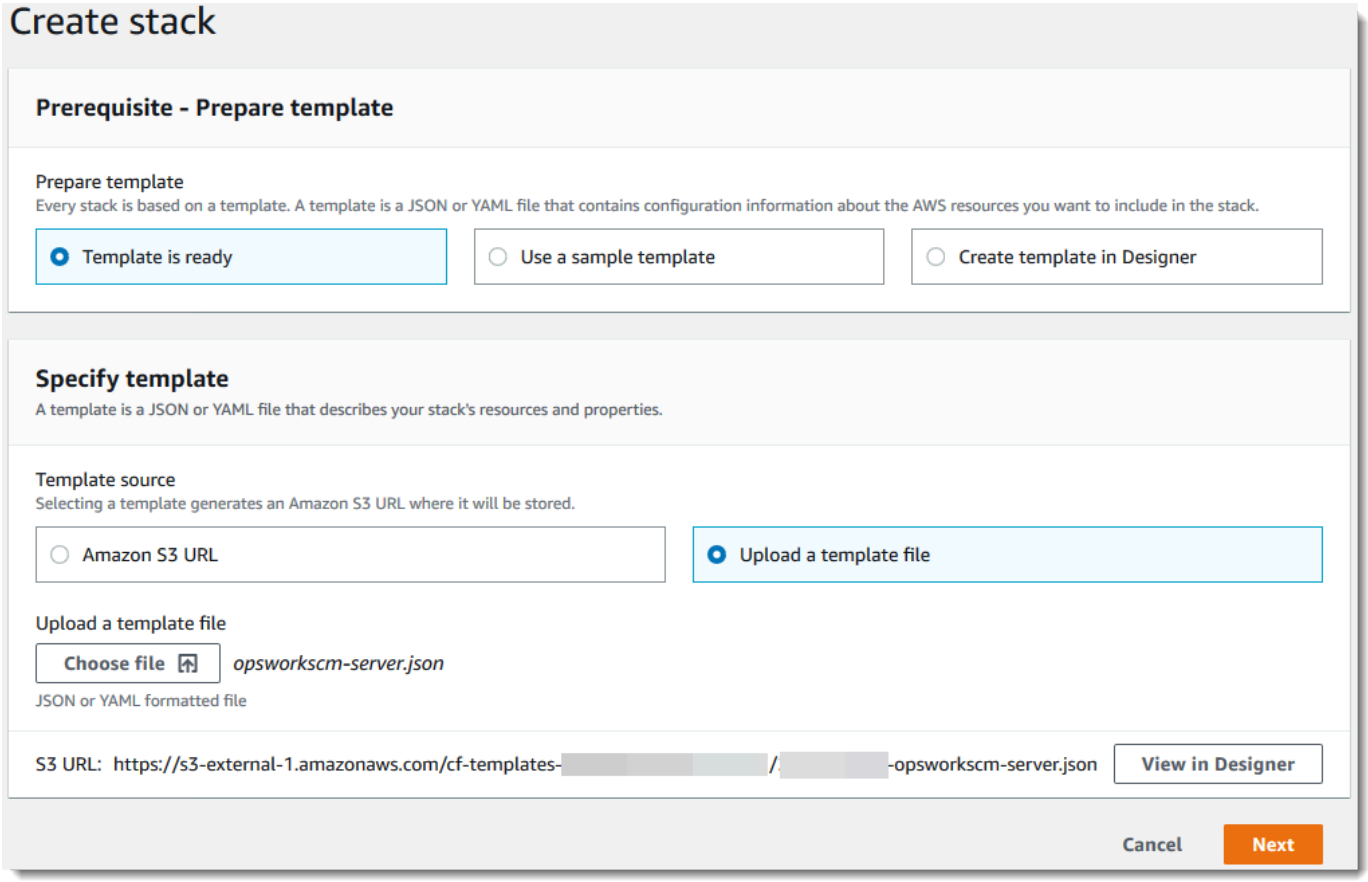
- 使用 AWS CloudFormation (主控台) 建立 Chef Automate 伺服器
- 使用 AWS CloudFormation (CLI) 建立 Chef Automate 伺服器

使用 AWS CloudFormation (主控台) 建立 Chef Automate 伺服器

1. 登入 AWS Management Console 並開啟位於 https：//https://console.aws.amazon.com/cloudformation 的 AWS CloudFormation 主控台。
2. 在 AWS CloudFormation 首頁上，選擇建立堆疊。
3. 在先決條件 - 準備範本中，如果您使用範例 AWS CloudFormation 範本，請選擇範本已就緒。

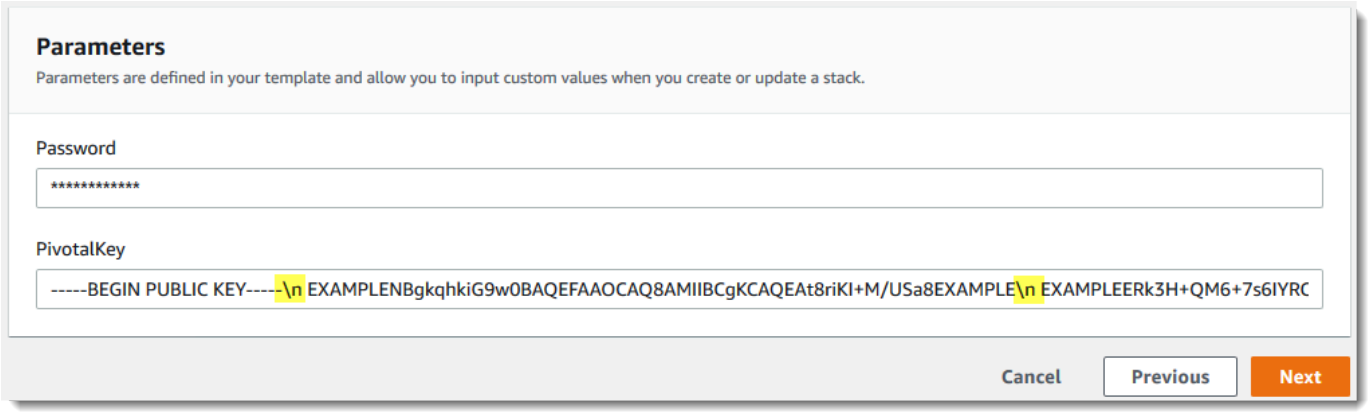
4. 在 Specify template (指定範本) 中，請選擇範本的來源。在此演練中，選擇上傳範本檔案，然後上傳建立 Chef Automate 伺服器的 AWS CloudFormation 範本。瀏覽您的範本檔案，然後選擇 Next (下一步)。

   AWS CloudFormation 範本可以是 YAML 或 JSON 格式。範例 AWS CloudFormation 範本可供您使用；請務必將範例值取代為您自己的值。您可以使用 AWS CloudFormation 範本設計工具來建立新的範本或驗證現有的範本。如需如何執行此操作的詳細資訊，請參閱AWS CloudFormation《使用者指南》中的AWS CloudFormation 設計工具界面概觀。



5. 在 Specify stack details (指定堆疊詳細資訊) 頁面上，輸入堆疊的名稱。這個名稱不會與您的伺服器名稱相同，這只是堆疊名稱。在 Parameters (參數) 區域中，貼上您已在 the section called “先決條件” 中建立的值。在 Password (密碼) 中輸入密碼。

   在 PivotalKey 中貼入 RSA 金鑰檔案的內容。在 AWS CloudFormation 主控台中，您必須在樞紐索引鍵值的每一行結尾新增換行 (**\n**) 字元，如下列螢幕擷取畫面所示。選擇下一步。



6. 在設定堆疊選項頁面上，您可以將標籤新增至使用堆疊建立的伺服器，如果您尚未指定要在範本中使用的 IAM 角色，請選擇用於建立資源的 IAM 角色。指定選項完成後，選擇 Next (下一步)。如需復原觸發等進階選項的詳細資訊，請參閱AWS CloudFormation 《 使用者指南》中的設定 AWS CloudFormation 堆疊選項。

7. 在 Review (檢閱) 頁面上，檢視您的選擇。當您準備好建立伺服器堆疊時，請選擇 Create stack (建立堆疊)。

   當您等待 AWS CloudFormation 建立堆疊時，請檢視堆疊建立狀態。如果堆疊建立失敗，則檢閱顯示在主控台中的錯誤訊息，以協助您解決問題。如需 AWS CloudFormation 針對堆疊中的錯誤進行故障診斷的詳細資訊，請參閱AWS CloudFormation 《 使用者指南》中的故障診斷錯誤。

   伺服器建立完成之後， AWS OpsWorks for Chef Automate 首頁即會顯示您的 Chef Automate 伺服器，且狀態為 online (線上)。從伺服器的「屬性」頁面，產生新的 Starter Kit (入門套件) 和 Chef Automate 儀表板登入資料。伺服器上線之後，伺服器的網域會提供 Chef Automate 儀表板，URL 格式如下：`https://your_server_name-randomID.region.opsworks-cm.io`。

   > **Note**
   >
   > 如果您為伺服器指定自訂網域、憑證和私有金鑰，請在企業的 DNS 管理工具中建立 CNAME 項目，將您的自訂網域映射到為伺服器 AWS OpsWorks for Chef Automate 自動產生的端點。您無法管理伺服器或連接到伺服器的 Chef Automate 儀表板，直到您將產生的端點對應到您的自訂網域值。
   >
   > 若要取得產生的端點值，請在伺服器上線後執行下列 AWS CLI 命令：
   >
   > ```
   > aws opsworks describe-servers --server-name server_name
   > ```

使用 AWS CloudFormation (CLI) 建立 Chef Automate 伺服器

如果您的本機電腦尚未執行 AWS CLI，AWS CLI 請依照《AWS Command Line Interface 使用者指南》中的安裝說明下載並安裝 。本節不會說明您可以搭配 create-stack 命令使用的所有參數。如需 create-stack 參數的詳細資訊，請參閱「參考」create-stack中的 AWS CLI 。

1. 請務必完成先決條件建立伺服器的 AWS OpsWorks for Chef Automate 。
2. 建立服務角色和執行個體描述檔。

   尋找並複製您帳戶中服務角色ARNs。

   ```
   aws iam list-roles --path-prefix "/service-role/" --no-paginate
   ```

   在 `list-roles` 命令的結果中，尋找類似如下的服務角色和執行個體描述檔項目。記下服務角色和執行個體描述檔的 ARNs，並將其新增至您用來建立伺服器堆疊的 AWS CloudFormation 範本。

   ```
   {
       "AssumeRolePolicyDocument": {
           "Version": "2012-10-17",
           "Statement": [
               {
                   "Action": "sts:AssumeRole",
                   "Effect": "Allow",
                   "Principal": {
                       "Service": "ec2.amazonaws.com"
                   }
               }
           ]
       },
       "RoleId": "AROZZZZZZZZZZQG6R22HC",
       "CreateDate": "2018-01-05T20:42:20Z",
       "RoleName": "aws-opsworks-cm-ec2-role",
       "Path": "/service-role/",
       "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-ec2-role"
   },
   {
       "AssumeRolePolicyDocument": {
           "Version": "2012-10-17",
           "Statement": [
               {
   ```

```
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Principal": {
                    "Service": "opsworks-cm.amazonaws.com"
                }
            }
        ]
    },
    "RoleId": "AROZZZZZZZZZZZZZZZ6QE",
    "CreateDate": "2018-01-05T20:42:20Z",
    "RoleName": "aws-opsworks-cm-service-role",
    "Path": "/service-role/",
    "Arn": "arn:aws:iam::000000000000:role/service-role/aws-opsworks-cm-service-
role"
}
```

3. 再次執行 create-stack命令來建立 AWS OpsWorks for Chef Automate 伺服器。

   - 將 *stack_name* 更換成您的堆疊名稱。這是 AWS CloudFormation 堆疊的名稱，而不是您的 Chef Automate 伺服器。Chef Automate 伺服器名稱是 AWS CloudFormation 範本ServerName中的 值。
   - 將 *template* 更換成您的範本檔案路徑，並視情況，將副檔名*yaml # json* 改成 .yaml 或 .json。
   - --parameters 的值會對應到 EngineAttributes，其來源是 CreateServer API。處理 Chef 時，用於建立伺服器、由使用者提供的引擎屬性是 CHEF_AUTOMATE_PIVOTAL_KEY，即您依據the section called “先決條件”所述步驟，使用公用程式產生的 base64 編碼 RSA 公用金鑰；以及 CHEF_AUTOMATE_ADMIN_PASSWORD，即您所建立，長度介於 8 和 32 個字元的密碼。如需 CHEF_AUTOMATE_ADMIN_PASSWORD 的詳細資訊，請參閱 使用 建立 Chef Automate 伺服器 AWS CLI。您可以依照範例所示，提供指向包含關鍵金鑰之 PEM 檔案的指標，做為 PivotalKey 參數的值。如果您的範本中CHEF_AUTOMATE_PIVOTAL_KEY未指定CHEF_AUTOMATE_ADMIN_PASSWORD和 的值，您必須在 AWS CLI 命令中提供這些值。

```
aws cloudformation create-stack --stack-name stack_name
 --template-body file://template.yaml or json --parameters
 ParameterKey=PivotalKey,ParameterValue="base64_encoded_RSA_public_key_value"
```

下面範例包括了 CHEF_AUTOMATE_ADMIN_PASSWORD 和 CHEF_AUTOMATE_PIVOTAL_KEY 屬性的示範值。如果您未在 AWS CloudFormation 範本中指定這些屬性的值，請執行類似的命令。

```
aws cloudformation create-stack --stack-name "OpsWorksCMChefServerStack"
 --template-body file://opsworkscm-server.yaml --parameters
 ParameterKey=PivotalKey,ParameterValue="$(openssl rsa -in "pivotalKey.pem" -
pubout)" ParameterKey=Password,ParameterValue="SuPer\$ecret890"
```

4. 堆疊建立完成後，請在 AWS OpsWorks for Chef Automate 主控台中開啟新伺服器的屬性頁面，並下載入門套件。下載新的入門套件會重設 Chef Automate 儀表板管理員密碼。
5. 如果您的伺服器將使用自訂網域、憑證和私有金鑰，請遵循在 knife.rb中設定的步驟(選用) 設定 knife 以使用自訂網域，然後前往步驟 7。
6. 若要在新伺服器上使用 knife 命令，請更新 Chef knife.rb 組態檔案設定。入門套件中隨附一個範例 knife.rb 檔案。下列範例顯示如何在不使用自訂網域的伺服器上設定 knife.rb。如果您正在使用自訂網域，請參閱 (選用) 設定 knife 以使用自訂網域 以取得 knife 組態指示。

   - 將 *ENDPOINT* 更換成伺服器的端點值。這是堆疊建立操作輸出的一部分。您可以執行以下命令來取得端點。

     ```
     aws cloudformation describe-stacks --stack-name stack_name
     ```

   - 將client_key組態中的 *key_pair_file.pem* 取代為 PEM 檔案名稱，其中包含CHEF_AUTOMATE_PIVOTAL_KEY您用來建立伺服器的 。

     ```
     base_dir = File.join(File.dirname(File.expand_path(__FILE__)), '..')

     log_level                :info
     log_location             STDOUT
     node_name                'pivotal'
     client_key               File.join(base_dir, '.chef', 'key_pair_file.pem')
     syntax_check_cache_path  File.join(base_dir, '.chef', 'syntax_check_cache')
     cookbook_path            [File.join(base_dir, 'cookbooks')]

     chef_server_url          'ENDPOINT/organizations/default'
     ssl_ca_file              File.join(base_dir, '.chef', 'ca_certs', 'opsworks-cm-
     ca-2020-root.pem')
     trusted_certs_dir        File.join(base_dir, '.chef', 'ca_certs')
     ```

7. 完成伺服器建立程序之後，繼續the section called “完成組態並上傳技術指南”。如果堆疊建立失敗，則檢閱顯示在主控台中的錯誤訊息，以協助您解決問題。如需 AWS CloudFormation 針對堆疊中的錯誤進行疑難排解的詳細資訊，請參閱AWS CloudFormation 《 使用者指南》中的對錯誤進行疑難排解。

更新 AWS OpsWorks for Chef Automate 伺服器以使用自訂網域

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或替代解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節說明如何使用伺服器備份來建立新 AWS OpsWorks for Chef Automate 伺服器，以更新現有伺服器來使用自訂網域和憑證。基本上，您要複製現有的 AWS OpsWorks for Chef Automate 2.0 伺服器，方法是從備份建立新伺服器，然後將新伺服器設定為使用自訂網域、憑證和私有金鑰。

主題

- 先決條件
- 限制
- 更新伺服器以使用自訂網域
- 另請參閱

先決條件

以下是更新現有 AWS OpsWorks for Chef Automate 伺服器以使用自訂網域和憑證的要求。

- 要更新 (或複製) 的伺服器必須執行 Chef Automate 2.0。
- 決定要使用哪個備份來建立新伺服器。您必須至少有一個可用的備份，以供您要更新的伺服器使用。如需 中備份的詳細資訊 AWS OpsWorks for Chef Automate，請參閱 備份 AWS OpsWorks for Chef Automate 伺服器。
- 準備好您要用來建立做為備份來源的現有伺服器的服務角色和執行個體描述檔 ARN。
- 請確定您正在執行最新版本的 AWS CLI。如需更新 AWS CLI 工具的詳細資訊，請參閱《AWS 命令列界面使用者指南》中的安裝 AWS CLI 。

限制

當您透過從備份建立新伺服器來更新現有伺服器時，新伺服器不能與現有 AWS OpsWorks for Chef Automate 伺服器完全相同。

- 您只能使用 AWS CLI 或其中一個 AWS SDKs來完成此程序。您無法使用 AWS Management Console從備份建立新伺服器。
- 新伺服器不能使用與帳戶內及 AWS 區域內現有伺服器相同的名稱。名稱必須與您做為備份來源使用的現有伺服器不同。
- 連接到現有伺服器的節點不會由新伺服器管理。您必須執行下列其中一項作業。
  - 連接不同的節點，因為節點不能由多個 Chef Automate 伺服器管理。
  - 將節點從現有伺服器 (備份的來源) 移轉至新伺服器和新的自訂網域端點。如需有關如何遷移節點的詳細資訊，請參閱 Chef 文件。

更新伺服器以使用自訂網域

若要更新現有的 Chef Automate 2.0 伺服器，您可以製作它的副本，方法是執行 `create-server` 命令，並新增參數以指定備份、自訂網域、自訂憑證和自訂私密金鑰。

1. 如果您沒有可在 `create-server` 命令中指定的服務角色或執行個體描述檔 ARN，請遵循 使用 建立 Chef Automate 伺服器 AWS CLI 中的步驟 1-5 來建立可使用的服務角色和執行個體設定檔。
2. 如果您還沒有這樣做，請找到現有的 Chef Automate 2.0 伺服器的備份 (您想以該伺服器為基礎來建立使用自訂網域之新伺服器)。執行下列命令，以顯示您 帳戶和 區域中所有 AWS OpsWorks for Chef Automate 備份的相關資訊。請記下您要使用的備份 ID。

   ```
   aws opsworks-cm --region region name describe-backups
   ```

3. 執行 `create-server`命令來建立 AWS OpsWorks for Chef Automate 伺服器。

   - `--engine` 值為 `ChefAutomate`、`--engine-model` 為 `Single`，而 `--engine-version` 為 12。
   - 伺服器名稱在 AWS 您的帳戶中在每個區域中必須是唯一的。伺服器名稱開頭必須是字母，後面允許字母、數字或連字號 (-)，最多可包含 40 個字元。
   - 使用步驟 1 中的執行個體描述檔 ARN 和服務角色 ARN。
   - 有效的執行個體類型為 `m5.large`、`r5.xlarge` 或 `r5.2xlarge`。如需這些執行個體類型規格的詳細資訊，請參閱《Amazon EC2 使用者指南》中的執行個體類型。
   - `--engine-attributes` 參數為選用；如果您未指定一或兩個值，伺服器建立程序會為您產生值。如果您新增 `--engine-attributes`，請指定您在步驟 2 中產生的 `CHEF_AUTOMATE_PIVOTAL_KEY` 值、`CHEF_AUTOMATE_ADMIN_PASSWORD` 或兩者。

如果您未設定 CHEF_AUTOMATE_ADMIN_PASSWORD 的值，這時系統會產生密碼，並在 create-server 回應中傳回。您也可以在主控台中再次下載入門套件，這會重新產生此密碼。密碼長度最少 八個字元，最多 32 個字元。密碼可包含字母、數字和特殊字元 (!/@#$%^+=_)。密碼必須包含至少一個小寫字母、一個大寫字母、一個數字和一個特殊字元。

- SSH 金鑰對為選用，但如果您需要重設 Chef Automate 儀表板管理員密碼，它可協助您連線至 Chef Automate 伺服器。如需建立 SSH 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。 Amazon EC2
- 若要使用自訂網域，請將下列參數新增至您的命令。否則，Chef Automate 伺服器建立程序會自動為您產生端點。需要所有三個參數才能設定自訂網域。如需使用這些參數的其他需求的相關資訊，請參閱 CM API AWS OpsWorks 參考中的 CreateServer。
  - --custom-domain - 伺服器的選用公有端點，例如 https://aws.my-company.com。
  - --custom-certificate - PEM 格式的 HTTPS 憑證。此值可以是單一、自我簽署的憑證或憑證鏈。
  - --custom-private-key - PEM 格式的私密金鑰，以便利用 HTTPS 連線至伺服器。私密金鑰不得加密，不能受密碼或密碼短語保護。
- 需要每週系統維護。有效值必須以下列格式指定：DDD:HH:MM。指定的時間是以國際標準時間 (UTC) 表示。如果您未指定 --preferred-maintenance-window 的值，預設值為星期二、星期三或星期五的隨機一小時時段。
- --preferred-backup-window 的有效值必須以下列其中一種格式指定：HH:MM 表示每日備份，或 DDD:HH:MM 表示每週備份。指定的時間是以 UTC 表示。預設值為隨機的每日開始時間。若要退出自動備份，請改為新增參數 --disable-automated-backup。
- 針對 --security-group-ids，輸入一或多個安全群組 ID，並以空格分隔。
- 針對 --subnet-ids，輸入子網路 ID。
- 針對 --backup-id，輸入您在步驟 2 中複製的備份 ID。

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single"
 --engine-version "12" --server-name "server_name" --instance-profile-arn
 "instance_profile_ARN" --instance-type "instance_type" --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"pivotal_key","CHEF_AUTOMATE_ADMIN_PASSWORD":"password"}'
 --key-pair "key_pair_name" --preferred-maintenance-window
 "ddd:hh:mm" --preferred-backup-window "ddd:hh:mm" --security-group-
ids security_group_id1 security_group_id2 --service-role-arn "service_role_ARN" --
subnet-ids subnet_ID --backup-id backup_ID
```

以下範例建立一個使用自訂網域的 Chef Automate 伺服器。

```
aws opsworks-cm create-server --engine "ChefAutomate" --engine-model "Single" --
engine-version "12" \
    --server-name "my-custom-domain-server" \
    --instance-profile-arn "arn:aws:iam::12345678912:instance-profile/aws-opsworks-
cm-ec2-role" \
    --instance-type "m5.large" \
    --engine-attributes
 '{"CHEF_AUTOMATE_PIVOTAL_KEY":"MZZE...Wobg","CHEF_AUTOMATE_ADMIN_PASSWORD":"zZZzDj2DLYXSZF
 \
    --custom-domain "my-chef-automate-server.my-corp.com" \
    --custom-certificate "-----BEGIN CERTIFICATE----- EXAMPLEqEXAMPLE== -----END
 CERTIFICATE-----" \
    --custom-private-key "-----BEGIN RSA PRIVATE KEY----- EXAMPLEqEXAMPLE= -----END
 RSA PRIVATE KEY-----" \
    --key-pair "amazon-test" \
    --preferred-maintenance-window "Mon:08:00" \
    --preferred-backup-window "Sun:02:00" \
    --security-group-ids sg-b00000001 sg-b0000008 \
    --service-role-arn "arn:aws:iam::12345678912:role/service-role/aws-opsworks-cm-
service-role" \
    --subnet-ids subnet-300aaa00 \
    --backup-id MyChefServer-20191004122143125
```

4. AWS OpsWorks for Chef Automate 建立新伺服器大約需要 15 分鐘。在 `create-server` 命令的輸出中，複製 `Endpoint` 屬性的值。以下是範例。

   ```
   "Endpoint": "automate-07-exampleexample.opsworks-cm.us-east-1.amazonaws.com"
   ```

   請勿關閉 `create-server` 命令的輸出或關閉您的 shell 工作階段，因為輸出可能包含不會再顯示的重要資訊。若要從 `create-server` 結果取得密碼和入門套件，請前往下一個步驟。

5. 如果您選擇為您 AWS OpsWorks for Chef Automate 產生金鑰和密碼，您可以使用 JSON 處理器，例如 jq，從`create-server`結果中擷取可用格式的金鑰和密碼。安裝 jq 之後，您可以執行下列命令來解壓縮樞紐金鑰、Chef Automate 儀表板管理員密碼和入門套件。如果您未在步驟 3 中提供自己的樞紐金鑰和密碼，請務必將解壓縮的樞紐金鑰和管理員密碼儲存在方便但安全的位置。

   ```
   #Get the Chef password:
   ```

```
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
 "CHEF_AUTOMATE_ADMIN_PASSWORD") | .Value'

#Get the Chef Pivotal Key:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
 "CHEF_AUTOMATE_PIVOTAL_KEY") | .Value'

#Get the Chef Starter Kit:
cat resp.json | jq -r '.Server.EngineAttributes[] | select(.Name ==
 "CHEF_STARTER_KIT") | .Value' | base64 -D > starterkit.zip
```

6. 或者，如果您沒有從`create-server`命令結果中擷取入門套件，您可以從 AWS OpsWorks for Chef Automate 主控台的伺服器屬性頁面下載新的入門套件。下載新的入門套件會重設 Chef Automate 儀表板管理員密碼。
7. 在企業的 DNS 管理工具中建立 CNAME 項目，將自訂網域指向您在步驟 4 中複製的 AWS OpsWorks for Chef Automate 端點。在完成此步驟之前，您無法連線或登入伺服器。
8. 完成伺服器建立程序之後，繼續the section called “完成組態並上傳技術指南”。

另請參閱

- 使用 建立 Chef Automate 伺服器 AWS CLI
- 從備份還原 AWS OpsWorks for Chef Automate 伺服器
- CM API AWS OpsWorks 參考中的 CreateServer
- 《AWS CLI 命令參考》中的 create-server 一節

重新產生 AWS OpsWorks for Chef Automate 伺服器的入門套件

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或替代解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

的入門套件 AWS OpsWorks for Chef Automate 包含 README 檔案，其中包含範例、`knife.rb`組態檔案，以及主要或樞紐使用者的私有金鑰。每次下載入門套件時，都會產生新的金鑰對，並重設舊

金鑰。您可以透過下列兩種方式之一，為 AWS OpsWorks for Chef Automate 伺服器重新產生入門套件：

- 在 AWS OpsWorks 主控台中，伺服器詳細資訊頁面的動作功能表上 AWS OpsWorks for Chef Automate 。系統會提示您確認是否要重新產生和重設舊的樞紐索引鍵。
- 透過在 中執行命令 AWS CLI。

如需如何使用入門套件的詳細資訊，請參閱使用入門套件設定 Chef 伺服器。

使用 重新產生 AWS OpsWorks for Chef Automate 入門套件 AWS CLI

> **Note**
>
> 當您重新產生入門套件時，您也可以為 Chef Automate 伺服器重新產生和重設身分驗證金鑰對，並刪除目前的金鑰對。

執行 update-server-engine-attributes命令來重新產生入門套件。在 AWS CLI 工作階段中，執行下列命令。將伺服器名稱指定為 的值`--server-name`。若要將您自己的公有金鑰設定為 的值`CHEF_AUTOMATE_PIVOTAL_KEY`，請在 中指定公有金鑰的值`--attribute-value`。否則，請將`--attribute-value` 設定為 null。

```
aws opsworks-cm update-server-engine-attributes \
   --server-name server_name \
   --attribute-name "CHEF_AUTOMATE_PIVOTAL_KEY" \
   --attribute-value your_public_key
```

下列命令是指定伺服器管理員想要使用的公有金鑰值的範例。

```
aws opsworks-cm update-server-engine-attributes \
   --server-name your-test-server \
   --attribute-name "CHEF_AUTOMATE_PIVOTAL_KEY" \
   --attribute-value "-----BEGIN PUBLIC KEY-----ExamplePublicKey-----END PUBLIC
 KEY-----"
```

下列命令是允許 AWS OpsWorks for Chef Automate 重新產生公有金鑰的範例。

```
aws opsworks-cm update-server-engine-attributes \
```

```
    --server-name your-test-server \
    --attribute-name "CHEF_AUTOMATE_PIVOTAL_KEY" \
    --attribute-value null
```

此命令的輸出是伺服器的相關資訊，以及 base64 編碼的 ZIP 檔案。ZIP 檔案包含 Chef 入門套件，其中包含 README、組態檔案和所需的 RSA 私有金鑰。儲存此檔案、解壓縮檔案，然後變更為您已解壓縮檔案內容的目錄。您可以從此目錄中執行`knife`命令。

在 AWS OpsWorks for Chef Automate 資源上使用標籤

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或替代解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

標籤是單字或片語，會以中繼資料形式用於識別和組織您的 AWS 資源。在 中 AWS OpsWorks for Chef Automate，資源最多可有 50 個使用者套用的標籤。每個標籤皆包含一個索引鍵與一個選用值。您可以將標籤套用至 AWS OpsWorks for Chef Automate中的下列資源：

- AWS OpsWorks for Chef Automate 伺服器
- 伺服器備份 AWS OpsWorks for Chef Automate

AWS 資源上的標籤可協助您追蹤成本、控制資源的存取、將任務自動化的群組資源，或依用途或生命週期階段組織資源。如需標籤優點的詳細資訊，請參閱AWS 帳單與成本管理 《 使用者指南》中的 AWS 答案中的 AWS 標記策略和使用成本分配標籤。

若要使用標籤控制對 AWS OpsWorks for Chef Automate 伺服器或備份的存取，您可以在 AWS Identity and Access Management (IAM) 中建立或編輯政策陳述式。如需詳細資訊，請參閱AWS Identity and Access Management 《 使用者指南》中的使用資源標籤控制 AWS 對資源的存取。

當您將標籤套用至 AWS OpsWorks for Chef Automate 伺服器時，標籤也會套用至伺服器的備份、存放備份的 Amazon S3 儲存貯體、伺服器的 Amazon EC2 執行個體、存放於 之伺服器的秘密 AWS Secrets Manager，以及伺服器使用的彈性 IP 地址。標籤不會傳播到 AWS OpsWorks 用於建立伺服器的 AWS CloudFormation 堆疊。

主題

- 標籤在 中的運作方式 AWS OpsWorks for Chef Automate
- 在 AWS OpsWorks for Chef Automate （主控台）中新增和管理標籤
- 在 AWS OpsWorks for Chef Automate (CLI) 中新增和管理標籤
- 另請參閱

標籤在 中的運作方式 AWS OpsWorks for Chef Automate

在此版本中，您可以使用 AWS OpsWorks CM API 或 AWS Management Console來新增和管理標籤。 AWS OpsWorks CM 也會嘗試將新增至伺服器的標籤新增至與伺服器相關聯的 AWS 資源，包括 EC2 執行個體、Secrets Manager 中的秘密、彈性 IP 地址、安全群組、S3 儲存貯體和備份。下表將概要介紹如何在 AWS OpsWorks for Chef Automate中新增和管理標籤。

動作	執行方式
將標籤新增至您要手動建立的新 AWS OpsWorks for Chef Automate 伺服器或備份。	• 選擇 Create Chef Automate server (建立 Chef Automate 伺服器)，然後在 Configure advanced settings (配置進階設定) 頁面上新增標籤。 • 在現有伺服器的備份頁面上選擇建立備份，然後在建立 Chef Automate 2 伺服器備份頁面上新增標籤。 • 將 `Tags` 參數新增到 `CreateServer` 或 `CreateBackup` 命令。
檢視資源上的標籤。	• 在伺服器的詳細資訊頁面上，選擇導覽窗格中的 Tags (標籤)。 • 在伺服器的 Backups (備份) 頁面上，選取備份，然後選擇 Edit backup (編輯備份)。 • 執行 `ListTagsForResource` 命令。
將標籤新增至現有 AWS OpsWorks for Chef Automate 伺服器或備份，無論備份是手動還是自動建立。	• 在伺服器的詳細資訊頁面上，選擇導覽窗格中的 Tags (標籤)，然後選擇 Edit (編輯)。 • 在伺服器的 Backups (備份) 頁面上，選取備份，然後選擇 Edit backup (編輯備份)。 • 執行 `TagResource` 命令。

動作	執行方式
從資源刪除標籤。	• 在伺服器的詳細資訊頁面上，選擇導覽窗格中的 Tags (標籤)，然後選擇 Edit (編輯)。選擇您要刪除的標籤旁的 X。 • 在伺服器的 Backups (備份) 頁面上，選取備份，然後選擇 Edit backup (編輯備份)。選擇您要刪除的標籤旁的 X。 • 執行 UntagResource 命令。

DescribeServers 和 DescribeBackups 回應不包含標籤資訊。若要顯示標籤，請使用 ListTagsForResource API。

在 AWS OpsWorks for Chef Automate （主控台）中新增和管理標籤

本節中的程序將在 AWS Management Console中執行。

新增標籤時，索引鍵不能為空。此索引鍵最多可包含 127 個字元，並且只能包含 Unicode 字母、數字或分隔符號，或是下列特殊字元：+ - = . _ : / @。標籤值是選擇性的。您可以新增具有索引鍵但沒有值的標籤。值最多可包含 255 個字元，並且只能包含 Unicode 字母、數字或分隔符號，或是下列特殊字元：+ - = . _ : / @

主題

- 將標籤新增至新 AWS OpsWorks for Chef Automate 伺服器（主控台）
- 新增標籤至新的備份 (主控台)
- 新增或檢視現有伺服器上的標籤 (主控台)
- 新增或檢視現有備份上的標籤 (主控台)
- 從伺服器刪除標籤 (主控台)
- 從備份刪除標籤 (主控台)

將標籤新增至新 AWS OpsWorks for Chef Automate 伺服器（主控台）

1. 請務必完成建立 AWS OpsWorks for Chef Automate 伺服器的任何先決條件。
2. 依照 建立 Chef Automate 伺服器 中的步驟 1-10 執行。

3. 指定自動備份設定後，請在設定進階設定頁面的標籤區域中新增標籤。您最多可新增 50 個標籤。完成標籤新增作業時，選擇 Next (下一步)。
4. 繼續進行 建立 Chef Automate 伺服器 的步驟 13，並檢閱您為新伺服器選擇的設定。

新增標籤至新的備份 (主控台)

1. 在 AWS OpsWorks for Chef Automate 首頁上，選擇現有的 Chef Automate 伺服器。
2. 在伺服器的詳細資訊頁面中，選擇導覽窗格中的 Backups (備份)。
3. 在 Backups (備份) 頁面上，選擇 Create backup (建立備份)。
4. 新增標籤。完成新增標籤作業時，選擇 Create (建立)。

新增或檢視現有伺服器上的標籤 (主控台)

1. 在 AWS OpsWorks for Chef Automate 首頁上，選擇現有的 Chef Automate 伺服器以開啟其詳細資訊頁面。
2. 在導覽窗格中選擇 Tags (標籤)，或在詳細資訊頁面底部選擇 View all tags (檢視所有標籤)。
3. 在 Tags (標籤) 頁面上，選擇 Edit (編輯)。
4. 新增或編輯伺服器上的標籤。完成時，請選擇 Save (儲存)。

> **Note**
>
> 請注意，變更 Chef Automate 伺服器上的標籤也會變更與該伺服器相關聯的資源 (例如 EC2 執行個體、彈性 IP 地址、安全群組、S3 儲存貯體和備份) 上的標籤。

新增或檢視現有備份上的標籤 (主控台)

1. 在 AWS OpsWorks for Chef Automate 首頁上，選擇現有的 Chef Automate 伺服器以開啟其詳細資訊頁面。
2. 在導覽窗格中選擇 Backups (備份)，或在詳細資訊頁面的 Recent backups (最近備份) 區域中，選擇 View all backups (檢視所有備份)。
3. 在 Backups (備份) 頁面上，選擇要管理的備份，然後選擇 Edit backup (編輯備份)。
4. 新增或編輯備份上的標籤。完成後，選擇 Update (更新)。

從伺服器刪除標籤 (主控台)

1. 在 AWS OpsWorks for Chef Automate 首頁上，選擇現有的 Chef Automate 伺服器以開啟其詳細資訊頁面。
2. 在導覽窗格中選擇 Tags (標籤)，或在詳細資訊頁面底部選擇 View all tags (檢視所有標籤)。
3. 在 Tags (標籤) 頁面上，選擇 Edit (編輯)。
4. 選擇標籤旁邊的 X 以刪除標籤。完成時，請選擇 Save (儲存)。

> **Note**
>
> 請注意，變更 Chef Automate 伺服器上的標籤也會變更與該伺服器相關聯的資源 (例如 EC2 執行個體、彈性 IP 地址、安全群組、S3 儲存貯體和備份) 上的標籤。

從備份刪除標籤 (主控台)

1. 在 AWS OpsWorks for Chef Automate 首頁上，選擇現有的 Chef Automate 伺服器以開啟其詳細資訊頁面。
2. 在導覽窗格中選擇 Backups (備份)，或在詳細資訊頁面的 Recent backups (最近備份) 區域中，選擇 View all backups (檢視所有備份)。
3. 在 Backups (備份) 頁面上，選擇要管理的備份，然後選擇 Edit backup (編輯備份)。
4. 選擇標籤旁邊的 X 以刪除標籤。完成後，選擇 Update (更新)。

在 AWS OpsWorks for Chef Automate (CLI) 中新增和管理標籤

本節中的程序將在 AWS CLI中執行。開始處理標籤 AWS CLI 之前，請務必執行最新版本的 。如需安裝或更新 的詳細資訊 AWS CLI，請參閱AWS Command Line Interface 《 使用者指南》中的安裝 AWS CLI。

新增標籤時，索引鍵不能為空。此索引鍵最多可包含 127 個字元，並且只能包含 Unicode 字母、數字或分隔符號，或是下列特殊字元：`+ - = . _ : / @`。標籤值是選擇性的。您可以新增具有索引鍵但沒有值的標籤。值最多可包含 255 個字元，並且只能包含 Unicode 字母、數字或分隔符號，或是下列特殊字元：`+ - = . _ : / @`

主題

- 將標籤新增至新 AWS OpsWorks for Chef Automate 伺服器 (CLI)

- 新增標籤至新的備份 (CLI)
- 新增標籤至現有伺服器或備份 (CLI)
- 列出資源標籤
- 從資源刪除標籤

將標籤新增至新 AWS OpsWorks for Chef Automate 伺服器 (CLI)

您可以在建立 AWS OpsWorks for Chef Automate 伺服器時使用 AWS CLI 新增標籤。此程序並未完整說明如何建立伺服器。如需如何使用本指南使用 建立 Chef Automate 伺服器 AWS CLI中的 建立 AWS OpsWorks for Chef Automate 伺服器的詳細資訊， AWS CLI 請參閱 。您最多可以為每個伺服器新增 50 個標籤。

1. 請務必完成建立 AWS OpsWorks for Chef Automate 伺服器的任何先決條件。
2. 完成 使用 建立 Chef Automate 伺服器 AWS CLI 的步驟 1-5。
3. 在步驟 6 時，執行 `create-server` 命令時請將 `--tags` 參數新增至命令，如下列範例所示。

   ```
   aws opsworks-cm create-server ... --tags Key=Key1,Value=Value1
    Key=Key2,Value=Value2
   ```

   以下示範僅顯示 `create-server` 命令的標籤部分。

   ```
   aws opsworks-cm create-server ... --tags Key=Stage,Value=Production
    Key=Department,Value=Marketing
   ```

4. 完成使用 建立 Chef Automate 伺服器 AWS CLI 中剩餘的步驟。若要確認您的標籤是否已新增至新的伺服器，請遵循本主題 列出資源標籤 中的步驟。

新增標籤至新的備份 (CLI)

當您建立新的 手動備份伺服器 AWS OpsWorks for Chef Automate 時，您可以使用 AWS CLI 新增標籤。此程序並未完整說明如何建立手動備份。如需如何建立手動備份的詳細資訊，請參閱 中的「在 中執行手動備份 AWS CLI」備份 AWS OpsWorks for Chef Automate 伺服器。您最多可以為備份新增 50 個標籤。如果伺服器有標籤，則新的備份會自動加上伺服器的標籤。

根據預設，當您建立新的 AWS OpsWorks for Chef Automate 伺服器時，會啟用自動備份。您可以依本主題中新增標籤至現有伺服器或備份 (CLI)所述，透過執行 `tag-resource` 命令，將標籤新增至自動備份。

- 若要在建立備份時將標籤新增至手動備份，請執行下列命令。僅展示命令的標籤部分。如需完整create-backup命令的範例，請參閱「若要在「 AWS CLI的「中執行手動備份備份 AWS OpsWorks for Chef Automate 伺服器。

```
aws opsworks-cm create-backup ... --tags Key=Key1,Value=Value1
 Key=Key2,Value=Value2
```

以下示範僅顯示 create-backup 命令的標籤部分。

```
aws opsworks-cm create-backup ... --tags Key=Stage,Value=Production
 Key=Department,Value=Marketing
```

新增標籤至現有伺服器或備份 (CLI)

您可以執行 tag-resource 命令，將標籤新增至現有的 AWS OpsWorks for Chef Automate 伺服器或備份 (無論備份是自動或手動建立)。指定要在其中新增標籤之目標資源的 Amazon 資源編號 (ARN)。

1. 若要取得標籤將套用其中之資源的 ARN：

   - 用於伺服器時，請執行 describe-servers --server-name *server_name*。此命令執行後結果是顯示伺服器 ARN。
   - 用於備份時，請執行 describe-backups --backup-id *backup_ID*。此命令執行後結果是顯示備份 ARN。您也可以執行 describe-backups --server-name *server_name* 以顯示特定 AWS OpsWorks for Chef Automate 伺服器所有備份的相關資訊。

   以下示範僅顯示 describe-servers --server-name opsworks-cm-test 命令結果的 ServerArn。此 ServerArn 值會新增至 tag-resource 命令，以便將標籤新增至伺服器。

```
{
    "Servers": [
        {
            ...
            "ServerArn": "arn:aws:opsworks-cm:us-west-2:123456789012:server/
opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
        }
    ]
}
```

2. 使用您在步驟 1 中傳回的 ARN 執行 `tag-resource` 命令。

```
aws opsworks-cm tag-resource --resource-arn "server_or_backup_ARN" --tags
 Key=Key1,Value=Value1 Key=Key2,Value=Value2
```

以下是範例。

```
aws opsworks-cm tag-resource --resource-arn "arn:aws:opsworks-cm:us-
west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
 --tags Key=Stage,Value=Production Key=Department,Value=Marketing
```

3. 若要確認標籤是否已成功新增，請繼續進行下一個程序 列出資源標籤。

列出資源標籤

您可以執行 `list-tags-for-resource`命令來顯示連接到 AWS OpsWorks for Chef Automate 伺服器或備份的標籤。指定要檢視其中標籤之目標資源的 ARN。

1. 若要取得將要列出其中標籤之資源的 ARN：

   - 用於伺服器時，請執行 `describe-servers --server-name server_name`。此命令執行後結果是顯示伺服器 ARN。
   - 用於備份時，請執行 `describe-backups --backup-id backup_ID`。此命令執行後結果是顯示備份 ARN。您也可以執行 `describe-backups --server-name server_name` 以顯示特定 AWS OpsWorks for Chef Automate 伺服器所有備份的相關資訊。

2. 使用您在步驟 1 中傳回的 ARN 執行 `list-tags-for-resource` 命令。

```
aws opsworks-cm list-tags-for-resource --resource-arn "server_or_backup_ARN"
```

以下是範例。

```
aws opsworks-cm tag-resource --resource-arn "arn:aws:opsworks-cm:us-
west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
```

如果資源上有標籤，此命令結果將會傳回如下。

```
{
    "Tags": [
```

```
        {
            "Key": "Stage",
            "Value": "Production"
        },
        {
            "Key": "Department",
            "Value": "Marketing"
        }
    ]
}
```

從資源刪除標籤

您可以執行 `untag-resource` 命令，從 AWS OpsWorks for Chef Automate 伺服器或備份刪除標籤。如果資源遭到刪除，資源上的標籤也會遭到刪除。指定要從其中移除標籤之目標資源的 Amazon 資源編號 (ARN)。

1. 若要取得將要移除其中標籤之資源的 ARN：

   - 用於伺服器時，請執行 `describe-servers --server-name` *`server_name`*。此命令執行後結果是顯示伺服器 ARN。
   - 用於備份時，請執行 `describe-backups --backup-id` *`backup_ID`*。此命令執行後結果是顯示備份 ARN。您也可以執行 `describe-backups --server-name` *`server_name`* 以顯示特定 AWS OpsWorks for Chef Automate 伺服器所有備份的相關資訊。

2. 使用您在步驟 1 中傳回的 ARN 執行 `untag-resource` 命令。僅指定您要刪除的標籤。

   ```
   aws opsworks-cm untag-resource --resource-arn "server_or_backup_ARN" --tags
    Key=Key1,Value=Value1 Key=Key2,Value=Value2
   ```

   在此範例中，`untag-resource` 命令只會移除索引鍵為 `Stage`、和值為 `Production` 的標籤。

   ```
   aws opsworks-cm untag-resource --resource-arn "arn:aws:opsworks-cm:us-
   west-2:123456789012:server/opsworks-cm-test/EXAMPLEd-66b0-4196-8274-d1a2bEXAMPLE"
    --tags Key=Stage,Value=Production
   ```

3. 若要確認已成功刪除標籤，請遵循本主題 列出資源標籤 中的步驟。

另請參閱

- 使用 建立 Chef Automate 伺服器 AWS CLI
- 備份 AWS OpsWorks for Chef Automate 伺服器
- AWS 加標籤策略
- AWS Identity and Access Management 使用者指南中的使用資源標籤控制 AWS 對資源的存取
- 《AWS 帳單與成本管理 使用者指南》中的使用成本分配標籤
- AWS OpsWorks CM API 參考中的 CreateBackup
- AWS OpsWorks CM API 參考中的 CreateServer
- AWS OpsWorks CM API 參考中的 TagResource
- AWS OpsWorks CM API 參考中的 ListTagsForResource
- AWS OpsWorks CM API 參考中的 UntagResource

備份和還原 AWS OpsWorks for Chef Automate 伺服器

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日終止服務，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或替代解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節說明如何備份和還原 AWS OpsWorks for Chef Automate 伺服器，以及如何刪除備份。

主題

- 備份 AWS OpsWorks for Chef Automate 伺服器
- 從備份還原 AWS OpsWorks for Chef Automate 伺服器

備份 AWS OpsWorks for Chef Automate 伺服器

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日終止服務，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或替代解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以定義每日或每週定期 AWS OpsWorks for Chef Automate 伺服器備份，並讓服務代您將備份存放在 Amazon Simple Storage Service (Amazon S3) 中。或者，您可以隨需手動備份。

由於備份存放在 Amazon S3 中，因此會產生額外費用。您可以定義備份保留期，最長為 30 代。您可以使用 AWS 支援管道提交服務請求，以變更該限制。傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

您可以將標籤新增至 AWS OpsWorks for Chef Automate 伺服器的備份。如果您已將標籤新增至 AWS OpsWorks for Chef Automate 伺服器，該伺服器的自動備份就會繼承這些標籤。如需有關如何新增和管理備份標籤的詳細資訊，請參閱本指南中的 在 AWS OpsWorks for Chef Automate 資源上使用標籤。

主題

- 自動備份
- 手動備份
- 刪除備份

自動備份

當您設定 AWS OpsWorks for Chef Automate 伺服器時，您可以選擇自動或手動備份。 會在設定進階設定頁面的自動備份區段中選擇一小時和當天 AWS OpsWorks for Chef Automate 啟動自動備份。當您的伺服器處於線上狀態之後，您即可從 Chef Automate 伺服器首頁的伺服器圖磚，或在伺服器的 Properties (屬性) 頁面中執行下列步驟，以變更備份設定。

變更自動備份設定

1. 在 Chef servers (Chef 伺服器) 首頁伺服器圖磚的 Actions (動作) 功能表中，選擇 Change settings (變更設定)

2. 若要關閉自動備份，請針對 Enable automated backups (啟用自動備份) 選項選擇 No (否)。儲存變更；您不需要繼續進行下一個步驟。
3. 在 Automated Backup (自動備份) 區段中，變更頻率、開始時間或要保留的版本。儲存您的變更。

手動備份

您可以隨時在 中啟動手動備份 AWS Management Console，或執行 AWS CLI create-backup 命令。手動備份不包含在儲存的最多 30 代自動備份中；最多儲存 10 個手動備份，而且必須從 Amazon S3 手動刪除。

您可以在建立新的 AWS OpsWorks for Chef Automate 伺服器的手動備份時新增標籤。如需如何在建立手動備份時新增標籤的詳細資訊，請參閱新增標籤至新的備份 (CLI) 。

在 中執行手動備份 AWS Management Console

1. 在 Chef Automate servers (Chef Automate 伺服器) 頁面中，選擇您要備份的伺服器。
2. 在伺服器屬性頁面的左側導覽窗格中，選擇 Backups (備份)。
3. 選擇 Create backup (建立備份)。
4. 當頁面的備份 Status (狀態) 欄中顯示綠色核取記號時，手動備份即已完成。

在 中執行手動備份 AWS CLI

- 若要啟動手動備份，請執行下列 AWS CLI 命令。

```
aws opsworks-cm --region region name create-backup --server-name "Chef server name"
 --description "optional descriptive string"
```

刪除備份

永久刪除備份時，即會將該備份從存放備份的 S3 儲存貯體中刪除。

在 中刪除備份 AWS Management Console

1. 在 Chef Automate servers (Chef Automate 伺服器) 頁面中，選擇您要備份的伺服器。
2. 在伺服器屬性頁面的左側導覽窗格中，選擇 Backups (備份)。
3. 選擇您要刪除的備份，然後選擇 Delete backup (刪除備份)。您一次只能選取一個備份。

4. 當系統提示您確認刪除時，請勾選 Delete the backup, which is stored in an S3 bucket (刪除存放在 S3 儲存貯體中的備份) 核取方塊，然後選擇 Yes, Delete (是，刪除)。

在 中刪除備份 AWS CLI

- 若要刪除備份，請執行下列 AWS CLI 命令，`--backup-id`以您要刪除的備份 ID 取代 。備份 ID 的格式為 *`ServerName-yyyyMMddHHmmssSSS`*。例如 **`test-chef-server-20171218132604388`**。

```
aws opsworks-cm --region region name delete-backup --backup-id ServerName-
yyyyMMddHHmmssSSS
```

從備份還原 AWS OpsWorks for Chef Automate 伺服器

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日終止服務，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或替代解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

瀏覽可用的備份後，您可以選擇從中還原 AWS OpsWorks for Chef Automate 伺服器的時間點。伺服器備份僅包含組態管理軟體持久性資料 (技術指南、註冊的節點等)。執行伺服器就地還原 （ 也就是將現有 AWS OpsWorks for Chef Automate 伺服器還原至新的 EC2 執行個體 ） 會重新註冊您在用來還原伺服器之備份時註冊的節點，並在還原成功且還原 AWS OpsWorks for Chef Automate 的伺服器狀態為 時將流量切換到新執行個體`Healthy`。還原到新建立的 AWS OpsWorks for Chef Automate 伺服器時，不會保留節點連線。還原伺服器時並不會更新 Chef 軟體的次要版本；其會套用您所選備份中可用的相同 Chef 版本和組態管理資料。

還原伺服器通常需要比建立新伺服器更長的時間；時間取決於您選擇的備份大小。還原完成後，舊的 EC2 執行個體會保留在 `Running` 或 `Stopped` 狀態，但只是暫時狀態。這個狀態最終將會結束。

在此版本中，您可以使用 AWS CLI 在 中還原 Chef 伺服器 AWS OpsWorks for Chef Automate。

> **Note**
>
> 您也可以執行 restore-server 命令，以變更目前的執行個體類型；或者，還原或設定您的 SSH 金鑰 (如果遺失或受損的話)。

從備份還原伺服器

1. 在 中 AWS CLI，執行下列命令，以傳回可用備份及其 IDs的清單。請記下您要使用的備份 ID。備份 ID 的格式為 *myServerName-yyyyMMddHHmmssSSS*。

   ```
   aws opsworks-cm --region region name describe-backups
   ```

2. 執行下列命令。

   ```
   aws opsworks-cm --region region name restore-server --backup-id "myServerName-
   yyyyMMddHHmmssSSS" --instance-type "Type of instance" --key-pair "name of your EC2
    key pair" --server-name "name of Chef server"
   ```

   以下是範例。

   ```
   aws opsworks-cm --region us-west-2 restore-server --backup-id
    "MyChefServer-20161120122143125" --server-name "MyChefServer"
   ```

3. 等待還原完成。

中的系統維護 AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或替代解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

強制性系統維護可確保 Chef Server 和 Chef Automate Server 的最新次要版本，包括安全性更新，一律在 AWS OpsWorks for Chef Automate 伺服器上執行。每週至少需要進行一次系統維護。如有需要

AWS CLI，您可以使用 設定每日自動維護。除了排定的系統維護之外，您也可以使用 隨需 AWS CLI 執行系統維護。

當有新的 Chef 軟體次要版本可用時，系統維護的設計會在其通過 AWS 測試之後，自動更新伺服器上 Chef Automate 和 Chef Server 的次要版本。AWS 會執行廣泛的測試，驗證 Chef 升級已生產就緒，而不會中斷現有的客戶環境，因此在 Chef 軟體發行後和可供現有 OpsWorks for Chef Automate 伺服器套用之間，可能會有延遲。若要隨需更新 Chef 軟體的次要版本，請參閱本主題中的隨需啟動系統維護。

系統維護會從作為維護程序一部分執行的備份啟動新的執行個體，這有助於降低因定期維護而降級或受損的 Amazon EC2 執行個體所造成的風險。

> **Important**
>
> 系統維護會刪除您新增至 AWS OpsWorks for Chef Automate 伺服器的任何檔案或自訂組態。如需如何修復組態或檔案遺失的詳細資訊，請參閱本主題中的在維護之後還原自訂組態和檔案。

主題

- 確保節點信任 AWS OpsWorks 憑證授權機構
- 設定系統維護
- 隨需啟動系統維護
- 在維護之後還原自訂組態和檔案

確保節點信任 AWS OpsWorks 憑證授權機構

> **Note**
>
> 如果您搭配 AWS OpsWorks for Chef Automate 伺服器使用自訂網域和憑證，則不需要本節中的步驟。

您使用 AWS OpsWorks for Chef Automate 伺服器管理的節點必須使用憑證向伺服器進行身分驗證。在系統維護期間， 會 AWS OpsWorks 取代伺服器執行個體，並透過憑證授權機構 (CA) 來重新產生新 AWS OpsWorks 憑證。若要在維護完成後自動還原與受管節點的通訊，節點應該信任隨附入門套件的

AWS OpsWorks CA，並且託管在 支援的區域中 AWS OpsWorks for Chef Automate。當您使用 AWS OpsWorks CA 建立節點與伺服器之間的信任時，節點會在維護後重新連線至新的伺服器執行個體。如果您使用中所述的 EC2 `userdata`指令碼來新增 EC2 節點在 中自動新增節點 AWS OpsWorks for Chef Automate，節點已設定為信任 AWS OpsWorks CA。

- 針對 Linux 式的節點，CA 的 S3 儲存貯體位置為 `https://opsworks-cm-${REGION}-prod-default-assets.s3.amazonaws.com/misc/opsworks-cm-ca-2020-root.pem`。 AWS OpsWorks 信任的 CA 必須存放在路徑 中`/etc/chef/opsworks-cm-ca-2020-root.pem`。
- 針對 Windows 式的節點，CA 的 S3 儲存貯體位置為 `https://opsworks-cm-$env:AWS_REGION-prod-default-assets.s3.amazonaws.com/misc/opsworks-cm-ca-2020-root.pem`。 AWS OpsWorks CA 必須存放在根 Chef 資料夾中；例如， `C:\chef\opsworks-cm-ca-2020-root.pem`

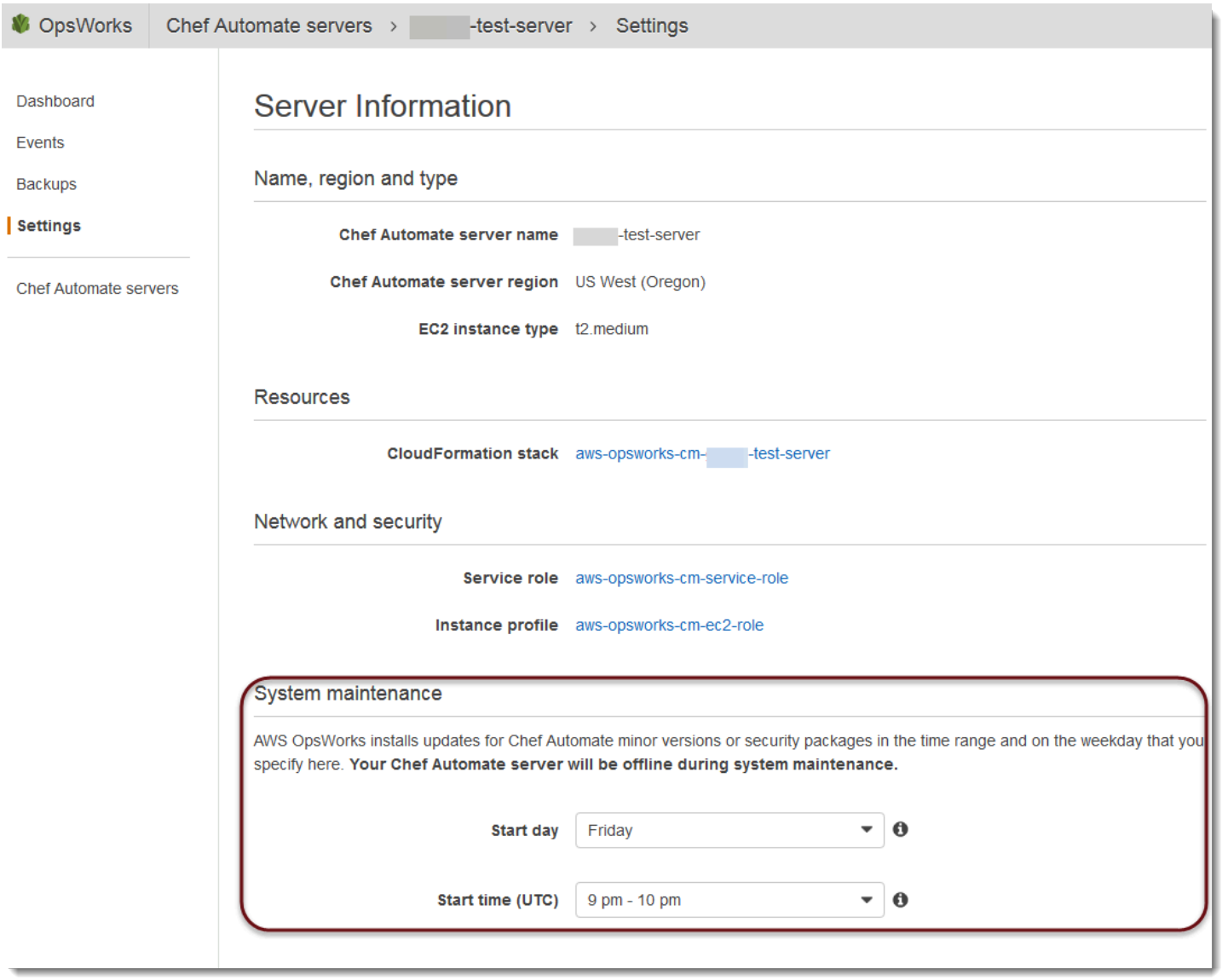
在這兩種路徑中，區域變數會解析為下列其中一項。

- `us-east-2`
- `us-east-1`
- `us-west-1`
- `us-west-2`
- `ap-northeast-1`
- `ap-southeast-1`
- `ap-southeast-2`
- `eu-central-1`
- `eu-west-1`

設定系統維護

當您建立新的 AWS OpsWorks for Chef Automate 伺服器時，您可以在國際標準時間 (UTC) 中設定工作日和時間，以啟動系統維護。維護會在您指定的小時啟動。因為您應預期伺服器在系統維護期間離線，請選擇一般工作時間內伺服器需求較低的時間。當維護正在進行中時，伺服器狀態將為 `UNDER_MAINTENANCE`。

您也可以變更現有 AWS OpsWorks for Chef Automate 伺服器上的系統維護設定，方法是變更伺服器設定頁面之系統維護區域中的設定，如下列螢幕擷取畫面所示。



在 System maintenance (系統維護) 區段中，將日和小時設為您希望開始系統維護的時間。

使用 設定系統維護 AWS CLI

您也可以使用 AWS CLI設定系統維護的自動開始時間。 AWS CLI 可視需要省略三字元工作日字首，以設定每日自動維護。

在 `create-server` 命令中，在指定建立伺服器執行個體的必要項目 (例如執行個體類型、執行個體描述檔 ARN 和伺服器角色 ARN) 之後，將 `--preferred-maintenance-window` 參數新增到您的命令。在下列 `create-server` 範例中，`--preferred-maintenance-window` 設為 `Mon:08:00`，表示您已將維護設定在每週一 UTC 上午 8:00 開始 data at rest。

```
aws opsworks-cm create-server --engine "Chef" --engine-model "Single" --
engine-version "12" --server-name "automate-06" --instance-profile-arn
 "arn:aws:iam::1019881987024:instance-profile/aws-opsworks-cm-ec2-role"
 --instance-type "t2.medium" --key-pair "amazon-test" --service-role-arn
 "arn:aws:iam::044726508045:role/aws-opsworks-cm-service-role" --preferred-maintenance-
window "Mon:08:00"
```

在 update-server 命令中，您可以單獨更新 --preferred-maintenance-window 的值 (若需要的話)。在下列範例中，維護時段已設為週五 UTC 下午 6:15。data at rest。

```
aws opsworks-cm update-server --server-name "shiny-kitchen" --preferred-maintenance-
window "Fri:18:15"
```

若要將維護時段的開始時間變更為每天 UTC 下午 6:15，請省略三個字元的工作日字首，如以下範例所示。

```
aws opsworks-cm update-server --server-name "shiny-kitchen" --preferred-maintenance-
window "18:15"
```

如需使用 設定偏好系統維護時段的詳細資訊 AWS CLI，請參閱 create-server 和 update-server。

隨需啟動系統維護

若要隨需開始系統維護，在您設定的每週或每日自動維護之外，請執行下列 AWS CLI 命令。您無法在 AWS Management Console中啟動隨需維護。

```
aws opsworks-cm start-maintenance --server-name server_name
```

如需此命令的詳細資訊，請參閱 start-maintenance。

在維護之後還原自訂組態和檔案

系統維護可以刪除或變更您已新增至 AWS OpsWorks for Chef Automate 伺服器的自訂檔案或組態。

如果在維護執行後，您的 Chef 伺服器缺少您使用 或 SSH 新增的檔案RunCommand或設定，您可以使用 Amazon Machine Image (AMI) 來啟動新的 Amazon EC2 執行個體。可用的 AMI 是由伺服器的維護前組態建置而成的。

新的執行個體狀態會和維護前的 Chef 伺服器相同，因此會包含您遺失的檔案和設定。

> **Important**
>
> 您無法使用新的執行個體還原您的伺服器。執行個體無法做為 Chef 伺服器執行。您只能使用執行個體復原您的檔案和組態設定。

若要從 AMI 啟動 EC2 執行個體，請在 Amazon EC2 主控台中開啟啟動精靈，選擇我的 AMIs，然後選擇具有伺服器名稱的 AMI。請依照任何其他執行個體啟動的作法，遵循 Amazon EC2 精靈步驟。

中的合規掃描 AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或替代解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

合規掃描可讓您根據預先定義的政策 (也稱為規則)，追蹤基礎設施中受管節點的合規狀況。合規檢視可讓您定期稽核應用程式是否有漏洞與不合規的組態。Chef 提供百種以上的預先定義合規描述檔，其為適用於特定節點組態的規則集合，可讓您用於合規掃描。您也可以使用 Chef InSpec 語言來建立自己的自訂描述檔。

如果您的伺服器尚未執行 Chef Automate 2.0，則您可以手動安裝 Audit 技術指南來設定 Chef Compliance。

> **Note**
>
> 與 AWS OpsWorks for Chef Automate 伺服器相關聯的節點上 Chef Infra 用戶端代理程式軟體 (`chef-client`) 支援的最低版本為 13.x。我們建議執行最新、穩定的`chef-client`版本，或至少 14.10.9。

主題

- 使用 Chef Automate 2.0 中的合規
- 使用 Chef Automate 1.x 中的合規

- 合規更新
- 社群和自訂合規描述檔
- 另請參閱

使用 Chef Automate 2.0 中的合規

如果您的 AWS OpsWorks for Chef Automate 伺服器正在執行 Chef Automate 2.0，請使用本節中的程序來設定 Chef Compliance。

使用 Chef Automate 2.0 來執行合規掃描任務

Chef Automate 2.0 包含 Chef InSpec 合規掃描功能，這項功能先前稱為必要的手動設定和技術指南組態。您可以在執行 Chef Automate 2.0 的 AWS OpsWorks for Chef Automate 伺服器上執行掃描任務。任務可能立即執行 (一次性)、排定稍後執行或是排定於指定時間間隔執行，例如每日一次或每兩小時一次。掃描任務的結果會傳送到合規報告。您可以在 Chef Automate 2.0 儀表板中檢視合規掃描結果，並採取因應動作。若要開啟 Compliance (合規) 標籤和檢視報告，請在 Chef Automate 儀表板上的 Scan Jobs (掃描任務) 標籤中，在受管節點列的右邊選擇 Report (報告)。

若要在受管節點上執行掃描任務，您必須具備下列條件。

- 您的命名空間至少已安裝一個合規描述檔。
- 至少一個目標節點，可透過手動新增或 EC2 執行個體自動新增。

在 中 AWS OpsWorks for Chef Automate，下列目標支援掃描任務。

- 手動新增的節點
- `aws-ec2` 執行個體
- AWS 區域

如需如何執行掃描任務的相關詳細資訊，請參閱 Chef 文件中的 Chef Automate 掃描任務。

(選用，Chef Automate 2.0) 搭配 Audit 技術指南來設定合規

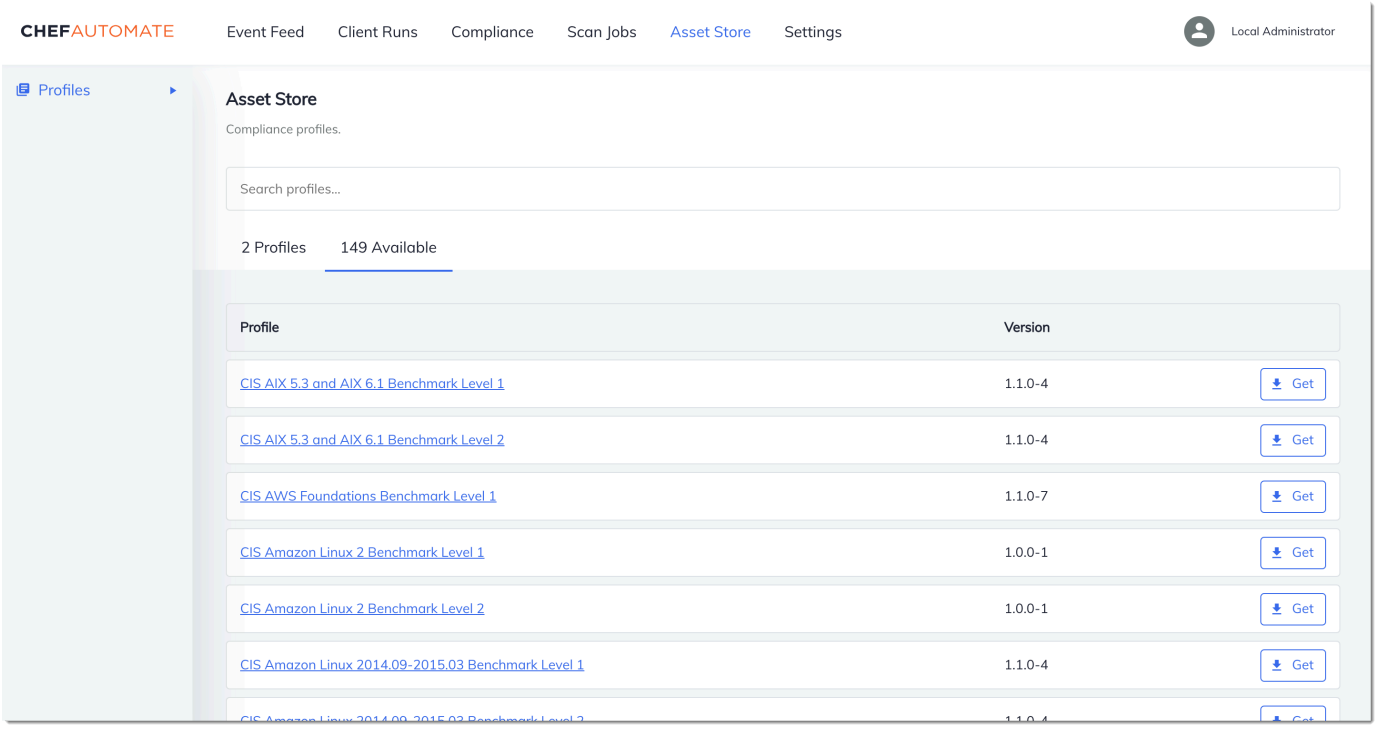
您可以在任何 AWS OpsWorks for Chef Automate 伺服器上設定合規。在啟動 AWS OpsWorks for Chef Automate 伺服器之後，您可以從 Chef Automate 儀表板上安裝描述檔，或是將所需的描述檔新增至 `Policyfile.rb` 政策檔案中的 Audit 技術指南屬性。入門套件中包含了預先填入的 `Policyfile.rb` 檔案。

在您透過描述檔編輯 `Policyfile.rb` 做為 Audit 技術指南屬性之後，請執行 `chef push` 命令，將 Audit 技術指南和 `Policyfile.rb` 中所指定的其他技術指南，上傳至您的 Chef Automate 伺服器。安裝 Audit 技術指南時，也會安裝 Chef InSpec 的 Gem，這是由 Chef 產生的開放原始碼測試和稽核架構。用於 Chef Automate 2.0 時，選擇 Audit 技術指南的 7.1.0 或更新版本。InSpec Gem 套件必須為 2.2.102 或更新版本。

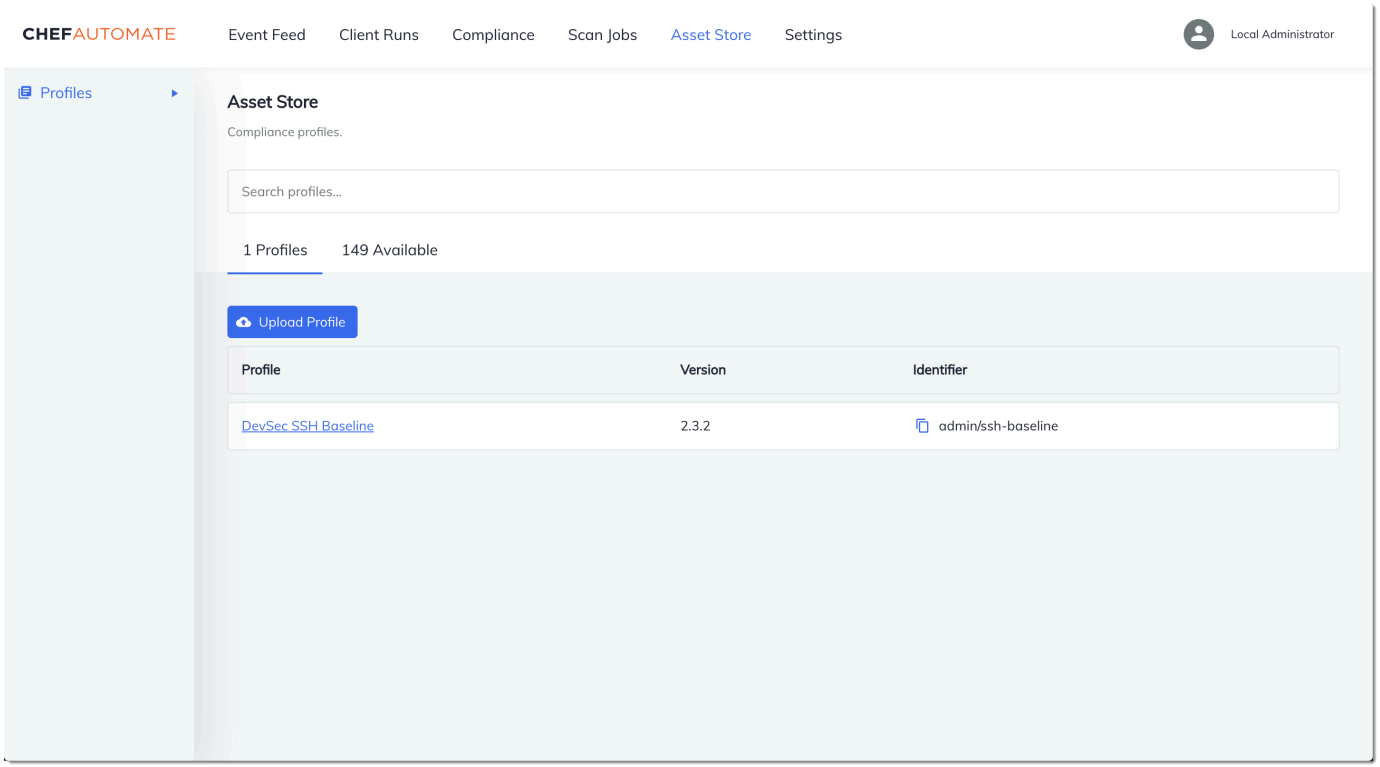
本節中的指示會說明如何實作 `opsworks-audit` 技術指南。Audit 技術指南會從 Chef Automate 伺服器下載指定的描述檔，針對 DevSec SSH Baseline (DevSec SSH 基準) 描述檔評估節點，並且報告每個 `chef-client` 執行的合規掃描結果。

安裝合規描述檔

1. 如果您尚未這麼做，請登入 Chef Automate Web 儀表板。請使用您在建立 AWS OpsWorks for Chef Automate 伺服器期間下載入門套件時收到的相同登入資料。
2. 在 Chef Automate 儀表板中，選擇 Asset Store (資產存放區) 標籤。



3. 選擇 Available (可用) 標籤以查看預先定義的描述檔。
4. 瀏覽描述檔清單。選擇至少符合其中一個受管節點之作業系統和組態的描述檔。若要查看描述檔的詳細資訊，包括描述檔的目標違反情況描述和基本規則程式碼，請選擇描述檔項目右側的 >。您可以選擇多個描述檔。如果您正在設定入門套件中的範例，請選擇 DevSec SSH Baseline (DevSec SSH 基準)。



5. 若要在您的 Chef Automate 伺服器上安裝選取的描述檔，請選擇 Get (取得)。
6. 安裝描述檔後，他們會顯示在 Profiles (描述檔) 標籤的 Chef Automate 儀表板。

搭配 **Policyfile.rb** 安裝技術指南

1. 查看入門套件 `Policyfile.rb` 中的屬性，以查看 Audit 技術指南在 `['profiles']` 中指定 `ssh-baseline` 描述檔。

```
# Define audit cookbook attributes
default["opsworks-demo"]["audit"]["reporter"] = "chef-server-automate"
default["opsworks-demo"]["audit"]["profiles"] = [
  {
    "name": "DevSec SSH Baseline",
    "compliance": "admin/ssh-baseline"
  }
]
```

2. 下載並安裝 `Policyfile.rb` 中所定義的技術指南。

```
chef install
```

所有技術指南的版本都是經由技術指南的 `metadata.rb` 檔案所控制。每次變更技術指南時，您都必須在技術指南的 `metadata.rb` 中提高其版本。

3. 將 `Policyfile.rb` 中所定義的政策 `opsworks-demo` 推送到您的伺服器。

```
chef push opsworks-demo
```

4. 驗證您的政策安裝情況。執行下列命令。

```
chef show-policy
```

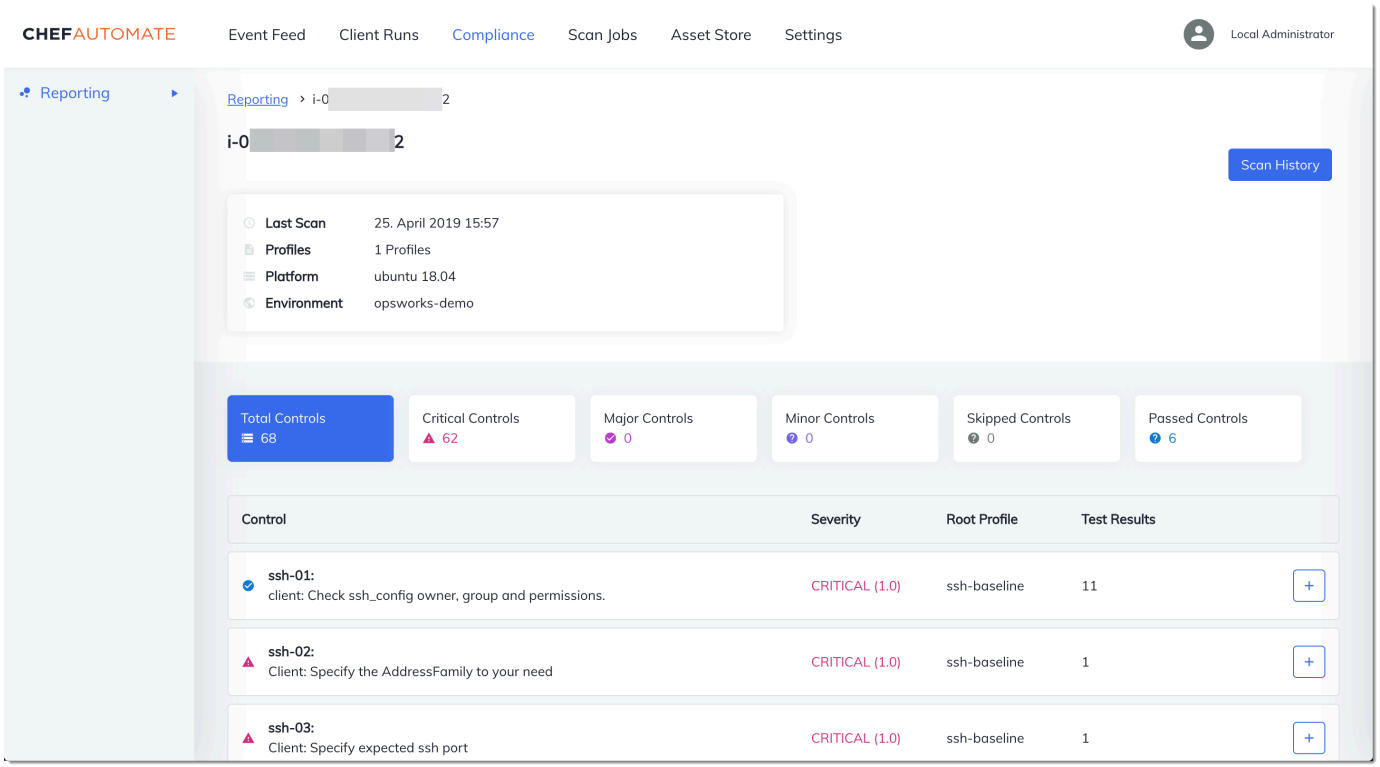
結果應類似以下內容：

```
opsworks-demo-webserver
=======================
* opsworks-demo:  ec0fe46314
```

5. 如果您尚未完成此作業，請先將節點新增到您的伺服器，以便管理。若要將第一個節點連接到 AWS OpsWorks for Chef Automate 伺服器，請使用此入門套件中包含的`userdata.sh`指令碼。它使用 AWS OpsWorks `AssociateNode` API 將節點連接到您的伺服器。

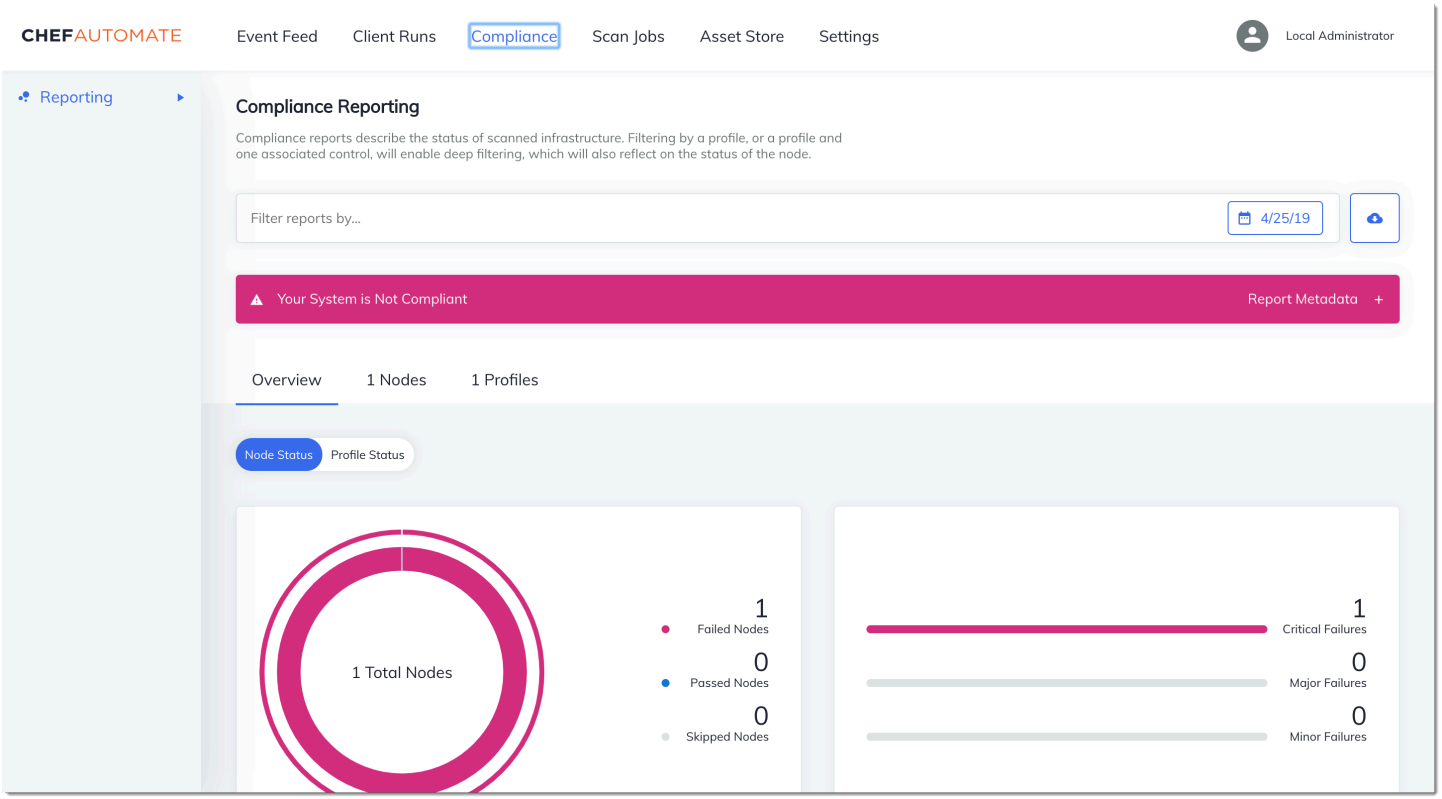
   您可以遵循在 中自動新增節點 AWS OpsWorks for Chef Automate中的步驟將節點的關聯自動化，或遵循個別新增節點中的步驟來逐一新增節點。

6. 在更新節點的執行清單之後，`chef-client` 代理程式即會在下一次執行時執行您指定的配方。根據預設，這種情況會每隔 1800 秒 (30 分鐘) 發生一次。執行之後，您可以從 Chef Automate 儀表板的 Compliance (合規) 標籤，檢視合規結果，並且採取因應動作。

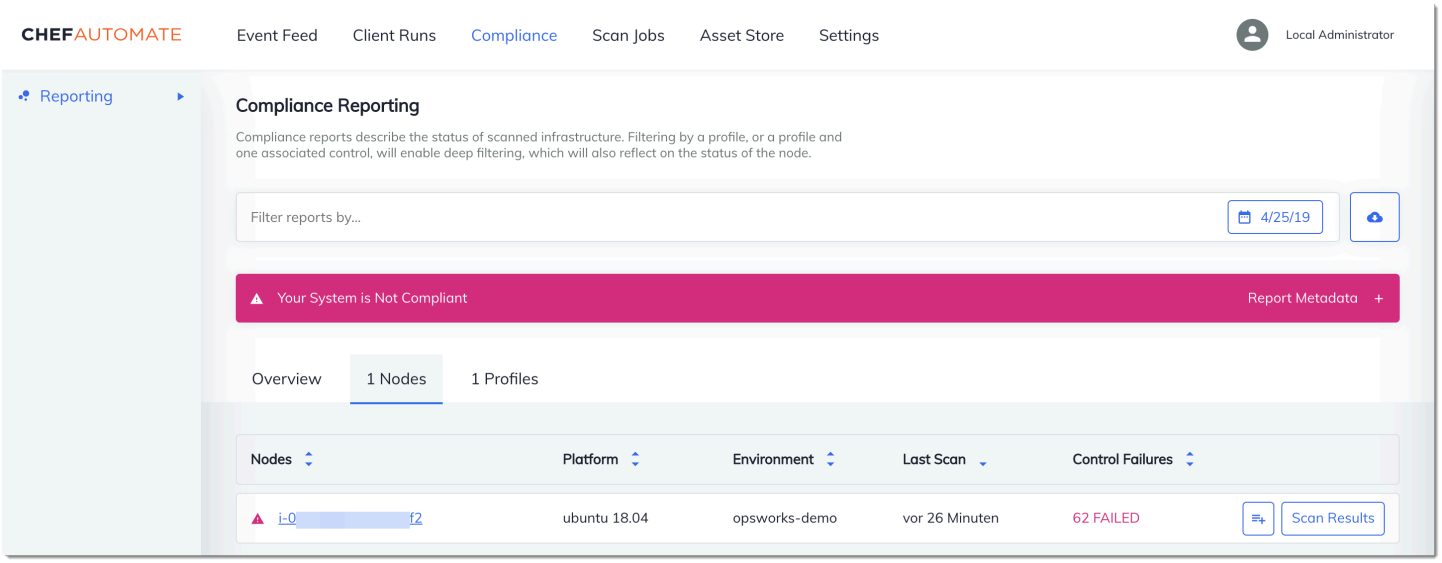


執行合規掃描

在您設定好節點執行清單，並執行第一次代理程式之後，應該很快就會在 Chef Automate 儀表板中看到合規掃描結果。



在 Chef Automate 儀表板上，選擇 Compliance (合規) 標籤。在左側導覽窗格中，選擇 Reporting (報告)。選擇 Profiles (描述檔) 標籤，選擇 Scan Results (掃描結果)，然後選擇含掃描錯誤的節點，以進一步了解導致節點發生錯誤的規則。



一般來說，當新節點尚未滿足 DevSec SSH Baseline (DevSec SSH 基準) 描述檔中的所有規則時，您就會看到未合規的掃描結果。DevSec 強化框架是一種以社群為基礎的專案，其中提供的技術指南可修復違反 DevSec SSH Baseline (DevSec SSH 基準) 描述檔規則的問題。

(選用) 解決未合規的結果

入門套件包含開放原始碼技術指南 `ssh-hardening`；您可以執行此技術指南來修正依據 DevSec SSH Baseline (DevSec SSH 基準) 描述檔執行時發生的未合規結果。

> **Note**
>
> `ssh-hardening` 技術指南會變更您的節點以符合 DevSec SSH Baseline (DevSec SSH 基準) 規則。在任何生產節點上執行此技術指南之前，請先檢閱 Chef Automate 主控台中 DevSec SSH Baseline (DevSec SSH 基準) 描述檔的詳細資訊，以了解技術指南目標的規則違反情況。在任何生產節點上執行之前，請先檢閱開放原始碼 `ssh-hardening` 技術指南的資訊。

執行 **`ssh-hardening`** 技術指南

1. 在文字編輯器中，將 `ssh-hardening` 技術指南附加到 `Policyfile.rb` 執行清單中。`Policyfile.rb` 執行清單應該符合下列項目。

   ```
   run_list  'chef-client', 'opsworks-webserver', 'audit', 'ssh-hardening'
   ```

2. 更新 `Policyfile.rb`，接著將其推送到您的 AWS OpsWorks for Chef Automate 伺服器。

   ```
   chef update Policyfile.rb
      chef push opsworks-demo
   ```

3. 與 `opsworks-demo` 政策相關聯的節點會自動更新執行清單，並將 `ssh-hardening` 技術指南套用到下一次的 `chef-client` 執行。

   由於您使用的是 `chef-client` 技術指南，因此節點會定期 (預設為每隔 30 分鐘一次) 進行狀態檢查。在下一次狀態檢查時，會執行 `ssh-hardening` 技術指南並協助提升節點安全性，以符合 DevSec SSH Baseline (DevSec SSH 基準) 描述檔的規則。

4. 在第一次執行 `ssh-hardening` 技術指南之後，請等待 30 分鐘再次執行 Compliance (合規) 掃描。在 Chef Automate 儀表板中查看結果。第一次執行 DevSec SSH Baseline (DevSec SSH 基準) 掃描時發生的未合規結果應該已經解決。

使用 Chef Automate 1.x 中的合規

如果您的 AWS OpsWorks for Chef Automate 伺服器正在執行 Chef Automate 1.x，請使用本節中的程序來設定 Chef Compliance。

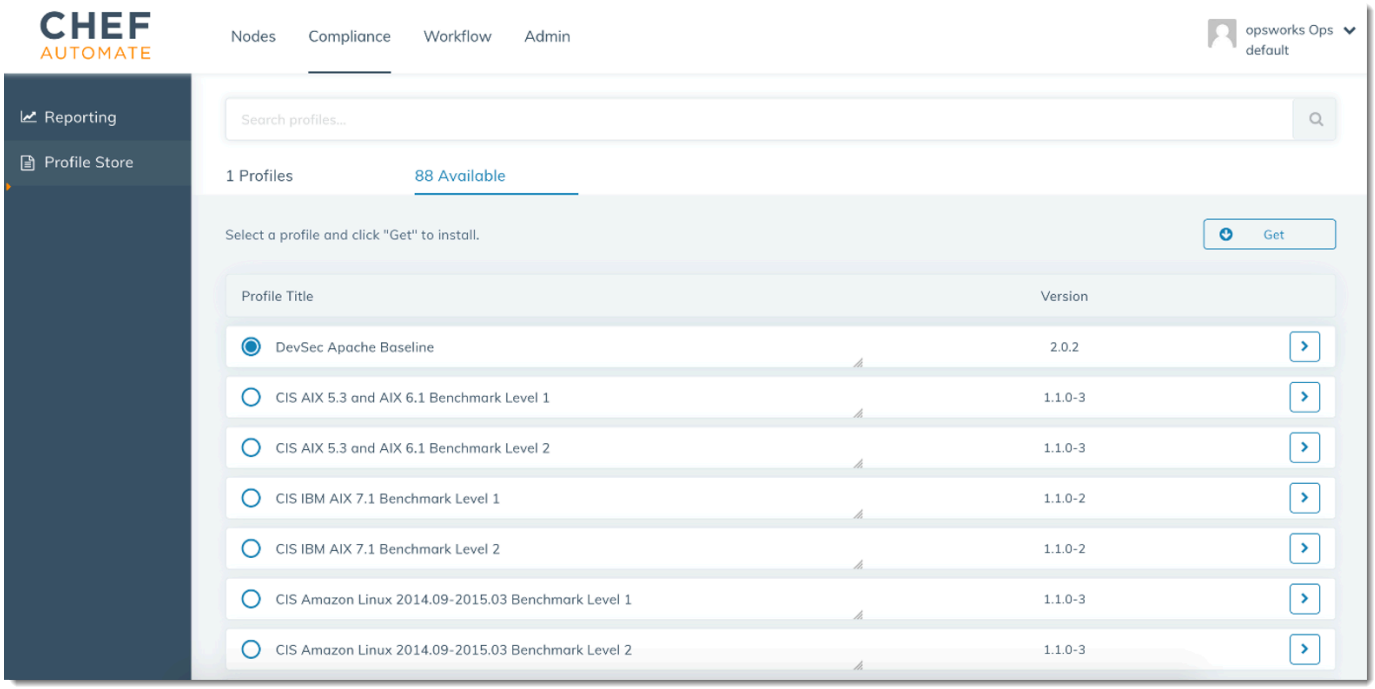
(選用、Chef Automate 1.x) 設定 Chef Compliance

您可以在任何 AWS OpsWorks for Chef Automate 伺服器上設定 Chef Compliance。啟動 AWS OpsWorks for Chef Automate 伺服器之後，請選擇您想要透過 Chef Automate 儀表板中的描述檔來執行的描述檔。安裝描述檔之後，請執行 `berks` 命令，將 Audit 技術指南上傳至您的 Chef Automate 伺服器。安裝 Audit 技術指南時，也會一併安裝適用於 InSpec 的 Gem 套件，這是由 Chef 產生的開放原始碼測試架構，可讓您將自動化測試整合到部署管道中的任何階段。用於 Chef Automate 1.x 時，選擇 Audit 技術指南的 5.1.0 或更新版本。InSpec Gem 套件必須為 1.24.0 版或更新版本。

AWS OpsWorks for Chef Automate 入門套件包含包裝技術指南 `opsworks-audit`，可為您下載並安裝正確版本的 Chef 稽核技術指南。`opsworks-audit` 技術指南也會指示 `chef-client` 代理程式依據 DevSec SSH Baseline (DevSec SSH 基準) 描述檔 (您將透過本主題稍後所述的 Chef Compliance 主控台來安裝) 來評估節點。依據您的偏好設定，您也可以使用技術指南來設定 Compliance (合規)。本節中的指示會說明如何實作 `opsworks-audit` 技術指南。

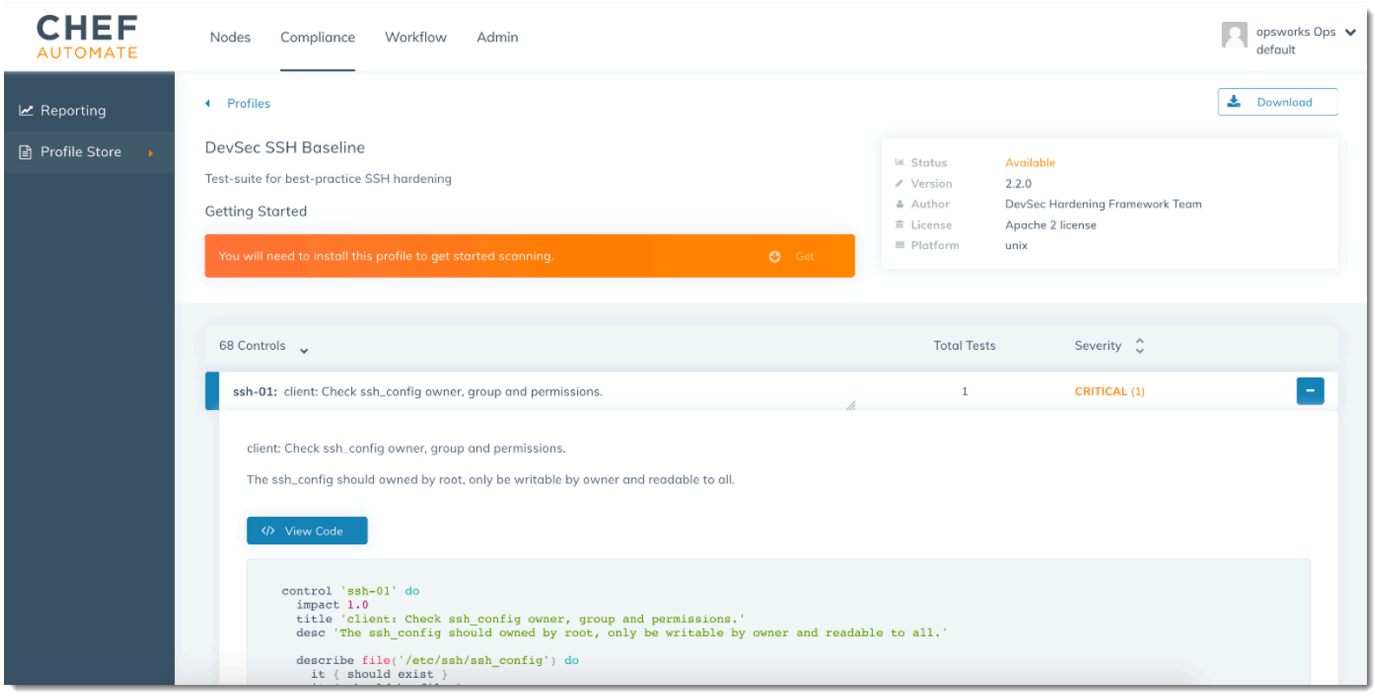
安裝 Compliance (合規) 描述檔

1. 如果您尚未這麼做，請登入 Chef Automate Web 儀表板。使用您在建立 AWS OpsWorks for Chef Automate 伺服器時下載入門套件時收到的登入資料。
2. 在 Chef Automate 儀表板上，選擇 Compliance (合規) 標籤。



3. 在左側導覽列中，選擇 Profile Store (描述檔存放區)，然後選擇 Available (可用) 標籤，以查看預先定義的描述檔。

4. 瀏覽描述檔清單。選擇至少符合其中一個受管節點之作業系統和組態的描述檔。若要查看描述檔的詳細資訊，包括描述檔的目標違反情況描述和基本規則程式碼，請選擇描述檔項目右側的 >。您可以選擇多個描述檔。



5. 若要在您的 Chef Automate 伺服器上安裝選取的描述檔，請選擇 Get (取得)。
6. 當下載完成後，請前往下一個步驟。

安裝和設定 **opsworks-audit** 技術指南

1. 此步驟是選用的，但可以節省步驟 6 將配方新增至節點執行清單的時間。編輯 `roles/opsworks-example-role.rb` 檔案 (其隨附於您建立 AWS OpsWorks for Chef Automate 伺服器期間所下載的入門套件中)。新增下列這幾行。最後一行會標示為註解，因為您可以選擇是否要在執行 Compliance (合規) 掃描之後新增 `ssh-hardening` 技術指南和配方，以解決未合規節點的問題。

```
run_list(
     "recipe[chef-client]",
     "recipe[apache2]",
     "recipe[opsworks-audit]"
     # "recipe[ssh-hardening]"
       )
```

2. 使用文字編輯器，在 Berksfile 中指定所需的技術指南。入門套件中會為您提供 Berksfile 範例。在這個範例中，我們會安裝 Chef Infra 用戶端 (`chef-client`) 技術指南、`apache2` 技術指南和 `opsworks-audit` 技術指南。您的 Berksfile 應該如下所示。

```
source 'https://supermarket.chef.io
     cookbook 'chef-client'
     cookbook 'apache2', '~> 5.0.1'
     cookbook 'opsworks-audit', path: 'cookbooks/opsworks-audit', '~> 1.0.0'
```

   所有技術指南的版本都是經由技術指南的 `metadata.rb` 檔案所控制。每次變更技術指南時，您都必須在技術指南的 `metadata.rb` 中提高其版本。

3. 執行下列命令，將技術指南下載並安裝到本機電腦上的 `cookbooks` 資料夾或工作用電腦中。

```
berks vendor cookbooks
```

4. 執行下列命令，將廠商處理的技術指南上傳到 AWS OpsWorks for Chef Automate 伺服器。

```
knife upload .
```

5. 執行下列命令，藉由顯示目前伺服器上可用的技術指南清單來驗證 `opsworks-audit` 技術指南的安裝。

```
knife cookbook list
```

6. 如果您尚未完成此作業，請先將節點新增到您的伺服器，以便管理。您可以遵循在 中自動新增節點 AWS OpsWorks for Chef Automate中的步驟將節點的關聯自動化，或遵循個別新增節點中的步驟來逐一新增節點。編輯節點執行清單，以新增您在步驟 1 中指定的 `opsworks-example-role` 角色。在這個範例中，我們會編輯 `RUN_LIST` 指令碼 (您用來自動化節點的關聯) 中的 `userdata` 屬性。

```
RUN_LIST="role[opsworks-example-role]"
```

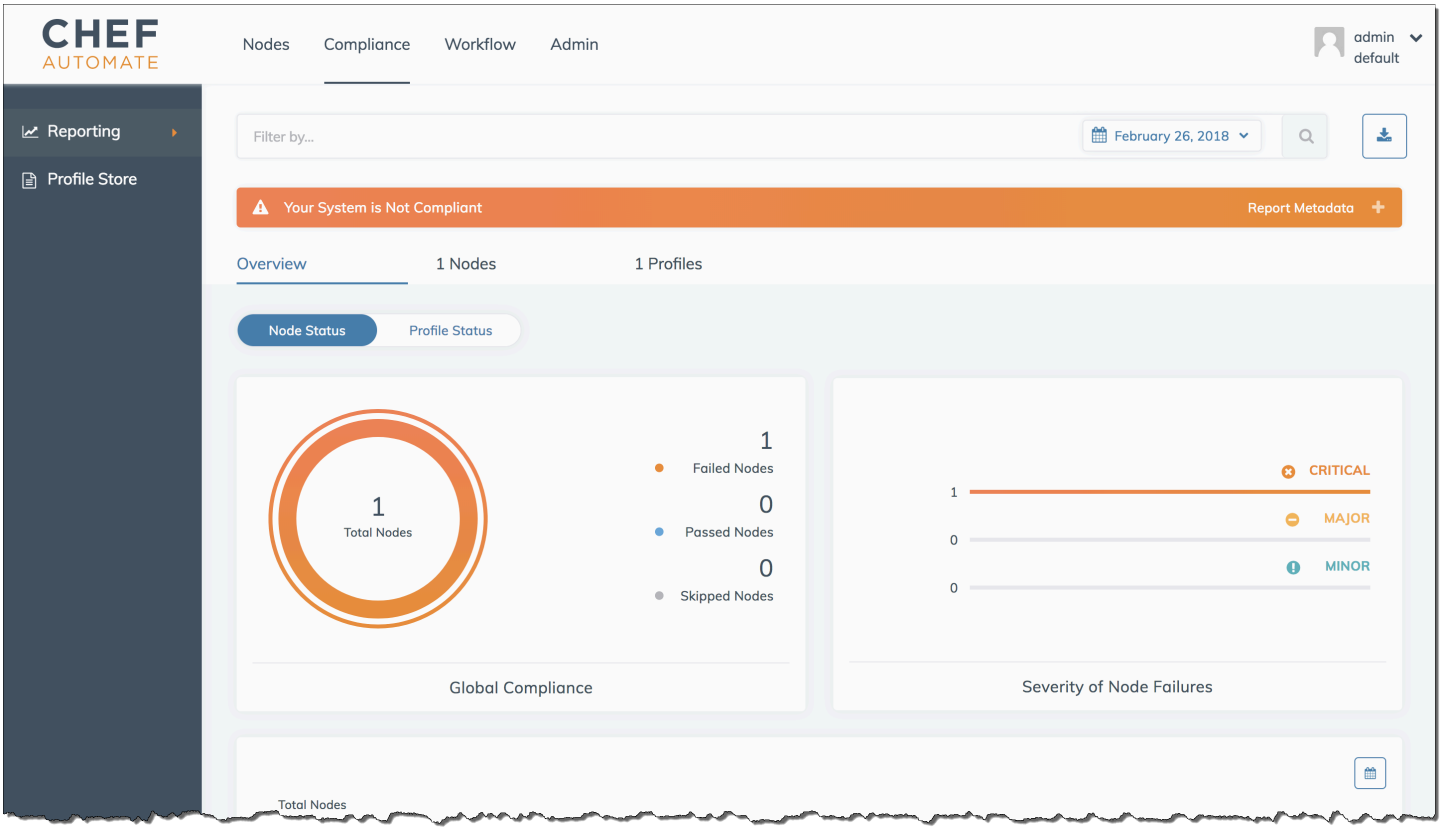
   如果您略過步驟 1，且尚未設定角色，請將個別配方的名稱新增至執行清單。儲存您的變更，並遵循 中的步驟步驟 3：使用無人執行的關聯指令碼建立執行個體，將您的使用者資料指令碼套用至 Amazon EC2 執行個體。

```
RUN_LIST="recipe[chef-client],recipe[apache2],recipe[opworks-audit]"
```

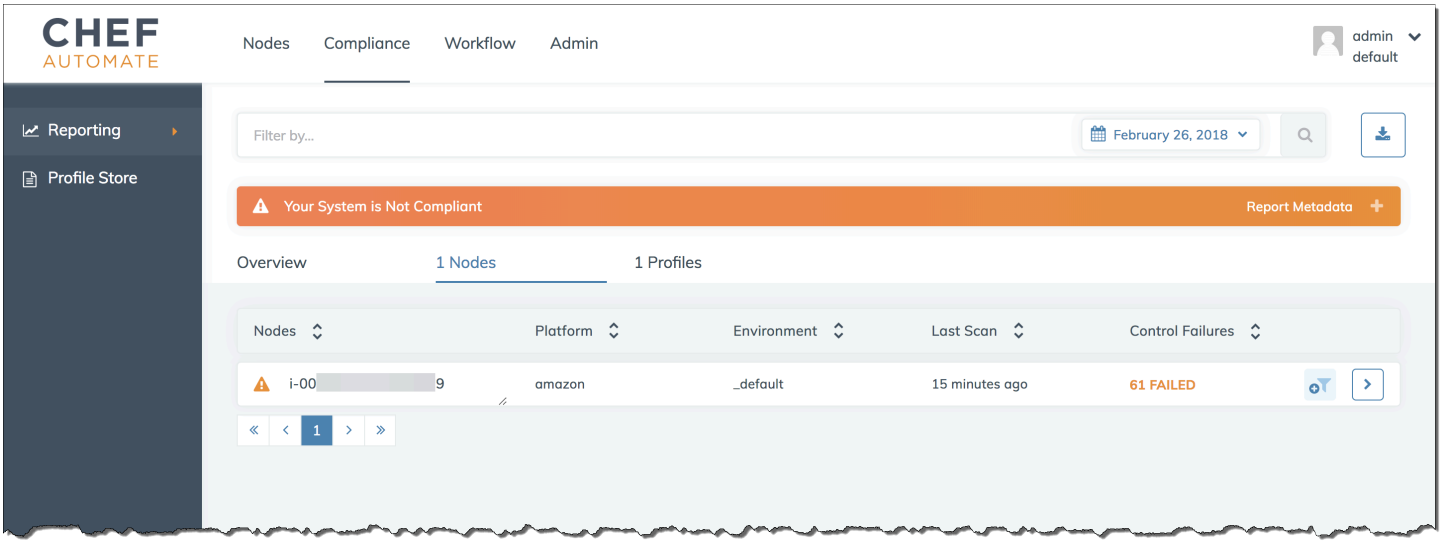
7. 在更新節點的執行清單之後，chef-client 代理程式即會在下一次執行時執行您指定的配方。根據預設，這種情況會每隔 1800 秒 (30 分鐘) 發生一次。執行之後，Chef Automate 儀表板中會顯示您的 Compliance (合規) 結果。

執行合規掃描

在您設定好節點執行清單，並執行第一次代理程式協助程式之後，應該很快就會在 Chef Automate 儀表板中看到合規掃描結果。



在 Chef Automate 儀表板上，選擇 Compliance (合規) 標籤。在左側導覽窗格中，選擇 Reporting (報告)。選擇 Profiles (描述檔) 標籤，選擇 Scan Results (掃描結果)，然後選擇含掃描錯誤的節點，以進一步了解導致節點發生錯誤的規則。



一般來說，當新節點尚未滿足 DevSec SSH Baseline (DevSec SSH 基準) 描述檔中的所有規則時，您就會看到未合規的掃描結果。DevSec 強化框架是一種以社群為基礎的專案，其中提供的技術指南可修復違反 DevSec SSH Baseline (DevSec SSH 基準) 描述檔規則的問題。

(選用) 解決未合規的結果

入門套件包含開放原始碼技術指南 `ssh-hardening`；您可以執行此技術指南來修正依據 DevSec SSH Baseline (DevSec SSH 基準) 描述檔執行時發生的未合規結果。

> **Note**
>
> `ssh-hardening` 技術指南會變更您的節點以符合 DevSec SSH Baseline (DevSec SSH 基準) 規則。在任何生產節點上執行此技術指南之前，請先檢閱 Chef Automate 主控台中 DevSec SSH Baseline (DevSec SSH 基準) 描述檔的詳細資訊，以了解技術指南目標的規則違反情況。在任何生產節點上執行之前，請先檢閱開放原始碼 `ssh-hardening` 技術指南的資訊。

執行 **`ssh-hardening`** 技術指南

1. 在文字編輯器中，將 `ssh-hardening` 技術指南附加到 Berksfile。您的 Berksfile 應該如下所示。

```
source 'https://supermarket.chef.io'
     cookbook 'chef-client'
     cookbook 'apache2', '~> 5.0.1'
     cookbook 'opsworks-audit', path: 'cookbooks/opsworks-audit', '~> 1.0.0' #
 optional
```

```
    cookbook 'ssh-hardening'
```

2. 執行下列命令，將 ssh-hardening 技術指南下載到本機技術指南資料夾，然後將其上傳至您的 AWS OpsWorks for Chef Automate 伺服器。

```
berks vendor cookbooks
knife upload .
```

3. 將 ssh-hardening 配方新增到您的節點執行清單中，如 安裝和設定 opsworks-audit 技術指南中的步驟 1 和 6 所述。

   如果您更新 opsworks-example-role.rb 檔案，請執行下列命令，將變更上傳至您的伺服器。

```
knife upload .
```

   如果您直接更新執行清單，請執行下列命令以上傳變更。節點名稱通常為執行個體 ID。

```
knife node run_list add <node name> 'recipe[ssh-hardening]'
```

4. 由於您使用的是 chef-client 技術指南，因此節點會定期 (預設為每隔 30 分鐘一次) 進行狀態檢查。在下一次狀態檢查時，會執行 ssh-hardening 技術指南並協助提升節點安全性，以符合 DevSec SSH Baseline (DevSec SSH 基準) 描述檔的規則。
5. 在第一次執行 ssh-hardening 技術指南之後，請等待 30 分鐘再次執行 Compliance (合規) 掃描。在 Chef Automate 儀表板中查看結果。第一次執行 DevSec SSH Baseline (DevSec SSH 基準) 掃描時發生的未合規結果應該已經解決。

合規更新

在 AWS OpsWorks for Chef Automate 伺服器上，合規功能會由您排定的系統維護自動更新。當提供 AWS OpsWorks for Chef Automate 伺服器可用的 Chef Automate、Chef Infra 伺服器和 Chef InSpec 的更新版本時，您可能需要檢查並更新正在伺服器上執行的 Audit 技術指南以及 Chef InSpec Gem 套件的支援版本。您已安裝在 AWS OpsWorks for Chef Automate 伺服器上的設定檔不會更新為維護的一部分。

社群和自訂合規描述檔

Chef 目前包含超過 100 種合規掃描描述檔。您可以將社群和自訂描述檔新增至清單，然後根據這些描述檔下載並執行合規掃描，如同已包含的描述檔一般。您可從 Chef Supermarket 取得以社群為基礎的

合規描述檔。自訂描述檔是以 Ruby 為基礎的程式，其中包括控制項資料夾，這些控制項可以指定您的掃描規則。

另請參閱

- Chef Compliance 公告部落格文章
- Chef Automate Compliance 線上培訓
- Chef InSpec 網站
- Chef InSpec 教學課程

取消節點與 AWS OpsWorks for Chef Automate 伺服器的關聯

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或替代解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節說明如何取消受管節點與 AWS OpsWorks for Chef Automate 伺服器的管理的關聯，或將其移除。此操作會在命令列上執行；您無法在 AWS OpsWorks for Chef Automate 管理主控台中取消節點的關聯。目前， AWS OpsWorks for Chef Automate API 不允許批次移除多個節點。本節中的命令會一次取消一個節點的關聯。

我們建議您在刪除伺服器前先從 Chef 伺服器取消與節點的關聯，使節點可在不嘗試重新連線到伺服器的情況下繼續運作。若要執行此操作，請執行 `disassociate-node` AWS CLI 命令。

取消與節點的關聯

1. 在 中 AWS CLI，執行下列命令來取消節點的關聯。*`Node_name`* 是您要取消關聯的節點名稱；對於 Amazon EC2 執行個體，這是執行個體 ID。*`Server_name`* 是您要將節點取消關聯的來源 Chef 伺服器名稱。`--engine-attributes` 會指定您的預設 `CHEF_AUTOMATE_ORGANIZATION` 名稱。這三個參數均為必要。

   `--region` 參數並非必要項目，除非您希望從不是位於您預設區域內的 Chef 伺服器取消與節點的關聯。

```
aws opsworks-cm --region Region_name disassociate-node --node-
name Node_name --server-name Server_name --engine-attributes
 "Name=CHEF_AUTOMATE_ORGANIZATION,Value='default'"
```

下列是範例命令。

```
aws opsworks-cm --region us-west-2 disassociate-node --node-name
 i-0010zzz00d66zzz90 --server-name opsworkstest --engine-attributes
 "Name=CHEF_AUTOMATE_ORGANIZATION,Value='default'"
```

2. 等待回應訊息指出取消關聯已完成。

   成功取消節點與 AWS OpsWorks for Chef Automate 伺服器的關聯後，它仍會出現在 Chef Automate 儀表板中。根據預設，Chef 會將節點狀態資訊強制保留一段時間，並在幾天後自動清除該節點。

如需如何刪除 AWS OpsWorks for Chef Automate 伺服器的詳細資訊，請參閱 [刪除 AWS OpsWorks for Chef Automate 伺服器]。

相關主題

下列 AWS 部落格文章提供有關自動將節點與 Chef Automate 伺服器建立關聯、使用 Auto Scaling 群組或在多個帳戶中建立關聯的詳細資訊。

- 使用 AWS OpsWorks for Chef Automate 搭配 Auto Scaling 管理 EC2 執行個體
- OpsWorks for Chef Automate – 自動引導不同帳戶中的節點

刪除 AWS OpsWorks for Chef Automate 伺服器

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或替代解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節說明如何刪除 AWS OpsWorks for Chef Automate 伺服器。刪除伺服器也會刪除其事件、日誌和任何存放在伺服器上的技術指南。支援的資源 (Amazon Elastic Compute Cloud 執行個體、Amazon Elastic Block Store 磁碟區等） 也會一併刪除，以及所有自動備份。

雖然刪除伺服器不會刪除節點，但節點將不再由刪除的伺服器管理，並會持續嘗試重新連線。因此，我們建議您在刪除 Chef 伺服器前取消關聯受管節點。在此版本中，您可以透過執行 AWS CLI 命令來取消節點的關聯。

步驟 1：取消與受管節點的關聯

在刪除伺服器前先從 Chef 伺服器取消關聯節點，使節點可在不嘗試重新連線到伺服器的情況下繼續運作。若要執行此操作，請執行 `disassociate-node` AWS CLI 命令。

取消與節點的關聯

1. 在 中 AWS CLI，執行下列命令來取消節點的關聯。*`Server_name`* 是您要將節點取消關聯的來源 Chef 伺服器名稱。

   ```
   aws opsworks-cm --region Region_name disassociate-node --node-name Node_name --
   server-name Server_name
   ```

2. 等待回應訊息指出取消關聯已完成。

步驟 2：刪除伺服器

1. 在儀表板上的伺服器磚上，展開 Actions (動作) 選單。
2. 選擇 Delete server (刪除伺服器)。
3. 出現提示要您確認刪除時，選擇 Yes (是)。

重設 Chef Automate 儀表板登入資料

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或替代解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

有時候，您可能會希望變更您用來登入 Chef Automate 儀表板的密碼。您也可以使用本節中顯示的 Amazon EC2 Systems Manager AWS CLI 命令，在遺失時變更 Chef Automate 儀表板密碼。您使用的命令取決於您的 Chef Automate 伺服器是否執行 Chef Automate 第 1 版或第 2 版。

1. 若要傳回 Chef 伺服器的執行個體 ID，請開啟 AWS Management Console 至下頁。

   https://console.aws.amazon.com/ec2/v2/home?region=*region_of_your_server*#Instances:search=aws-opsworks-cm-*server_name*

   例如，在美國西部（奧勒岡）區域中名為 MyChefServer 的 Chef 伺服器，主控台 URL 會如下。

   https://console.aws.amazon.com/ec2/v2/home?region=us-west-2#Instances:search=aws-opsworks-cm-MyChefServer

   記下主控台中顯示的執行個體 ID，您將會在變更密碼時用到。

2. 若要重設 Chef Automate 儀表板登入密碼，請執行下列其中一個 AWS CLI 命令，視您的伺服器是否執行 Chef Automate 1 或 Chef Automate 2 而定。將 *enterprise_name* 取代為您的企業或組織名稱、將 *user_name* 取代為伺服器上管理員的使用者名稱、將 *new_password* 取代為您要使用的密碼，並將*region_name* 取代為伺服器所在的區域。若您沒有指定企業名稱，則企業名稱將為 `default`。根據預設，*enterprise_name* 為 `default` (此為一律會用來佈建之組織的名稱)。對於 *user_name*， AWS OpsWorks for Chef Automate 僅建立名為 的使用者`admin`。記下新的密碼，然後將其存放在安全但便利的位置。

   對於 Chef Automate 1：

   ```
   aws ssm send-command --document-name "AWS-RunShellScript" --comment "reset admin
    password" --instance-ids "instance_id"
   --parameters commands="sudo delivery-ctl reset-
   password enterprise_name user_name new_password" --region region_name --output text
   ```

   對於 Chef Automate 2：

   ```
   aws ssm send-command --document-name "AWS-RunShellScript" --comment "reset admin
    password" --instance-ids "instance_id"
   --parameters commands="sudo chef-automate iam admin-access restore new_password" --
   region region_name --output text
   ```

3. 等待輸出文字 (在此案例中為命令 ID) 以顯示密碼變更已完成。

使用 記錄 AWS OpsWorks for Chef Automate API 呼叫 AWS CloudTrail

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或替代解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks for Chef Automate 已與 整合 AWS CloudTrail，此服務提供 IAM 身分所採取動作的記錄，或 AWS 中的服務 AWS OpsWorks for Chef Automate。CloudTrail 會將 的所有 API 呼叫擷取 AWS OpsWorks for Chef Automate 為事件，包括來自 AWS OpsWorks for Chef Automate 主控台的呼叫，以及來自對 AWS OpsWorks for Chef Automate APIs的程式碼呼叫。如果您建立追蹤，則可以啟用 CloudTrail 事件持續交付至 Amazon S3 儲存貯體，包括 的事件 AWS OpsWorks for Chef Automate。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。使用 CloudTrail 收集的資訊，您可以判斷提出的請求 AWS OpsWorks for Chef Automate、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 OpsWorks for Chef Automate CloudTrail 中的資訊

建立 AWS 帳戶時，會在您的帳戶上啟用 CloudTrail。當活動在 中發生時 AWS OpsWorks for Chef Automate，該活動會記錄於 CloudTrail 事件，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 的事件 AWS OpsWorks for Chef Automate，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，當您在主控台建立權杖時，權杖會套用到所有區域。追蹤會記錄 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 CloudTrail 日誌中收集的事件資料並對其採取行動。如需詳細資訊，請參閱：

- 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，以及從多個帳戶接收 CloudTrail 日誌檔案

CloudTrail 會記錄所有 AWS OpsWorks for Chef Automate 動作，並記錄在 AWS OpsWorks for Chef Automate API 參考中。例如，呼叫 CreateServer、CreateBackup 和 DescribeServers 動作會在 CloudTrail 日誌檔案中產生項目。

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：

- 該請求是否使用根或 IAM 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 OpsWorks for Chef Automate 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 日誌檔案並非依公有 API 呼叫追蹤記錄的堆疊排序，因此不會以任何特定順序出現。

以下範例顯示 AWS OpsWorks for Chef Automate `CreateServer` 動作的 CloudTrail 日誌項目。

```
{"eventVersion":"1.05",
"userIdentity":{
    "type":"AssumedRole",
    "principalId":"ID number:OpsWorksCMUser",
    "arn":"arn:aws:sts::831000000000:assumed-role/Admin/OpsWorksCMUser",
    "accountId":"831000000000","accessKeyId":"ID number",
    "sessionContext":{
        "attributes":{
            "mfaAuthenticated":"false",
            "creationDate":"2017-01-05T22:03:47Z"
            },
        "sessionIssuer":{
            "type":"Role",
            "principalId":"ID number",
            "arn":"arn:aws:iam::831000000000:role/Admin",
            "accountId":"831000000000",
            "userName":"Admin"
```

```
            }
        }
    },
"eventTime":"2017-01-05T22:18:23Z",
"eventSource":"opsworks-cm.amazonaws.com",
"eventName":"CreateServer",
"awsRegion":"us-west-2",
"sourceIPAddress":"101.25.190.51",
"userAgent":"console.amazonaws.com",
"requestParameters":{
    "serverName":"OpsChef-test-server",
    "engineModel":"Single",
    "engine":"Chef",
    "instanceProfileArn":"arn:aws:iam::831000000000:instance-profile/aws-opsworks-cm-
ec2-role",
    "backupRetentionCount":3,"serviceRoleArn":"arn:aws:iam::831000000000:role/service-
role/aws-opsworks-cm-service-role",
    "engineVersion":"12",
    "preferredMaintenanceWindow":"Fri:21:00",
    "instanceType":"t2.medium",
    "subnetIds":["subnet-1e111f11"],
    "preferredBackupWindow":"Wed:08:00"
    },
"responseElements":{
    "server":{
        "endpoint":"OpsChef-test-server-thohsgreckcnwgz3.us-west-2.opsworks-cm.io",
        "createdAt":"Jan 5, 2017 10:18:22 PM",
        "serviceRoleArn":"arn:aws:iam::831000000000:role/service-role/aws-opsworks-cm-
service-role",
        "preferredBackupWindow":"Wed:08:00",
        "status":"CREATING",
        "subnetIds":["subnet-1e111f11"],
        "engine":"Chef",
        "instanceType":"t2.medium",
        "serverName":"OpsChef-test-server",
        "serverArn":"arn:aws:opsworks-cm:us-west-2:831000000000:server/OpsChef-test-
server/8epp7f6z-e91f-4z10-89z5-8c6219cdb09f",
        "engineModel":"Single",
        "backupRetentionCount":3,
        "engineAttributes":[
            {"name":"CHEF_STARTER_KIT","value":"*** Redacted ***"},
            {"name":"CHEF_PIVOTAL_KEY","value":"*** Redacted ***"},
            {"name":"CHEF_DELIVERY_ADMIN_PASSWORD","value":"*** Redacted ***"}],
        "engineVersion":"12.11.1",
```

```
        "instanceProfileArn":"arn:aws:iam::831000000000:instance-profile/aws-opsworks-
cm-ec2-role",
        "preferredMaintenanceWindow":"Fri:21:00"
        }
    },
"requestID":"de7f64f9-d394-12ug-8081-7bb0386fbcb6",
"eventID":"8r7b18df-6c90-47be-87cf-e8346428cfc3",
"eventType":"AwsApiCall",
"recipientAccountId":"831000000000"
}
```

故障診斷 AWS OpsWorks for Chef Automate

> **Important**
>
> AWS OpsWorks for Chef Automate 已於 2024 年 5 月 5 日結束生命週期，並已針對新客戶和現有客戶停用。我們建議現有客戶遷移至 Chef SaaS 或替代解決方案。如果您有任何問題，可以在 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本主題包含一些常見問題 AWS OpsWorks for Chef Automate ，以及這些問題的建議解決方案。

主題

- 一般疑難排解秘訣
- 對特定錯誤進行故障診斷
- 其他協助及支援

一般疑難排解秘訣

若您無法建立或使用 Chef 伺服器，您可以檢視錯誤訊息或日誌，來協助您故障診斷問題。以下任務說明在您故障診斷 Chef 伺服器問題時一般開始著手的位置。如需特定錯誤和解決方案的資訊，請參閱本主題的對特定錯誤進行故障診斷一節。

- 如果 Chef 伺服器無法啟動，請使用 AWS OpsWorks for Chef Automate 主控台檢視錯誤訊息。在 Chef 伺服器詳細資訊頁面上，與啟動和執行伺服器相關的錯誤訊息會顯示在頁面的頂端。錯誤可能來自用來建立 Chef 伺服器的服務 AWS OpsWorks for Chef Automate AWS CloudFormation，或 Amazon EC2。在詳細資訊頁面上，您也可以檢視在執行中伺服器上發生的事件，其中也可能包含故障事件訊息。

- 為協助解決 EC2 問題，請使用 SSH 連線到您伺服器的執行個體並檢視日誌。EC2 執行個體日誌存放在 `/var/log/aws/opsworks-cm` 目錄。這些日誌會在 AWS OpsWorks for Chef Automate 啟動 Chef 伺服器時擷取命令輸出。

對特定錯誤進行故障診斷

主題

- 伺服器處於連線中斷狀態
- 受管節點顯示在 Chef Automate 儀表板中的 Missing (遺失) 資料行內
- 無法建立 Chef 保存庫。knife vault 命令失敗並顯示錯誤
- 伺服器建立失敗，並顯示 "requested configuration is currently not supported" 訊息
- Chef 伺服器無法辨識在 Chef Automate 儀表板中新增的組織名稱
- 無法建立伺服器的 Amazon EC2 執行個體
- 服務角色錯誤導致伺服器無法建立
- 超過彈性 IP 地址限制
- 無法登入 Chef Automate 儀表板
- 自動節點關聯失敗
- 系統維護失敗

伺服器處於連線中斷狀態

問題：伺服器的狀態顯示為連線中斷。

原因：當 外部的實體 AWS OpsWorks 變更 AWS OpsWorks for Chef Automate 伺服器或其支援資源時，這最常發生。 AWS OpsWorks 無法連線到連線中斷狀態的 Chef Automate 伺服器，以處理維護任務，例如建立備份、套用作業系統修補程式或更新 Chef Automate。因此，您的伺服器可能缺少重要的更新、容易發生安全問題，或是無法如預期般運作。

解決方案：嘗試下列步驟來還原伺服器的連線。

1. 請確定您的服務角色具有所有必要的許可。

    a. 在伺服器的設定頁面上，在網路和安全性中，選擇伺服器正在使用之服務角色的連結。這會開啟要在 IAM 主控台中檢視的服務角色。

b. 在許可索引標籤上，確認 AWSOpsWorksCMServiceRole 位於許可政策清單中。如果未列出，請手動將 AWSOpsWorksCMServiceRole 受管政策新增至角色。

c. 在信任關係索引標籤上，確認服務角色具有信任政策，信任opsworks-cm.amazonaws.com服務代表您擔任角色。如需如何將信任政策與角色搭配使用的詳細資訊，請參閱修改角色（主控台）或 AWS 安全部落格文章：如何將信任政策與 IAM 角色搭配使用。

2. 請確定您的執行個體設定檔具有所有必要的許可。

a. 在伺服器的設定頁面上，在網路和安全性中，選擇伺服器正在使用的執行個體描述檔連結。這會開啟執行個體設定檔，以便在 IAM 主控台中檢視。

b. 在許可索引標籤上，確認 AmazonEC2RoleforSSM和 AWSOpsWorksCMInstanceProfileRole 都位於許可政策清單中。如果其中之一或兩者未列出，請手動將這些受管政策新增至角色。

c. 在信任關係索引標籤上，確認服務角色具有信任政策，信任ec2.amazonaws.com服務代表您擔任角色。如需如何搭配角色使用信任政策的詳細資訊，請參閱修改角色（主控台）或 AWS 安全部落格文章，如何搭配 IAM 角色使用信任政策。

3. 在 Amazon EC2 主控台中，請確定您位於與伺服器區域相同的區域，然後重新啟動伺服器正在使用的 AWS OpsWorks for Chef Automate EC2 執行個體。

a. 選擇名為 aws-opsworks-cm-instance-*server-name* 的 EC2 執行個體。

b. 在執行個體狀態功能表中，選擇重新啟動執行個體。

c. 您的伺服器最多需要 15 分鐘才能重新啟動並完全上線。

4. 在 AWS OpsWorks for Chef Automate 主控台的伺服器詳細資訊頁面上，確認伺服器狀態現在正常運作。

如果執行上述步驟後，伺服器狀態仍然遺失連線，請嘗試下列其中一項。

- 透過建立新的伺服器並刪除原始伺服器來取代伺服器。如果目前伺服器上的資料對您很重要，請從最近的備份還原伺服器，並在刪除原始、無回應的伺服器之前驗證資料是否為最新。
- 聯絡 AWS 支援。

受管節點顯示在 Chef Automate 儀表板中的 Missing (遺失) 資料行內

問題：受管節點顯示在 Chef Automate 儀表板中的 Missing (遺失) 資料行內。

原因：當節點沒有連線至 Chef Automate 伺服器超過 12 小時，且 `chef-client` 無法在節點上執行時，節點會從 12 小時前的狀態變更，並移動到 Chef Automate 儀表板的 Missing (遺失) 資料行。

解決方案：驗證節點處於線上狀態。嘗試執行 `knife node show` *`node_name`* `--run-list` 以查看 `chef-client` 是否能在節點上執行，或是 `knife node show -l` *`node_name`* 以顯示節點的所有相關資訊。節點可能處於離線狀態，或並未與網路連線。

無法建立 Chef 保存庫。`knife vault` 命令失敗並顯示錯誤

問題：您嘗試透過執行 `knife vault` 命令，在您的 Chef Automate 伺服器 (例如存放網域連接 Windows 式節點登入資料的保存庫) 建立保存庫。命令傳回類似以下內容的錯誤訊息。

```
WARN: Auto inflation of JSON data is deprecated. Please pass in the class to inflate or
 use #edit_hash (CHEF-1)
at /opt/chefdk/embedded/lib/ruby/2.3.0/forwardable.rb:189:in `edit_data'.Please see
 https://docs.chef.io/deprecations_json_auto_inflate.html
for further details and information on how to correct this problem.
WARNING: pivotal not found in users, trying clients.
ERROR: ChefVault::Exceptions::AdminNotFound: FATAL: Could not find pivotal in users or
 clients!
```

樞紐使用者並未在您遠端執行 `knife user list` 傳回，但當您在您的 Chef Automate 伺服器上本機執行 `chef-server-ctl user-show` 命令時，卻又能在結果中看到樞紐使用者。換句話說，您的 `knife vault` 無法找到樞紐使用者，但您知道它存在。

原因：雖然樞紐使用者在 Chef 中被視為超級使用者，因此具備完整許可，但它並非任何組織的成員，包括在 AWS OpsWorks for Chef Automate中使用的 `default` 組織。`knife user list` 命令會傳回所有您 Chef 組態中目前組織內的使用者。`chef-server-ctl user-show` 命令則會傳回所有使用者 (包含樞紐使用者)，而與組織無關。

解決方案：若要修正此問題，請透過執行 `knife opc`，將樞紐使用者新增至預設組織。

首先，您需要安裝 knife-opc 外掛程式。

```
chef gem install knife-opc
```

在您安裝外掛程式之後，執行下列命令以將樞紐使用者新增至預設組織。

```
knife opc org user add default pivotal
```

您可以透過再次執行 `knife user list`，來驗證樞紐使用者是否已是預設組織的一部分。結果中應會列出 `pivotal`。然後，請再次嘗試執行 `knife vault`。

伺服器建立失敗，並顯示 "requested configuration is currently not supported" 訊息

問題：您嘗試建立 Chef Automate 伺服器，但伺服器建立失敗並顯示類似下列錯誤訊息："The requested configuration is currently not supported. Please check the documentation for supported configurations."

原因：Chef Automate 伺服器可能指定了不支援的執行個體類型。若您選擇在擁有非預設租用的 VPC (例如一個專用執行個體) 中建立 Chef Automate 伺服器，所有指定 VPC 中的執行個體也都必須是專用或主控租用。因為有些執行個體類型 (例如 t2) 僅能使用預設租用，指定 VPC 便可能無法支援 Chef Automate 伺服器執行個體類型，因此導致伺服器建立失敗。

解決方案：若您選擇具有非預設租用的 VPC，請使用支援專用租用的 m4 執行個體類型。

Chef 伺服器無法辨識在 Chef Automate 儀表板中新增的組織名稱

問題：您已在 Chef Automate 儀表板中新增工作流程組織名稱，或在無人看管的節點關聯指令碼"default"中指定 值以外的CHEF_AUTOMATE_ORGANIZATION值，但節點關聯失敗。您的 AWS OpsWorks for Chef Automate 伺服器無法識別新的組織名稱。

原因：工作流程組織名稱和 Chef 伺服器的組織名稱不相同。您可以在 web 式的 Chef Automate 儀表板中建立新的工作流程組織，但無法建立 Chef 伺服器組織名稱。您只能使用 Chef Automate 儀表板檢視現有的 Chef 伺服器組織。您在 Chef Automate 儀表板中建立的新組織為工作流程組織，Chef 伺服器無法識別。您無法透過在節點關聯指令碼中指定他們，來建立新的組織名稱。在組織並未新增到 Chef 伺服器時於節點關聯指令碼中參考組織名稱，便會導致節點關聯失敗。

解決方案：若要建立在 Chef 伺服器上可識別的新組織，請使用 `knife opc org create` 命令，或執行 `chef-server-ctl org-create`。

無法建立伺服器的 Amazon EC2 執行個體

問題：伺服器建立失敗，並顯示與下列內容相似的錯誤訊息："The following resource(s) failed to create: [EC2Instance]. Failed to receive 1 resource signal(s) within the specified duration."

原因：這最有可能是因為 EC2 執行個體沒有網路存取。

解決方案：確保執行個體具有傳出網際網路存取，且 AWS 服務代理程式能夠發出命令。確認您的 VPC (使用單一公有子網路的 VPC) 已啟用 DNS resolution (DNS 解析)，並且您的子網路也已啟用 Auto-assign Public IP (自動指派公有 IP) 設定。

服務角色錯誤導致伺服器無法建立

問題：伺服器建立失敗，並顯示下列錯誤訊息："Not authorized to perform sts:AssumeRole."

原因：這可能會在您使用的服務角色缺少適當的許可，無法建立新伺服器時發生。

解決方案：開啟 AWS OpsWorks for Chef Automate 主控台；使用 主控台來產生新的服務角色和執行個體描述檔角色。若您偏好使用您自己的服務角色，請將 AWSOpsWorksCMServiceRole 政策連接到角色。確認 opsworks-cm.amazonaws.com 已列在角色信任關係中的服務之間。確認與 Chef 伺服器相關聯的服務角色已連接 AWSOpsWorksCMServiceRole 受管政策。

超過彈性 IP 地址限制

問題：伺服器建立失敗，並顯示下列錯誤訊息："The following resource(s) failed to create: [EIP, EC2Instance]. Resource creation cancelled, the maximum number of addresses has been reached."

原因：當您的帳戶使用的彈性 IP (EIP) 地址數超過最大值時，便可能發生此情況。預設 EIP 地址限制為 5 個。

解決方案：您可以釋出現有的 EIP 地址，或刪除您的帳戶未主動使用的 EIP 地址，也可以聯絡 AWS 客戶支援來提高與您的帳戶相關聯的 EIP 地址限制。

無法登入 Chef Automate 儀表板

問題：Chef Automate 儀表板顯示類似下列內容的錯誤："Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at https://myserver-name.region.opsworks-cm.io/api/v0/e/default/verify-token. (Reason: CORS header 'Access-Control-Allow-Origin' missing)". 錯誤也可能類似 "The User Id / Password combination entered is incorrect."。

原因：Chef Automate 儀表板明確設定 FQDN，不接受相對 URL。此時，您無法使用 Chef 伺服器的 IP 地址登入。您只能使用伺服器的 DNS 名稱登入。

解決方案：只使用 Chef 伺服器的 DNS 名稱項目登入 Chef Automate 儀表板，而非 IP 地址。您也可以透過執行 AWS CLI 命令嘗試重設 Chef Automate 儀表板登入資料，如重設 Chef Automate 儀表板登入資料中所述。

自動節點關聯失敗

問題：新 Amazon EC2 節點的自動或全自動關聯失敗。應新增到 Chef 伺服器的節點並未在 Chef Automate 儀表板中出現，也沒有在 `knife client show` 或 `knife node show` 命令的結果中列出。

原因：這可能會在您沒有將 IAM 角色設為允許 opsworks-cm API 呼叫，以和新的 EC2 執行個體通訊的執行個體描述檔時發生。

解決方案：將政策連接到您的 EC2 執行個體描述檔，以允許 AssociateNode 和 DescribeNodeAssociationStatus API 呼叫使用 EC2，如在 中自動新增節點 AWS OpsWorks for Chef Automate中所述。

系統維護失敗

AWS OpsWorks CM 會執行每週系統維護，以確保 Chef Server 和 Chef Automate Server 的最新次要版本，包括安全性更新，一律在 AWS OpsWorks for Chef Automate 伺服器上執行。如果系統維護因任何原因失敗， 會 AWS OpsWorks CM 通知您失敗。如需系統維護的詳細資訊，請參閱中的系統維護 AWS OpsWorks for Chef Automate。

本節說明失敗的可能原因，並建議解決方案。

主題

- 服務角色或執行個體描述檔錯誤導致系統維護無法進行

服務角色或執行個體描述檔錯誤導致系統維護無法進行

問題：系統維護失敗，並出現錯誤訊息，指出「未獲授權執行 sts：AssumeRole」，或有關許可的類似錯誤訊息。

原因：當您正在使用的服務角色或執行個體描述檔缺少足夠的許可，在伺服器上執行系統維護時，可能會發生這種情況。

解決方案：確保您的服務角色和執行個體描述檔具有所有必要的許可。

1. 請確定您的服務角色具有所有必要的許可。
   a. 在伺服器的設定頁面上，於網路和安全性中，選擇伺服器正在使用之服務角色的連結。這會開啟要在 IAM 主控台中檢視的服務角色。
   b. 在許可索引標籤上，確認 AWSOpsWorksCMServiceRole 已連接至服務角色。如果 AWSOpsWorksCMServiceRole 未列出，請將此政策新增至角色。
   c. 確認 opsworks-cm.amazonaws.com 已列在角色信任關係中的服務之間。如需如何搭配角色使用信任政策的詳細資訊，請參閱修改角色（主控台）或 AWS 安全部落格文章，了解如何搭配 IAM 角色使用信任政策。
2. 請確定您的執行個體設定檔具有所有必要的許可。

a. 在伺服器的設定頁面上，在網路和安全性中，選擇伺服器正在使用之執行個體描述檔的連結。這會開啟執行個體設定檔，以便在 IAM 主控台中檢視。

b. 在許可索引標籤上，確認 AmazonEC2RoleforSSM和 AWSOpsWorksCMInstanceProfileRole 都位於許可政策清單中。如果其中之一或兩者未列出，請手動將這些受管政策新增至角色。

c. 在信任關係索引標籤上，確認服務角色具有信任政策，信任ec2.amazonaws.com服務代表您擔任角色。如需如何將信任政策與角色搭配使用的詳細資訊，請參閱修改角色（主控台）或 AWS 安全部落格文章，如何將信任政策與 IAM 角色搭配使用。

其他協助及支援

若您在本主題中沒有看到您的特定問題，或是您已嘗試本主題中的建議，但仍然發生問題，請造訪 AWS OpsWorks 論壇。

您也可以前往 AWS Support 中心。 AWS 支援中心是建立和管理 AWS 支援案例的中樞。 AWS 支援中心也包含其他實用資源的連結，例如論壇、技術FAQs、服務運作狀態和 AWS Trusted Advisor。

AWS OpsWorks 組態管理 (CM) 中的安全性

的雲端安全性 AWS 是最高優先順序。身為 AWS 客戶，您可以從資料中心和網路架構中受益，該架構旨在滿足最安全敏感組織的需求。

安全性是 AWS 和 之間的共同責任。共同責任模型 將此描述為雲端的安全和雲端內的安全：

- 雲端的安全性 – AWS 負責保護在 Cloud AWS 中執行 AWS 服務的基礎設施。 AWS 也提供您可以安全使用的服務。在 AWS 合規計畫中，第三方稽核員會定期測試並驗證我們的安全功效。若要了解適用於 AWS OpsWorks CM 的合規計劃，請參閱合規計劃範圍內的AWS 服務。
- 雲端安全 – 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 CM AWS OpsWorks 時套用共同的責任模型。下列主題說明如何設定 AWS OpsWorks CM 以符合您的安全和合規目標。您也會了解如何使用其他 AWS 服務來協助您監控和保護 AWS OpsWorks CM 資源。

主題

- CM AWS OpsWorks 中的資料保護
- 資料加密
- CM AWS OpsWorks 的身分和存取管理
- 網際網路流量隱私權
- 在 CM AWS OpsWorks 中記錄和監控
- AWS OpsWorks CM 的合規驗證
- CM AWS OpsWorks 中的彈性
- AWS OpsWorks CM 中的基礎設施安全
- CM AWS OpsWorks 中的組態和漏洞分析
- CM AWS OpsWorks 的安全最佳實務

CM AWS OpsWorks 中的資料保護

AWS 共同責任模型適用於 AWS OpsWorks Configuration Management 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。

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 OpsWorks CM 或其他 AWS 服務 使用主控台、API AWS CLI或 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。

OpsWorks CM 伺服器的名稱不會加密。

OpsWorks CM 會在建立和維護您的 AWS OpsWorks for Chef Automate 和 AWS OpsWorks for Puppet Enterprise 伺服器的過程中收集下列客戶資料。

- 對於 OpsWorks for Puppet Enterprise，我們會收集 Puppet Enterprise 用來啟用 Puppet 主節點與受管節點之間通訊的私有金鑰。
- 對於 AWS OpsWorks for Chef Automate，如果您使用的是自訂網域，我們會收集您連接到服務的憑證私有金鑰。您在使用自訂網域建立 Chef Automate 伺服器時提供的私密金鑰將傳遞到您的伺服器。

OpsWorks CM 伺服器會存放您的組態程式碼，例如 Chef 技術指南或 Puppet Enterprise 模組。雖然此程式碼存放在伺服器備份中， AWS 無法存取它。此內容會加密，只有您 AWS 帳戶中的管理員才能存取。我們建議您使用來源儲存庫建議的通訊協定來保護 Chef 或 Puppet 組態程式碼。例如，您可以限制 中儲存庫的許可 AWS CodeCommit，或遵循 GitHub 網站上的準則來保護 GitHub 儲存庫。

OpsWorks CM 不會使用客戶提供的內容來維護服務，或保留客戶日誌。有關您的 OpsWorks CM 伺服器的記錄會存放在您的帳戶中，Amazon S3 儲存貯體中。連線至 OpsWorks CM 伺服器的使用者 IP 地址由 記錄 AWS。

與 整合 AWS Secrets Manager

從 2021 年 5 月 3 日開始，當您在 OpsWorks CM 中建立新伺服器時，OpsWorks CM 會將伺服器的秘密存放在其中 AWS Secrets Manager。對於新伺服器，下列屬性會儲存為 Secrets Manager 中的秘密。

- Chef Automate 伺服器
  - HTTPS 私有金鑰（僅限不使用自訂網域的伺服器）
  - Chef Automate 管理密碼 (CHEF_AUTOMATE_ADMIN_PASSWORD)
- Puppet Enterprise 主控端
  - HTTPS 私有金鑰（僅限不使用自訂網域的伺服器）
  - Puppet 管理密碼 (PUPPET_ADMIN_PASSWORD)
  - Puppet r10k 遠端 (PUPPET_R10K_REMOTE)

對於未使用自訂網域的現有伺服器，Chef Automate 和 Puppet Enterprise 伺服器存放在 Secrets Manager 中的唯一秘密是 HTTPS 私有金鑰，因為這是在自動每週系統維護期間產生的。

OpsWorks CM 會自動在 Secrets Manager 中存放秘密，而且此行為無法使用者設定。

資料加密

AWS OpsWorks CM 會加密授權 AWS 使用者與其 AWS OpsWorks CM 伺服器之間的伺服器備份和通訊。不過，不會加密 CM 伺服器的根 Amazon EBS AWS OpsWorks 磁碟區。

靜態加密

AWS OpsWorks CM 伺服器備份會加密。不過，不會加密 CM 伺服器的根 Amazon EBS AWS OpsWorks 磁碟區。這不是使用者可設定的。

傳輸中加密

AWS OpsWorks CM 使用 HTTP 搭配 TLS 加密。如果使用者未提供已簽章的憑證， AWS OpsWorks CM 會預設為自我簽署的憑證來佈建和管理伺服器。我們建議您使用憑證授權單位 (CA) 所簽署的憑證。

金鑰管理

AWS Key Management Service CM 目前不支援客戶受管金鑰和 AWS AWS OpsWorks 受管金鑰。

CM AWS OpsWorks 的身分和存取管理

AWS Identity and Access Management (IAM) 是一種 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 OpsWorks CM 資源。IAM 是一項服務 AWS ，您可以免費使用。

主題

- 目標對象
- 使用身分來驗證
- 使用政策管理存取權
- CM AWS OpsWorks 如何與 IAM 搭配使用
- AWS OpsWorks CM 身分型政策範例
- CM Identity and Access AWS OpsWorks 故障診斷
- AWS for AWS OpsWorks Configuration Management 的 受管政策
- 中的跨服務混淆代理預防 AWS OpsWorks CM

目標對象

使用方式 AWS Identity and Access Management (IAM) 會有所不同，取決於您在 OpsWorks CM 中執行的工作。

服務使用者 – 如果您使用 OpsWorks CM 服務來執行您的任務，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 OpsWorks CM 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 OpsWorks CM 中的功能，請參閱 CM Identity and Access AWS OpsWorks 故障診斷。

服務管理員 – 如果您在公司負責 OpsWorks CM 資源，您可能可以完整存取 OpsWorks CM。您的任務是判斷服務使用者應存取哪些 OpsWorks CM 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搭配 OpsWorks CM 使用 IAM，請參閱 CM AWS OpsWorks 如何與 IAM 搭配使用。

IAM 管理員 – 如果您是 IAM 管理員，建議您了解撰寫政策以管理 OpsWorks CM 存取的詳細資訊。若要檢視您可以在 IAM 中使用的 OpsWorks CM 身分型政策範例，請參閱 AWS OpsWorks CM 身分型政策範例。

使用身分來驗證

驗證是 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。

視您身分的使用者類型而定，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)，以使用您的登入資料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 建議您使用多重要素驗證 (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。

IAM 使用者和群組

IAM 使用者是 中具有單一個人或應用程式特定許可 AWS 帳戶 的身分。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。

> **Warning**
>
> IAM 使用者具有長期憑證，這會造成安全風險。為了協助降低此風險，建議您只為這些使用者提供執行任務所需的許可，並在不再需要這些使用者時將其移除。

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，您可以直接將政策連接到資源（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 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，並結合 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》。
- 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 當與身分或資源建立關聯時， 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。

OpsWorks CM 支援您在 IAM 中建立並連接到使用者、角色或群組的自訂政策。

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。委託人可以包括帳戶、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。

OpsWorks CM 不支援以資源為基礎的政策。

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。

OpsWorks CM 不使用 ACLs。

其他政策類型

OpsWorks CM 不支援下列其他政策類型。

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 – 許可界限是一種進階功能，您可以在其中設定身分型政策可授予 IAM 實體 （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 是一種服務，用於分組和集中管理您企業擁有的多個 AWS 帳戶。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。如需有關 Organizations 和 SCP 的詳細資訊，請參閱《AWS Organizations 使用者指南》中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，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 在涉及多種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。

CM AWS OpsWorks 如何與 IAM 搭配使用

使用 IAM 管理 CM AWS OpsWorks 的存取權之前，您應該了解哪些 IAM 功能可與 CM AWS OpsWorks 搭配使用。若要取得 CM AWS OpsWorks 和其他 AWS 服務如何與 IAM 搭配使用的高階檢視，請參閱《AWS IAM 使用者指南》中的與 IAM 搭配使用的 服務。

主題

- AWS OpsWorks CM 身分型政策
- AWS OpsWorks CM 和資源型政策
- 以 AWS OpsWorks CM 標籤為基礎的授權
- AWS OpsWorks CM IAM 角色

AWS OpsWorks CM 身分型政策

透過 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及允許或拒絕動作的條件。 AWS OpsWorks CM 支援特定動作、資源和條件索引鍵。若要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

在 AWS OpsWorks CM 中，您可以將自訂政策陳述式連接至使用者、角色或群組。

動作

IAM 身分類型政策的 Action 元素會描述政策將允許或拒絕的特定動作。政策動作通常具有與相關聯 AWS API 操作相同的名稱。政策會使用動作來授予執行相關聯操作的許可。

CM AWS OpsWorks 中的政策動作在動作之前使用下列字首：opsworks-cm:。例如，若要授予某人使用 API 操作建立 AWS OpsWorks CM 伺服器的許可，請在其政策中包含 opsworks-cm:CreateServer 動作。政策陳述式必須包含 Action或 NotAction元素。 AWS OpsWorks CM 會定義自己的動作集，描述您可以使用此服務執行的任務。

若要在單一陳述式中指定多個動作，請用逗號分隔，如下所示：

```
"Action": [
      "opsworks-cm:action1",
      "opsworks-cm:action2"
```

您也可以使用萬用字元 (*) 來指定多個動作。例如，若要指定開頭是 Describe 文字的所有動作，請包含以下動作：

```
"Action": "opsworks-cm:Describe*"
```

當您使用萬用字元來允許政策陳述式中的多個動作時，請小心只允許對授權的服務或使用者執行這些動作。

若要查看 AWS OpsWorks CM 動作清單，請參閱《IAM 使用者指南》中的 AWS OpsWorks 的動作、資源和條件金鑰。

資源

Resource 元素可指定動作套用的物件。陳述式必須包含 Resource 或 NotResource 元素。您可以使用 ARN 來指定資源，或是使用萬用字元 (*) 來指定陳述式套用到所有資源。

您可以執行 或 DescribeServers DescribeBackups API 操作，以及這些資源的基礎資源層級政策，以取得 AWS OpsWorks CM 伺服器或備份的 Amazon Resource Number (ARN)。

AWS OpsWorks CM 伺服器資源具有下列格式的 ARN：

```
arn:aws:opsworks-cm:{Region}:${Account}:server/${ServerName}/${UniqueId}
```

AWS OpsWorks CM 備份資源具有下列格式的 ARN：

```
arn:aws:opsworks-cm:{Region}:${Account}:backup/${ServerName}-{Date-and-Time-Stamp-of-Backup}
```

如需 ARNs 格式的詳細資訊，請參閱 Amazon Resource Names (ARNs) AWS 和服務命名空間。

例如，若要在陳述式中指定 `test-chef-automate` Chef Automate 伺服器，請使用以下 ARN：

```
"Resource": "arn:aws:opsworks-cm:us-west-2:123456789012:server/test-chef-automate/EXAMPLE-d1a2bEXAMPLE"
```

若要指定屬於特定帳戶的所有 AWS OpsWorks CM 伺服器，請使用萬用字元 (*)：

```
"Resource": "arn:aws:opsworks-cm:us-west-2:123456789012:server/*"
```

下列範例指定 AWS OpsWorks CM 伺服器備份做為資源：

```
"Resource": "arn:aws:opsworks-cm:us-west-2:123456789012:backup/test-chef-automate-server-2018-05-20T19:06:12.399Z"
```

某些 AWS OpsWorks CM 動作，例如用於建立資源的動作，無法對特定資源執行。在這些情況下，您必須使用萬用字元 (*)。

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

許多 API 動作都涉及多個資源。若要在單一陳述式中指定多項資源，請使用逗號分隔 ARN。

```
"Resource": [
      "resource1",
      "resource2"
```

若要查看 AWS OpsWorks CM 資源類型及其 ARNs 的清單，請參閱《IAM 使用者指南》中的 AWS OpsWorks CM 的動作、資源和條件金鑰。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱《IAM 使用者指南》中的 AWS OpsWorks CM 的動作、資源和條件金鑰。

條件金鑰

AWS OpsWorks CM 沒有可在政策陳述式的 `Condition`元素中使用的服務特定內容索引鍵。如需所有 服務可用的全域內容金鑰清單，請參閱 IAM 政策參考中的AWS 全域條件內容金鑰。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。

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。您可以建置使用條件運算子的條件表達式 (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，只有在使用者使用使用者名稱標記時，您才能授予使用者存取資源的許可。如需詳細資訊，請參閱「IAM 使用者指南」中的 IAM 政策元素：變數和標籤。

範例

若要檢視 AWS OpsWorks CM 身分型政策的範例，請參閱 AWS OpsWorks CM 身分型政策範例。

AWS OpsWorks CM 和資源型政策

AWS OpsWorks CM 不支援以資源為基礎的政策。

以資源為基礎的政策是 JSON 政策文件，這些文件會指定指定的委託人可對資源以及在怎樣的條件下執行哪些動作。

以 AWS OpsWorks CM 標籤為基礎的授權

您可以將標籤連接至 AWS OpsWorks CM 資源，或在請求中將標籤傳遞至 AWS OpsWorks CM。若要根據標籤控制存取，您可以使用 aws:RequestTag/*key-name*或 條件索引鍵，在政策的條件元素中提供標籤資訊。 aws:TagKeys如需標記 AWS OpsWorks CM 資源的詳細資訊，請參閱本指南在 AWS OpsWorks for Puppet Enterprise 資源上使用標籤中的 在 AWS OpsWorks for Chef Automate 資源上使用標籤或 。

AWS OpsWorks CM IAM 角色

IAM 角色是您 AWS 帳戶中具有特定許可的實體。

AWS OpsWorks CM 使用兩個角色：

- 授予 CM AWS OpsWorks 服務許可以在使用者帳戶內運作的服務角色 AWS 。如果您使用 OpsWorks CM 提供的預設服務角色，此角色的名稱為 aws-opsworks-cm-service-role。
- 執行個體描述檔角色，可讓 AWS OpsWorks CM 服務呼叫 OpsWorks CM API。此角色會授予 Amazon S3 和 的存取權 AWS CloudFormation ，以建立伺服器和 S3 儲存貯體進行備份。如果

您使用 OpsWorks CM 提供的預設執行個體描述檔，則此執行個體描述檔角色的名稱為 `aws-opsworks-cm-ec2-role`。

AWS OpsWorks CM 不使用服務連結角色。

使用暫時登入資料搭配 AWS OpsWorks CM

AWS OpsWorks CM 支援使用臨時憑證，並從 繼承該功能 AWS Security Token Service。

您可以搭配聯合使用暫時憑證、擔任 IAM 角色，或是擔任跨帳戶角色。您可以透過呼叫 AssumeRole 或 GetFederationToken 等 AWS STS API 操作來取得臨時安全登入資料。

服務連結角色

AWS OpsWorks CM 不使用服務連結角色。

服務連結角色可讓 AWS 服務存取其他服務中的資源，以代表您完成 動作。服務連結角色會顯示在您的 IAM 帳戶中，並由該服務所擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

服務角色

此功能可讓服務代表您擔任服務角色。此角色可讓服務存取其他服務中的資源，以代表您完成動作。服務角色會出現在您的 IAM 帳戶中，且由該帳戶所擁有。這表示 IAM 管理員可以變更此角色的許可。不過，這樣可能會破壞此服務的功能。

AWS OpsWorks CM 使用兩個角色：

- 授予 CM AWS OpsWorks 服務許可以在使用者帳戶內運作的服務角色 AWS 。如果您使用 OpsWorks CM 提供的預設服務角色，則此角色的名稱為 `aws-opsworks-cm-service-role`。
- 執行個體描述檔角色，可讓 AWS OpsWorks CM 服務呼叫 OpsWorks CM API。此角色會授予 Amazon S3 和 的存取權 AWS CloudFormation ，以建立伺服器和 S3 儲存貯體進行備份。如果您使用 OpsWorks CM 提供的預設執行個體描述檔，則此執行個體描述檔角色的名稱為 `aws-opsworks-cm-ec2-role`。

在 CM 中選擇 AWS OpsWorks IAM 角色

當您在 CM AWS OpsWorks 中建立伺服器時，您必須選擇一個角色，以允許 AWS OpsWorks CM 代表您存取 Amazon EC2。如果您已建立服務角色，CM AWS OpsWorks 會提供您可供選擇的角色清單。如果您未指定角色，OpsWorks CM 可以為您建立角色。請務必選擇允許存取啟動和停止 Amazon EC2 執行個體的角色。如需詳細資訊，請參閱 建立 Chef Automate 伺服器 或 建立 Puppet Enterprise 主伺服器。

AWS OpsWorks CM 身分型政策範例

根據預設，使用者或角色沒有建立或修改 AWS OpsWorks CM 資源的許可。他們也無法使用 AWS Management Console AWS CLI或 AWS API 來執行任務。IAM 管理員必須建立 IAM 政策，授予 IAM 身分許可，以對他們所需的指定資源執行特定 API 操作。管理員接著必須將這些政策連接至需要這些許可的使用者或群組。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

在 AWS OpsWorks CM 中，您可以將`AWSOpsWorksCMServiceRole`政策指派給使用者，讓使用者使用 AWS Management Console 或 建立和管理 Chef Automate 或 Puppet Enterprise 伺服器 AWS CLI。

主題

- 政策最佳實務
- 允許使用者檢視自己的許可
- 根據標籤檢視 AWS OpsWorks CM 伺服器

政策最佳實務

身分型政策會判斷是否有人可以在您的帳戶中建立、存取或刪除 OpsWorks CM 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您的使用者和工作負載，請使用 AWS 受管政策，將許可授予許多常見使用案例。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 使用服務動作，您也可以使用條件來授予存取服務動作的權限 AWS 服務，例如 AWS CloudFormation。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access

Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您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) – 如果您有需要 IAM 使用者或 中根使用者的案例 AWS 帳戶，請開啟 MFA 以增加安全性。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。

允許使用者檢視自己的許可

此範例會示範如何建立政策，允許 使用者檢視連接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上完成此動作的許可，或使用 AWS CLI 或 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": "*"
        }
    ]
}
```

根據標籤檢視 AWS OpsWorks CM 伺服器

您可以在身分型政策中使用條件，根據標籤控制對 AWS OpsWorks CM 伺服器和備份的存取。此範例示範如何建立允許檢視 AWS OpsWorks CM 伺服器的政策。不過，只有在 AWS OpsWorks CM 伺服器標籤Owner具有該使用者使用者名稱的值時，才會授予許可。此政策也會授予在主控台完成此動作的必要許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListServersInConsole",
            "Effect": "Allow",
            "Action": "opsworks-cm:DescribeServers",
            "Resource": "*"
        },
        {
            "Sid": "ViewServerIfOwner",
            "Effect": "Allow",
            "Action": "opsworks-cm:DescribeServers",
            "Resource": "arn:aws:opsworks-cm:region:master-account-ID:server/server-
name",
            "Condition": {
                "StringEquals": {"opsworks-cm:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

您可以將此政策連接到您帳戶中的 使用者。如果名為 AWS OpsWorks 的使用者richard-roe嘗試檢視 CM 伺服器，伺服器必須加上標籤Owner=richard-roe或 owner=richard-roe。否則他便會被拒絕存取。條件標籤鍵 Owner 符合 Owner 和 owner，因為條件索引鍵名稱不區分大小寫。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。

CM Identity and Access AWS OpsWorks 故障診斷

使用下列資訊來協助您診斷和修正使用 IAM 時可能遇到的常見問題。如需 CM AWS OpsWorks 特定的疑難排解資訊，請參閱 故障診斷 AWS OpsWorks for Chef Automate和 故障診斷 Puppet Enterprise 的 OpsWorks 。

主題

- 我未獲授權在 CM AWS OpsWorks 中執行動作
- 我未獲得執行 iam：PassRole 的授權
- 我想要允許 AWS 帳戶外的人員存取我的 AWS OpsWorks CM 資源

我未獲授權在 CM AWS OpsWorks 中執行動作

如果 AWS Management Console 告訴您未獲授權執行 動作，則必須聯絡管理員尋求協助。您的管理員是為您提供簽署憑證的人員。

當使用者`mateojackson`嘗試使用主控台檢視 AWS OpsWorks CM 伺服器的詳細資訊，但沒有`opsworks-cm:DescribeServers`許可時，會發生下列範例錯誤。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 opsworks-cm:DescribeServers on resource: test-chef-automate-server
```

在此情況下，Mateo 會請求管理員更新政策，允許他使用 `opsworks-cm:DescribeServers` 動作存取 `test-chef-automate-server` 資源。

我未獲得執行 iam：PassRole 的授權

若您收到錯誤，告知您並未獲得執行 `iam:PassRole` 動作的授權，您必須聯絡您的管理員以取得協助。您的管理員是為您提供簽署憑證的人員。請該人員更新您的政策，以允許您將角色傳遞給 OpsWorks CM。

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 `marymajor` 的使用者嘗試使用主控台在 OpsWorks CM 中執行動作時，會發生下列範例錯誤。但是，動作要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 會請求管理員更新她的政策，允許她執行 `iam:PassRole` 動作。

我想要允許 AWS 帳戶外的人員存取我的 AWS OpsWorks CM 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：

- AWS OpsWorks CM 支援授予來自多個帳戶的使用者管理 AWS OpsWorks CM 伺服器。
- 若要了解如何在您擁有 AWS 的帳戶中提供資源的存取權，請參閱《IAM 使用者指南》中的為您擁有的另一個 AWS 帳戶中的 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》中的提供存取權給第三方擁有 AWS 的帳戶。
- 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)。
- 若要了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 角色與資源型政策的差異。

AWS for AWS OpsWorks Configuration Management 的 受管政策

若要將許可新增至使用者、群組和角色，使用 AWS 受管政策比自行撰寫政策更容易。建立 IAM 客戶受管政策需要時間和專業知識，而受管政策可為您的團隊提供其所需的許可。若要快速開始使用，您可以使用我們的 AWS 受管政策。這些政策涵蓋常見的使用案例，並且可在您的帳戶中使用 AWS 。如需受 AWS 管政策的詳細資訊，請參閱《IAM 使用者指南》中的AWS 受管政策。

AWS 服務會維護和更新 AWS 受管政策。您無法變更 AWS 受管政策中的許可。服務偶爾會在 AWS 受管政策中新增其他許可以支援新功能。此類型的更新會影響已連接政策的所有身分識別 (使用者、群組和角色)。當新功能啟動或新操作可用時，服務很可能會更新 AWS 受管政策。服務不會從 AWS 受管政策中移除許可，因此政策更新不會破壞您現有的許可。

此外， AWS 支援跨多個 服務之任務函數的受管政策。例如，ReadOnlyAccess AWS 受管政策提供所有 AWS 服務和資源的唯讀存取權。當服務啟動新功能時， 會為新操作和資源 AWS 新增唯讀許可。如需任務職能政策的清單和說明，請參閱 IAM 使用者指南中有關任務職能的AWS 受管政策。

AWS 受管政策： **AWSOpsWorksCMServiceRole**

您可以將 AWSOpsWorksCMServiceRole 連接到 IAM 實體。OpsWorks CM 也會將此政策連接至服務角色，允許 OpsWorks CM 代表您執行動作。

此政策授予##許可，允許 OpsWorks CM 管理員建立、管理和刪除 OpsWorks CM 伺服器和備份。

許可詳細資訊

此政策包含以下許可。

- opsworks-cm – 允許主體刪除現有的伺服器，並開始維護執行。
- acm – 允許主體從中刪除或匯入憑證 AWS Certificate Manager ，讓使用者連線至 OpsWorks CM 伺服器。
- cloudformation – 允許 OpsWorks CM 在主體建立、更新或刪除 OpsWorks CM 伺服器時建立和管理 AWS CloudFormation 堆疊。
- ec2 – 允許 OpsWorks CM 在主體建立、更新或刪除 OpsWorks CM 伺服器時啟動、佈建、更新和終止 Amazon Elastic Compute Cloud 執行個體。
- iam – 允許 OpsWorks CM 建立建立和管理 OpsWorks CM 伺服器所需的服務角色。
- tag – 允許主體從 OpsWorks CM 資源套用和移除標籤，包括伺服器和備份。
- s3 – 允許 OpsWorks CM 建立 Amazon S3 儲存貯體以存放伺服器備份、根據委託人請求管理 S3 儲存貯體中的物件（例如，刪除備份），以及刪除儲存貯體。
- secretsmanager – 允許 OpsWorks CM 建立和管理 Secrets Manager 秘密，以及從秘密套用或移除標籤。
- ssm – 允許 OpsWorks CM 在 OpsWorks CM 伺服器的執行個體上使用 Systems Manager Run Command。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket*"
            ],
            "Action": [
                "s3:CreateBucket",
                "s3:DeleteObject",
```

```
                "s3:DeleteBucket",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:PutBucketPolicy",
                "s3:PutObject",
                "s3:GetBucketTagging",
                "s3:PutBucketTagging"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Action": [
                "tag:UntagResources",
                "tag:TagResources"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Action": [
                "ssm:DescribeInstanceInformation",
                "ssm:GetCommandInvocation",
                "ssm:ListCommandInvocations",
                "ssm:ListCommands"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringLike": {
                    "ssm:resourceTag/aws:cloudformation:stack-name": "aws-opsworks-cm-
*"
                }
            },
            "Action": [
                "ssm:SendCommand"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:ssm:*::document/*",
                "arn:aws:s3:::amzn-s3-demo-bucket*"
            ],
            "Action": [
                "ssm:SendCommand"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Action": [
                "ec2:AllocateAddress",
                "ec2:AssociateAddress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateImage",
                "ec2:CreateSecurityGroup",
                "ec2:CreateSnapshot",
                "ec2:CreateTags",
                "ec2:DeleteSecurityGroup",
                "ec2:DeleteSnapshot",
                "ec2:DeregisterImage",
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAddresses",
                "ec2:DescribeImages",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeInstances",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSnapshots",
                "ec2:DescribeSubnets",
                "ec2:DisassociateAddress",
                "ec2:ReleaseAddress",
                "ec2:RunInstances",
                "ec2:StopInstances"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/aws:cloudformation:stack-name": "aws-opsworks-cm-
*"
                }
            },
            "Action": [
                "ec2:TerminateInstances",
                "ec2:RebootInstances"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:opsworks-cm:*:*:server/*"
            ],
            "Action": [
                "opsworks-cm:DeleteServer",
                "opsworks-cm:StartMaintenance"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:cloudformation:*:*:stack/aws-opsworks-cm-*"
            ],
            "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStackEvents",
                "cloudformation:DescribeStackResources",
                "cloudformation:DescribeStacks",
                "cloudformation:UpdateStack"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::*:role/aws-opsworks-cm-*",
                "arn:aws:iam::*:role/service-role/aws-opsworks-cm-*"
            ],
```

```
            "Action": [
                "iam:PassRole"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "*",
            "Action": [
                "acm:DeleteCertificate",
                "acm:ImportCertificate"
            ]
        },
        {
            "Effect": "Allow",
            "Resource": "arn:aws:secretsmanager:*:*:opsworks-cm!aws-opsworks-cm-
secrets-*",
            "Action": [
                "secretsmanager:CreateSecret",
                "secretsmanager:GetSecretValue",
                "secretsmanager:UpdateSecret",
                "secretsmanager:DeleteSecret",
                "secretsmanager:TagResource",
                "secretsmanager:UntagResource"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "ec2:DeleteTags",
            "Resource": [
                "arn:aws:ec2:*:*:instance/*",
                "arn:aws:ec2:*:*:elastic-ip/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        }
    ]
}
```

AWS 受管政策： **AWSOpsWorksCMInstanceProfileRole**

您可以將 AWSOpsWorksCMInstanceProfileRole 連接到 IAM 實體。OpsWorks CM 也會將此政策連接至服務角色，允許 OpsWorks CM 代表您執行動作。

此政策會授予##許可，允許用作 OpsWorks CM 伺服器的 Amazon EC2 執行個體從 AWS CloudFormation 和 取得資訊 AWS Secrets Manager，並將伺服器備份存放在 Amazon S3 儲存貯體中。

許可詳細資訊

此政策包含以下許可。

- acm – 允許 OpsWorks CM 伺服器 EC2 執行個體從 取得憑證 AWS Certificate Manager ，讓使用者連線至 OpsWorks CM 伺服器。
- cloudformation – 允許 OpsWorks CM 伺服器 EC2 執行個體在執行個體建立或更新程序期間取得 AWS CloudFormation 堆疊的相關資訊，並將 AWS CloudFormation 有關其狀態的訊號傳送至 。
- s3 – 允許 OpsWorks CM 伺服器 EC2 執行個體上傳和存放 S3 儲存貯體中的伺服器備份、視需要停止或復原上傳，以及刪除 S3 儲存貯體中的備份。
- secretsmanager – 允許 OpsWorks CM 伺服器 EC2 執行個體取得 OpsWorks CM 相關 Secrets Manager 秘密的值。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudformation:DescribeStackResource",
                "cloudformation:SignalResource"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ]
        },
        {
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:ListMultipartUploadParts",
                "s3:PutObject"
            ],
```

```
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket*",
            "Effect": "Allow"
        },
        {
            "Action": "acm:GetCertificate",
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:*:*:opsworks-cm!aws-opsworks-cm-
secrets-*",
            "Effect": "Allow"
        }
    ]
}
```

AWS 受管政策的 OpsWorks CM 更新

檢視自此服務開始追蹤這些變更以來，OpsWorks CM AWS 受管政策更新的詳細資訊。如需此頁面變更的自動提醒，請訂閱 OpsWorks CM 文件歷史記錄頁面上的 RSS 摘要。

變更	描述	日期
AWSOpsWorksCMInstanceProfileRole - 更新的受管政策	OpsWorks CM 已更新 受管政策，允許做為 OpsWorks CM 伺服器的 EC2 執行個體與 CloudFormation 和 Secrets Manager 共用資訊，並管理備份。變更會opsworks-cm! 新增至 Secrets Manager 秘密的資源名稱，以便允許 OpsWorks CM 擁有秘密。	2021 年 4 月 23 日
AWSOpsWorksCMServiceRole - 更新的受管政策	OpsWorks CM 已更新受管政策，允許 OpsWorks CM 管理員建立、管理和刪除 OpsWorks CM 伺服器和備份。變更會opsworks-cm! 新增至 Secrets Manager	2021 年 4 月 23 日

變更	描述	日期
	秘密的資源名稱，以便允許 OpsWorks CM 擁有秘密。	
OpsWorks CM 開始追蹤變更	OpsWorks CM 開始追蹤其 AWS 受管政策的變更。	2021 年 4 月 23 日

中的跨服務混淆代理預防 AWS OpsWorks CM

混淆代理人問題屬於安全性議題，其中沒有執行動作許可的實體可以強制具有更多許可的實體執行該動作。在 中 AWS，跨服務模擬可能會導致混淆代理人問題。在某個服務 (呼叫服務) 呼叫另一個服務 (被呼叫服務) 時，可能會發生跨服務模擬。可以操縱呼叫服務來使用其許可，以其不應有存取許可的方式對其他客戶的資源採取動作。為了預防這種情況， AWS 提供的工具可協助您保護所有服務的資料，而這些服務主體已獲得您帳戶中資源的存取權。

我們建議在資源政策中使用 aws:SourceArn和 aws:SourceAccount全域條件內容索引鍵，以限制將另一個 服務 AWS OpsWorks CM 提供給資源的許可。如果 aws:SourceArn 值不包含帳戶 ID (例如 Amazon S3 儲存貯體 ARN)，您必須使用這兩個全域條件內容金鑰來限制許可。如果同時使用這兩個全域條件內容金鑰，且 aws:SourceArn 值包含帳戶 ID，則在相同政策陳述式中使用 aws:SourceAccount 值和 aws:SourceArn 值中的帳戶時，必須使用相同的帳戶 ID。如果您想要僅允許一個資源與跨服務存取相關聯，則請使用 aws:SourceArn。如果您想要允許該帳戶中的任何資源與跨服務使用相關聯，請使用 aws:SourceAccount。

的值aws:SourceArn必須是 OpsWorks CM Chef 或 Puppet 伺服器的 ARN。

防範混淆代理人問題最有效的方式，是使用aws:SourceArn全域條件內容索引鍵搭配伺服器的完整 ARN AWS OpsWorks CM 。如果您不知道完整的 ARN，或如果您指定多個伺服器 ARNs，請將aws:SourceArn全域內容條件索引鍵與萬用字元 (*) 搭配使用，用於 ARN 的未知部分。例如：arn:aws:*servicename*:*:*123456789012*:*。

下一節說明如何在 中使用 aws:SourceArn和 aws:SourceAccount全域條件內容索引鍵 AWS OpsWorks CM ，以防止混淆代理人問題。

防止 中的混淆代理人入侵 AWS OpsWorks CM

本節說明如何協助防止混淆代理人入侵 AWS OpsWorks CM，並包含您可以連接到您用來存取之 IAM 角色的許可政策範例 AWS OpsWorks CM。作為安全最佳實務，我們建議您將 aws:SourceArn和

aws:SourceAccount條件金鑰新增至 IAM 角色與其他 服務的信任關係。信任關係 AWS OpsWorks CM 允許 擔任角色，在建立或管理您的 AWS OpsWorks CM 伺服器所需的其他服務中執行動作。

編輯信任關係以新增aws:SourceArn和aws:SourceAccount條件索引鍵

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在左側導覽窗格中，選擇 Roles (角色)。
3. 在搜尋方塊中，搜尋您用於存取的角色 AWS OpsWorks CM。 AWS 受管角色為 aws-opsworks-cm-service-role。
4. 在角色的摘要頁面上，選擇信任關係索引標籤。
5. 在 Trust relationships (信任關係) 標籤上，選擇 Edit trust relationship (編輯信任關係)。
6. 在政策文件中，將至少一個 aws:SourceArn或 aws:SourceAccount條件索引鍵新增至政策。使用 aws:SourceArn 來限制跨服務 （ 例如 AWS Certificate Manager 和 Amazon EC2) AWS OpsWorks CM 與特定 AWS OpsWorks CM 伺服器之間的信任關係，這更嚴格。新增 aws:SourceAccount 以限制跨服務與特定帳戶中 AWS OpsWorks CM 伺服器之間的信任關係，這較不嚴格。以下是範例。請注意，如果您使用兩個條件索引鍵，帳戶 IDs相同。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks-cm.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:opsworks-cm:us-east-2:123456789012:server/my-
opsworks-server/EXAMPLEabcd-1234-efghEXAMPLE-ID"
        }
      }
    }
  ]
}
```

7. 新增條件金鑰完成後，請選擇更新信任政策。

以下是使用 aws:SourceArn和 限制對 AWS OpsWorks CM 伺服器存取的其他角色範例aws:SourceAccount。

主題

- 範例：存取特定區域的 AWS OpsWorks CM 伺服器
- 範例：將多個伺服器 ARN 新增至 aws:SourceArn

範例：存取特定區域的 AWS OpsWorks CM 伺服器

下列角色信任關係陳述式會存取美國東部（俄亥俄）區域 () 中的任何 AWS OpsWorks CM 伺服器us-east-2。請注意，區域是在 ARN 值 中指定aws:SourceArn，但伺服器 ID 值是萬用字元 (*)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks-cm.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:opsworks-cm:us-east-2:123456789012:server/*"
        }
      }
    }
  ]
}
```

範例：將多個伺服器 ARN 新增至 **aws:SourceArn**

下列範例限制存取帳戶 ID 123456789012 中兩個 AWS OpsWorks CM 伺服器的陣列。

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks-cm.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": [
             "arn:aws:opsworks-cm:us-east-2:123456789012:server/my-chef-
server/unique_ID",
             "arn:aws:opsworks-cm:us-east-2:123456789012:server/my-puppet-
server/unique_ID"
           ]
       }
      }
    }
  ]
}
```

網際網路流量隱私權

AWS OpsWorks CM 使用與 AWS HTTPS 或 HTTP 搭配 TLS 加密一般使用的相同傳輸安全通訊協定。

在 CM AWS OpsWorks 中記錄和監控

AWS OpsWorks CM 會將所有 API 動作記錄到 CloudTrail。如需詳細資訊，請參閱下列主題：

- 使用 記錄 Puppet Enterprise API 呼叫的 OpsWorks AWS CloudTrail
- 使用 記錄 AWS OpsWorks for Chef Automate API 呼叫 AWS CloudTrail

AWS OpsWorks CM 的合規驗證

AWS OpsWorks CM 支援下列合規計劃和法規：

- 支付卡產業 (PCI)
- 1996 年美國健康保險流通與責任法案 (HIPAA)
- AWS 系統和組織控制 (SOC) 1、2 和 3
- 一般資料保護規則 (GDPR)

第三方稽核人員會在多個合規計畫中評估 AWS OpsWorks CM 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 AWS Artifact 中下載報告。

使用 AWS OpsWorks CM 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 提供下列資源以協助合規：

- 安全與合規快速入門指南 – 這些部署指南討論在 AWS上部署以安全及合規為重心基準環境的架構考量和步驟。
- 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 – 此工作手冊和指南的集合可能適用於您的產業和位置。
- AWS Config – AWS 此服務會評估您的資源組態是否符合內部實務、產業準則和法規。
- AWS Security Hub – AWS 此服務提供 內安全狀態的全面檢視 AWS ，可協助您檢查是否符合安全產業標準和最佳實務。

CM AWS OpsWorks 中的彈性

AWS OpsWorks 根據預設，CM 會在您建立伺服器時啟用伺服器的每日備份。備份會加密，並存放在 Amazon S3 儲存貯體中。根據預設，只有建立伺服器的帳戶才能存取此儲存貯體。您可以為其他使用者新增儲存貯體存取權，或自行決定在 Amazon S3 中設定跨區域備份。Chef 和 Puppet 支援跨區域加密，因為這兩項產品都會加密 AWS OpsWorks CM 伺服器和受管節點之間的流量。

AWS OpsWorks CM 不支援高可用性 (HA) 組態。

AWS 全域基礎設施是以 AWS 區域和可用區域為基礎。 AWS 區域提供多個實體分隔和隔離的可用區域，這些區域與低延遲、高輸送量和高備援聯網連接。透過可用區域，您所設計與操作的應用程式和資

料庫，就能夠在可用區域之間自動容錯移轉，而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

如需如何在 CM AWS OpsWorks 中備份和還原伺服器的詳細資訊，請參閱下列內容：

- 備份和還原 Puppet Enterprise Server 的 OpsWorks
- 備份和還原 AWS OpsWorks for Chef Automate 伺服器

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。

AWS OpsWorks CM 中的基礎設施安全

作為受管服務，AWS OpsWorks Configuration Management 受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及如何 AWS 保護基礎設施的相關資訊，請參閱AWS 雲端安全。若要使用基礎設施安全的最佳實務設計您的 AWS 環境，請參閱 Security Pillar AWS Well-Architected Framework 中的基礎設施保護。

您可以使用 AWS 發佈的 API 呼叫，透過網路存取 OpsWorks CM。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 OpsWorks CM 不支援私有連結或 VPC 私有端點。

AWS OpsWorks CM 不支援以資源為基礎的政策。如需詳細資訊，請參閱AWS Identity and Access Management 《使用者指南》中的AWS 使用 IAM 的服務。

CM AWS OpsWorks 中的組態和漏洞分析

AWS OpsWorks CM 會對在 AWS OpsWorks CM 伺服器上執行的作業系統執行定期核心和安全性更新。使用者可以為自動更新設定從目前日期算起最多兩週的時段。 AWS OpsWorks CM 會推送 Chef 和 Puppet Enterprise 次要版本的自動更新。如需設定 更新的詳細資訊 AWS OpsWorks for Chef Automate，請參閱本指南中的系統維護 (Chef)。如需設定 OpsWorks for Puppet Enterprise 更新的詳細資訊，請參閱本指南中的系統維護 (Puppet)。

CM AWS OpsWorks 的安全最佳實務

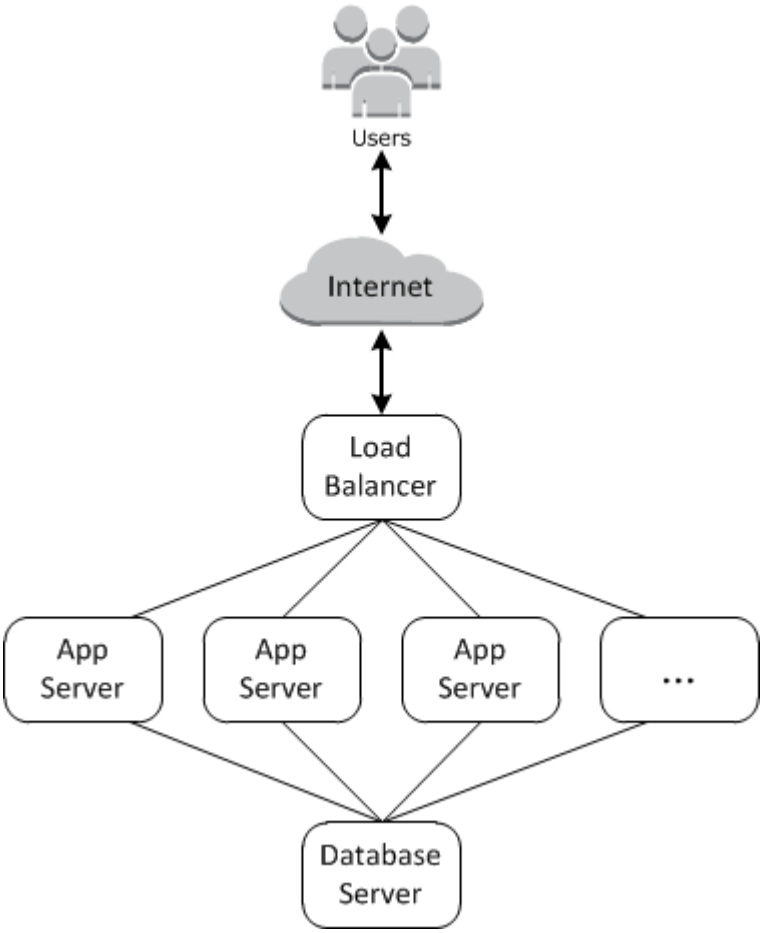
AWS OpsWorks CM 與所有 AWS 服務一樣，提供安全功能供您在開發和實作自己的安全政策時考慮。以下最佳實務為一般準則，並不代表完整的安全解決方案。這些最佳實務可能不適用或無法滿足您的環境需求，因此請將其視為實用建議就好，而不要當作是指示。

- 保護您的入門套件和下載的登入憑證。當您建立新的 AWS OpsWorks CM 伺服器或從 CM AWS OpsWorks 主控台下載新的入門套件和登入資料時，請將這些項目存放在至少需要一個驗證因素的安全位置。認證可提供您伺服器的系統管理員層級存取權。
- 保護您的組態程式碼。使用來源儲存庫建議的通訊協定來保護 Chef 或 Puppet 組態程式碼 (食譜和模組)。例如，您可以限制 AWS CodeCommit中儲存庫的許可權，或遵循 GitHub 網站上的指導方針來保護 GitHub 儲存庫。
- 使用 CA 簽署的憑證來連線到節點。雖然您可以在 AWS OpsWorks CM 伺服器上註冊或引導節點時使用自我簽署的憑證，但最佳實務是使用 CA 簽署的憑證。我們建議您使用憑證授權單位 (CA) 所簽署的憑證。
- 請勿與其他使用者共用 Chef 或 Puppet 管理主控台登入認證。管理員應為 Chef 或 Puppet 主控台網站的每個使用者建立個別的使用者。
  - 管理 Chef Automate 中的使用者
  - 管理 Puppet Enterprise 中的使用者
- 設定自動備份和系統維護更新。在 AWS OpsWorks CM 伺服器上設定自動維護更新，有助於確保您的伺服器執行最新的安全性相關作業系統更新。設定自動備份有助於簡化災難復原，並在發生事件或失敗時加快還原時間。限制對存放 CM 伺服器備份的 Amazon S3 AWS OpsWorks 儲存貯體的存取；不要授予對每個人的存取。視需要個別授予其他使用者的讀取或寫入存取權，或在 IAM 中為這些使用者建立安全群組，並將存取權指派給安全群組。
  - 系統維護 (Chef)
  - 系統維護 (Puppet)
  - 備份和還原 AWS OpsWorks for Chef Automate 伺服器
  - 備份和還原 Puppet Enterprise Server 的 OpsWorks
  - AWS Identity and Access Management 使用者指南中的建立您的第一個 IAM 委派使用者和群組
  - Amazon Simple Storage Service 開發人員指南中的 Amazon S3 安全最佳實務

AWS OpsWorks 堆疊

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

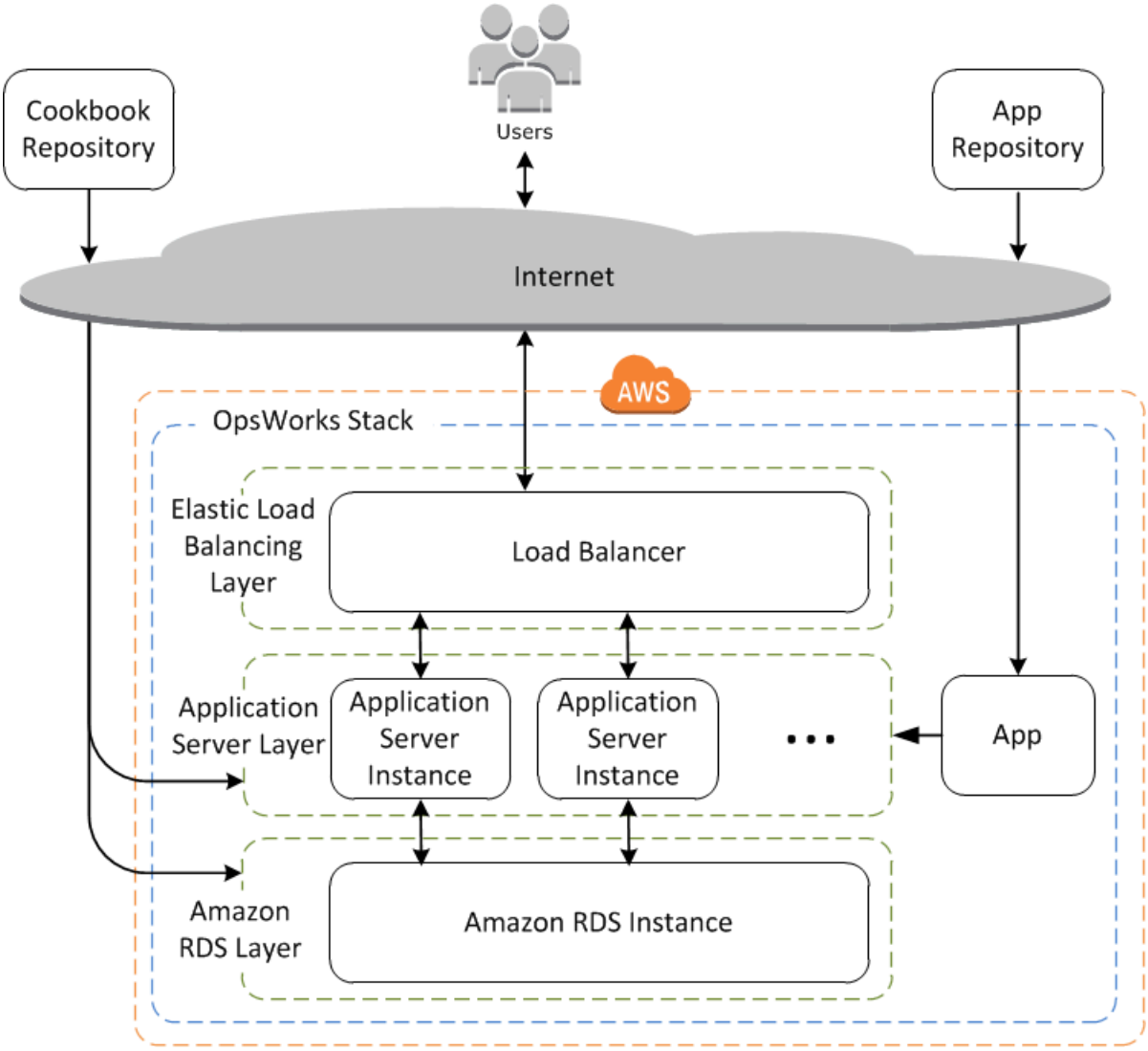
雲端型運算通常涉及 AWS 資源群組，例如 Amazon EC2 執行個體和 Amazon Relational Database Service (RDS) 執行個體，這些資源必須共同建立和管理。例如，web 應用程式通常需要應用程式伺服器、資料庫伺服器、負載平衡器等。這個執行個體群組通常稱為「堆疊」。一個簡易的應用程式伺服器堆疊看起來可能會如下。



除了建立執行個體和安裝必要的套件之外，您通常需要一種將應用程式分佈到應用程式伺服器、監控堆疊效能、管理安全及許可等的方式。

AWS OpsWorks Stacks 提供簡單且靈活的方法來建立和管理堆疊和應用程式。

以下是基本應用程式伺服器堆疊在 Stacks AWS OpsWorks 中的外觀。它包含一組在 Elastic Load Balancing 負載平衡器後方執行的應用程式伺服器，以及後端 Amazon RDS 資料庫伺服器。



雖然相對簡單，但此堆疊會顯示所有關鍵 Stacks AWS OpsWorks 功能。以下是它構成的方式。

主題

- 堆疊
- 層
- 配方和生命週期 (LifeCycle) 事件
- 執行個體
- 應用程式

- 自訂您的 Stack
- 資源管理
- 安全與許可
- 監控和記錄
- CLI、軟體開發套件和 AWS CloudFormation 範本
- AWS OpsWorks Stacks 生命週期結束FAQs
- 將您的 AWS OpsWorks Stacks 應用程式遷移至 AWS Systems Manager Application Manager
- 使用 AWS OpsWorks Stacks 分離就地工具
- Stacks AWS OpsWorks 入門
- AWS OpsWorks Stacks 最佳實務
- 堆疊
- 層
- 執行個體
- 應用程式
- 技術指南和配方
- 資源管理
- 標籤
- 監控
- 安全與許可
- AWS OpsWorks Chef 12 Linux 的 Stacks 支援
- 在 Stacks 中支援先前的 Chef AWS OpsWorks 版本
- 搭配其他 AWS 服務使用 AWS OpsWorks Stacks
- 使用 AWS OpsWorks Stacks CLI
- 偵錯和故障診斷指南
- AWS OpsWorks Stacks 代理程式 CLI
- AWS OpsWorks Stacks Data Bag 參考
- OpsWorks 代理程式變更

堆疊

堆疊是核心 AWS OpsWorks Stacks 元件。基本上，它是適用於 AWS 資源的容器，例如 Amazon EC2 執行個體、Amazon RDS 資料庫執行個體等，這些資源具有共同目的，並且應該一起進行邏輯管理。堆疊可協助您將這些資源做為群組管理，也可以讓您定義一些預設組態設定，例如執行個體的作業系統和 AWS 區域。若您希望隔離一些堆疊元件，避免與使用者直接互動，您可以在 VPC 中執行堆疊。

層

您可以藉由新增一或多個 layer 來定義堆疊的組成內容。layer 代表一組提供特定用途的 Amazon EC2 執行個體，例如服務應用程式或託管資料庫伺服器。

您可以透過修改套件的預設組態，新增 Chef 配方以執行像是安裝額外套件等的任務，來自訂或延伸 layer。

對於所有堆疊， AWS OpsWorks Stacks 包含服務層，代表下列 AWS 服務。

- Amazon Relational Database Service
- Elastic Load Balancing
- Amazon Elastic Container Service

Layer 可讓您完全控制要安裝的套件、其設定的方式、部署應用程式的方式等。

配方和生命週期 (LifeCycle) 事件

Layer 會根據 Chef 配方處理像是在執行個體上安裝套件、部署應用程式、執行指令碼等任務。Stacks AWS OpsWorks 其中一個主要功能是一組生命週期事件：設定、設定、部署、取消部署和關機，這些事件會在每個執行個體的適當時間自動執行一組指定的配方。

每一 layer 都可以為每個生命週期事件指派一組配方，處理該事件和 layer 的各種任務。例如，在屬於 Web 伺服器層的執行個體完成開機後， AWS OpsWorks Stacks 會執行下列動作。

1. 執行 layer 的安裝配方，執行像是安裝和設定 web 伺服器等的任務。
2. 執行 layer 的部署配方，將 layer 的應用程式從儲存庫部署到執行個體，並執行相關任務 (例如重新啟動服務)。
3. 在堆疊中的每個執行個體上執行設定配方，使每個執行個體可視需要調整自身的組態，以適應新的執行個體。

例如，在執行負載平衡器的執行個體上，設定配方可能會修改負載平衡器的組態，以包含新的執行個體。

如果執行個體屬於多個 layer， AWS OpsWorks Stacks 會為每個 layer 執行配方，例如，您可以擁有支援 PHP 應用程式伺服器和 MySQL 資料庫伺服器的執行個體。

如果您已實作配方，您可以將每個配方指派給適當的 layer 和事件，而 AWS OpsWorks Stacks 會在適當的時間自動為您執行這些配方。您也可以隨時手動執行配方。

執行個體

執行個體代表單一運算資源，例如 Amazon EC2 執行個體。它定義了資源的基本組態，例如作業系統和大小。其他組態設定，例如彈性 IP 地址或 Amazon EBS 磁碟區，是由執行個體的 layer 所定義。layer 的配方會透過執行像是安裝和設定套件及部署應用程式等任務，來完成組態。

您可以使用 AWS OpsWorks Stacks 建立執行個體並將其新增至 layer。當您啟動執行個體時， AWS OpsWorks Stacks 會使用執行個體及其 layer 指定的組態設定啟動 Amazon EC2 執行個體。Amazon EC2 執行個體完成開機後， AWS OpsWorks Stacks 會安裝處理執行個體與服務之間通訊的代理程式，並執行適當的配方以回應生命週期事件。

AWS OpsWorks Stacks 支援下列執行個體類型，其特性在於它們的啟動和停止方式。

- 全年無休執行個體為手動啟動，並會持續執行到您停止他們。
- 時間型執行個體由 AWS OpsWorks Stacks 依指定的每日和每週排程執行。

  他們可讓您的堆疊自動調整執行個體的數目，以適應可預測的用量模式。
- Stacks 會根據指定的負載指標自動啟動和停止負載型執行個體，例如 CPU AWS OpsWorks 使用率。

  他們可讓您的堆疊自動調整執行個體的數目，以適應傳入流量的變化。負載式執行個體僅適用於 Linux 式的堆疊。

AWS OpsWorks Stacks 支援執行個體自動修復。如果代理程式停止與服務的通訊， AWS OpsWorks Stacks 會自動停止並重新啟動執行個體。

您也可以將 Linux 型運算資源整合到在 Stacks AWS OpsWorks 外部建立的堆疊中。

- 您使用 Amazon EC2 主控台、CLI 或 API 直接建立的 Amazon EC2 執行個體。

- 在您自己的硬體上執行的「現場部署」執行個體，包括在虛擬機器中執行的執行個體。

註冊其中一個執行個體之後，該執行個體會成為 AWS OpsWorks Stacks 執行個體，而且您可以像使用 AWS OpsWorks Stacks 建立的執行個體一樣進行管理。

應用程式

您可以將應用程式和相關檔案存放在儲存庫中，例如 Amazon S3 儲存貯體。每個應用程式都會以一個「應用程式」代表，指定應用程式類型並包含從儲存庫將應用程式部署到您執行個體所需要的資訊，例如儲存庫 URL 和密碼。當您部署應用程式時， AWS OpsWorks Stacks 會觸發部署事件，在堆疊的執行個體上執行部署配方。

您可採用以下方式來部署應用程式：

- 自動 - 當您啟動執行個體時， AWS OpsWorks Stacks 會自動執行執行個體的部署配方。
- 手動 – 若您有新的應用程式，或希望更新現有的應用程式，您可以手動執行線上執行個體的部署配方。

您通常會讓 AWS OpsWorks Stacks 在整個堆疊上執行部署配方，以允許其他 layer 的執行個體適當地修改其組態。但是，舉例來說，若您希望在將應用程式部署到每個應用程式伺服器執行個體前測試新的應用程式，您可以將部署限制在執行個體一部分的子集。

自訂您的 Stack

AWS OpsWorks Stacks 提供各種自訂 layer 的方式，以滿足您的特定需求：

- 您可以藉由覆寫代表各種組態設定的屬性，或甚至覆寫用於建立組態檔案的範本，來修改 AWS OpsWorks Stacks 如何設定套件。
- 您可以透過提供您自己的配方，執行像是執行指令碼或安裝和設定非標準套件等任務來延伸現有的 layer。

所有堆疊可包含一或多個 layer。一開始的 layer 只會有最小的配方組。您透過實作配方，處理像是安裝套件、部署應用程式等任務，來將功能新增到 layer。您可以將自訂配方和相關檔案封裝在一或多個技術指南中，並將技術指南存放在 Amazon S3 或 Git 等儲存庫中。

您可以手動執行配方，但 AWS OpsWorks Stacks 也可讓您支援一組五個生命週期事件來自動化程序：

- Setup (安裝) 會在新的執行個體成功開機之後發生。
- Configure (設定) 會在執行個體進入或離開線上狀態時，在所有堆疊的執行個體上發生。
- Deploy (部署) 會在您部署應用程式時發生。
- Undeploy (解除部署) 會在您刪除應用程式時發生。
- Shutdown (關機) 會在您停止執行個體時發生。

每個 layer 都可以將任何數目的配方指派給每個事件。當 layer 執行個體上發生生命週期事件時， AWS OpsWorks Stacks 會執行相關聯的配方。例如，當應用程式伺服器執行個體上發生部署事件時， AWS OpsWorks Stacks 會執行 layer 的部署配方來下載應用程式或執行相關任務。

資源管理

您可以將其他 AWS 資源 (例如彈性 IP 地址) 併入您的堆疊。您可以使用 AWS OpsWorks Stacks 主控台或 API 向堆疊註冊資源、將已註冊資源連接至執行個體或從執行個體分離資源，以及將資源從一個執行個體移至另一個執行個體。

安全與許可

AWS OpsWorks Stacks 與 AWS Identity and Access Management (IAM) 整合，提供強大的方法來控制使用者存取 AWS OpsWorks Stacks 的方式，包括下列項目：

- 個別使用者如何與每個堆疊互動，例如是否可以建立堆疊資源，例如 layer 和執行個體，或者是否可以使用 SSH 或 RDP 連線到堆疊的 Amazon EC2 執行個體。
- Stacks AWS OpsWorks 如何代表您與 Amazon EC2 執行個體等 AWS 資源互動。
- 在 AWS OpsWorks Stacks 執行個體上執行的應用程式如何存取 AWS 資源，例如 Amazon S3 儲存貯體。
- 如何管理使用者的公有 SSH 金鑰和 RDP 密碼，以及連線到執行個體。

監控和記錄

AWS OpsWorks Stacks 提供多種功能，可協助您監控堆疊，並針對堆疊和任何配方的問題進行疑難排解。針對所有堆疊：

- AWS OpsWorks Stacks 提供一組適用於 Linux 堆疊的自訂 CloudWatch 指標，為了方便起見而摘要在監控頁面上。

AWS OpsWorks Stacks 支援 Windows 堆疊的標準 CloudWatch 指標。您可以使用 CloudWatch 主控台監控它們。

- CloudTrail 日誌，記錄您 AWS 帳戶中由 Stacks 或代表 AWS OpsWorks Stacks 發出的 API 呼叫。
- 事件日誌會列出您堆疊中的所有事件。
- Chef 日誌會詳細記載針對每個執行個體上每個生命週期事件進行的行為，例如執行了哪些配方，以及發生了哪些錯誤。

以 Linux 為基礎的堆疊也可以包含 Ganglia 主層，您可以用來收集和顯示堆疊中執行個體的詳細監控資料。

CLI、軟體開發套件和 AWS CloudFormation 範本

除了 主控台之外， AWS OpsWorks Stacks 還支援適用於多種語言的命令列界面 (CLI) 和 SDKs，可用於執行任何操作。考量這些功能：

- AWS OpsWorks Stacks CLI 是 AWS CLI 的一部分，可用來從命令列執行任何操作。

  AWS CLI 支援多個 AWS 服務，並且可安裝在 Windows、Linux 或 OS X 系統上。
- AWS OpsWorks Stacks 包含在適用於 Windows PowerShell 的 AWS 工具中，可用於從 Windows PowerShell 命令列執行任何操作。
- Stacks SDK AWS OpsWorks 包含在 AWS SDKs中，可由實作於下列中的應用程式使用：Java、JavaScript（瀏覽器型和 Node.js)、.NET、PHP、Python (boto) 或 Ruby。

您也可以使用 AWS CloudFormation 範本來佈建堆疊。如需一些範例，請參閱 AWS OpsWorks 程式碼片段。

AWS OpsWorks Stacks 生命週期結束FAQs

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。

主題

- 現有客戶將受到此生命週期結束的影響？
- 是否 AWS OpsWorks Stacks 接受新客戶？
- 我應該將現有堆疊遷移到哪裡？
- 如何在生命週期結束後保留現有的 Amazon EC2 執行個體？
- 生命週期結束會 AWS 區域 同時影響所有 嗎？
- 提供哪些層級的技術支援 AWS OpsWorks Stacks？
- 是否有任何新的功能版本 AWS OpsWorks Stacks？

現有客戶將受到此生命週期結束的影響？

現有客戶在 的生命週期結束日期 2024 年 5 月 26 日之前都不會受到影響 AWS OpsWorks Stacks。2024 年 5 月 26 日之後，客戶將無法使用 OpsWorks 主控台、API、CLI 和 CloudFormation 資源。

是否 AWS OpsWorks Stacks 接受新客戶？

否。 AWS OpsWorks Stacks 不再接受新客戶，目前只有現有客戶才能建立新的堆疊。

我應該將現有堆疊遷移到哪裡？

我們建議 AWS OpsWorks Stacks 客戶將工作負載遷移到 AWS Systems Manager ，以便他們能夠利用下列功能：

- 現代 Chef 版本
- SSM Agent
- Application Load Balancer
- 透過 Auto Scaling 群組增強擴展功能
- 能夠使用 EC2 啟動範本定義所需的主機特性
- 較新的執行個體類型
- 較新的 EBS 磁碟區類型

如需有關 Systems Manager 的資訊，請參閱 AWS Systems Manager 使用者指南。如需遷移至 的資訊 AWS Systems Manager，請參閱 將您的 AWS OpsWorks Stacks 應用程式遷移至 AWS Systems Manager Application Manager

如何在生命週期結束後保留現有的 Amazon EC2 執行個體？

達到生命週期終止日期後，您的 Amazon EC2 執行個體將保留在您的帳戶中，但您將無法再使用 OpsWorks Stacks 服務來控制和管理執行個體。

您可以使用 AWS OpsWorks Stacks 分離就地工具，將您的 OpsWorks 執行個體與 OpsWorks Stacks 服務分離。分離後，您可以使用 Amazon EC2 AWS Systems Manager或任何 EC2 相容方法來設定和管理執行個體。如需詳細資訊，請參閱使用 AWS OpsWorks Stacks 分離就地工具。

生命週期結束會 AWS 區域 同時影響所有 嗎？

是。OpsWorks 主控台、API、CLI 和 CloudFormation 資源將於 2024 年 5 月 26 日 AWS 區域 同時停止。如需 AWS OpsWorks Stacks 可用 AWS 區域 的清單，請參閱AWS 區域服務清單。

提供哪些層級的技術支援 AWS OpsWorks Stacks？

AWS 將持續為 AWS OpsWorks Stacks 該客戶提供相同層級的支援，直到生命週期結束日為止。如果您有問題或疑慮，可以透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

是否有任何新的功能版本 AWS OpsWorks Stacks？

否。由於服務即將生命週期結束，我們將不會發行任何新功能。不過，我們將繼續如預期進行安全性改善和管理 Amazon EC2 執行個體，直到生命週期結束日為止。

將您的 AWS OpsWorks Stacks 應用程式遷移至 AWS Systems Manager Application Manager

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。

您現在可以使用遷移指令碼，將 AWS OpsWorks Stacks 應用程式遷移至 Application Manager AWS Systems Manager，這是一種 功能。將 Stacks 應用程式遷移至 Systems Manager Application Manager 可讓您使用 中無法使用 AWS 的功能 AWS OpsWorks Stacks，例如 Graviton 等新的 Amazon EC2 執行個體類型、gp3 等新的 Amazon Elastic Block Store (EBS) 磁碟區、新的作業系統、與 Auto Scaling 群組的整合，以及應用程式負載平衡器。

透過此版本，您現在可以使用 Systems Manager Application Manager 提供的新執行個體索引標籤，監控和執行遷移執行個體的操作。您可以使用執行個體索引標籤，在一個位置檢視多個 AWS 執行個體。使用此標籤，您可以檢視執行個體運作狀態的相關資訊，並對問題進行疑難排解。如需使用執行個體索引標籤的詳細資訊，請參閱AWS Systems Manager 《 使用者指南》中的使用應用程式執行個體。

主題

- 指令碼的運作方式
- 先決條件
- 限制
- 開始使用
- 常見問答集
- 故障診斷

指令碼的運作方式

AWS OpsWorks 提供您可以使用 CloudFormation 範本執行的指令碼，將 AWS OpsWorks Stacks 應用程式遷移至 Systems Manager Application Manager。指令碼會取得現有 OpsWorks 層的相關資訊，並根據指令碼`--provision-application`參數的值，佈建應用程式的複製，或提供您可以使用的入門 CloudFormation 範本 AWS CloudFormation。

先決條件

- 請確定 AWS CLI 已安裝並設定 。如需安裝 的詳細資訊 AWS CLI，請參閱AWS Command Line Interface 《 使用者指南》中的安裝或更新最新版本的 AWS CLI。

  > **Note**
  >
  > 如果您不想設定 AWS CLI，也可以使用 執行命令 AWS CloudShell。如需使用 CloudShell 的詳細資訊，請參閱AWS CloudShell 《 使用者指南》中的使用 AWS CloudShell 。

- 請確定 Python 3.6 版或更新版本已安裝或隨附 Amazon Machine Image (AMI)。
- 請確定您的作業系統受到支援。您可以在下列作業系統下載並執行遷移指令碼。
  - Amazon Linux 和 Amazon Linux 2
  - Ubuntu 18.04 LTS、20.04 LTS、22.04 LTS
  - Red Hat Enterprise Linux 8

- Windows Server 2019、Windows 10 企業版

> **Note**
>
> 不支援 Windows Server 2022。

限制

新的 OpsWorks 架構與 的架構不同 AWS OpsWorks Stacks。本節說明此架構的已知限制。

新 OpsWorks 架構不支援下列項目。

- 在 Windows 和 CentOS 執行個體上執行 Chef 配方
- 內建 Chef 11 層和 Berkshelf
- Chef 屬性和資料包
- 現場部署執行個體
- 從 EC2 匯入的執行個體
- 不支援安裝使用者指定的作業系統套件清單
- 不支援或遷移應用程式

以下支援有限制的 。

- 遷移指令碼會複製 EBS 磁碟區資訊，但不包含掛載點和磁碟區中包含的實際資料。
- 時間型和負載型擴展執行個體會遷移，但與這些執行個體相關聯的任何擴展規則都不會遷移。您可以修改 Auto Scaling 群組，以達到類似的結果。
- 在 OpsWorks 主控台的堆疊許可頁面中定義的 IAM 實體不會建立或產生。
- 遷移指令碼只能在 Systems Manager 中佈建單層應用程式。例如，如果您在同一堆疊中為兩層執行指令碼兩次，您會在 Systems Manager 中取得兩個不同的應用程式。

開始使用

遷移指令碼 是 Python 指令碼`stack_exporter.py`，您可以在本機或在 EC2 執行個體上執行。執行指令碼之前，請確定符合所有先決條件。如需先決條件的詳細資訊，請參閱 先決條件。

以下各節中的步驟說明如何將 OpsWorks 堆疊遷移至 Systems Manager Application Manager。

主題

- 步驟 1：準備您的環境以執行指令碼
- 步驟 2：下載遷移指令碼
- 步驟 3：設定您的環境以執行指令碼
- 步驟 4：執行指令碼
- 步驟 5：佈建 CloudFormation 堆疊
- 步驟 6：檢閱佈建的資源
- 步驟 7：啟動執行個體
- 步驟 8：檢閱執行個體
- 步驟 9：使用 Systems Manager Application Manager 在執行個體上監控和執行操作

步驟 1：準備您的環境以執行指令碼

為您的作業系統執行適當的命令，以準備您的環境。

主題

- Amazon Linux 2
- Amazon Linux
- Ubuntu 18.04、20.04、22.04
- Red Hat Enterprise Linux 8
- Windows Server 2019、Windows 10 企業版

Amazon Linux 2

```
sudo su
python3 -m pip install pipenv
PATH="$PATH:/usr/local/bin"
yum update
yum install git
```

Amazon Linux

```
sudo su
PATH="$PATH:/usr/local/bin"
export LC_ALL=en_US.utf-8
```

```
export LANG=en_US.utf-8
yum update
yum list | grep python3
yum install python36 // Any python version
yum install git
```

對於 Python 3.6 版， 也會執行：

```
python3 -m pip install pipenv==2022.4.8
```

對於 Python 3.7 版和更新版本，也執行：

```
python3 -m pip install pipenv
```

Ubuntu 18.04、20.04、22.04

```
sudo su
export PATH="${HOME}/.local/bin:$PATH"
apt-get update
apt install python3-pip
apt-get install git // if git is not installed
python3 -m pip install --user pipenv==2022.4.8
```

Red Hat Enterprise Linux 8

```
sudo su
sudo dnf install python3
PATH="$PATH:/usr/local/bin"
yum update
yum install git
python3 -m pip install pipenv==2022.4.8
```

Windows Server 2019、Windows 10 企業版

> **Note**
>
> 針對 Windows Server 2019，安裝 Python 3.6.1 版或更新版本。

```
pip install pipenv
```

如果尚未安裝 Git，請下載並安裝 Git。

如果您使用 Git 做為技術指南來源，請在 Windows 上執行指令碼之前，將 Git 伺服器新增至 known_hosts 檔案。您可以使用 PowerShell 建立下列函數。

```
function add_to_known_hosts($server){
    $new_host=$(ssh-keyscan $server 2> $null)
    $existing_hosts=''
    if (!(test-path "$env:userprofile\.ssh")) {
        md "$env:userprofile\.ssh"
    }
    if ((test-path "$env:userprofile\.ssh\known_hosts")) {
        $existing_hosts=Get-Content "$env:userprofile\.ssh\known_hosts"
    }
    $host_added=0
    foreach ($line in $new_host) {
        if (!($existing_hosts -contains $line)) {
            Add-Content -Path "$env:userprofile\.ssh\known_hosts" -Value $line
            $host_added=1
    }
   }
   if ($host_added) {
       echo "$server has been added to known_hosts."
   } else {
       echo "$server already exists in known_hosts."
   }
}
```

然後，您可以在執行函數時提供 Git 伺服器 （ 例如 github.com、git-codecommit.*repository_region*.amazonaws.com)。

```
add_to_known_hosts "myGitServer"
```

步驟 2：下載遷移指令碼

執行下列命令，下載包含遷移指令碼和所有相關檔案的 zip 檔案。

```
aws s3api get-object \
    --bucket export-opsworks-stacks-bucket-prod-us-east-1 \
    --key export_opsworks_stacks_script.zip export_opsworks_stacks_script.zip
```

如果您使用的是 Linux，請使用下列命令安裝 unzip 公用程式。

```
sudo apt-get install unzip
sudo yum install unzip
```

使用適用於您作業系統的適當命令解壓縮檔案。

針對 Linux，請使用下列命令。

```
unzip export_opsworks_stacks_script.zip
```

對於 Windows，請在 PowerShell 中使用 `Expand-Archive`命令。

```
Expand-Archive -LiteralPath PathToZipFile -DestinationPath PathToDestination
```

檔案解壓縮後，即可使用下列目錄和檔案。

- README.md
- 授權
- NOTICE
- requirements.txt
- 範本/
  - OpsWorksCFNTemplate.yaml
  - MountEBSVolumes.yaml
- opsworks/
- cloudformation/
- instance_tab/
- cfn_stack_deployer.py
- s3.py
- stack_exporter_context.py
- stack_exporter.py

步驟 3：設定您的環境以執行指令碼

使用下列命令將環境設定為執行指令碼。

```
pipenv install -r requirements.txt
```

```
pipenv shell
```

> **Note**
>
> 目前，指令碼只能在 Application Manager 中佈建單層應用程式。例如，如果您在同一堆疊中為兩個層執行指令碼兩次，則該指令碼會在 Application Manager 中建立兩個不同的應用程式。

設定環境後，請檢閱指令碼參數。您可以執行 `python3 stack_exporter.py --help`命令來檢視遷移指令碼的可用選項。

參數	描述	必要	Type	預設值
`--layer-id`	匯出此 OpsWorks layer ID 的 CloudFormation 範本。	是	string	
`--region`	OpsWorks 堆疊 AWS 的區域。如果您的 OpsWorks 堆疊區域和 API 端點區域不同，請使用堆疊區域。這與 OpsWorks 堆疊的其他資源部分（例如 EC2 執行個體和子網路）相同。	否	string	us-east-1
`--provision-application`	根據預設，指令碼會佈建 CloudFormation 範本匯出的應用程式。將此參數傳遞至值為 FALSE 的指令碼，以略過 CloudFormation 範本的佈建。	否	Boolean	`TRUE`
`--launch-template`	此參數定義是使用現有的啟動範本，還是建立新的啟動範本。您可以建立新的啟動範本，使用建議的執行個體屬性，或使用符合線上執行個體的執行個體屬性。 有效值包含：	否	string	`RECOMMENDED`

參數	描述	必要	Type	預設值
	• RECOMMENDED - 針對 OpsWorks 堆疊的作業系統和 c5.large 執行個體大小，使用來自最新 AMI 的執行個體特性。 • MATCH_LAST_INSTANCE - 使用最新可用的線上執行個體特性。 • *LaunchTemplateID* / [*LaunchTemplateVersion*] - 使用現有的啟動範本。或者，您可以提供範本版本。如果您不提供範本版本，則指令碼會使用預設版本。			
--system-updates	定義執行個體開機時是否要執行核心和套件更新。 有效值包含： • ALL_UPDATES - 在執行個體開機時，執行核心和套件的系統更新。 • NO_UPDATES - 執行個體開機時， 不會執行系統更新。 • MATCH_LAYER_SETTINGS - 使用 OpsWorks layer 或執行個體的 InstallUpdatesOnBoot 屬性來決定是否安裝系統更新。	否	string	ALL_UPDATES
--http-username	Systems Manager SecureString 參數的名稱，該參數會存放用於向包含自訂技術指南的 HTTP 封存進行身分驗證的使用者名稱。	否	string	

參數	描述	必要	Type	預設值
`--http-password`	Systems Manager `SecureString` 參數的名稱，該參數會存放用於向包含自訂技術指南的 HTTP 封存進行身分驗證的密碼。	否	string	
`--repo-private-key`	Systems Manager `SecureString` 參數的名稱，該參數會存放用於向包含自訂技術指南的儲存庫進行身分驗證的 SSH 金鑰。如果儲存庫位於 GitHub，您必須產生新的 `Ed25519` SSH 金鑰。如果您未產生新的 `Ed25519` SSH 金鑰，則與 GitHub 儲存庫的連線會失敗。	否	string	
`--lb-type`	遷移現有負載平衡器時要建立的負載平衡器類型，如果有的話。 有效值包含： • `ALB` (Application Load Balancer) • `Classis` (Classic Load Balancer) • `None` （如果您不想建立負載平衡器）	否	string	`ALB`
`--lb-access-logs-path`	儲存負載平衡器存取日誌的現有 S3 儲存貯體和字首路徑。S3 儲存貯體和負載平衡器必須位於相同的區域。如果您未提供值，且`--lb-type` 參數值設定為 `None`，則指令碼會建立新的 S3 儲存貯體和字首。請確定此字首有適當的儲存貯體政策。	否	string	

參數	描述	必要	Type	預設值
`--enable-instance-protection`	如果設定為 TRUE ，指令碼會為您的 Auto Scaling 群組建立自訂終止政策 (Lambda 函數）。具有`protected_instance` 標籤的 EC2 執行個體會受到保護，免於縮減事件。將`protected_instance` 標籤新增至您要保護免於縮減事件的每個 EC2 執行個體。	否	Boolean	FALSE
`--command-logs-bucket`	用來存放 AWS ApplyChef Recipe 和 MountEBSVolumes 日誌的現有 S3 儲存貯體名稱。如果您不提供值，指令碼會建立新的 S3 儲存貯體。	否	string	`aws-opsworks-application-manager-logs-`*account-id*
`--custom-json-bucket`	儲存自訂 JSON 的現有 S3 儲存貯體名稱。如果您不提供值，指令碼會建立新的 S3 儲存貯體。	否	string	`aws-apply-chef-application-manager-transition-data-`*account-id*

備註：

- 如果您使用私有 GitHub 儲存庫，則必須為 SSH 建立新的`Ed25519`主機金鑰。這是因為 GitHub 變更了 SSH 中支援的金鑰，並移除了未加密的 Git 通訊協定。如需`Ed25519`主機金鑰的詳細資訊，請參閱 GitHub 上的 Git 部落格文章改善 GitHub通訊協定安全性。產生新的`Ed25519`主機金鑰後，請為 SSH 金鑰建立 Systems Manager `SecureString` 參數，並使用 `SecureString` 參數名稱做為 `--repo-private-key` 參數的值。如需如何建立 Systems Manager `SecureString` 參數的詳細資訊，請參閱AWS Systems Manager 《 使用者指南》中的建立 SecureString 參數 (AWS CLI) 或建立 Systems Manager 參數 （主控台）。

- `--http-username`、 `--http-password`和 `--repo-private-key` 參數是指 Systems Manager SecureString 參數的名稱。當您執行`AWS-ApplyChefRecipes`文件時，遷移指令碼會使用這些參數。
- `--http-username` 參數要求您也為 `--http-password` 參數指定值。
- `--http-password` 參數要求您也為 `--http-username` 參數指定值。
- 請勿同時設定 `--http-password`和 的值`--repo-private-key`。提供 SSH 金鑰 (`--repo-private-key`) 的 Systems Manager SecureString 參數名稱，或儲存庫使用者名稱 (`--http-username`) 和密碼 ()`--http-password`。

步驟 4：執行指令碼

當您執行 時`python3 stack_exporter.py`，您可以佈建應用程式，或將 `--provision-application` 參數的值設定為 來建立入門範本`FALSE`。

範例 1：佈建 Systems Manager Application Manager 應用程式

下列命令會取得現有 OpsWorks 層的相關資訊，並使用較新的 OpsWorks 架構佈建應用程式，其結果類似於為堆疊設定的 Chef 版本。指令碼會使用 CloudFormation 佈建所有必要的資源，例如 Auto Scaling 群組，然後在 Systems Manager Application Manager 中註冊應用程式。

將 *`stack-region`* 和 *`layer-id`* 取代為 OpsWorks 堆疊和 layer 的值。

```
python3 stack_exporter.py \
     --layer-id layer-id \
     --region stack-region
```

範例 2：產生範本

下列命令會取得現有 OpsWorks 層的相關資訊，並產生 CloudFormation 範本。如果佈建範本，則 會達成類似於使用 Chef 14 的結果。在此範例中，不會佈建資源，因為 `--provision-application` 參數設定為 `FALSE`。

將 *`stack-region`* 和 *`layer-id`* 取代為 OpsWorks 堆疊和 layer 的值。

```
python3 stack_exporter.py \
    --layer-id layer-id \
    --region stack-region \
    --provision-application FALSE
```

執行 命令後，您可以在 Systems Manager 的 Application Manager 範本程式庫中檢閱範本，也可以佈建範本。如需檢視範本程式庫的詳細資訊，請參閱AWS Systems Manager 《 使用者指南》中的使用範本程式庫。

步驟 5：佈建 CloudFormation 堆疊

> **Note**
>
> 只有在您將指令碼的 `--provision-application` 參數設定為 時，才需要完成此步驟FALSE。

當您以 值指定 `--provision-application` 參數時FALSE，指令碼輸出會提供 CloudFormation 範本的名稱和 URL。此範本代表現有 OpsWorks 堆疊和層的建議替換。

您可以使用 Application Manager 範本程式庫 （ 建議 ） 或使用 CloudFormation 佈建範本。如需使用範本程式庫的詳細資訊，請參閱AWS Systems Manager 《 使用者指南》中的使用範本程式庫。

步驟 6：檢閱佈建的資源

您現在可以檢閱佈建的資源。

1. 使用 AWS CloudFormation 主控台檢閱佈建堆疊的資源。
   a. 在 https：//https://console.aws.amazon.com/cloudformation 開啟 AWS CloudFormation 主控台，然後選擇 Stacks。
   b. 在堆疊頁面上，選擇堆疊，然後選擇資源索引標籤。
   c. 在資源索引標籤上，檢閱堆疊的列出的資源。資源清單包含 EC2 Auto Scaling 群組，您可以在 Auto Scaling 主控台中檢閱該群組，或 AWS CLI。
2. 使用 Systems Manager Application Manager 檢閱應用程式的 資源。
   a. 在 https：//https://console.aws.amazon.com/systems-manager/ 開啟 Systems Manager 主控台。
   b. 在導覽窗格中，選擇 Application Manager。
   c. 在應用程式區段中，選擇自訂應用程式。Application Manager 會開啟概觀索引標籤。
   d. 選擇 Resources (資源) 標籤。資源索引標籤會顯示已為您的 OpsWorks 堆疊和 layer 遷移的所有資源。應用程式名稱包含 OpsWorks 堆疊的名稱，格式為 *app-stack-*

name-suffix，*####*代表堆疊 ID 的前六個字元。如需在 Application Manager 中檢視資源的詳細資訊，請參閱AWS Systems Manager 《使用者指南》中的檢視應用程式資源。

步驟 7：啟動執行個體

佈建執行個體之後，您就可以測試執行個體。此時，沒有執行個體正在執行。

若要讓執行個體上線，請將 Auto Scaling 群組的 `Min`、 `Max`和 `Desired capacity`值調整為對您的應用程式有意義的數字。一開始，您可能想要將這些值設定為 1，讓單一執行個體上線，並確認執行個體執行所有預期的動作，包括執行您的自訂 Chef 配方。

步驟 8：檢閱執行個體

啟動執行個體之後，請確認其如預期般執行。

1. 檢閱位於指令碼`--command-logs-bucket`參數所指定 S3 儲存貯體中的 Chef `startup`和`terminate`日誌。根據預設，日誌會存放在名為 的儲存貯體中`aws-opsworks-application-manager-logs-`*`account-id`*。

   a. 登入 AWS Management Console ，並在 https://console.aws.amazon.com/s3/：// 開啟 Amazon S3 主控台。

   b. 選擇包含日誌的儲存貯體。

   c. 導覽至`ApplyChefRecipes`字首以檢視您的日誌。

2. 檢查 Application Load Balancer 連線和運作狀態。

   執行下列步驟來檢視負載平衡器的存取日誌。您可以使用指令碼的 `--lb-access-logs-path` 參數，指定要存放負載平衡器存取日誌的 S3 儲存貯體。

   a. 登入 AWS Management Console ，並在 https://console.aws.amazon.com/s3/：// 開啟 Amazon S3 主控台。

   b. 選擇您的 S3 儲存貯體，然後導覽至包含日誌的字首。

3. 驗證執行個體是否通過所有 Auto Scaling 和 Application Load Balancer 運作狀態檢查 （如果您已設定）。

   您可以在新的執行個體索引標籤上檢視 Auto Scaling 運作狀態的相關資訊。

   a. 在 https：//https://console.aws.amazon.com/systems-manager/ 開啟 Systems Manager 主控台。

b. 在導覽窗格中，選擇 Application Manager。

c. 在應用程式區段中，選擇自訂應用程式。

d. 在清單中選擇應用程式。Application Manager 會開啟概觀索引標籤。

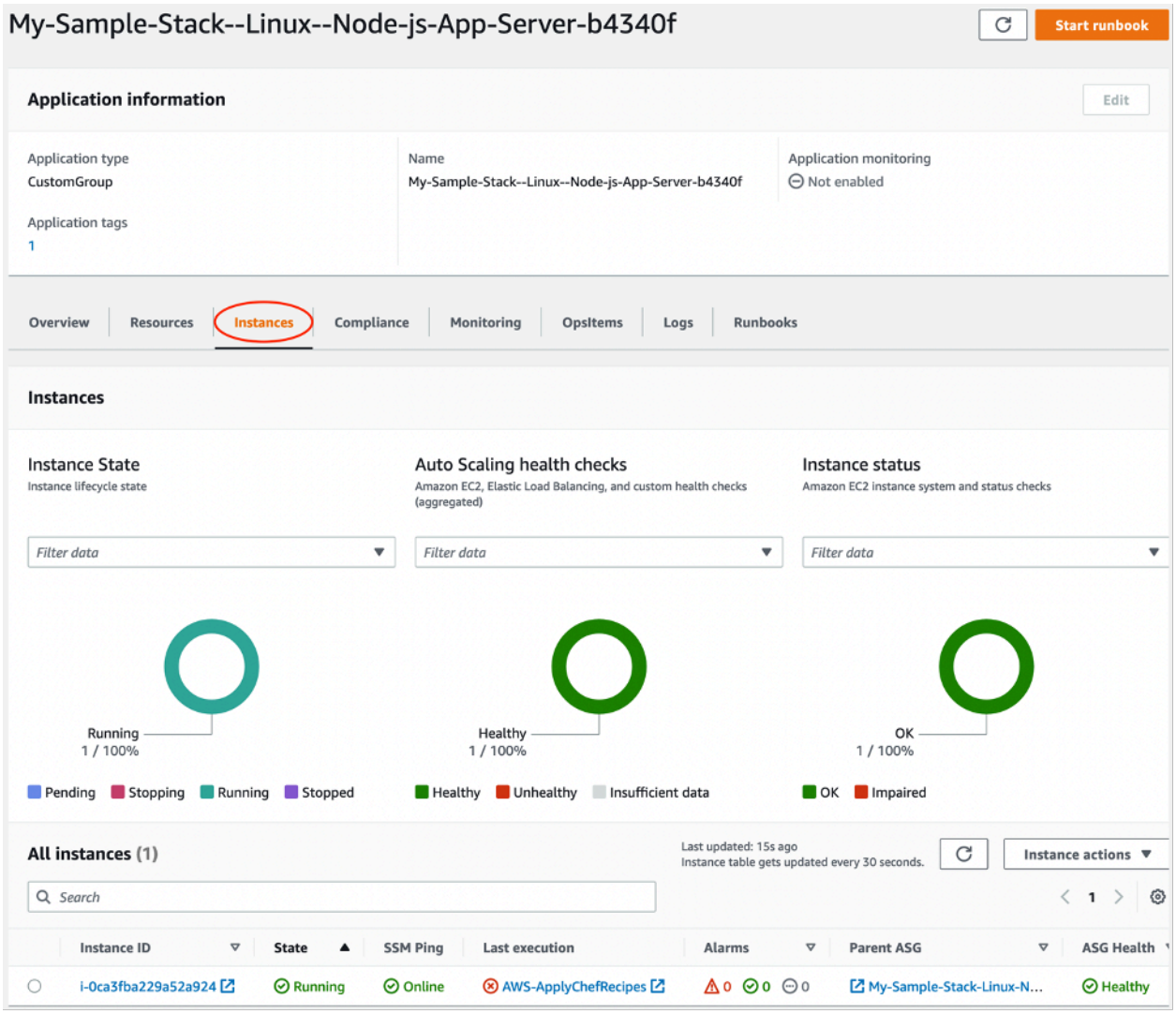
e. 選擇執行個體索引標籤以檢視 Auto Scaling 運作狀態的相關資訊。

驗證 Chef 配方是否成功執行後，您可以減少 Auto Scaling 群組容量來終止執行個體。如果您有任何自訂終止配方，請確認配方可如預期運作。

步驟 9：使用 Systems Manager Application Manager 在執行個體上監控和執行操作

您現在可以使用 Application Manager 頁面上的新執行個體索引標籤來監控和執行執行個體上的操作。如需使用執行個體索引標籤的詳細資訊，請參閱AWS Systems Manager 《 使用者指南》中的使用應用程式執行個體。

您可以使用執行個體索引標籤，在一個位置檢視多個 AWS 執行個體。使用此索引標籤，您可以檢視執行個體運作狀態的相關資訊，並對問題進行疑難排解。



執行下列步驟來檢視執行個體索引標籤。

1. 在 https：//https://console.aws.amazon.com/systems-manager/ 開啟 Systems Manager 主控台。
2. 在導覽窗格中，選擇 Application Manager。
3. 在應用程式區段中，選擇自訂應用程式。
4. 在清單中選擇應用程式。Application Manager 會開啟概觀索引標籤。
5. 選擇執行個體索引標籤，以檢視執行個體狀態和 EC2 運作狀態的相關資訊。

常見問答集

下列FAQs提供一些常見問題的答案。

主題

- 我可以遷移哪些 AWS OpsWorks Stacks 版本？
- 我的遷移執行個體可以使用哪些 Chef 版本？
- 我可以遷移哪些儲存庫類型？
- 我可以繼續使用私有 Git 儲存庫嗎？
- 我可以使用哪些 SSH 金鑰來存取執行個體？
- 為什麼我的執行個體會自動向內和向外擴展？
- 我可以關閉 Auto Scaling 嗎？
- 我可以在啟動的 EC2 執行個體上執行核心和套件更新嗎？
- 為什麼我執行個體中的 EBS 磁碟區不包含任何資料？
- 為什麼我的啟動範本中未掛載 EBS 磁碟區？
- 在哪裡可以找到 Chef 配方和掛載 EBS 磁碟區日誌？
- 哪裡可以找到遷移指令碼的偵錯日誌？
- 遷移指令碼是否支援 CloudFormation 範本版本控制？
- 我可以遷移多個層嗎？
- 如何建立 SecureString 參數？
- 如何保護新 Auto Scaling 群組中的執行個體免於終止事件？
- 遷移指令碼可使用哪些負載平衡器？
- 自訂技術指南設定配方是否已遷移？
- 我可以在新建立的執行個體上執行部署和取消部署配方嗎？
- 我可以變更 Auto Scaling 群組跨越哪些子網路嗎？

我可以遷移哪些 AWS OpsWorks Stacks 版本？

您只能遷移 Chef 11.10 和 Chef 12、Amazon Linux、Amazon Linux 2、Ubuntu 和 Red Hat Enterprise Linux 7 堆疊。

我的遷移執行個體可以使用哪些 Chef 版本？

遷移的執行個體可以使用 Chef 第 11 版到第 14 版。

> **Note**
>
> 不支援 Windows 堆疊遷移。

我可以遷移哪些儲存庫類型？

您可以遷移 S3、Git 和 HTTP 儲存庫類型。

我可以繼續使用私有 Git 儲存庫嗎？

可以，您可以繼續使用私有 Git 儲存庫。

如果您使用私有 GitHub 儲存庫，則必須為 SSH 建立新的Ed25519主機金鑰。這是因為 GitHub 變更了 SSH 中支援的金鑰，並移除了未加密的 Git 通訊協定。如需Ed25519主機金鑰的詳細資訊，請參閱 GitHub 上的 Git 部落格文章改善 GitHub通訊協定安全性。產生新的Ed25519主機金鑰後，請為此 SSH 金鑰建立 Systems Manager SecureString 參數，並使用 參數名稱做為 --repo-private-key 參數的值。如需如何建立 Systems Manager SecureString 參數的詳細資訊，請參閱AWS Systems Manager 《 使用者指南》中的建立 SecureString 參數 (AWS CLI)。

對於任何其他 Git 儲存庫類型，請為此 SSH 金鑰建立 Systems Manager SecureString 參數，並使用 參數名稱做為指令碼--repo-private-key參數的值。

我可以使用哪些 SSH 金鑰來存取執行個體？

當您執行指令碼時，指令碼會遷移堆疊中設定的 SSH 金鑰和執行個體。您可以使用 SSH 金鑰來存取執行個體。如果為堆疊和執行個體提供 SSH 金鑰，則指令碼會使用來自堆疊的金鑰。如果您不確定要使用哪些 SSH 金鑰，請在 EC2 主控台 (https://console.aws.amazon.com/ec2/：//) 中檢視執行個體。EC2 主控台中的詳細資訊頁面會顯示執行個體的 SSH 金鑰。

為什麼我的執行個體會自動向內和向外擴展？

Auto Scaling 會根據 Auto Scaling 群組的擴展規則來擴展執行個體。您可以設定群組的最小、最大和所需容量值。當您更新這些值時，Auto Scaling 群組會自動相應地擴展您的容量。

我可以關閉 Auto Scaling 嗎？

您可以將 Auto Scaling 群組的最小、最大和所需容量值設定為相同的數字，以關閉 Auto Scaling。例如，如果您想要一律有十個執行個體，請將最小、最大和所需容量值設定為十。

我可以在啟動的 EC2 執行個體上執行核心和套件更新嗎？

根據預設，核心和套件會在 EC2 執行個體開機時更新。使用下列步驟，在啟動的 EC2 執行個體上執行核心或套件更新。例如，您可能想要在執行部署或設定配方之後套用更新。

1. 連線至 EC2 執行個體。
2. 建立下列perform_upgrade函數，並在您的執行個體上執行。

```
perform_upgrade() {
    #!/bin/bash
    if [ -e '/etc/system-release' ] || [ -e '/etc/redhat-release' ]; then
     sudo yum -y update
    elif [ -e '/etc/debian_version' ]; then
     sudo apt-get update
     sudo apt-get dist-upgrade -y
    fi
}
perform_upgrade
```

3. 在核心和套件更新之後，您可能需要重新啟動 EC2 執行個體。若要檢查是否需要重新啟動，請建立下列reboot_if_required函數，並在 EC2 執行個體上執行。

```
reboot_if_required () {
 #!/bin/bash
 if [ -e '/etc/debian_version' ]; then
   if [ -f /var/run/reboot-required ]; then
     echo "reboot is required"
   else
     echo "reboot is not required"
   fi
 elif [ -e '/etc/system-release' ] || [ -e '/etc/redhat-release' ]; then
  export LC_CTYPE=en_US.UTF-8
  export LC_ALL=en_US.UTF-8
  LATEST_INSTALLED_KERNEL=`rpm -q --last kernel | perl -X -pe 's/^kernel-(\S+).*/
$1/' | head -1`
  CURRENTLY_USED_KERNEL=`uname -r`
  if [ "${LATEST_INSTALLED_KERNEL}" != "${CURRENTLY_USED_KERNEL}" ];then
     echo "reboot is required"
  else
     echo "reboot is not required"
  fi
 fi
}
reboot_if_required
```

4. 如果執行 reboot_if_required會產生reboot is required訊息，請重新啟動 EC2 執行個體。如果您收到reboot is not required訊息，則不需要重新啟動 EC2 執行個體。

為什麼我執行個體中的 EBS 磁碟區不包含任何資料？

當您執行指令碼時，指令碼會遷移 EBS 磁碟區的組態，為 OpsWorks 堆疊和層建立替代架構。指令碼不會遷移實際執行個體或執行個體中包含的資料。指令碼只會遷移層層級的 EBS 磁碟區組態，並將空的 EBS 磁碟區連接至啟動的 EC2 執行個體。

執行下列步驟，從您先前執行個體的 EBS 磁碟區提取資料。

1. 拍攝先前執行個體 EBS 磁碟區的快照。如需建立快照的詳細資訊，請參閱《Amazon EC2 使用者指南》中的建立 Amazon EBS 快照。 Amazon EC2
2. 從快照建立磁碟區。如需從快照建立磁碟區的詳細資訊，請參閱《Amazon EC2 使用者指南》中的從快照建立磁碟區。
3. 將您建立的磁碟區連接至執行個體。如需連接磁碟區的詳細資訊，請參閱《Amazon EC2 使用者指南》中的將 Amazon EBS 磁碟區連接至執行個體。 Amazon EC2

為什麼我的啟動範本中未掛載 EBS 磁碟區？

如果您為具有 EBS 磁碟區的 `--launch-template` 參數提供啟動範本 ID，則指令碼會連接 EBS 磁碟區，但不會掛載磁碟區。您可以執行為啟動的 EC2 執行個體建立指令碼的 `MountEBSVolumes` RunCommand 文件，來掛載連接的 EBS 磁碟區。

如果您未設定`--launch-template`參數，指令碼會建立範本，當 Auto Scaling 群組啟動新的 EC2 執行個體時，Auto Scaling 群組會自動連接 EBS 磁碟區，然後執行 `SetupAutomation`命令，將連接的磁碟區掛載到層設定中設定的掛載點。

在哪裡可以找到 Chef 配方和掛載 EBS 磁碟區日誌？

OpsWorks 會將日誌交付到 S3 儲存貯體，您可以為 `--command-logs-bucket` 參數提供值來指定。預設 S3 儲存貯體名稱的格式為：`aws-opsworks-stacks-application-manager-logs-`*`account-id`*。Chef 配方日誌存放在`ApplyChefRecipes`字首中。掛載 EBS 磁碟區日誌存放在 `MountEBSVolumes` 字首中。從堆疊遷移的所有圖層都會將日誌交付至相同的 S3 儲存貯體。

> **Note**
>
> - S3 儲存貯體的生命週期組態包含 30 天後刪除日誌的規則。如果您想要將日誌保留超過 30 天，您必須在 S3 儲存貯體的生命週期組態中更新規則。
> - 目前，OpsWorks 只會記錄 Chef `setup`和`terminate`配方。

哪裡可以找到遷移指令碼的偵錯日誌？

指令碼會將偵錯日誌放在名為 的儲存貯體中`aws-opsworks-stacks-transition-logs-`*`account-id`*。您可以在 S3 儲存貯體的`migration_script`資料夾中找到偵錯日誌，這些日誌位於與您遷移層名稱相符的資料夾下。

遷移指令碼是否支援 CloudFormation 範本版本控制？

指令碼會產生 CloudFormation 類型的 Systems Manager 文件，為您要遷移的 layer 或堆疊建立替換。即使具有相同的參數， 仍會再次執行指令碼，匯出先前匯出的層範本的新版本。範本版本與指令碼日誌存放在相同的 S3 儲存貯體中。

我可以遷移多個層嗎？

指令碼的 `--layer-id` 參數會在單一圖層中傳遞。若要遷移多個圖層，請重新執行指令碼並傳遞不同的 `--layer-id`。

屬於相同 OpsWorks 堆疊的圖層會列在 Application Manager 中的相同應用程式下。

1. 在 https：//https://console.aws.amazon.com/systems-manager/ 開啟 Systems Manager 主控台。
2. 在導覽窗格中，選擇 Application Manager。
3. 在應用程式區段中，選擇自訂應用程式。
4. 選擇您的應用程式。應用程式名稱開頭為 app-*`stack-name`*-*`first-six-characters-stack-id`*。
5. 最上層元素從應用程式開始，會顯示對應至 OpsWorks 堆疊的所有元件。這包括對應至 OpsWorks layer 的元件。
6. 選擇與 layer 對應的元件，以檢視 layer 的資源。代表 OpsWorks 層的元件也會以個別應用程式的形式從自訂應用程式區段中顯示。

如何建立 **SecureString** 參數？

您可以使用 Systems Manager 來建立`SecureString`參數。如需如何建立 Systems Manager `SecureString` 參數的詳細資訊，請參閱AWS Systems Manager 《 使用者指南》中的建立 SecureString 參數 (AWS CLI) 或建立 Systems Manager 參數 （ 主控台 ）。

您必須提供 `SecureString` 參數做為 `--http-username`、 `--http-password`或 `--repo-private-key` 參數的值。

如何保護新 Auto Scaling 群組中的執行個體免於終止事件？

您可以將 `--enable-instance-protection` 參數設定為 ，TRUE並將protected_instance標籤索引鍵新增至您要保護免於終止事件的每個 EC2 執行個體，以保護執行個體。當您將 `--enable-instance-protection` 參數設定為 TRUE並新增protected_instance標籤金鑰時，指令碼會將自訂終止政策新增至新的 Auto Scaling 群組，並暫停ReplaceUnhealthy程序。具有protected_instance標籤金鑰的執行個體會受到保護，免於發生下列終止事件：

- 在事件中擴展
- 執行個體重新整理
- 重新平衡
- 執行個體生命週期上限
- 允許列出執行個體終止
- 終止和取代運作狀態不佳的執行個體

> **Note**
>
> 您必須在要保護的執行個體上設定protected_instance標籤金鑰。標籤索引鍵區分大小寫。無論標籤值為何，任何具有該標籤索引鍵的執行個體都會受到保護。
> 若要縮短自訂終止政策的執行時間，您可以更新函數default_sample_size程式碼變數的值，增加 Lambda 函數用來篩選受保護執行個體的預設執行個體數量。預設值為 15。如果您增加 `default_sample_size`，您可能需要增加配置給 Lambda 函數的記憶體，這會增加 Lambda 函數的成本。如需 AWS Lambda 定價的資訊，請參閱 AWS Lambda 定價。

遷移指令碼可使用哪些負載平衡器？

指令碼提供三個負載平衡器選項。

- （建議）建立新的 Application Load Balancer。根據預設，指令碼會建立新的 Application Load Balancer。您也可以將 `--lb-type` 參數設定為 ALB。如需 Application Load Balancer 的詳細資訊，請參閱 Elastic Load Balancing 使用者指南中的什麼是 Application Load Balancer？。
- 如果 Application Load Balancer 不是 選項，請將 `--lb-type` 參數設定為 來建立 Classic Load BalancerClassic。如果您選取此選項，連接至 OpsWorks 層的現有 Classic Load Balancer 會與您的應用程式分開。如需 Application Load Balancer 的詳細資訊，請參閱 Elastic Load Balancing Load Balancing：Classic Load Balancer 使用者指南中的什麼是 Classic Load Balancer？。

- 您可以將 `--lb-type` 參數設定為 ，以連接現有的負載平衡器None。

> **Important**
>
> 建議您為 AWS OpsWorks Stacks 層建立新的 Elastic Load Balancing 負載平衡器。如果您選擇使用現有的 Elastic Load Balancing 負載平衡器，您應該先確認它不是用於其他用途，而且沒有連接的執行個體。在負載平衡器連接到 layer 之後，OpsWorks 會移除任何現有的執行個體，並設定負載平衡器，使其僅處理 layer 的執行個體。雖然在技術上可以使用 Elastic Load Balancing 主控台或 API 在將負載平衡器連接到 layer 之後修改負載平衡器的組態，但您不應該這麼做；變更不會是永久的。

將現有的 OpsWorks layer 負載平衡器連接至 Auto Scaling 群組

1. 執行遷移指令碼，並將 `--lb-type` 參數設為 None。當值設為 時None，指令碼不會複製或建立負載平衡器。
2. 指令碼部署 CloudFormation 堆疊後，更新 Auto Scaling 群組`MinMax`和`Desired capacity`值，然後測試您的應用程式。
3. 選擇`Link to the template`顯示在指令碼的輸出中。如果您關閉終端機，請採取這些步驟來存取範本。

   a. 在 https：//https://console.aws.amazon.com/systems-manager/ 開啟 Systems Manager 主控台。

   b. 在導覽窗格中，選擇 Application Manager。

   c. 選擇 CloudFormation 堆疊，然後選擇範本程式庫。

   d. 選擇由我擁有，並找到您的範本。
4. 從 CloudFormation 範本中，從動作功能表中選擇編輯。
5. 更新 CloudFormation 範本`ApplicationAsg`資源區段中的 `LabelBalancerNames` 屬性。

```
ApplicationAsg:
   DependsOn: CustomTerminationLambdaPermission
   Properties:
   #(other properties in ApplicationAsg to remain unchanged)
      LoadBalancerNames:
        - load-balancer-name
      HealthCheckType: ELB
```

6. 如果您希望 Auto Scaling 群組執行個體的運作狀態檢查也使用負載平衡器的運作狀態檢查，請移除以下區段`HealthCheckType`並輸入 `ELB`。如果您只需要 EC2 運作狀態檢查，則不需要變更範本。
7. 儲存您的變更。儲存會建立新的範本預設版本。如果這是您第一次執行 layer 的指令碼，以及第一次在主控台中儲存變更，則較新版本為 2。
8. 從動作 中，選擇佈建堆疊。
9. 確認您想要使用範本的預設版本。請確定選取現有的堆疊，然後選擇要更新的 CloudFormation 堆疊。
10. 針對每個後續頁面選擇下一步，直到您看到檢閱和佈建頁面為止。在檢閱和佈建頁面上，選擇我同時確認 AWS CloudFormation 可能會使用自訂名稱建立 IAM 資源，而且我了解所選範本中的變更可能會導致 AWS CloudFormation 更新或移除現有 AWS 資源。
11. 選擇 Provision stack (佈建堆疊)。

如果您需要復原更新，請執行下列步驟。

1. 選擇動作，然後選擇佈建堆疊。
2. 選擇挑選其中一個現有版本，然後選擇先前的範本版本。
3. 選擇選取現有堆疊，然後選擇要更新的 CloudFormation 堆疊。

自訂技術指南設定配方是否已遷移？

設定自訂技術指南不支援在設定事件期間執行。指令碼會遷移自訂技術指南來設定配方，並為您建立 Systems Manager Automation Runbook。不過，您必須手動執行配方。

執行下列步驟來執行您的設定配方。

1. 在 https：//https://console.aws.amazon.com/systems-manager/ 開啟 Systems Manager 主控台。
2. 在導覽窗格中，選擇 Application Manager。
3. 在應用程式區段中，選擇自訂應用程式。
4. 選擇您的應用程式。應用程式名稱開頭為 `app-`*`stack-name`*。
5. 選擇資源，然後選擇設定 Runbook。
6. 選擇執行自動化。
7. 選擇您要執行設定配方的執行個體 IDs，然後選擇執行。

我可以在新建立的執行個體上執行部署和取消部署配方嗎？

指令碼可以根據您的 layer 組態建立三個可能的 Automation Runbook。

- 設定
- 設定
- 終止

指令碼也可以建立下列 Systems Manager 參數，其中包含`AWS-ApplyChefRecipes Run Command`文件的輸入值。

- 設定
- 部署
- 設定
- Undeploy (解除部署)
- 終止

發生橫向擴展事件時，安裝自動化執行手冊會自動執行。這包括從原始 OpsWorks 層設定和部署自訂技術指南配方。發生縮減事件時，終止 Automation Runbook 會自動執行。終止 Automation Runbook 包含來自原始 OpsWorks 層的關機配方。

如果您想要手動執行取消部署或設定配方，請執行下列步驟。

1. 在 https：//https://console.aws.amazon.com/systems-manager/ 開啟 Systems Manager 主控台。
2. 在導覽窗格中，選擇 Application Manager。
3. 在應用程式區段中，選擇自訂應用程式。
4. 選擇您的應用程式。應用程式名稱開頭為 app-*stack-name*-*first-six-characters-stack-id*。Application Manager 會開啟概觀索引標籤。
5. 選擇資源，然後選擇設定 Automation Runbook。
6. 選擇執行自動化。
7. 如需 `applyChefRecipesPropertiesParameter` Automation Runbook 輸入參數，請參閱正確的 Systems Manager 參數。Systems Manager 參數名稱遵循格式 `/ApplyChefRecipes-Preset/`*`OpsWorks-stack-name`*-*`OpsWorks-layer-name`*-*`first-six-characters-stack-id`*`/event` ，其中`##`的值為 `Configure`、 或 `Deploy`，`Undeploy`取決於您要執行的配方。

8. 選擇您要執行配方的執行個體 IDs，然後選擇執行。

我可以變更 Auto Scaling 群組跨越哪些子網路嗎？

根據預設，Auto Scaling 群組會跨越 OpsWorks 堆疊 VPC 中的所有子網路。若要更新要跨越哪些子網路，請執行下列步驟。

1. 選擇`Link to the template`顯示在指令碼的輸出中。如果您關閉終端機，請採取這些步驟來存取範本。

   a. 在 https：//https://console.aws.amazon.com/systems-manager/ 開啟 Systems Manager 主控台。

   b. 在導覽窗格中，選擇 Application Manager。

   c. 選擇 CloudFormation 堆疊，然後選擇範本程式庫。

   d. 選擇由我擁有，並找到您的範本。
2. 從動作中，選擇佈建堆疊。
3. 確認您想要使用預設範本。選擇選取現有堆疊，然後選擇要更新的 CloudFormation 堆疊。

   > **Note**
   >
   > 如果您執行的指令碼的`--provision-application`參數設為 `FALSE`，則必須建立新的 CloudFormation 堆疊。

4. 針對 `SubnetIDs` 參數，提供您希望 Auto Scaling 群組跨越的以逗號分隔的子網路 IDs 清單。
5. 選擇下一步，直到您看到檢閱和佈建頁面為止。
6. 在檢閱和佈建頁面上，選擇我確認 AWS CloudFormation 可能會使用自訂名稱建立 IAM 資源，而且我了解所選範本中的變更可能會導致 AWS CloudFormation 更新或移除現有 AWS 資源。
7. 選擇 Provision stack (佈建堆疊)。

故障診斷

本節包含一些常見問題，以及這些問題的建議解決方案。

主題

- 提供的主體無效
- 啟用 Auto Scaling 群組保護的執行個體時，無法刪除 CloudFormation 堆疊

- 提供現有 S3 儲存貯體和字首時存取遭拒錯誤

提供的主體無效

問題：您收到錯誤訊息，指出您提供的委託人無效。

原因：這是因為 Auto Scaling 群組沒有服務角色。

解決方案：在發生錯誤的區域中建立 Auto Scaling 群組。建立 Auto Scaling 群組會為您的自訂終止政策建立必要的服務連結角色。

啟用 Auto Scaling 群組保護的執行個體時，無法刪除 CloudFormation 堆疊

問題： `--enable-instance-protection` 參數設定為 TRUE，且 Auto Scaling 群組的部分 EC2 執行個體受到`protected_instance`標籤金鑰的保護，這會導致您的 AWS CloudFormation 堆疊無法完全刪除。

原因：EC2 執行個體具有`protected_instance`標籤索引鍵，可保護它們免於發生終止事件。

解決方案：從 EC2 執行個體移除`protected_instance`標籤金鑰。這可讓 Auto Scaling 群組縮減規模。Auto Scaling 群組縮減後，您可以刪除 AWS CloudFormation 堆疊。

提供現有 S3 儲存貯體和字首時存取遭拒錯誤

問題：當您提供現有的 S3 儲存貯體和字首時，會收到`AccessDenied`錯誤。

原因：S3 儲存貯體政策不提供將負載平衡器日誌交付至儲存貯體的必要許可。

解決方案：更新 S3 儲存貯體政策，以允許指令碼將負載平衡器存取日誌交付至儲存貯體。如需如何更新儲存貯體政策的詳細資訊，請參閱《Elastic Application Load Balancer》中的啟用 Application Load Balancer 的存取日誌。 Elastic Load Balancing

使用 AWS OpsWorks Stacks 分離就地工具

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。

本節說明如何使用就地 AWS OpsWorks Stacks 分離工具，將您的 OpsWorks 執行個體與 OpsWorks Stacks 服務分離。

您分離的執行個體將保留在 中 AWS 帳戶，但您將不再能夠使用 OpsWorks 管理它們。反之，您將使用 Amazon EC2 AWS Systems Manager或任何 EC2 相容方法來設定和管理執行個體。

在高階，分離程序涉及下列步驟：

1. 此工具會執行驗證檢查，以確保資源已準備好進行分離。
2. 工具會從 OpsWorks 堆疊匯出自訂 JSON，並將其儲存為 Amazon S3 中的物件。
3. 此工具會建立 Systems Manager Automation 文件，代表每個 OpsWorks Stacks 生命週期事件。
4. 此工具會為正在分離的所有執行個體建立 AWS Service Catalog AppRegistry 目錄，並將任何 Elastic Load Balancing (ELB) 負載平衡器從 OpsWorks 層分離。
5. 最後，工具會分離和取消註冊其他資源，包括 Amazon Relational Database Service (Amazon RDS) 執行個體。

程序的運作方式

就地分離工具提供以下 3 個命令和類似體驗的精靈，引導您完成一系列步驟來檢查和設定執行個體，然後再繼續分離您的 layer。

Command	描述
`handle-prerequisites`	此命令會分析 layer 中的所有執行個體是否符合分離資格，並解決先決條件。執行個體在 OpsWorks 中必須處於運作狀態、無法有時間或負載型自動擴展器，而且必須安裝最新的 OpsWorks Agent 版本。 此外，命令會檢查所有執行個體是否具有支援 SSM Agent 所需的許可，以及是否已安裝最新的 SSM Agent 版本。如果 SSM 代理程式不存在，則命令會安裝該 SSM 代理程式，如果它未使用最新版本，則會更新 SSM 代理程式。命令也會新增任何必要的許可。
`detach`	此命令會分離指定 layer 的所有 OpsWorks 執行個體。

Command	描述
	首先，命令會執行先決條件檢查，以確保 layer 符合分離資格。如果您不想解決先決條件，您可以選擇強制分離。 接下來， 命令會指出透過 OpsWorks 標記 APIs 或透過從您的圖層和堆疊傳播標籤新增至執行個體的所有標籤都會保留。您可以在分離完成後，使用相關的 EC2 APIs移除任何這些標籤。 然後，命令會檢查您是否要將 Chef 相關組態匯出至 SSM 參數。 如果您將 Classic Load Balancer 連接到 layer，則命令會詢問是否可以分離負載平衡器，以防止任何停機時間。
`cleanup`	此命令會從您的帳戶刪除 OpsWorks 中的所有實體。它將終止執行個體並刪除所有堆疊。這應該用於不再需要的資源，做為清除帳戶的最後一步。 **Note** 我們建議您在執行 `cleanup`命令之前，先執行新的設定幾天。這可確保堆疊中任何必要的組態在需要時隨時可用。

限制

就地分離工具的主要目的是安全地分離 OpsWorks Stacks 執行個體。本節摘要說明工具的限制。

- Windows SSM 代理程式 – 如果 SSM 代理程式未安裝在執行個體上，您將需要手動安裝它。如果客服人員未更新至最新版本，則同樣適用。
- Time/Load Auto Scaling 執行個體 – 分離工具不支援已啟用 Auto Scaling 的執行個體。您必須停用您要分離之執行個體的 Auto Scaling。

- 許可 – 分離工具不會建立或產生 OpsWorks 主控台許可頁面上指定的 IAM 實體。
- 應用程式 – 分離工具不會在 OpsWorks 外部建立或產生應用程式。

開始使用

步驟 1：確認符合先決條件

分離就地工具的所有 3 個命令都是 Python 指令碼，您可以在本機、EC2 執行個體或使用 執行AWS CloudShell。

AWS CloudShell 是一種瀏覽器型 shell，可讓您以命令列方式存取已預先安裝熱門工具 （ 例如 AWS CLI 和 Python) 的 AWS selected AWS 區域. AWS CloudShell comes 中的資源。使用 時 AWS CloudShell，您可以使用您用來登入 主控台的相同登入資料。

本演練假設您使用 AWS CloudShell。

步驟 2：下載指令碼

1. 執行下列命令，下載包含遷移指令碼和所有相關檔案的 zip 檔案：

   ```
   aws s3api get-object \
   --bucket detach-in-place-bucket-prod-us-east-1 \
   --key detach_in_place_script.zip detach_in_place_script.zip
   ```

2. 執行下列命令來解壓縮 檔案。

   ```
   unzip detach_in_place_script.zip
   ```

   檔案解壓縮後，即可使用下列檔案：

   - README.md
   - 授權
   - NOTICE
   - requirements.txt
   - TODO.py

3. 如有必要，請執行下列命令`pipenv`來安裝 。

   ```
   pip install pipenv
   ```

步驟 3：執行指令碼

首先，設定您的環境，以便您可以執行下列命令來執行指令碼。

```
pipenv install -r requirements.txt
pipenv shell
```

然後，檢閱指令碼參數。

Command	參數	Description (描述)	Type	必要	預設
handle-prerequisites	--layer-id	您要分離的圖層 ID。	字串	是	-
	--region	OpsWorks 堆疊的區域。如果您的 OpsWorks 堆疊區域和 API 端點區域不同，請使用堆疊區域。這與 OpsWorks 堆疊的其他資源部分（例如 EC2 執行個體和子網路）相同。	字串	否	us-east-1
detach	--layer-id	您要分離的圖層 ID。	字串	是	-
	--batch-size	要從 layer 分離的執行個體數目（例如 5)。	字串	否	-
	--region	OpsWorks 堆疊的區域。如果您的 OpsWorks 堆疊區域和 API 端點區域不同，請使用堆疊區域。這與 OpsWorks 堆疊的其他資源部分（例如 EC2 執行個體和子網路）相同。	字串	否	us-east-1
cleanup	--stack-id	您要刪除之堆疊的 ID。	字串	否	互斥時，您必須指定圖層

Command	參數	Description (描述)	Type	必要	預設
	`--layer-id`	您要刪除的圖層 ID	字串	否	ID 或堆疊 ID
	`--region`	OpsWorks 堆疊的區域。如果您的 OpsWorks 堆疊區域和 API 端點區域不同，請使用堆疊區域。這與 OpsWorks 堆疊的其他資源部分 （ 例如 EC2 執行個體和子網路 ） 相同。	字串	否	us-east-1

您可以使用 選項執行命令`detach`，來查看 `handle-prerequisites`和 `cleanup` 命令的可用`--help`選項，如下所示：

```
python3 layer_detacher.py detach --help
python3 layer_detacher.py handle-prerequisites --help
python3 layer_detacher.py cleanup --help
```

您現在可以開始了。下列範例示範如何針對不同的使用案例執行命令。

範例：

- 範例 1：檢查 layer 是否符合所有先決條件，且符合分離資格
- 範例 2：分離 layer 的所有執行個體
- 範例 3：分批分離 layer 的所有執行個體
- 範例 4：清除 layer 的所有資源，並刪除 layer
- 範例 5：清除堆疊的所有資源，並刪除堆疊

範例 1：檢查 layer 是否符合所有先決條件，且符合分離資格

下列命令會讀取 OpsWorks 層 （ 及其包含的執行個體 ） 的相關資訊，並檢查是否符合下列先決條件：

- 所有執行個體都在線上。
- 沒有 Load/Time Auto Scaling 執行個體。
- 所有執行個體都有最新的 OpsWorks 代理程式。

- 所有執行個體都已安裝並設定最新的 SSM Agent。
- 所有執行個體都有 SSH 金鑰對。
- 每個執行個體都只屬於一個 layer。

```
python3 layer_detacher.py handle-prerequisites \
--layer-id opsworks-layer-id \
--region opsworks-stack-region
```

範例 2：分離 layer 的所有執行個體

下列命令將反覆查看 layer 的所有執行個體，檢查執行個體是否符合先決條件，並嘗試平行分離符合先決條件的所有執行個體。如果不符合一或多個先決條件，命令將為其餘不合規執行個體提供強制分離選項。

分離任何執行個體之前，命令將：

1. 儲存自訂 JSON 並將其上傳至 S3。
2. 為 layer 的每個 OpsWorks 生命週期事件建立 SSM Automation 文件，並將 Automation 文件的執行日誌上傳至 S3。
3. 為所有將分離的執行個體建立 AppRegistry 應用程式。應用程式具有與其相關聯的資源群組，可存放所有分離的執行個體和資源。資源包括 SSM Automation 文件和 SSM 參數，這些文件和參數會保存生命週期事件和自訂 Chef 配方的相關資訊。
4. 如果存在 Classic Load Balancer，則從 layer 分離 Classic Load Balancer。

此命令只會修改 OpsWorks 資源。EC2 執行個體的狀態將保持不變。

```
python3 layer_detacher.py detach \
--layer-id opsworks-layer-id \
--region opsworks-stack-region
```

範例 3：分批分離 layer 的所有執行個體

下列命令的作用與上一個範例相同。唯一的差別是它會分批分離執行個體。

此命令只會修改 OpsWorks 資源。EC2 執行個體的狀態將保持不變。

```
python3 layer_detacher.py detach \
--layer-id opsworks-layer-id \
```

```
--region opsworks-stack-region \
--batch-size 5
```

範例 4：清除 layer 的所有資源，並刪除 layer

下列命令將反覆查看圖層的所有資源，並將其刪除。更詳細地說，它會停止和刪除 OpsWorks 和 EC2 中的所有執行個體，分離負載平衡器並取消註冊 Amazon RDS 執行個體、彈性 IPs和磁碟區。清除資源後，它會刪除 layer。

此命令會刪除 OpsWorks 資源和 EC2 執行個體。如果您希望 EC2 執行個體保持未修補狀態，請在使用 `detach`命令之前使用 `cleanup`命令。如此一來，`cleanup`命令就會刪除所有剩餘的資源。

```
python3 layer_detacher.py cleanup \
--layer-id opsworks-layer-id \
--region opsworks-stack-region
```

範例 5：清除堆疊的所有資源，並刪除堆疊

下列命令將反覆運算所有圖層，然後反覆運算每個圖層的資源。對於每個 layer，命令會停止和刪除 OpsWorks 和 EC2 中的所有執行個體、分離負載平衡器，以及取消註冊 Amazon RDS 執行個體、彈性 IPs和磁碟區。然後，命令會刪除 layer。相同的程序將在屬於此堆疊的每個圖層中執行。最後，刪除所有圖層後，堆疊將被移除。

此命令會刪除 OpsWorks 資源和 EC2 執行個體。如果您希望 EC2 執行個體保持未修補狀態，請在使用 `detach`命令之前使用 `cleanup`命令。如此一來，`cleanup`命令就會刪除所有剩餘的資源。

```
python3 layer_detacher.py cleanup \
--stack-id opsworks-stack-id \
--region opsworks-stack-region
```

步驟 4：從 OpsWorks 分離後繼續操作您的資源

執行 `detach`命令後，工具會建立新的 AWS Service Catalog AppRegistry 應用程式，對應至分離的 layer。應用程式名稱遵循格式 *`layer-name`*`---`*`layer-id`*。它也會新增`OpsWorksLayerId`標籤，以唯一識別符合分離層的應用程式。

若要將新 AWS 資源新增至此應用程式（例如，新的 EC2 執行個體），您可以執行下列其中一項操作：

1. 使用 AppRegistry 應用程式的唯一應用程式標籤來標記資源：

標籤索引鍵： awsApplication

值：arn:aws:resource-groups:*region*:*account-id*:group/*application-name*/*application-id*>

2. 執行 associate-resource 命令。

此外，會為每個 AppRegistry 應用程式建立資源群組。資源群組包含下列標籤。

標籤鍵	Value
EnableAWSServiceCatalogAppRegistry	TRUE
aws:servicecatalog:applicationName	*application-name*
aws:servicecatalog:applicationId	*application-id*
aws:servicecatalog:applicationArn	arn:aws:servicecatalog: *region*:*account-id* :/applications/ *application-id*

分離後執行任務

下表提供分離後如何執行任務的相關資訊：

任務	描述
執行生命週期事件	執行 detach命令後，如果您選取 選項，指令碼會建立 5 個符合 5 個 OpsWorks 生命週期事件的自動化文件。 每個自動化文件的名稱都遵循此格式：*layer-id*_*lifecycle-event* _automation_document 。

任務	描述
	若要模擬 Systems Manager 中的 OpsWorks 行為，您需要在佈建、終止 EC2 執行個體或部署/移除配方時手動觸發自動化執行。
更新自訂 JSON	堆疊和 layer 的自訂 JSON 存放在分離期間指定的 S3 儲存貯體中，或者存放在建立的新 S3 儲存貯體中。 存放給 JSON 檔案的檔案名稱如下： • `layercustomjson.json` • `stackcustomjson.json`
變更生命週期事件的執行清單	每個生命週期事件的執行清單定義於對應的自動化文件中。若要變更執行清單，請在 AppRegistry 應用程式中尋找自動化文件並修改 `RunList` 參數。 更新配方和技術指南的程序保持不變`AWS-ApplyChefRecipes` ，因為 自動化文件觸發的 支援與 OpsWorks 相同的來源。
管理自動修復/自動擴展	當您分離執行個體時，OpsWorks 代理程式會解除安裝 。如果沒有代理程式，OpsWorks 就無法自動修復或取代運作狀態不佳的執行個體，也無法自動擴展機群。若要繼續自動擴展和取代失敗的執行個體，請建立 Amazon EC2 Auto Scaling 群組。當 Amazon EC2 偵測到運作狀態不佳且需要取代的執行個體時，群組將啟動新的執行個體，以維持其所需的容量。

任務	描述
管理Load Balancer	如果您的 layer 使用 Classic Load Balancer，`detach`命令會在取消註冊執行個體之前將其分離。這是為了確保所有 ELB 執行個體關聯在整個分離過程中都保留在 Amazon EC2 上，因此導致零停機時間。程序完成後，您將能夠在 EC2 上管理 ELB。
連線到您的執行個體	當您執行 `handle-prerequisites` 或 `detach`命令時，會執行兩個檢查： • SSM Agent 的版本和許可 • SSH 金鑰 命令也提供更新 SSM Agent 和新增必要許可的選項，讓您可以使用 Session Manager 連線到執行個體。如果存在 SSH 金鑰，您也可以選擇將 SSH 放入執行個體。

使用 Systems Manager Application Manager 執行個體索引標籤

分離後，您可以在 Application Manager Instances 索引標籤上檢視和管理執行個體。

執行個體索引標籤提供應用程式 EC2 執行個體的彙總資訊，例如其狀態、運作狀態和最後一個命令狀態。使用此標籤，您可以檢視個別執行個體的詳細資訊，例如命令歷史記錄、警示狀態、Systems Manager 代理程式運作狀態等。執行個體索引標籤也提供各種動作，例如套用 Chef 配方、啟動或停止執行個體，或從 Auto Scaling 群組新增或移除執行個體。

Stacks AWS OpsWorks 入門

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 提供一組豐富的可自訂元件，您可以混合和比對這些元件，以建立符合您特定用途的堆疊。新使用者的挑戰是了解如何將這些元件組合成運作堆疊並有效管理它。以下是開始使用的方法。

如果您想要...	完成本演練：
盡快建立範例堆疊	入門：範例
實驗 Linux 式堆疊	入門：Linux
實驗 Windows 式堆疊	入門：Windows
了解如何建立自己的 Chef 技術指南	入門：技術指南

如果您有現有的運算資源 - Amazon EC2 執行個體，甚至是在您自己的硬體上執行的現場部署執行個體 - 您可以將它們與使用 Stacks 建立的執行個體整合到堆疊中。 AWS OpsWorks 然後，您可以使用 AWS OpsWorks Stacks 以群組形式管理所有相關執行個體，無論它們的建立方式為何。

區域支援

您可以全域存取 AWS OpsWorks Stacks；也可以全域建立和管理執行個體。使用者可以設定要在 AWS GovCloud AWS OpsWorks （美國西部） 和中國 （北京） AWS 區域以外的任何區域中啟動的 Stacks 執行個體。若要使用 AWS OpsWorks Stacks，執行個體必須能夠連線至下列其中一個 AWS OpsWorks Stacks 執行個體服務 API 端點。

只有在已建立的區域中才能管理資源。在一個區域端點中建立的資源為不可用，也無法複製到另一個區域端點。您可以下列任意區域內啟動執行個體。

- 美國東部 (俄亥俄) 區域
- 美國東部 (維吉尼亞北部) 區域
- 美國西部 (奧勒岡) 區域
- 美國西部 (加利佛尼亞北部) 區域
- 加拿大 （中部） 區域 （僅限 API，不適用於在 中建立的堆疊 AWS Management Console。）
- 亞太 (孟買) 區域
- 亞太區域 (新加坡) 區域
- 亞太 (雪梨) 區域
- 亞太 (東京) 區域

- 亞太區域 (首爾) 區域
- 歐洲 (法蘭克福) 區域
- 歐洲 (愛爾蘭) 區域
- 歐洲 (倫敦) 區域
- 歐洲 (巴黎) 區域
- 南美洲 (聖保羅) 區域

範例堆疊入門

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本演練說明如何使用 AWS OpsWorks Stacks 快速建立範例 Node.js 應用程式環境，只要按幾下滑鼠，無需編寫程式碼。完成後，您會有一個執行 Chef 12 的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體、Node.js HTTP 伺服器，以及可用來與 Twitter 互動並在網頁上留下評論的 Web 應用程式。

> **Note**
>
> 由於完成此演練會自動建立類型為 c3.large 的執行個體，因此您無法在 AWS 免費方案中使用此演練或 AWS OpsWorks Stacks 中的範例堆疊建立工具。雖然在 VPC 中使用範例堆疊建立工具會建立 t2.medium 執行個體，但 VPCs 目前無法在 AWS 免費方案中使用。

步驟 1：完成事前準備

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您必須先完成下列步驟，才能開始演練。這些設定步驟包括註冊 AWS 帳戶、建立管理使用者，以及將存取許可指派給 AWS OpsWorks Stacks。

主題

- 註冊 AWS 帳戶
- 建立具有管理存取權的使用者
- 指派服務存取許可

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。

指派服務存取許可

將 AWS OpsWorks `AWSOpsWorks_FullAccess`和 `AmazonS3FullAccess`許可新增至您的角色或使用者，以啟用 Stacks 服務的存取權 （以及 AWS OpsWorks Stacks 依賴的相關服務）。

如需新增許可的詳細資訊，請參閱新增 IAM 身分許可 （主控台）。

您現已完成所有設定步驟，可開始本演練。

步驟 2：建立堆疊

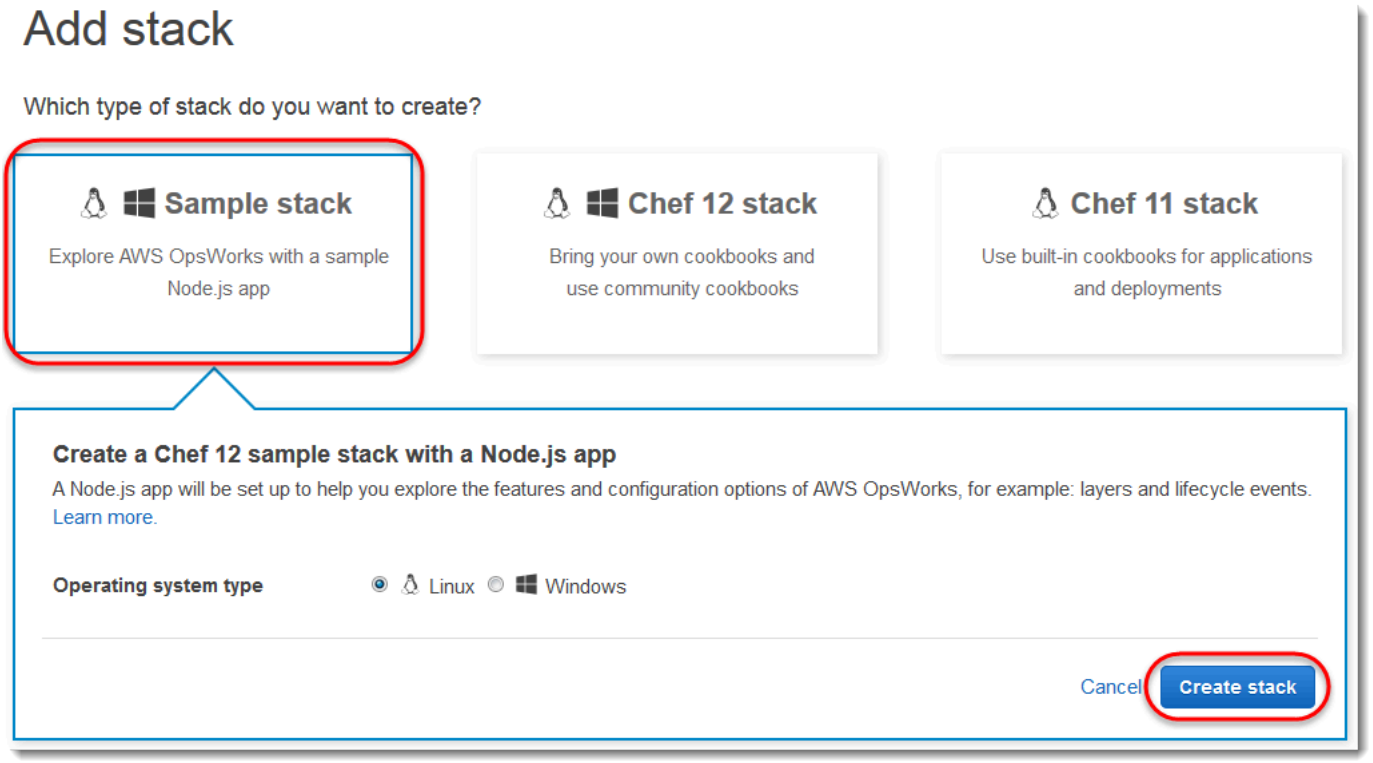
> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

在此步驟中，您會使用 AWS OpsWorks Stacks 主控台來建立堆疊。堆疊是執行個體 （ 例如 Amazon EC2 執行個體 ） 和相關 AWS 資源的集合，這些資源具有共同目的且您想要一起管理。(如需詳細資訊，請參閱 堆疊。) 本演練只有一個執行個體。

在您開始此步驟之前，請先完成事前準備。

建立堆疊

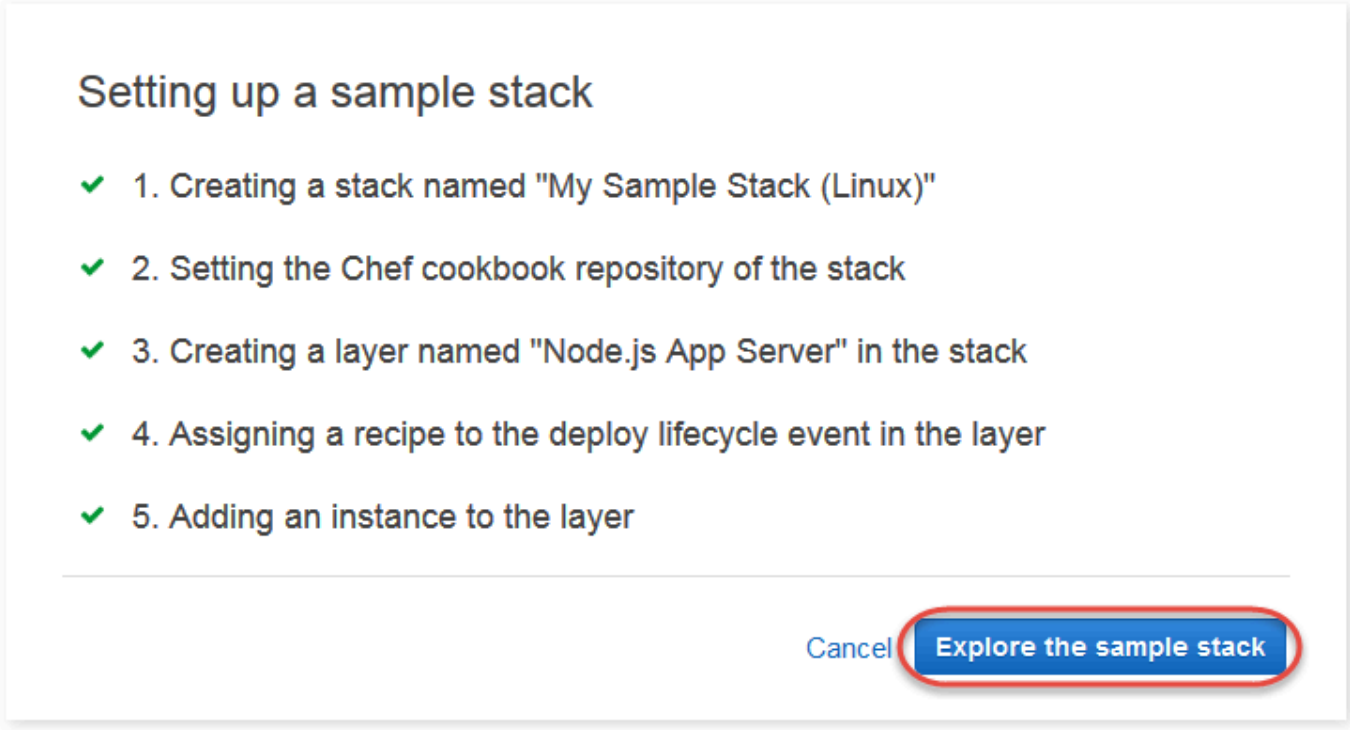
1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/opsworks/ 開啟 AWS OpsWorks 主控台。
2. 執行下列任一作業 (若適用的話)：
   - 如果顯示歡迎使用 AWS OpsWorks Stacks 頁面，請選擇新增您的第一個堆疊或新增您的第一個 Stacks AWS OpsWorks 堆疊 （ 兩個選項都執行相同的動作 ）。即會顯示 Add stack (新增堆疊) 頁面。
   - 若顯示 OpsWorks Dashboard (OpsWorks 儀表板) 頁面，請選擇 Add stack (新增堆疊)。即會顯示 Add stack (新增堆疊) 頁面。
3. 在顯示的 Add stack (新增堆疊) 頁面中，如果尚未選擇，請選擇 Sample stack (範例堆疊)。
4. 為已選擇 Linux 的 Operating system type (作業系統類型)，選擇 Create stack (建立堆疊)：



5. AWS OpsWorks Stacks 會建立名為 My Sample Stack (Linux) 的堆疊。 AWS OpsWorks Stacks 也會新增所有必要的元件，以將應用程式部署至堆疊：

   - layer 是一組執行個體的藍圖。它會指定諸如執行個體的設定、資源、安裝的套件和安全群組等項目。(如需詳細資訊，請參閱 層。) 此 layer 名為 Node.js App Server (Node.js 應用程式伺服器)。
   - 在這種情況下，「執行個體」是 Amazon Linux 2 EC2 執行個體。(如需執行個體的詳細資訊，請參閱執行個體。) 該執行個體的主機名稱是 nodejs-server1。
   - 「應用程式」是在執行個體上執行的程式碼。(如需應用程式的詳細資訊，請參閱應用程式。) 此應用程式名為 Node.js Sample App (Node.js 範例應用程式)。

6. Stacks AWS OpsWorks 建立堆疊後，請選擇探索範例堆疊以顯示我的範例堆疊 (Linux) 頁面 （ 如果您完成此演練多次，則我的範例堆疊 (Linux) 之後可能會有序號，例如 2 或 3)：



在下一個步驟中，您會啟動執行個體並將應用程式部署到執行個體。

步驟 3：啟動執行個體並部署應用程式

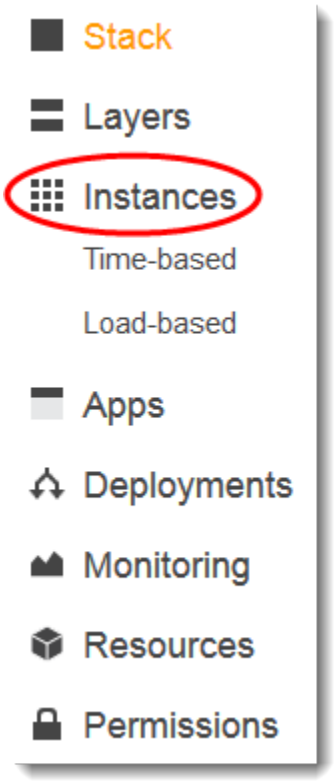
> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

現在您有執行個體和應用程式，可啟動執行個體並將應用程式部署到執行個體。

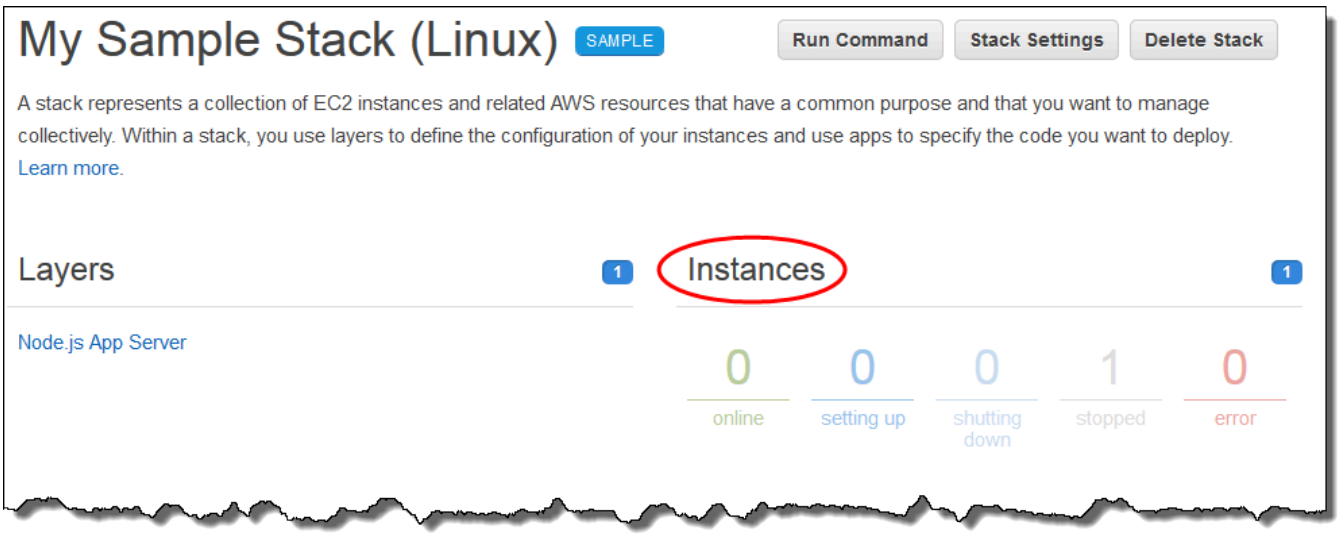
啟動執行個體並部署應用程式

1. 執行以下任意一項：

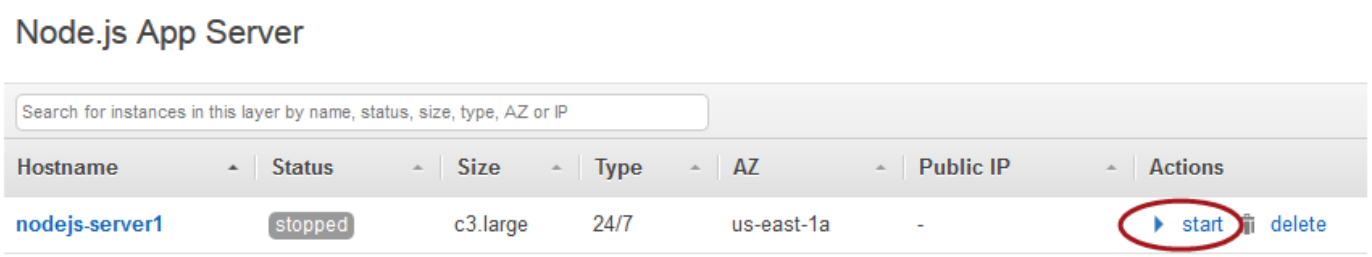
   - 在服務導覽窗格中，選擇 Instances (執行個體)：



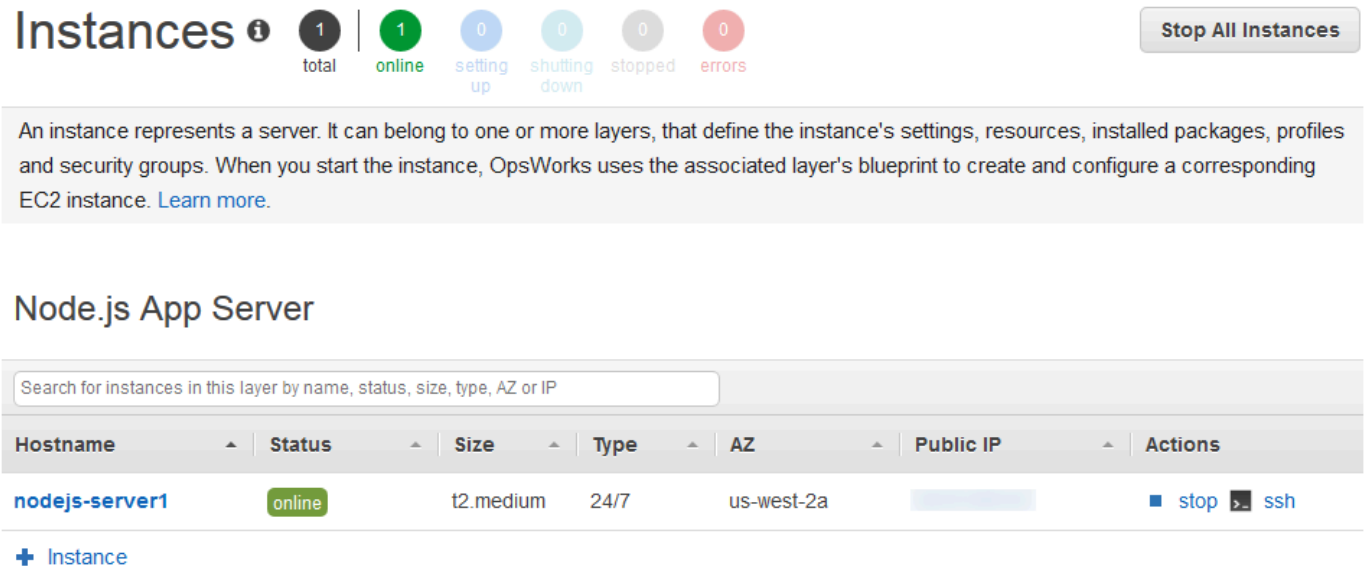
- 在 My Sample Stack (Linux) 頁面上，選擇 Instances (執行個體)：



2. 在 Instances (執行個體) 頁面上，針對 Node.js App Server (Node.js 應用程式伺服器) 的 nodejs-server1，選擇 start (開始)：



3. 請在 online (線上) 圓圈變成亮綠色之後，再繼續。(如果看到故障訊息，請參考偵錯和故障診斷指南。)
4. 當執行個體設定時， AWS OpsWorks Stacks 會將應用程式部署到執行個體。
5. 繼續之前，您的結果應該類似下列螢幕擷取畫面 (如果您收到故障訊息，建議您參考偵錯和故障診斷指南)：



您的執行個體現有已部署到執行個體的應用程式。

在下一個步驟中，您要測試執行個體上的應用程式。

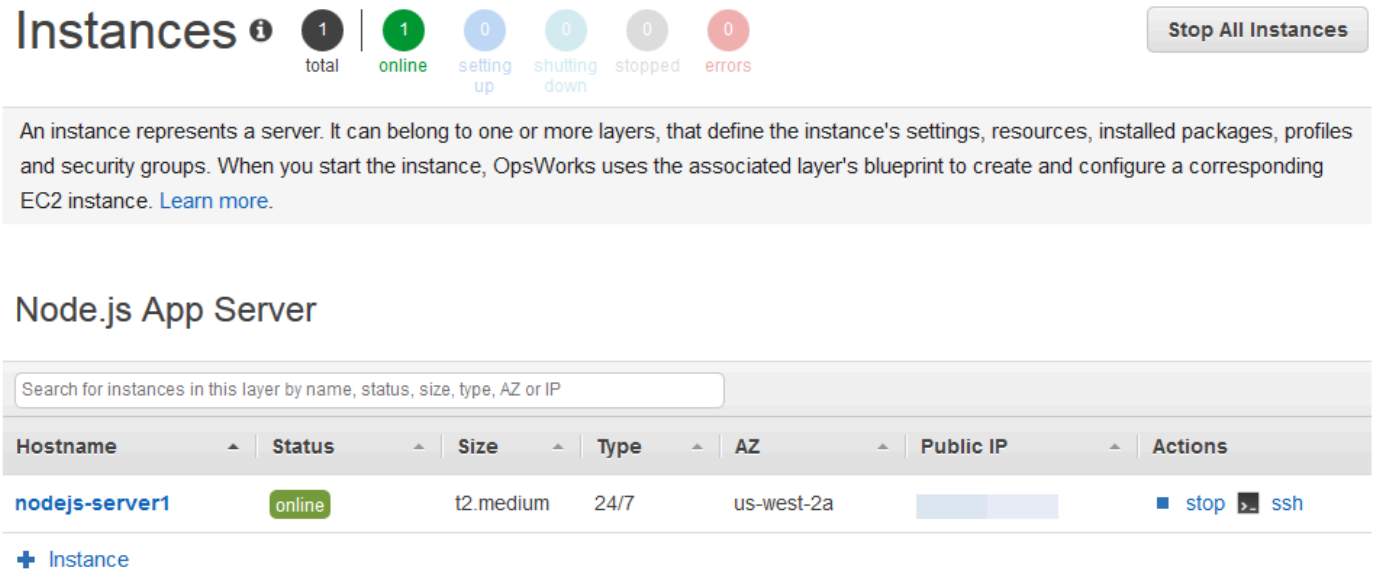
步驟 4：測試執行個體上已部署的應用程式

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

測試執行個體上的應用程式部署結果。

測試執行個體上的部署

1. 在上一個步驟顯示的 Instances (執行個體) 頁面中，針對 Node.js App Server (Node.js 應用程式伺服器) 之 nodejs-server1 的 Public IP (公有 IP)，選擇 IP 地址。



2. 在賀辭網頁的 Leave a comment (留下評論) 文字方塊中輸入評論，然後選擇 Send (傳送) 測試應用程式。應用程式會將您的評論新增到網頁。繼續留下評論，並依您需要隨時選擇 Send (傳送)。

3. 如果您有 Twitter 帳戶，請選擇 Tweet (推文) 或 Follow @AWSOpsWorks (關注 @AWSOpsWorks)，然後遵循畫面指示發佈有關此應用程式的推文或關注 @AWSOpsWorks。

您現已成功測試執行個體上所部署的應用程式。

在其餘步驟中，您可以使用 AWS OpsWorks Stacks 主控台來探索堆疊及其元件的設定。在下一個步驟中，您可以從檢查堆疊設定開始您的探索。

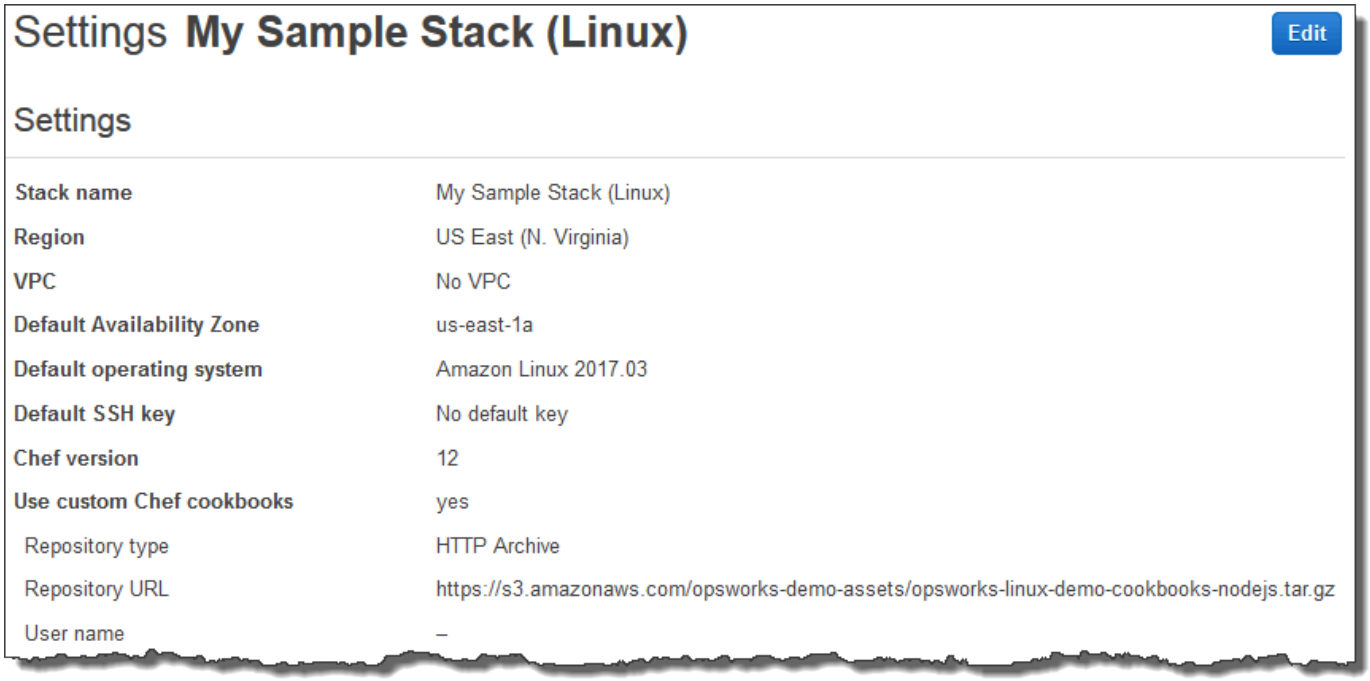
步驟 5：探索堆疊的設定

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

檢查 AWS OpsWorks Stacks 如何設定堆疊。

顯示堆疊的設定

1. 在服務導覽列中，選擇 Stack (堆疊)。即會顯示 My Sample Stack (Linux) 頁面。
2. 選擇 Stack Settings (堆疊設定)。即會顯示 Settings My Sample Stack (Linux) (設定 My Sample Stack (Linux)) 頁面：



若要進一步了解許多設定，請選擇 Edit (編輯)，然後將滑鼠的游標移到每項設定上。(並非所有設定在畫面上皆有描述。) 如需這些設定的詳細資訊，請參閱 建立新的堆疊。

若要探索本演練中使用的 Chef 技術指南，請開啟 GitHub 的 opsworks-linux-demo-cookbooks-nodejs 儲存庫。

在下一個步驟中，您可以探索 layer 的設定。

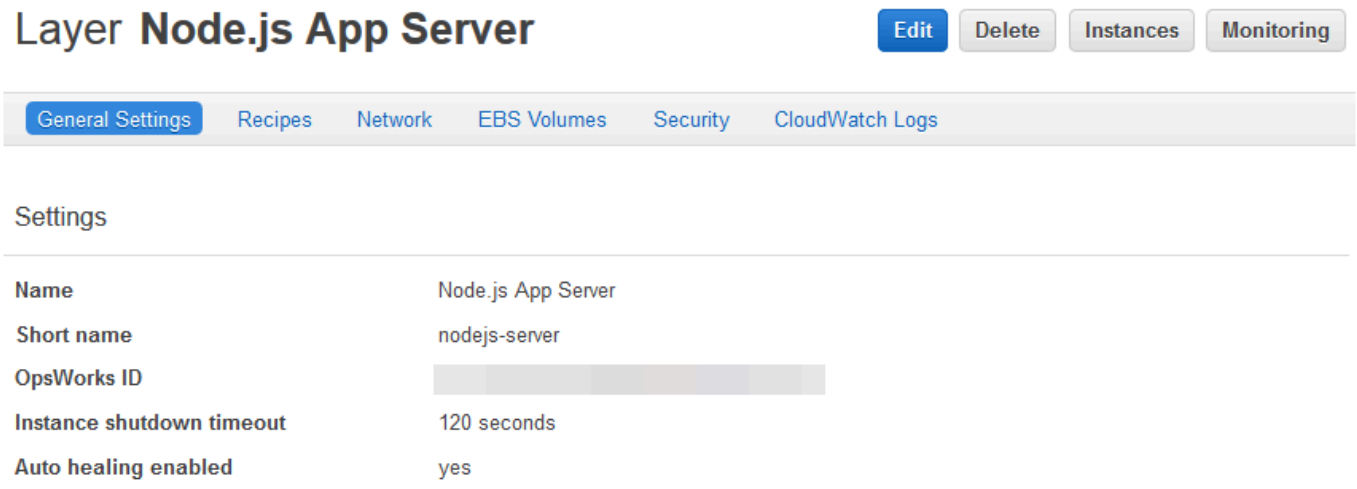
步驟 6：探索 Layer 的設定

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

檢查 AWS OpsWorks Stacks 如何設定 layer。

顯示 layer 的設定

1. 在服務導覽窗格中，選擇 Layers (Layer)。即會顯示 Layers (Layer) 頁面。
2. 選擇 Node.js App Server (Node.js 應用程式伺服器)。即會顯示 Layer Node.js App Server (Layer Node.js 應用程式伺服器) 頁面。若要查看 layer 的設定，請選擇 General Settings (一般設定)、Recipes (配方)、Network (網路)、EBS Volumes (EBS 磁碟區) 和 Security (安全性)：



若要進一步了解許多設定，請選擇 Edit (編輯)，然後將滑鼠的游標移到每項設定上。(並非所有設定在畫面上皆有描述。) 如需這些設定的詳細資訊，請參閱 編輯 OpsWorks Layer 的組態。

在下一個步驟中，您可以探索執行個體的設定和日誌。

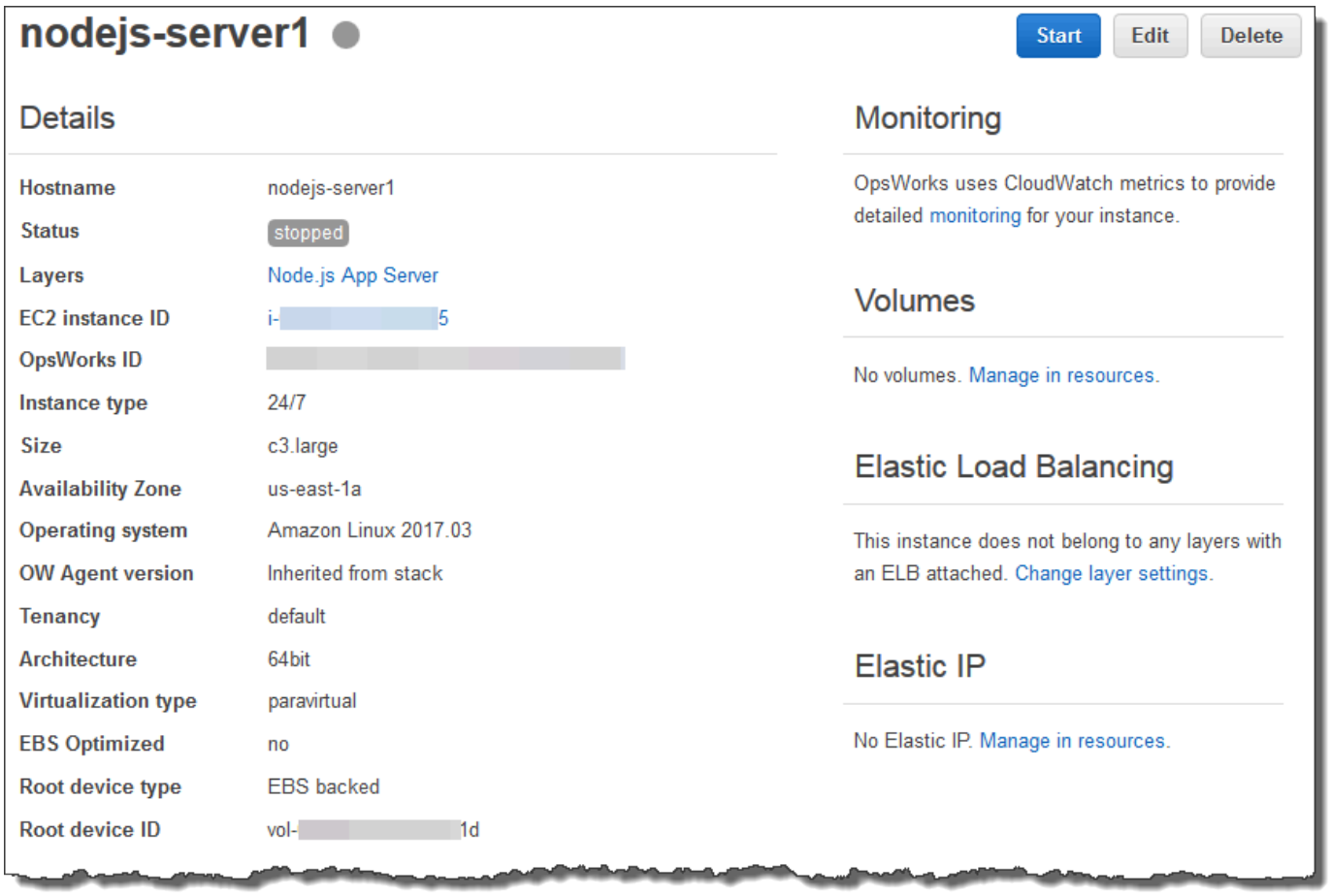
步驟 7：探索執行個體的設定和日誌

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

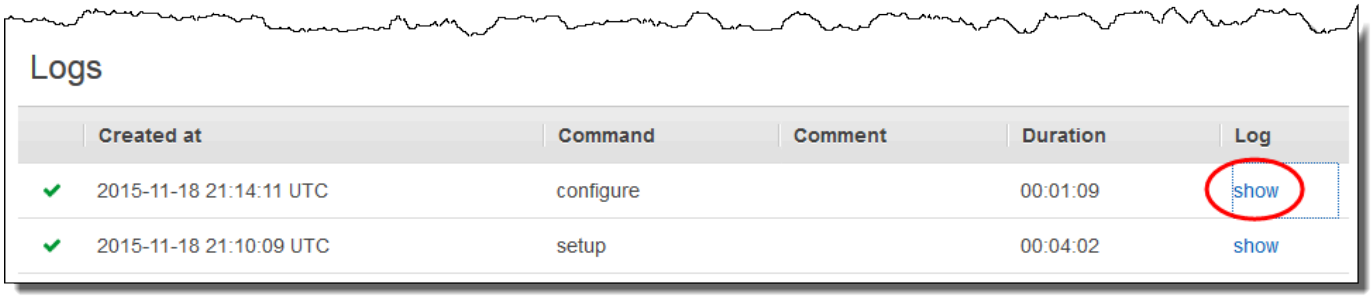
檢查 AWS OpsWorks Stacks 用來啟動執行個體的設定。您也可以檢查 Stacks AWS OpsWorks 建立的執行個體日誌。

顯示執行個體的設定和日誌

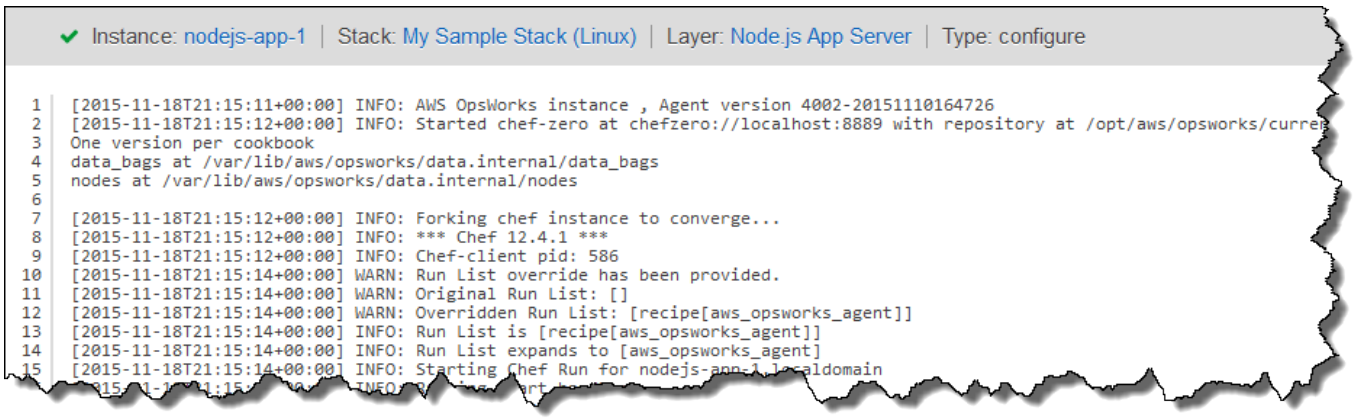
1. 在服務導覽窗格中，選擇 Instances (執行個體)。即會顯示 Instances (執行個體) 頁面。
2. 針對 Node.js App Server (Node.js 應用程式伺服器)，選擇 nodejs-server1。即會顯示執行個體的 properties (屬性) 頁面。



3. 若要探索執行個體日誌，請在 Logs (日誌) 區段中，針對 Log (日誌)，選擇 show (顯示)。



4. AWS OpsWorks Stacks 會在單獨的 Web 瀏覽器索引標籤中顯示日誌。



若要進一步了解某些執行個體設定所代表的意義，請返回 nodejs-server1 頁面，選擇 Stop (停止)，然後在您看到確認訊息時，選擇 Stop (停止)。在狀態從停止變更為停止後選擇編輯，然後將滑鼠游標移至每個設定。(並非所有設定在畫面上皆有描述。) 如需這些設定的詳細資訊，請參閱 將執行個體新增至 Layer。

完成檢閱設定後，選擇 Start (啟動) 重新啟動執行個體，然後等待 Status (狀態) 變更為 online (線上)。否則，稍後您將無法測試應用程式，因為執行個體會維持停止狀態。

> **Note**
>
> 如果您想要登入執行個體以進一步探索，您必須先向 AWS OpsWorks Stacks 提供公有 SSH 金鑰的相關資訊 （ 您可以使用 ssh-keygen 或 PuTTYgen 等工具建立 ），然後您必須在 My Sample Stack (Linux) 堆疊上設定許可，讓使用者能夠登入執行個體。如需說明，請參閱 註冊使用者的公有 SSH 金鑰 與 使用 SSH 登入。

在下一個步驟中，探索應用程式的設定。

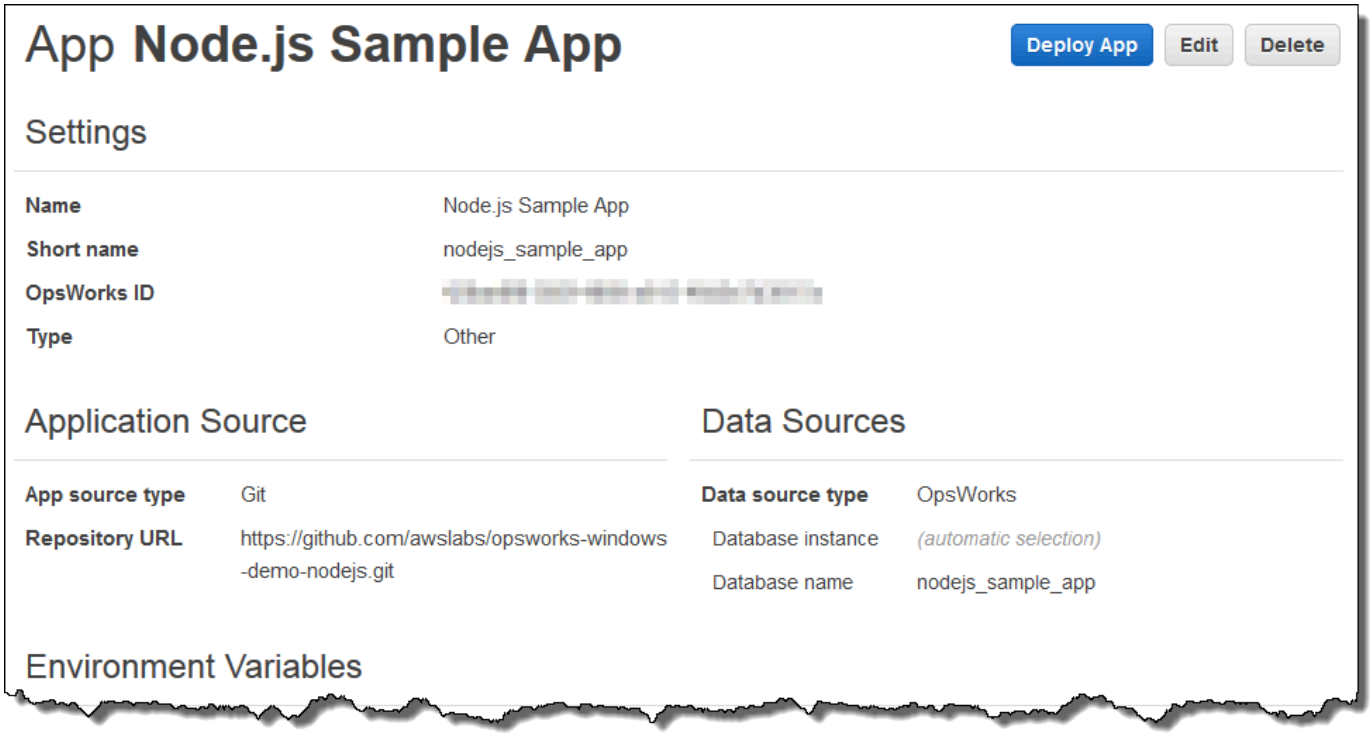
步驟 8：探索應用程式的設定

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

檢查 AWS OpsWorks Stacks 用於應用程式的設定。

顯示應用程式的設定

1. 在服務導覽窗格中，選擇 Apps (應用程式)。即會顯示 Apps (應用程式) 頁面。
2. 選擇 Node.js Sample App (Node.js 範例應用程式)。即會顯示 App Node.js Sample App (應用程式 Node.js 範例應用程式) 頁面：



若要了解某些設定代表的意義，請選擇 Edit (編輯)，然後將滑鼠的游標移到每項設定上。(並非所有設定在畫面上皆有描述。) 如需這些設定的詳細資訊，請參閱新增應用程式。

在下一個步驟中，您可以探索 layer 監控報告。

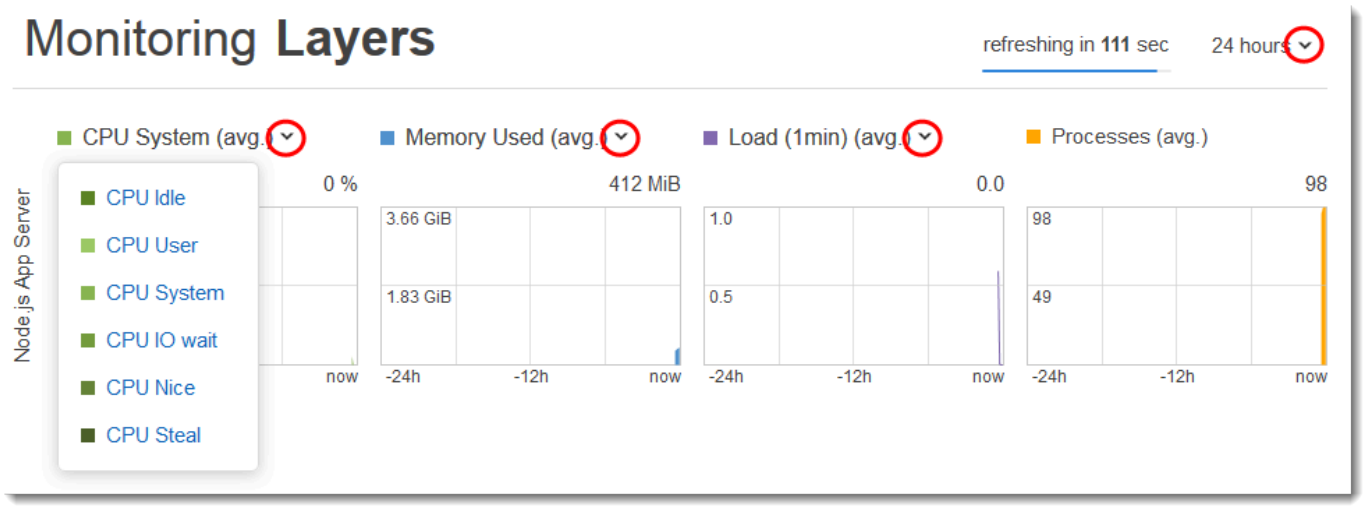
步驟 9：探索 Layer 監控報告

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

檢查 AWS OpsWorks Stacks 針對 layer 運算效能產生的報告。

顯示 layer 監控報告

1. 在服務導覽窗格中，選擇 Monitoring (監控)。即會顯示 Monitoring Layers (監控 Layer) 頁面。
2. 若要探索其他檢視，請選擇 CPU、Memory (記憶體)、Load (載入) 和時間旁的箭頭：



如需這些和其他報告的資訊，請參閱使用 Amazon CloudWatch和監控。

在下一個步驟中，您可以探索其他堆疊設定。

步驟 10：探索其他堆疊設定

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

在這個步驟中，您可以檢查其他堆疊設定。

AWS OpsWorks Stacks 不會執行個別部署，不會佈建其他資源，也不會調整任何額外的許可做為此堆疊的一部分，因此在部署和命令、資源和許可頁面上沒有太多興趣。如果您還是要查看這些設定，請分別選擇服務導覽窗格中的 Deployments (部署)、Resources (資源) 和 Permissions (許可)。如需要這些頁面代表意義的詳細資訊，請參閱部署應用程式、資源管理和管理使用者許可。

在下一個步驟中，您可以清除用於本演練 AWS 的資源。此步驟為選用。當您繼續進一步了解 Stacks AWS OpsWorks 時，建議您繼續使用這些 AWS 資源。不過，保留這些 AWS 資源可能會對 AWS 您的帳戶產生持續費用。如果您想要將這些 AWS 資源保留以供日後使用，您現在已完成本演練，您可以直接跳到 後續步驟。

步驟 11 (選用)：清理

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

為了避免 AWS 您的帳戶產生額外費用，您可以刪除應用程式和用於此演練 AWS 的資源，包括執行個體和 Stacks AWS OpsWorks 堆疊。（如需詳細資訊，請參閱 AWS OpsWorks 定價。）不過，當您繼續進一步了解 Stacks AWS OpsWorks 時，建議您繼續使用這些 AWS 資源。如果您想要讓這些 AWS 資源保持可用，您現在已完成本演練，您可以跳至 後續步驟。

存放在您為此逐步解說建立的資源裡的內容，可包含個人識別資訊。如果您不希望再將此資訊存放在 AWS，請遵循本主題的步驟。

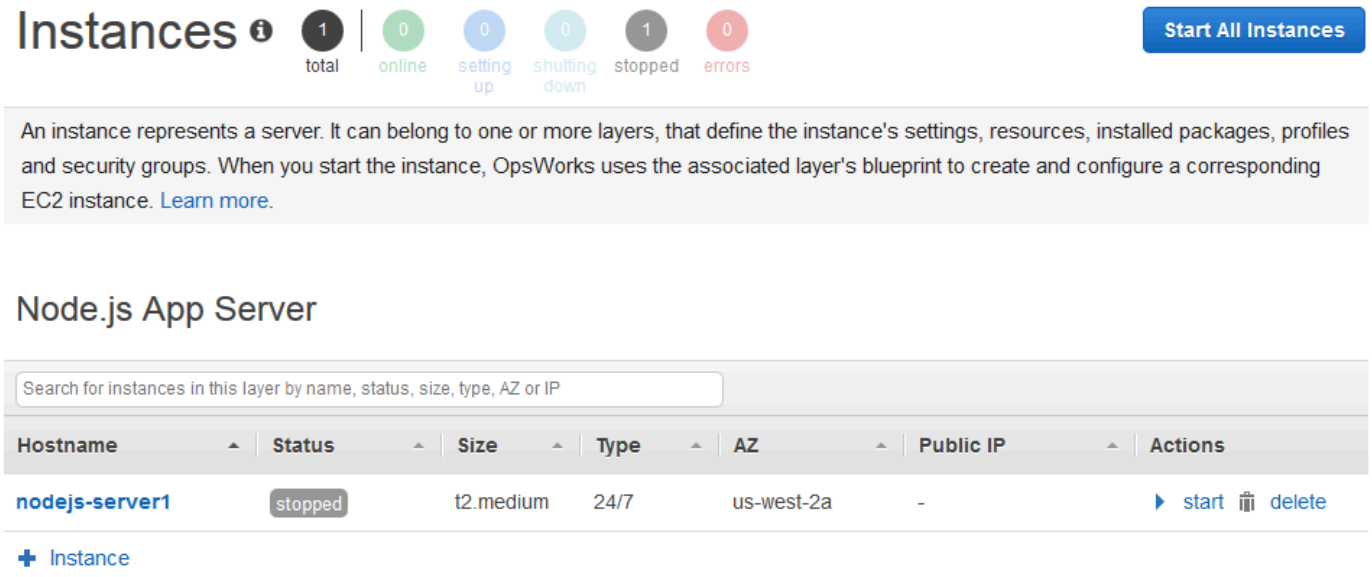
從堆疊刪除應用程式

1. 在服務導覽窗格中，選擇 Apps (應用程式)。即會顯示 Apps (應用程式) 頁面。
2. 針對 Node.js Sample App (Node.js 範例應用程式) 的 Actions (動作)，選擇 delete (刪除)。在您看見確認訊息時，選擇 Delete (刪除)。刪除應用程式時，您會看到 No apps (無應用程式) 訊息。

刪除堆疊的執行個體

1. 在服務導覽窗格中，選擇 Instances (執行個體)。即會顯示 Instances (執行個體) 頁面。
2. 針對 Node.js App Server (Node.js 應用程式伺服器) 之 nodejs-server1 的 Actions (動作)，選擇 stop (停止)。在您看見確認訊息時，選擇 Stop (停止)。

   此程序需要幾分鐘的時間。當 AWS OpsWorks Stacks 完成時，會顯示下列結果。



3. 針對 Actions (動作)，選擇 delete (刪除)。在您看見確認訊息時，選擇 Delete (刪除)。執行個體已刪除，並顯示 No instances (無執行個體) 訊息。

刪除堆疊

1. 在服務導覽窗格中，選擇 Stack (堆疊)。即會顯示 My Sample Stack (Linux) 頁面。
2. 選擇 Delete Stack (刪除堆疊)。在您看見確認訊息時，選擇 Delete (刪除)。堆疊已刪除，並顯示 OpsWorks Dashboard (OpsWorks 儀表板) 頁面。

您也可以選擇性地刪除您用於本演練的使用者和 Amazon EC2 金鑰對，如果您不想重複使用這些金鑰對來存取其他 AWS 服務和 EC2 執行個體。如需說明，請參閱刪除 IAM 使用者、Amazon EC2 金鑰對和 Linux 執行個體。

您現已完成本演練。如需詳細資訊，請參閱後續步驟。

後續步驟

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

現在您已完成本演練，您可以進一步了解如何使用 AWS OpsWorks Stacks：

- 練習使用 Stacks AWS OpsWorks 自行手動重新建立此堆疊。請參閱 入門：Linux。
- 探索技術指南和 AWS OpsWorks Stacks 用於本演練的應用程式。請參閱配套深入了解：探索本演練中使用的技術指南 演練中的深入了解：探索本演練中使用的應用程式和入門：Linux。
- 練習搭配 Windows 執行個體使用 AWS OpsWorks Stacks。請參閱 入門：Windows。
- 了解如何建立新的堆疊以進一步了解堆疊。
- 透過編輯 OpsWorks Layer 的組態進一步了解 layer。
- 透過將執行個體新增至 Layer進一步了解執行個體。
- 透過部署應用程式進一步了解應用程式。
- 進一步了解 技術指南和配方。
- 建立您自己的技術指南。請參閱 入門：技術指南。
- 了解使用安全與許可控制堆疊的存取權。

Linux 堆疊入門

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

在此演練中，您將了解如何使用 AWS OpsWorks Stacks 來建立 Node.js 應用程式環境。完成後，您將擁有執行 Chef 12 的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體、Node.js HTTP 伺服器，以及可用來與 Twitter 互動並在網頁上留下評論的 Web 應用程式。

Chef 是一個第三方框架，用來設定和維護伺服器 (如 EC2 執行個體) 以及在這些伺服器上部署和維護應用程式的方式。如果您不熟悉 Chef，在完成本演練後，建議您進一步了解 Chef，以便充分利用 Stacks AWS OpsWorks 提供的所有功能。(如需詳細資訊，請參閱了解 Chef 網站。)

AWS OpsWorks Stacks 支援四種 Linux 發行版本：Amazon Linux、Ubuntu Server、CentOS 和 Red Hat Enterprise Linux。在此演練中，我們使用 Ubuntu Server。 AWS OpsWorks Stacks 也適用於 Windows Server。雖然我們有同等的 Windows Server 堆疊逐步解說，但建議您先完成此逐步解說，以了解 Stacks AWS OpsWorks 和 Chef 的基本概念，這些概念不會在此重複。完成本演練後，請參閱 入門：Windows 演練。

主題

- 步驟 1：完成事前準備
- 步驟 2：建立堆疊
- 步驟 3：將 Layer 新增至堆疊
- 步驟 4：指定要部署到執行個體的應用程式
- 步驟 5：啟動執行個體
- 步驟 6：將應用程式部署到執行個體
- 步驟 7：測試執行個體上已部署的應用程式
- 步驟 8 (選用)：清理
- 後續步驟
- 深入了解：探索本演練中使用的技術指南
- 深入了解：探索本演練中使用的應用程式

步驟 1：完成事前準備

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

完成下列步驟，才能開始演練。這些設定步驟包括註冊 AWS 帳戶、建立管理使用者，以及將存取許可指派給 AWS OpsWorks Stacks。

若您已完成入門：範例演練，您便已完成本演練的事前準備，可直接跳到步驟 2：建立堆疊。

主題

- 註冊 AWS 帳戶
- 建立具有管理存取權的使用者
- 指派服務存取許可

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權 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。

指派服務存取許可

將 AWS OpsWorks `AWSOpsWorks_FullAccess`和 `AmazonS3FullAccess`許可新增至您的角色或使用者，以啟用對 Stacks 服務的存取 （以及 AWS OpsWorks Stacks 依賴的相關服務）。

如需新增許可的詳細資訊，請參閱新增 IAM 身分許可 （主控台）。

您現已完成所有設定步驟，可開始本演練。

步驟 2：建立堆疊

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您將使用 AWS OpsWorks Stacks 主控台來建立堆疊。堆疊是執行個體和相關 AWS 資源的集合，具有共同目的且您想要一起管理。(如需詳細資訊，請參閱 堆疊。) 本演練只有一個執行個體。

開始之前，請先完成事前準備 (若尚未完成)。

建立堆疊

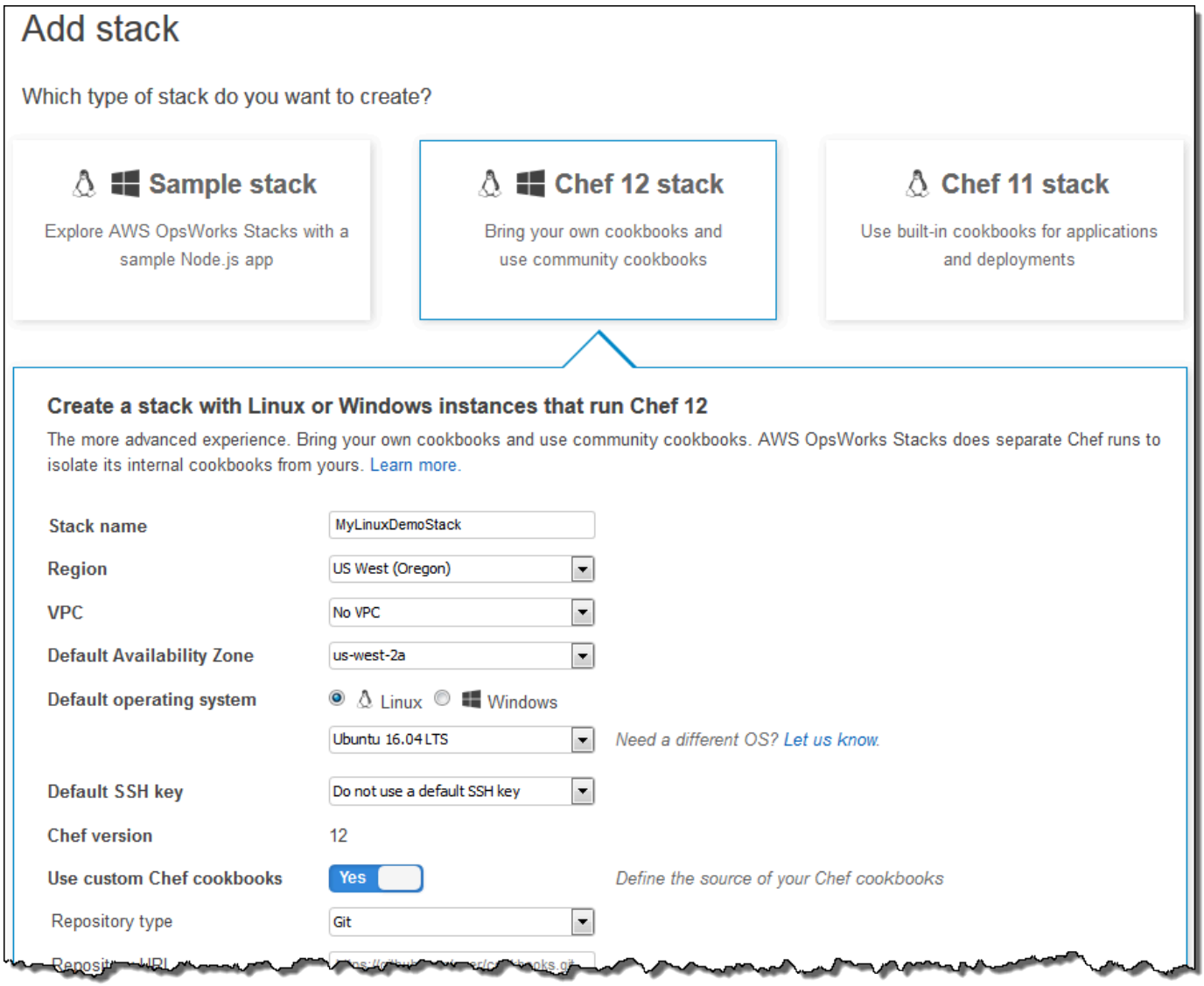
1. 登入 AWS Management Console 並開啟位於 https：//https://console.aws.amazon.com/opsworks/ 的 AWS OpsWorks 主控台。
2. 執行下列任一作業 (若適用的話)：
   - 如果顯示歡迎使用 AWS OpsWorks Stacks 頁面，請選擇新增您的第一個堆疊或新增您的第一個 Stacks AWS OpsWorks 堆疊 （ 兩個選項都執行相同的動作 ）。即會顯示 Add stack (新增堆疊) 頁面。
   - 若顯示 OpsWorks Dashboard (OpsWorks 儀表板) 頁面，請選擇 Add stack (新增堆疊)。即會顯示 Add stack (新增堆疊) 頁面。
3. 在顯示的 Add stack (新增堆疊) 頁面中，請選擇 Chef 12 stack (Chef 12 堆疊) (若尚未選擇)。
4. 在 Stack name (堆疊名稱) 方塊中輸入一個名稱 (例如：**MyLinuxDemoStack**)。(您可以輸入不同的名稱，但請務必在本演練的後續部分一律將 MyLinuxDemoStack 取代為您選擇的名稱。)
5. 針對區域，選擇美國西部 （ 奧勒岡 ）。
6. 針對 VPC，執行下列其中一項作業：
   - 若 VPC 可用，請選擇它。(如需詳細資訊，請參閱 在 VPC 中執行堆疊。)
   - 否則，請選擇 No VPC (無 VPC)。
7. 針對 Default operating system (預設作業系統)，選擇 Linux 和 Ubuntu 18.04 LTS。
8. 針對 Use custom Chef cookbooks (使用自訂 Chef 技術指南)，選擇 Yes (是)。
9. 針對 Repository type (儲存庫類型)，選擇 Http Archive (Http 封存)。
10. 針對 Repository URL (儲存庫 URL)，輸入 **https://s3.amazonaws.com/opsworks-demo-assets/opsworks-linux-demo-cookbooks-nodejs.tar.gz**
11. 保留下列項目的預設值：
    - Default Availability Zone (預設可用區域) (us-west-2a)
    - Default SSH key (預設 SSH 金鑰) (Do not use a default SSH key (不使用預設 SSH 金鑰))
    - User name (使用者名稱) (空白)
    - Password (密碼) (空白)

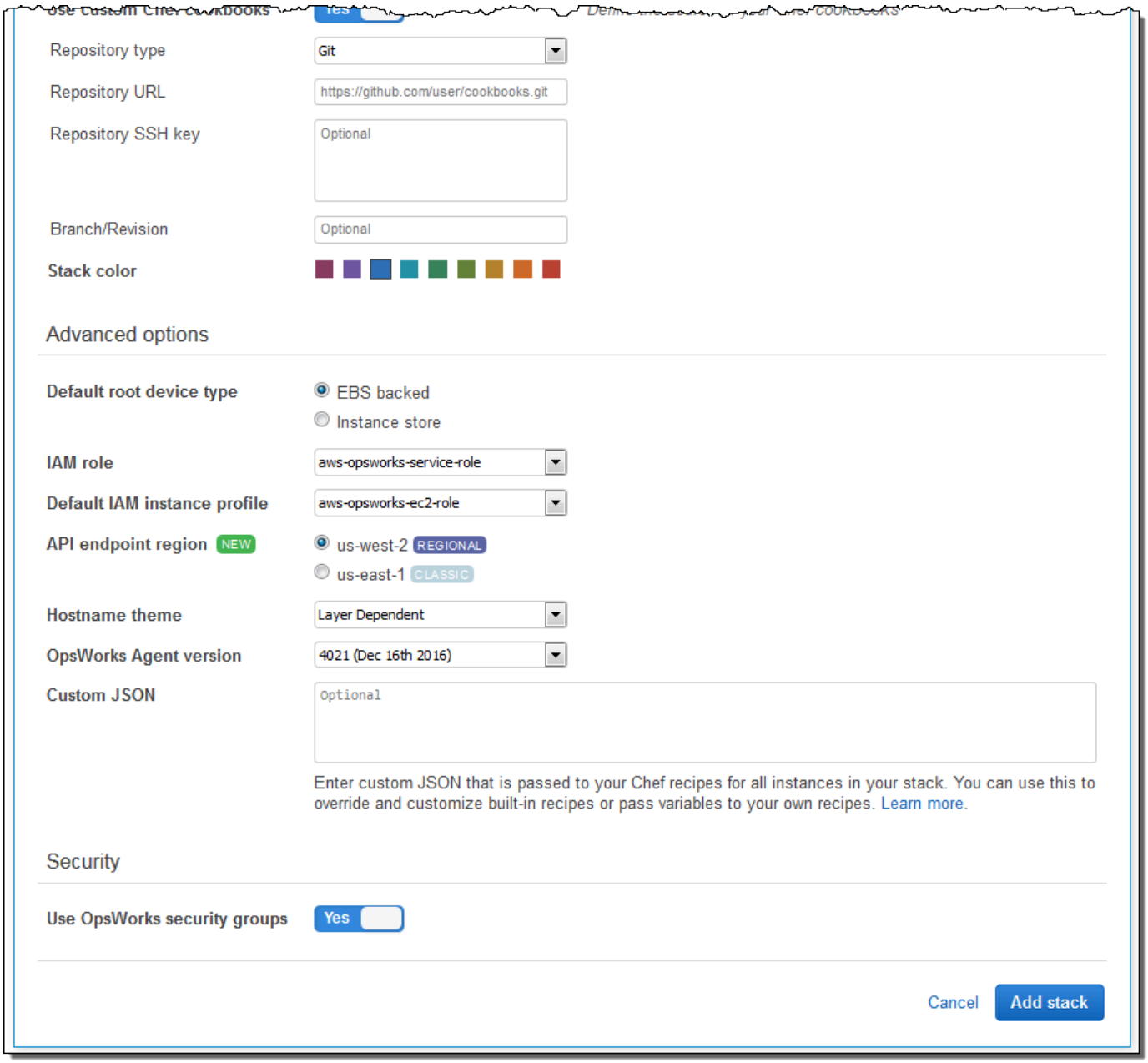
- Stack color (堆疊色彩) (深藍色)

12. 選擇 Advanced (進階)。
13. 對於 IAM 角色，請執行下列其中一項操作（如需詳細資訊，請參閱 允許 AWS OpsWorks Stacks 代表您)：
    - 若 aws-opsworks-service-role 可用，請選擇它。
    - 若 aws-opsworks-service-role 不可用，請選擇 New IAM role (新 IAM 角色)。
14. 對於預設 IAM 執行個體描述檔，請執行下列其中一項（如需詳細資訊，請參閱 指定在 EC2 執行個體上執行之應用程式的許可)：
    - 若 aws-opsworks-ec2-role 可用，請選擇它。
    - 如果無法使用 aws-opsworks-ec2-role，請選擇新的 IAM 執行個體描述檔。
15. 針對 API endpoint region (API 端點區域)，選擇您希望與堆疊建立關聯的區域 API 端點。如果您希望堆疊位於美國東部（維吉尼亞北部）區域端點的美國西部（奧勒岡）區域，請選擇 us-east-1。如果您希望堆疊同時位於美國西部（奧勒岡）區域，並與美國西部（奧勒岡）區域端點相關聯，請選擇 us-west-2。

    > **Note**
    >
    > 美國東部（維吉尼亞北部）區域端點包含較舊 AWS 區域 的回溯相容性，但最佳實務是選擇離您管理位置最近的區域端點 AWS。如需詳細資訊，請參閱區域支援。

16. 保留下列項目的預設值：
    - Default root device type (預設根設備類型) (EBS backed (EBS 後端))
    - Hostname theme (主機名稱主題) (Layer Dependent (依存於 Layer))
    - OpsWorks Agent version (OpsWorks 代理程式版本) (最新版本)
    - Custom JSON (自訂 JSON) (空白)
    - Use OpsWorks security groups (使用 OpsWorks 安全群組) (Yes (是))
17. 您的結果應該符合下列螢幕擷取畫面，但 VPC、IAM 角色和預設 IAM 執行個體描述檔除外：


Add stack
Which type of stack do you want to create?
Sample stack
Explore AWS OpsWorks Stacks with a sample Node.js app
Chef 12 stack
Bring your own cookbooks and use community cookbooks
Chef 11 stack
Use built-in cookbooks for applications and deployments
Create a stack with Linux or Windows instances that run Chef 12
The more advanced experience. Bring your own cookbooks and use community cookbooks. AWS OpsWorks Stacks does separate Chef runs to isolate its internal cookbooks from yours. Learn more.
Stack name
MyLinuxDemoStack
Region
US West (Oregon)
VPC
No VPC
Default Availability Zone
us-west-2a
Default operating system
Linux
Windows
Ubuntu 16.04 LTS
Need a different OS? Let us know.
Default SSH key
Do not use a default SSH key
Chef version
12
Use custom Chef cookbooks
Yes
Define the source of your Chef cookbooks
Repository type
Git



18. 選擇新增 Stack. AWS OpsWorks Stacks 建立堆疊並顯示 MyLinuxDemoStack 頁面。

現在，您已擁有一個設定正確的堆疊可用於本演練。

在下一個步驟中，您將會將 layer 新增至堆疊中。

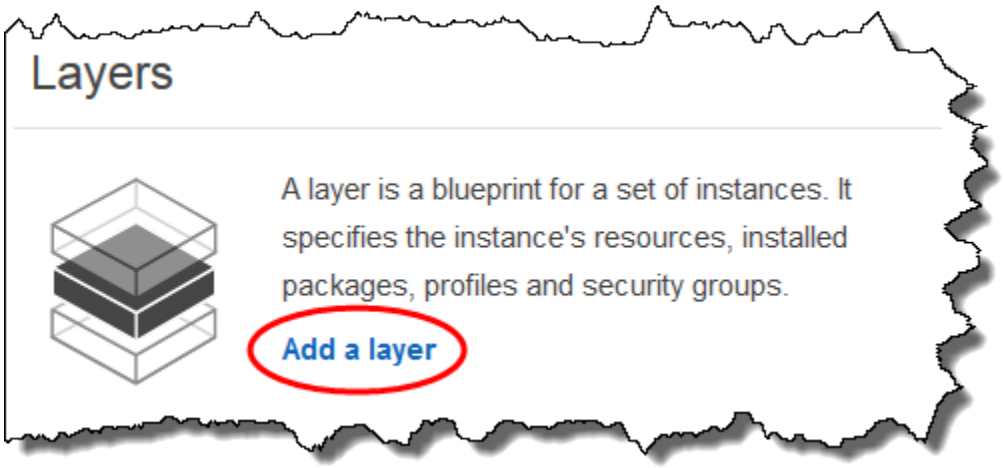
步驟 3：將 Layer 新增至堆疊

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

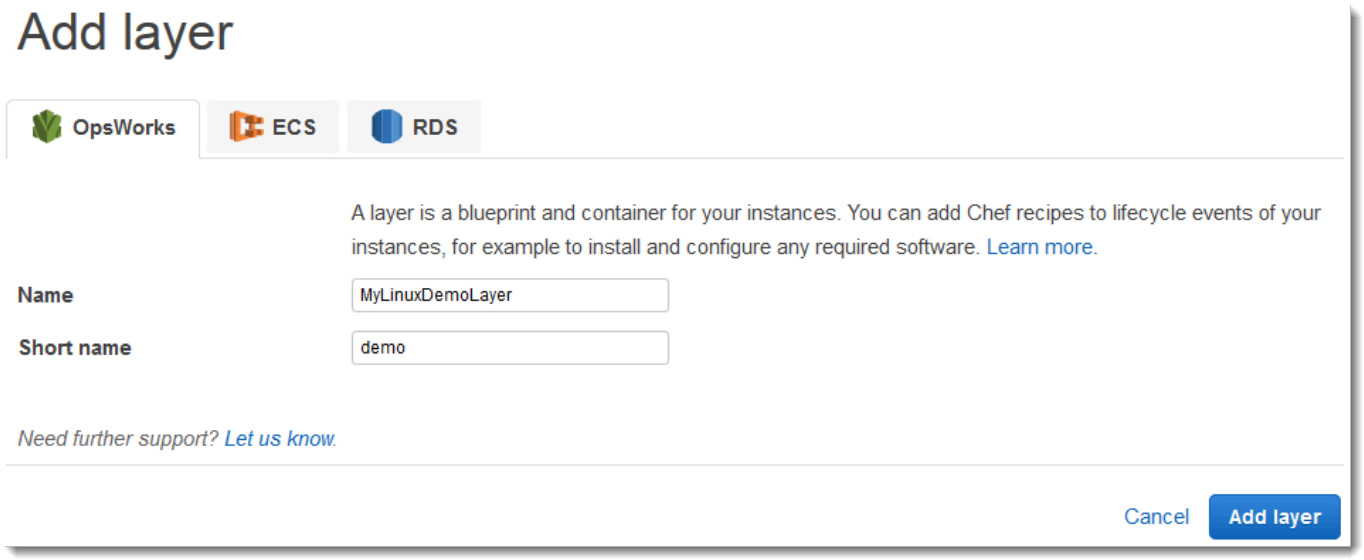
layer 是一組執行個體的藍圖，例如 Amazon EC2 執行個體。它會指定例如執行個體的設定、資源、安裝的套件和安全群組等資訊。接下來，將 layer 新增至堆疊。(如需 layer 的詳細資訊，請參閱 層。)

將 layer 新增至堆疊

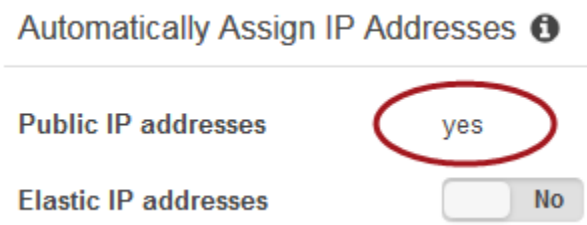
1. 在先前步驟中顯示的 MyLinuxDemoStack 頁面上，針對 Layers (Layer)，選擇 Add a layer (新增 layer)：



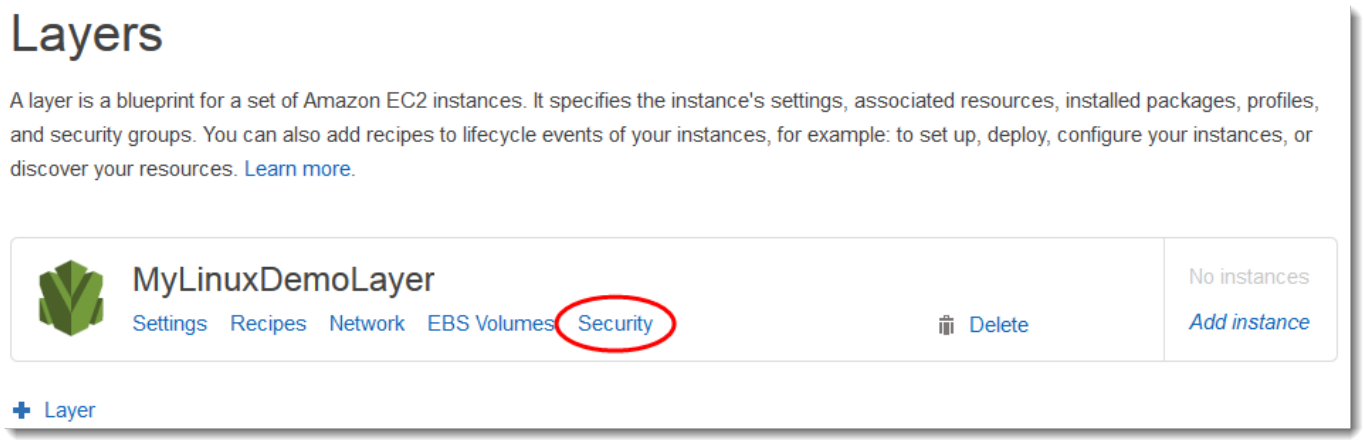
2. 即會顯示 Add Layer (新增 Layer) 頁面。在 OpsWorks 標籤上，針對 Name (名稱)，輸入 **`MyLinuxDemoLayer`**。(您可以輸入不同的名稱，但請務必在本演練的後續部分一律將 `MyLinuxDemoLayer` 取代為您選擇的名稱。)
3. 針對 Short name (短名)，輸入 **demo** (您可以輸入不同的值，但請務必在本演練全部範圍內將 demo 取代為您選擇的名稱)：



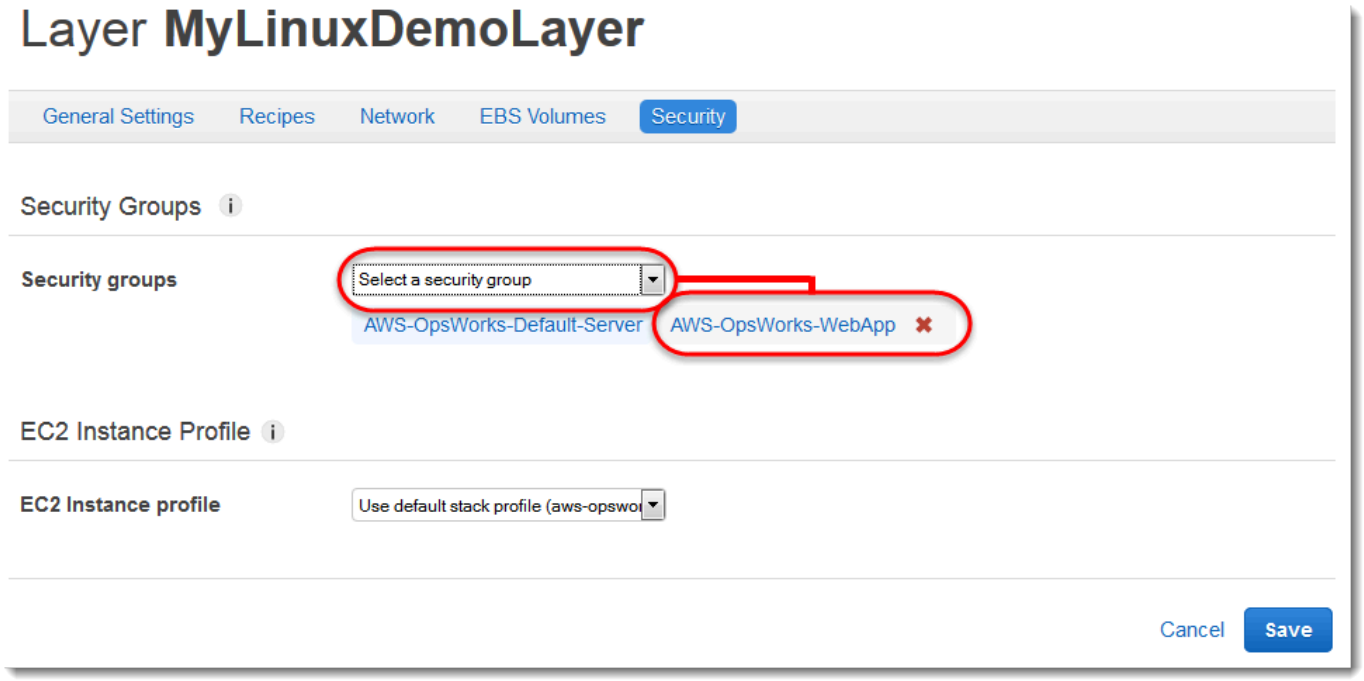
4. 選擇新增 layer。 AWS OpsWorks Stacks 會建立 layer 並顯示 Layers 頁面。
5. 在 Layers (Layer) 頁面上，針對 MyLinuxDemoLayer，選擇 Network (網路)。
6. 在 Network (網路) 標籤上的 Automatically Assign IP Addresses (自動指派 IP 地址) 之下，驗證 Public IP addresses (公有 IP 地址) 已設為 yes (是)。如果已進行變更，請選擇 Save (儲存)。



7. 在 Layers (Layer) 頁面上，選擇 Security (安全性)。



8. Layer MyLinuxDemoLayer 頁面即會顯示，並開啟 Security (安全性) 標籤。針對 Security groups (安全群組)，選擇 AWS-OpsWorks-WebApp，然後選擇 Save (儲存)：



9. AWS-OpsWorks-WebApp 安全群組即會新增至 layer。（此安全群組可讓使用者在本演練稍後連線至執行個體上的應用程式。 如果沒有此安全群組，使用者會在網頁瀏覽器中收到無法連線至執行個體的訊息。）

現在，您已擁有一個設定正確的 layer 可用於本演練。

在下一個步驟中，您將會指定要部署到執行個體的應用程式。

步驟 4：指定要部署到執行個體的應用程式

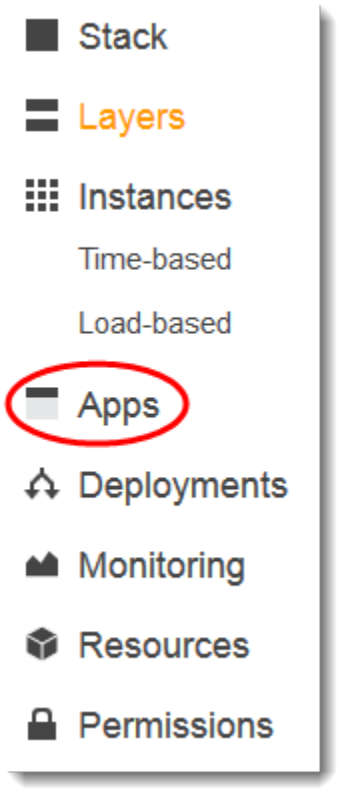
> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

告訴 AWS OpsWorks Stacks 您將在本演練稍後部署到執行個體的應用程式。在此內容中， AWS OpsWorks Stacks 會將應用程式定義為您想要在執行個體上執行的程式碼。(如需詳細資訊，請參閱 應用程式。)

本節中的程序適用於 Chef 12 和更新的堆疊。如需如何在 Chef 11 堆疊中將應用程式新增至 layer 的資訊，請參閱步驟 2.4：建立和部署應用程式 - Chef 11。

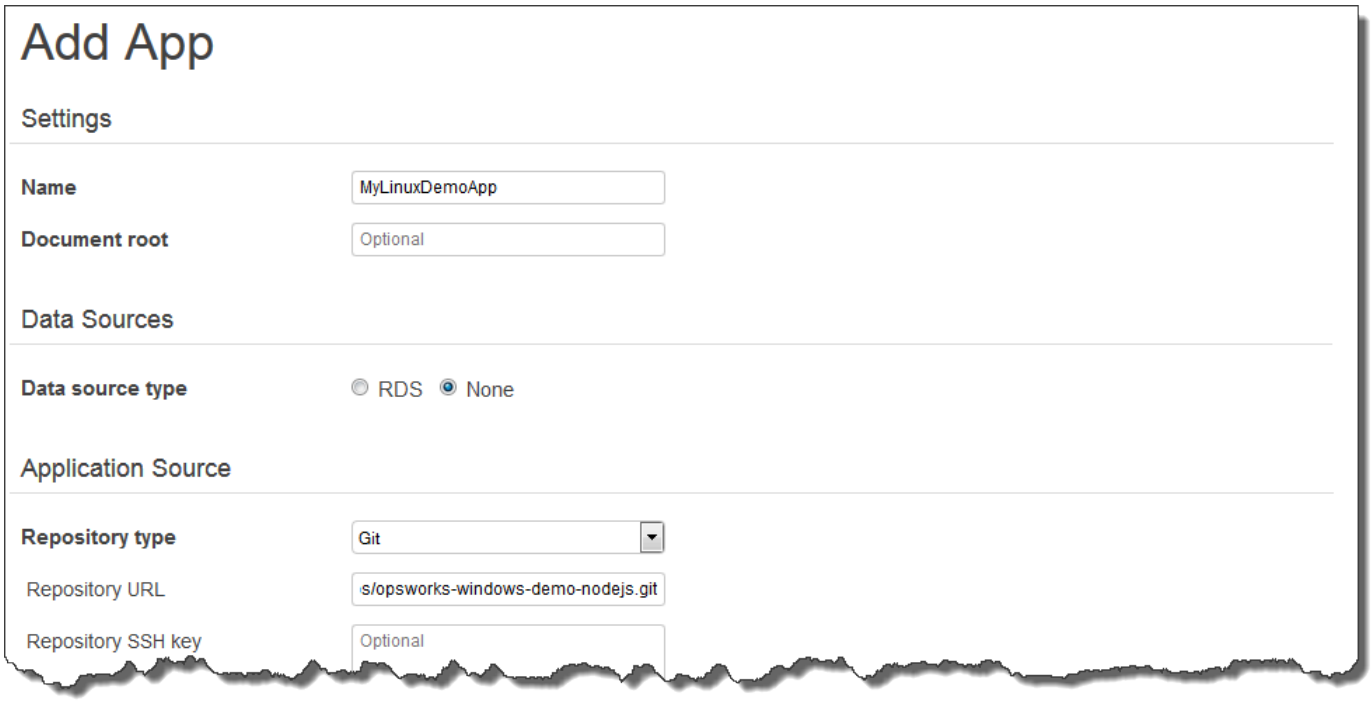
指定要部署的應用程式

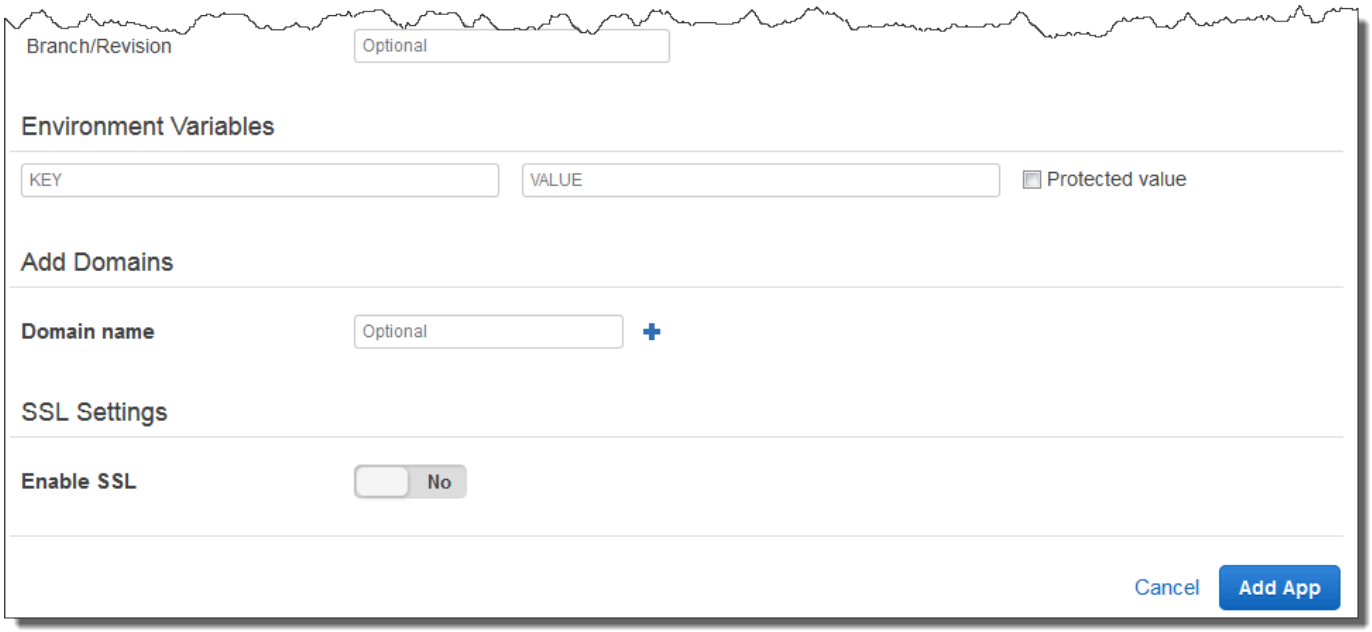
1. 在服務導覽窗格中，選擇 Apps (應用程式)：



2. 即會顯示 Apps (應用程式) 頁面。選擇 Add an app (新增應用程式用戶端)。即會顯示 Add App (新增應用程式) 頁面。
3. 針對 Settings (設定) 的 Name (名稱)，請輸入 **MyLinuxDemoApp**。(您可以輸入不同的名稱，但請務必在本演練的後續部分一律將 MyLinuxDemoApp 取代為您選擇的名稱。)
4. 針對 Application Source (應用程式來源) 的 Repository URL (儲存庫 URL)，輸入 **https://github.com/awslabs/opsworks-windows-demo-nodejs.git**
5. 保留下列項目的預設值：

- Settings (設定)、Document root (文件根) (空白)
- Data Sources (資料來源)、Data source type (資料來源類型) (None (無))
- Repository type (儲存庫類型) (Git)
- Repository SSH key (儲存庫 SSH 金鑰) (空白)
- Branch/Revision (分支/修訂) (空白)
- Environment Variables (環境變數) (空白 KEY (金鑰)、空白 VALUE (值)、不勾選 Protected Value (受保護的值))
- Add Domains (新增網域)、Domain Name (網域名稱) (空白)
- SSL Settings (SSL 設定)、Enable SSL (啟用 SSL) (No (否))





6. 選擇新增 App. AWS OpsWorks Stacks 新增應用程式，並顯示應用程式頁面。

現在，您已擁有一個設定正確的應用程式可用於本演練。

在下一個步驟中，您將啟動執行個體。

步驟 5：啟動執行個體

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

使用 AWS OpsWorks Stacks 啟動 Ubuntu Server Amazon EC2 執行個體。該執行個體會使用您稍早在本演練中建立之 layer 中定義的設定。(如需詳細資訊，請參閱 執行個體。)

啟動執行個體

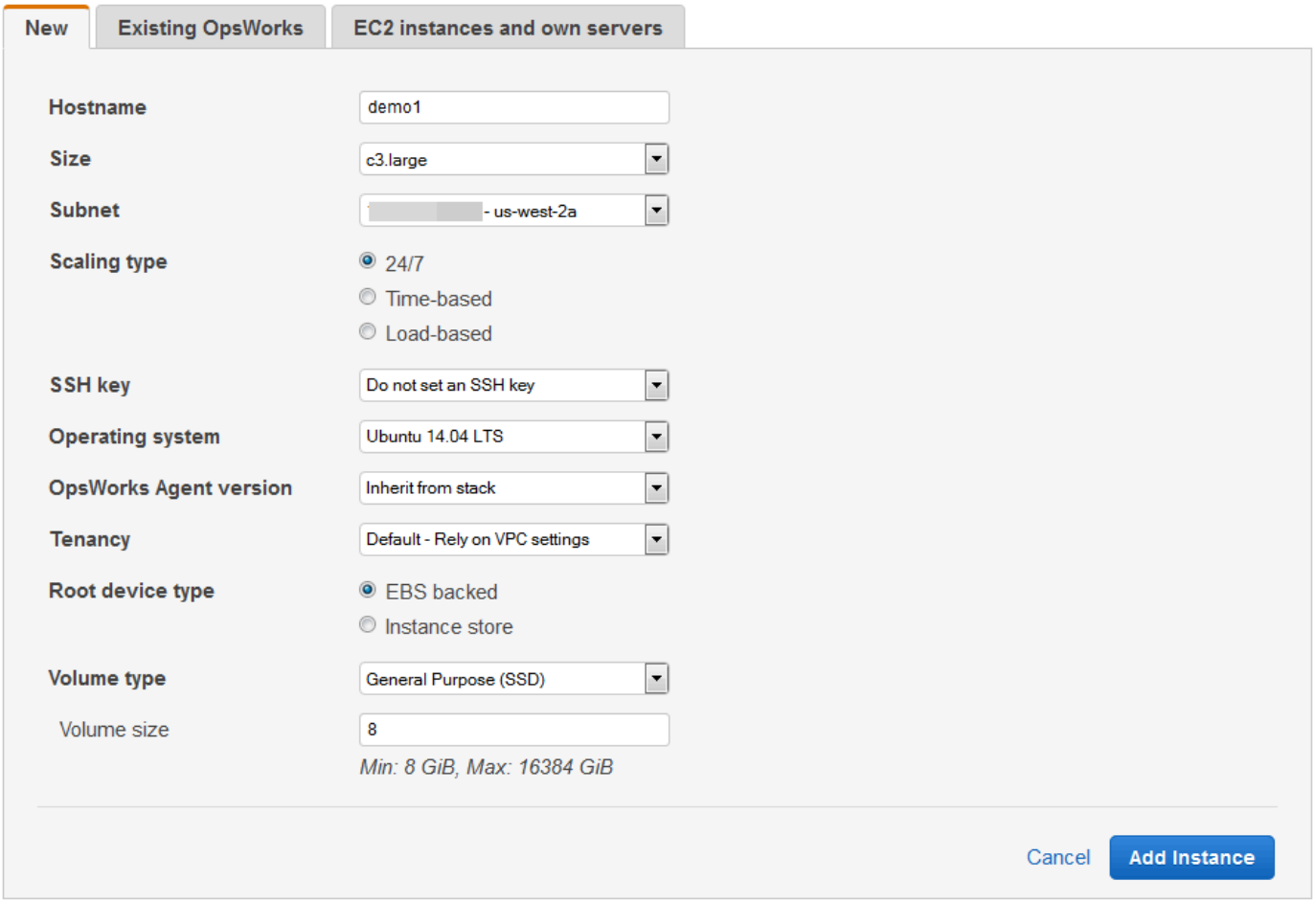
1. 在服務導覽窗格中，選擇 Instances (執行個體)。即會顯示 Instances (執行個體) 頁面。
2. 針對 MyLinuxDemoLayer，選擇 Add an instance (新增執行個體)。
3. 在 New (新增) 標籤上，請為下列各項保留預設值：

- Hostname (主機名稱) (demo1)
- Size (大小) (c3.large)
- Subnet (子網路) (*IP ##* us-west-2a)

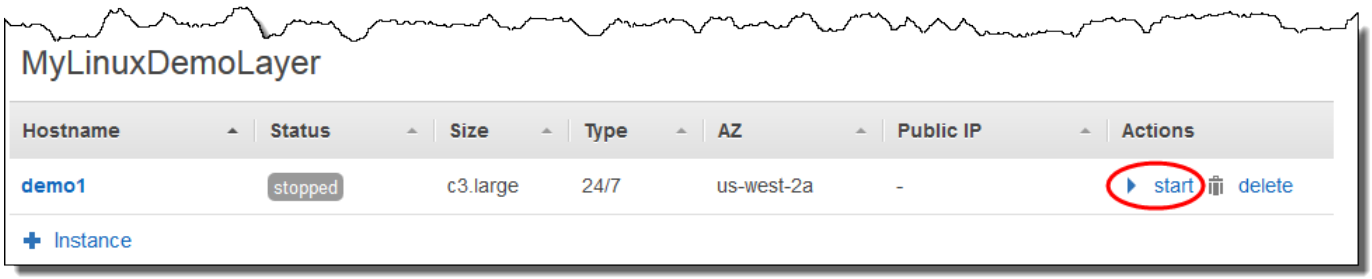
4. 選擇 Advanced (進階)。
5. 保留下列項目的預設值：

   - Scaling type (擴展類型) (24/7 (全年無休))
   - SSH key (SSH 金鑰) (Do not use a default SSH key (不使用預設 SSH 金鑰))
   - Operating system (作業系統) (Ubuntu 18.04 LTS)
   - OpsWorks Agent version (OpsWorks 代理程式版本) (Inherit from stack (繼承自堆疊))
   - Tenancy (租用) (Default - Rely on VPC settings (預設 – 依存 VPC 設定))
   - Root device type (根設備類型) (EBS backed (EBS 後端))
   - Volume type (磁碟區類型) (General Purpose (SSD) (一般用途 (SSD)))
   - Volume size (磁碟區大小) (8)

6. 您的結果將類似下列螢幕擷取畫面：



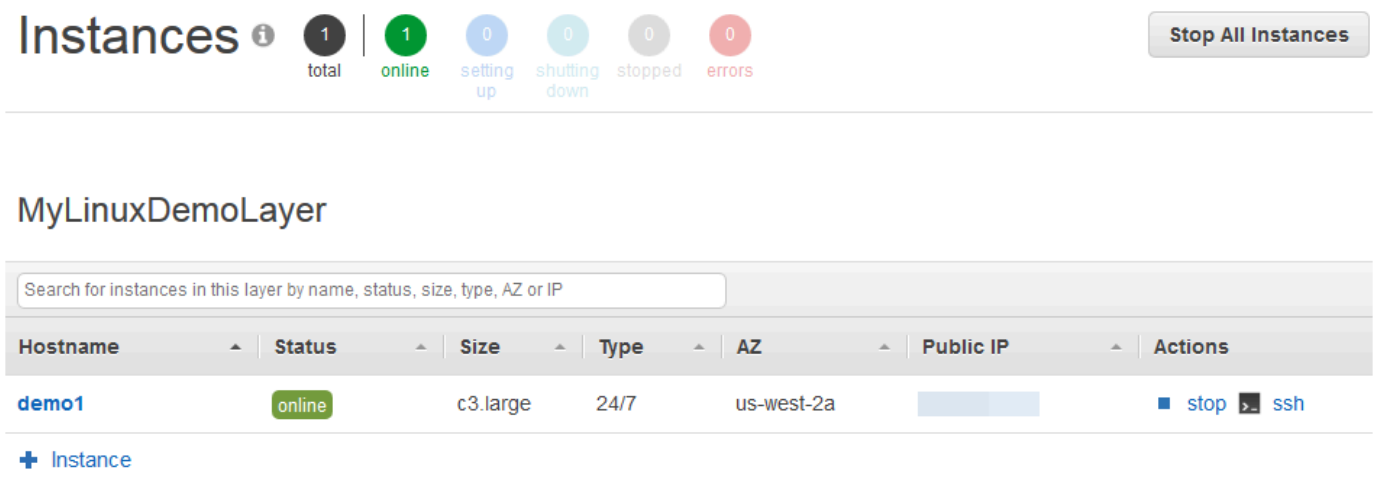
7. 選擇新增執行個體。 AWS OpsWorks Stacks 會將執行個體新增至 layer，並顯示執行個體頁面。
8. 針對 MyLinuxDemoLayer 之 demo1 的 Actions (動作)，選擇 start (啟動)：



9. 幾分鐘後，會發生下列情況：
   - setting up (設定) 圓圈會從 0 變更為 1。
   - Status (狀態) 會從 stopped 依次變更為 requested、pending、booting、running_setup，最後成為 online。請注意，此程序需要幾分鐘的時間。

- 當 Status (狀態) 變更為 online 後，setting up 圓圈指標會從 1 變更為 0，online 圓圈會從 0 變更為 1 並變更為亮綠色。請在 online (線上) 圓圈變更為亮綠色並顯示 1 個執行個體在線上後再繼續。

10. 繼續之前，您的結果必須符合下列的螢幕擷取畫面 (如果您收到故障訊息，建議您參閱偵錯和故障診斷指南)：



現在，您已經準備好用於部署應用程式的執行個體。

> **Note**
>
> 如果您想要登入執行個體以進一步探索，您必須先向 AWS OpsWorks Stacks 提供公有 SSH 金鑰的相關資訊（您可以使用 ssh-keygen 或 PuTTYgen 等工具建立），然後您必須在`MyLinuxDemoStack`堆疊上設定許可，讓使用者能夠登入執行個體。如需說明，請參閱 註冊使用者的公有 SSH 金鑰 與 使用 SSH 登入。如果您計劃使用 SSH 透過 PuTTY 連線至執行個體，請參閱 AWS 文件中的使用 PuTTY 從 Windows 連線至 Linux 執行個體。

在下一個步驟中，您會將應用程式部署到執行個體。

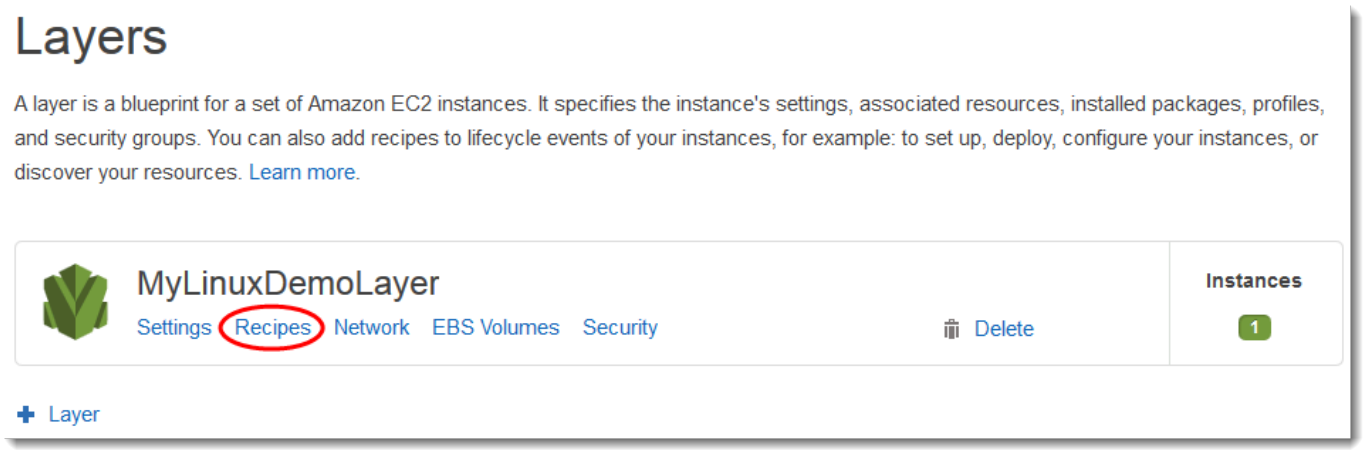
步驟 6：將應用程式部署到執行個體

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

在此步驟中，您會將來自 GitHub 的應用程式部署到執行中的執行個體。(如需詳細資訊，請參閱 部署應用程式。) 部署應用程式之前，您必須指定「配方」用於協調部署。方法是一個 Chef 概念。配方是以 Ruby 語言語法編寫的指示，指定要使用的資源以及套用那些資源的順序。(如需詳細資訊，請前往了解 Chef 網站上的關於配方。)

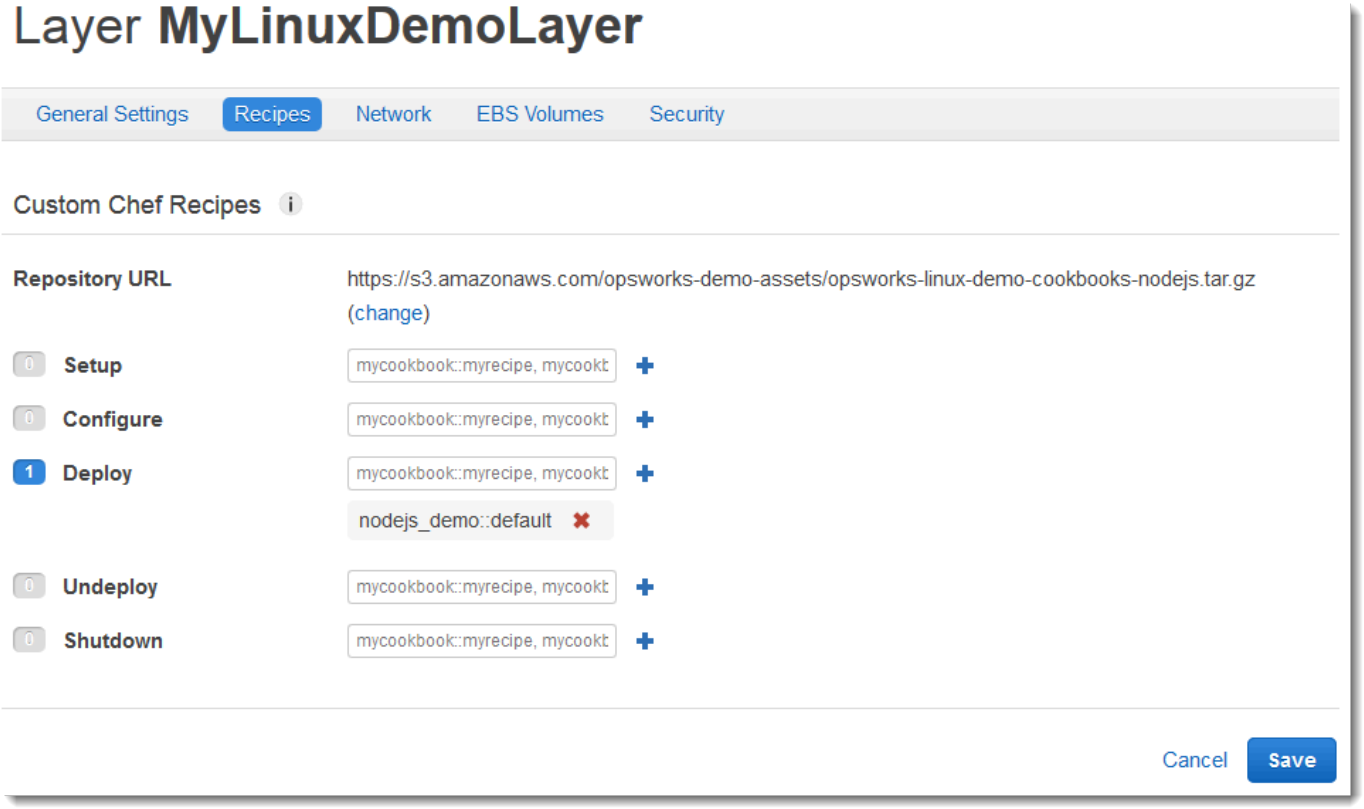
指定配方以用來將應用程式部署到執行個體

1. 在服務導覽窗格中，選擇 Layers (Layer)。即會顯示 Layers (Layer) 頁面。
2. 針對 MyLinuxDemoLayer，選擇 Recipes (配方)。



Layer MyLinuxDemoLayer 頁面即會顯示，並開啟 Recipes (配方) 標籤。

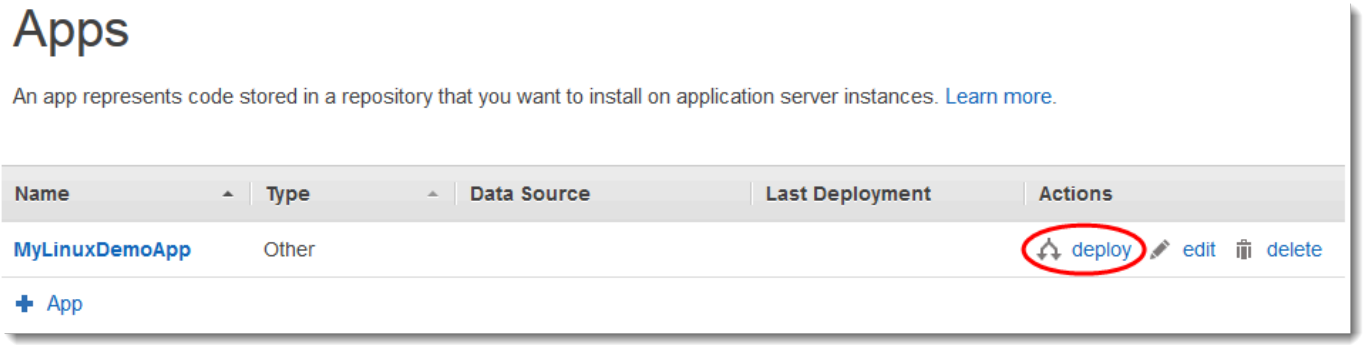
3. 針對 Custom Chef Recipes (自訂 Chef 配方) 的 Deploy (部署)，輸入 **`nodejs_demo::default`**，然後按 Enter 鍵。`nodejs_demo` 是技術指南的名稱，`default` 是技術指南內目標配方的名稱。(若要探索配方的程式碼，請參閱 深入了解：探索本演練中使用的技術指南。) 您的結果必須符合下列螢幕擷取畫面：



4. 選擇 Save. AWS OpsWorks Stacks 將配方新增至 layer 的部署生命週期事件。

將應用程式部署至執行個體

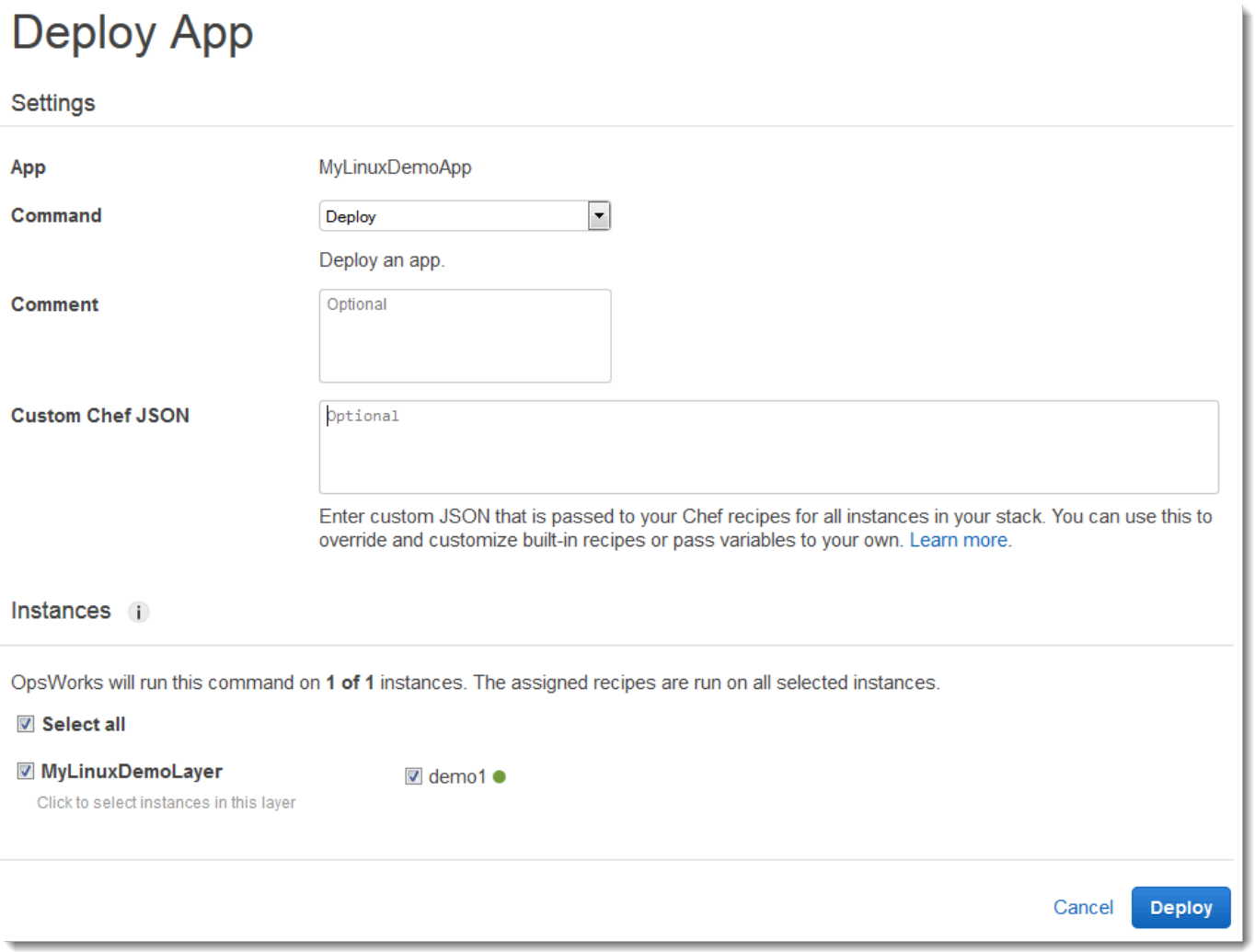
1. 在服務導覽窗格中，選擇 Apps (應用程式)。即會顯示 Apps (應用程式) 頁面。
2. 針對 MyLinuxDemoApp 的 Actions (動作)，選擇 deploy (部署)，如下列螢幕擷取畫面所示：



3. 在 Deploy App (部署應用程式) 頁面上，請為下列各項保留預設值：
   - Command (命令) (Deploy (部署))

- Comment (註解) (空白)
- Settings (設定)、Advanced (進階)、Custom Chef JSON (自訂 Chef JSON) (空白)
- Instances (執行個體)、Advanced (進階) (已核取 Select all (全選)、已核取 MyLinuxDemoLayer、已核取 demo1)。

4. 您的結果必須符合下列螢幕擷取畫面：



5. 選擇部署。Deployment MyLinuxDemoApp – deploy 頁面即會顯示。Status (狀態) 會從 running (執行中) 變更為 successful (成功)。在 demo1 旁會顯示一個旋轉圓圈，然後變為綠色核取記號。請注意，此程序需要幾分鐘的時間。請在您看到 Status (狀態) 變為 successful (成功) 以及綠色核取記號後，再繼續。
6. 您的結果必須符合下列螢幕擷取畫面，當然 Created at (建立於)、Completed at (完成於)、Duration (持續時間) 和 User (使用者) 這幾項除外。如果 status (狀態) 為 failed (失敗)，則進行故障診斷；請針對 Log (日誌)，選擇 show (顯示) 以取得失敗的詳細資訊：

您現已成功將應用程式部署到執行個體。

在下一個步驟中，您將會測試執行個體上已部署的應用程式。

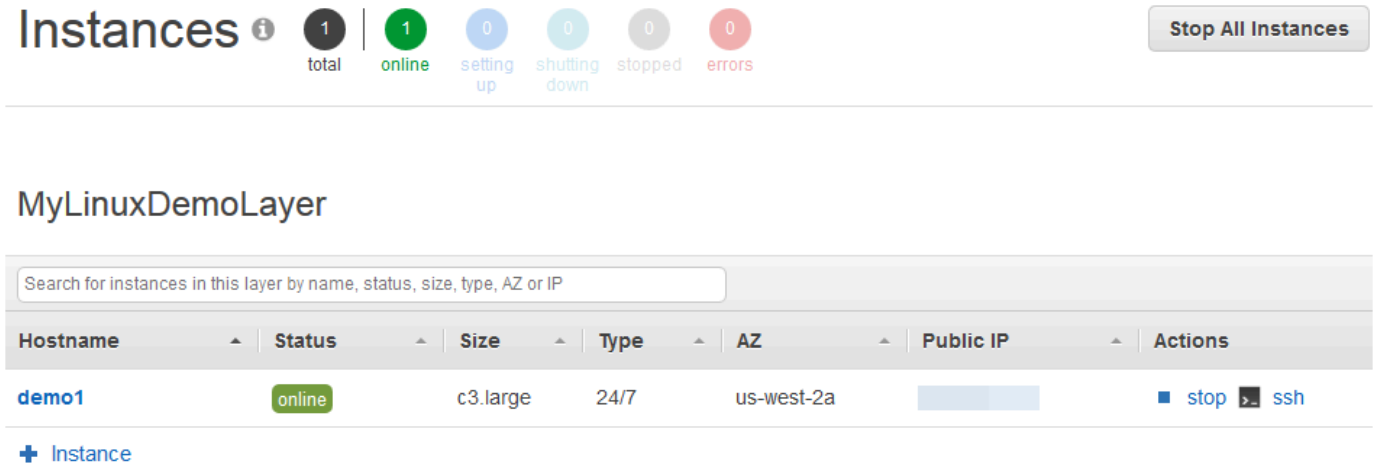
步驟 7：測試執行個體上已部署的應用程式

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

現在，測試執行個體上已部署的應用程式。

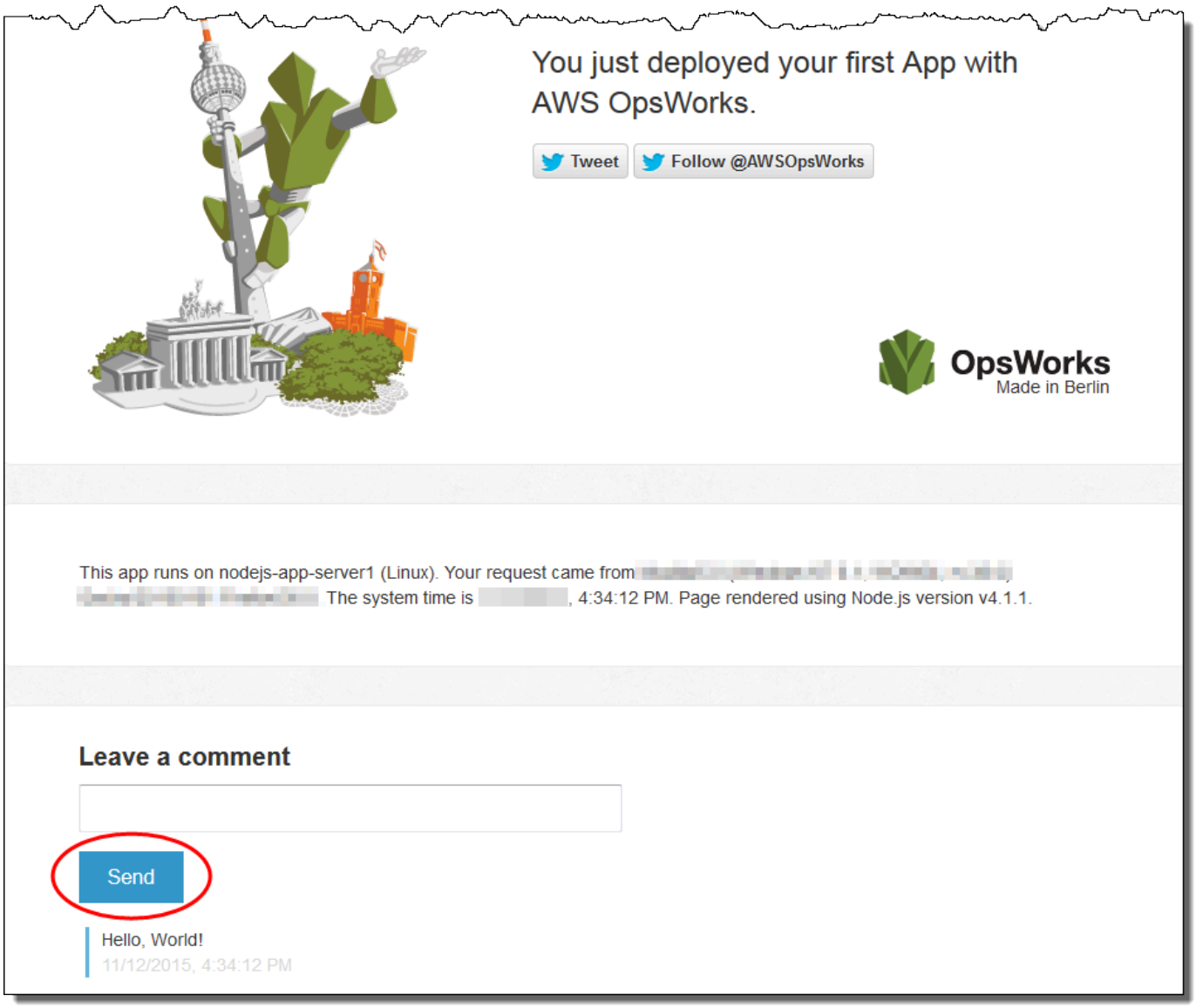
測試執行個體上的部署

1. 在服務導覽窗格中，選擇 Instances (執行個體)。即會顯示 Instances (執行個體) 頁面。
2. 針對 MyLinuxDemoLayer 之 demo1 的 Public IP (公有 IP)，選擇 IP 地址：



新的 Web 瀏覽器標籤會顯示該應用程式。

3. 在賀辭網頁的 Leave a comment (留下評論) 文字方塊中輸入評論，然後選擇 Send (傳送) 測試應用程式。應用程式會將您的評論新增到網頁。繼續留下評論，並依您需要隨時選擇 Send (傳送)：



4. 如果您有 Twitter 帳戶，請選擇 Tweet (推文) 或 Follow @AWSOpsWorks (關注 @AWSOpsWorks)，然後遵循畫面指示發佈有關此應用程式的推文或關注 @AWSOpsWorks。

您現已成功測試執行個體上所部署的應用程式。

在下一個步驟中，您可以清除用於本演練 AWS 的資源。此步驟為選用。當您繼續進一步了解 Stacks AWS OpsWorks 時，建議您繼續使用這些 AWS 資源。不過，保留這些 AWS 資源可能會對 AWS 您的帳戶產生持續費用。如果您想要將這些 AWS 資源保留以供日後使用，您現在已完成本演練，您可以直接跳到 後續步驟。

步驟 8 (選用)：清理

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

為了避免 AWS 您的帳戶產生額外費用，您可以刪除 AWS 用於此演練的資源。這些 AWS 資源包括 AWS OpsWorks Stacks 堆疊和堆疊的元件。（如需詳細資訊，請參閱 AWS OpsWorks 定價。）不過，當您繼續進一步了解 Stacks AWS OpsWorks 時，建議您繼續使用這些 AWS 資源。如果您想要讓這些 AWS 資源保持可用，您現在已完成本演練，您可以跳至 後續步驟。

存放在您為此逐步解說建立的資源裡的內容，可包含個人識別資訊。如果您不希望再將此資訊存放在 AWS，請遵循本主題的步驟。

從堆疊刪除應用程式

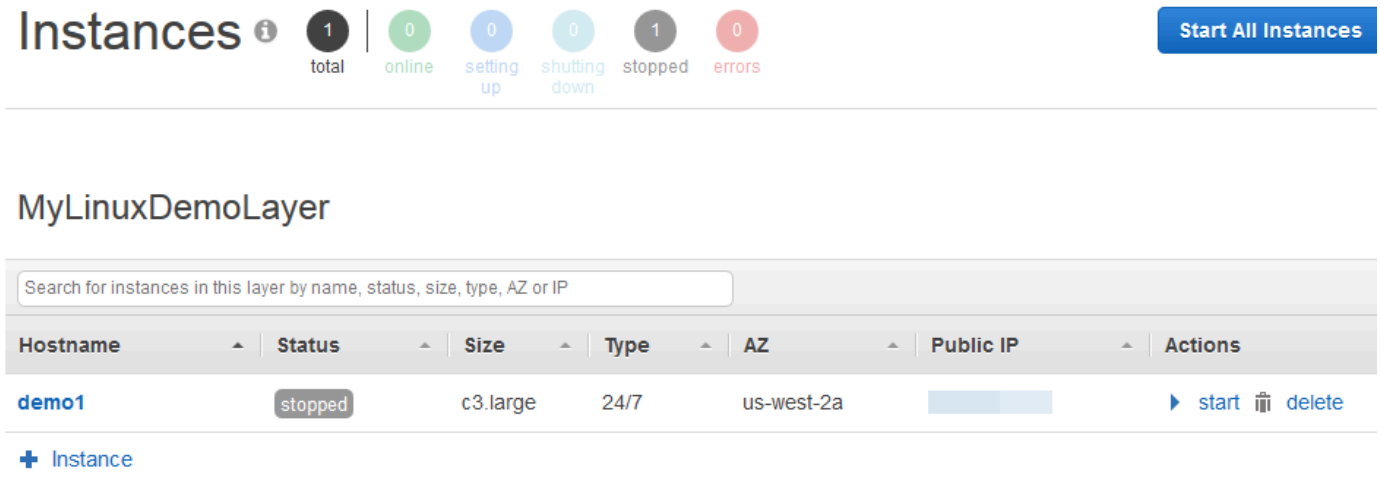
1. 在 AWS OpsWorks Stacks 主控台的服務導覽窗格中，選擇應用程式。即會顯示 Apps (應用程式) 頁面。
2. 針對 MyLinuxDemoApp 的 Actions (動作)，選擇 delete (刪除)。顯示確認訊息時，選擇 Delete. AWS OpsWorks Stacks 會刪除應用程式。

刪除堆疊的執行個體

1. 在服務導覽窗格中，選擇 Instances (執行個體)。即會顯示 Instances (執行個體) 頁面。
2. 針對 MyLinuxDemoLayer 之 demo1 的 Actions (動作)，選擇 stop (停止)。在您看見確認訊息時，選擇 Stop (停止)。會發生下列情況。

   - Status (狀態) 從 online (線上) 變更為 stopping (停止中)，最後變更為 stopped (已停止)。
   - online (線上) 從 1 變更為 0。
   - shutting down (關機中) 從 0 變更為 1，最後變回 0。
   - stopped (已停止) 最後從 0 變更為 1。

   此程序需要幾分鐘的時間。當 AWS OpsWorks Stacks 完成時，會顯示下列結果。



3. 針對 Actions (動作)，選擇 delete (刪除)。當您看到確認訊息時，請選擇 Delete。 AWS OpsWorks Stacks 會刪除執行個體，並顯示無執行個體訊息。

刪除堆疊

1. 在服務導覽窗格中，選擇 Stack (堆疊)。即會顯示 MyLinuxDemoStack 頁面。
2. 選擇 Delete Stack (刪除堆疊)。當您看到確認訊息時，請選擇 Delete. AWS OpsWorks Stacks 刪除堆疊並顯示 OpsWorks Dashboard 頁面。

或者，如果您不想重複使用使用者和 Amazon EC2 金鑰對來存取其他 AWS 服務和 EC2 執行個體，您可以刪除用於此演練的使用者和 Amazon EC2 金鑰對。如需說明，請參閱刪除 IAM 使用者、Amazon EC2 金鑰對和 Linux 執行個體。

您現已完成本演練。如需詳細資訊，請參閱後續步驟。

後續步驟

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

現在您已完成本演練，您可以進一步了解如何使用 AWS OpsWorks Stacks：

- 探索您在本演練中使用的技術指南和應用程式。請參閱 深入了解：探索本演練中使用的技術指南 和 深入了解：探索本演練中使用的應用程式。
- 練習搭配 Windows 執行個體使用 AWS OpsWorks Stacks。請參閱 入門：Windows。
- 了解如何建立新的堆疊以進一步了解堆疊。
- 透過編輯 OpsWorks Layer 的組態進一步了解 layer。
- 透過將執行個體新增至 Layer進一步了解執行個體。
- 透過部署應用程式進一步了解應用程式。
- 進一步了解 技術指南和配方。
- 建立您自己的技術指南。請參閱 入門：技術指南。
- 了解使用安全與許可控制堆疊的存取權。

深入了解：探索本演練中使用的技術指南

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本主題說明 AWS OpsWorks Stacks 用於演練的技術指南。

「技術指南」是一個 Chef 概念。技術指南是封存檔案，其中包含組態資訊，如配方、屬性值、檔案、範本、程式庫、定義和自訂資源。「配方」也是一個 Chef 概念。配方是以 Ruby 語言語法編寫的指示，指定要使用的資源以及套用那些資源的順序。如需詳細資訊，請前往了解 Chef 網站上的關於技術指南和關於配方。

若要查看本演練中使用的技術指南內容，您可以將 opsworks-linux-demo-cookbooks-nodejs.tar.gz 檔案內容擷取到本機工作站的空目錄中。(您也可以登入已部署技術指南的執行個體，來探索 `/var/chef/cookbooks` 目錄中的內容)。

在 `cookbooks/nodejs_demo/recipes` 目錄中的 `default.rb` 檔案，是技術指南執行其程式碼的位置：

```
app = search(:aws_opsworks_app).first
app_path = "/srv/#{app['shortname']}"
```

```
package "git" do
  options "--force-yes" if node["platform"] == "ubuntu" && node["platform_version"] ==
 "18.04"
end

application app_path do
  javascript "4"
  environment.update("PORT" => "80")

  git app_path do
    repository app["app_source"]["url"]
    revision app["app_source"]["revision"]
  end

  link "#{app_path}/server.js" do
    to "#{app_path}/index.js"
  end

  npm_install
  npm_start
end
```

下列是該檔案執行的作業：

- `search(:aws_opsworks_app).first` 使用 Chef 搜尋來查詢最終將會部署到執行個體的應用程式。此資訊包含如應用程式短名及其來源儲存庫詳細資訊等設定。由於本演練只部署一個應用程式，Chef 搜尋會在執行個體上 `aws_opsworks_app` 搜尋索引內的第一個項目取得這些設定。每當執行個體啟動時， AWS OpsWorks Stacks 都會將此和其他相關資訊儲存為執行個體本身上的一組資料包，而您可以透過 Chef 搜尋取得資料包內容。雖然您可以使用將這些設定硬式編碼在此配方中，但使用資料包和 Chef 搜尋是更穩健的方法。如需資料包的詳細資訊，請參閱 AWS OpsWorks Stacks Data Bag 參考。另請參閱 Learn Chef 網站上的關於資料包。如需 Chef 搜尋的詳細資訊，請前往了解 Chef 網站上的關於搜尋。
- `package` 資源會在執行個體上安裝 Git。
- `application` 資源說明和部署 Web 應用程式：
  - `javascript` 是要安裝的 JavaScript 執行時間版本。
  - `environment` 設定環境變數。
  - `git` 從指定的儲存庫和分支取得來源碼。
  - `app_path` 是要複製儲存庫到的目標路徑。如果該路徑不存在於執行個體上， AWS OpsWorks Stacks 會建立路徑。

- `link` 建立符號連結。
- `npm_install` 安裝 Node Package Manager，是 Node.js 的預設套件管理工具。
- `npm_start` 執行 Node.js。

雖然 AWS OpsWorks Stacks 已建立用於本演練的技術指南，但您可以建立自己的技術指南。如要瞭解如何作業，請參閱入門：技術指南。此外，請前往了解 Chef 網站上的關於技術指南、關於配方和了解 Ubuntu 上的 Chef 基本概念，以及 Chef 入門網站上的使用 Chef 的第一步。

深入了解：探索本演練中使用的應用程式

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本主題說明 AWS OpsWorks Stacks 為此演練部署至執行個體的應用程式。

若要查看應用程式的來源碼，請將 opsworks-windows-demo-nodejs GitHub 儲存庫內容擷取到本機工作站的空目錄中。您也可以登入已部署技術指南的執行個體，來探索 `/srv/mylinuxdemoapp` 目錄中的內容。

`index.js` 檔案包含該應用程式最重要的程式碼：

```
var express = require('express');
var app = express();
var path = require('path');
var os = require('os');
var bodyParser = require('body-parser');
var fs = require('fs');

var add_comment = function(comment) {
  var comments = get_comments();
  comments.push({"date": new Date(), "text": comment});
  fs.writeFileSync('./comments.json', JSON.stringify(comments));
};

var get_comments = function() {
  var comments;
```

```
  if (fs.existsSync('./comments.json')) {
    comments = fs.readFileSync('./comments.json');
    comments = JSON.parse(comments);
  } else {
    comments = [];
  }
  return comments;
};

app.use(function log (req, res, next) {
  console.log([req.method, req.url].join(' '));
  next();
});
app.use(express.static('public'));
app.use(bodyParser.urlencoded({ extended: false }))

app.set('view engine', 'jade');
app.get('/', function(req, res) {
  var comments = get_comments();
  res.render("index",
    { agent: req.headers['user-agent'],
      hostname: os.hostname(),
      nodeversion: process.version,
      time: new Date(),
      admin: (process.env.APP_ADMIN_EMAIL || "admin@unconfigured-value.com" ),
      comments: get_comments()
    });
});

app.post('/', function(req, res) {
  var comment = req.body.comment;
  if (comment) {
    add_comment(comment);
    console.log("Got comment: " + comment);
  }
  res.redirect("/#form-section");
});

var server = app.listen(process.env.PORT || 3000, function() {
  console.log('Listening on %s', process.env.PORT);
});
```

下列是該檔案執行的作業：

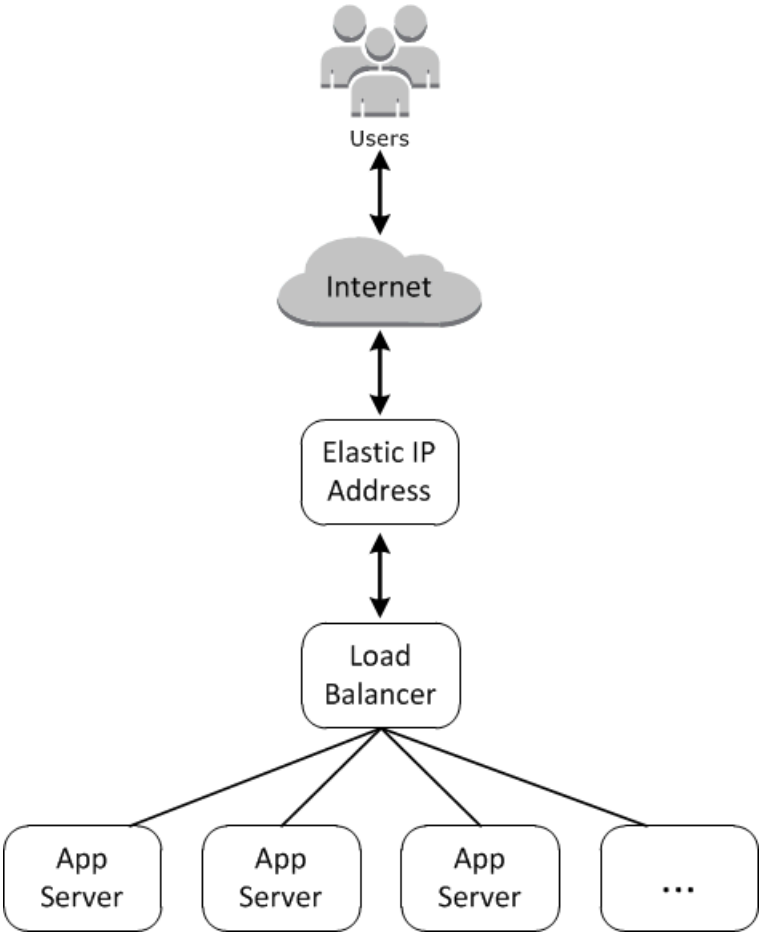
- `require` 載入模組，其中包含此 Web 應用程式依預期執行所需要的一些相依程式碼。
- `add_comment` 和 `get_comments` 函數將資訊寫入 `comments.json` 檔案並從中讀取資訊。
- 如需 `app.get`、`app.listen`、`app.post`、`app.set` 和 `app.use` 的資訊，請參閱 Express API 參考。

若要了解如何建立和封裝您的應用程式以便部署，請參閱 應用程式來源。

Windows 堆疊入門

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

雲端型應用程式通常需要一組必須共同建立和管理的相關資源，例如應用程式伺服器、資料庫伺服器等。此執行個體的集合稱為「堆疊」。簡易的應用程式堆疊看起來可能如下。



基本架構由下列項目組成：

- 用於接收使用者請求的彈性 IP 地址。
- 將傳入請求平均分散至應用程式伺服器的負載平衡器。
- 一組數量足以處理流量的應用程式伺服器執行個體。

此外，您通常需要一種將應用程式分散至應用程式伺服器、管理使用者許可等的方式。

AWS OpsWorks Stacks 提供簡單且直接的方式來建立和管理堆疊及其相關聯的應用程式和資源。本章介紹 AWS OpsWorks Stacks 的基本概念，以及一些更複雜的功能，引導您完成圖表中建立應用程式伺服器堆疊的程序。它使用 Stacks AWS OpsWorks 易於遵循的增量開發模型：設定基本堆疊，並在正常運作後新增元件，直到您達到功能完整的實作為止。

- 步驟 1：完成事前準備示範如何進行設定以開始演練。
- 步驟 2：建立基本應用程式伺服器堆疊 說明如何建立基本堆疊以支援網際網路資訊服務 (IIS) 並將應用程式部署到伺服器。

- 步驟 3：橫向擴展 IISExample說明如何透過新增更多應用程式伺服器來擴展堆疊以處理增加的負載、將傳入流量分散的負載平衡器，以及接收傳入請求的彈性 IP 地址。

主題

- 步驟 1：完成事前準備
- 步驟 2：建立基本應用程式伺服器堆疊
- 步驟 3：橫向擴展 IISExample
- 後續步驟

步驟 1：完成事前準備

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

完成下列步驟，才能開始演練。這些設定步驟包括註冊 AWS 帳戶、建立管理使用者，以及將存取許可指派給 AWS OpsWorks Stacks。

若您已完成入門：範例或入門：Linux演練，您便已完成本演練的事前準備，可直接跳到步驟 2：建立基本應用程式伺服器堆疊。

主題

- 註冊 AWS 帳戶
- 建立具有管理存取權的使用者
- 指派服務存取許可
- 確保 Stacks AWS OpsWorks 使用者已新增至您的網域

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權 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。

指派服務存取許可

將 AWS OpsWorks `AWSOpsWorks_FullAccess`和 `AmazonS3FullAccess`許可新增至您的角色或使用者，以啟用 Stacks 服務的存取權 （以及 AWS OpsWorks Stacks 依賴的相關服務）。

如需新增許可的詳細資訊，請參閱新增 IAM 身分許可 （主控台）。

確保 Stacks AWS OpsWorks 使用者已新增至您的網域

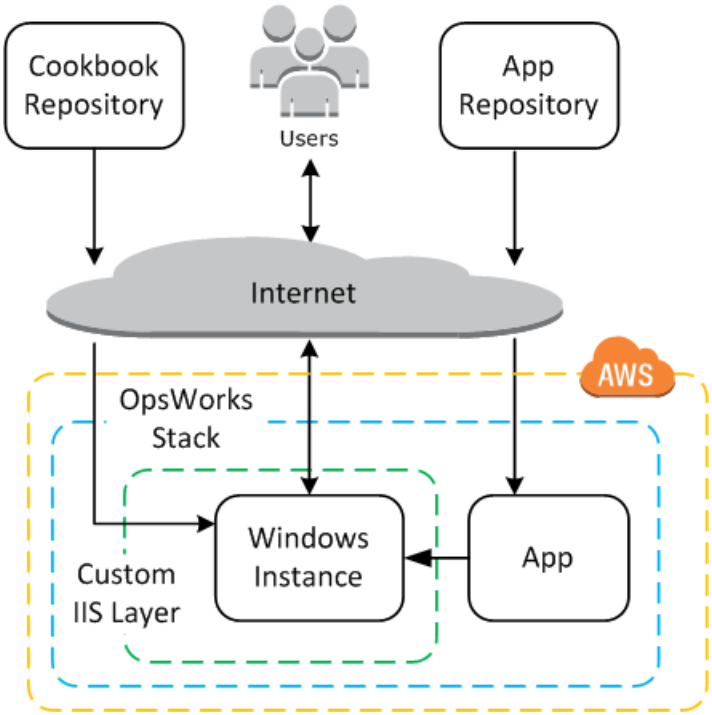
在 Chef 12.2 堆疊中，隨附的 `aws_opsworks_users` 技術指南會建立對 Windows 型的執行個體擁有 SSH 和遠端桌面通訊協定 (RDP) 存取權的使用者。當您將堆疊中的 Windows 執行個體加入 Active Directory 網域時，如果 AWS OpsWorks Stacks 使用者不存在於 Active Directory 中，此技術指南執行可能會失敗。如果在 Active Directory 中未識別出使用者，則在您將執行個體加入網域後重新啟動時，執行個體會進入 `setup failed` 狀態。針對加入網域的 Windows 執行個體，在使用者許可頁面對 AWS OpsWorks Stacks 使用者授予 SSH/RDP 存取權並不足夠。

在將 Chef 12.2 堆疊中的 Windows 執行個體加入 Active Directory 網域之前，請確定 Windows AWS OpsWorks 型堆疊的所有 Stacks 使用者都是網域的成員。最好的方法是在建立以 Windows 為基礎的堆疊之前，使用 IAM 設定聯合身分，然後在將堆疊中的執行個體加入網域之前，將聯合身分使用者匯入 AWS OpsWorks Stacks。如需如何執行此操作的詳細資訊，請參閱《AWS 安全部落格》中的使用 Windows Active Directory、ADFS 和 SAML 2.0 啟用與 AWS 的聯合，以及《IAM 使用者指南》中的聯合現有使用者。

步驟 2：建立基本應用程式伺服器堆疊

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

基本應用程式伺服器堆疊包含一個具有公有 IP 地址的應用程式伺服器執行個體，以接收使用者請求。應用程式程式碼和任何相關檔案均存放於單獨的儲存庫中，並從該位置部署到伺服器。下圖說明此類堆疊。



該堆疊具有下列元件：

- layer 代表執行個體的群組，並指定設定它們的方式。

  此範例中的 layer 代表 IIS 執行個體群組。

- 執行個體，代表 Amazon EC2 執行個體。

  在此案例中，layer 設定單一執行個體以執行 IIS，但 layer 可具有任意數目的執行個體。

- 「應用程式」包含在執行個體上安裝應用程式的必要資訊。

- 「技術指南」包含支援自訂 IIS layer 的 Chef 配方。技術指南和應用程式程式碼存放在遠端儲存庫中，例如 Amazon S3 儲存貯體或 Git 儲存庫中的封存檔案。

下列各節說明如何使用 AWS OpsWorks Stacks 主控台來建立堆疊並部署應用程式。

主題

- 步驟 2.1：建立堆疊
- 步驟 2.2：授權 RDP 存取
- 步驟 2.3：實作自訂技術指南
- 步驟 2.4：新增 IIS Layer
- 步驟 2.5：部署應用程式

步驟 2.1：建立堆疊

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以透過建立堆疊來啟動 AWS OpsWorks Stacks 專案，堆疊可做為執行個體和其他資源的容器。堆疊組態會指定某些由堆疊的所有執行個體共享之基本設定，如 AWS 區域和預設作業系統。

建立新堆疊

1. 新增堆疊

   如果您尚未執行此作業，請登入 AWS OpsWorks Stacks 主控台。

   - 若帳戶沒有任何現有的堆疊，您會看到 Welcome to AWS OpsWorks (歡迎使用 AWS OpsWorks) 頁面。請選擇 Add your first stack (新增您的第一個堆疊)。
   - 否則，您會看到 AWS OpsWorks Stacks 儀表板，其中列出您帳戶的堆疊；選擇新增堆疊。

2. 設定堆疊

   在 Add Stack (新增堆疊) 頁面上，選擇 Chef 12 stack (Chef 12 堆疊) 並指定下列設定：

Stack name (堆疊名稱)

輸入堆疊的名稱，其中可包含英數字元 (a–z、A–Z 和 0–9) 和連字號 (-)。本演練的範例堆疊名為 **IISWalkthrough**。

區域

選取美國西部 （奧勒岡） 做為堆疊的區域。

您可以在任何區域中建立堆疊，但我們建議美國西部 （奧勒岡） 進行教學課程。

預設作業系統

選擇 Windows，然後指定 Microsoft Windows Server 2022 Base，這是預設設定。

使用自訂 Chef 技術指南

針對本演練的用途，請為此選項指定 No (否)。

3. 選擇 Advanced (進階) 以確認您已選取一個 IAM 角色，以及預設的 IAM 執行個體描述檔。

   IAM 角色

   指定堆疊的 IAM (AWS Identity and Access Management) 角色。 AWS OpsWorks Stacks 需要存取其他 AWS 服務才能執行任務，例如建立和管理 Amazon EC2 執行個體。IAM 角色會指定 AWS OpsWorks Stacks 擔任的角色，以代表您使用其他 AWS 服務。如需詳細資訊，請參閱允許 AWS OpsWorks Stacks 代表您。

   - 如果您的帳戶具有現有的 AWS OpsWorks Stacks IAM 角色，您可以從清單中選取它。

     如果角色是由 AWS OpsWorks Stacks 建立，則會命名為 `aws-opsworks-service-role`。

   - 否則，請選取新 IAM 角色以指示 AWS OpsWorks Stacks 為您建立具有正確許可的新角色。

     注意：如果您有 AWS OpsWorks Stacks 完整存取許可，則建立新角色需要幾個額外的 IAM 許可。如需詳細資訊，請參閱範例政策。

4. 接受其他設定的預設值，然後選擇 Add Stack (新增堆疊)。如需各種堆疊設定的詳細資訊，請參閱建立新的堆疊。

步驟 2.2：授權 RDP 存取

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

現在您已建立堆疊，接下來您將建立一個 layer，並將 Windows 執行個體新增至該 layer。但是，您必須先設定堆疊以讓您使用 RDP 連線到自訂 layer 的執行個體，然後才能執行上述作業。對此，您必須執行下列作業：

- 將傳入規則新增至控制 RDP 存取的安全群組。
- 設定此堆疊的 AWS OpsWorks Stacks 許可，以允許 RDP 存取。

當您在區域中建立第一個堆疊時， AWS OpsWorks Stacks 會建立一組安全群組。其中包括一個名為的名稱，例如 `AWS-OpsWorks-RDP-Server`， Stacks AWS OpsWorks 會連接到所有 Windows 執行個體以允許 RDP 存取。不過，此安全群組預設沒有任何規則，因此您必須新增傳入規則來允許 RDP 存取您的執行個體。

允許 RDP 存取

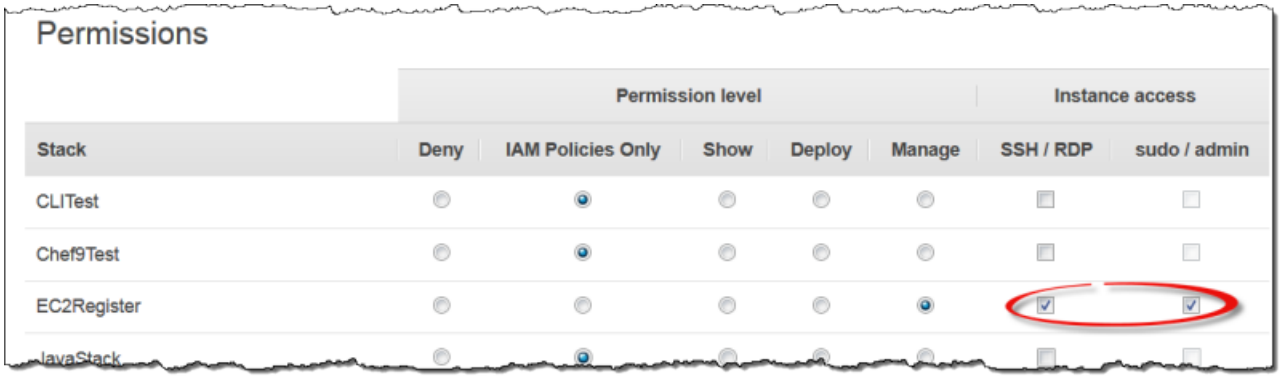
1. 開啟 Amazon EC2 主控台，將其設定為堆疊的區域，然後從導覽窗格中選擇安全群組。
2. 選擇 AWS-OpsWorks-RDP-Server，並選擇 Inbound (傳入) 標籤，然後選擇 Edit (編輯)。
3. 選擇 Add Rule (新增規則)，然後指定下列設定：
   - 類型 – RDP。
   - 來源 – 允許的來源 IP 地址。

     通常，您會允許來自您 IP 地址或指定 IP 地址範圍 (通常是公司的 IP 地址範圍) 的傳入 RDP 請求。針對學習用途，通常指定 0.0.0.0/0 已足夠，這會允許來自任何 IP 地址的 RDP 存取。

此安全群組允許執行個體接收 RDP 連線請求，但這只是此安全群組所能發揮的一半作用。一般使用者將使用 Stacks AWS OpsWorks 提供的密碼登入執行個體。若要讓 AWS OpsWorks Stacks 產生該密碼，您必須明確授權使用者的 RDP 存取。

為使用者授權 RDP

1. 在 AWS OpsWorks Stacks 儀表板中，選擇 IISWalkthrough 堆疊。
2. 在該堆疊的導覽窗格中選擇 Permissions (許可)。
3. 在許可頁面上，選擇 Edit (編輯)。
4. 在使用者清單中，針對您要授予必要許可的使用者，選取 SSH/RDP 的核取方塊。如果您希望使用者也具有管理員許可，請同時選取 sudo/admin。



5. 選擇儲存。

使用者隨即可以取得密碼，並用來登入執行個體，如下文所述。

> **Note**
>
> 您也可以使用管理員身分登入。如需詳細資訊，請參閱以管理員身分登入。

步驟 2.3：實作自訂技術指南

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

儘管堆疊基本上是執行個體的容器，但您無法直接將執行個體新增到堆疊。您將會新增一或多個 layer，每一個代表相關執行個體的群組，然後將執行個體新增至這些 layer。

layer 基本上是 Stacks AWS OpsWorks 用來建立一組具有相同組態的 Amazon EC2 執行個體的藍圖。執行個體從基礎版本的作業系統開始，且執行個體的 layer 會在執行個體上執行各種任務來實作該藍圖，這些任務可能包括：

- 建立目錄和檔案
- 管理使用者
- 安裝並設定軟體
- 啟動或停止伺服器
- 部署應用程式程式碼以及相關檔案。

layer 透過執行 Chef 配方在執行個體上執行任務，簡稱為 配方。配方是使用 Chef 網域專用語言 (DSL) 來描述執行個體最終狀態的 Ruby 應用程式。透過 AWS OpsWorks Stacks，每個配方通常會指派給 layer 的生命週期事件之一：設定、組態、部署、取消部署和關機。當執行個體上發生生命週期事件時， AWS OpsWorks Stacks 會執行事件的配方來執行適當的任務。例如，安裝事件會在執行個體完成開機後發生。 AWS OpsWorks 堆疊接著會執行安裝配方，通常會執行安裝和設定伺服器軟體以及啟動相關服務等任務。

AWS OpsWorks Stacks 為每個 layer 提供一組執行標準任務的內建配方。您可以實作自訂配方來執行額外任務，並將其指派到 layer 的生命週期事件，以擴展 layer 的功能。Windows 堆疊支援自訂 layer，這些 layer 具有一組僅執行幾項基本任務的最少量配方。若要將功能新增至您的 Windows 執行個體，您必須實作自訂配方來安裝軟體、部署應用程式等等。本主題說明如何建立簡單的自訂 layer 來支援 IIS 執行個體。

主題

- 技術指南和配方的快速簡介
- 實作配方來安裝和啟動 IIS
- 啟用自訂技術指南

技術指南和配方的快速簡介

配方定義執行個體預期狀態的一或多個 layer 面：應包含哪些目錄、應安裝哪些軟體套件、應部署哪些應用程式等等。配方封裝在「技術指南」中，而技術指南可以包含一或多個相關配方，再加上相關檔案 (例如建立組態檔案的範本)。

本主題為配方做了基本的介紹，足以為您示範如何實作技術指南以支援簡單的自訂 IIS layer。如需技術指南一般簡介的詳細資訊，請參閱 技術指南和配方。如需詳細的實作技術指南教學介紹，包括某些 Windows 特定主題，請參閱技術指南 101。

技術上而言，Chef 配方是 Ruby 應用程式，但大部分 (如非全部) 程式碼位於 Chef DSL 中。DSL 主要由一組「資源」組成，可用於宣告性指定執行個體狀態的某一 layer 面。例如，`directory` 資源會定義要新增至系統的目錄。下列範例定義一個具有完整控制權的 `C:\data` 目錄，此目錄屬於指定使用者，且不會從父目錄繼承權限。

```
directory 'C:\data' do
  rights :full_control, 'WORKGROUP\username'
  inherits false
  action :create
end
```

當 Chef 執行配方時，會透過將資料傳遞給相關聯的「提供者」(這是處理修改執行個體狀態詳細資訊的 Ruby 物件) 來執行每項資源。在此範例中，供應者會使用指定組態來建立新目錄。

自訂 IIS layer 的自訂技術指南必須執行下列任務：

- 安裝 IIS 功能，並開始服務。

  您通常會在完成啟動執行個體後的安裝期間執行此任務。
- 將應用程式部署到執行個體 (此範例中為一個簡單的 HTML 頁面)。

  您通常會在設定期間執行此任務。不過，應用程式通常需要定期更新，所以您也需要在執行個體處於上線狀態時部署更新。

您可以使用單一配方來執行所有這些任務。但是，首選方法是將單獨的配方用於設定和部署任務。這樣一來，您無需執行設定程式碼，也可隨時部署應用程式更新。下列說明如何設定技術指南來支援自訂 IIS layer。後續主題將說明如何實作配方。

開始使用

1. 在您的工作站中方便的位置上，建立名為 `iis-cookbook` 的目錄。
2. 將具有下列內容的 `metadata.rb` 檔案新增至 `iis-cookbook`。

   ```
   name "iis-cookbook"
   version "0.1.0"
   ```

此範例使用最少量的 metadata.rb。如需如何使用此檔案的詳細資訊，請參閱 metadata.rb。

3. 將 recipes 目錄新增至 iis-cookbook。

   此目錄 (必須命名為 recipes) 包含技術指南的配方。

一般而言，技術指南可以包含各種其他目錄。例如，如果配方使用範本來建立組態檔案，範本通常會位於 templates\default 目錄中。此範例的技術指南完全由配方所組成，所以不需要其他目錄。此外，此範例僅使用單一技術指南，但您可以使用所需數量的技術指南；針對複雜的專案，通常使用多個技術指南會更適當。例如，針對設定和部署任務，您可以使用個別的技術指南。如需更多技術指南範例，請參閱 技術指南和配方。

實作配方來安裝和啟動 IIS

IIS 是一項 Windows「功能」，是可安裝在 Windows Server 上的選用系統元件之一。您可以讓配方透過下列其中一種方法來安裝 IIS：

- 透過使用 powershell_script 資源，來執行 Install-WindowsFeature cmdlet。
- 透過使用 Chef Windows 技術指南的 windows_feature 資源。

  windows 技術指南包含一組資源，這些資源的供應者使用部署映像服務和管理 (DISM) 在 Windows 執行個體上執行各種任務，包括功能安裝。

> **Note**
>
> powershell_script 位於 Windows 配方的最有用資源之間。您可使用其透過執行適當的 PowerShell 指令碼或 cmdlet 在執行個體上執行各種任務。尤其適用於不受 Chef 資源支援的任務。

此範例使用 PowerShell 指令碼來安裝和啟動 Web Server (IIS)。下文將介紹 windows 技術指南。如需如何使用 windows_feature 來安裝 IIS 的範例，請參閱 安裝 Windows 功能：IIS

將下列內容和名為 install.rb 的配方新增至技術指南的 recipes 目錄。

```
powershell_script 'Install IIS' do
  code 'Install-WindowsFeature Web-Server'
  not_if "(Get-WindowsFeature -Name Web-Server).Installed"
```

```
end

service 'w3svc' do
  action [:start, :enable]
end
```

此配方包含兩種資源。

powershell_script

`powershell_script` 執行指定的 PowerShell 指令碼或 cmdlet。此範例具有下列屬性設定：

- `code` – 要執行的 PowerShell cmdlet。

  此範例執行 `Install-WindowsFeature` cmdlet，這會安裝 Web 伺服器 (IIS)。一般而言，`code` 屬性可以具有任意數目的行，因此您可以執行所需數量的 cmdlet。
- `not-if` – 防護屬性，可確保配方只有在尚未安裝時才安裝 IIS。

  您通常會希望配方是「等冪」，因此配方不會浪費時間重複執行相同的任務。

每個資源都有一個動作，指定提供者要採取的動作。在此範例中沒有明確的動作，因此供應者會採取預設 `:run` 動作，該動作會執行指定 PowerShell 指令碼。如需詳細資訊，請參閱執行 Windows PowerShell 指令碼。

服務

一個 `service` 管理一項服務，在此範例中為 Web Server IIS 服務 (W3SVC)。此範例使用預設屬性並指定 `:start` 與 `:enable` 這兩個動作來啟動和啟用 IIS。

> **Note**
>
> 如果您希望安裝使用套件安裝程式的軟體 (例如 MSI)，您可以使用 `windows_package` 資源。如需詳細資訊，請參閱安裝套件。

啟用自訂技術指南

AWS OpsWorks Stacks 會在每個執行個體上從本機快取執行配方。若要執行您的自訂配方，您必須執行下列作業：

- 將技術指南存放在遠端儲存庫中。

AWS OpsWorks Stacks 會將技術指南從此儲存庫下載到每個執行個體的本機快取。

- 編輯堆疊以啟用自訂技術指南。

  自訂技術指南預設為停用，所以您必須為堆疊啟用自訂技術指南，並提供儲存庫的 URL 和相關資訊。

AWS OpsWorks Stacks 支援自訂技術指南的 S3 封存和 Git 儲存庫；此範例使用 S3 封存。如需詳細資訊，請參閱技術指南儲存庫。

使用 S3 封存

1. 建立 `iis-cookbook` 目錄的 `.zip` 存檔。

   AWS OpsWorks Stacks 也支援 Windows 堆疊的 `.tgz`(gzip 壓縮 tar) 封存。
2. 將封存上傳至美國西部 （ 加利佛尼亞北部 ） 區域的 S3 儲存貯體，並將檔案設為公有。您也可以使用私有 S3 封存，但在這個範例中公有封存已足夠，且使用起來更簡單。

   a. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/s3/ 開啟 Amazon S3 主控台。
   b. 如果您在 中還沒有儲存貯體`us-west-1`，請選擇建立儲存貯體，並在美國西部 （ 加利佛尼亞北部 ） 區域中建立儲存貯體。
   c. 在儲存貯體清單中，選擇您要上傳檔案的儲存貯體名稱，然後選擇 Upload (上傳)。
   d. 選擇 Add Files (新增檔案)。
   e. 選取要上傳的存檔，然後選擇 Open (開啟)。
   f. 在 Upload - Select Files and Folders (上傳 - 選取檔案與資料夾) 對話方塊的底部，選擇 Set Details (設定詳細資訊)。
   g. 在 Set Details (設定詳細資訊) 對話方塊中，選擇 Set Permissions (設定許可)。
   h. 在 Set Permissions (設定許可) 對話方塊中，選擇 Make everything public (公開所有項目)。
   i. 在 Set Permissions (設定許可) 對話方塊中，選擇 Start Upload (開始上傳)。上傳完成時，`iis-cookbook.zip` 檔案會出現在您的儲存貯體中。
   j. 選擇儲存貯體，然後選擇儲存貯體的 Properties (屬性) 標籤。在 Link (連結) 旁，記錄封存檔案的 URL 以供日後使用。

   如需將檔案上傳至 Amazon S3 儲存貯體的詳細資訊，請參閱《Amazon S3 主控台使用者指南》中的如何將檔案和資料夾上傳至 S3 儲存貯體？。 Amazon S3

> **Important**
>
> 到目前為止，本演練只需您稍微花一點時間； AWS OpsWorks Stacks 服務本身是免費的。不過，您必須支付您使用的任何 AWS 資源，例如 Amazon S3 儲存體。一旦您上傳封存，即會開始產生費用。如需詳細資訊，請參閱 AWS 定價。

為堆疊啟用自訂技術指南。

1. 在 AWS OpsWorks Stacks 主控台中，選擇導覽窗格中的堆疊，然後選擇右上角的堆疊設定。
2. 在 Settings (設定) 頁面的右上角，選擇 Edit (編輯)。
3. 在 Settings (設定) 頁面上，將 Use custom Chef cookbooks (使用自訂 Chef 技術指南) 設定為 Yes (是) 然後輸入下列資訊：
   - 儲存庫類型 – S3 Archive。
   - 儲存庫 URL – 您先前記錄之技術指南封存檔案的 S3 URL。
4. 選擇 Save (儲存) 來更新堆疊組態。

AWS OpsWorks Stacks 會在所有新的執行個體上安裝您的自訂技術指南。請注意， AWS OpsWorks Stacks 不會自動在線上執行個體上安裝或更新自訂技術指南。您可以手動執行此作業，如下所述。

步驟 2.4：新增 IIS Layer

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您的技術指南具有一個安裝和啟動 IIS 的配方。這足以用來建立 layer，並確認您具有正在運作的 IIS 執行個體。之後，您會將應用程式部署功能新增到 layer。

建立 Layer

首先，將 layer 新增堆疊。然後，透過指派自訂配方至適當的生命週期事件，以將功能新增至該 layer。

將 IIS layer 新增至堆疊

1. 在導覽窗格中，選擇 Layers (Layer)，然後選擇 Add a layer (新增 layer)。
2. 如下所示來設定 layer：
   - 名稱 – **`IISExample`**
   - 簡短名稱 – **`iisexample`**

     AWS OpsWorks Stacks 使用簡短名稱在內部識別 layer。您也可以在配方中使用短名來識別 layer，但此範例不執行此動作。您可以指定任何短名，但只能包含小寫英數字元和少量標點符號。如需詳細資訊，請參閱自訂 Layer。
3. 選擇 Add Layer (新增 Layer)。

如果您此時將執行個體新增至 IISWalkthrough 並將其啟動， AWS OpsWorks Stacks 將自動安裝技術指南，但不會執行 `install.rb`。執行個體上線之後，您可以使用執行配方堆疊命令來手動執行配方。不過，更好的方法是將配方指派給 layer 的生命週期事件之一。 AWS OpsWorks Stacks 接著會在執行個體生命週期的適當時間點自動執行配方。

在執行個體啟動完成後立即安裝並啟動 IIS。若要這樣做，請將 `install.rb` 指派給 layer 的 `Setup` 事件。

將配方指派給生命週期事件

1. 在導覽窗格中選擇 Layers (Layer)
2. 在 IISExample layer 的方塊中，選擇 Recipes (配方)。
3. 在右上角，選擇 Edit (編輯)。
4. 在 Custom Chef Recipes (自訂 Chef 配方) 下方的 Setup (安裝) 配方方塊中，輸入 **`iis-cookbook::install`**。

   > **Note**
   >
   > 使用 *`cookbook-name`*`::`*`recipe-name`* 來識別配方，其中會省略配方名稱的 `.rb` 尾碼。
5. 選擇 + 將配方新增至 layer。紅色 x 會顯示在配方旁，以便稍後輕鬆移除。
6. 選擇 Save (儲存) 以儲存新組態。自訂設定配方現在應包含 `iis-cookbook::install`。

將執行個體新增至 Layer 並啟動

您可以透過將執行個體新增至 layer 並啟動執行個體來試用配方。當執行個體完成開機時， AWS OpsWorks Stacks 會自動安裝技術指南並在設定`install.rb`期間執行。

將執行個體新增至 layer 並啟動

1. 在 AWS OpsWorks 堆疊導覽窗格中，選擇執行個體。
2. 在 IISExample layer 下方，選擇 Add an instance (新增執行個體)。
3. 選取適當的大小。t2.micro (或可供您使用的最低大小) 對此範例應已足夠使用。
4. 選擇 Add Instance (新增執行個體)。根據預設， AWS OpsWorks Stacks 會透過將整數附加至 layer 的簡短名稱來產生執行個體名稱，因此執行個體應該命名為 iisexample1。
5. 在執行個體的動作欄中選擇開始以啟動執行個體。然後， AWS OpsWorks Stacks 將啟動 EC2 執行個體並執行安裝配方來設定執行個體。如果 layer 目前有任何部署配方， AWS OpsWorks Stacks 會在安裝配方完成後執行這些配方。

   此程序可能需費時數分鐘，期間 Status (狀態) 欄會顯示一系列的狀態。當您進入 online (線上) 狀態時，設定程序即完成，並且執行個體已可供使用。

確認 IIS 已安裝且正在執行

您可以使用 RDP 連線到執行個體並驗證您的設定配方是否運作正常。

驗證 IIS 已安裝且正在執行

1. 在導覽窗格中選擇執行個體，然後在 iisexample1 執行個體的動作欄中選擇 rdp。 AWS OpsWorks Stacks 會自動為您產生在指定時段後過期的 RDP 密碼。
2. 將 Session valid for (工作階段有效期) 設定為 2 個小時，然後選擇 Generate Password (產生密碼)。
3. AWS OpsWorks Stacks 會顯示密碼，而且為了方便起見，也會顯示執行個體的公有 DNS 名稱和使用者名稱。複製全部三項，然後按一下 Acknowledge and close (確認並關閉)。
4. 開啟您的 RDP 用戶端，並使用步驟 3 中的資料來連線到執行個體。
5. 在執行個體上，開啟 Windows 檔案總管並檢查 `C:` 磁碟機。它應有 `C:\inetpub` 目錄，其由 IIS 安裝所建立。
6. 開啟控制台 Administrative Tools (管理工具) 應用程式，然後開啟 Services (服務)。您應該會在清單底部附近看到 IIS 服務。該服務名為 World Wide Web Publishing Service (全球資訊網發佈服務)，且狀態應為 running (執行中)。

7. 返回 AWS OpsWorks Stacks 主控台並選擇 iisexample1 執行個體的公有 IP 地址。請務必在 AWS OpsWorks Stacks 中執行此操作，而不是在 Amazon EC2 主控台中執行此操作。這會將 HTTP 請求自動傳送到地址，且應會開啟預設的 IIS 歡迎頁面。

下一個主題討論如何部署應用程式到執行個體，以一個簡單的靜態 HTML 頁面為範例。不過，如果您想要休息，請在 iisexample1 執行個體的動作欄中選擇停止，以停止執行個體並避免產生不必要的費用。當您準備好繼續時，您可以重新啟動執行個體。

步驟 2.5：部署應用程式

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

IIS 安裝會為應用程式的程式碼和相關檔案建立 `C:\inetpub\wwwroot` 目錄。下一步是在該目錄中安裝應用程式。在此範例中，您將在 `default.html` 中安裝靜態 HTML 首頁 `C:\inetpub\wwwroot`。您可以輕鬆擴展一般方法以處理更複雜的案例，如 ASP.NET 應用程式。

您可以在技術指南中包含應用程式的檔案，並讓 `install.rb` 複製這些檔案到 `C:\inetpub\wwwroot`。如需如何執行此作業的範例，請參閱 範例 6：建立檔案。但是，此方法並不靈活且效率不高，通常更好的方法是將技術指南開發與應用程式開發分隔開來。

偏好的解決方案是實作個別的部署配方，從儲存庫擷取應用程式的程式碼和相關檔案，任何您偏好的儲存庫，而不只是技術指南儲存庫，並在每個 IIS 伺服器執行個體上安裝。此方法會將技術指南開發與應用程式開發分隔開來，當您需要更新應用程式時，只需再次執行部署配方即可，無需更新技術指南。

本主題說明如何實作將 `default.htm` 部署到 IIS 伺服器的簡單部署配方。您可以隨時將此範例擴展到更複雜的應用程式。

主題

- 建立應用程式並存放在儲存庫中。
- 實作配方來部署應用程式
- 更新執行個體的技術指南
- 將配方新增至自訂 IIS Layer

- 新增應用程式
- 部署應用程式和開啟應用程式

建立應用程式並存放在儲存庫中。

您可以為應用程式使用任何您偏好的儲存庫。為求簡化，此範例會將 default.htm 存放在公有 S3 儲存貯體中。

建立應用程式

1. 在您的工作站中方便的位置上，建立名為 iis-application 的目錄。
2. 將 default.htm 檔案新增至具有以下內容的 iis-application。

```
<!DOCTYPE html>
<html>
  <head>
    <title>IIS Example</title>
  </head>
  <body>
    <h1>Hello World!</h1>
  </body>
</html>
```

3. 建立 S3 儲存貯體，將 default.htm 上傳到儲存貯體，並記錄此 URL 供以後使用。為求簡化，請將檔案設為公有。

> **Note**
>
> 這是一個非常簡單的應用程式，但您可以擴展基本原則以處理生產層級的應用程式。
>
> - 針對具有多個檔案的更複雜應用程式，為 iis-application 建立 .zip 封存並將其上傳至您的 S3 儲存貯體通常會更簡單。
>
> 然後，您可以下載 .zip 檔並將其內容解壓縮到適當的目錄。不需要下載多個檔案或建立目錄結構等作業。
> - 針對生產應用程式，建議您將檔案保留為私有。如需如何讓配方從私有 S3 儲存貯體下載檔案的範例，請參閱 在 Stacks Windows 執行個體上使用適用於 Ruby AWS OpsWorks 的 開發套件。
> - 您可以將您的應用程式存放於任何適當的儲存庫。

通常，您會使用儲存庫的公有 API 來下載應用程式。此範例使用 Amazon S3 API。例如，如果您將應用程式存放在 GitHub 上，則可以使用 GitHub API。

實作配方來部署應用程式

將名為 deploy.rb 的配方新增至 iis-cookbook recipes 目錄中，其中包含下列內容。

```
chef_gem "aws-sdk-s3" do
  compile_time false
  action :install
end

ruby_block "download-object" do
  block do
    require 'aws-sdk-s3'

    #1
    # Aws.config[:ssl_ca_bundle] = 'C:\ProgramData\Git\bin\curl-ca-bundle.crt'
    Aws.use_bundled_cert!

    #2
    query = Chef::Search::Query.new
    app = query.search(:aws_opsworks_app, "type:other").first
    s3region = app[0][:environment][:S3REGION]
    s3bucket = app[0][:environment][:BUCKET]
    s3filename = app[0][:environment][:FILENAME]

    #3
    s3_client = Aws::S3::Client.new(region: s3region)
    s3_client.get_object(bucket: s3bucket,
                         key: s3filename,
                         response_target: 'C:\inetpub\wwwroot\default.htm')
  end
  action :run
end
```

此範例使用適用於 Ruby v2 的 SDK 下載 檔案。不過， AWS OpsWorks Stacks 不會在 Windows 執行個體上安裝此開發套件，因此配方會從處理該任務chef_gem的資源開始。

> **Note**
>
> chef_gem 資源會將 gem 安裝到 Chef 專用的 Ruby 版本 (配方使用的版本) 中。如果您要為全系統範圍的 Ruby 版本安裝 gem，請使用 gem_package 資源。

配方大部分是 ruby_block 資源，其會執行一個 Ruby 程式碼區塊，使用適用於 Ruby 的 SDK 下載 default.htm。ruby_block 中的程式碼可分為下列區段，這些區段分別對應於程式碼範例中的編號註解。

1：指定憑證套件

Amazon S3 使用 SSL，因此您需要適當的憑證才能從 S3 儲存貯體下載物件。適用於 Ruby v2 的 SDK 不包含憑證套件，因此您必須提供憑證套件，並設定適用於 Ruby 的 SDK 來使用它。AWS OpsWorks Stacks 不會直接安裝憑證套件，但會安裝 Git，其中包含憑證套件 (curl-ca-bundle.crt)。為了方便起見，此範例會將適用於 Ruby 的 SDK 設定為使用適用於 SSL 的 Git 憑證套件。您也可以安裝自己的套件，並依照需要來設定軟體開發套件。

2：擷取儲存庫資料

若要從 Amazon S3 下載物件，您需要 AWS 區域、儲存貯體名稱和金鑰名稱。如下文所述，此範例透過將一組環境變數與應用程式建立關聯來提供此資訊。當您部署應用程式時， AWS OpsWorks Stacks 會將一組屬性新增至執行個體的節點物件。這些屬性實際上是包含應用程式組態 (包括環境變數) 的雜湊表。此應用程式的應用程式屬性看起來會類似下列內容 (JSON 格式)。

```
{
  "app_id": "8f71a9b5-de7f-451c-8505-3f35086e5bb3",
  "app_source": {
      "password": null,
      "revision": null,
      "ssh_key": null,
      "type": "other",
      "url": null,
      "user": null
  },
  "attributes": {
      "auto_bundle_on_deploy": true,
      "aws_flow_ruby_settings": {},
      "document_root": null,
      "rails_env": null
  },
```

```
  "data_sources": [{"type": "None"}],
  "domains": ["iis_example_app"],
  "enable_ssl": false,
  "environment": {
      "S3REGION": "us-west-2",
      "BUCKET": "windows-example-app",
      "FILENAME": "default.htm"
  },
  "name": "IIS-Example-App",
  "shortname": "iis_example_app",
  "ssl_configuration": {
      "certificate": null,
      "private_key": null,
      "chain": null
  },
  "type": "other",
  "deploy": true
}
```

此應用程式的環境變數存放在 `[:environment]` 屬性中。若要擷取，請使用 Chef 搜尋查詢來擷取應用程式的雜湊表，位於 `aws_opsworks_app` 節點之下。此應用程式定義為 `other` 類型，因此查詢會搜尋該類型的應用程式。配方會利用此執行個體上只有一個應用程式的事實，因此特定的雜湊表為 `app[0]`。為了方便起見，配方隨後會將區域、儲存貯體和檔案名稱指派至變數。

如需如何使用 Chef 搜尋的詳細資訊，請參閱 使用 Chef 搜尋取得屬性值。

3：下載檔案

配方的第三個部分是建立 S3 用戶端物件 並使用其 `get_object` 方法來下載 `default.htm` 到執行個體的 `C:\inetpub\wwwroot` 目錄。

> **Note**
>
> 配方是 Ruby 應用程式，因此 Ruby 程式碼不一定需要位於 `ruby_block` 中。不過，配方主體中的程式碼會先執行，接著才會執行資源。在此範例中，如果您將下載程式碼放在配方內文中，它會失敗，因為`chef_gem`資源尚未安裝適用於 Ruby 的 SDK。`ruby_block` 資源中的程式碼會在資源執行時，在`chef_gem`資源安裝適用於 Ruby 的 SDK 之後執行。

更新執行個體的技術指南

AWS OpsWorks Stacks 會自動在新執行個體上安裝自訂技術指南。但是因為您正在使用現有的執行個體，所以您必須手動更新技術指南。

更新執行個體的技術指南

1. 建立 `iis-cookbook` 的 `.zip` 存檔，並將其上傳至 S3 儲存貯體。

   這會覆寫現有的技術指南，但 URL 會保持不變，因此您不需要更新堆疊組態。
2. 如果您的執行個體並未處於線上狀態，請將其重新啟動。
3. 在執行個體上線之後，在導覽窗格中選擇 Stack (堆疊)，然後選擇 Run Command (執行命令)。
4. 針對 Command (命令)，選擇 Update Custom Cookbooks (更新自訂技術指南)。此命令會在執行個體上安裝更新技術指南。
5. 選擇 Update Custom Cookbooks (更新自訂技術指南)。此命令可能需要幾分鐘的時間來完成。

將配方新增至自訂 IIS Layer

如同 `install.rb` 一樣，處理部署的較佳方式是將 `deploy.rb` 指派給適當的生命週期事件。您通常會將部署配方指派至部署事件，並統稱為部署配方。將配方指派至部署事件不會觸發此事件。反之：

- 對於新的執行個體， AWS OpsWorks Stacks 會在設定配方完成後自動執行部署配方，因此新的執行個體會自動擁有目前的應用程式版本。
- 針對線上執行個體，請使用部署命令來手動安裝新的或更新應用程式。

  此命令會在堆疊的執行個體上觸發部署事件，該事件會執行部署配方。

將 deploy.rb 指派給 layer 的部署事件

1. 在導覽窗格中選擇 Layers (Layer)，然後在 Layer IISExample 下方選擇 Recipes (配方)。
2. 在 Custom Chef Recipes (自訂 Chef 配方) 下，將 **`iis-cookbook::deploy`** 新增至 Deploy (部署) 配方方塊，並選擇 + 來將配方新增至 layer。
3. 選擇 Save (儲存) 以儲存新組態。自訂部署配方現在應包含 `iis-cookbook::deploy`。

新增應用程式

最後一個任務是將應用程式新增至堆疊，以在 Stacks AWS OpsWorks 環境中代表您的應用程式。應用程式包含中繼資料 (如應用程式的顯示名稱) 和從儲存庫下載的應用程式所需資料。

將應用程式新增至堆疊

1. 在導覽窗格中選擇 Apps (應用程式)，然後選擇 Add an app (新增應用程式)。
2. 使用以下設定來設定應用程式。
   - 名稱 – **IIS-Example-App**
   - 儲存庫類型 – 其他
   - 環境變數 – 新增下列三個環境變數：
     - **S3REGION** – 儲存貯體的區域 （ 在此情況下為 `us-west-1`)。
     - **BUCKET** – 儲存貯體名稱，例如 `windows-example-app`。
     - **FILENAME** – 檔案名稱：**default.htm**。
3. 接受其餘設定的預設值，然後選擇 Add App (新增應用程式) 將應用程式新增至堆疊。

> **Note**
>
> 此範例使用環境變數來提供下載資料。另一種方法是使用 S3 封存儲存庫類型，並提供檔案的 URL。 AWS OpsWorks Stacks 會將資訊以及選用的資料，例如您的 AWS 登入資料，新增至應用程式的 `app_source` 屬性。您的部署配方必須從應用程式屬性中取得 URL 並進行剖析，以擷取區域、儲存貯體名稱和檔案名稱。

部署應用程式和開啟應用程式

AWS OpsWorks Stacks 會自動將應用程式部署到新執行個體，但不會部署到線上執行個體。由於您的執行個體已在執行中，因此您必須手動部署應用程式。

部署應用程式

1. 在導覽窗格中選擇 Apps (應用程式)，然後在應用程式的 Actions (動作) 欄中選擇 deploy (部署)。
2. Command (命令) 應該設為 Deploy (部署)。選擇 Deploy App (部署應用程式) 頁面右下方的 Deploy (部署)。此命令可能需要幾分鐘的時間來完成。

在部署完成後，您可以返回 Apps (應用程式) 頁面。Status (狀態) 指標會顯示綠色的 successful (成功)，且應用程式名稱旁會顯示綠色核取標記，指出部署成功。

> **Note**
>
> Windows 應用程式一律為 Other (其他) 應用程式類型，因此部署應用程式會執行下列作業：
>
> - 將應用程式的資料新增至堆疊組態和部署屬性，如前所述。
> - 在堆疊的執行個體上觸發部署事件，該事件會執行您的自訂部署配方。

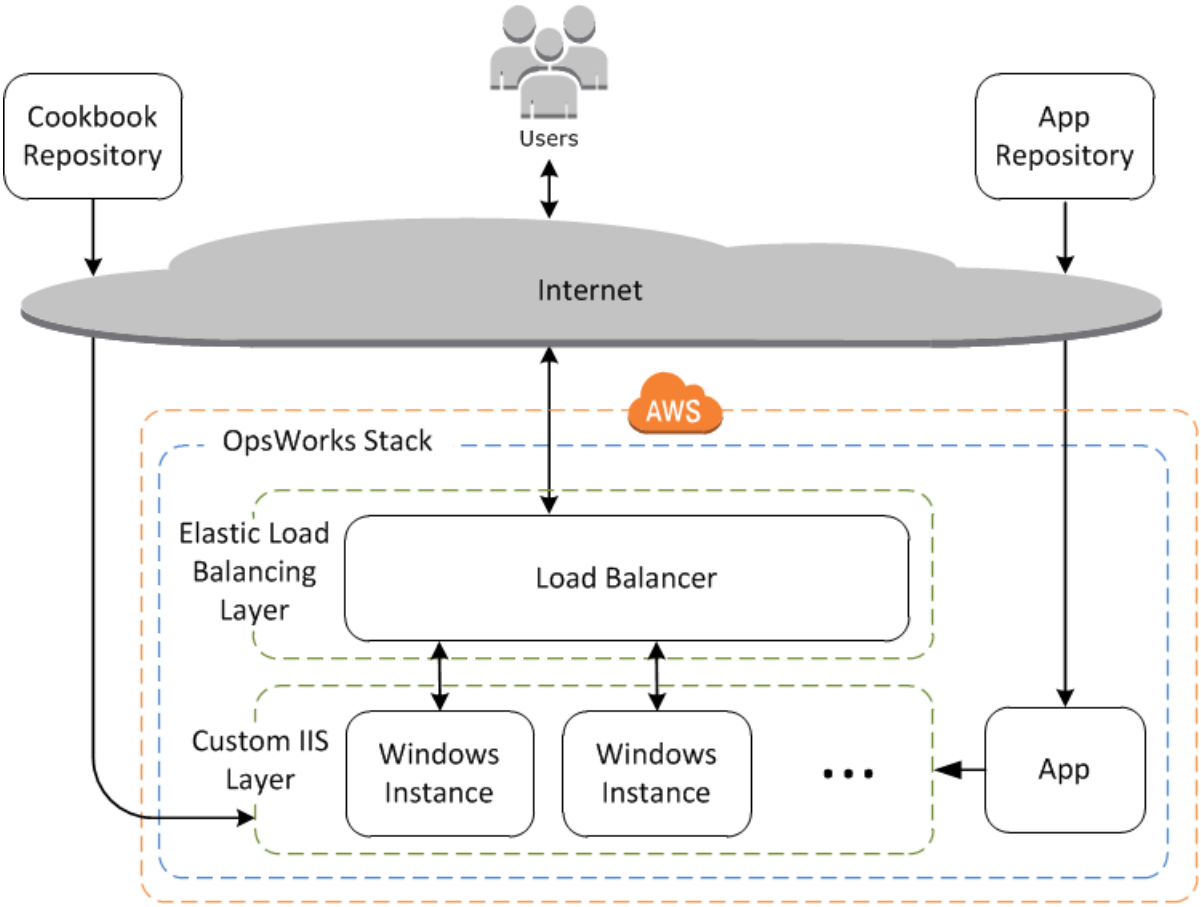
> **Note**
>
> 如需如何故障診斷故障之部署或應用程式的詳細資訊，請參閱除錯配方。

應用程式現在已安裝。您可以在導覽窗格中選擇執行個體，然後選擇執行個體的公有 IP 地址來開啟它。這會將 HTTP 請求傳送到執行個體，您應該會在瀏覽器中看到類似以下的內容。

步驟 3：橫向擴展 IISExample

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

如果您的傳入使用者請求數量開始接近您可透過單一 t2.micro 執行個體處理之請求限制，則您需要增加伺服器容量。您可以移至較大的執行個體，但具有限制。更靈活的方法是將執行個體新增到您的堆疊，然後將其放置在負載平衡器後方。此基本架構看起來類似下列內容。



除了其他優勢以外，此方法比單一大型執行個體更穩健。

- 如果其中某個執行個體故障，負載平衡器會將傳入請求分配到其餘的執行個體，且您的應用程式可繼續運作。
- 如果您將執行個體放置在不同的可用區域 (建議的實務)，即使某個可用區域遇到問題，您的應用程式仍將繼續運作。

AWS OpsWorks Stacks 可讓您輕鬆地向外擴展堆疊。本節說明如何將第二個全年無休 PHP App Server 執行個體新增至 IISExample，並將這兩個執行個體放在 Elastic Load Balancing 負載平衡器後方，以擴展堆疊的基本概念。您可以輕鬆擴展程序來新增 24 小時全年無休的任意執行個體，也可以使用時間型執行個體讓 AWS OpsWorks Stacks 自動擴展您的堆疊。如需詳細資訊，請參閱使用時間型和負載型執行個體管理負載。

新增負載平衡器

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Elastic Load Balancing 是一種 AWS 服務，可在多個 Amazon EC2 執行個體之間自動分配傳入的應用程式流量。負載平衡器有兩種用途。其中一個顯而易見的用途，是使應用程式伺服器上的負載達到均衡。許多網站都偏好隔離其應用程式伺服器和資料庫，讓使用者無法直接存取。除了分佈流量之外，Elastic Load Balancing 還會執行下列動作：

- 偵測運作狀態不佳的 Amazon EC2 執行個體。

  它會將流量重新路由至狀況良好的執行個體，直到狀況不良的執行個體恢復為止。
- 自動擴展處理容量的請求，以回應傳入的流量。

> **Note**
>
> AWS OpsWorks Stacks 不支援 Application Load Balancer。您只能搭配 AWS OpsWorks Stacks 使用 Classic Load Balancer。

雖然 Elastic Load Balancing 通常稱為 layer，但其運作方式與其他內建 layer 略有不同。您可以使用 Amazon EC2 主控台建立 Elastic Load Balancing 負載平衡器，然後將其連接到其中一個現有 layer，通常是應用程式伺服器 layer。 AWS OpsWorks Stacks 接著會向 服務註冊 layer 的現有執行個體，並自動新增任何新執行個體。下列程序說明如何新增負載平衡器。

將負載平衡器連接至自訂 IIS layer

1. 使用 Amazon EC2 主控台為 IISExample 建立新的負載平衡器。如需詳細資訊，請參閱 Elastic Load Balancing 入門。當您執行 Create Load Balancer (建立負載平衡器) 精靈時，請設定負載平衡器，如下所示：

1: Define Load Balancer (定義負載平衡器)

指派可輕鬆辨識的名稱給負載平衡器，例如 IIS-LB，讓您更輕鬆地在 Stacks AWS OpsWorks 主控台中找到 。接受其餘設定的預設值，然後選擇 Next: Assign Security Groups (下一步：指派安全群組)。

2: Assign Security Groups (指派安全群組)

如果您的帳戶支援預設 VPC，精靈會顯示此頁面以決定負載平衡器的安全群組。但不會為 EC2 Classic 顯示此頁面。

在本演練中，指定 default VPC security group (預設 VPC 安全群組)，然後選擇 Next: Configure Security Settings (下一步：設定安全設定)。

3: Configure Security Settings (設定安全設定)

本演練需要您的負載平衡器使用安全接聽程式 (也就是在其前端連線使用 HTTPS 或 SSL)，因此請選擇 Next: Configure Health Check (下一步：設定運作狀態檢查) 以繼續。

4: Configure Health Check (設定運作狀態檢查)

將 ping 路徑設定為 **/**。接受其餘設定的預設值，然後選擇 Next: Add EC2 Instances (下一步：新增 EC2 執行個體)。

5: Add EC2 Instances (新增 EC2 執行個體)

AWS OpsWorks Stacks 會自動負責向負載平衡器註冊執行個體。選擇 Next: Add Tags (下一步：新增標籤) 以繼續。

6: Add Tags (新增標籤)

在此範例中，您不需要使用標籤。選擇 Review and Create (檢閱和建立)。

7: Review (檢閱)

檢閱您的選擇，並選擇 Create (建立)，然後選擇 Close (關閉)，這會啟動負載平衡器。

2. 如果您的帳戶支援預設 VPC，則在您啟動負載平衡器之後，必須確保其安全群組具有適當的傳入規則。預設規則不接受任何傳入流量。

   1. 在 Amazon EC2 導覽窗格中選擇安全群組。
   2. 選擇 default VPC security group (預設 VPC 安全群組)。
   3. 在 Inbound (傳入) 標籤上，選擇 Edit (編輯)。

4. 在本演練中，將 Source (來源) 設定為 Anywhere (隨處)，以指示負載平衡器接受來自任何 IP 地址的傳入流量。
5. 按一下 Save (儲存)。

3. 返回 AWS OpsWorks Stacks 主控台。在 Layers (Layer) 頁面上，選擇 Network (網路)。
4. 在 Elastic Load Balancing 下，選取您在步驟 1 建立的 IIS-LB 負載平衡器，然後按一下 Save (儲存)。

   將負載平衡器連接到 layer 之後， AWS OpsWorks Stacks 會自動註冊 layer 的目前執行個體，並在新執行個體上線時新增執行個體。
5. 在 Layers (Layer) 頁面上，按一下負載平衡器的名稱，以開啟其詳細資訊頁面。負載平衡器頁面上的執行個體旁之綠色核取標記指出該執行個體已通過運作狀態檢查。

您現在可以將請求傳送到負載平衡器來執行 IIS-Example-App。

透過負載平衡器執行 IIS-Example-App

1. 選擇 Layers (Layer)。IIS-ELB 負載平衡器應做為 layer 列出，且運作狀態欄應有一個綠色的執行個體，指出該執行個體運作狀態良好。
2. 選擇負載平衡器的 DNS 名稱來執行 IIS-Example-App。應會列在負載平衡器的名稱下方，看起來類似於 `IIS-LB-1802910859.us-west-2.elb.amazonaws.com`。負載平衡器會將請求轉遞給執行個體，並傳回回應，而此回應看起來應該與您按一下執行個體的公有 IP 地址時所取得的回應完全相同。

此時您只有一個執行個體，所以負載平衡器實際上並未新增更多執行個體。不過，您現在可以將額外的執行個體新增至 layer。

將執行個體新增至 layer

1. 選擇 Instances (執行個體)，然後選擇 + instance (+ 執行個體) 將其他執行個體新增至 layer。
2. 啟動實例。

由於它們是新的執行個體， AWS OpsWorks Stacks 會自動安裝目前的自訂技術指南，並在設定期間部署目前的應用程式版本。當執行個體上線時， AWS OpsWorks Stacks 會自動將其新增至負載平衡器，因此您的執行個體會立即開始處理請求。若要驗證應用程式是否仍在運作，您可以選擇負載平衡器的 DNS 名稱。

後續步驟

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本演練帶領您完成設定簡易 Windows 應用程式伺服器堆疊的基本作業。以下是有關後續作業的一些建議。

- 如果您想要進一步了解， 入門：技術指南會提供實作技術指南的教學簡介，並包含許多 AWS OpsWorks Stacks 特定的範例。
- 您可以將 Amazon Relational Database Service (Amazon RDS) 層新增至堆疊，以用作後端資料庫伺服器。如需如何將應用程式連線至資料庫的資訊，請參閱 使用自訂配方。

Stacks AWS OpsWorks 中的技術指南入門

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

生產層級 AWS OpsWorks 的 Stacks 堆疊通常需要一些自訂，這通常表示實作自訂的 Chef 技術指南。「技術指南」為包含組態資訊 (其中包含稱為「配方」的指示) 的套件檔案。「配方」為一組一或多個以 Ruby 語言語法撰寫的指示，指定要使用的資源以及套用那些資源的順序。Chef 中所使用的「資源」為組態政策的陳述式。本演練提供實作 Stacks Chef AWS OpsWorks 技術指南的基本簡介。若要進一步了解 Chef、技術指南、配方及資源，請參閱後續步驟中的連結。

本演練主要說明如何建立您自己的技術指南。您也可以使用社群提供的技術指南，這可在 Chef Supermarket 這類網站上取得。為協助您開始使用社群技術指南，後續的演練將包含使用來自 Chef Supermarket 的社群技術指南說明。

在您開始本演練前，請完成幾個設定步驟。若您已完成本章中的其他演練 (例如入門：範例)，您即完成本演練的事前準備，可直接跳到開始本演練。否則，請務必完成事前準備，然後再返回本演練。

主題

- 步驟 1：建立技術指南
- 步驟 2：建立堆疊及其元件
- 步驟 3：執行及測試配方
- 步驟 4：更新技術指南以安裝套件
- 步驟 5：更新執行個體上的技術指南及執行配方
- 步驟 6：更新技術指南以新增使用者
- 步驟 7：更新技術指南以建立目錄
- 步驟 8：更新技術指南以建立及複製檔案
- 步驟 9：更新技術指南以執行命令
- 步驟 10：更新技術指南以執行指令碼
- 步驟 11：更新技術指南以管理服務
- 步驟 12：更新技術指南以使用自訂 JSON
- 步驟 13：更新技術指南以使用資料包
- 步驟 14：更新技術指南以使用反覆運算
- 步驟 15：更新技術指南以使用條件式邏輯
- 步驟 16：更新技術指南以使用社群技術指南
- 步驟 17：(選用) 清除
- 後續步驟

步驟 1：建立技術指南

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

從建立技術指南開始。此技術指南對開始作用不大，但它會作為本演練其餘部分的基礎。

> **Note**
>
> 此步驟示範如何手動建立技術指南。您可以透過使用 Chef 開發套件 (Chef DK) 在您的本機工作站上執行 chef generate cookbook 命令，來更快速的建立技術指南。但是，此命令會建立數個您在本演練中不需要的資料夾及檔案。

建立技術指南

1. 在您的本機工作站上，建立名為 opsworks_cookbook_demo 的目錄。您可以使用不同的名稱，但請務必在本演練的後續部分一律將 opsworks_cookbook_demo 取代為您選擇的名稱。
2. 在 opsworks_cookbook_demo 目錄中，使用文字編輯器建立一個名為 metadata.rb 的檔案。新增以下程式碼來指定技術指南的名稱。如需 metadata.rb 的詳細資訊，請參閱 Chef 網站上的 metadata.rb。

   ```
   name "opsworks_cookbook_demo"
   ```

3. 在 opsworks_cookbook_demo 目錄中，建立名為 recipes 的子目錄。此子目錄包含所有您為本演練的技術指南建立的配方。
4. 在 recipes 目錄中，建立名為 default.rb 的檔案。此檔案包含具有與檔案名稱相同的配方，但不帶有副檔名：default。將下列單行程式碼新增至 default.rb 檔案。此程式碼為一個單行配方，會在配方執行時於日誌中顯示簡易訊息：

   ```
   Chef::Log.info("********** Hello, World! **********")
   ```

5. 在終端機或命令提示中，使用 tar 命令建立名為 opsworks_cookbook_demo.tar.gz 的檔案，其中包含 opsworks_cookbook_demo 目錄及其內容。例如：

   ```
   tar -czvf opsworks_cookbook_demo.tar.gz opsworks_cookbook_demo/
   ```

   您可以使用不同的檔案名稱，但請務必在本演練的後續部分一律將 opsworks_cookbook_demo.tar.gz 取代為您選擇的名稱。

> **Note**
>
> 當您在 Windows 上建立 `tar` 檔案時，最上層目錄必須為技術指南的父系目錄。此演練已在 Linux 上使用 `tar`套件提供的tar命令進行測試，並在 Windows 上使用 Git Bash 提供的tar命令進行測試。使用其他命令或程式建立壓縮 TAR (.tar.gz) 檔案可能無法正常運作。

6. 建立 S3 儲存貯體，或使用現有的儲存貯體。如需詳細資訊，請參閱建立儲存貯體。
7. 將 `opsworks_cookbook_demo.tar.gz` 檔案上傳至 S3 儲存貯體。如需詳細資訊，請參閱將物件新增至儲存貯體。

您現在已有您可以在本演練中使用的技術指南。

在下一個步驟中，您會建立 Stacks AWS OpsWorks 堆疊，以供稍後用來上傳技術指南和執行技術指南的配方。

步驟 2：建立堆疊及其元件

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

建立 AWS OpsWorks Stacks 堆疊及其元件，其中包含 layer 和執行個體。在後續步驟中，您會將您的技術指南上傳至執行個體，並在該執行個體上執行技術指南的配方。

建立堆疊

1. 登入 Stacks AWS OpsWorks 主控台，網址為 https：//https://console.aws.amazon.com/opsworks。
2. 執行下列其中一個作業 (若適用的話)：
   - 如果顯示歡迎使用 AWS OpsWorks Stacks 頁面，請選擇新增您的第一個堆疊或新增您的第一個 Stacks AWS OpsWorks 堆疊 （這兩個選項都執行相同的動作）。即會顯示 Add stack (新增堆疊) 頁面。

- 若顯示 OpsWorks Dashboard (OpsWorks 儀表板) 頁面，請選擇 Add stack (新增堆疊)。即會顯示 Add Stack (新增堆疊) 頁面。

3. 選擇 Chef 12 stack (Chef 12 堆疊)。
4. 在 Stack name (堆疊名稱) 方塊中，輸入堆疊的名稱 (例如 **MyCookbooksDemoStack**)。您可以輸入不同的名稱，但請務必在本演練的後續部分一律將 MyCookbooksDemoStack 取代為您選擇的名稱。
5. 針對區域，選擇美國西部 （ 奧勒岡 ）。
6. 針對 VPC，執行下列其中一項作業：

   - 若 VPC 可用，請選擇它。如需詳細資訊，請參閱在 VPC 中執行堆疊。
   - 否則，請選擇 No VPC (無 VPC)。

7. 針對 Use custom Chef cookbooks (使用自訂 Chef 技術指南)，選擇 Yes (是)。
8. 針對 Repository type (儲存庫類型)，選擇 S3 Archive (S3 封存)。

   > **Note**
   >
   > 在入門：Linux 演練中，您選擇了 Http Archive (Http 封存)。請務必在此改為選擇 S3 Archive (S3 封存)。

9. 針對 Repository URL (儲存庫 URL)，輸入指向您 S3 中 opsworks_cookbook_demo.tar.gz 檔案的路徑。若要取得路徑，請在 S3 主控台中，選取 opsworks_cookbook_demo.tar.gz 檔案。在 Properties (屬性) 窗格中，複製 Link (連結) 欄位的值。(其內容大致如下：https://s3.amazonaws.com/amzn-s3-demo-bucket/opsworks_cookbook_demo.tar.gz。)
10. 如果您的 S3 儲存貯體是私有的，這是預設的，則對於存取金鑰 ID 和私密存取金鑰，輸入您在此演練中使用的 IAM 使用者的存取金鑰 ID 和私密存取金鑰。如需詳細資訊，請參閱編輯物件許可及與其他人分享物件。
11. 保留下列項目的預設值：

    - Default Availability Zone (預設可用區域) (us-west-2a)
    - 預設作業系統 (Linux 和 Amazon Linux 2016.09)
    - Default SSH key (預設 SSH 金鑰) (Do not use a default SSH key (不使用預設 SSH 金鑰))
    - Stack color (堆疊色彩) (深藍色)

12. 選擇 Advanced (進階)。
13. 針對 IAM role (IAM 角色)，執行以下其中一項作業：

- 若 aws-opsworks-service-role 可用，請選擇它。
- 若 aws-opsworks-service-role 不可用，請選擇 New IAM role (新 IAM 角色)。

14. 針對預設 IAM 執行個體描述檔，執行下列其中一項：

    - 若 aws-opsworks-ec2-role 可用，請選擇它。
    - 如果無法使用 aws-opsworks-ec2-role，請選擇新的 IAM 執行個體描述檔。

15. 保留下列項目的預設值：

    - Default root device type (預設根設備類型) (EBS backed (EBS 後端))
    - Hostname theme (主機名稱主題) (Layer Dependent (依存於 Layer))
    - OpsWorks Agent version (OpsWorks 代理程式版本) (最新版本)
    - Custom Chef JSON (自訂 Chef JSON) (空白)
    - Security (安全)、Use OpsWorks security groups (使用 OpsWorks 安全群組) (Yes (是))

16. 選擇新增堆疊。 AWS OpsWorks 堆疊會建立堆疊，並顯示 MyCookbooksDemoStack 頁面。

建立 layer

1. 在服務導覽窗格中，選擇 Layers (Layer)。即會顯示 Layers (Layer) 頁面。
2. 選擇 Add a layer (新增 layer)。
3. 在 OpsWorks 標籤上，針對 Name (名稱)，輸入 **`MyCookbooksDemoLayer`**。您可以輸入不同的名稱，但請務必在本演練的後續部分一律將 `MyCookbooksDemoLayer` 取代為您選擇的名稱。
4. 針對 Short name (簡短名稱)，輸入 **`cookbooks-demo`**。您可以輸入不同的名稱，但請務必在本演練的後續部分一律將 `cookbooks-demo` 取代為您選擇的名稱。
5. 選擇新增 layer. AWS OpsWorks Stacks 新增 layer 並顯示 Layers 頁面。

建立及啟動執行個體

1. 在服務導覽窗格中，選擇 Instances (執行個體)。即會顯示 Instances (執行個體) 頁面。
2. 選擇 新增執行個體。
3. 在 New (新增) 標籤上，選擇 Advanced (進階)。
4. 保留下列項目的預設值：

    - Hostname (主機名稱) (cookbooks-demo1)

- Size (大小) (c3.large)
- Subnet (子網路) (*IP ##* us-west-2a)
- Scaling type (擴展類型) (24/7 (全年無休))
- SSH key (SSH 金鑰) (Do not use a default SSH key (不使用預設 SSH 金鑰))
- 作業系統 (Amazon Linux 2016.09)
- OpsWorks Agent version (OpsWorks 代理程式版本) (Inherit from stack (繼承自堆疊))
- Tenancy (租用) (Default - Rely on VPC settings (預設 – 依存 VPC 設定))
- Root device type (根設備類型) (EBS backed (EBS 後端))
- Volume type (磁碟區類型) (General Purpose (SSD) (一般用途 (SSD)))
- Volume size (磁碟區大小) (8)

5. 選擇 Add instance (新增執行個體)。
6. 針對 MyCookbooksDemoLayer 之 cookbooks-demo1 的 Actions (動作)，選擇 start (啟動)。在 Status (狀態) 變更為 online (線上) 前請不要繼續。此程序可能需要花費數分鐘，敬請耐心等待。

您現在已擁有一個堆疊、一個 layer，及一個執行個體，技術指南已自動從您的 S3 儲存貯體複製到其中。在下一個步驟中，您會執行及測試執行個體上技術指南中的預設配方。

步驟 3：執行及測試配方

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

從 Stacks AWS OpsWorks 複製到執行個體的技術指南中執行並測試`default`配方。您可能還記得，此為在配方執行時於日誌中顯示簡易訊息的單行配方。

執行配方

1. 在服務導覽窗格中，選擇 Stack (堆疊)。即會顯示 MyCookbooksDemoStack 頁面。
2. 選擇 Run Command (執行命令)。即會顯示 Run Command (執行命令) 頁面。
3. 針對 Command (命令)，選擇 Execute Recipes (執行配方)。
4. 針對 Recipes to execute (要執行的配方)，輸入 **`opsworks_cookbook_demo::default`**。

opsworks_cookbook_demo 為在 `metadata.rb` 檔案中定義的技術指南名稱。**default** 為要執行的配方名稱，即技術指南的 `recipes` 子目錄中 `default.rb` 檔案的名稱，不帶副檔名。

5. 保留下列預設值：
   - Comment (註解) (空白)
   - Advanced (進階)、Custom Chef JSON (自訂 Chef JSON) (空白)
   - Instances (執行個體) (已核取 Select all (全選)，已核取 MyCookbooksDemoLayer，已核取 cookbooks-demo1)
6. 選擇 Execute Recipes (執行配方)。即會顯示 Running command execute_recipes (執行 execute_recipes 命令) 頁面。在 Status (狀態) 變更為 successful (成功) 前請不要繼續。此程序可能需要花費幾分鐘，敬請耐心等待。

檢查配方的結果

1. 在 Running command execute_recipes (執行 execute_recipes 命令) 頁面顯示時，針對 cookbooks-demo1 的 Log (日誌)，選擇 show (顯示)。即會顯示 execute_recipes 日誌頁面。
2. 向下捲動日誌，尋找與下列內容相似的項目：

```
[2015-11-13T19:14:39+00:00] INFO: ********** Hello, World! **********
```

您已成功執行您的第一個配方！在下一個步驟中，您會透過新增在執行個體上安裝套件的配方，來更新您的技術指南。

步驟 4：更新技術指南以安裝套件

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

透過新增在執行個體上安裝包含熱門文字編輯器 GNU Emacs 套件的配方，來更新您的技術指南。

雖然您可以輕鬆登入執行個體並安裝套件一次，但撰寫配方可讓您從 AWS OpsWorks Stacks 執行配方一次，同時在堆疊中的多個執行個體上安裝多個套件。

更新技術指南以安裝套件

1. 回到您的本機工作站，在 opsworks_cookbook_demo 目錄中的 recipes 子目錄中，建立名為 install_package.rb 的檔案，其中包含下列程式碼：

```
package "Install Emacs" do
  package_name "emacs"
end
```

此配方會在執行個體上安裝 emacs 套件。(如需詳細資訊，請前往 package。)

> **Note**
>
> 您可以給予配方任何您希望的檔案名稱。每當您想要 Stacks AWS OpsWorks 執行配方時，請務必指定正確的配方名稱。

2. 在終端機或命令提示中，使用 tar 命令建立 opsworks_cookbook_demo.tar.gz 檔案的新版本，其中包含 opsworks_cookbook_demo 目錄及其更新後的內容。
3. 將更新後的 opsworks_cookbook_demo.tar.gz 檔案上傳至 S3 儲存貯體。

這個新配方會在您每次更新執行個體上的技術指南，並且從更新後的技術指南中執行新的配方時執行。下一個步驟說明如何執行此作業。

在您完成下一個步驟之後，您將能登入執行個體，並從命令提示輸入 emacs 以啟動 GNU Emacs。(如需詳細資訊，請參閱連線至您的 Linux 執行個體。) 若要離開 GNU Emacs，請按下 Ctrl+X、Ctrl+C。

> **Important**
>
> 若要登入執行個體，您必須先向 AWS OpsWorks Stacks 提供公有 SSH 金鑰（您可以使用 ssh-keygen 或 PuTTYgen 等工具建立）的相關資訊，然後您必須在MyCookbooksDemoStack堆疊上設定許可，讓使用者能夠登入執行個體。如需說明，請參閱 註冊使用者的公有 SSH 金鑰 與 使用 SSH 登入。

步驟 5：更新執行個體上的技術指南及執行配方

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

更新執行個體上的技術指南，以及從執行個體上已更新的技術指南執行配方。在本演練的後續部分中，每次您新增配方藉以更新技術指南時，都會重複此一步驟。

更新執行個體上的技術指南

1. 在服務導覽窗格中，選擇 Stack (堆疊)。即會顯示 MyCookbooksDemoStack 頁面。
2. 選擇 Run Command (執行命令)。即會顯示 Run Command (執行命令) 頁面。
3. 針對 Command (命令)，選擇 Update Custom Cookbooks (更新自訂技術指南)。
4. 保留下列預設值：
   - Comment (註解) (空白)
   - Advanced (進階)、Custom Chef JSON (自訂 Chef JSON) (空白)
   - Advanced (進階)、Instances (執行個體) (已核取 Select all (全選)，已核取 MyCookbooksDemoLayer，已核取 cookbooks-demo1)。
5. 選擇 Update Custom Cookbooks (更新自訂技術指南)。即會顯示 Running command update_custom_cookbooks (執行 update_custom_cookbooks 命令) 頁面。在 Status (狀態) 變更為 successful (成功) 前請不要繼續。此程序可能需要花費數分鐘，敬請耐心等待。

執行配方

1. 在服務導覽窗格中，選擇 Stack (堆疊)。即會顯示 MyCookbooksDemoStack 頁面。
2. 選擇 Run Command (執行命令)。即會顯示 Run Command (執行命令) 頁面。
3. 針對 Command (命令)，選擇 Execute Recipes (執行配方)。
4. 針對 Recipes to execute (要執行的配方)，輸入要執行的配方名稱。您第一次執行此作業時，配方會命名為 **`opsworks_cookbook_demo::install_package`**。

> **Note**
>
> 當您在稍後重複此程序時，請輸入技術指南的名稱 (**opsworks_cookbook_demo**)，其後跟隨兩個冒號 (**::**)，其後再跟隨配方的名稱 (不帶有 .rb 副檔名的配方檔案名稱)。

5. 保留下列預設值：
   - Comment (註解) (空白)
   - Advanced (進階)、Custom Chef JSON (自訂 Chef JSON) (空白)
   - Instances (執行個體)、已核取 Select all (全選)，已核取 MyCookbooksDemoLayer，已核取 cookbooks-demo1
6. 選擇 Execute Recipes (執行配方)。即會顯示 Running command execute_recipes (執行 execute_recipes 命令) 頁面。在 Status (狀態) 變更為 successful (成功) 前請不要繼續。此程序可能需要花費幾分鐘，敬請耐心等待。

> **Note**
>
> 您不需要手動執行配方。您可以將配方指派給 layer 的生命週期事件，例如設定和設定事件，而 AWS OpsWorks Stacks 會在事件發生時自動執行這些配方。如需詳細資訊，請參閱AWS OpsWorks Stacks 生命週期事件。

在下一個步驟中，您會更新技術指南，將使用者新增至執行個體。

步驟 6：更新技術指南以新增使用者

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

透過新增將本機使用者新增至執行個體，並設定使用者的主目錄與殼層的配方，來更新您的技術指南。這與執行 Linux adduser 或 useradd 命令，或 Windows net user 命令相似。您可以將本機使用者新增至執行個體，例如：當您希望控制執行個體檔案與目錄的存取時。

您也可以在不使用技術指南的情況下管理使用者。如需詳細資訊，請參閱管理使用者。

更新執行個體上的技術指南及執行新的配方

1. 在您的本機工作站上，於 opsworks_cookbook_demo 目錄中的 recipes 子目錄內，建立名為 add_user.rb 的檔案，其中包含下列程式碼 (如需詳細資訊，請前往 user)：

```
user "Add a user" do
  home "/home/jdoe"
  shell "/bin/bash"
  username "jdoe"
end
```

2. 在終端機或命令提示中，使用 tar 命令建立 opsworks_cookbook_demo.tar.gz 檔案的新版本，其中包含 opsworks_cookbook_demo 目錄及其更新後的內容。
3. 將更新後的 opsworks_cookbook_demo.tar.gz 檔案上傳至 S3 儲存貯體。
4. 遵循步驟 5：更新執行個體上的技術指南及執行配方中的程序，更新執行個體上的技術指南及執行配方。在「執行配方」程序中，針對 Recipes to execute (要執行的配方)，輸入 **opsworks_cookbook_demo::add_user**。

測試配方

1. 登入執行個體 (若您尚未登入的話)。
2. 從命令提示中，執行下列命令以確認新的使用者已新增：

```
grep jdoe /etc/passwd
```

即會顯示與下列內容相似的使用者相關資訊，包含像是使用者名稱、ID 號碼、群組 ID 號碼、主目錄、殼層等詳細資訊：

```
jdoe:x:501:502::/home/jdoe:/bin/bash
```

在下一個步驟中，您會更新技術指南，在執行個體上建立目錄。

步驟 7：更新技術指南以建立目錄

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

透過新增會將目錄新增至執行個體的配方，來更新您的技術指南。這與執行 Linux mkdir 命令，或 Windows md 或 mkdir 命令相似。

更新執行個體上的技術指南及執行新的配方

1. 在您的本機工作站上，在 opsworks_cookbook_demo 目錄中的 recipes 子目錄中，建立名為 create_directory.rb 的檔案，其中包含下列程式碼。如需詳細資訊，請前往 directory：

```
directory "Create a directory" do
  group "root"
  mode "0755"
  owner "ec2-user"
  path "/tmp/create-directory-demo"
end
```

2. 在終端機或命令提示中，使用 tar 命令建立 opsworks_cookbook_demo.tar.gz 檔案的新版本，其中包含 opsworks_cookbook_demo 目錄及其更新後的內容。
3. 將更新後的 opsworks_cookbook_demo.tar.gz 檔案上傳至 S3 儲存貯體。
4. 遵循步驟 5：更新執行個體上的技術指南及執行配方中的程序，更新執行個體上的技術指南及執行配方。在「執行配方」程序中，針對 Recipes to execute (要執行的配方)，輸入 **opsworks_cookbook_demo::create_directory**。

測試配方

1. 登入執行個體 (若您尚未登入的話)。
2. 從命令提示中，執行下列命令以確認新的目錄已新增：

```
ls -la /tmp/create-directory-demo
```

即會顯示新建立目錄的相關資訊，包含像是許可、擁有者名稱和群組名稱等資訊：

```
drwxr-xr-x 2 ec2-user root 4096 Nov 18 00:35 .
drwxrwxrwt 6 root     root 4096 Nov 24 18:17 ..
```

在下一個步驟中，您會更新技術指南，在執行個體上建立檔案。

步驟 8：更新技術指南以建立及複製檔案

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

透過新增會將兩個檔案新增至執行個體的配方，來更新您的技術指南。配方中的第一個資源會完全使用配方程式碼來建立檔案。這與執行 Linux cat、echo 或 touch 命令，或 Windows echo 或 fsutil 命令相似。此技術對少量、小型或簡單的檔案相當有用。配方中的第二個資源會將技術指南中的檔案複製到執行個體上的另一個目錄。這與執行 Linux cp 命令或 Windows copy 命令相似。此技術對於大量、大型或複雜的檔案相當有用。

在您開始此步驟前，請先完成步驟 7：更新技術指南以建立目錄以確保檔案的父系目錄已存在。

更新執行個體上的技術指南及執行新的配方

1. 在您的本機工作站上，於 `opsworks_cookbook_demo` 目錄中，建立名為 `files` 的子目錄。
2. 在 `files` 子目錄中，建立名為 `hello.txt` 的檔案，其中包含下列文字：**`Hello, World!`**
3. 在 `opsworks_cookbook_demo` 目錄的 `recipes` 子目錄中，建立名為 `create_files.rb` 的檔案，其中包含下列程式碼。如需詳細資訊，請前往 file 和 cookbook_file。

```
file "Create a file" do
  content "<html>This is a placeholder for the home page.</html>"
  group "root"
  mode "0755"
  owner "ec2-user"
  path "/tmp/create-directory-demo/index.html"
end
```

```
cookbook_file "Copy a file" do
  group "root"
  mode "0755"
  owner "ec2-user"
  path "/tmp/create-directory-demo/hello.txt"
  source "hello.txt"
end
```

`file` 資源會在指定的路徑中建立檔案。`cookbook_file` 資源會從您在技術指南中建立的 `files` 目錄 (Chef 預期會找到名為 `files` 的標準命名目錄，並從中複製檔案)，將檔案複製到執行個體上的另一個目錄。

4. 在終端機或命令提示中，使用 tar 命令建立 `opsworks_cookbook_demo.tar.gz` 檔案的新版本，其中包含 `opsworks_cookbook_demo` 目錄及其更新後的內容。
5. 將更新後的 `opsworks_cookbook_demo.tar.gz` 檔案上傳至 S3 儲存貯體。
6. 遵循步驟 5：更新執行個體上的技術指南及執行配方中的程序，更新執行個體上的技術指南及執行配方。在「執行配方」程序中，針對 Recipes to execute (要執行的配方)，輸入 **`opsworks_cookbook_demo::create_files`**。

測試配方

1. 登入執行個體 (若您尚未登入的話)。
2. 從命令提示中，逐項執行下列命令以確認新的檔案已新增：

```
sudo cat /tmp/create-directory-demo/index.html

sudo cat /tmp/create-directory-demo/hello.txt
```

即會顯示檔案的內容：

```
<html>This is a placeholder for the home page.</html>

Hello, World!
```

在下一個步驟中，您會更新技術指南，在執行個體上執行命令。

步驟 9：更新技術指南以執行命令

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

透過新增會執行在執行個體上建立 SSH 金鑰之命令的配方，來更新您的技術指南。

更新執行個體上的技術指南及執行新的配方

1. 在您的本機工作站上，在 `opsworks_cookbook_demo` 目錄中的 `recipes` 子目錄中，建立名為 `run_command.rb` 的檔案，其中包含下列程式碼。如需詳細資訊，請前往 execute。

   ```
   execute "Create an SSH key" do
     command "ssh-keygen -f /tmp/my-key -N fLyC3jbY"
   end
   ```

2. 在終端機或命令提示中，使用 tar 命令建立 `opsworks_cookbook_demo.tar.gz` 檔案的新版本，其中包含 `opsworks_cookbook_demo` 目錄及其更新後的內容。
3. 將更新後的 `opsworks_cookbook_demo.tar.gz` 檔案上傳至 S3 儲存貯體。
4. 遵循步驟 5：更新執行個體上的技術指南及執行配方中的程序，更新執行個體上的技術指南及執行配方。在「執行配方」程序中，針對 Recipes to execute (要執行的配方)，輸入 **`opsworks_cookbook_demo::run_command`**。

測試配方

1. 登入執行個體 (若您尚未登入的話)。
2. 從命令提示中，逐項執行下列命令以確認 SSH 金鑰已建立：

   ```
   sudo cat /tmp/my-key

   sudo cat /tmp/my-key.pub
   ```

   即會顯示 SSH 私有和公有金鑰的內容：

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,DEF7A09C...541583FA
A5p9dCuo...wp0YYH1c
-----END RSA PRIVATE KEY-----

ssh-rsa AAAAB3N...KaNogZkT root@cookbooks-demo1
```

在下一個步驟中，您會更新技術指南，在執行個體上執行指令碼。

步驟 10：更新技術指南以執行指令碼

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

透過新增會在執行個體上執行指令碼的配方，來更新您的技術指南。此配方會建立目錄，然後在該目錄中建立檔案。撰寫配方執行包含多個命令的指令碼，會比一次執行一個命令輕鬆許多。

更新執行個體上的技術指南及執行新的配方

1. 在您的本機工作站上，在 opsworks_cookbook_demo 目錄中的 recipes 子目錄中，建立名為 run_script.rb 的檔案，其中包含下列程式碼。如需詳細資訊，請前往 script。

   ```
   script "Run a script" do
     interpreter "bash"
     code <<-EOH
       mkdir -m 777 /tmp/run-script-demo
       touch /tmp/run-script-demo/helloworld.txt
       echo "Hello, World!" > /tmp/run-script-demo/helloworld.txt
     EOH
   end
   ```

2. 在終端機或命令提示中，使用 tar 命令建立 opsworks_cookbook_demo.tar.gz 檔案的新版本，其中包含 opsworks_cookbook_demo 目錄及其更新後的內容。
3. 將更新後的 opsworks_cookbook_demo.tar.gz 檔案上傳至 S3 儲存貯體。

4. 遵循步驟 5：更新執行個體上的技術指南及執行配方中的程序，更新執行個體上的技術指南及執行配方。在「執行配方」程序中，針對 Recipes to execute (要執行的配方)，輸入 **opsworks_cookbook_demo::run_script**。

測試配方

1. 登入執行個體 (若您尚未登入的話)。
2. 從命令提示中，執行下列命令以確認新的檔案已新增：

```
sudo cat /tmp/run-script-demo/helloworld.txt
```

即會顯示檔案的內容：

```
Hello, World!
```

在下一個步驟中，您會更新技術指南，在執行個體上管理服務。

步驟 11：更新技術指南以管理服務

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

透過新增會在執行個體上管理服務的配方，來更新您的技術指南。這和執行 Linux service 命令，或 Windows net stop、net start 和其他類似命令相似。此配方會停止執行個體上的 crond 服務。

更新執行個體上的技術指南及執行新的配方

1. 在您的本機工作站上，在 opsworks_cookbook_demo 目錄中的 recipes 子目錄中，建立名為 manage_service.rb 的檔案，其中包含下列程式碼。如需詳細資訊，請前往 service。

```
service "Manage a service" do
  action :stop
  service_name "crond"
```

```
end
```

2. 在終端機或命令提示中，使用 tar 命令建立 opsworks_cookbook_demo.tar.gz 檔案的新版本，其中包含 opsworks_cookbook_demo 目錄及其更新後的內容。
3. 將更新後的 opsworks_cookbook_demo.tar.gz 檔案上傳至 S3 儲存貯體。
4. 遵循步驟 5：更新執行個體上的技術指南及執行配方中的程序，更新執行個體上的技術指南及執行配方。在「執行配方」程序中，針對 Recipes to execute (要執行的配方)，輸入 **opsworks_cookbook_demo::manage_service**。

測試配方

1. 登入執行個體 (若您尚未登入的話)。
2. 從命令提示中，執行下列命令以確認 crond 服務已停止：

   ```
   service crond status
   ```

   即會顯示下列資訊：

   ```
   crond is stopped
   ```

3. 若要重新啟動 crond 服務，請執行下列命令：

   ```
   sudo service crond start
   ```

   即會顯示下列資訊：

   ```
   Starting crond:  [  OK  ]
   ```

4. 若要確認 crond 服務已啟動，請再次執行下列命令：

   ```
   service crond status
   ```

   即會顯示與下列內容相似的資訊：

   ```
   crond (pid  3917) is running...
   ```

在下一個步驟中，您會更新技術指南，在執行個體上參考以自訂 JSON 存放的資訊。

步驟 12：更新技術指南以使用自訂 JSON

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

透過新增會參考存放於執行個體上之自訂 JSON 的配方，來更新您的技術指南。

您可以在您建立、更新、複製堆疊，或在您執行部署或堆疊命令時以自訂 JSON 格式指定資訊。舉例來說，這在您要讓一小部分不會變動的資料供您執行個體上的配方使用 (而非從資料庫取得資料) 時會非常有用。如需詳細資訊，請參閱使用自訂 JSON。

針對本演練，您會使用自訂 JSON 提供一些關於客戶發票的虛擬資訊。自訂 JSON 會在此步驟的稍後進行說明。

更新執行個體上的技術指南及執行新的配方

1. 在您的本機工作站上，在 `recipes` 目錄中的 `opsworks_cookbook_demo` 子目錄中，建立名為 `custom_json.rb` 的檔案，其中包含下列配方程式碼：

   ```
   Chef::Log.info("********** For customer '#{node['customer-id']}' invoice
    '#{node['invoice-number']}' **********")
   Chef::Log.info("********** Invoice line number 1 is a '#{node['line-items']
   ['line-1']}' **********")
   Chef::Log.info("********** Invoice line number 2 is a '#{node['line-items']
   ['line-2']}' **********")
   Chef::Log.info("********** Invoice line number 3 is a '#{node['line-items']
   ['line-3']}' **********")
   ```

   此配方會在日誌中顯示關於自訂 JSON 內值的訊息。

2. 在終端機或命令提示中，使用 tar 命令建立 `opsworks_cookbook_demo.tar.gz` 檔案的新版本，其中包含 `opsworks_cookbook_demo` 目錄及其更新後的內容。

3. 將更新後的 `opsworks_cookbook_demo.tar.gz` 檔案上傳至 S3 儲存貯體。

4. 遵循步驟 5：更新執行個體上的技術指南及執行配方中的程序，更新執行個體上的技術指南及執行配方。在「執行配方」程序中，針對 Recipes to execute (要執行的配方)，輸入

opsworks_cookbook_demo::custom_json。針對 Advanced (進階)、Custom Chef JSON (自訂 Chef JSON)，輸入下列自訂 JSON：

```
{
  "customer-id": "0123",
  "invoice-number": "9876",
  "line-items": {
    "line-1": "tractor",
    "line-2": "passenger car",
    "line-3": "trailer"
  }
}
```

測試配方

1. 在先前程序中的 Running command execute_recipes (執行 execute_recipes 命令) 頁面顯示時，針對 cookbooks-demo1 的 Log (日誌)，選擇 show (顯示)。即會顯示 execute_recipes 日誌頁面。
2. 向下捲動日誌，尋找看起來與下列內容相似的項目：

```
[2015-11-14T14:18:30+00:00] INFO: ********** For customer '0123' invoice '9876'
 **********
[2015-11-14T14:18:30+00:00] INFO: ********** Invoice line number 1 is a 'tractor'
 **********
[2015-11-14T14:18:30+00:00] INFO: ********** Invoice line number 2 is a 'passenger
 car' **********
[2015-11-14T14:18:30+00:00] INFO: ********** Invoice line number 3 is a 'trailer'
 **********
```

這些項目會顯示在 Advanced (進階)、Custom Chef JSON (自訂 Chef JSON) 方塊中輸入之自訂 JSON 的資訊。

在下一個步驟中，您將更新技術指南，從資料包取得資訊，這是 Stacks AWS OpsWorks 在每個執行個體上存放的堆疊設定集合。

步驟 13：更新技術指南以使用資料包

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

新增參考 Stacks AWS OpsWorks 存放在執行個體上一組資料包的堆疊設定的配方，以更新您的技術指南。此配方會在日誌中顯示存放於執行個體上之特定堆疊設定的相關訊息。如需更多資訊，請參閱AWS OpsWorks Stacks Data Bag 參考。

更新執行個體上的技術指南及執行新的配方

1. 在您的本機工作站上，在 `recipes` 目錄中的 `opsworks_cookbook_demo` 子目錄中，建立名為 `data_bags.rb` 的檔案，其中包含下列程式碼：

   ```
   instance = search("aws_opsworks_instance").first
   layer = search("aws_opsworks_layer").first
   stack = search("aws_opsworks_stack").first

   Chef::Log.info("********** This instance's instance ID is
    '#{instance['instance_id']}' **********")
   Chef::Log.info("********** This instance's public IP address is
    '#{instance['public_ip']}' **********")
   Chef::Log.info("********** This instance belongs to the layer '#{layer['name']}'
    **********")
   Chef::Log.info("********** This instance belongs to the stack '#{stack['name']}'
    **********")
   Chef::Log.info("********** This stack gets its cookbooks from
    '#{stack['custom_cookbooks_source']['url']}' **********")
   ```

   此配方會在日誌中顯示存放於執行個體上之特定堆疊設定的相關訊息。

2. 在終端機或命令提示中，使用 tar 命令建立 `opsworks_cookbook_demo.tar.gz` 檔案的新版本，其中包含 `opsworks_cookbook_demo` 目錄及其更新後的內容。

3. 將更新後的 `opsworks_cookbook_demo.tar.gz` 檔案上傳至 S3 儲存貯體。

4. 遵循步驟 5：更新執行個體上的技術指南及執行配方中的程序，更新執行個體上的技術指南及執行配方。在「執行配方」程序中，針對 Recipes to execute (要執行的配方)，輸入 **opsworks_cookbook_demo::data_bags**。

測試配方

1. 在先前程序中的 Running command execute_recipes (執行 execute_recipes 命令) 頁面顯示時，針對 cookbooks-demo1 的 Log (日誌)，選擇 show (顯示)。即會顯示 execute_recipes 日誌頁面。
2. 向下捲動日誌，尋找看起來與下列內容相似的項目：

```
[2015-11-14T14:39:06+00:00] INFO: ********** This instance's instance ID is
 'f80fa119-81ab-4c3c-883d-6028e52c89EX' **********
[2015-11-14T14:39:06+00:00] INFO: ********** This instance's public IP address is
 '192.0.2.0' **********
[2015-11-14T14:39:06+00:00] INFO: ********** This instance belongs to the layer
 'MyCookbooksDemoLayer' **********
[2015-11-14T14:39:06+00:00] INFO: ********** This instance belongs to the stack
 'MyCookbooksDemoStack' **********
[2015-11-14T14:39:06+00:00] INFO: ********** This stack gets its cookbooks from
 'https://s3.amazonaws.com/amzn-s3-demo-bucket/opsworks_cookbook_demo.tar.gz'
 **********
```

此配方會顯示存放於執行個體上之特定堆疊設定的相關訊息。

在下一個步驟中，您會更新技術指南，以多次執行配方程式碼。

步驟 14：更新技術指南以使用反覆運算

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

透過新增使用「反覆運算」(一種重複配方程式碼多次的技術) 的配方，來更新您的技術指南。此配方會在日誌中顯示包含多個內容之資料包項目的訊息。

更新執行個體上的技術指南及執行新的配方

1. 在您的本機工作站上，在 recipes 目錄中的 opsworks_cookbook_demo 子目錄中，建立名為 iteration_demo.rb 的檔案，其中包含下列程式碼：

```
stack = search("aws_opsworks_stack").first
Chef::Log.info("********** Content of 'custom_cookbooks_source' **********")

stack["custom_cookbooks_source"].each do |content|
  Chef::Log.info("********** '#{content}' **********")
end
```

> **Note**
>
> 相較於撰寫下列不使用反覆運算的配方程式碼，撰寫前述配方程式碼更簡短、彈性更佳，也較不容易發生錯誤：
>
> ```
> stack = search("aws_opsworks_stack").first
> Chef::Log.info("********** Content of 'custom_cookbooks_source' **********")
>
> Chef::Log::info("********** '[\"type\", \"#{stack['custom_cookbooks_source']
> ['type']}\"]' **********")
> Chef::Log::info("********** '[\"url\", \"#{stack['custom_cookbooks_source']
> ['url']}\"]' **********")
> Chef::Log::info("********** '[\"username\",
>  \"#{stack['custom_cookbooks_source']['username']}\"]' **********")
> Chef::Log::info("********** '[\"password\",
>  \"#{stack['custom_cookbooks_source']['password']}\"]' **********")
> Chef::Log::info("********** '[\"ssh_key\",
>  \"#{stack['custom_cookbooks_source']['ssh_key']}\"]' **********")
> Chef::Log::info("********** '[\"revision\",
>  \"#{stack['custom_cookbooks_source']['revision']}\"]' **********")
> ```

2. 在終端機或命令提示中，使用 tar 命令建立 opsworks_cookbook_demo.tar.gz 檔案的新版本，其中包含 opsworks_cookbook_demo 目錄及其更新後的內容。
3. 將更新後的 opsworks_cookbook_demo.tar.gz 檔案上傳至 S3 儲存貯體。
4. 遵循步驟 5：更新執行個體上的技術指南及執行配方中的程序，更新執行個體上的技術指南及執行配方。在「執行配方」程序中，針對 Recipes to execute (要執行的配方)，輸入 **opsworks_cookbook_demo::iteration_demo**。

測試配方

1. 在先前程序中的 Running command execute_recipes (執行 execute_recipes 命令) 頁面顯示時，針對 cookbooks-demo1 的 Log (日誌)，選擇 show (顯示)。即會顯示 execute_recipes 日誌頁面。
2. 向下捲動日誌，尋找看起來與下列內容相似的項目：

```
[2015-11-16T19:56:56+00:00] INFO: ********** Content of 'custom_cookbooks_source'
 **********
[2015-11-16T19:56:56+00:00] INFO: ********** '["type", "s3"]' **********
[2015-11-16T19:56:56+00:00] INFO: ********** '["url", "https://s3.amazonaws.com/
amzn-s3-demo-bucket/opsworks_cookbook_demo.tar.gz"]' **********
[2015-11-16T19:56:56+00:00] INFO: ********** '["username", "secret-key-value"]'
 **********
[2015-11-16T19:56:56+00:00] INFO: ********** '["password", "secret-access-key-
value"]' **********
[2015-11-16T19:56:56+00:00] INFO: ********** '["ssh_key", nil]' **********
[2015-11-16T19:56:56+00:00] INFO: ********** '["revision", nil]' **********
```

此配方會在日誌中顯示包含多個內容之資料包項目的訊息。資料包項目位於 `aws_opsworks_stack` 資料包中。資料包項目包含名為 `custom_cookbooks_source` 的內容。此內容的內部為六個名為 `type`、`url`、`username`、`password`、`ssh_key` 和 `revision` 的內容。此處也會顯示他們的值。

在下一個步驟中，您會更新技術指南，僅於滿足特定條件時執行配方程式碼。

步驟 15：更新技術指南以使用條件式邏輯

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

現在，透過新增使用「條件式邏輯」(一種只在條件滿足時才執行程式碼的技術) 的配方，來更新您的技術指南。如需詳細資訊，請前往 if Statements 和 case Statements。

此配方會根據資料包的內容執行兩項作業：在日誌中顯示訊息，識別執行執行個體的作業系統，並且只在作業系統為 Linux 時，透過使用特定 Linux 版本的正確套件管理員來安裝套件。此套件名為 tree。該套件為可將目錄清單視覺化的簡易應用程式。

更新執行個體上的技術指南及執行新的配方

1. 在您的本機工作站上，於 opsworks_cookbook_demo directory 中的 recipes 子目錄中，建立名為 conditional_logic.rb 的檔案，其中包含下列程式碼：

```
instance = search("aws_opsworks_instance").first
os = instance["os"]

if os == "Red Hat Enterprise Linux 7"
  Chef::Log.info("********** Operating system is Red Hat Enterprise Linux.
 **********")
elsif os == "Ubuntu 14.04 LTS" || os == "Ubuntu 16.04 LTS" || os == "Ubuntu 18.04
 LTS"
  Chef::Log.info("********** Operating system is Ubuntu. **********")
elsif os == "Microsoft Windows Server 2012 R2 Base"
  Chef::Log.info("********** Operating system is Windows. **********")
elsif os == "Amazon Linux 2015.03" || os == "Amazon Linux 2015.09" || os == "Amazon
 Linux 2016.03" || os == "Amazon Linux 2016.09" || os == "Amazon Linux 2017.03"
 || os == "Amazon Linux 2017.09" || os == "Amazon Linux 2018.03" || os == "Amazon
 Linux 2"
  Chef::Log.info("********** Operating system is Amazon Linux. **********")
elsif os == "CentOS Linux 7"
  Chef::Log.info("********** Operating system is CentOS 7. **********")
else
  Chef::Log.info("********** Cannot determine operating system. **********")
end

case os
when "Ubuntu 14.04 LTS", "Ubuntu 16.04 LTS", "Ubuntu 18.04 LTS"
  apt_package "Install a package with apt-get" do
    package_name "tree"
  end
when "Amazon Linux 2015.03", "Amazon Linux 2015.09", "Amazon Linux 2016.03",
 "Amazon Linux 2016.09", "Amazon Linux 2017.03", "Amazon Linux 2017.09", "Amazon
 Linux 2018.03", "Amazon Linux 2", "Red Hat Enterprise Linux 7", "CentOS Linux 7"
  yum_package "Install a package with yum" do
    package_name "tree"
  end
else
```

```
  Chef::Log.info("********** Cannot determine operating system type, or operating
 system is not Linux. Package not installed. **********")
end
```

2. 在終端機或命令提示中，使用 tar 命令建立 opsworks_cookbook_demo.tar.gz 檔案的新版本，其中包含 opsworks_cookbook_demo 目錄及其更新後的內容。
3. 將更新後的 opsworks_cookbook_demo.tar.gz 檔案上傳至 S3 儲存貯體。
4. 遵循步驟 5：更新執行個體上的技術指南及執行配方中的程序，更新執行個體上的技術指南及執行配方。在「執行配方」程序中，針對 Recipes to execute (要執行的配方)，輸入 **opsworks_cookbook_demo::conditional_logic**。

測試配方

1. 在先前程序中的 Running command execute_recipes (執行 execute_recipes 命令) 頁面顯示時，針對 cookbooks-demo1 的 Log (日誌)，選擇 show (顯示)。即會顯示 execute_recipes 日誌頁面。
2. 向下捲動日誌，尋找與下列內容相似的項目：

   ```
   [2015-11-16T19:59:05+00:00] INFO: ********** Operating system is Amazon Linux.
    **********
   ```

   由於執行個體的作業系統是 Amazon Linux 2016.09，因此日誌中只會顯示上述項目（配方程式碼中五個可能項目中的項目）。
3. 若作業系統為 Linux，配方會安裝 tree 套件。若要查看目錄內容的視覺化，請從希望查看的目錄，或使用該目錄的路徑 (例如 `tree /var/chef/runs`)，在命令提示中輸入 **tree**。

在下一個步驟中，您會更新技術指南，使用由 Chef 社群提供之外部技術指南的功能。

步驟 16：更新技術指南以使用社群技術指南

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

最後，更新技術指南以使用 Chef 社群提供之外部技術指南中的功能。您在本演練中會使用的外部技術指南可從 Chef Supermarket 取得。該網站為存取外部 Chef 技術指南的熱門位置。此外部技術指南提供自訂資源，可讓您下載及安裝應用程式，與您在步驟 4：更新技術指南以安裝套件中執行的作業相似。而除了套件外，此資源還可安裝 Web 應用程式和其他應用程式類型。

當技術指南依存於另一個技術指南時，您必須指定其對其他技術指南的依存性。若要宣告及管理技術指南依存性，我們建議您使用名為 Berkshelf 的工具。如需如何在本機工作站上安裝 Berkshelf 的詳細資訊，請參閱 Chef 網站上的關於 Berkshelf。

在您安裝 Berkshelf 之後，遵循這些程序以宣告技術指南的依存性，然後建立呼叫外部技術指南中資源的配方。

宣告技術指南依存性

1. 在您的本機工作站上，於 opsworks_cookbook_demo 目錄中，在 metadata.rb 檔案的結尾新增此行：

   ```
   depends "application", "5.0.0"
   ```

   這會宣告其依存於名為 application，版本 5.0.0 的技術指南。
2. 從 opsworks_cookbook_demo 目錄的根，執行下列命令。命令結尾的句號是有意的。

   ```
   berks init .
   ```

   Berkshelf 會建立您在稍後針對更進階的案例可使用的資料夾及檔案。我們在本演練中所需要的檔案為名為 Berksfile 的檔案。
3. 在 Berksfile 檔案的結尾新增此行：

   ```
   cookbook "application", "5.0.0"
   ```

   這會通知 Berkshelf 您希望使用 應用程式技術指南，版本 5.0.0。Berkshelf 可從 Chef Supermarket 下載。
4. 在終端機或命令提示中，從 opsworks_cookbook_demo 目錄的根執行下列命令：

   ```
   berks install
   ```

   Berkshelf 會為您的技術指南和應用程式技術指南建立依存性清單。Berkshelf 會在下一個程序中使用此依存性清單。

更新執行個體上的技術指南及執行新的配方

1. 在 recipes 目錄中的 opsworks_cookbook_demo 子目錄內，建立名為 dependencies_demo.rb 的檔案，其中包含下列程式碼：

   ```
   application "Install NetHack" do
     package "nethack.x86_64"
   end
   ```

   此配方會依存應用程式技術指南的應用程式資源，以在執行個體上安裝熱門的文字式冒險遊戲 NetHack。(您當然可以將其取代為任何其他的套件名稱，只要執行個體上的套件管理員可取得該套件即可。)

2. 從 opsworks_cookbook_demo 目錄的根，執行下列命令：

   ```
   berks package
   ```

   Berkshelf 使用先前程序中的依存性清單，來建立名為 cookbooks-*timestamp*.tar.gz 的檔案，其中包含 opsworks_cookbook_demo 目錄及其更新後的內容，包含技術指南的依存技術指南。將此檔案重新命名為 opsworks_cookbook_demo.tar.gz。

3. 將更新並重新命名後的 opsworks_cookbook_demo.tar.gz 檔案上傳至 S3 儲存貯體。

4. 遵循步驟 5：更新執行個體上的技術指南及執行配方中的程序，更新執行個體上的技術指南及執行配方。在「執行配方」程序中，針對 Recipes to execute (要執行的配方)，輸入 **opsworks_cookbook_demo::dependencies_demo**。

5. 在您執行配方之後，應該可以登入執行個體，然後在命令提示輸入 **nethack** 來開始遊玩。(如需遊戲的詳細資訊，請參閱 NetHack 和 NetHack Guidebook。)

在下一個步驟中，您可以清除用於本演練 AWS 的資源。此步驟為選用。當您繼續進一步了解 Stacks AWS OpsWorks 時，建議您繼續使用這些 AWS 資源。不過，保留這些 AWS 資源可能會導致 AWS 您的帳戶持續產生一些費用。如果您想要將這些 AWS 資源保留以供日後使用，您現在已完成本演練，您可以直接跳到 後續步驟。

步驟 17：(選用) 清除

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

為了避免 AWS 您的帳戶產生額外費用，您可以刪除 AWS 用於此演練的資源。這些 AWS 資源包括 S3 儲存貯體、 Stacks AWS OpsWorks 堆疊和堆疊的元件。(如需詳細資訊，請參閱 AWS OpsWorks 定價。) 不過，當您繼續進一步了解 AWS OpsWorks Stacks 時，建議您繼續使用這些 AWS 資源。如果您想要讓這些 AWS 資源保持可用，您現在已完成本演練，您可以跳至 後續步驟。

存放在您為此逐步解說建立的資源裡的內容，可包含個人識別資訊。如果您不希望再將此資訊存放在 AWS，請遵循本主題的步驟。

刪除 S3 儲存貯體

- 請參閱刪除 Amazon S3 儲存貯體。

刪除堆疊的執行個體

1. 在 AWS OpsWorks Stacks 主控台的服務導覽窗格中，選擇執行個體。即會顯示 Instances (執行個體) 頁面。
2. 針對 MyCookbooksDemoLayer 之 cookbooks-demo1 的 Actions (動作)，選擇 stop (停止)。在您看見確認訊息時，選擇 Stop (停止)。
3. 下列變更會在數分鐘內發生。請不要在下列所有項目皆完成之前繼續。
   - Status (狀態) 從 online (線上) 變更為 stopping (停止中)，最後變更為 stopped (已停止)。
   - online (線上) 從 1 變更為 0。
   - shutting down (關機中) 從 0 變更為 1，最後變回 0。
   - stopped (已停止) 最後從 0 變更為 1。
4. 針對 Actions (動作)，選擇 delete (刪除)。當您看到確認訊息時，請選擇 Delete。 AWS OpsWorks Stacks 會刪除執行個體並顯示無執行個體。

刪除堆疊

1. 在服務導覽窗格中，選擇 Stack (堆疊)。即會顯示 MyCookbooksDemoStack 頁面。
2. 選擇 Delete Stack (刪除堆疊)。當您看到確認訊息時，請選擇刪除。 AWS OpsWorks 堆疊會刪除堆疊並顯示儀表板頁面。

您也可以選擇性地刪除您用於本演練的 IAM 使用者和 Amazon EC2 金鑰對，如果您不想重複使用這些金鑰對來存取其他 AWS 服務和 EC2 執行個體。如需說明，請參閱刪除 IAM 使用者、Amazon EC2 金鑰對和 Linux 執行個體。

您現已完成本演練。如需詳細資訊，請參閱後續步驟。

後續步驟

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

現在您已完成本演練，您可以透過檢閱下列資源，進一步了解 AWS OpsWorks Stacks 對 Chef 技術指南的支援：

- 技術指南和配方 – 說明 Stacks AWS OpsWorks 目前支援的 Chef 和 Ruby 版本。也示範如何在執行個體上安裝及更新自訂技術指南，以及如何在執行個體上執行配方。
- Learn Chef – 提供 Chef 教學的連結、Chef 技能程式庫、完整的 Chef 文件，以及 Chef 培訓課程。
- 關於 Chef – 提供完整的 Chef 文件。特定參考主題包含：
  - About Cookbooks – 說明關鍵技術指南元件，例如屬性、配方、檔案、中繼資料和範本。
  - About Recipes – 說明配方的基礎，例如如何使用資料包、包含其他配方，以及在配方中使用 Ruby 程式碼。
  - Resources – 說明如何使用所有內建 Chef 資源，例如 `apt_package`、`cookbook_file`、`directory`、`execute`、`file` 和 `package`。
  - About the Recipe DSL – 說明如何使用陳述式撰寫 Chef 配方的程式碼，例如 `if`、`case`、`data_bag`、`data_bag_item` 及 `search`。
- About Templates – 說明如何使用內嵌式 Ruby (ERB) 範本動態產生靜態文字檔案，例如組態檔案。

- 學習追蹤 – 說明如何使用 Chef 管理執行個體、管理基本 Web 應用程式、開發和測試基礎設施程式碼、使用 Chef 分析等。
- Learning Chef – Chef 簡介。由 O'Reilly Media 出版。
- Learning Chef code examples – 提供程式碼範例，搭配由 O'Reilly Media 出版的《Learning Chef》書籍。

AWS OpsWorks Stacks 最佳實務

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節中的策略、技術和建議可協助您從 Stacks AWS OpsWorks 獲得最大效益和最佳成果。

主題

- 最佳實務：執行個體的根設備儲存
- 最佳實務：最佳化應用程式伺服器的數目
- 最佳實務：管理許可
- 最佳實務：管理和部署應用程式與技術指南
- 本機封裝技術指南依存性

最佳實務：執行個體的根設備儲存

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 本主題不適用於 Windows 執行個體，其必須是 Amazon Elastic Block Store 後端。

Amazon Elastic Compute Cloud (Amazon EC2) Linux 執行個體具有下列根裝置儲存選項。

- 執行個體存放區後端執行個體 – 根裝置是暫時的。

  如果您停止執行個體，根設備中的資料便會消失且無法復原。如需詳細資訊，請參閱 Amazon EC2 執行個體存放區。
- Amazon EBS 後端執行個體 – 根裝置是 Amazon EBS 磁碟區。

  如果您停止執行個體，Amazon EBS 磁碟區會持續存在。如果您重新啟動執行個體，系統會自動重新掛載磁碟區，並還原執行個體狀態和任何存放的資料。您也可以在不同的執行個體上掛載磁碟區。如需詳細資訊，請參閱 Amazon Elastic Block Store (Amazon EBS)。

要決定使用哪一種根設備儲存選項時，請考慮下列各項。

開機時間

初始啟動後，Amazon EBS 執行個體通常會更快重新啟動。

每種儲存類型的第一次啟動時間約略相同。這兩種類型都必須執行完整設定，其中包含相對較耗時的任務，例如從遠端儲存庫安裝套件。不過，在您之後重新啟動執行個體時請注意下列差異性：

- 執行個體存放區後端執行個體執行的設定任務與第一次啟動時相同，包括套件安裝。

  重新啟動所需的時間也與第一次啟動時間大致相同。
- Amazon EBS 後端執行個體會重新掛載根磁碟區並執行安裝配方。

  重新啟動通常比第一次啟動更加快速，因為安裝配方不需要執行任務 (例如重新安裝已安裝在根磁碟區的套件)。

成本

Amazon EBS 支援的執行個體成本更高：

- 使用執行個體存放區後端執行個體時，您只需要在執行個體執行時付費。
- 使用 Amazon EBS 後端執行個體時，無論執行個體是否正在執行，您都會支付 Amazon EBS 磁碟區的費用。

如需詳細資訊，請參閱 Amazon EBS 定價。

日誌

Amazon EBS 支援的執行個體會自動保留日誌：

- 使用執行個體存放區後端執行個體時，日誌會在執行個體停止時消失。

  您必須先擷取日誌，才能停止執行個體，或使用 CloudWatch Logs 等服務遠端存放選取的日誌。
- 使用 Amazon EBS 後端執行個體時，日誌會存放在 Amazon EBS 磁碟區中。

  您可以重新啟動執行個體或將磁碟區掛載在其他執行個體上，以檢視日誌。

相依性

這兩種儲存類型有不同的相依性：

- 執行個體存放區支援的執行個體取決於 Amazon S3。

  當您啟動執行個體時，必須從 Amazon S3 下載 AMI。
- Amazon EBS 後端執行個體取決於 Amazon EBS。

  當您啟動執行個體時，它必須掛載 Amazon EBS 根磁碟區。

建議：如果您不確定哪種儲存類型最適合您的需求，建議您從 Amazon EBS 執行個體開始。雖然 Amazon EBS 磁碟區會產生小額費用，但發生意外資料遺失的風險較低。

最佳實務：最佳化應用程式伺服器的數目

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

生產堆疊通常包含分散於多個可用區域的多部應用程式伺服器。不過，依據日期時間或週中的日，傳入的請求數目可能會有大幅差異。您可以執行足以處理最大預期負載數量的伺服器，但大多數情況下，您最後支付的伺服器容量都比實際需要的多。為了有效地執行您的站點，建議的實務就是讓伺服器數目與目前的請求數量相符。

AWS OpsWorks Stacks 提供三種方法來管理伺服器執行個體的數量。

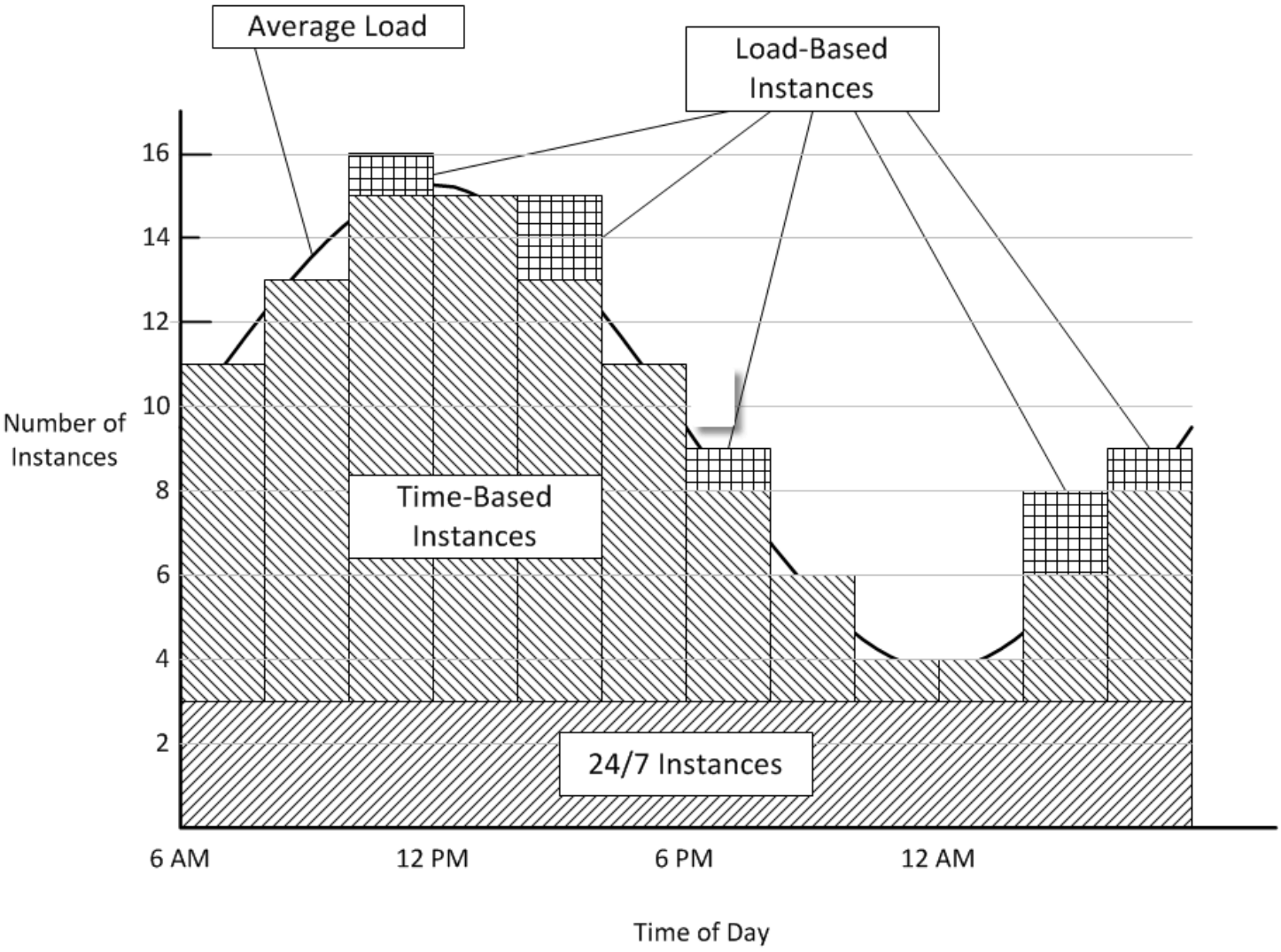
- 全年無休執行個體為手動啟動，並會持續執行到手動停止為止。
- Stacks AWS OpsWorks 會根據使用者指定的排程自動啟動和停止時間型執行個體。
- 當 Stacks 超過 CPU 或記憶體使用率等使用者指定的負載指標閾值時，負載型執行個體會自動啟動和停止。 AWS OpsWorks

> **Note**
>
> 在建立和設定堆疊的時間式和負載式執行個體之後， AWS OpsWorks Stacks 即會根據指定的組態自動啟動和停止這些執行個體。除非您決定變更組態或執行個體數目，否則您都不需要更動它們。

建議：如果您管理的堆疊上有很多個應用程式伺服器執行個體，我們建議您混合使用這三種執行個體類型。下列範例說明如何管理堆疊的伺服器容量，以處理具備下列特點的不同每日請求數量。

- 平均請求數量會在一天中出現正弦曲線般的差異。
- 平均請求數量最少要 5 個應用程式伺服器執行個體。
- 平均請求數量最多要 16 個應用程式伺服器執行個體。
- 通常，一或兩個應用程式伺服器執行個體即可處理遽增的請求數量。

這種便利的模型主要是基於討論用途而設，但您可以輕鬆地順應請求數量的變動來調整，也可以擴展這個模型以處理每週的變動。下圖說明如何使用這三種執行個體類型來管理此請求數量。



此範例具有下列特性：

- 堆疊具備 3 個全年無休執行個體，這些執行個體絕不中斷並會持續處理基本負載。
- 堆疊具備 12 個時間式執行個體，這些執行個體是設定為處理平均每日變動。

  其中一個從下午 10 點執行到上午 2 點，另外兩個從下午 8 點執行到下午 10 點和上午 2 點執行到上午 4 點，以此類推。為求簡化，此圖表會每兩小時修改一次時間式執行個體，但如果您需要更精確的控制，則可以每小時修改一次數目。
- 此堆疊具備的負載式執行個體足以處理超過全年無休和時間式執行個體可處理的流量高峰。

  AWS OpsWorks Stacks 只有在所有目前執行中的伺服器負載超過指定的指標時，才會啟動負載型執行個體。非執行中執行個體的成本極低 (Amazon EBS 後端執行個體） 或沒有成本 （執行個體後端執行個體），因此建議的做法是建立足夠的執行個體，以舒適地處理您預期的最大請求磁碟區。在這個範例中，堆疊應該至少有三個負載式執行個體。

> **Note**
>
> 請務必將這所有三種執行個體類型分散於多個可用區域，以降低任何服務中斷的影響。

最佳實務：管理許可

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您必須具備某種形式的 AWS 登入資料，才能存取您帳戶的資源。以下是一些將存取提供給您員工的一般性準則。

- 首先，也是第一步，我們建議您不要使用您帳戶的根登入資料存取 AWS 資源。

  反之，為您的員工建立 IAM 身分，並新增提供適當存取權的許可。然後，每個員工都可以使用自己的登入資料來存取資源。
- 員工應僅具備存取他們需要用來執行其工作之資源的許可。

  例如，應用程式開發人員僅需要存取執行其應用程式的堆疊。
- 員工應僅具備使用他們需要用來執行其工作之動作的許可。

  應用程式開發人員可能需要開發堆疊的完整許可，以及將他們的應用程式部署到對應生產堆疊的許可。他們可能不需要啟動或停止生產堆疊上執行個體的許可、建立或刪除 layer 的許可等。

如需管理許可的詳細一般性資訊，請參閱 AWS 安全登入資料。

您可以使用 AWS OpsWorks Stacks 或 IAM 來管理使用者許可。請注意，這兩種選項並非互斥關係，有時候您可能會希望同時使用兩者。

AWS OpsWorks Stacks 許可管理

每個堆疊都有一個 Permissions (許可) 頁面，您可以使用該頁面來授予使用者存取堆疊的許可，以及指定他們能夠執行的動作。您可以透過設定下列其中一個許可層級，來指定使用者的許可。每個關卡都代表一個 IAM 政策，授予一組標準動作的許可。

- Deny (拒絕) 會拒絕以任何方式與堆疊互動的許可。
- Show (顯示) 會授予檢視堆疊組態的許可，但是不會授予以任何方式修改堆疊狀態的許可。
- Deploy (部署) 包含 Show (顯示) 許可，並且也會授予使用者部署應用程式的許可。
- Manage (管理) 包含 Deploy (部署) 許可，並且也會允許使用者執行各種堆疊管理動作，例如建立或刪除執行個體和 layer。

> **Note**
>
> 管理許可層級不會授予少量高階 Stacks AWS OpsWorks 動作的許可，包括建立或複製堆疊。您必須使用 IAM 政策來授予這些許可。

除了設定許可層級之外，您也可以使用堆疊的 Permissions (許可) 頁面指定使用者是否在堆疊的執行個體上具有 SSH/RDP 和 sudo/admin 權限。如需 AWS OpsWorks Stacks 許可管理的詳細資訊，請參閱授予每個堆疊的許可。如需管理 SSH 存取的詳細資訊，請參閱管理 SSH 存取。

IAM 許可管理

透過 IAM 許可管理，您可以使用 IAM 主控台、API 或 CLI 將 JSON 格式政策連接至明確指定其許可的使用者。如需 IAM 許可管理的詳細資訊，請參閱什麼是 IAM？。

建議：從 AWS OpsWorks Stacks 許可管理開始。若您需要微調使用者的許可，或授予未包含在 Manage (管理) 許可層級中的使用者許可，您可以合併使用兩種方法。 AWS OpsWorks Stacks 接著會評估這兩個政策，以判斷使用者的許可。

> **Important**
>
> 如果使用者有多個具有衝突許可的政策，拒絕一律會獲勝。例如，假設您將 IAM 政策連接到允許存取特定堆疊的使用者，但也使用堆疊的許可頁面來為使用者指派拒絕許可層級。Deny (拒絕) 許可層級會取得優先順序，使用者將無法存取堆疊。如需詳細資訊，請參閱 IAM 政策評估邏輯。

例如，假設您希望除了新增和刪除 layer 之外，使用者能夠在堆疊上執行大多數的操作。

- 請指定 Manage (管理) 許可 layer 級，允許使用者執行大多數的堆疊管理動作，包含建立和刪除 layer。

- 再將以下客戶受管政策連接到使用者，拒絕在該堆疊上使用 CreateLayer 和 DeleteLayer 動作的許可。您可以透過其 Amazon Resource Name (ARN)https://docs.aws.amazon.com/glossary/latest/reference/glos-chap.html#ARN 識別堆疊，該資訊可在堆疊的 Settings (設定) 頁面上取得。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "opsworks:CreateLayer",
        "opsworks:DeleteLayer"
      ],
      "Resource": "arn:aws:opsworks:*:*:stack/2f18b4cb-4de5-4429-a149-ff7da9f0d8ee/"
    }
  ]
}
```

如需包含範例政策的詳細資訊，請參閱連接 AWS OpsWorks IAM 政策來管理 Stacks 許可。

> **Note**
>
> 使用 IAM 政策的另一種方法是設定條件，限制具有指定 IP 地址或地址範圍的員工對堆疊的存取。例如，若要確保員工僅能從您公司的防火牆內部存取堆疊，請設定將存取限制在您公司 IP 地址範圍內的條件。如需詳細資訊，請參閱條件。

最佳實務：管理和部署應用程式與技術指南

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 從遠端儲存庫將應用程式和技術指南部署到每個新執行個體。在執行個體的生命週期內，您必須經常更新堆疊的線上執行個體應用程式或技術指南，以新增功能、修正錯誤等等。有多種方法可以管理堆疊的應用程式和技術指南，但您使用的方法應該滿足下列一般要求：

- 所有生產堆疊執行個體應具有相同的應用程式和自訂技術指南程式碼，但某些有限的例外用於 A/B 測試等用途。
- 即使發生問題，部署更新也不應中斷網站的操作。

本節說明管理和部署應用程式與自訂技術指南的建議實務。

主題

- 維持一致性
- 部署程式碼到線上執行個體

維持一致性

通常，您需要維持嚴格控制在生產堆疊上執行的應用程式或技術指南程式碼。一般而言，所有執行個體應執行目前已核准版本的程式碼。更新應用程式或技術指南以及因應特殊情況 (例如執行 A/B 測試) 時會出現例外情況，如稍後所述。

應用程式和技術指南程式碼會以兩種方式從指定來源儲存庫部署到堆疊的執行個體：

- 當您啟動執行個體時， AWS OpsWorks Stacks 會自動將目前的應用程式和技術指南程式碼部署到執行個體。
- 對於線上執行個體，您必須透過執行部署命令 (適用於應用程式) 或更新自訂技術指南命令 (適用於技術指南) 來手動部署目前的應用程式或技術指南。

因為有兩種部署機制，所以仔細管理來源碼以避免在不同執行個體上無意中執行不同程式碼非常重要。例如，如果您從 Git 主分支部署應用程式或技術指南， AWS OpsWorks Stacks 會部署該分支當時的內容。如果您更新主分支中的程式碼，然後啟動新執行個體，該執行個體會將具有比舊執行個體的更新版程式碼。較新的版本甚至可能不受批准用於生產。

建議：Amazon S3 Archives

> 為了確保您的所有執行個體都具有核准的程式碼版本，我們建議您從 Amazon Simple Storage Service (Amazon S3) 封存部署應用程式和技術指南。這可確保程式碼是靜態成品，也就是 .zip 或其他封存檔案，必須明確更新。此外，Amazon S3 非常可靠，因此您很少會無法存取封存。為了進

一步確保一致性，請使用命名慣例或使用 Amazon S3 版本控制來明確版本化每個封存檔案，以提供稽核線索並輕鬆還原至較早版本。

例如，您可以使用像是 Jenkins 的工具來建立部署管道。在您要部署的程式碼完成遞交和測試之後，請建立封存檔案並將其上傳至 Amazon S3。所有應用程式部署或技術指南更新，都會在該封存檔案和安裝程式碼的每個執行個體中具有相同程式碼。

建議：Git 或 Subversion 儲存庫

如果您希望使用 Git 或 Subversion 儲存庫，請不要從主要分支部署。反之，請標記已核准版本並指定該版本為應用程式或技術指南來源。

部署程式碼到線上執行個體

AWS OpsWorks Stacks 不會自動將更新的程式碼部署到線上執行個體。您必須手動執行該操作，這會有下列挑戰：

- 有效部署更新，而不影響網站在部署程序中處理客戶請求的能力。
- 處理不成功的部署，可能是因為部署的應用程式或技術指南問題，也可能是部署程序本身的問題。

最簡單的方法是執行預設的部署命令 (適用於應用程式) 或更新自訂技術指南命令 (適用於技術指南)，會同時部屬每個執行個體的更新。這種方法非常簡單且快速，但沒有錯誤的餘地。若部署失敗或更新的程式碼發生任何問題，每個您生產堆疊中的執行個體都會受到影響，可能中斷或停用您的網站，直到您修正問題或是還原至先前版本為止。

建議：使用穩健的部署策略，可讓執行舊版本程式碼的執行個體繼續處理請求，直到您成功驗證部署並可以放心地傳輸所有傳入流量到新的版本。

下列各節提供兩個穩健部署策略的範例，接著會討論如何在部署期間管理後端資料庫。為簡潔起見，僅說明應用程式更新，但您也可以對技術指南使用類似的策略。

主題

- 使用滾動部署
- 使用單獨的堆疊
- 管理後端資料庫

使用滾動部署

滾動部署會以多個階段更新堆疊之線上應用程式伺服器執行個體的應用程式。對於每個階段，您將更新線上執行個體的子集合，並在下一階段開始之前驗證更新是否成功。如果您遇到問題，仍在執行舊應用程式版本的執行個體可以繼續處理傳入流量，直到您解決問題為止。

下列範例假設您使用的建議做法，是跨多個可用區域分發堆疊的應用程式伺服器執行個體。

執行滾動部署

1. 在部署應用程式頁面上，選擇 Advanced (進階)，然後選擇單一應用程式伺服器執行個體，並將應用程式部署到該執行個體。

   如果您想要謹慎行事，您可以在部署應用程式之前從負載平衡器移除該執行個體。這可確保使用者不會遇到更新的應用程式，直到您驗證其正確運作。如果您使用 Elastic Load Balancing，請使用 Elastic Load Balancing 主控台、CLI 或 SDK 從負載平衡器移除執行個體。
2. 請驗證更新的應用程式是否正常運作，以及執行個體是否具有可接受的效能指標。

   如果您從 Elastic Load Balancing 負載平衡器移除執行個體，請使用 Elastic Load Balancing 主控台、CLI 或 SDK 來還原執行個體。更新的應用程式版本現在會處理使用者請求。
3. 將更新部署到可用區域中的其餘執行個體，並驗證其是否正確運作並具有可接受的指標。
4. 對堆疊的其他可用區域重複步驟 3，一次進行一個區域。如果您想要特別謹慎，請重複步驟 1 – 3。

> **Note**
>
> 如果您使用 Elastic Load Balancing 負載平衡器，您可以使用其運作狀態檢查來驗證部署是否成功。但是，請將 ping 路徑設定為檢查相依性並驗證一切是否正常運作的應用程式，而不是僅確認應用程式伺服器正在執行的靜態檔案。

使用單獨的堆疊

管理應用程式的另一種方法，是為應用程式生命週期的每個階段使用單獨的堆疊。不同的堆疊有時稱為環境。此模式可讓您對不可公開存取的堆疊進行開發和測試。當您準備好部署更新時，請將使用者流量從託管目前應用程式版本的堆疊，切換到託管更新版本的堆疊。

主題

- 使用開發、暫存和生產堆疊
- 使用藍/綠部署策略

使用開發、暫存和生產堆疊

最常見的方法使用下列堆疊。

開發堆疊

使用開發堆疊執行任務，例如實作新功能或修正錯誤。開發堆疊實質上是一個原型生產堆疊，包含在生產堆疊中的相同 layer、應用程式、資源等等。由於開發堆疊通常不必處理與生產堆疊相同的負載，因此通常可以使用較少或較小的執行個體。

開發堆疊不公開；您可以控制存取，如下所示：

- 透過設定應用程式伺服器或負載平衡器安全群組入站規則來限制網路存取，以僅接受來自指定 IP 地址或地址範圍的傳入請求。

  例如，限制對公司地址範圍內地址的 HTTP、HTTPS 和 SSH 存取。
- 使用堆疊的許可頁面控制對 AWS OpsWorks Stacks 堆疊管理功能的存取。

  例如，授予開發團隊管理許可層級，並授予其他所有員工顯示許可。

預備堆疊

使用預備堆疊來測試和完成更新生產堆疊的候選。當您完成開發，請透過複製開發堆疊來建立預備堆疊。然後在預備堆疊上執行測試套件，並將更新部署到該堆疊，以修正發生的問題。

預備堆疊也不公開；您可以像控制開發堆疊一樣控制堆疊和網路存取。請注意，當您複製開發堆疊以建立預備堆疊時，您可以複製 AWS OpsWorks Stacks 許可管理授予的許可。不過，複製並不影響使用者的 IAM 政策所授予之許可。您必須使用 IAM 主控台、CLI 或軟體開發套件來修改這些許可。如需詳細資訊，請參閱管理使用者許可。

生產堆疊

生產堆疊是公開堆疊，可支援您目前的應用程式。當預備堆疊通過測試，您可將其提升為生產堆疊，並淘汰舊的生產堆疊。如需如何執行此作業的範例，請參閱 使用藍/綠部署策略。

> **Note**
>
> 請不要使用 AWS OpsWorks Stacks 主控台手動建立堆疊，而是為每個堆疊建立 AWS CloudFormation 範本。這種方法具有下列優勢：

- 速度和便利性 – 當您啟動範本時， AWS CloudFormation 會自動建立堆疊，包括所有必要的執行個體。
- 一致性 – 將每個堆疊的範本存放在您的來源儲存庫中，以確保開發人員將相同的堆疊用於相同的用途。

使用藍/綠部署策略

「藍/綠」部署策略的常見方法，是有效使用單獨堆疊來部署應用程式更新到生產堆疊。

- 藍色環境是生產堆疊，託管目前的應用程式。
- 綠色環境是預備堆疊，託管更新的應用程式。

當您準備好將更新的應用程式部署到生產環境，請將使用者流量從藍色堆疊切換到綠色堆疊，此綠色堆疊將成為新的生產堆疊。然後淘汰舊的藍色堆疊。

下列範例說明如何搭配 Route 53 和 Elastic Load Balancing 負載平衡器集區，使用 AWS OpsWorks Stacks 堆疊執行藍綠部署。進行切換之前，您應該先確保下列各項：

- 綠色堆疊上的應用程式更新已通過測試，並準備好用於生產。
- 綠色堆疊與藍色堆疊相同，差別在於其包含已更新的應用程式且不公開。

  兩個堆疊在每個 layer 中具有相同的許可、數目以及執行個體類型，相同以時間為基礎和以負載為基礎的組態等等。
- 所有綠色堆疊的全年無休執行個體、排定之以時間為基礎的執行個體都是在線。
- 您有一個 Elastic Load Balancing 負載平衡器集區，可以動態連接到任一堆疊中的圖層，並且可以預先暖機處理預期的流量。
- 您已使用 Route 53 加權路由功能，在包含混合負載平衡器的託管區域中建立記錄集。
- 您已為負載平衡器指派了非零權重，該負載平衡器連接至藍色堆疊的應用程式伺服器 layer，並且為未使用的負載平衡器指派了零權重。這可確保藍堆疊的負載平衡器處理所有傳入流量。

將使用者切換至綠色堆疊

1. 連接集區中其中一個未使用的負載平衡器到綠色堆疊的應用程式伺服器 layer。在某些情況下，例如，當您預期出現瞬間流量或如果您無法設定負載測試以逐步增加流量，請預先培養負載平衡器以處理預期的流量。

2. 在所有綠色堆疊的執行個體都通過 Elastic Load Balancing 運作狀態檢查後，變更 Route 53 記錄集中的權重，以便綠色堆疊的負載平衡器具有非零權重，而藍色堆疊的負載平衡器具有對應的減重。我們建議您先讓綠色堆疊處理一小部分請求，比方說 5%，而藍色堆疊處理其餘請求。您現在有兩個生產堆疊，綠色堆疊處理一些傳入請求，藍色堆疊處理其餘請求。
3. 監控綠色堆疊的效能指標。如果可接受，請增加綠色堆疊的權重，以便其可以處理約 10% 的傳入流量。
4. 請重複步驟 3，直到綠色堆疊處理約一半的傳入流量。此時任何問題都應已浮現，因此如果綠色堆疊的表現令人滿意，您可以透過將藍色堆疊的權重減少到零來完成程序。綠色堆疊現在是新的藍色堆疊，並處理所有傳入的流量。
5. 從舊藍色堆疊的應用程式伺服器 layer 分離負載平衡器並傳回到集區。
6. 雖然舊的藍色堆疊不再處理使用者請求，但我們建議保留一段時間，以防新藍色堆疊出現問題。若發生該情況，您可以透過反轉程序來復原更新，以將傳入流量引導回舊的藍色堆疊。當您確信新藍色堆疊的運作可接受，請關機舊藍色堆疊。

管理後端資料庫

如果您的應用程式依賴後端資料庫，您將需要從舊應用程式轉換到新的應用程式。 AWS OpsWorks Stacks 支援下列資料庫選項。

Amazon RDS Layer

使用 Amazon Relational Database Service (Amazon RDS) 層，您可以分別建立 RDS 資料庫執行個體，然後向堆疊註冊。您可以一次只用一個堆疊來註冊 RDS 資料庫執行個體，但您可以將 RDS 資料庫執行個體從某個堆疊切換到另一個堆疊。

AWS OpsWorks Stacks 會在您的應用程式伺服器上安裝具有連線資料的檔案，其格式可輕鬆供應用程式使用。 AWS OpsWorks Stacks 也會將資料庫連線資訊新增至堆疊組態和部署屬性，而這些屬性可由配方存取。您也可以使用 JSON 提供應用程式的連線資料。如需詳細資訊，請參閱連線至資料庫。

更新依賴於資料庫的應用程式會帶來兩個基本挑戰：

- 請確保在轉換期間正確記錄每個交易，同時避免新舊應用程式版本之間的競爭條件。
- 以限制對網站效能影響的方式執行轉換，並最大限度減少或排除停機時間。

使用本主題中說明的部署策略時，您不能僅從舊應用程式分離資料庫，再將其重新連接到新的應用程式。兩個版本的應用程式在轉換期間會平行執行，並且必須能夠存取相同的資料。下列說明了兩種管理轉換的方法，這兩種方法各自有其優點和挑戰之處。

方法 1：讓兩個應用程式連線到相同的資料庫

優點

- 在轉換期間沒有停機時間。

  一個應用程式會逐漸停止存取資料庫，而另一個應用程式會逐漸接管。

- 您不需要同步兩個資料庫之間的資料。

挑戰

- 兩個應用程式都存取相同的資料庫，因此您必須管理存取以防止資料遺失或損壞。
- 如果您需要遷移到新的資料庫結構描述，則舊的應用程式版本必須能夠使用新的結構描述。

如果您使用的是不同的堆疊，此方法可能最適合 Amazon RDS，因為執行個體不會永久綁定到特定堆疊，並且可以由在不同堆疊上執行的應用程式存取。但是，您無法一次註冊具有多個堆疊的 RDS 資料庫執行個體，因此您必須向兩個應用程式提供連線資料，例如使用 JSON。如需詳細資訊，請參閱使用自訂配方。

如果您使用滾動升級，新舊應用程式版本會託管在相同的堆疊上，因此您可以使用 Amazon RDS 或 MySQL layer。

方法 2：為每個應用程式版本提供自己的資料庫

優點

- 每個版本都有自己的資料庫，所以結構描述不需相容。

挑戰

- 在轉換期間同步兩個資料庫之間的資料，而不遺失或損壞資料。
- 確保您的同步程序不會導致嚴重停機時間，或大幅降低網站效能。

如果您使用單獨的堆疊，每個堆疊都有自己的資料庫。如果您使用滾動部署，則可以將兩個資料庫連接至堆疊，每個應用程式一個。如果舊應用程式和更新應用程式沒有相容的資料庫結構描述，則此方法更佳。

建議：通常，我們建議您使用 Amazon RDS layer 做為應用程式的後端資料庫，因為它更靈活，可用於任何轉換案例。如需如何處理轉換的詳細資訊，請參閱 Amazon RDS 使用者指南。

本機封裝技術指南依存性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以使用 Berkshelf 在本機封裝技術指南相依性、將套件上傳至 Amazon S3，以及修改堆疊以使用 Amazon S3 上的套件做為技術指南來源。傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

下列逐步解說說明如何將技術指南及其相依性預先封裝至 .zip 檔案，然後使用 .zip 檔案做為 Stacks 中 Linux AWS OpsWorks 執行個體的技術指南來源。第一個演練說明如何封裝單一技術指南。第二個演練說明如何封裝多個技術指南。

在開始之前，請先安裝 Chef Development Kit (又稱為 Chef DK)，此為由 Chef 社群建置的各種工具。您需要此項目才能使用 `chef` 命令列工具。

在 Chef 12 中本機封裝依存性

在 Chef 12 Linux 中，根據預設，Berkshelf 不會再安裝到堆疊執行個體。我們建議您在本機開發電腦上安裝及使用 Berkshelf，以在本機封裝您的技術指南相依性。將包含相依性的套件上傳至 Amazon S3。最後，修改您的 Chef 12 Linux 堆疊，使其使用上傳的套件做為技術指南來源。請在於 Chef 12 中封裝技術指南時注意下列差異。

1. 在本機電腦上，透過執行 `chef` 命令列工具建立技術指南。

   ```
   chef generate cookbook "server-app"
   ```

   此命令會建立技術指南、一個 Berksfile、一個 `metadata.rb` 檔案和配方目錄，並將他們置放在一個具有和技術指南相同名稱的資料夾中。以下範例顯示建立項目的結構。

   ```
   server-app <-- the cookbook you've just created
       ### Berksfile
       ### metadata.rb
       ### recipes
   ```

2. 在文字編輯器中，編輯 Berksfile，使其指向 server-app 技術指南依存的技術指南。在我們的範例中，我們希望 server-app 依存於來自 Chef Supermarket 的 java 技術指南。我們會指定版本 1.50.0 或更新的 minor (次要) 版本，但您可以在單引號中輸入任何發佈的版本。儲存您的變更並結束檔案。

```
source 'https://supermarket.chef.io'
cookbook 'java', '~> 1.50.0'
```

3. 編輯 metadata.rb 檔案來新增依存性。儲存您的變更並結束檔案。

```
depends 'java' , '~> 1.50.0'
```

4. 變更為 Chef 為您建立的 server-app 技術指南目錄，然後執行 package 命令建立技術指南的 tar 檔案。若您要封裝多個技術指南，建議您在所有技術指南存放的根目錄中執行此命令。若要封裝單一技術指南，請在技術指南目錄層級執行此命令。在此範例中，我們會在 server-app 目錄執行此命令。

```
berks package cookbooks.tar.gz
```

輸出結果與以下內容相似。tar.gz 檔案會在您的本機目錄內建立。

```
Cookbook(s) packaged to /Users/username/tmp/berks/cookbooks.tar.gz
```

5. 在 中 AWS CLI，將您剛建立的套件上傳至 Amazon S3。在您上傳至 S3 後記下技術指南套件的新 URL，您在堆疊設定中將需要此 URL。

```
aws s3 cp cookbooks.tar.gz s3://bucket-name/
```

輸出結果與以下內容相似。

```
upload: ./cookbooks.tar.gz to s3://bucket-name/cookbooks.tar.gz
```

6. 在 AWS OpsWorks Stacks 中，修改堆疊以使用您上傳做為技術指南來源的套件。

   a. 將 Use custom Chef cookbooks (使用自訂 Chef 技術指南) 設定設為 Yes (是)。

   b. 將 Repository type (儲存庫類型) 設定為 S3 Archive (S3 封存)。

   c. 在 Repository URL (儲存庫 URL) 中，貼上您在步驟 5 中上傳之技術指南套件的 URL。

儲存您的堆疊變更。

本機封裝單一技術指南依存性

1. 在本機電腦上，透過使用 chef 命令列工具建立技術指南：

   ```
   chef generate cookbook "server-app"
   ```

   此命令會建立技術指南及一個 Berksfile，並將他們置放在一個具有和技術指南相同名稱的資料夾中。

2. 變更為 Chef 為您建立的技術指南目錄，然後透過執行下列命令封裝所有項目：

   ```
   berks package cookbooks.tar.gz
   ```

   輸出如下：

   ```
   Cookbook(s) packaged to /Users/username/tmp/berks/cookbooks.tar.gz
   ```

3. 在 中 AWS CLI，將您剛建立的套件上傳至 Amazon S3：

   ```
   aws s3 cp cookbooks.tar.gz s3://bucket-name/
   ```

   輸出如下：

   ```
   upload: ./cookbooks.tar.gz to s3://bucket-name/cookbooks.tar.gz
   ```

4. 在 AWS OpsWorks Stacks 中，修改堆疊以使用您上傳做為技術指南來源的套件。

本機封裝多個技術指南依存性

此範例會建立兩個技術指南，並為他們封裝依存性。

1. 在本機電腦上，執行下列 chef 命令以產生兩個技術指南：

   ```
   chef generate cookbook "server-app"
   ```

```
chef generate cookbook "server-utils"
```

在此範例中，server-app 技術指南會執行 Java 組態，因此我們需要新增 Java 的依存性。

2. 編輯 `server-app/metadata.rb` 以在社群 Java 技術指南上新增依存性：

```
maintainer "The Authors"
maintainer_email "you@example.com"
license "all_rights"
description "Installs/Configures server-app"
long_description "Installs/Configures server-app"
version "0.1.0"
depends "java"
```

3. 透過編輯技術指南根目錄中的 Berksfile 檔案，告知 Berkshelf 要封裝的內容為何，如下所示：

```
source "https://supermarket.chef.io"
cookbook "server-app", path: "./server-app"
cookbook "server-utils", path: "./server-utils"
```

您的檔案結構現在看起來應該會像是這樣：

```
..
   ### Berksfile
   ### server-app
   ### server-utils
```

4. 最後，建立 zip 套件，將其上傳至 Amazon S3，並修改 Stacks AWS OpsWorks 堆疊以使用新的技術指南來源。若要執行此作業，請遵循本機封裝單一技術指南依存性中的步驟 2 到 4。

其他資源

如需封裝技術指南依存性的詳細資訊，請參閱下列項目。

- AWS DevOps 部落格上的如何使用 Berkshelf 本機封裝技術指南依存性
- AWS OpsWorks 論壇上的 Linux Chef 12 搭配 Berkshelf
- Chef 12 中的 Berkshelf 在 AWS OpsWorks 論壇上
- 本指南中的安裝自訂技術指南
- 本指南中的技術指南儲存庫

堆疊

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

堆疊是最上層 AWS OpsWorks 的 Stacks 實體。它代表您想要集合管理的一組執行個體，通常是因為它們具有常見用途，例如服務 PHP 應用程式。除當成容器使用之外，堆疊還可以處理適用於整個執行個體群組的任務，例如管理應用程式和技術指南。

例如，堆疊的用途是為可能看起來類似以下項目的 Web 應用程式提供服務：

- 一組應用程式伺服器執行個體，每個處理一部分的傳入流量。
- 負載平衡器執行個體，接收傳入流量並將它分發至應用程式伺服器。
- 資料庫執行個體，做為應用程式伺服器的後端資料存放區。

常見的做法是讓多個堆疊代表不同的環境。堆疊集一般包含：

- 開發人員使用的開發堆疊，新增功能、修復錯誤及執行其他開發和維護任務。
- 預備堆疊，驗證更新或修正後再公開它們。
- 生產堆疊，這是處理使用者傳入請求的公開版本。

本節說明使用堆疊的基本概念。

主題

- 從 Amazon EC2-Classic 遷移堆疊至 VPC
- 建立新的堆疊
- 在 VPC 中執行堆疊
- 更新堆疊
- 複製堆疊
- 執行 AWS OpsWorks Stacks Stack 命令
- 使用自訂 JSON

- 刪除堆疊

從 Amazon EC2-Classic 遷移堆疊至 VPC

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本主題說明如何將 AWS OpsWorks Stacks 堆疊從 Amazon EC2 Classic 網路平台遷移至 Amazon Virtual Private Cloud (Amazon VPC) 網路。

如果您在 2013-12-04 之前建立 AWS 帳戶，您可能在某些 AWS 區域中支援 EC2-Classic。有些 Amazon EC2 資源和功能 (例如增強型聯網和較新的執行個體類型) 需要 Virtual Private Cloud (VPC)。有些資源可在 EC2-Classic 與 VPC 之間共用，而有些卻不能。為了避免服務中斷，建議您將 AWS OpsWorks Stacks 堆疊遷移至 VPC。

主題

- 先決條件
- 將 AWS OpsWorks Stacks 堆疊遷移至 VPC
- 另請參閱

先決條件

開始之前，您必須擁有符合 AWS OpsWorks Stacks 組態需求的 VPC。若要在 VPC 中設定私有子網路 AWS OpsWorks Stacks，請參閱本指南在 VPC 中執行堆疊中的 。您可以使用 Amazon VPC 管理主控台建立自訂 VPC。如需詳細資訊，請參閱《Amazon Virtual Private Cloud 使用者指南》中的 Amazon VPC 主控台精靈組態和 VPCs 和子網路。 Amazon Virtual Private Cloud

若要繼續遷移，您需要 VPC ID 和要使用的子網路 ID。

將 AWS OpsWorks Stacks 堆疊遷移至 VPC

首先，使用 AWS OpsWorks Stacks 主控台或 API 複製現有的 EC2-Classic 堆疊。然後，將現有堆疊的資源移至新堆疊。在複製的堆疊中啟動新的執行個體，並部署應用程式。驗證新堆疊是否正常運作。最後，從 EC2-Classic 堆疊刪除 EC2-Classic 資源，然後刪除舊堆疊。

1. 將現有的 EC2-Classic 堆疊複製到 VPC。複製堆疊會將堆疊設定、 layer、應用程式、使用者和使用者許可複製到新堆疊。如需如何複製堆疊的詳細資訊，請參閱本指南複製堆疊中的 。

   您也可以使用 AWS OpsWorks Stacks API 複製堆疊。當您使用 AWS CLI AWS SDKs複製堆疊時，請將 `VpcId` 參數的值設定為您在 中建立的 VPC ID先決條件。如需詳細資訊，請參閱 AWS OpsWorks Stacks API 參考中的 CloneStack。

2. 在複製堆疊的圖層中建立新的執行個體。請務必指定您在 中建立的子網路 ID先決條件。如需如何在堆疊中建立執行個體的詳細資訊，請參閱本指南將執行個體新增至 Layer中的 。

3. 將傳統資源遷移至 VPC，例如 EC2 安全群組、Elastic Load Balancing 負載平衡器和彈性 IP 地址，然後將它們與複製的堆疊建立關聯。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的將您的資源遷移至 VPC。

4. 向複製的堆疊註冊 Amazon EBS 磁碟區和 Amazon RDS 執行個體。如需使用堆疊註冊資源的詳細資訊，請參閱本指南向堆疊註冊資源中的 。

   Amazon EBS 磁碟區未與 VPC 相關聯，您可以在 EC2-Classic 堆疊和 VPC 中的堆疊中跨執行個體使用它們。您可以在 EC2-Classic 中向 EC2-Classic 堆疊和 VPC 中的堆疊註冊 Amazon RDS 執行個體。

5. 在複製的堆疊中啟動執行個體，然後將一小部分的工作負載移至複製的堆疊。例如，將一小部分的流量移至複製堆疊中的 Elastic Load Balancing 負載平衡器。如果您使用的是 Amazon Route 53，請參閱《Amazon Route 53 開發人員指南》中的將流量路由到 ELB 負載平衡器。

   只路由一小部分的流量，直到您確定新堆疊正常運作並支援您的應用程式為止。讓新的堆疊在試用期間使用一小部分的流量，例如一週。驗證新堆疊是否正常運作後，請將剩餘的流量路由到堆疊。

6. 確定複製的堆疊正常運作後，請將剩餘的生產流量或工作負載移至複製的堆疊。您現在可以停止 EC2-Classic 堆疊中的執行個體。我們建議您將舊堆疊保持可用數週，以便在遷移後數週內新堆疊發生任何問題時，將工作負載移回舊堆疊。

7. 當新堆疊已運作幾週時，請刪除 EC2-Classic 堆疊中的執行個體。如需如何刪除執行個體的詳細資訊，請參閱本指南刪除 AWS OpsWorks Stacks 執行個體中的 。

   > **Important**
   >
   > 請勿使用 Amazon EC2 主控台或 API 來停止或刪除 AWS OpsWorks 執行個體。

8. 刪除 EC2-Classic 堆疊中的應用程式。如需如何刪除應用程式的詳細資訊，請參閱本指南中的從堆疊刪除應用程式。

9. 刪除 EC2-Classic 堆疊。如需如何刪除堆疊的詳細資訊，請參閱本指南刪除堆疊中的 。

另請參閱

- 從 EC2-Classic 遷移至 VPC
- 偵錯和故障診斷指南
- 在 VPC 中執行堆疊

建立新的堆疊

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

若要建立新的堆疊，請在 Stacks AWS OpsWorks 儀表板上按一下新增堆疊。然後，您可以使用 Add Stack (新增堆疊) 頁面來設定堆疊。在您完成後，按一下 Add Stack (新增堆疊)。

主題

- 選擇要建立的堆疊類型
- 基本選項
- 進階選項

選擇要建立的堆疊類型

在您建立堆疊前，您必須決定要建立的堆疊類型。如需協助，請參閱下表。

如果您想要建立...	在下列情況下建立此堆疊類型...	若要了解建立方法，請遵循這些說明：
堆疊範例	探索 AWS OpsWorks 的基本概念，包括 Linux Chef 12 堆疊和 Node.js 應用程式範例。	入門：範例
Linux Chef 12 堆疊	建立使用 AWS OpsWorks 支援之最新版 Chef 的 Linux 堆	入門：Linux

如果您想要建立...	在下列情況下建立此堆疊類型...	若要了解建立方法，請遵循這些說明：
	疊。如果您是進階的 Chef 使用者，並想要享有廣泛的社群 cookbook (技術指南) 選擇或撰寫您自己的自訂 cookbook (技術指南)，請選擇此選項。如需詳細資訊，請參閱Chef 12 Linux。	
Windows Chef 12.2 堆疊	建立 Windows 堆疊。	入門：Windows
Linux Chef 11.10 堆疊	如果您的組織需要搭配 Linux 使用 Chef 11.10 以回溯相容，請建立此堆疊。	Chef 11 Linux 堆疊入門

基本選項

Add Stack (新增堆疊) 頁面具有下列基本選項。

Stack name (堆疊名稱)

（必要）用於在 Stacks AWS OpsWorks 主控台中識別堆疊的名稱。名稱不需要是唯一的。 AWS OpsWorks Stacks 也會產生堆疊 ID，這是可唯一識別堆疊的 GUID。例如，使用 update-stack 等 AWS CLI 命令，您可以使用堆疊 ID 來識別特定堆疊。在您建立堆疊後，您可以選擇導覽窗格中的 Stack (堆疊)，然後選擇 Stack Settings (堆疊設定)，來尋找其 ID。此 ID 會標記為 OpsWorks ID。

區域

(必要) 要啟動執行個體的 AWS 區域。

VPC

(選用) 要啟動堆疊之目標 VPC 的 ID。所有執行個體都會啟動至此 VPC，而且您之後無法變更 ID。

- 如果您的帳戶支援 EC2 Classic，而且您不想要使用 VPC，您可以指定 No VPC (無 VPC) (預設值)。

  如需 EC2 Classic 的詳細資訊，請參閱「支援的平台」。
- 如果您的帳戶不支援 EC2 Classic，您必須指定 VPC。

預設設定為 Default VPC (預設 VPC)，其結合 EC2 Classic 簡單易用的特點與 VPC 聯網功能的優點。如果您想要在一般 VPC 中執行堆疊，您必須使用 VPC 主控台、API 或 CLI 來建立該 VPC。如需如何為 AWS OpsWorks Stacks 堆疊建立 VPC 的詳細資訊，請參閱在 VPC 中執行堆疊。如需一般資訊，請參閱 Amazon Virtual Private Cloud。

預設可用區域/預設子網路

(選用) 此設定取決於您是否在 VPC 中建立堆疊：

- 如果您的帳戶支援 EC2 Classic，而且您將 VPC 設為 No VPC (無 VPC)，此設定會標記為 Default Availability Zone (預設可用區域)，其會指定要啟動執行個體的預設 AWS 可用區域。
- 如果您的帳戶不支援 EC2 Classic，或您選擇指定 VPC，此欄位會標記為 Default subnet (預設子網路)，其會指定要啟動執行個體的預設子網路。您可以在建立執行個體時覆寫此值，藉此在其他子網路中啟動執行個體。每個子網路皆會與一個可用區域相關聯。

您可以在建立執行個體時覆寫此設定，讓 AWS OpsWorks Stacks 在不同可用區域或子網路中啟動執行個體。

如需如何在 VPC 中執行堆疊的詳細資訊，請參閱「在 VPC 中執行堆疊」。

預設作業系統

(選用) 根據預設安裝於每個執行個體上的作業系統。您有下列選項：

- 其中一個內建 Linux 作業系統。
- Microsoft Windows Server 2012 R2。
- 以其中一個支援作業系統為基礎的自訂 AMI。

如果您選取 Use custom AMI (使用自訂 AMI)，則作業系統取決於您在建立執行個體時指定的自訂 AMI。如需詳細資訊，請參閱使用自訂 AMI。

如需可用作業系統的詳細資訊，請參閱「AWS OpsWorks Stacks 作業系統」。

> **Note**
>
> 您可以在建立執行個體時覆寫預設作業系統。不過，您無法覆寫 Linux 作業系統以指定 Windows，或覆寫 Windows 以指定 Linux 作業系統。

預設 SSH 金鑰

（選用）來自堆疊區域的 Amazon EC2 金鑰對。預設值為 none (無)。如果您指定金鑰對， AWS OpsWorks Stacks 會在執行個體上安裝公有金鑰。

- 若是 Linux 執行個體，您可以使用私有金鑰透過 SSH 用戶端登入堆疊的執行個體。

  如需詳細資訊，請參閱使用 SSH 登入。
- 使用 Windows 執行個體時，您可以使用私有金鑰搭配 Amazon EC2 主控台或 CLI 來擷取執行個體的管理員密碼。

  然後您可以使用該密碼透過 RDP 用戶端以管理員身分登入執行個體。如需詳細資訊，請參閱使用 RDP 登入。

如需如何管理 SSH 金鑰的詳細資訊，請參閱「管理 SSH 存取」。

> **Note**
>
> 您可以在建立執行個體時指定不同金鑰對或不指定任何金鑰對，來覆寫此設定。

Chef 版本

這會顯示您已選擇的 Chef 版本。

如需 Chef 版本的詳細資訊，請參閱「Chef 版本」。

使用自訂 Chef 技術指南

是否要在堆疊的執行個體上安裝您的自訂 Chef cookbook (技術指南)。

若是 Chef 12，預設設定為 Yes (是)。對於 Chef 11，預設設定為否。 是選項會顯示數個額外的設定，為 AWS OpsWorks Stacks 提供從其儲存庫部署自訂技術指南至堆疊執行個體所需的資訊，例如儲存庫 URL。詳細資訊取決於您為 cookbook (技術指南) 使用的儲存庫。如需詳細資訊，請參閱安裝自訂技術指南。

堆疊色彩

（選用）用來代表 Stacks AWS OpsWorks 主控台上堆疊的色調。您可以為不同的堆疊使用不同的色彩來協助區分，例如區分開發、預備和生產堆疊。

堆疊標籤

您可以在堆疊和 layer 層級套用標籤。建立標籤時，您會將標籤套用於已加上標籤的結構中每項資源。例如，如果您將標籤套用至堆疊，則您要將標籤套用至 layer 中的每個 layer，以及每個 layer 中的每個執行個體、Amazon EBS 磁碟區或 Elastic Load Balancing 負載平衡器。如需如何啟用標籤並使用它們來追蹤和管理 AWS OpsWorks Stacks 資源成本的詳細資訊，請參閱 帳單與成本管理使用者指南中的使用成本分配標籤和啟用使用者定義的成本分配標籤。如需在 Stacks AWS OpsWorks 中標記的詳細資訊，請參閱 標籤。

進階選項

對於進階設定，按一下 Advanced >> (進階 >>) 以顯示 Advanced options (進階選項) 和 Security (安全) 區段。

Advanced options (進階選項) 區段具有下列選項：

預設根設備類型

決定要用於執行個體根磁碟區的儲存體類型。如需詳細資訊，請參閱儲存體。

- Linux 堆疊預設使用 Amazon EBS 支援的根磁碟區，但您也可以指定執行個體存放區支援的根磁碟區。
- Windows 堆疊必須使用 Amazon EBS 支援的根磁碟區。

IAM 角色

（選用） 堆疊的 AWS Identity and Access Management (IAM) 角色，由 AWS OpsWorks Stacks 代表您與 AWS 互動。

預設 IAM 執行個體描述檔

（選用） 要與堆疊的 Amazon EC2 執行個體相關聯的預設 IAM 角色。此角色會將存取 AWS 資源 (例如 S3 儲存貯體) 的許可授予在堆疊執行個體上執行的應用程式。

- 若要將特定許可授予應用程式，請選擇具備適當政策的現有執行個體描述檔 (角色)。
- 一開始，設定檔的角色不會授予任何許可，但您可以使用 IAM 主控台、API 或 CLI 來連接適當的政策。如需詳細資訊，請參閱指定在 EC2 執行個體上執行之應用程式的許可。

API 端點區域

此設定會使用您在堆疊之基本設定中選擇的區域做為其值。您可以從下列區域端點中選擇。

- 美國東部 (維吉尼亞北部) 區域

- 美國東部 (俄亥俄) 區域
- 美國西部 (奧勒岡) 區域
- 美國西部 (加利佛尼亞北部) 區域
- 加拿大 （ 中部 ） 區域 （ 僅限 API；不適用於在 中建立的堆疊 AWS Management Console
- 亞太 (孟買) 區域
- 亞太區域 (新加坡) 區域
- 亞太 (雪梨) 區域
- 亞太 (東京) 區域
- 亞太區域 (首爾) 區域
- 歐洲 (法蘭克福) 區域
- 歐洲 (愛爾蘭) 區域
- 歐洲 (倫敦) 區域
- 歐洲 (巴黎) 區域
- 南美洲 (聖保羅) 區域

在某個 API 端點中建立的堆疊無法在另一個 API 端點中使用。由於 AWS OpsWorks Stacks 使用者也是區域特定的，如果您希望其中一個端點區域中的 AWS OpsWorks Stacks 使用者管理另一個端點區域中的堆疊，您必須將使用者匯入至與堆疊相關聯的端點。如需匯入使用者的詳細資訊，請參閱將使用者匯入 AWS OpsWorks Stacks。

主機名稱主題

(選用) 用來產生每個執行個體之預設主機名稱的字串。預設值為 Layer Dependent (與堆疊層相依)，這會使用執行個體堆疊層的簡短名稱，並對每個執行個體附加唯一號碼。例如，與角色相依之負載平衡器主題的根為 "lb"。您新增至該堆疊層的第一個執行個體會命名為 "lb1"，第二個執行個體會命名為 "lb2"，以此類推。

OpsWorks Agent version (OpsWorks 代理程式版本)

（ 選用 ） 是否在新版本可用時自動更新 AWS OpsWorks Stacks 代理程式，或使用指定的代理程式版本並手動更新。此功能適用於 Chef 11.10 和 Chef 12 堆疊。預設設定為 Manual update (手動更新)，並已設為最新代理程式版本。

AWS OpsWorks Stacks 會在與服務通訊的每個執行個體上安裝 代理程式，並處理任務，例如啟動 Chef 執行以回應生命週期事件。此代理程式會定期更新。您可使用兩種選項指定堆疊的代理程式版本。

- 自動更新 – 一旦更新可用， AWS OpsWorks 堆疊會在堆疊的執行個體上自動安裝每個新的代理程式版本。
- 手動更新 – AWS OpsWorks Stacks 會在堆疊的執行個體上安裝指定的代理程式版本。

  AWS OpsWorks 當新的代理程式版本可用時，Stacks 會在堆疊頁面上張貼訊息，但不會更新堆疊的執行個體。若要更新代理程式，您必須手動更新堆疊設定以指定新的代理程式版本，然後 AWS OpsWorks Stacks 會更新堆疊的執行個體。

您可以更新其組態，覆寫特定執行個體的預設 OpsWorks 代理程式版本設定。在此情況下，會優先使用執行個體的設定。例如，假設預設設定為 Auto-update (自動更新)，但您為特定執行個體指定 Manual update (手動更新)。當 AWS OpsWorks Stacks 發行新的代理程式版本時，它會自動更新堆疊的所有執行個體，但設定為手動更新的執行個體除外。若要在該執行個體上安裝新的代理程式版本，您必須手動更新其組態並指定新的版本。

> **Note**
>
> 主控台會顯示縮寫的代理程式版本號碼。若要查看完整版本號碼，請呼叫 AWS CLI describe-agent-versions 命令，或是同等的 API 或 SDK 方法。它們會傳回可用代理程式版本的完整版本號碼。

自訂 JSON

(選用) 一或多個自訂屬性，並會格式化為 JSON 結構。這些屬性會合併為堆疊組態和部署屬性，並安裝在每個執行個體上且可供配方使用。例如，您可以透過覆寫指定預設設定的內建屬性，來使用自訂 JSON 自訂組態設定。如需詳細資訊，請參閱使用自訂 JSON。

安全性有一個選項：使用 OpsWorks 安全群組，可讓您指定是否要將 AWS OpsWorks Stacks 內建安全群組與堆疊層建立關聯。

AWS OpsWorks Stacks 提供一組標準內建安全群組，每個 layer 各一個，預設會與 layer 相關聯。Use OpsWorks security groups (使用 OpsWorks 安全群組) 可讓您改為提供自己的自訂安全群組。如需詳細資訊，請參閱使用安全群組。

Use OpsWorks security groups (使用 OpsWorks 安全群組) 具有下列設定：

- 是 - AWS OpsWorks Stacks 會自動將適當的內建安全群組與每個 layer 建立關聯 （ 預設設定 ）。

  您可以在建立堆疊層後，將其他安全群組與其建立關聯，但無法刪除內建安全群組。

- 否 - AWS OpsWorks Stacks 不會將內建安全群組與 layer 建立關聯。

  您必須建立適當的 EC2 安全群組，並將安全群組與您建立的每個堆疊層建立關聯。不過，您仍然可以在建立時手動將內建安全群組與堆疊層建立關聯；只有需要自訂設定的堆疊層才需要自訂安全群組。

注意下列事項：

- 如果 Use OpsWorks security groups (使用 OpsWorks 安全群組) 設為 Yes (是)，您無法透過將更嚴格的安全群組新增至堆疊層，來限制預設安全群組的連接埠存取設定。對於多個安全群組，Amazon EC2 會使用最寬鬆的設定。此外，您無法透過修改內建安全群組組態，來建立更嚴格的設定。當您建立堆疊時， AWS OpsWorks Stacks 會使用標準設定覆寫內建安全群組的組態，因此您所做的任何變更都會在下次建立堆疊時遺失。如果 layer 需要比內建安全群組更嚴格的安全群組設定，請將 Use OpsWorks security groups (使用 OpsWorks 安全群組) 設為 No (否)，並使用您偏好的設定建立自訂安全群組，然後在建立時將其指派給 layer。
- 如果您不小心刪除 Stacks AWS OpsWorks 安全群組，並想要重新建立該群組，它必須完全複製原始群組，包括群組名稱的大寫。與其手動重新建立群組，建議您讓 AWS OpsWorks Stacks 為您執行這項任務。只要在相同的 AWS 區域和 VPC 中建立新的堆疊，如果有的話， Stacks AWS OpsWorks 就會自動重新建立所有內建安全群組，包括您刪除的安全群組。您接著可以刪除您不再需要使用的堆疊，安全群組仍會留下。

在 VPC 中執行堆疊

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以在 virtual private cloud (VPC) 中建立堆疊的執行個體，以控制使用者對其的存取權。例如，您可能不想讓使用者直接存取您堆疊的應用程式伺服器或資料庫，而要求所有公有流量都經由 Elastic Load Balancer 導引。

在 VPC 中執行堆疊的基本步驟如下：

1. 使用 Amazon VPC 主控台或 API 或 AWS CloudFormation 範本建立適當設定的 VPC。

2. 在建立堆疊時指定 VPC ID。
3. 在適當子網路中，啟動堆疊的執行個體。

下列簡要說明 VPC 在 AWS OpsWorks Stacks 中的運作方式。

Important

如果您使用 VPC 端點功能，請注意堆疊中的每個執行個體都必須能夠從 Amazon Simple Storage Service (Amazon S3) 完成下列動作：

- 安裝執行個體代理程式。
- 安裝 Ruby 這類資產。
- 上傳 Chef 執行日誌。
- 擷取堆疊命令。

若要啟用這些動作，您必須確保堆疊的執行個體可以存取下列符合堆疊區域的儲存貯體。否則，上述動作將會失敗。
對於 Chef 12 Linux 和 Chef 12.2 Windows，儲存貯體如下所示。

代理程式儲存貯體	資產儲存貯體	日誌儲存貯體	DNA 儲存貯體
• opsworks-instance-agent-sa-east-1 • opsworks-instance-agent-ap-south-1 • opsworks-instance-agent-ap-northeast-1 • opsworks-instance-agent-ap-northeast-2	• opsworks-instance-assets-us-east-2 • opsworks-instance-assets-us-east-1 • opsworks-instance-assets-ap-south-1 • opsworks-instance-assets-ap-northeast-1	• opsworks-us-east-2-log • opsworks-us-east-1-log • opsworks-ap-south-1-log • opsworks-ap-northeast-1-log • opsworks-ap-northeast-2-log • opsworks-ap-southeast-1-log • opsworks-ap-southeast-2-log	• opsworks-us-east-2-dna • opsworks-us-east-1-dna • opsworks-ap-south-1-dna • opsworks-ap-northeast-1-dna • opsworks-ap-northeast-2-dna • opsworks-ap-southeast-1-dna • opsworks-ap-southeast-2-dna

代理程式儲存貯體	資產儲存貯體	日誌儲存貯體	DNA 儲存貯體
• opsworks-instance-agent-ap-southeast-1 • opsworks-instance-agent-ap-southeast-2 • opsworks-instance-agent-ca-central-1 • opsworks-instance-agent-eu-central-1 • opsworks-instance-agent-eu-west-1 • opsworks-instance-agent-eu-west-2 • opsworks-instance-agent-eu-west-3 • opsworks-instance-agent-us-east-1 • opsworks-instance-agent-us-east-2 • opsworks-instance-agent-us-west-1	• opsworks-instance-assets-ap-northeast-2 • opsworks-instance-assets-ap-southeast-1 • opsworks-instance-assets-ap-southeast-2 • opsworks-instance-assets-ca-central-1 • opsworks-instance-assets-eu-central-1 • opsworks-instance-assets-eu-west-1 • opsworks-instance-assets-eu-west-2 • opsworks-instance-assets-eu-west-3 • opsworks-instance-assets-sa-east-1 • opsworks-instance-assets-us-west-1	• opsworks-ca-central-1-log • opsworks-eu-central-1-log • opsworks-eu-west-1-log • opsworks-eu-west-2-log • opsworks-eu-west-3-log • opsworks-sa-east-1-log • opsworks-us-west-1-log • opsworks-us-west-2-log	• opsworks-ca-central-1-dna • opsworks-eu-central-1-dna • opsworks-eu-west-1-dna • opsworks-eu-west-2-dna • opsworks-eu-west-3-dna • opsworks-sa-east-1-dna • opsworks-us-west-1-dna • opsworks-us-west-2-dna

代理程式儲存貯體	資產儲存貯體	日誌儲存貯體	DNA 儲存貯體
• opsworks-instance-agent-us-west-2	• opsworks-instance-assets-us-west-2		

若是 Linux 的 Chef 11.10 和更早版本，儲存貯體如下所示。美國東部（維吉尼亞北部）區域以外的區域端點不支援 Chef 11.4 堆疊。

代理程式儲存貯體	資產儲存貯體	日誌儲存貯體	DNA 儲存貯體
• opsworks-instance-agent-us-east-2 • opsworks-instance-agent-us-east-1 • opsworks-instance-agent-ap-south-1 • opsworks-instance-agent-ap-northeast-1 • opsworks-instance-agent-ap-northeast-2 • opsworks-instance-agent-ap-southeast-1 • opsworks-instance-agent-ap-southeast-2 • opsworks-instance-agent-ca-central-1 • opsworks-instance-agent-eu-central-1 • opsworks-instance-agent-eu-west-1	• opsworks-instance-assets-us-east-2 • opsworks-instance-assets-us-east-1 • opsworks-instance-assets-ap-south-1 • opsworks-instance-assets-ap-northeast-1 • opsworks-instance-assets-ap-northeast-2 • opsworks-instance-assets-ap-southeast-1 • opsworks-instance-assets-ap-southeast-2 • opsworks-instance-assets-ca-central-1 • opsworks-instance-assets-eu-central-1 • opsworks-instance-assets-eu-west-1	• prod_stage-log	• prod_stage-dna

代理程式儲存貯體	資產儲存貯體	日誌儲存貯體	DNA 儲存貯體
• opsworks-instance-agent-eu-west-2 • opsworks-instance-agent-eu-west-3 • opsworks-instance-agent-us-east-1 • opsworks-instance-agent-us-west-1 • opsworks-instance-agent-us-west-2	• opsworks-instance-assets-eu-west-2 • opsworks-instance-assets-eu-west-3 • opsworks-instance-assets-sa-east-1 • opsworks-instance-assets-us-west-1 • opsworks-instance-assets-us-west-2		

如需詳細資訊，請參閱 VPC 端點。

> **Note**
>
> 若要讓 AWS OpsWorks Stacks 連線到您啟用的 VPC 端點，您也必須設定 NAT 或公有 IP 的路由，因為 AWS OpsWorks Stacks 代理程式仍然需要存取公有端點。

主題

- VPC 基本概念
- 為 AWS OpsWorks Stacks Stack 建立 VPC

VPC 基本概念

如需 VPC 的詳細討論，請參閱 Amazon Virtual Private Cloud。簡短來說，VPC 包含一或多個「子網路」，而其中每個都包含一或多個執行個體。每個子網路都有一個相關聯的路由表，以根據其目標 IP 地址指示傳出流量。

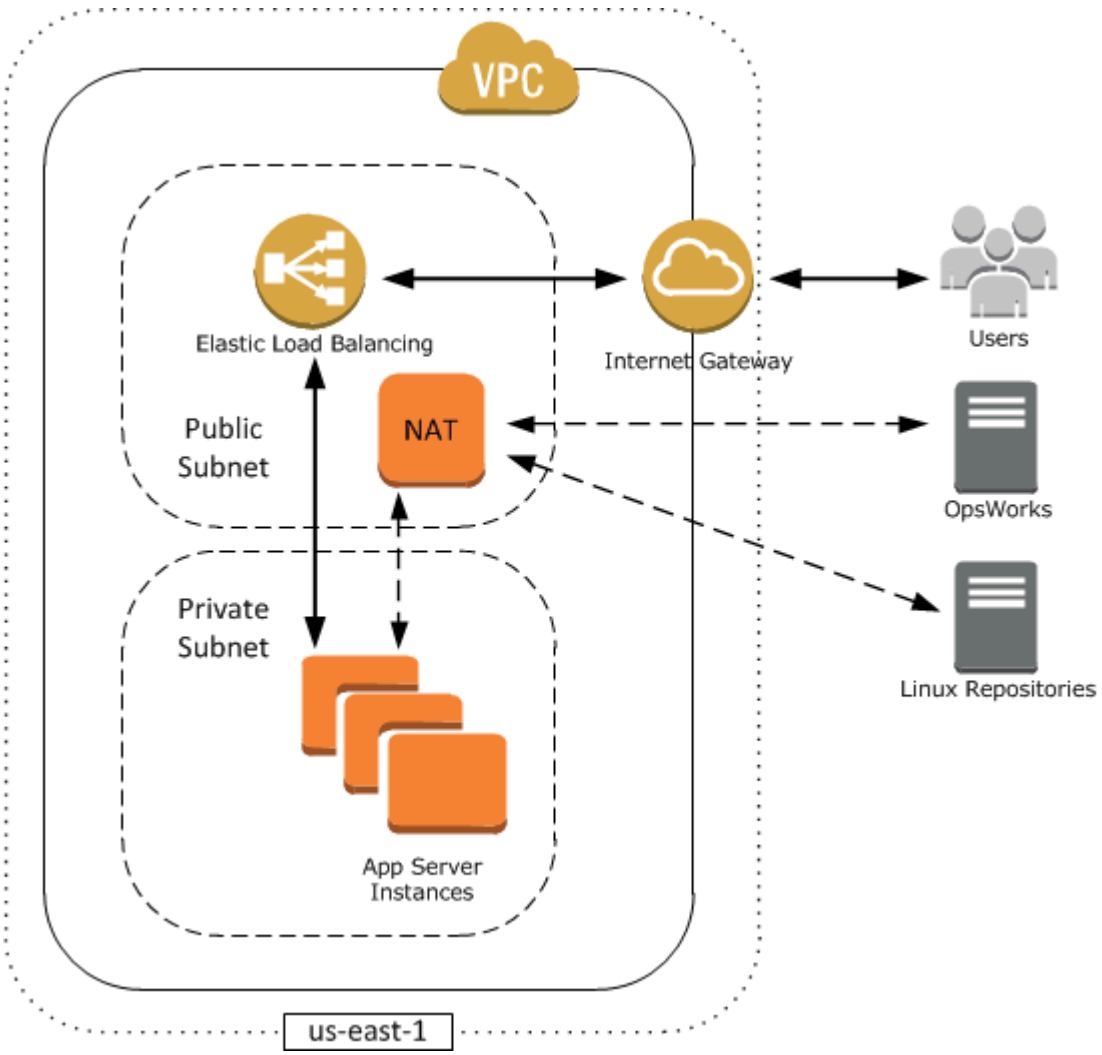
- VPC 中的執行個體預設為可互相通訊，而不管它們的子網路為何。不過，變更網路存取控制清單 (ACL)、安全群組政策，或使用靜態 IP 地址的話，可能會切斷此通訊。
- 如果子網路的執行個體可以與網際網路通訊，則這種子網路稱為「公有子網路」。
- 如果子網路的執行個體只能與 VPC 中的其他執行個體通訊，而無法直接與網際網路通訊，則這種子網路稱為「私有子網路」。

AWS OpsWorks Stacks 需要設定 VPC，以便堆疊中的每個執行個體，包括私有子網路中的執行個體，都能存取下列端點：

- 列於「區域支援」一節的其中一個 AWS OpsWorks Stacks 服務端點Stacks AWS OpsWorks 入門。
- 下列其中一個執行個體服務端點，由 Stacks AWS OpsWorks 代理程式使用。受管的客戶執行個體上執行的代理程式會與服務交換資料。
  - opsworks-instance-service.us-east-2.amazonaws.com
  - opsworks-instance-service.us-east-1.amazonaws.com
  - opsworks-instance-service.us-west-1.amazonaws.com
  - opsworks-instance-service.us-west-2.amazonaws.com
  - opsworks-instance-service.ap-south-1.amazonaws.com
  - opsworks-instance-service.ap-northeast-1.amazonaws.com
  - opsworks-instance-service.ap-northeast-2.amazonaws.com
  - opsworks-instance-service.ap-southeast-1.amazonaws.com
  - opsworks-instance-service.ap-southeast-2.amazonaws.com
  - opsworks-instance-service.ca-central-1.amazonaws.com
  - opsworks-instance-service.eu-central-1.amazonaws.com
  - opsworks-instance-service.eu-west-1.amazonaws.com
  - opsworks-instance-service.eu-west-2.amazonaws.com
  - opsworks-instance-service.eu-west-3.amazonaws.com
- Amazon S3

- 作業系統仰賴的任何套件儲存庫，例如 Amazon Linux 或 Ubuntu Linux 儲存庫。
- 您的應用程式和自訂技術指南儲存庫。

有多種方法可以將 VPC 設定為提供此連線能力。以下是如何為 Stacks 應用程式伺服器堆疊設定 VPC AWS OpsWorks 的簡單範例。



此 VPC 有多個元件：

子網路

VPC 有兩個子網路：一個公有子網路和一個私有子網路。

- 公有子網路包含一個負載平衡器和網路地址轉譯 (NAT) 裝置，其可與外部地址和私有子網路中的執行個體通訊。
- 私有子網路包含應用程式伺服器，其可與公有子網路中的 NAT 和負載平衡器通訊，但無法直接與外部地址通訊。

網際網路閘道

網際網路閘道允許含公有 IP 地址的執行個體 (例如負載平衡器) 與 VPC 外部地址通訊。

負載平衡器

Elastic Load Balancing 負載平衡器會接收來自使用者的傳入流量，將它分發至私有子網路中的應用程式伺服器，並將回應傳回給使用者。

NAT

(NAT) 裝置提供的應用程式伺服器只能對網際網路進行有限的存取；通常是用來從外部儲存庫下載軟體更新。所有 AWS OpsWorks Stacks 執行個體都必須能夠與 AWS OpsWorks Stacks 和適當的 Linux 儲存庫進行通訊。其中一種處理此問題的方法是，將含相關聯彈性 IP 地址的 NAT 裝置放在公有子網路中。然後，您就可以將來自私有子網路執行個體的傳出流量透過 NAT 進行路由。

> **Note**
>
> 單一 NAT 執行個體會在私有子網路的傳出流量中建立單一故障點。您可以使用一對 NAT 執行個體來設定 VPC，以在其中一個執行個體發生故障時由另一個接替，藉此提升可靠性。如需詳細資訊，請參閱 Amazon VPC NAT 執行個體的高可用性。您也可以使用 NAT 閘道。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 NAT。

最佳 VPC 組態取決於您的 Stacks AWS OpsWorks 堆疊。下列的一些範例說明您可能會使用特定 VPC 組態的時機。如需其他 VPC 案例的範例，請參閱 Amazon VPC 使用情境。

使用公有子網路中的一個執行個體

如果您有沒有相關聯私有資源的單一執行個體堆疊，例如不應公開存取的 Amazon RDS 執行個體，您可以建立一個具有一個公有子網路的 VPC，並將執行個體放在該子網路中。如果您不是使用預設 VPC，則必須讓執行個體 layer 指派彈性 IP 地址給該執行個體。如需詳細資訊，請參閱OpsWorks Layer 基本概念。

使用私有資源

如果您的資源不應公開存取，您可以建立一個 VPC，其中含有一個公有子網路和一個私有子網路。例如，在負載平衡的自動擴展環境中，您可以將所有 Amazon EC2 執行個體放在私有子網路中，並將負載平衡器放在公有子網路中。如此一來，就無法直接從網際網路存取 Amazon EC2 執行個體；所有傳入流量都必須透過負載平衡器路由。

私有子網路會將執行個體與 Amazon EC2 直接使用者存取隔離，但仍必須將傳出請求傳送至 AWS 和適當的 Linux 套件儲存庫。若要允許這類請求，您可以使用自帶彈性 IP 地址的網路地址轉譯 (NAT) 裝置，然後透過 NAT 路由該執行個體的傳出流量。您可以將 NAT 放在負載平衡器的相同公有子網路中，如上述範例所示。

- 如果您使用的是後端資料庫，例如 Amazon RDS 執行個體，您可以將這些執行個體放在私有子網路中。對於 Amazon RDS 執行個體，您必須在不同的可用區域中指定至少兩個不同的子網路。
- 如果您需要直接存取私有子網路中的執行個體，例如，您想要使用 SSH 登入執行個體，您可以將堡壘主機放在代理網際網路請求的公有子網路中。

將自己的網路擴展至 AWS

若您想要將自己的網路擴展至雲端，並直接從 VPC 存取網際網路，您可以建立 VPN 閘道。如需詳細資訊，請參閱案例 3：具有公有子網路和私有子網路以及硬體 VPN 存取的 VPC。

為 AWS OpsWorks Stacks Stack 建立 VPC

本節說明如何使用範例 AWS CloudFormation 範本建立 AWS OpsWorks Stacks 堆疊的 VPC。您可以在 OpsWorksVPCtemplates.zip 檔案中下載範本。如需如何手動建立一個本主題所述 VPC 的詳細資訊，請參閱案例 2：具有公有子網路和私有子網路的 VPC。如需如何設定路由表、安全群組等，請參閱範本範例。

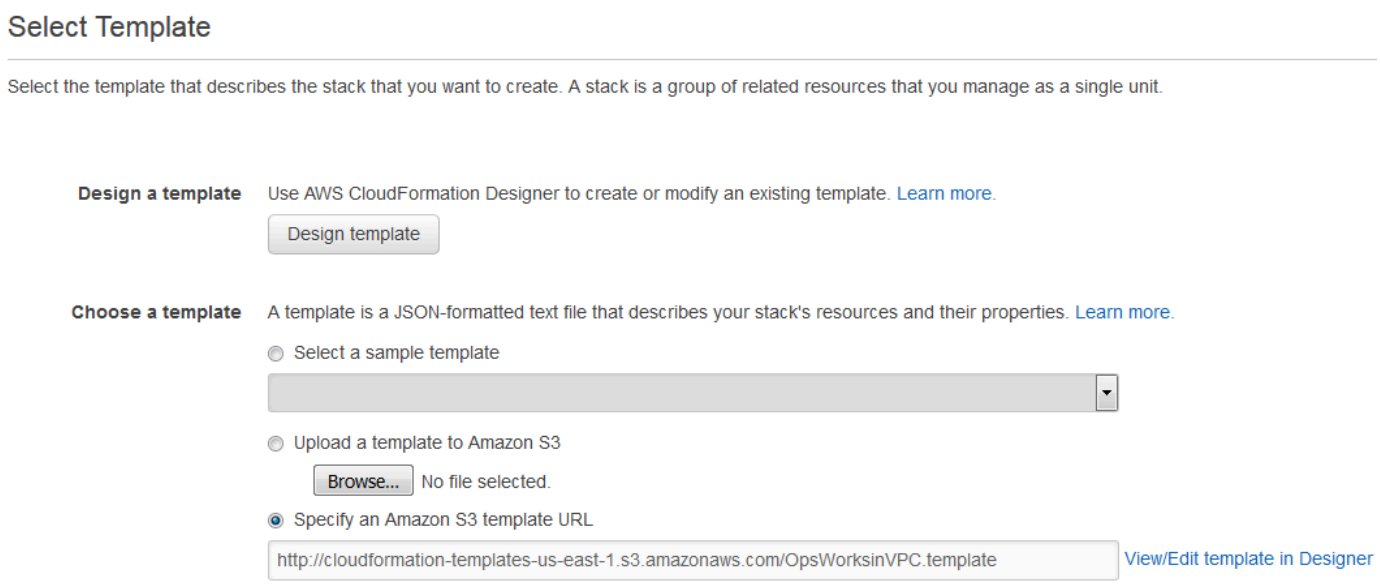
> **Note**
>
> 根據預設， AWS OpsWorks Stacks 會透過串連其 CIDR 範圍和可用區域來顯示子網路名稱，例如 `10.0.0.1/24 - us-east-1b`。若要讓名稱更易讀，請為每個子網路建立標籤，並將金鑰設定為 **`Name`** ，並將值設定為子網路名稱。然後， AWS OpsWorks Stacks 會將子網路名稱附加至預設名稱。例如，下列範例中的私有子網路，其標籤的 Name (名稱) 設定為 **`Private`**，因此 OpsWorks 會顯示為 `10.0.0.1/24 us-east - 1b - Private`。

您可以使用 AWS CloudFormation 主控台啟動 VPC 範本，只需幾個步驟。下列程序使用範例範本在美國東部（維吉尼亞北部）區域中建立 VPC。如需如何使用範本在其他區域中建立 VPC 的指示說明，請參閱程序之後的備註。

若要建立 VPC

1. 開啟 AWS CloudFormation 主控台，選取 US East (N. Virginia) (美國東部 (維吉尼亞北部)) 區域，然後選擇 Create Stack (建立堆疊)。

2. 在 Select Template (選取範本) 頁面上，選取 Upload a template (上傳範本)。瀏覽您在 OpsWorksVPCtemplates.zip `OpsWorksinVPC.template` 檔案中下載的檔案。選擇繼續。 OpsWorksVPCtemplates.zip



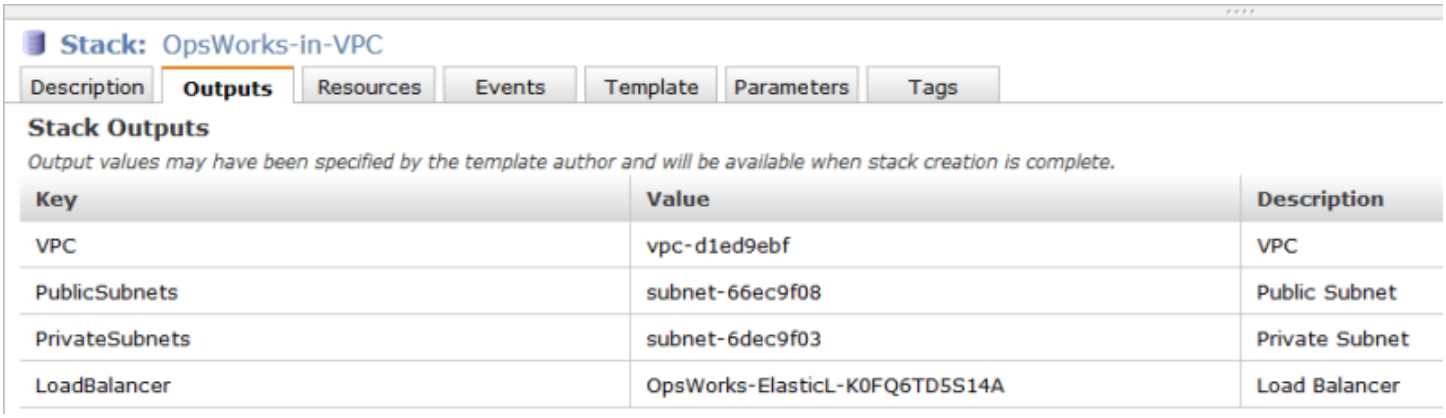
   您也可以開啟 AWS CloudFormation 範本、尋找 AWS OpsWorks Stacks VPC 範本，然後選擇啟動堆疊來啟動此堆疊。

3. 接受 Specify Parameters (指定參數) 頁面中的預設值，然後選擇 Continue (繼續)。
4. 在 Add Tags (新增標籤) 頁面中，將 Key (索引鍵) 設定為 **`Name`**，將 Value (值) 設定為 VPC 名稱，以建立標籤。此標籤可讓您在建立 Stacks AWS OpsWorks 堆疊時更輕鬆地識別 VPC。
5. 選擇 Continue (繼續)，然後選擇 Close (關閉) 以啟動堆疊。

備註：您可以使用下列任一種方法，在其他區域中建立 VPC。

- 移至在不同區域中使用範本，選擇適當的區域，找到 AWS OpsWorks Stacks VPC 範本，然後選擇啟動堆疊。
- 將範本檔案複製到您的系統中，並在 AWS CloudFormation 主控台中選取適當的區域，然後使用 Create Stack (建立堆疊) 精靈的 Upload a template to Amazon S3 (上傳範本到 Amazon S3) 選項，從您的系統上傳範本。

範例範本包含輸出，提供建立 Stacks AWS OpsWorks 堆疊所需的 VPC、子網路和負載平衡器 IDs。您可以選擇 AWS CloudFormation 主控台視窗底部的輸出索引標籤來查看它們。



更新堆疊

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

建立好堆疊之後，您即可隨時更新組態。在 Stack (堆疊) 頁面中，按一下 Stack Settings (堆疊設定)，然後按一下 Edit (編輯)，以顯示 Settings (設定) 頁面。視需要進行變更，然後按一下 Save (儲存)。

這些設定與建立新的堆疊所述的設定相同。如需詳細資訊，請參閱該主題。但是，請注意以下內容：

- 您不能修改區域或 VPC ID。
- 如果您的堆疊是在 VPC 中執行，則設定含有 Default subnet (預設子網路) 設定，其會列出 VPC 的子網路。如果您的堆疊不是在 VPC 中執行，則設定會標記為 Default Availability Zones (預設可用區域)，並列出區域的可用區域。
- 您可以變更預設的作業系統，但您不能為 Windows 堆疊指定 Linux 作業系統，或為 Linux 堆疊指定 Windows 作業系統。
- 如果您變更任何預設的執行個體設定，例如 Hostname theme (主機名稱主題) 或 Default SSH key (預設 SSH 金鑰)，則新值只會套用至您建立的任何新執行個體，而不會套用至現有的執行個體。
- 變更名稱會變更主控台顯示的名稱；不會變更 Stacks AWS OpsWorks 用來識別堆疊的基礎簡短名稱。
- 在您將 Use OpsWorks security groups (使用 OpsWorks 安全群組) 從 Yes (是) 變更為 No (否) 之前，每個 layer 除了 layer 的內建安全群組之外，還必須至少有一個安全群組。如需詳細資訊，請參閱編輯 OpsWorks Layer 的組態。

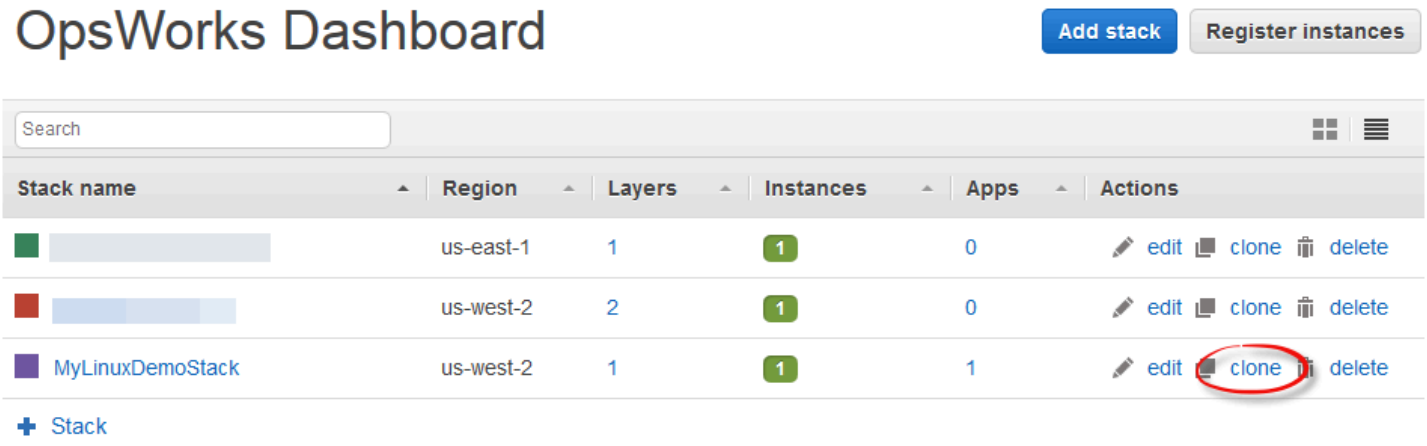
AWS OpsWorks Stacks 接著會從每一層刪除內建的安全群組。

- 如果您將 Use OpsWorks 安全群組從否變更為是， AWS OpsWorks Stacks 會將適當的內建安全群組新增至每個 layer，但不會刪除現有的安全群組。

複製堆疊

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

建立堆疊的多個副本，有時會很有幫助。例如，建議您將冗餘新增為災難復原或預防措施，或者可以使用現有堆疊做為新堆疊的起點。最簡單的方法是複製來源堆疊。在 Stacks AWS OpsWorks 儀表板上，在您要複製之堆疊列的動作欄中，選擇複製，這會開啟複製堆疊頁面。



複製堆疊的設定與來源堆疊相同，差別在「複製」這個字附加到堆疊名稱。如需這些設定的資訊，請參閱 建立新的堆疊。還有兩個額外的選用設定：

許可

如果 all permissions (所有許可) 已選取 (預設)，則來源堆疊許可會套用至複製堆疊。

應用程式

列出部署在來源堆疊上的應用程式。針對每個列出的應用程式，如果選取對應的核取方塊 (預設)，則會將應用程式部署到複製堆疊。

> **Note**
>
> 您無法將堆疊從一個區域端點複製到另一個區域端點；例如，您無法將堆疊從美國西部（奧勒岡）區域 (us-west-2) 複製到亞太區域（孟買）區域 (ap-south-1)。

當您完成設定時，請選擇複製堆疊。 AWS OpsWorks 堆疊會建立新的堆疊，其中包含來源堆疊的層，以及選擇性的應用程式和許可。這些 layer 與原始 layer 具有相同的組態，但您可以進行任何修改。不過，複製不會建立任何執行個體。您必須為複製堆疊的每個 layer 新增一組適當執行個體，然後將其啟動。如同使用任何堆疊，您可以在複製堆疊上執行一般管理任務，例如新增、刪除或修改 layer，或新增和部署應用程式。

若要讓複製的堆疊運作，請啟動執行個體。 AWS OpsWorks 堆疊會根據其層成員資格來設定每個執行個體。也會部署任何應用程式，如同使用新堆疊一樣。

執行 AWS OpsWorks Stacks Stack 命令

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 提供一組堆疊命令，您可以用來在堆疊的執行個體上執行各種操作。若要執行堆疊命令，請按一下堆疊頁面上的執行命令。您接著可選擇適當的命令，指定任何選項，然後按下位於右下方的按鈕 (會標上命令的名稱)。

> **Note**
>
> AWS OpsWorks Stacks 也支援一組部署命令，您可用來管理應用程式部署。如需詳細資訊，請參閱部署應用程式。

您可以在任何堆疊上執行下列堆疊命令。

Update Custom Cookbooks (更新自訂技術指南)

使用目前儲存庫的版本更新執行個體的自訂技術指南。此命令不會執行任何配方。若要執行更新後的配方，您可以使用 `Execute Recipes`、`Setup`，或 `Configure` 堆疊命令，或重新部署您的應用程式來執行部署配方。如需自訂技術指南的詳細資訊，請參閱技術指南和配方。

Execute Recipes (執行配方)

在執行個體上執行指定的一組配方。如需詳細資訊，請參閱手動執行配方。

設定

執行執行個體的安裝配方。

設定

執行執行個體的設定配方。

> **Note**
>
> 若要使用 Setup (安裝) 或 Configure (設定) 來在執行個體上執行配方，配方必須指派給執行個體 layer 的對應生命週期事件。如需詳細資訊，請參閱執行配方。

您僅能在 Linux 式堆疊上執行下列堆疊命令。

安裝相依性

安裝執行個體套件。從 Chef 12 開始，此命令無法使用。

Update Dependencies (更新依存項目)

（僅限 Linux。 從 Chef 12 開始，此命令無法使用。) 安裝一般作業系統更新及套件更新。詳細資訊取決於執行個體的作業系統。如需詳細資訊，請參閱管理安全性更新。

使用 Upgrade Operating System (升級作業系統) 命令，將執行個體升級至新的 Amazon Linux 版本。

Upgrade Operating System (升級作業系統)

(僅限 Linux) 將執行個體的 Amazon Linux 作業系統升級至最新版本。如需詳細資訊，請參閱AWS OpsWorks Stacks 作業系統。

> **Important**
>
> 在執行 Upgrade Operating System (升級作業系統) 之後，我們建議您也執行 Setup (安裝)。這可確保服務正確重新啟動。

堆疊命令具有下列選項，其中有些選項只會在特定命令內出現。

註解

(選擇性) 輸入任何您欲新增的自訂備註。

Recipes to execute (要執行的配方)

(必要項目) 此設定只會在您選取 Execute Recipes (執行配方) 命令時才會出現。使用標準 *cookbook_name*::*recipe_name* 格式，輸入要執行的配方，並以逗點分隔。如果您指定多個配方， AWS OpsWorks Stacks 會依列出的順序執行它們。

Allow reboot (允許重新開機)

(選擇性) 此設定只會在您選取 Upgrade Operating System (升級作業系統) 命令時才會出現。預設值為是，這會指示 AWS OpsWorks Stacks 在安裝升級之後重新啟動執行個體。

Custom Chef JSON (自訂 Chef JSON)

(選擇性) 選擇 Advanced (進階) 以顯示此選項，允許您指定要併入堆疊組態及部署屬性的自訂 JSON 屬性。

執行個體

(選擇性) 指定要執行命令的執行個體。根據預設，會選取所有線上執行個體。若要在一部分的執行個體上執行命令，請選取適當的 layer 或執行個體。

> **Note**
>
> 您可能會看到您並未執行的 execute_recipes 執行在 Deployment (部署) 和 Commands (命令) 頁面上列出。這通常是許可變更的結果，例如授予或移除使用者的 SSH 許可。當您進行這類變更時， AWS OpsWorks Stacks 會使用 execute_recipes 來更新執行個體的許可。

使用自訂 JSON

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

數個 AWS OpsWorks Stacks 動作可讓您指定自訂 JSON，而 AWS OpsWorks Stacks 會安裝在執行個體上，並且可供配方使用。

您可以在下列情況中指定自訂的 JSON：

- 當您建立、更新或複製堆疊時。

  AWS OpsWorks Stacks 會為所有後續生命週期事件在所有執行個體上安裝自訂 JSON。
- 當您執行部署或堆疊命令時。

  AWS OpsWorks Stacks 只會針對該事件將自訂 JSON 傳遞給執行個體。

自訂 JSON 必須以有效的 JSON 物件表示，且格式化為有效的 JSON 物件。例如：

```
{
  "att1": "value1",
  "att2": "value2"
  ...
}
```

AWS OpsWorks Stacks 會將自訂 JSON 存放在下列位置：

Linux 執行個體為：

- `/var/chef/runs/run-ID/attribs.json`
- `/var/chef/runs/run-ID/nodes/hostname.json`

Windows 執行個體為：

- *drive*:\chef\runs*run-ID*\attribs.json
- *drive*:\chef\runs*run-ID*\nodes*hostname*.json

> **Note**
>
> 適用於 Linux 的 Chef 11.10 和舊版中，自訂 JSON 位於 Linux 執行個體的下列路徑，Windows 執行個體不可用，而且沒有 attribs.json 檔案。日誌存放在和 JSON 相同的資料夾或目錄中。如需適用於 Linux 之 Chef 11.10 和舊版中自訂 JSON 的詳細資訊，請參閱使用自訂的 JSON 覆寫屬性和 Chef 日誌檔。
> /var/lib/aws/opsworks/chef/*hostname*.json

在上述的路徑中，*run-ID* 是 AWS OpsWorks Stacks 指派給在執行個體上執行之每個 Chef 的唯一 ID，而 *hostname* 則是執行個體的主機名稱。

若要從 Chef 配方存取自訂的 JSON，請使用標準的 Chef node 語法。

例如，假設您想要為要部署的應用程式定義簡單的設定；例如，是否一開始即顯示應用程式，以及應用程式的初始前景和背景顏色。假設您使用 JSON 物件定義這些應用程式設定，如下所示：

```
{
  "state": "visible",
  "colors": {
    "foreground": "light-blue",
    "background": "dark-gray"
  }
}
```

為堆疊宣告自訂的 JSON：

1. 在堆疊頁面中，選擇 Stack Settings (堆疊設定)，然後選擇 Edit (編輯)。
2. 針對 Custom Chef JSON (自訂 Chef JSON)，輸入 JSON 物件，然後選擇 Save (儲存)。

> **Note**
>
> 您可以在部署、layer 和堆疊層級宣告自訂 JSON。如果您只想向個別的部署或 layer 顯示一些自訂的 JSON，建議您執行此作業。或者，例如，您可能希望使用在 layer 層級宣告的自訂 JSON 暫時覆寫在堆疊層級宣告的自訂 JSON。如果您在多個層級宣告自訂 JSON，則在部署

層級宣告的自訂 JSON，會覆寫在 layer 和堆疊層級宣告的任何自訂 JSON。在 layer 層級宣告的自訂 JSON，將會覆寫僅在堆疊層級宣告的任何自訂 JSON。
若要使用 AWS OpsWorks Stacks 主控台指定部署的自訂 JSON，請在部署應用程式頁面上選擇進階。在 Custom Chef JSON (自訂 Chef JSON) 方塊中輸入自訂的 JSON，然後選擇 Save (儲存)。
若要使用 AWS OpsWorks Stacks 主控台為圖層指定自訂 JSON，請在圖層頁面上，選擇所需圖層的設定。在 Custom JSON (自訂 JSON) 方塊中輸入自訂的 JSON，然後選擇 Save (儲存)。
如需詳細資訊，請參閱編輯 OpsWorks Layer 的組態及部署應用程式。

當您執行部署或堆疊命令時，配方會使用標準的 Chef node 語法擷取這些自訂值，直接將它們映射在自訂 JSON 物件的階層。例如，下列配方程式碼會將有關先前自訂 JSON 值的訊息寫入 Chef 日誌：

```
Chef::Log.info("********** The app's initial state is '#{node['state']}' **********")
Chef::Log.info("********** The app's initial foreground color is '#{node['colors']
['foreground']}' **********")
Chef::Log.info("********** The app's initial background color is '#{node['colors']
['background']}' **********")
```

此方法有助於將資料傳遞至配方。 AWS OpsWorks Stacks 會將該資料新增至執行個體，而配方可以使用標準 Chef node 語法擷取資料。

> **Note**
>
> 自訂 JSON 的大小限制為 120 KB。如果您需要更多容量，建議您將部分資料儲存在 Amazon Simple Storage Service (Amazon S3) 上。然後，您的自訂配方可以使用 AWS CLI 或 適用於 Ruby 的 AWS SDK，將資料從 Amazon S3 儲存貯體下載到您的執行個體。

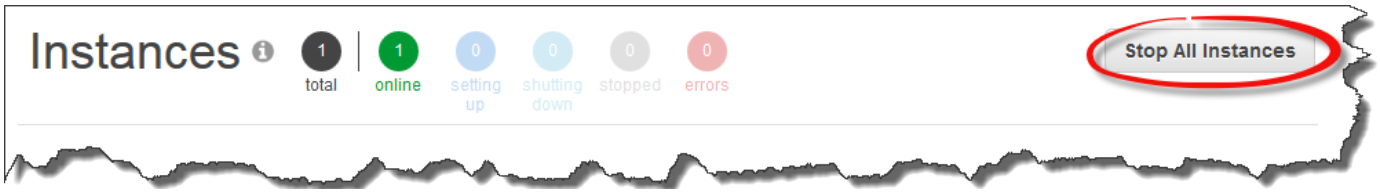
刪除堆疊

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

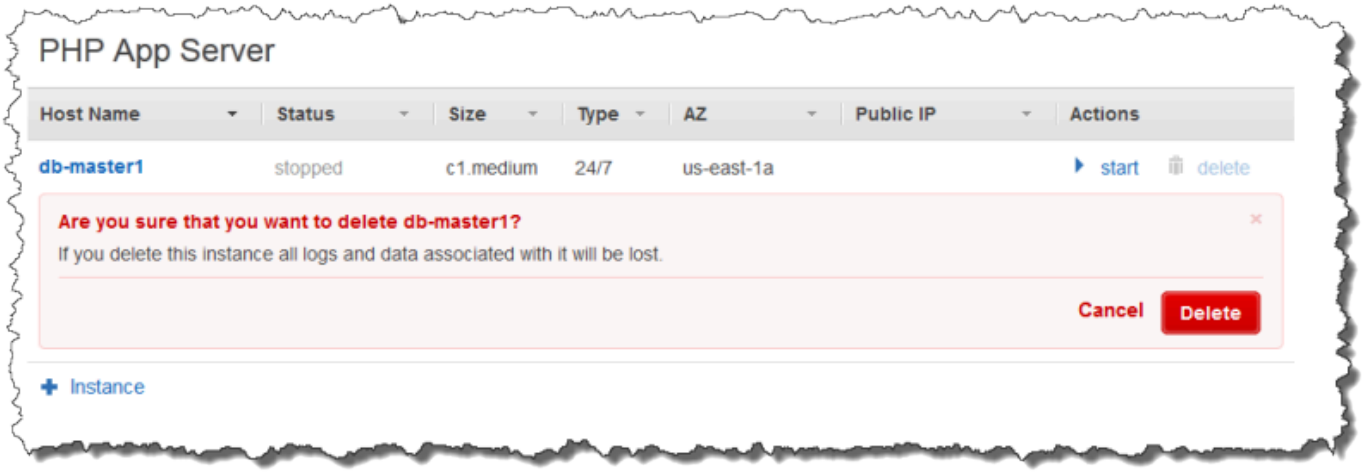
您可以刪除不再需要的堆疊。只能刪除空的堆疊，您必須先刪除堆疊中所有執行個體、應用程式和 Layer。

刪除 堆疊

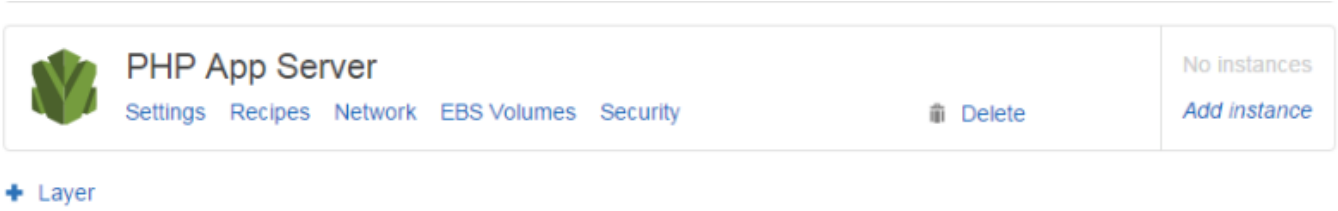
1. 在 AWS OpsWorks Stacks 儀表板上，選擇您要關閉和刪除的堆疊。
2. 在導覽窗格中，選擇執行個體。
3. 在 Instances (執行個體) 頁面上，選擇 Stop all Instances (停止所有執行個體)。



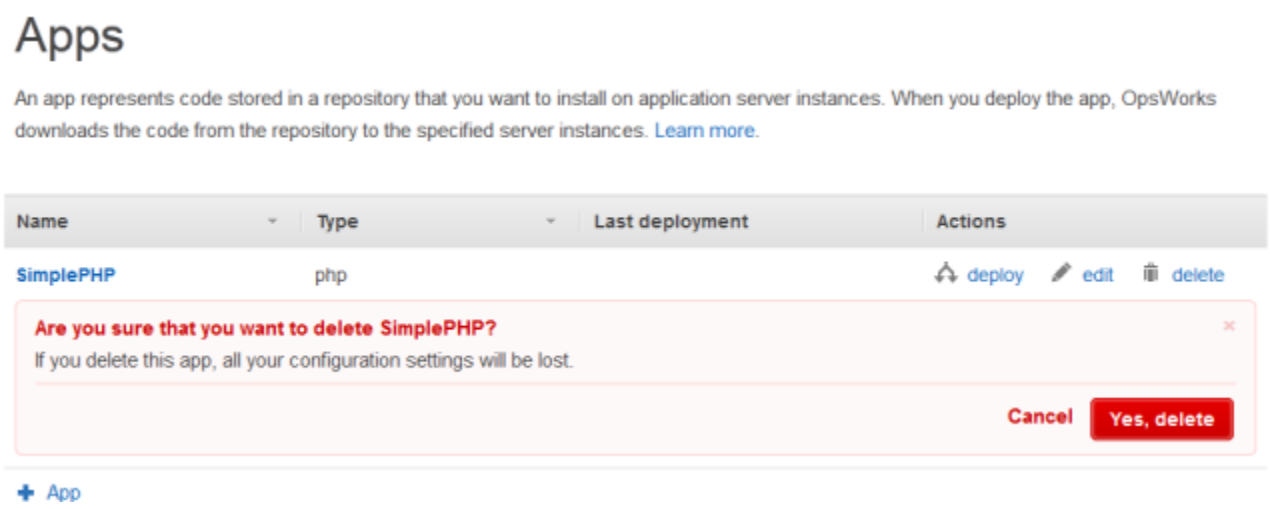
4. 執行個體停止後，針對 layer 中的每個執行個體，在動作欄中選擇刪除。出現確認提示時，請選擇 Yes, Delete (是，刪除)。



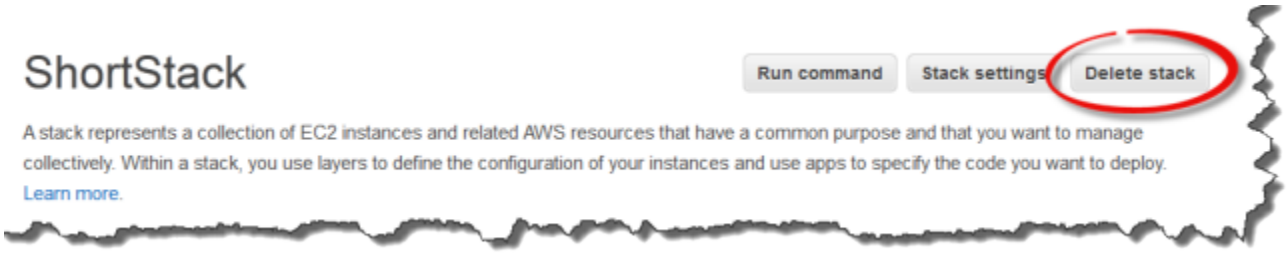
5. 刪除所有執行個體後，在導覽窗格中，選擇 Layers (Layer)。
6. 在 Layers (Layer) 頁面上，為堆疊中的每個 layer，選擇 delete (刪除)。在 (刪除堆疊) 確認提示上，選擇 Yes, Delete (是的，請刪除)。



7. 刪除所有 layer 後，在導覽窗格中，選擇 Apps (應用程式)。
8. 在應用程式頁面上，針對堆疊中的每個應用程式，選擇動作欄中的刪除。在 (刪除堆疊) 確認提示上，選擇 Yes, Delete (是的，請刪除)。



9. 刪除所有應用程式後，在導覽窗格中，選擇 Stack (堆疊)。
10. 在 stack (堆疊) 頁面上，選擇 Delete stack (刪除堆疊)。在 (刪除堆疊) 確認提示上，選擇 Yes, Delete (是的，請刪除)。



刪除堆疊使用的其他 AWS 資源

您可以使用其他 AWS 資源搭配 AWS OpsWorks Stacks 來建立和管理堆疊。當您刪除堆疊時，如果另一個堆疊未使用，且 Stacks AWS OpsWorks 以外的資源未使用，請考慮也刪除使用堆疊的資源。以下是清理與堆疊搭配使用之外部 AWS 資源的建議原因。

- 外部 AWS 資源可以繼續在您的 AWS 帳戶中產生費用。
- Amazon S3 儲存貯體等資源可能包含個人識別、敏感或機密資訊。

> **Important**
>
> 請勿刪除其他的堆疊正在使用的資源。請注意，IAM 角色和安全群組是全域的，因此其他區域中的堆疊可能會使用這些相同的資源。

以下是堆疊使用的其他 AWS 資源，以及有關如何刪除這些資源的資訊連結。

服務角色和執行個體描述檔

當您建立堆疊時，您可以指定 IAM 角色和執行個體描述檔，讓 AWS OpsWorks Stacks 代表您建立允許的資源。 AWS OpsWorks 如果您未選擇現有的角色和執行個體描述檔， 會為您建立角色和執行個體描述檔。為您 AWS OpsWorks 建立的角色和執行個體描述檔`aws-opsworks-ec2-role`分別名為 `aws-opsworks-service-role`和 。如果您帳戶中沒有其他堆疊正在使用 IAM 角色和執行個體描述檔，則可以安全地刪除這些資源。如需有關如何刪除 IAM 角色和執行個體描述檔的資訊，請參閱《IAM 使用者指南》中的刪除角色或執行個體描述檔。

安全群組

在 AWS OpsWorks Stacks 中，您可以在 layer 層級指定使用者定義的安全群組。您可以使用 Amazon EC2 主控台或 API 建立安全群組。堆疊和 layer 可以在其他區域中使用相同的安全群組，因為安全群組是全域通用。如果其他 AWS 資源未使用安全群組，您可以將其刪除。如需如何刪除安全群組的詳細資訊，請參閱《Amazon EC2 使用者指南》中的刪除安全群組。

Amazon EBS 磁碟區

在 AWS OpsWorks Stacks 中，您可以在 layer 層級新增 EBS 磁碟區，並將它們連接到 layer 中的執行個體。您可以使用 Amazon EC2 服務主控台或 API 建立 EBS 磁碟區，然後將它們連接到 layer 層級的 AWS OpsWorks Stacks 執行個體。EBS 磁碟區專屬的可用區域。如果您不再使用特定區域和可用區域中堆疊裡的 EBS 磁碟區，您可以刪除。如需如何刪除 Amazon EBS 磁碟區的詳細資訊，請參閱《Amazon Amazon EC2刪除 Amazon EBS 磁碟區。

Amazon Simple Storage Service (Amazon S3) 儲存貯體

在 AWS OpsWorks Stacks 中，您可以針對下列項目使用 Amazon S3 儲存貯體。傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

- 儲存應用程式程式碼
- 儲存技術指南和配方
- 如果您已在 CloudTrail Stacks 中啟用 CloudTrail 記錄，CloudTrail AWS OpsWorks 日誌
- 如果您已在 Stacks 中啟用 Amazon CloudWatch Logs AWS OpsWorks 串流

彈性 IP 位址

如果您已向 Stacks AWS OpsWorks 註冊彈性 IP 地址，且不再需要彈性 IP 地址，則可以釋出彈性 IP 地址。

Elastic Load Balancing 負載平衡器

如果您不再需要與堆疊中的層搭配使用的 Elastic Load Balancing 傳統負載平衡器，則可以將其刪除。如需詳細資訊，請參閱 Classic Load Balancers 使用者指南中的「刪除您的負載平衡器」。

Amazon Relational Database Service (Amazon RDS) 執行個體

如果您已向 Stacks AWS OpsWorks 註冊 Amazon RDS 資料庫 (DB) 執行個體，且不再需要這些執行個體，則可以刪除資料庫執行個體。如需如何刪除資料庫執行個體的詳細資訊，請參閱《Amazon RDS 使用者指南》中的刪除資料庫執行個體。

Amazon Elastic Container Service (Amazon ECS) 叢集

如果您的堆疊包含 ECS 叢集層，而且您不再使用以 layer 註冊的 ECS 叢集，您可以刪除 ECS 叢集。如需如何刪除 ECS 叢集的詳細資訊，請參閱《Amazon ECS 開發人員指南》中的刪除叢集。

層

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

每個堆疊各包含一或多個層，每個 layer 各代表一個堆疊元件，例如負載平衡器或一組應用程式伺服器。

當您使用 AWS OpsWorks Stacks layer 時，請記住下列事項：

- 堆疊中的每個 layer 必須至少具有一個執行個體，並可選擇性具有多個執行個體。
- 堆疊中的每個執行個體必須至少是一個 layer 的成員，但已註冊的執行個體除外。

  除了部分基本設定 (例如 SSH 金鑰和主機名稱) 以外，您無法直接設定執行個體。您必須建立和設定適當的 layer，並將執行個體新增至該 layer。

Amazon EC2 執行個體可以選擇性地成為多層的成員。在這種情況下， AWS OpsWorks Stacks 會執行配方，為每個執行個體的 layer 安裝和設定套件、部署應用程式等。

透過將執行個體指派至多個 layer，您可以執行諸如以下動作：

- 透過在單一執行個體上託管資料庫伺服器和負載平衡器來降低費用。
- 使用其中一個應用程式伺服器進行管理。

  建立自訂管理 layer，並將其中一個應用程式伺服器執行個體新增至 layer。管理 layer 的配方會設定該應用程式伺服器執行個體來執行管理任務，並安裝任何其他必要軟體。其他應用程式伺服器執行個體則僅是應用程式伺服器。

本節說明如何使用 layer。

主題

- OpsWorks Layer 基本概念
- Elastic Load Balancing Layer
- Amazon RDS 服務層
- ECS 叢集層
- 自訂 AWS OpsWorks Stacks Layers
- 個別 layer 作業系統套件安裝

OpsWorks Layer 基本概念

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節說明如何執行所有 Stacks AWS OpsWorks layer 通用的操作。

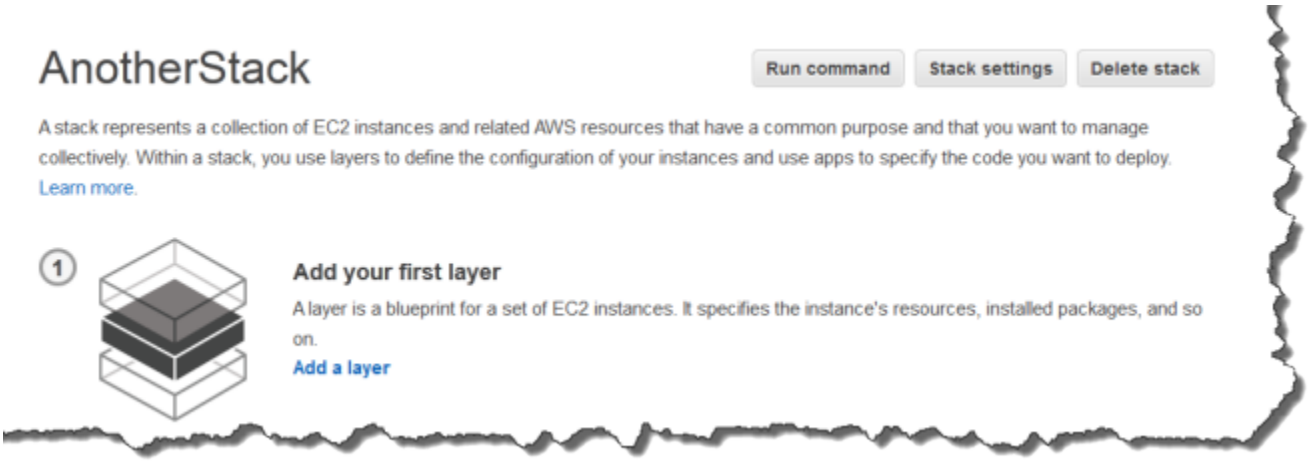
主題

- 建立 OpsWorks Layer
- 編輯 OpsWorks Layer 的組態
- 使用自動修復來替換故障的執行個體
- 刪除 OpsWorks Layer

建立 OpsWorks Layer

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

當您建立新的堆疊，您會看到以下頁面：



新增第一個 OpsWorks layer

1. 按一下 Add a Layer (新增 Layer)。
2. 在 Add Layer (新增 Layer) 頁面上，選取會顯示 layer 組態選項的適當 layer。
3. 適當設定該 layer，然後按一下 Add Layer (新增 Layer) 以新增到堆疊。下列各節說明如何設定各種 layer。

   > **Note**
   >
   > Add Layer (新增 Layer) 頁面只會顯示每個 layer 較常用的組態設定。您可以透過編輯 layer 來指定額外的設定。

4. 將執行個體新增至 layer 並啟動。

> **Note**
>
> 如果執行個體是多個 layer 的成員，您必須先將其新增到所屬的所有 layer，才能啟動該執行個體。您不能將線上執行個體新增到 layer。

若要新增更多 layer，請開啟 Layers (Layer) 頁面，然後按一下 +Layer 以開啟 Add Layer (新增 Layer) 頁面。

當您啟動執行個體時， AWS OpsWorks Stacks 會自動為每個執行個體的 layer 執行設定和部署配方，以安裝和設定適當的套件，並部署適當的應用程式。您可以透過各種方式自訂 layer 的設定和組態程序，例如將自訂配方指派給適當的生命週期事件。 AWS OpsWorks Stacks 會在每個事件的標準配方之後執行自訂配方。如需詳細資訊，請參閱技術指南和配方。

下列圖層特定章節說明 如何處理各種 Stacks 圖層的步驟 2 AWS OpsWorks 和 3。如需如何新增執行個體的詳細資訊，請參閱將執行個體新增至 Layer。

編輯 OpsWorks Layer 的組態

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

在您建立 layer 後，某些屬性 (例如 AWS 區域) 是不可變的，但您可以變更大多數的 layer 設定。編輯 layer 也可以存取 Add Layer (新增 Layer) 頁面上不可用的組態設定。儲存新組態後，設定將立即生效。

編輯 OpsWorks layer

1. 在導覽窗格中，按一下 Layers (Layer)。
2. 在 Layers (Layer) 頁面上，選擇 layer 名稱以開啟詳細資訊頁面，會顯示目前的組態。

> **Note**
>
> 選擇 layer 名稱下的其中一個名稱，會直接帶您到詳細資訊頁面上的相關聯標籤。

3. 按一下 Edit (編輯)，然後選取適當的標籤：General Settings (一般設定)、Recipes (配方)、Network (網路)、EBS Volumes (EBS 磁碟區) 或 Security (安全性)。

下列各節說明可用於所有 layer 之各種標籤上的設定。某些 layer 具有額外 layer 特定的設定，這些設定會顯示在頁面頂端。此外，某些設定僅適用於 Linux 系統的堆疊，如上所述。

主題

- 一般設定
- 配方
- 網路
- EBS 磁碟區
- 安全
- CloudWatch Logs
- 標籤

一般設定

所有 layer 都具有以下設定：

已啟用自動修復

Layer 的執行個體是否已啟用自動修復。預設設定為 Yes (是)。

自訂 JSON

JSON 格式資料，傳遞給此 layer 中所有執行個體的 Chef 配方。例如，您可以使用此配方將資料傳遞至您自己的配方。如需詳細資訊，請參閱使用自訂 JSON。

> **Note**
>
> 您可以在部署、layer 和堆疊層級宣告自訂 JSON。如果您希望在堆疊中顯示某些自訂 JSON 或僅在個別的部署中顯示，則建議您執行此作業。或者，例如，您可能希望使用在部署層級宣告的自訂 JSON，暫時覆寫在 layer 層級宣告的自訂 JSON。如果您在多個層級宣告自訂 JSON，則在部署層級宣告的自訂 JSON，會覆寫在 layer 和堆疊層級宣告的任何自訂 JSON。在 layer 層級宣告的自訂 JSON，將會覆寫僅在堆疊層級宣告的任何自訂 JSON。

若要使用 AWS OpsWorks Stacks 主控台指定部署的自訂 JSON，請在部署應用程式頁面上選擇進階。在 Custom Chef JSON (自訂 Chef JSON) 方塊中輸入自訂的 JSON，然後選擇 Save (儲存)。
若要使用 AWS OpsWorks Stacks 主控台指定堆疊的自訂 JSON，請在堆疊設定頁面上，在自訂 JSON 方塊中輸入自訂 JSON，然後選擇儲存。
如需詳細資訊，請參閱使用自訂 JSON及部署應用程式。

執行個體關機逾時

指定觸發關機生命週期事件後， AWS OpsWorks Stacks 在停止或終止 Amazon EC2 執行個體之前的等待時間 （以秒為單位）。預設設定為 120 秒。設定的目的是在終止執行個體之前，為執行個體的關機配方提供足夠時間來完成其任務。如果您的自訂關機可能會需要更多時間，則請視情況修改設定。如需執行個體關機的詳細資訊，請參閱停止執行個體。

此標籤上其餘的設定會因 layer 類型而異，並與 layer 之 Add Layer (新增 Layer) 頁面上的設定相同。

配方

Recipes (配方) 標籤包含下列設定。

自訂 Chef 配方

您可以將自訂 Chef 配方指派給 layer 的生命週期事件。如需詳細資訊，請參閱執行配方。

網路

Network (網路) 標籤包含下列設定。

Elastic Load Balancing

您可以將 Elastic Load Balancing 負載平衡器連接至任何 layer。 AWS OpsWorks Stacks 接著會自動向負載平衡器註冊 layer 的線上執行個體，並在它們離線時取消註冊。如果您已啟用負載平衡器的連線耗盡功能，您可以指定 Stacks AWS OpsWorks 是否支援它。如需詳細資訊，請參閱Elastic Load Balancing Layer。

自動指派 IP 地址

您可以控制 AWS OpsWorks Stacks 是否自動將公有或彈性 IP 地址指派給 layer 的執行個體。以下是當您啟用此選項時會發生的情況：

- 對於執行個體後端執行個體， AWS OpsWorks Stacks 會在每次執行個體啟動時自動指派地址。
- 對於 Amazon EBS 後端執行個體，當執行個體第一次啟動時， AWS OpsWorks Stacks 會自動指派地址。
- 如果執行個體屬於多個 layer，且您已為至少一個 layer 啟用自動指派， AWS OpsWorks 則 Stacks 會自動指派地址。

> **Note**
>
> 如果您啟用公有 IP 地址的自動指派，它僅適用於新的執行個體。 AWS OpsWorks Stacks 無法更新現有執行個體的公有 IP 地址。

如果您的堆疊在 VPC 中執行，則您可以對公有 IP 地址和彈性 IP 地址進行個別設定。下表說明這些互動方式：

		Public IP addresses: Yes	Public IP addresses: No
Elastic IP addresses	Yes	Instances receive an Elastic IP address when they are started for the first time, or a public IP address if an Elastic IP cannot be assigned.	Instances receive an Elastic IP address when they are started for the first time.
	No	Instances receive a public IP address each time they are started.	Instances receive only a private IP address, which is not accessible from outside the VPC.

> **Note**
>
> 執行個體必須能夠與 AWS OpsWorks Stacks 服務、Linux 套件儲存庫和技術指南儲存庫進行通訊。如果未指定公有或彈性 IP 地址，則 VPC 必須包含允許 layer 的執行個體與外部網站通訊之元件 (如 NAT)。如需詳細資訊，請參閱在 VPC 中執行堆疊。

如果您的堆疊不是在 VPC 中執行，則 Elastic IP addresses (彈性 IP 地址) 是您唯一的設定：

- Yes (是)：執行個體在第一次啟動時會收到彈性 IP 地址，而如果無法指派彈性 IP 地址則會收到公有 IP 地址。
- No (否)：執行個體在每次啟動時都會收到公有 IP 地址。

EBS 磁碟區

EBS Volumes (EBS 磁碟區) 標籤包含下列設定。

EBS 最佳化執行個體

是否應為 Amazon Elastic Block Store (Amazon EBS) 最佳化 layer 的執行個體。如需詳細資訊，請參閱 Amazon EBS 最佳化執行個體。

額外的 EBS 磁碟區

（僅限 Linux) 您可以將 Amazon EBS 磁碟區新增至 layer 執行個體或從中移除它們。當您啟動執行個體時， AWS OpsWorks Stacks 會自動建立磁碟區並將其連接至執行個體。您可以使用 Resources (資源) 頁面來管理堆疊的 EBS 磁碟區。如需詳細資訊，請參閱資源管理。

- 掛載點 –（必要）指定要掛載 EBS 磁碟區的掛載點或目錄。
- # 磁碟 –（選用）如果您指定 RAID 陣列，則為陣列中的磁碟數量。

  每個 RAID 層級都有預設的磁碟數目，但您可以從清單中選取更大的數目。
- 大小總計 (GiB) –（必要）磁碟區的大小，以 GiB 為單位。

  對於 RAID 陣列，此設定會指定陣列的總大小，而不是每個磁碟的大小。

  下表說明每種磁碟區類型允許的磁碟區大小下限和上限。

磁碟區類型	大小下限 (GiB)	大小上限 (GiB)
磁帶	1	1024
佈建 IOPS (SSD)	4	16384
一般用途 (SSD)	1	16384
輸送量最佳化 (HDD)	500	16384
Cold HDD	500	16384

- 磁碟區類型 –（選用）指定要建立磁性、一般用途 SSD、輸送量最佳化 HDD、冷 HDD 或 PIOPS 磁碟區。

  預設值為 Magnetic (磁帶)。
- 加密 –（選用）指定是否加密 EBS 磁碟區的內容。

- 每個磁碟的 IOPS – （佈建 IOPS SSD 和一般用途 SSD 磁碟區需要） 如果您指定佈建 IOPS SSD 或一般用途 SSD 磁碟區，您也必須指定每個磁碟的 IOPS。

  在建立磁碟區時，您可以指定佈建 IOPS 磁碟區的 IOPS 速率。已佈建的 IOPS 和所請求的磁碟區大小之比例最高為 30 (換言之，一個 3000 IOPS 的磁碟區必須至少有 100 GB)。一般用途 (SSD) 磁碟區類型的基準 IOPS 為磁碟區大小乘以 3，上限為 10000 IOPS，並可突發最高 3000 IOPS 達 30 分鐘。

當您將磁碟區新增至 layer，或是從 layer 移除磁碟區時，請注意以下事項：

- 如果您新增磁碟區，每個新執行個體都會取得新的磁碟區，但 AWS OpsWorks Stacks 不會更新現有的執行個體。
- 如果您移除某個磁碟區，則僅會套用於新的執行個體；現有的執行個體會保留其磁碟區。

指定掛載點

您可以指定您喜歡的任何掛載點。不過，請注意，某些掛載點保留供 AWS OpsWorks Stacks 或 Amazon EC2 使用，不應用於 Amazon EBS 磁碟區。請勿使用典型的 Linux 系統資料夾，例如 /home 或 /etc。

下列掛載點保留供 Stacks AWS OpsWorks 使用。

- /srv/www
- /var/log/apache2 (Ubuntu)
- /var/log/httpd (Amazon Linux)
- /var/log/mysql
- /var/www

當執行個體啟動或重新啟動時，autofs (自動掛載協助程式) 會使用臨時裝置掛載點，例如 /media/ephemeral0 用於綁定掛載。此操作會在掛載 Amazon EBS 磁碟區之前進行。為了確保 Amazon EBS 磁碟區的掛載點不會與 autofs 衝突，請勿指定暫時性裝置掛載點。可能的暫時性裝置掛載點取決於特定的執行個體類型，以及它是執行個體後端還是 Amazon EBS 後端。為了避免與 autofs 衝突，請執行下列作業：

- 驗證您希望使用之特定執行個體類型與備用存放區的臨時裝置掛載點。

- 請注意，如果您切換到 Amazon EBS 支援的執行個體，則適用於執行個體後端執行個體的掛載點可能會與 autofs 衝突，反之亦然。

> **Note**
>
> 如果您想要變更執行個體存放區塊型設備映射，您可以建立自訂 AMI。如需詳細資訊，請參閱 Amazon EC2 執行個體存放區。如需如何為 Stacks 建立自訂 AMI AWS OpsWorks 的詳細資訊，請參閱 使用自訂 AMI。

下列範例說明如何使用自訂配方來確保磁碟區的掛載點不與 autofs 衝突。您可以依照特定使用案例來調整。

避免衝突的掛載點

1. 將 Amazon EBS 磁碟區指派給所需的 layer，但使用掛載點，例如 `/mnt/workspace` 永遠不會與 autofs 衝突。
2. 實作下列自訂配方，該配方會在 Amazon EBS 磁碟區上建立應用程式目錄，並從 連結至該目錄`/srv/www/`。如需如何實作和自訂配方的詳細資訊，請參閱 技術指南和配方 和 自訂 AWS OpsWorks Stacks。

```
mount_point = node['ebs']['raids']['/dev/md0']['mount_point'] rescue nil

if mount_point
  node[:deploy].each do |application, deploy|
    directory "#{mount_point}/#{application}" do
      owner deploy[:user]
      group deploy[:group]
      mode 0770
      recursive true
    end

    link "/srv/www/#{application}" do
      to "#{mount_point}/#{application}"
    end
  end
end
```

3. 新增 `depends 'deploy'` 行到自訂技術指南的 `metadata.rb` 檔案。

4. 將此配方指派至 layer 的設定事件。

安全

Security (安全性) 標籤包含下列設定。

安全群組

Layer 必須至少具有一個相關聯的安全群組。您可以指定如何在建立或更新堆疊時關聯安全群組。AWS OpsWorks Stacks 提供一組標準的內建安全群組。

- 預設選項是讓 AWS OpsWorks Stacks 自動將適當的內建安全群組與每個 layer 建立關聯。
- 您也可以選擇不自動為內建安全群組建立關聯，而在建立 layer 時將自訂安全群組與每個 layer 建立關聯。

如需安全群組的詳細資訊，請參閱 使用安全群組。

建立 layer 後，您可以使用 Security Groups (安全群組) 來將更多安全群組新增至 layer，藉由從 Custom security groups (自訂安全群組) 清單中選取這些安全群組。將安全群組新增至 layer 之後，AWS OpsWorks Stacks 會將其新增至所有新的執行個體。（請注意，重新啟動的執行個體存放區執行個體會變成新的執行個體，因此也會有新的安全群組。) AWS OpsWorks Stacks 不會將安全群組新增至線上執行個體。

您可以按一下 x 來刪除現有的安全群組，如下所示：

- 如果您選擇讓 AWS OpsWorks Stacks 自動關聯內建安全群組，您可以按一下 x 來刪除先前新增的自訂安全群組，但無法刪除內建群組。
- 如果您選擇不自動為內建安全群組建立關聯，則可以刪除任何現有安全群組 (包括原始安全群組)，只要該 layer 至少保留一個群組即可。

從 layer 移除安全群組之後， AWS OpsWorks Stacks 不會將其新增至任何新的或重新啟動的執行個體。 AWS OpsWorks Stacks 不會從線上執行個體移除安全群組。

> **Note**
>
> 如果您的堆疊是在 VPC 中執行，您可以使用 Amazon EC2 主控台、API 或 CLI 來新增或移除線上執行個體的安全群組。不過，此安全群組不會在 Stacks AWS OpsWorks 主控台中顯示。如果您想要移除安全群組，您還必須使用 Amazon EC2。如需詳細資訊，請參閱安全群組。

注意下列事項：

- 您無法藉由新增更嚴格的安全群組，來限制內建安全群組的連接埠存取設定。有多個安全群組時，Amazon EC2 會使用最寬鬆的設定。
- 您不應修改內建安全群組的組態。當您建立堆疊時， AWS OpsWorks Stacks 會覆寫內建安全群組的組態，因此您所做的任何變更都會在您下次建立堆疊時遺失。

如果您發現一或多個 layer 需要更嚴格的安全群組設定，請執行下列步驟：

1. 使用適當的設定來建立自訂安全群組，並將其新增至適當的 layer。

   除了內建群組之外，堆疊中的每個 layer 都必須至少具有一個安全群組，即使只有一個 layer 需要自訂設定。

2. 編輯堆疊組態並將 Use OpsWorks security groups (使用 OpsWorks 安全群組) 設定切換至 No (否)。

   AWS OpsWorks Stacks 會自動從每一層移除內建的安全群組。

如需安全群組的詳細資訊，請參閱 Amazon EC2 安全群組。

EC2 執行個體描述檔

您可以為 layer 的執行個體變更 EC2 描述檔。如需詳細資訊，請參閱指定在 EC2 執行個體上執行之應用程式的許可。

CloudWatch Logs

CloudWatch Logs 索引標籤可讓您啟用或停用 Amazon CloudWatch Logs。CloudWatch Logs 整合適用於 Chef 11.10 和 Chef 12 Linux 型堆疊。如需啟用 CloudWatch Logs 整合和指定要在 CloudWatch Logs 主控台中管理之日誌的詳細資訊，請參閱 搭配 Stacks AWS OpsWorks 使用 Amazon CloudWatch Logs。

標籤

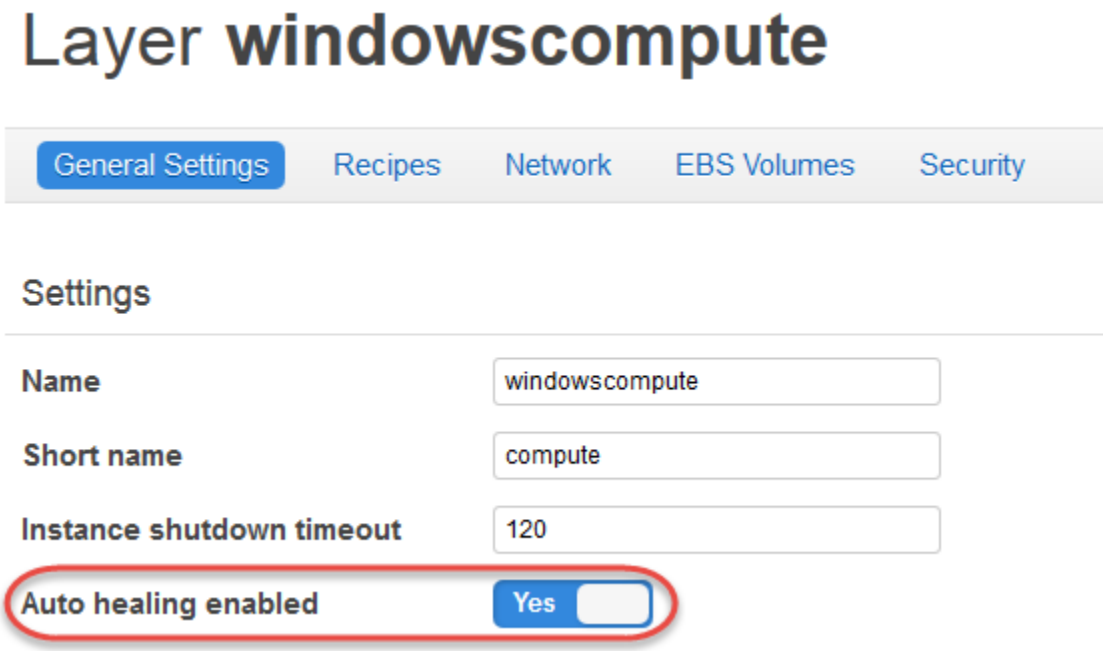
Tags (標籤) 標籤可讓您將成本分配標籤套用至您的 layer。新增標籤後，您可以在 AWS 帳單與成本管理 主控台中啟用標籤。建立標籤時，您會將標籤套用於已加上標籤的結構中每項資源。例如，如果您將標籤套用至 layer，您要將標籤套用至 layer 中的每個執行個體、Amazon EBS 磁碟區或 Elastic Load Balancing 負載平衡器。如需如何啟用標籤並使用它們來追蹤和管理 AWS OpsWorks Stacks 資源成本的詳細資訊，請參閱 帳單與成本管理使用者指南中的使用成本分配標籤和啟用使用者定義的成本分配標籤。如需 AWS OpsWorks Stacks 中標記的詳細資訊，請參閱標籤。

使用自動修復來替換故障的執行個體

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

每個執行個體都有 Stacks AWS OpsWorks 代理程式，可定期與服務通訊。 AWS OpsWorks Stacks 使用該通訊來監控執行個體的運作狀態。如果代理程式超過約五分鐘未與服務通訊， AWS OpsWorks Stacks 會將執行個體視為失敗。

自動修復是在 layer 層級設定；您可以透過編輯 layer 設定來變更自動修復設定，如下列螢幕擷取畫面所示。



> **Note**
>
> 執行個體可以是多個 layer 的成員。如果其中任何圖層已停用自動修復， AWS OpsWorks 則 Stacks 不會在執行個體失敗時修復執行個體。

如果 layer 已啟用自動修復 — 預設設定 —AWS OpsWorks Stacks 會自動取代 layer 的失敗執行個體，如下所示：

執行個體後端執行個體

1. 停止 Amazon EC2 執行個體，並驗證其已關閉。
2. 刪除根磁碟區上的資料。
3. 建立具有相同主機名稱、組態和層成員資格的新 Amazon EC2 執行個體。
4. 重新連接任何 Amazon EBS 磁碟區，包括最初啟動舊執行個體之後連接的磁碟區。
5. 指派新的公有及私有 IP 地址。
6. 如果舊執行個體與彈性 IP 地址相關聯，則會將新執行個體與相同的 IP 地址建立關聯。

Amazon EBS 支援的執行個體

1. 停止 Amazon EC2 執行個體，並驗證其已停止。
2. 啟動 EC2 執行個體。

自動修復的執行個體恢復上線後， AWS OpsWorks Stacks 會在所有堆疊的執行個體上觸發設定生命週期事件。相關聯的堆疊設定和部署屬性包括執行個體的公有和私有 IP 地址。自訂設定配方可以從節點物件取得新的 IP 地址。

如果您為 layer 執行個體指定 Amazon EBS 磁碟區， AWS OpsWorks Stacks 會建立新的磁碟區，並在執行個體啟動時將其連接至每個執行個體。如果您稍後想要將磁碟區從執行個體分離，請使用資源頁面。

當 AWS OpsWorks Stacks 自動修復 layer 的其中一個執行個體時，它會以下列方式處理磁碟區：

- 如果磁碟區在執行個體故障時連接至執行個體，則會儲存磁碟區及其資料，而 AWS OpsWorks Stacks 會將磁碟區連接至新的執行個體。
- 如果在執行個體故障時未將磁碟區連接至執行個體，則 AWS OpsWorks Stacks 將使用 layer 指定之組態建立新的空磁碟區，並將該磁碟區連接至新執行個體。

根據預設會為所有 layer 啟用自動修復，但您可以編輯 layer 的一般設定來將其停用。

> **Important**
>
> 如果您已啟用自動修復，請務必執行下列作業：
>
> - 僅使用 AWS OpsWorks Stacks 主控台、CLI 或 API 來停止執行個體。
>
> 如果您以任何其他方式停止執行個體，例如使用 Amazon EC2 主控台， AWS OpsWorks Stacks 會將執行個體視為失敗，並自動修復。

- 如果執行個體已自動修復，請使用 Amazon EBS 磁碟區來存放您不想遺失的任何資料。

  自動修復會停止舊的 Amazon EC2 執行個體，這會銷毀任何未存放在 Amazon EBS 磁碟區的資料。Amazon EBS 磁碟區會重新連接至新的執行個體，以保留任何儲存的資料。

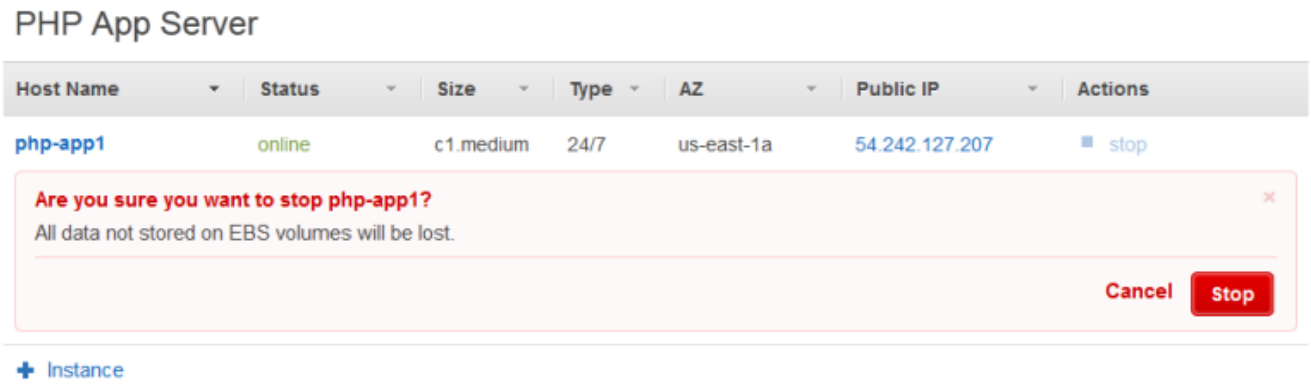
刪除 OpsWorks Layer

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

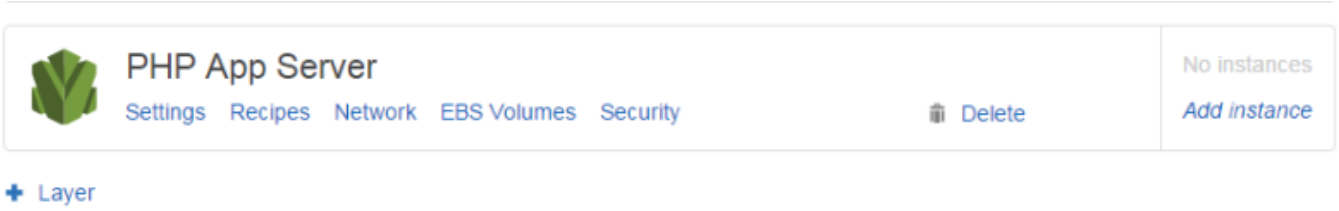
如果您不再需要 AWS OpsWorks Stacks layer，您可以從堆疊中刪除它。

刪除 OpsWorks 層

1. 在導覽窗格中，按一下 Instances (執行個體)。
2. 在 Instances (執行個體) 頁面上，在您要刪除的 layer 名稱下，按一下每個執行個體 Actions (動作) 欄位中的 stop (停止)。



3. 每個執行個體停止後，按一下 delete (刪除) 將其從 layer 中刪除。
4. 在導覽窗格中，按一下 Layers (Layer)。
5. 在 Layers (Layer) 頁面上，選擇 Delete (刪除)。



Elastic Load Balancing Layer

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Elastic Load Balancing 的運作方式與 AWS OpsWorks Stacks layer 略有不同。您可以使用 Elastic Load Balancing 主控台或 API 來建立負載平衡器，然後將其連接到現有的 layer，而不是建立 layer 並將其新增至其中。除了將流量分配到 layer 的執行個體之外，Elastic Load Balancing 還會執行下列動作：

- 偵測運作狀態不佳的 Amazon EC2 執行個體，並將流量重新路由至其餘運作狀態良好的執行個體，直到還原運作狀態不佳的執行個體為止。
- 自動擴展處理容量的請求，以回應傳入的流量。
- 若您啟用連接耗盡，負載平衡器會停止將新的請求傳送至狀況不良或即將取消註冊的執行個體，但會在到達指定的逾時值之前保持連線，讓執行個體完成任何傳遞中的請求。

將負載平衡器連接到 layer 之後， AWS OpsWorks Stacks 會執行下列動作：

- 取消註冊任何目前註冊的執行個體。
- 在該層的執行個體上線時自動註冊，並在執行個體離線時取消註冊，包含負載式和時間式執行個體。
- 自動開始在可用區域將請求路由傳送到註冊的執行個體。

如果您已啟用負載平衡器的連線耗盡功能，您可以指定 Stacks AWS OpsWorks 是否支援此功能。如果您啟用連線耗盡支援（預設設定），則在執行個體關閉後， AWS OpsWorks Stacks 會執行下列動作：

- 從負載平衡器取消註冊執行個體。

  負載平衡器會停止傳送新的請求，並啟動連接耗盡。
- 延遲觸發關機生命週期事件，直到負載平衡器完成連接耗盡。

如果您未啟用連線耗盡支援，即使執行個體仍連接至負載平衡器， AWS OpsWorks Stacks 仍會在執行個體關閉時觸發關機事件。

若要搭配堆疊使用 Elastic Load Balancing，您必須先使用 Elastic Load Balancing 主控台、CLI 或 API，在相同區域中建立一或多個負載平衡器。建議您注意以下事項：

- 您只能將一個負載平衡器連接到一個 layer。
- 每個負載平衡器只能處理一個 layer。
- AWS OpsWorks Stacks 不支援 Application Load Balancer。您只能搭配 AWS OpsWorks Stacks 使用 Classic Load Balancer。

這表示您必須為每個要平衡的堆疊中的每個層建立單獨的 Elastic Load Balancing 負載平衡器，並僅用於該用途。建議的做法是為您計劃與 AWS OpsWorks Stacks 搭配使用的每個 Elastic Load Balancing 負載平衡器指派獨特的名稱，例如 MyStack1-RailsLayer-ELB，以避免將負載平衡器用於多個用途。

> **Important**
>
> 建議您為 Stacks 層建立新的 Elastic Load Balancing AWS OpsWorks 負載平衡器。如果您選擇使用現有的 Elastic Load Balancing 負載平衡器，您應該先確認它未用於其他用途，也沒有連接的執行個體。在負載平衡器連接到 layer 之後，OpsWorks 會移除任何現有的執行個體，並設定負載平衡器，使其僅處理 layer 的執行個體。雖然在技術上可以使用 Elastic Load Balancing 主控台或 API 在將負載平衡器連接到 layer 之後修改負載平衡器的組態，但您不應該這麼做；變更不會永久存在。

將 Elastic Load Balancing 負載平衡器連接到 layer

1. 如果您尚未這麼做，請使用 Elastic Load Balancing 主控台、API 或 CLI 在堆疊的區域中建立負載平衡器。當您建立負載平衡器時，請執行下列作業：

   - 請務必指定適合您應用程式的運作狀態檢查 ping 路徑。

預設 ping 路徑為 `/index.html`，因此若您的應用程式根並未包含 `index.html`，您必須指定適當的 ping 路徑，否則運作狀態檢查會失敗。

- 若您欲使用連接耗盡，請確認已啟用該功能，並且具有適當的逾時值。

如需詳細資訊，請參閱 Elastic Load Balancing。

2. 建立您希望平衡的 layer 或編輯現有 layer 的網路設定。

> **Note**
>
> 您無法在建立自訂 layer 時連接負載平衡器。您必須編輯 layer 的設定。

3. 在 Elastic Load Balancing 下，選取您要連接到 layer 的負載平衡器，並指定是否要 Stacks AWS OpsWorks 支援連線耗盡。

將負載平衡器連接到 layer 之後， AWS OpsWorks Stacks 會在堆疊的執行個體上觸發設定生命週期事件，通知他們變更。當您分離負載平衡器時， AWS OpsWorks Stacks 也會觸發設定事件。

> **Note**
>
> 執行個體啟動後， AWS OpsWorks Stacks 會執行安裝和部署配方，以安裝套件和部署應用程式。在這些配方完成後，執行個體會處於線上狀態，且 AWS OpsWorks Stacks 會向 Elastic Load Balancing 註冊執行個體。當執行個體上線後， AWS OpsWorks Stacks 也會觸發設定事件。這表示 Elastic Load Balancing 註冊和設定配方可以同時執行，而且執行個體可以在設定配方完成之前註冊。為了確保配方在向 Elastic Load Balancing 註冊執行個體之前完成，您應該將配方新增至 layer 的設定或部署生命週期事件。如需詳細資訊，請參閱執行配方。

有時候將執行個體從負載平衡器移除也會非常有用。例如，當您更新應用程式時，我們建議您將應用程式部署至單一執行個體，並在將其部署到每個執行個體前，驗證應用程式已正常運作。您通常會將執行個體從負載平衡器移除，使其在您完成驗證更新之前不會接收到使用者請求。

您必須使用 Elastic Load Balancing 主控台或 API，暫時從負載平衡器移除線上執行個體。以下說明如何使用主控台。

暫時將執行個體從負載平衡器移除

1. 開啟 Amazon EC2 主控台，然後選擇負載平衡器。

2. 選擇適當的負載平衡器，然後開啟 Instances (執行個體) 標籤。
3. 在執行個體的 Actions (動作) 資料行中，選擇 Remove from Load Balancer (從負載平衡器移除)。
4. 當您完成時，請選擇 Edit Instances (編輯執行個體)，然後將執行個體返回負載平衡器。

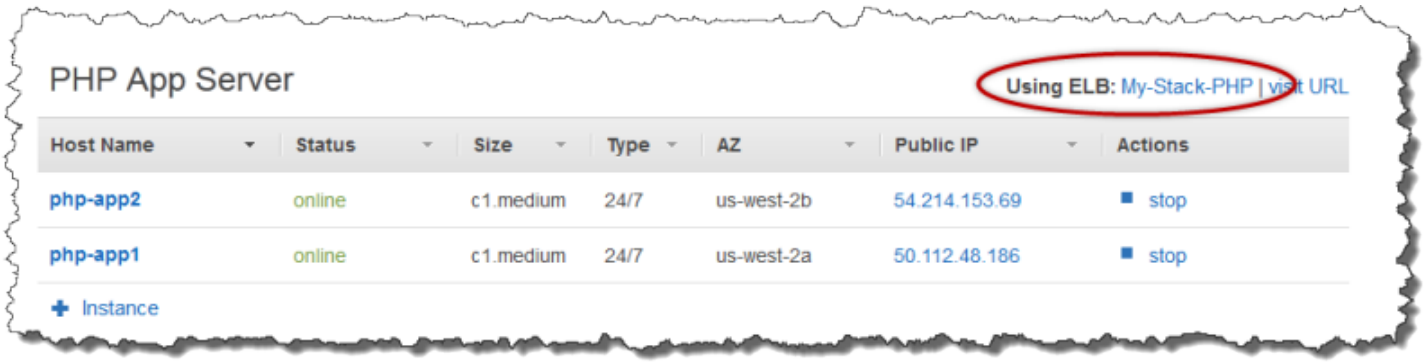
> **Important**
>
> 如果您使用 Elastic Load Balancing 主控台或 API 從負載平衡器移除執行個體，您還必須使用 Elastic Load Balancing 將其放回。 AWS OpsWorks Stacks 不知道您使用其他服務主控台或 APIs 執行的操作，而且不會將執行個體傳回負載平衡器。有時候， AWS OpsWorks Stacks 可以將執行個體新增回 ELB，但這並不保證行為，而且不會在所有情況下發生。

您可以將多個負載平衡器連接到特定的執行個體組，如下所示：

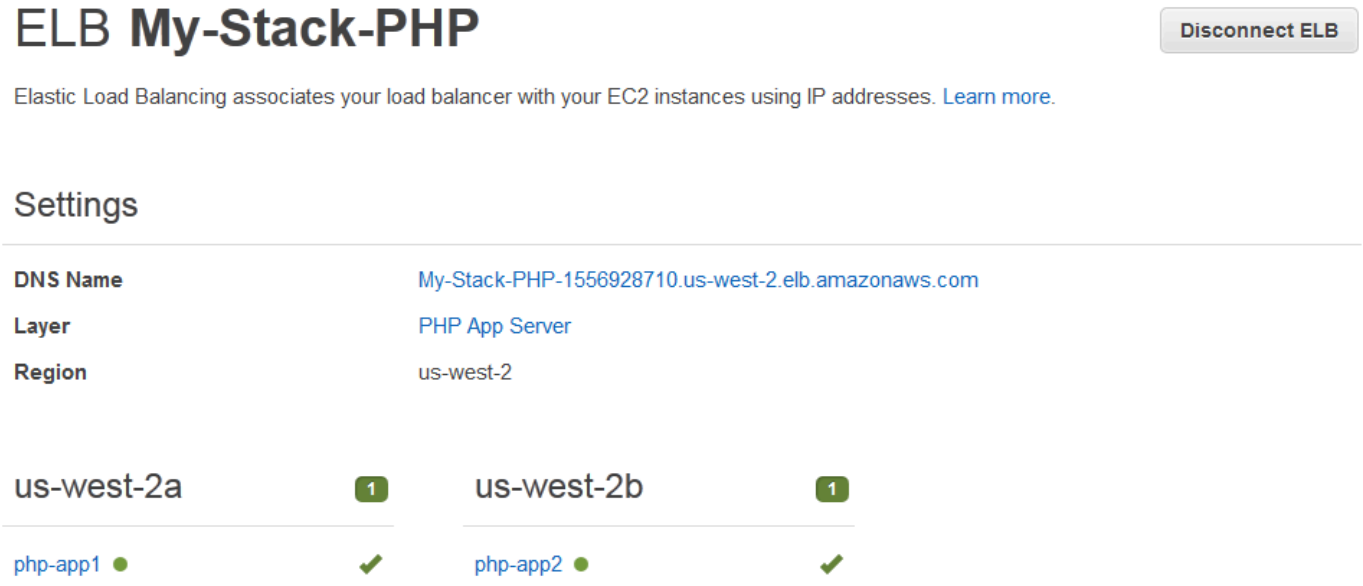
連接多個負載平衡器

1. 使用 Elastic Load Balancing 主控台、API 或 CLI 來建立一組負載平衡器。
2. 為每個負載平衡器建立自訂 layer，並將其中一個負載平衡器連接到該 layer。您不需要為這些 layer 實作任何自訂配方。預設自訂 layer 已足夠。
3. 將執行個體組新增至每個自訂 layer。

您可以藉由前往執行個體頁面並按一下適當的負載平衡器名稱，來檢查負載平衡器的屬性。



ELB 頁面會顯示負載平衡器的基本屬性，包含其 DNS 名稱和關聯執行個體的運作狀態。若堆疊正在 VPC 中執行，頁面會顯示子網路，而非可用區域。綠色的核取記號表示運作狀態良好的執行個體。您可以按一下名稱透過負載平衡器以連線到伺服器。



Amazon RDS 服務層

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Amazon RDS 服務層代表 Amazon RDS 執行個體。layer 只能代表現有的 Amazon RDS 執行個體，您必須使用 Amazon RDS 主控台或 API 分別建立。

將 Amazon RDS 服務層整合到您的堆疊的基本程序如下：

1. 使用 Amazon RDS 主控台、API 或 CLI 來建立執行個體。

   請務必記錄執行個體的 ID、主要使用者名稱、主要密碼和資料庫名稱。
2. 若要將 Amazon RDS layer 新增至堆疊，請向堆疊註冊 Amazon RDS 執行個體。
3. 將 layer 連接至應用程式，將 Amazon RDS 執行個體的連線資訊新增至應用程式的`deploy`屬性。
4. 使用特定語言檔案或`deploy`屬性中的資訊，將應用程式連線至 Amazon RDS 執行個體。

   如需如何將應用程式連線至資料庫伺服器的詳細資訊，請參閱the section called “連線至資料庫”

> **Warning**
>
> 請確定執行個體之主要密碼和使用者名稱中的字元與您的應用程式伺服器相容。例如，使用 Java App Server layer，包括在任一字串&中會導致 XML 剖析錯誤，以防止 Tomcat 伺服器啟動。

主題

- 指定安全群組
- 向堆疊註冊 Amazon RDS 執行個體
- 將 Amazon RDS 服務層與應用程式建立關聯
- 從堆疊移除 Amazon RDS Service Layer

指定安全群組

若要搭配 Stacks 使用 Amazon RDS AWS OpsWorks 執行個體，資料庫或 VPC 安全群組必須允許從適當的 IP 地址存取。針對生產用途，安全群組通常會限制僅存取這些需要存取資料庫的 IP 地址。它通常包含您用來管理資料庫的系統地址，以及需要存取資料庫的 AWS OpsWorks Stacks 執行個體。當您在區域中建立第一個堆疊時， AWS OpsWorks Stacks 會自動為每種類型的 layer 建立 Amazon EC2 安全群組。為 AWS OpsWorks Stacks 執行個體提供存取權的簡單方法是將適當的 AWS OpsWorks Stacks 安全群組指派給 Amazon RDS 執行個體或 VPC。

指定現有 Amazon RDS 執行個體的安全群組

1. 前往 https://console.aws.amazon.com/rds/，開啟 Amazon RDS 主控台。
2. 按一下導覽窗格中的執行個體，然後選取適當的 Amazon RDS 執行個體。按一下 Instance Actions (執行個體動作)、Modify (修改)。
3. 從 Security Group (安全群組) 清單中選取下列安全群組，然後按一下 Continue (繼續) 和 Modify DB Instance (修改資料庫執行個體) 更新執行個體。
   - AWS-OpsWorks-DB-Master-Server (***security_group_id***) 安全群組。
   - 其執行個體將連線至資料庫之應用程式伺服器 layer 的安全群組。群組名稱包括 layer 名稱。例如，若要提供 PHP App Server 執行個體的資料庫存取權，請指定 AWS-OpsWorks-PHP-App-Server 群組。

如果您要建立新的 Amazon RDS 執行個體，您可以在啟動資料庫執行個體精靈的設定進階設定頁面上指定適當的 AWS OpsWorks Stacks 安全群組。如需如何使用此精靈的描述，請參閱建立 MySQL 資料庫執行個體和連線至 MySQL 資料庫執行個體上的資料庫。

如需如何指定 VPC 安全群組的資訊，請參閱 VPC 的安全群組。

向堆疊註冊 Amazon RDS 執行個體

若要在堆疊中新增 Amazon RDS 服務層，您必須向堆疊註冊執行個體。

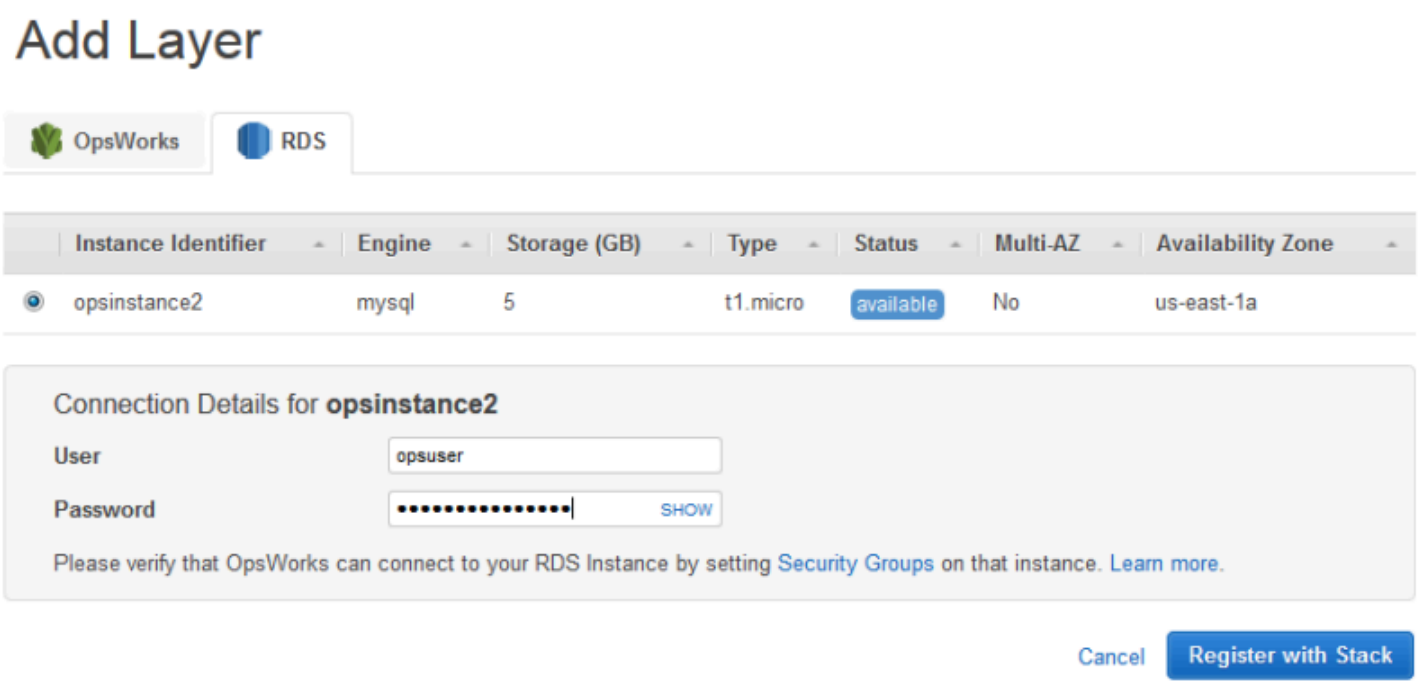
向堆疊註冊 Amazon RDS 執行個體

1. 在 AWS OpsWorks Stacks 主控台中，按一下導覽窗格中的圖層，按一下 + Layer 或新增圖層以開啟新增圖層頁面，然後按一下 RDS 標籤。
2. 如有必要，請更新堆疊的服務角色，如更新堆疊的服務角色中所述。
3. 按一下 RDS 索引標籤，列出可用的 Amazon RDS 執行個體。

> **Note**
>
> 如果您的帳戶沒有任何 Amazon RDS 執行個體，您可以按一下 RDS 標籤上的新增 RDS 執行個體來建立執行個體，這會帶您前往 Amazon RDS 主控台並啟動啟動資料庫執行個體精靈。您也可以直接前往 Amazon RDS 主控台，然後按一下啟動資料庫執行個體，或使用 Amazon RDS API 或 CLI。如需如何建立 Amazon RDS 執行個體的詳細資訊，請參閱 Amazon RDS 入門。

4. 選取適當的執行個體，並將 User (使用者) 和 Password (密碼) 設定為適當的使用者和密碼值，然後按一下 Register to Stack (向堆疊註冊)。

> **Important**
>
> 您必須確保您用來註冊 Amazon RDS 執行個體的使用者和密碼對應至有效的使用者和密碼。否則，您的應用程式將無法連線至執行個體。不過，您可以編輯 layer 以提供有效的使用者和密碼值，然後重新部署應用程式。



當您將 Amazon RDS 服務層新增至堆疊時， AWS OpsWorks Stacks 會為其指派 ID，並將相關聯的 Amazon RDS 組態新增至堆疊組態和部署屬性的 `[:opsworks][:stack]` 屬性。

> **Note**
>
> 如果您變更已註冊的 Amazon RDS 執行個體密碼，則必須手動更新 Stacks AWS OpsWorks 中的密碼，然後重新部署您的應用程式，以更新堆疊執行個體上的堆疊組態和部署屬性。

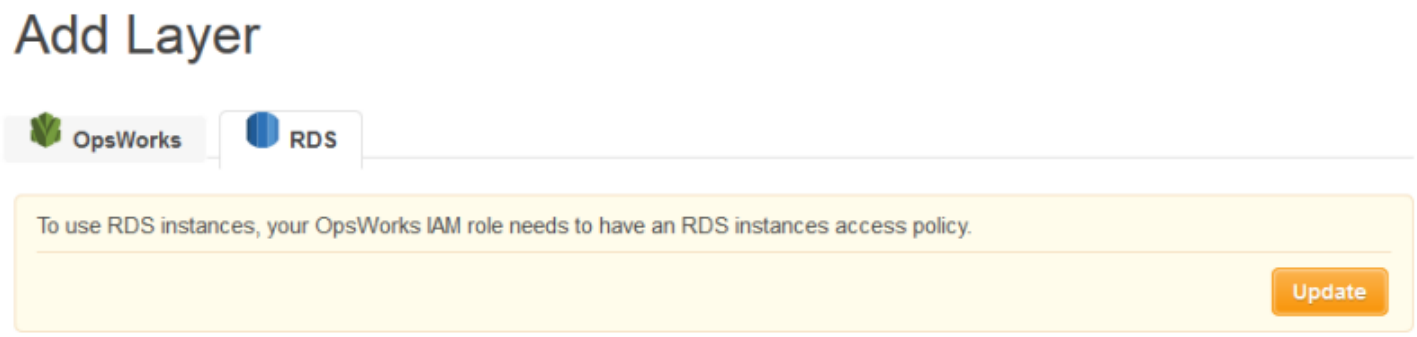
主題

- 更新堆疊的服務角色

更新堆疊的服務角色

每個堆疊都有一個 IAM 服務角色，指定 AWS OpsWorks Stacks 可以使用其他 AWS 服務代表您執行的動作。若要向堆疊註冊 Amazon RDS 執行個體，其服務角色必須授予 AWS OpsWorks Stacks 存取 Amazon RDS 的許可。

第一次將 Amazon RDS 服務層新增至其中一個堆疊時，服務角色可能缺少必要的許可。若是如此，當您按一下 Add Layer (新增 Layer) 頁面上的 RDS 標籤時，將會看到下列內容。



按一下更新，讓 AWS OpsWorks Stacks 將服務角色的政策更新為以下內容。

```
{"Statement": [{"Action": ["ec2:*", "iam:PassRole",
                       "cloudwatch:GetMetricStatistics",
                       "elasticloadbalancing:*",
                       "rds:*"],
            "Effect": "Allow",
            "Resource": ["*"] }]
}
```

> **Note**
>
> 您只需要執行更新一次。所有堆疊接著會自動使用更新過的角色。

將 Amazon RDS 服務層與應用程式建立關聯

新增 Amazon RDS 服務層之後，您可以將其與應用程式建立關聯。

- 您可以在建立應用程式時，或稍後編輯應用程式的組態，將 Amazon RDS layer 與應用程式建立關聯。
- 若要取消 Amazon RDS layer 與應用程式的關聯，請編輯應用程式的組態以指定不同的資料庫伺服器，或沒有伺服器。

  Amazon RDS layer 仍是堆疊的一部分，並且可以與不同的應用程式建立關聯。

將 Amazon RDS 執行個體與應用程式建立關聯後， AWS OpsWorks Stacks 會將資料庫連線資訊放在應用程式的伺服器上。每個伺服器執行個體上的應用程式接著都可以使用此資訊連線至資料庫。如需如何連線至 Amazon RDS 執行個體的詳細資訊，請參閱 the section called “連線至資料庫”。

從堆疊移除 Amazon RDS Service Layer

若要從堆疊中移除 Amazon RDS 服務層，請將其取消註冊。

取消註冊 Amazon RDS 服務層

1. 按一下導覽窗格中的圖層，然後按一下 Amazon RDS 服務圖層的名稱。
2. 按一下 Deregister (取消註冊)，並確認您想要將 layer 取消註冊。

此程序會從堆疊中移除 layer，但不會刪除基礎 Amazon RDS 執行個體。執行個體和任何資料庫都會保留在您的帳戶中，而且可以向其他堆疊註冊。您必須使用 Amazon RDS 主控台、API 或 CLI 來刪除執行個體。如需詳細資訊，請參閱刪除資料庫執行個體。

ECS 叢集層

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Amazon Elastic Container Service (Amazon ECS) 會管理 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體叢集上的 Docker 容器，稱為容器執行個體。ECS 叢集層代表 Amazon ECS 叢集，並提供以下功能來簡化叢集管理：

- 簡化的容器執行個體佈建及管理
- 容器執行個體作業系統及套件更新
- 使用者許可管理
- 容器執行個體效能監控
- Amazon Elastic Block Store (Amazon EBS) 磁碟區管理
- 公有和彈性 IP 地址管理
- 安全群組管理

ECS 叢集層具有下列限制和要求：

- layer 僅適用於在 VPC 中執行的 Chef 11.10 或 Chef 12 Linux 堆疊，包含預設 VPC。
- layer 的執行個體必須執行以下任一作業系統。
  - Amazon Linux 2
  - Amazon Linux 2018.03
  - Amazon Linux 2017.09
  - Amazon Linux 2017.03
  - Amazon Linux 2016.09
  - Amazon Linux 2016.03
  - Amazon Linux 2015.09
  - Amazon Linux 2015.03
  - Ubuntu 18.04 LTS
  - Ubuntu 16.04 LTS
  - Ubuntu 14.04 LTS
  - 自訂
- layer 的執行個體上的 AWS OpsWorks Stacks 代理程式版本必須為 3425-20150727112318 或更新版本。

主題

- 將 ECS 叢集層新增至堆疊
- 管理 ECS 叢集
- 從堆疊刪除 ECS 叢集層

將 ECS 叢集層新增至堆疊

AWS OpsWorks Stacks 可簡化啟動和維護現有 Amazon ECS 叢集容器執行個體的程序。若要建立或啟動其他 Amazon ECS 實體，例如叢集和任務，請使用 Amazon ECS 主控台、命令列界面 (CLI) 或 API。（如需詳細資訊，請參閱 Amazon Elastic Container Service 開發人員指南。）然後，您可以透過建立 ECS 叢集層來建立叢集與堆疊的關聯，以用於在 Stacks AWS OpsWorks 中管理叢集。

您可以將叢集與堆疊建立關聯，如下所示：

- 每個堆疊可以有一個 ECS 叢集層，代表單一叢集。
- 一個叢集只能與一個堆疊建立關聯。

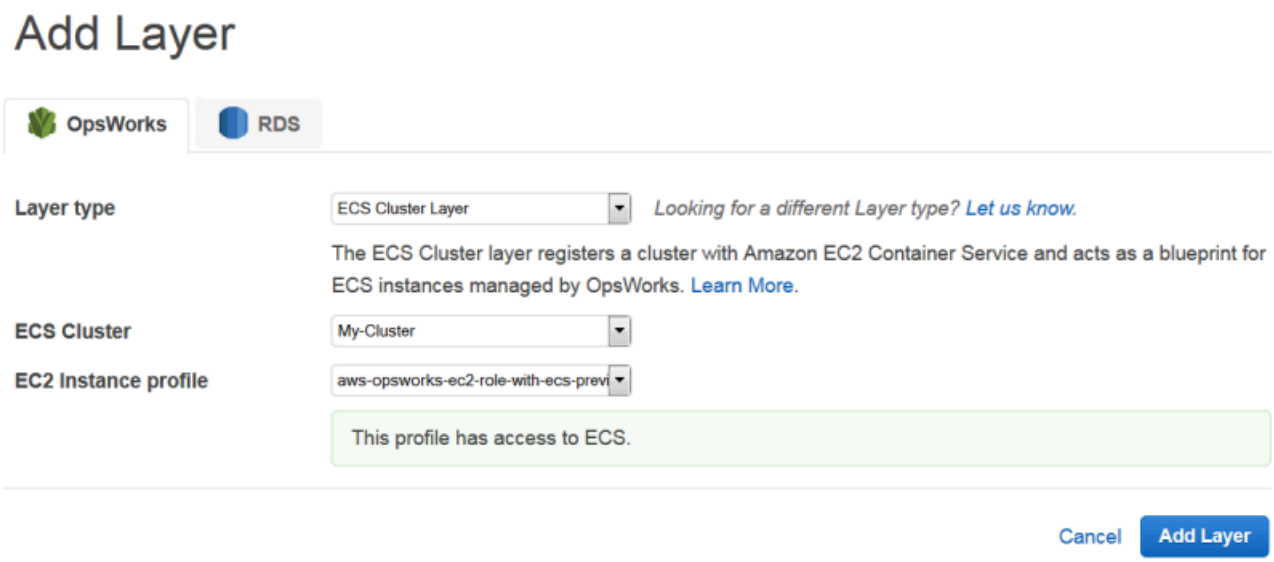
您必須先更新通常名為 AWS OpsWorks 的 Stacks AWS Identity and Access Management (IAM) 服務角色，以允許 AWS OpsWorks Stacks 代表您與 Amazon ECS 互動`aws-opsworks-service-role`，才能將 ECS 叢集層新增至堆疊。如需服務角色的詳細資訊，請參閱[允許 AWS OpsWorks Stacks 代表您]。

第一次建立 ECS 叢集層時，主控台會提供更新按鈕，您可以選擇引導 AWS OpsWorks Stacks 為您更新角色。 AWS OpsWorks 堆疊接著會顯示新增層頁面，讓您可以將層新增至堆疊。您只需要更新服務角色一次。然後，您可以使用更新的角色將 ECS 叢集層新增至任何堆疊。

> **Note**
>
> 若您偏好的話，您可以透過將 `ecs:*` 許可新增至現有的政策來手動更新服務角色的政策，如下所示：

```
{
  "Statement": [
    {
      "Action": [
        "ec2:*",
        "iam:PassRole",
        "cloudwatch:GetMetricStatistics",
        "elasticloadbalancing:*",
        "rds:*",
        "ecs:*"
      ],
      "Effect": "Allow",
      "Resource": ["*"]
    }
  ]
}
```

將叢集與堆疊建立關聯需要兩項操作：使用堆疊註冊叢集，然後建立關聯 layer。Stacks AWS OpsWorks 主控台結合了這些步驟； layer 建立會自動註冊指定的叢集。如果您使用 AWS OpsWorks Stacks API、CLI 或 SDK，則必須使用個別的操作來註冊叢集並建立相關聯的 layer。若要使用主控台將 ECS 叢集層新增至您的堆疊，請選擇層、選擇 + 層或新增層，然後選擇 ECS 叢集層類型。



Add Layer (新增 Layer) 頁面包含下列組態選項：

ECS 叢集

您要向堆疊註冊的 Amazon ECS 叢集。

EC2 Instance profile (EC2 執行個體描述檔)

叢集的 Amazon Elastic Compute Cloud(Amazon EC2) 執行個體描述檔。此設定檔授予在叢集容器執行個體上執行的應用程式存取其他 AWS 服務的許可，包括 Amazon ECS。當您建立第一個 ECS 叢集層時，請選擇具有 ECS 存取權的新設定檔以指示 AWS OpsWorks Stacks 建立必要的設定檔，其名稱為 `aws-opsworks-ec2-role-with-ecs`。然後，您可以將該設定檔用於所有後續的 ECS 叢集層。如需執行個體描述檔的詳細資訊，請參閱指定在 EC2 執行個體上執行之應用程式的許可。

您可以藉由編輯 layer 的組態指定其他設定，包含：

- 將 Elastic Load Balancing 負載平衡器連接至 layer。

  此方法可能適用於某些使用案例，但 Amazon ECS 提供更複雜的選項。如需詳細資訊，請參閱服務負載平衡。
- 指定是否要自動將公有 IP 地址或彈性 IP 地址指派給容器執行個體。

  若您停用兩種地址類型的自動指派，執行個體會無法上線，除非子網路已有適當設定的 NAT。如需詳細資訊，請參閱在 VPC 中執行堆疊。

管理 ECS 叢集

建立 ECS 叢集層之後，您可以使用 AWS OpsWorks Stacks 來管理叢集，如下所示：

佈建及管理容器執行個體

一開始，ECS 叢集層不包含任何容器執行個體，即使原始叢集已包含。一個選項是透過使用下列項目的適當組合，管理 layer 的執行個體：

- 手動新增全年無休執行個體至 layer 及在不需要時刪除執行個體。
- 透過將時間式執行個體新增至 layer，藉以在排程上新增或刪除執行個體。
- 將負載型執行個體新增至 layer，以根據 AWS OpsWorks Stacks 主機指標或 CloudWatch 警示新增或刪除執行個體。 使用自動負載型擴展

> **Note**
>
> 如果堆疊的預設作業系統不支援 Amazon ECS，您必須在建立容器執行個體時明確指定支援的作業系統：Amazon Linux 2、Amazon Linux 2018.03、Amazon Linux 2017.09、Amazon Linux 2017.03、Amazon Linux 2016.09、Amazon Linux 2016.03、Amazon Linux 2015.09、Amazon Linux 2015.03、Ubuntu 18.04 LTS、Ubuntu 16.04 LTS、Ubuntu 14.04 LTS 或 Custom。請勿使用 ECS Optimized AMI 在 ECS layer 中建立執行個體，因為此 AMI 已包含 ECS 代理程式。 AWS OpsWorks Stacks 也會嘗試在執行個體設定程序期間安裝 ECS 代理程式，且衝突可能會導致設定失敗。

如需詳細資訊，請參閱 最佳化應用程式伺服器的數目. AWS OpsWorks Stacks 將 AWS-OpsWorks-ECS-Cluster 安全群組指派給每個執行個體。每個新執行個體完成開機後， AWS OpsWorks Stacks 會安裝 Docker 和 Amazon ECS 代理程式，然後向叢集註冊執行個體，將其轉換為容器執行個體。

如果您偏好使用現有的容器執行個體，則可以向堆疊註冊它們，並將其指派給 ECS 叢集層。請注意，執行個體必須執行支援的作業系統，Amazon Linux 2015.03 或更新版本、Ubuntu 14.04 LTS 或更新版本。

> **Note**
>
> 容器執行個體不能同時屬於 ECS 叢集層和另一個內建層。不過，容器執行個體可以屬於 ECS 叢集層和一或多個自訂層。

執行作業系統及套件更新

新執行個體完成開機後， AWS OpsWorks Stacks 會安裝最新的更新。然後，您可以使用 AWS OpsWorks Stacks 將容器執行個體保持在最新狀態。如需詳細資訊，請參閱管理安全性更新。

管理使用者許可

AWS OpsWorks Stacks 提供簡單的方法來管理容器執行個體的許可，包括管理使用者的 SSH 金鑰。如需詳細資訊，請參閱管理使用者許可及管理 SSH 存取。

監控效能指標

AWS OpsWorks Stacks 提供各種方法來監控堆疊、 layer 或個別執行個體的效能指標。如需詳細資訊，請參閱監控。

您可以透過 Amazon ECS 處理其他管理任務，例如建立任務或服務。如需詳細資訊，請參閱《Amazon Elastic Container Service 開發人員指南》。

> **Note**
>
> 若要直接前往 Amazon ECS 主控台上的叢集頁面，請選擇執行個體，然後選擇靠近 ECS 叢集層區段右上角的 ECS 叢集。

從堆疊刪除 ECS 叢集層

當您不再需要叢集時，請刪除 ECS 叢集層並取消註冊相關聯的叢集。從堆疊移除叢集需要兩項操作：取消註冊叢集，然後刪除關聯 layer。Stacks AWS OpsWorks 主控台結合了這些步驟； layer 刪除會自動取消註冊指定的叢集。如果您使用 AWS OpsWorks Stacks API、CLI 或 SDK，則必須使用個別的操作來取消註冊叢集並刪除相關聯的 layer。

使用主控台刪除 ECS 叢集層

1. 如果您想要控制任務關閉的方式，請使用 Amazon ECS 主控台、API 或 CLI 來縮減規模並刪除叢集的服務。如需詳細資訊，請參閱清除 Amazon ECS 資源。
2. 停止 layer 的執行個體，然後刪除執行個體。當您停止容器執行個體時， AWS OpsWorks Stacks 會自動停止任何執行中的任務、從叢集取消註冊執行個體，以及終止執行個體。

> **Note**
>
> 若您有已使用堆疊註冊的現有容器執行個體，您可以從 layer 取消指派執行個體，然後取消註冊他們，使執行個體回歸 ECS 的控制。

3. 刪除 layer. AWS OpsWorks Stacks 會取消註冊相關聯的叢集，但不會將其刪除。叢集會保留在 Amazon ECS 中。

自訂 AWS OpsWorks Stacks Layers

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

自訂 layer 僅包含一組最少的配方。您可以實作自訂配方並將其指派到 layer 的 生命週期事件，以將適當的功能新增至 layer。

自訂 layer 具有下列組態設定。

> **Note**
>
> AWS OpsWorks Stacks 會自動在 layer 的執行個體上安裝 Ruby。如果您想要在執行個體上執行 Ruby 程式碼，但不想使用預設的 Ruby 版本，則可以使用自訂 JSON 或自訂屬性檔案，來指定您慣用的版本。如需詳細資訊，請參閱Ruby 版本。

建立自訂 layer 的基本程序有下列步驟：

1. 實作技術指南，其中包含安裝和設定套件、處理組態變更、部署應用程式等所需的配方與相關聯檔案。

   根據您的需求，您可能也需要取消部署和關機任務的配方。如需詳細資訊，請參閱技術指南和配方。
2. 建立自訂 layer。

3. 將您的配方指定給適當的生命週期事件。

接著，您可以將執行個體新增至 layer、啟動執行個體，然後部署應用程式到這些執行個體中。

> **Important**
>
> 若要將應用程式部署到自訂 layer 的執行個體，您必須實作配方來處理部署操作，並將配方指派給該 layer 的部署事件。

個別 layer 作業系統套件安裝

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

從 Chef 12 開始，您必須使用自訂配方，在執行不同作業系統的 layer 上安裝套件。此方法提供套件安裝的最大彈性和控制。

例如，假設您想要在執行 RedHat、Ubuntu 和 Amazon 版本之 Linux 作業系統的 layer 上安裝 Apache。RedHat 和 Amazon Linux 的 Apache 套件稱為 `httpd`，但在 Ubuntu 上稱為 `apache2`。

若要解決套件命名的差異，您可以使用下列範例配方中的類似語法。此配方會安裝每個作業系統適用的 Apache 套件。此範例根據 Chef 文件。

```
package "Install Apache" do
   case node[:platform]
      when "redhat", "amazon"
         package_name "httpd"
      when "ubuntu"
         package_name "apache2"
   end
end
```

如需如何使用 `package` 資源管理套件的詳細資訊，請前往 Chef 文件中的套件頁面。

或者，您可以從 Chef 配方 DSL (網域特定語言) 使用 `value_for_platform` 協助程式方法，更簡潔地完成相同的事項：

```
package "Install Apache" do
   package_name value_for_platform(
      ["redhat", "amazon"] => { "default" => "httpd" },
      ["ubuntu"] => { "default" => "apache2" }
   )
end
```

如需使用 `value_for_platform` 協助程式方法的資訊，請前往關於配方 DSL。

執行個體

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

執行個體代表運算資源，例如 Amazon EC2 執行個體，可處理服務應用程式、平衡流量等工作。執行個體的作業系統可以有多個 Linux 發行版本之一或 Windows Server 2012 R2。

您可用下列任一方法將執行個體新增至堆疊：

- 使用 AWS OpsWorks Stacks 將執行個體新增至堆疊。您新增的執行個體代表 Amazon EC2 執行個體。
- 對於 Linux 型堆疊，您可以註冊在其他地方建立的執行個體，包括您使用 Amazon EC2 建立的執行個體，以及在您自己的硬體上執行的現場部署執行個體。

  然後，您可以使用 AWS OpsWorks Stacks，以與使用 Stacks AWS OpsWorks 建立的執行個體大致相同的方式管理這些執行個體

本節說明如何使用 AWS OpsWorks Stacks 來建立和管理執行個體。

主題

- 使用 AWS OpsWorks Stacks 執行個體

- 使用在 AWS OpsWorks Stacks 之外建立的運算資源
- 編輯執行個體組態
- 刪除 AWS OpsWorks Stacks 執行個體
- 使用 SSH 登入 Linux 執行個體
- 使用 RDP 登入 Windows 執行個體

使用 AWS OpsWorks Stacks 執行個體

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以使用 AWS OpsWorks Stacks 來建立執行個體，並將其新增至堆疊。

主題

- AWS OpsWorks Stacks 作業系統
- 將執行個體新增至 Layer
- 使用自訂 AMI
- 手動啟動、停止和重新開機全年無休的執行個體
- 使用時間型和負載型執行個體管理負載

AWS OpsWorks Stacks 作業系統

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 支援多個內建作業系統的 64 位元版本，包括 Amazon 和 Ubuntu Linux 發行版本，以及 Microsoft Windows Server。下列為一般注意事項：

- 堆疊的執行個體可以執行 Linux 或 Windows。

  堆疊可以具備不同 Linux 發行版本或不同執行個體，但您不可以混合 Linux 和 Windows 執行個體。
- 您可以使用自訂 AMIs (Amazon Machine Image)，但這些 AMI 必須以本節主題中所述的其中一個 AWS OpsWorks Stacks 支援的 AMIs 為基礎。雖然可以使用從自訂或社群所產生 AMI 建立而成的其他作業系統 (例如 CentOS 6.x) 建立或註冊執行個體，但並未正式受到支援。
  - Linux 作業系統
  - Microsoft Windows Server
- 您可以手動啟動和停止執行個體，或讓 AWS OpsWorks Stacks 自動擴展執行個體的數目。

  您可以針對任何堆疊使用以時間為基礎的自動擴展；Linux 堆疊也可以使用以負載為基礎的擴展。
- 除了使用 AWS OpsWorks Stacks 來建立 Amazon EC2 執行個體之外，您也可以向在 Stacks 外部建立的 Linux 堆疊註冊執行個體。 AWS OpsWorks

  這包括在您自己的硬體上執行的 Amazon EC2 執行個體和執行個體。不過，它們必須執行其中一種支援的 Linux 發行版本。您無法註冊 Amazon EC2 或內部部署 Windows 執行個體。

您可以執行 AWS OpsWorks Stacks `DescribeOperatingSystems` API 來傳回支援的作業系統及其支援的 Chef 版本清單。下面是使用 AWS CLI的範例命令。

```
aws opsworks describe-operating-systems
```

以下是回應範例。

```
{
    "OperatingSystems": [
        {
            "Name": "Amazon Linux",
            "Id": "Amazon Linux",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
```

```
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2014.03",
            "Supported": false
        },
        {
            "Name": "Amazon Linux 2",
            "Id": "Amazon Linux 2",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2"
        },
        {
            "Name": "Amazon Linux 2014.09",
            "Id": "Amazon Linux 2014.09",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2014.09",
            "Supported": false
        },
```

```
        {
            "Name": "Amazon Linux 2015.03",
            "Id": "Amazon Linux 2015.03",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2015.03",
            "Supported": false
        },
        {
            "Name": "Amazon Linux 2015.09",
            "Id": "Amazon Linux 2015.09",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
```

```
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2015.09",
            "Supported": false
        },
        {
            "Name": "Amazon Linux 2016.03",
            "Id": "Amazon Linux 2016.03",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2016.03"
        },
        {
            "Name": "Amazon Linux 2016.09",
            "Id": "Amazon Linux 2016.09",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
```

```
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2016.09"
        },
        {
            "Name": "Amazon Linux 2017.03",
            "Id": "Amazon Linux 2017.03",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2017.03"
        },
        {
            "Name": "Amazon Linux 2017.09",
            "Id": "Amazon Linux 2017.09",
            "Type": "Linux",
            "ConfigurationManagers": [
```

```
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2017.09"
        },
        {
            "Name": "Amazon Linux 2018.03",
            "Id": "Amazon Linux 2018.03",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                }
            ],
            "ReportedName": "amazon",
            "ReportedVersion": "2018.03"
        },
        {
            "Name": "CentOS Linux 7",
            "Id": "CentOS Linux 7",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
```

```
                    "Version": "12"
                }
            ],
            "ReportedName": "CentOS Linux",
            "ReportedVersion": "7"
        },
        {
            "Name": "Microsoft Windows Server 2012 R2 Base",
            "Id": "Microsoft Windows Server 2012 R2 Base",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2012 r2 standard",
            "Supported": false
        },
        {
            "Name": "Microsoft Windows Server 2012 R2 with SQL Server Express",
            "Id": "Microsoft Windows Server 2012 R2 with SQL Server Express",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2012 r2 standard",
            "Supported": false
        },
        {
            "Name": "Microsoft Windows Server 2012 R2 with SQL Server Standard",
            "Id": "Microsoft Windows Server 2012 R2 with SQL Server Standard",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
```

```
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2012 r2 standard",
            "Supported": false
        },
        {
            "Name": "Microsoft Windows Server 2012 R2 with SQL Server Web",
            "Id": "Microsoft Windows Server 2012 R2 with SQL Server Web",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2012 r2 standard",
            "Supported": false
        },
        {
            "Name": "Microsoft Windows Server 2019 Base",
            "Id": "Microsoft Windows Server 2019 Base",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2019 datacenter"
        },
        {
            "Name": "Microsoft Windows Server 2019 with SQL Server Express",
            "Id": "Microsoft Windows Server 2019 with SQL Server Express",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2019 datacenter"
        },
```

```
        {
            "Name": "Microsoft Windows Server 2019 with SQL Server Standard",
            "Id": "Microsoft Windows Server 2019 with SQL Server Standard",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2019 datacenter"
        },
        {
            "Name": "Microsoft Windows Server 2019 with SQL Server Web",
            "Id": "Microsoft Windows Server 2019 with SQL Server Web",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2019 datacenter"
        },
        {
            "Name": "Microsoft Windows Server 2022 Base",
            "Id": "Microsoft Windows Server 2022 Base",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2022 datacenter"
        },
        {
            "Name": "Microsoft Windows Server 2022 with SQL Server Express",
            "Id": "Microsoft Windows Server 2022 with SQL Server Express",
            "Type": "Windows",
            "ConfigurationManagers": [
```

```
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2022 datacenter"
        },
        {
            "Name": "Microsoft Windows Server 2022 with SQL Server Standard",
            "Id": "Microsoft Windows Server 2022 with SQL Server Standard",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2022 datacenter"
        },
        {
            "Name": "Microsoft Windows Server 2022 with SQL Server Web",
            "Id": "Microsoft Windows Server 2022 with SQL Server Web",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ],
            "ReportedName": "microsoft windows server",
            "ReportedVersion": "2022 datacenter"
        },
        {
            "Name": "Red Hat Enterprise Linux 7",
            "Id": "Red Hat Enterprise Linux 7",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
```

```
                    "Name": "Chef",
                    "Version": "11.10"
                }
            ],
            "ReportedName": "Red Hat Enterprise Linux",
            "ReportedVersion": "7"
        },
        {
            "Name": "Ubuntu 12.04 LTS",
            "Id": "Ubuntu 12.04 LTS",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ],
            "ReportedName": "ubuntu",
            "ReportedVersion": "12.04",
            "Supported": false
        },
        {
            "Name": "Ubuntu 14.04 LTS",
            "Id": "Ubuntu 14.04 LTS",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
```

```
                "Version": "11.10"
            }
        ],
        "ReportedName": "ubuntu",
        "ReportedVersion": "14.04"
    },
    {
        "Name": "Ubuntu 16.04 LTS",
        "Id": "Ubuntu 16.04 LTS",
        "Type": "Linux",
        "ConfigurationManagers": [
            {
                "Name": "Chef",
                "Version": "12"
            }
        ],
        "ReportedName": "ubuntu",
        "ReportedVersion": "16.04"
    },
    {
        "Name": "Ubuntu 18.04 LTS",
        "Id": "Ubuntu 18.04 LTS",
        "Type": "Linux",
        "ConfigurationManagers": [
            {
                "Name": "Chef",
                "Version": "12"
            }
        ],
        "ReportedName": "ubuntu",
        "ReportedVersion": "18.04"
    },
    {
        "Name": "Ubuntu 20.04 LTS",
        "Id": "Ubuntu 20.04 LTS",
        "Type": "Linux",
        "ConfigurationManagers": [
            {
                "Name": "Chef",
                "Version": "12"
            }
        ],
        "ReportedName": "ubuntu",
        "ReportedVersion": "20.04"
```

```
        },
        {
            "Name": "Custom",
            "Id": "Custom",
            "Type": "Linux",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12"
                },
                {
                    "Name": "Chef",
                    "Version": "11.10"
                },
                {
                    "Name": "Chef",
                    "Version": "11.4"
                },
                {
                    "Name": "Chef",
                    "Version": "0.9"
                }
            ]
        },
        {
            "Name": "CustomWindows",
            "Id": "CustomWindows",
            "Type": "Windows",
            "ConfigurationManagers": [
                {
                    "Name": "Chef",
                    "Version": "12.2"
                }
            ]
        }
    ]
}
```

主題

- Linux 作業系統
- Microsoft Windows Server

Linux 作業系統

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 支援下列 Linux 作業系統的 64 位元版本。

- Amazon Linux 和 Amazon Linux 2 （請參閱 AWS OpsWorks Stacks 主控台以取得目前支援的版本）
- Ubuntu 20.04 LTS
- CentOS 7 (CentOS 7)
- Red Hat Enterprise Linux 7

您也可以根據這些作業系統使用自訂 AMI。

下列為 Linux 執行個體的一般注意事項：

支援的套件版本

支援的套件 (例如 Ruby) 版本和修補程式層級，取決於下列各節中所述的作業系統和版本。

更新

根據預設， AWS OpsWorks Stacks 會透過自動呼叫執行個體`yum update`或在執行個體開機`apt-get update`之後，確保 Linux 執行個體擁有最新的安全修補程式。若要停用自動更新，請使用 CreateInstance、UpdateInstance、CreateLayer 或 UpdateLayer 動作，或同等的 AWS SDK 方法或 AWS CLI 命令，將 `InstallUpdatesOnBoot` 參數設定為 `false`。

為了避免服務中斷， AWS OpsWorks Stacks 不會在執行個體上線後自動安裝更新。您可以執行 Upgrade Operating System (升級作業系統) 堆疊命令，隨時手動更新線上執行個體的作業系統。如需如何管理安全性更新的詳細資訊，請參閱管理安全性更新。

如需進一步了解 AWS OpsWorks Stacks 如何更新執行個體，請根據其中一個支援的作業系統建立自訂 AMI。例如，您可以使用自訂 AMI 來指定要在執行個體上安裝哪些套件版本。每個 Linux 發行版本都有不同的支援時程和套件合併政策，因此您應該考慮哪種方法最符合您的需求。如需詳細資訊，請參閱使用自訂 AMI。

主機檔案

每個線上執行個體都有一個將 IP 地址映射到主機名稱/etc/hosts的檔案。 AWS OpsWorks Stacks 包含每個執行個體hosts檔案中堆疊所有線上執行個體的公有和私有地址。例如，假設您有一個堆疊，其中包含兩個 Node.js App Server 執行個體，Nodejs-app1 和 nodejs-app2，以及一個 MySQL 執行個體，db-master1。nodejs-app1 執行個體的 hosts 檔案看起來如下列範例，而另一個執行個體則會有類似的 hosts 檔案。

```
...
# OpsWorks Layer State
192.0.2.0 nodejs-app1.localdomain nodejs-app1
10.145.160.232 db-master1
198.51.100.0 db-master1-ext
10.243.77.78 nodejs-app2
203.0.113.0 nodejs-app2-ext
10.84.66.6 nodejs-app1
192.0.2.0 nodejs-app1-ext
```

AWS OpsWorks Stacks 代理程式代理支援

Chef 11.10 AWS OpsWorks 和更新版本的 Stacks 代理程式包含代理伺服器的基本支援，通常與隔離VPCs 搭配使用。若要啟用代理伺服器支援，執行個體必須具備 /etc/environment 檔案以提供適當的 HTTP 和 HTTPS 流量設定。此檔案應該會如下列所示；請將反白的文字取代為您的代理伺服器 URL 和連接埠：

```
http_proxy="http://myproxy.example.com:8080/"
https_proxy="http://myproxy.example.com:8080/"
no_proxy="169.254.169.254"
```

若要啟用代理支援，我們建議您建立自訂 AMI，以包含適當的 /etc/environment 檔案，並使用該 AMI 來建立您的執行個體。

> **Note**
>
> 我們不建議使用自訂配方在您的執行個體上建立/etc/environment檔案。在任何自訂配方執行之前， AWS OpsWorks Stacks 需要在設定程序的早期取得代理伺服器資料。

主題

- Amazon Linux
- Ubuntu LTS
- CentOS
- Red Hat Enterprise Linux

Amazon Linux

AWS OpsWorks Stacks 支援 Amazon Linux 和 Amazon Linux 2 的 64 位元版本。除了定期更新和修補程式，Amazon Linux 大約每六個月會發行新版本，其中涉及大量變更。當您建立堆疊或新執行個體時，您必須指定要使用的 Amazon Linux 版本。當 AWS 發行新版本時，您的執行個體會繼續執行指定的版本，直到您明確變更版本為止。當新的 Amazon Linux 版本發行之後，會有四週的遷移期間，在這段期間，AWS 會持續提供舊版本的定期更新。遷移期間結束之後，您的執行個體可以繼續執行舊版本，但 AWS 不會提供進一步的更新。如需詳細資訊，請參閱 Amazon Linux AMI 常見問答集。

當新的 Amazon Linux 版本發行之後，建議您在遷移期間內更新至新版本，以讓執行個體持續接收安全性更新。在您更新生產堆疊的執行個體之前，建議您先啟動新的執行個體，並確認應用程式可在新版本上正確執行。接著，您即可更新生產堆疊的執行個體。

> **Note**
>
> 根據預設，當新的 Amazon Linux 版本發行時，以其為依據的自訂 AMI 會自動更新到該版本。建議作法是將自訂 AMI 鎖定為特定 Amazon Linux 版本，以便將更新延遲直到您測試過新版本為止。如需詳細資訊，請參閱如何將 AMI 鎖定為特定版本？。
> 如果您使用 AWS CloudFormation 範本建立具有執行 Amazon Linux 之執行個體的堆疊，範本應明確指定 Amazon Linux 版本。尤其是，若您的範本指定 `Amazon Linux`，執行個體會持續執行 2016.09 版本。如需詳細資訊，請參閱 AWS::OpsWorks::Stack 及 AWS::OpsWorks::Instance。

若要更新執行個體的 Amazon Linux 版本，請執行下列其中一項作業：

- 針對線上執行個體，執行 Upgrade Operating System (升級作業系統) 堆疊命令。

  當新的 Amazon Linux 版本可用時，Instances (執行個體) 和 Stack (堆疊) 頁面會顯示一則通知與連結，以帶您前往 Run Command (執行命令) 頁面。然後，您可以執行 Upgrade Operating System (升級作業系統) 以升級執行個體。

- 對於離線 Amazon Elastic Block Store-backed (EBS-backed) 執行個體，啟動執行個體並執行升級作業系統，如上述陳述式所述。
- 針對離線的執行個體存放區後端執行個體 (包括時間式和負載式執行個體)，請編輯執行個體的 Operating system (作業系統) 設定以指定新的版本。

  AWS OpsWorks Stacks 會在執行個體重新啟動時，自動將執行個體更新為新版本。

Amazon Linux：支援的 Node.js 版本

Amazon Linux 版本	Node.js 版本
2	(Not applicable to operating systems that are available for Chef 12 and higher stacks only)
2018.03	0.12.18
2017.09	0.12.18
2017.03	0.12.18
2016.09	0.12.18 0.12.17 0.12.16 0.12.15
2016.03	0.12.18 0.12.17 0.12.16 0.12.15 0.12.14 0.12.13 0.12.12 0.12.10

Amazon Linux：支援的 Chef 版本

Chef 版本	支援的 Amazon Linux 版本
12	Amazon Linux 2 Amazon Linux 2018.03 Amazon Linux 2017.09 Amazon Linux 2017.03 Amazon Linux 2016.09 Amazon Linux 2016.03
11.10	Amazon Linux 2018.03 Amazon Linux 2017.09 Amazon Linux 2017.03 Amazon Linux 2016.09 Amazon Linux 2016.03
11.4 (deprecated)	Amazon Linux 2016.09 Amazon Linux 2016.03

Important

更新 t1.micro 執行個體之前，請確認它們具備 `/var/swapfile` 暫時置換檔。Chef 0.9 堆疊上的 t1.micro 執行個體不具備置換檔。若是 Chef 11.4 和 Chef 11.10 堆疊，最新版本的執行個體代理程式會自動為 t1.micro 執行個體建立置換檔。不過，這項變更已推出數週，因此如果執行個體大約是在 2014 年 3 月 24 日前建立，您應該檢查其中是否具備 `/var/swapfile`。
如果 t1.micro 執行個體缺少置換檔，您可以建立一個置換檔，如下所示：

- 若是 Chef 11.10 和更新版本的堆疊，請建立新的 t1.micro 執行個體，其中即會自動含有置換檔。
- 若是 Chef 0.9 堆疊，請以根使用者的身分在每個執行個體上執行下列命令。

```
dd if=/dev/zero of=/var/swapfile bs=1M count=256
 mkswap /var/swapfile
 chown root:root /var/swapfile
 chmod 0600 /var/swapfile
 swapon /var/swapfile
```

如果您不想建立新的執行個體，也可以在 Chef 11.10 和更新版本的堆疊上使用這些命令。

Ubuntu LTS

Ubuntu 約每兩年會發行新的 Ubuntu LTS 版本，每個版本約支援 5 年。Ubuntu 會在作業系統支援期間提供安全性修補程式和更新。如需詳細資訊，請參閱 LTS - Ubuntu Wiki。

- 您不能將現有 Ubuntu 執行個體更新到較新版本的 Ubuntu。

  您必須建立新的 Ubuntu 執行個體，並刪除較舊的執行個體。
- Ubuntu 20.04 LTS 僅支援 Chef 12 和更新版本的堆疊。

CentOS

AWS OpsWorks Stacks 支援 64 位元版本的 CentOS 7。最初支援的版本是 CentOS 7，而 CentOS 約每兩年會發行新版本。

當您在 CentOS 堆疊中啟動新的執行個體時， AWS OpsWorks Stacks 會自動安裝最新的 CentOS 版本。由於 Stacks AWS OpsWorks 不會在發行新的 CentOS 次要版本時自動更新現有執行個體上的作業系統，因此新建立的執行個體可能會收到比堆疊現有執行個體更新的版本。為了保持堆疊間的版本一致性，您可以將現有的執行個體更新至目前的 CentOS 版本，如下所示：

- 針對線上執行個體，請執行 Upgrade Operating System (升級作業系統) 堆疊命令，其會在指定執行個體上執行 `yum update` 以將其更新至目前的版本。

  當新的 CentOS 7 次要版本可用時，Instances (執行個體) 和 Stack (堆疊) 頁面會顯示一則通知與連結，以帶您前往 Run Command (執行命令) 頁面。然後，您可以執行 Upgrade Operating System (升級作業系統) 以升級執行個體。
- 對於離線 Amazon EBS 後端執行個體，啟動執行個體並執行升級作業系統，如上述清單項目所述。
- 對於離線執行個體後端執行個體， AWS OpsWorks Stacks 會在執行個體重新啟動時自動安裝新版本。

CentOS：支援的 Chef 版本

Chef 版本	支援的 CentOS 版本
12	CentOS 7
11.10	(None supported)
11.4 (deprecated)	(None supported)

> **Note**
>
> AWS OpsWorks Stacks 支援適用於 CentOS 執行個體的 Apache 2.4。

Red Hat Enterprise Linux

AWS OpsWorks Stacks 支援 64 位元版本的 Red Hat Enterprise Linux 7 (RHEL 7)。最初支援的版本是 RHEL 7.1，而 Red Hat 約每九個月會發行新的次要版本。次要版本應與 RHEL 7.0 相容。如需詳細資訊，請參閱生命週期和更新政策。

當您啟動新的執行個體時， AWS OpsWorks Stacks 會自動安裝目前的 RHEL 7 版本。由於 Stacks AWS OpsWorks 不會在發行新的 RHEL 7 次要版本時自動更新現有執行個體上的作業系統，因此新建立的執行個體可能會收到比堆疊現有執行個體更新的版本。為了保持堆疊間的版本一致性，您可以將現有的執行個體更新至目前的 RHEL 7 版本，如下所示：

- 針對線上執行個體，請執行 Upgrade Operating System (升級作業系統) 堆疊命令，其會在指定執行個體上執行 `yum update` 以將其更新至目前的版本。

  當新的 RHEL 7 次要版本可用時，Instances (執行個體) 和 Stack (堆疊) 頁面會顯示一則通知與連結，以帶您前往 Run Command (執行命令) 頁面。然後，您可以執行 Upgrade Operating System (升級作業系統) 以升級執行個體。
- 對於離線 Amazon EBS 後端執行個體，啟動執行個體並執行升級作業系統，如上述清單項目所述。
- 對於離線執行個體後端執行個體， AWS OpsWorks Stacks 會在執行個體重新啟動時自動安裝新版本。

Red Hat Enterprise Linux：支援的 Node.js 版本

RHEL 版本	Node.js 版本
7	(Node.js versions only apply to Chef 11.10 stacks) 0.8.19 0.8.26 0.10.11 0.10.21 0.10.24 0.10.25 0.10.27 0.10.29 0.10.40 0.12.10 0.12.12 0.12.13 0.12.15

Red Hat Enterprise Linux：支援的 Chef 版本

Chef 版本	支援的 RHEL 版本
12	Red Hat Enterprise Linux 7
11.10	Red Hat Enterprise Linux 7
11.4 (deprecated)	(None supported)

所有超過 0.10.40 的 Node.js 版本都已棄用。 0.12.7 和 0.12.9 也已棄用。

> **Note**
>
> AWS OpsWorks Stacks 支援適用於 RHEL 7 執行個體的 Apache 2.4。

Microsoft Windows Server

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

下列備註說明 Stacks 對 Windows AWS OpsWorks 執行個體的支援。Windows 執行個體僅適用於 Chef 12.2 堆疊。Windows 堆疊中的 Chef 明確版本為 12.22。

目前，Stacks AWS OpsWorks 代理程式無法安裝在使用英文 - 美國 (en-US) 以外的系統使用者介面語言的 Windows 執行個體上，且 AWS OpsWorks Stacks 無法管理。

版本

AWS OpsWorks Stacks 支援下列 Windows 64 位元版本：

- Microsoft Windows Server 2022 基礎
- Microsoft Windows Server 2022 搭配 SQL Server Express
- Microsoft Windows Server 2022 搭配 SQL Server Standard
- Microsoft Windows Server 2022 搭配 SQL Server Web
- Microsoft Windows Server 2019 基礎
- Microsoft Windows Server 2019 搭配 SQL Server Express
- Microsoft Windows Server 2019 搭配 SQL Server Standard
- Microsoft Windows Server 2019 搭配 SQL Server Web

建立執行個體

您可以使用 Stacks AWS OpsWorks 主控台、API 或 CLI 建立 Windows 執行個體。Windows 執行個體為 Amazon EBS 後端，但您無法掛載額外的 Amazon EBS 磁碟區。

Windows 堆疊可以使用全年無休執行個體；您可以手動將其啟動和停止。也可以使用時間式自動擴展功能，根據使用者指定的排程自動啟動和停止執行個體。以 Windows 為基礎的堆疊無法使用負載式自動擴展功能。

您無法向堆疊註冊在 Stacks 外部建立的 Windows 執行個體。 AWS OpsWorks

更新

AWS 會更新 Windows AMI 的每組修補程式，因此當您建立執行個體時，執行個體即具備最新的更新。不過， AWS OpsWorks Stacks 不提供將更新套用至線上 Windows 執行個體的方法。若要確保 Windows 為最新狀態，最簡單的方法是定期取代您的執行個體，讓它們始終執行最新的 AMI。

層

若要處理安裝軟體、設定軟體或部署應用程式等任務，您需要使用自訂配方實作一或多個自訂 layer。

Chef

Windows 執行個體會使用 Chef 12.22 並執行本機模式中的 chef-client，以啟動名稱為 chef-zero 的本機記憶體內 Chef 伺服器。此伺服器的存在可讓自訂配方使用 Chef 搜尋和資料包。

遠端登入

AWS OpsWorks Stacks 提供授權的 IAM 使用者一組密碼，可用來登入 Windows 執行個體。此密碼會在指定的時間後過期。管理員可以使用 SSH 金鑰對擷取執行個體的管理員密碼，該密碼提供不受限制的 RDP 存取權。如需詳細資訊，請參閱使用 RDP 登入。

AWS 開發套件

AWS OpsWorks Stacks 會自動在每個執行個體適用於 .NET 的 AWS SDK上安裝 。此套件包含 AWS .NET 程式庫和適用於 Windows 的 AWS 工具 (包括 適用於 PowerShell 的 AWS 工具)。若要使用 Ruby 軟體開發套件，您可以使用自訂配方來安裝適當的 Gem 套件。

監控與指標

Windows 執行個體支援標準 Amazon CloudWatch (CloudWatch) 指標，您可以在 CloudWatch 主控台中檢視。

Ruby

Stacks 在 Windows 執行個體上安裝的 Chef 12.22 AWS OpsWorks 用戶端隨附 Ruby 2.3.6。不過， AWS OpsWorks Stacks 不會將可執行檔的目錄新增至 PATH 環境變數。若要讓應用程式使用此 Ruby 版本，您通常可以在 `C:\opscode\chef\embedded\bin\` 中找到它。

AWS OpsWorks Stacks 代理程式 CLI

Windows 執行個體上的 AWS OpsWorks Stacks 代理程式不會公開命令列界面。

代理支援

若要設定 Windows 執行個體的代理支援，請執行下列作業：

1. 修改 `machine.config` 以新增下列項目，將代理支援新增至 Windows PowerShell （初始引導） 和 .NET (AWS OpsWorks Stacks 代理程式） 應用程式：

```
<system.net>
  <defaultProxy>
    <proxy autoDetect="false" bypassonlocal="true"
 proxyaddress="http://10.100.1.91:3128"  usesystemdefault="false" />
    <bypasslist>
      <add address="localhost" />
      <add address="169.254.169.254" />
    </bypasslist>
  </defaultProxy>
</system.net>
```

2. 執行下列命令來設定環境變數，供 Chef 和 Git 日後使用：

```
setx /m no_proxy "localhost,169.254.169.254"
setx /m http_proxy "http://10.100.1.91:3128"
setx /m https_proxy "http://10.100.1.91:3128"
```

> **Note**
>
> 如需進一步了解 AWS OpsWorks Stacks 如何更新執行個體，請建立以 Microsoft Windows Server 2022 Base 為基礎的自訂 AMI。例如，您可以使用自訂 AMI 來指定要在執行個體上安裝哪些軟體，例如 Web 伺服器 (IIS)。如需詳細資訊，請參閱使用自訂 AMI。

將執行個體新增至 Layer

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

在您建立 layer 之後，通常會新增至少一個執行個體。如果目前集合無法處理負載，則您稍後可以新增多個執行個體。您也可以使用負載類型或時間類型執行個體自動擴展執行個體數目。

您可以將新的或現有執行個體新增至 layer：

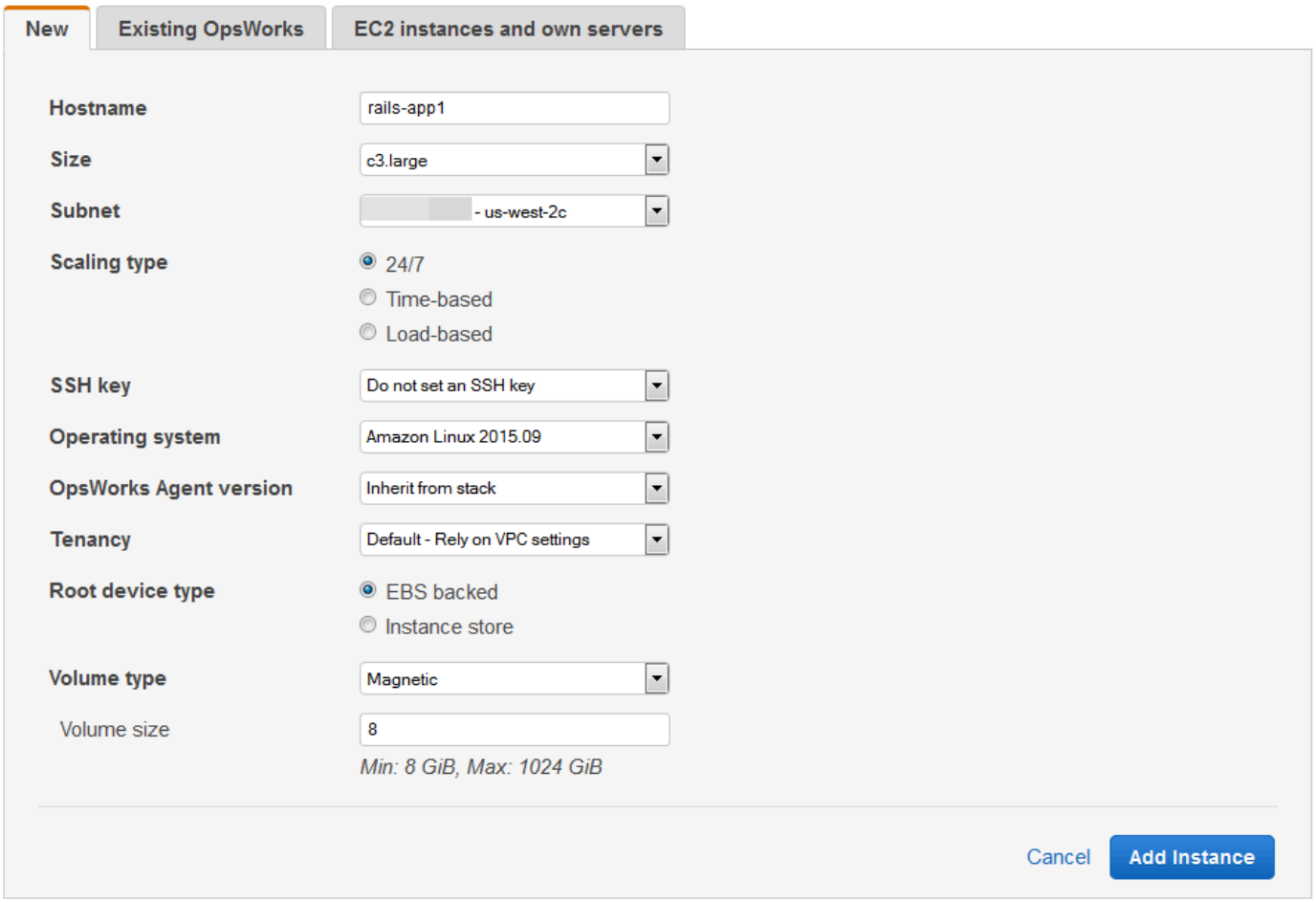
- New–OpsWorks 會建立新的執行個體，並根據您的規格設定，使其成為 layer 的成員。
- 現有 – 您可以從任何相容的 layer 新增現有執行個體，但必須處於離線（已停止）狀態。

如果執行個體屬於多 layer，則 AWS OpsWorks Stacks 會在發生生命週期事件時，或執行 stack 或 deployment 命令時，執行執行個體每個 layer 的配方。

您也可以編輯執行個體的組態，讓執行個體成為多 layer 的成員。如需詳細資訊，請參閱編輯執行個體組態。

將新的執行個體新增至 layer

1. 在 Instances (執行個體) 頁面上，選擇適當 layer 的 +Instance (+執行個體)，並視需要選擇 New (新建) 標籤。如果您不只是想要設定 Host name (主機名稱)、Size (大小) 和 Subnet (子網路) 或 Availability Zone (可用區域)，則請選擇 Advanced >> (進階 >>) 查看其他選項。以下顯示一組完整的選項：



2. 如有需要，您可以覆寫預設組態，而其中大多數都是您在建立堆疊時指定。如需詳細資訊，請參閱建立新的堆疊。

   Hostname (主機名稱)

   識別網路上的執行個體。根據預設， AWS OpsWorks Stacks 會使用您在建立堆疊時指定的主機名稱主題來產生每個執行個體的主機名稱。您可以覆寫此值，並指定您慣用的主機名稱。

   大小

   Amazon EC2 執行個體類型，可指定執行個體的資源，例如記憶體數量或虛擬核心數量。AWS OpsWorks Stacks 會指定每個執行個體的預設大小，您可以使用您偏好的執行個體類型覆寫。

   Stacks AWS OpsWorks 支援的執行個體類型取決於堆疊是否位於 VPC 中。若您的帳戶使用 AWS 免費方案，執行個體類型也會受限。下拉式 Size (大小) 清單會顯示您堆疊所支援 Chef 版本的支援執行個體類型。請注意，微型執行個體 (例如 t1.micro) 的資源可能不足，無法支援一些 layer。如需詳細資訊，請參閱 執行個體類型。

   > **Note**
   >
   > 如果您使用負載平衡執行個體，請注意，設定生命週期事件可以產生可能持續一分鐘或更久的重要 CPU 負載峰值。如果執行個體較小，則此負載峰值就足以觸發向上擴展，特別是具有頻繁設定事件的大型負載平衡堆疊。下列一些方法可以減少設定事件造成不必要向上調整的可能性。
   >
   > - 使用較大的執行個體，因此，設定事件的額外負載不足以觸發向上調整。
   > - 不使用執行個體類型 (例如共享 CPU 資源的 T2)。
   >
   >   這確保在發生設定事件時，執行個體的所有 CPU 資源都立即可用。
   > - 讓 `exceeded threshold` 時間遠大於處理設定事件所需的時間，可能是 5 分鐘。
   >
   >   如需詳細資訊，請參閱使用自動負載型擴展。

   Availability Zone/Subnet (可用區域/子網路)

   如果堆疊不在 VPC 中，則此設定的標籤為 Availability Zone (可用區域)，並列出區域 (region) 的區域 (zone)。您可以使用此設定來覆寫您在建立堆疊時所指定的預設可用區域。

如果堆疊正在 VPC 中執行，則此設定的標籤為 Subnet (子網路)，並列出 VPC 的子網路。您可以使用此設定來覆寫您在建立堆疊時所指定的預設子網路。

> **Note**
>
> 根據預設， AWS OpsWorks Stacks 會列出子網路的 CIDR 範圍。若要讓清單更易讀，請使用 VPC 主控台或 API，將標籤新增至每個子網路，並將金鑰設為 **Name**，並將值設為子網路的名稱。 AWS OpsWorks Stacks 會將該名稱附加至 CIDR 範圍。在上述範例中，子網路的 Name (名稱) 標籤已設為 **Private**。

Scaling Type (擴展類型)

判斷執行個體的啟動和停止方式。

- 預設值為 24/7 (全年無休) 執行個體，您可以手動將其啟動和停止。
- AWS OpsWorks Stacks 會根據指定的排程啟動和停止以時間為基礎的執行個體。
- （僅限 Linux) AWS OpsWorks Stacks 會根據指定的負載指標啟動和停止負載型執行個體。

> **Note**
>
> 您無法自行啟動或停止負載類型或時間類型執行個體。反之，您可以設定執行個體，而 AWS OpsWorks Stacks 會根據組態啟動和停止執行個體。如需詳細資訊，請參閱使用時間型和負載型執行個體管理負載。

SSH 金鑰

Amazon EC2 金鑰對。 AWS OpsWorks Stacks 會在執行個體上安裝公有金鑰。

- 針對 Linux 執行個體，您可以搭配使用對應的私有金鑰與 SSH 用戶端來登入執行個體。
- 針對 Windows 執行個體，您可以使用對應的私有金鑰來擷取執行個體的管理員密碼。您接著可以搭配使用該密碼與 RDP，以管理員身分登入執行個體。

一開始，此設定是您在建立堆疊時所指定的 Default SSH key (預設 SSH 金鑰) 值。

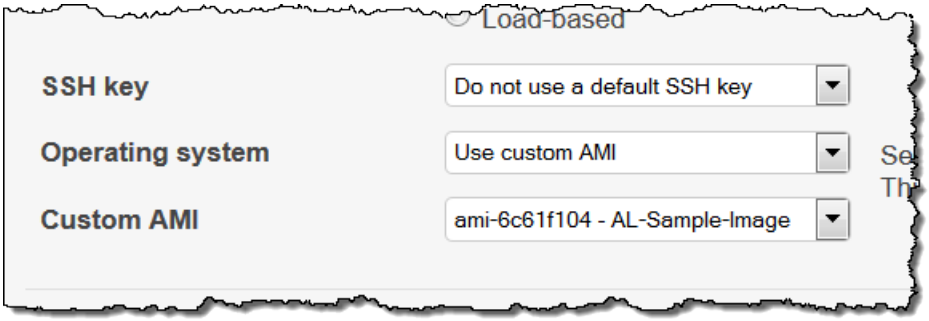
- 如果預設值設為不使用預設 SSH 金鑰，您可以指定帳戶的其中一個 Amazon EC2 金鑰。
- 如果預設值設為 Amazon EC2 金鑰，您可以指定不同的金鑰或無金鑰。

作業系統

作業系統指定執行個體正在執行的作業系統。 AWS OpsWorks Stacks 僅支援 64 位元作業系統。

一開始，此設定是您在建立堆疊時所指定的 Default operating system (預設作業系統) 值。您可以覆寫預設值，以指定不同的 Linux 作業系統或自訂 Amazon Machine Image (AMI)。不過，您無法從 Linux 切換至 Windows，或從 Windows 切換至 Linux。

如果您選取 Use custom AMI (使用自訂 AMI)，則此頁面會顯示自訂 AMI 清單，而非 Architecture (架構) 和 Root device type (根設備類型)。



如需詳細資訊，請參閱使用自訂 AMI。

OpsWorks Agent version (OpsWorks 代理程式版本)

OpsWorks 代理程式版本會指定您要在執行個體上執行的 AWS OpsWorks Stacks 代理程式版本。如果您希望 AWS OpsWorks Stacks 自動更新代理程式，請選擇 Inherit from stack (繼承自堆疊)。若要安裝代理程式的特定版本，並手動更新執行個體上的代理程式，請從下拉式清單中選擇版本。

> **Note**
>
> 並非所有代理程式版本都會使用所有作業系統版本。如果您的執行個體正在執行代理程式，或您在執行個體作業系統上未完全支援的執行個體上安裝代理程式，Stacks AWS OpsWorks 主控台會顯示錯誤訊息，指示您安裝相容的代理程式。

租用

選擇您執行個體的租用選項。您可以選擇在專供您使用的實體伺服器上執行執行個體。

- Default - Rely on VPC settings (預設 - 依賴 VPC 設定)。無租用，或繼承您 VPC 中的租用設定。
- Dedicated - Run a dedicated instance (專用 - 執行專用執行個體)。依時數為單一租戶硬體上執行的執行個體付費。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的專用執行個體，以及 Amazon EC2 專用執行個體。
- Dedicated host - Run this instance on a dedicated host (專用主機 - 在專用主機上執行此執行個體)。付費使用專供您執行執行個體的實體主機，並使用您現有的每個通訊端、每個核心或每個 VM 軟體的授權，以降低成本。如需詳細資訊，請參閱 Amazon EC2 文件中的專用主機概觀，以及 Amazon EC2 專用主機。

Root device type (根設備類型)

指定執行個體的根設備儲存體。

- Linux 執行個體可以是 Amazon EBS 後端或執行個體存放區後端。
- Windows 執行個體必須為 Amazon EBS 後端。

如需詳細資訊，請參閱儲存體。

> **Note**
>
> 初次開機後，Amazon EBS 後端執行個體開機速度會比執行個體存放區後端執行個體更快，因為 AWS OpsWorks Stacks 不需要從頭重新安裝執行個體的軟體。如需詳細資訊，請參閱根設備儲存。

磁碟區類型

指定根設備磁碟區類型：Magnetic (磁性)、Provisioned IOPS (SSD) (佈建 IOPS (SSD)) 或 General Purpose (SSD) (一般用途 (SSD))。如需詳細資訊，請參閱 Amazon EBS 磁碟區類型。

磁碟區大小

指定所指定磁碟區類型的根設備磁碟區大小。如需詳細資訊，請參閱 Amazon EBS 磁碟區類型。

- General Purpose (SSD) (一般用途 (SSD))。最小允許大小為 8 GiB，最大大小為 16384 GiB。

- Provisioned IOPS (SSD) (佈建 IOPS (SSD))。最小允許大小為 8 GiB，最大大小為 16384 GiB。您可以設定每秒最少 100 個輸入/輸出操作 (IOPS)，最多則為 240 個 IOPS。
- 磁帶. 最小允許大小為 8 GiB，最大大小為 1024 GiB。

3. 選擇 Add Instance (新增執行個體) 建立新的執行個體。

> **Note**
>
> 當您建立執行個體時，無法覆寫堆疊預設代理程式版本設定。若要指定自訂代理程式版本設定，您必須建立執行個體，然後編輯其組態。

將現有執行個體新增至 layer

1. 在 Instances (執行個體) 頁面上，選擇適當 layer 的 +Instance (+執行個體)，然後開啟 Existing (現有) 標籤。

   > **Note**
   >
   > 如果您不想要使用現有執行個體，則請選擇 New (新建) 建立新的執行個體，如之前程序中所述。

2. 在 Existing (現有) 標籤上，從清單中選取執行個體。
3. 選擇 Add Instance (新增執行個體) 建立新的執行個體。

執行個體代表 Amazon EC2 執行個體，但基本上只是 Stacks AWS OpsWorks 資料結構。執行個體必須開始建立執行中的 Amazon EC2 執行個體，如以下各節所述。

> **Important**
>
> 如果您將執行個體啟動到預設 VPC，則必須小心地修改 VPC 組態。執行個體必須一律能夠與 AWS OpsWorks Stacks 服務、Amazon S3 和套件儲存庫通訊。例如，如果您移除預設閘道，執行個體將失去與 Stacks AWS OpsWorks 服務的連線，然後會將執行個體視為失敗並自動修復。不過， AWS OpsWorks Stacks 將無法在修復的執行個體上安裝執行個體代理程式。如果沒有代理程式，則執行個體無法與服務通訊，而且啟動程序無法離開 `booting` 狀態。如需預設 VPC 的詳細資訊，請參閱支援的平台。

您也可以將 Linux 運算資源納入在 Stacks AWS OpsWorks 外部建立的堆疊：

- 您使用 Amazon EC2 主控台、CLI 或 API 直接建立的 Amazon EC2 執行個體。
- 在您自己的硬體上執行的「現場部署」執行個體，包括在虛擬機器中執行的執行個體。

如需詳細資訊，請參閱使用在 AWS OpsWorks Stacks 之外建立的運算資源。

使用自訂 AMI

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 支援兩種自訂執行個體的方式：自訂 Amazon Machine Image AMIs) 和 Chef 配方。這兩種方法都可讓您控制所安裝的套件和套件版本、其設定方式，以此類推。不過，每種方法都有不同的優點，因此最好的方法取決於您的需求。

以下是考慮使用自訂 AMI 的主要原因：

- 您想要預先綁定特定套件，而不是在啟動執行個體之後安裝它們。
- 您想要控制套件更新的時間，以提供 layer 的一致基礎映像。
- 您想要盡快啟動執行個體 (特別是負載類型執行個體)。

以下是考慮使用 Chef 配方的主要原因：

- 它們比自訂 AMI 更具彈性。
- 它們更容易更新。
- 它們可以在執行中執行個體上執行更新。

實際上，最佳解決方案可能是這兩種方法的組合。如需配方的詳細資訊，請參閱技術指南和配方。

主題

- 自訂 AMIs 如何使用 AWS OpsWorks Stacks
- 為 AWS OpsWorks Stacks 建立自訂 AMI

自訂 AMIs 如何使用 AWS OpsWorks Stacks

若要為執行個體指定自訂 AMI，請選取在建立新執行個體時，使用自訂 AMI 做為執行個體的作業系統。 AWS OpsWorks 堆疊接著會顯示堆疊區域中的自訂 AMIs 清單，並從清單中選擇適當的 AMI。如需詳細資訊，請參閱將執行個體新增至 Layer。

> **Note**
>
> 您無法將特定自訂 AMI 指定為堆疊的預設作業系統。您可以將 `Use custom AMI` 設定為堆疊的預設作業系統，但只有在將新的執行個體新增至 layer 時才能指定特定 AMI。如需詳細資訊，請參閱 將執行個體新增至 Layer 和 建立新的堆疊。雖然可以使用其他作業系統 (例如 CentOS 6.x) (從自訂或社群產生的 AMI 所建立) 建立執行個體，但並未正式受到支援。

本主題討論您在建立或使用自訂 AMI 之前應該考慮的一些一般問題。

主題

- 啟動行為
- 選擇 Layer
- 處理應用程式

啟動行為

當您啟動執行個體時， AWS OpsWorks Stacks 會使用指定的自訂 AMI 來啟動新的 Amazon EC2 執行個體。 AWS OpsWorks Stacks 接著會使用 cloud-init 在執行個體上安裝 AWS OpsWorks Stacks 代理程式，代理程式會執行執行個體的設定配方，後面接著部署配方。執行個體上線之後，代理程式會執行堆疊中每個執行個體的設定配方 (包括新增的執行個體)。

選擇 Layer

Stacks AWS OpsWorks 代理程式通常不會與已安裝的套件衝突。不過，執行個體必須至少是一個 layer 的成員。 AWS OpsWorks Stacks 一律會執行該 layer 的配方，這可能會導致問題。您應該確切了解 layer 的配方對執行個體所執行的作業，再將具有自訂 AMI 的執行個體新增至 layer。

若要查看特定 layer 類型在您執行個體上執行的配方，請開啟包含該 layer 的堆疊。然後按一下導覽窗格中的 Layers (Layer)，然後按一下感興趣 layer 的 Recipes (配方)。若要查看實際程式碼，請按一下配方名稱。

> **Note**
>
> 對於 Linux AMIs，降低衝突可能性的一種方法是使用 AWS OpsWorks Stacks 來佈建和設定執行個體，這是自訂 AMI 的基礎。如需詳細資訊，請參閱從 Stacks AWS OpsWorks 執行個體建立自訂 Linux AMI。

處理應用程式

除了套件之外，也建議您在 AMI 中包括應用程式。如果您有大型複雜應用程式，則將它包括在 AMI 中可以縮短執行個體的啟動時間。您可以在 AMI 中包含小型應用程式，但相對於讓 AWS OpsWorks Stacks 部署應用程式，通常沒有或很少的時間優勢。

其中一個選項是將應用程式包括在 AMI 中，同時建立應用程式，以將應用程式從儲存庫部署至執行個體。這種方法可縮短開機時間，同時提供方便的方法以在執行個體執行之後更新應用程式。請注意，Chef 配方為等冪操作，因此只要儲存庫中的版本與執行個體上版本相同，部署配方就不會修改應用程式。

為 AWS OpsWorks Stacks 建立自訂 AMI

若要搭配 Stacks AWS OpsWorks 使用自訂 AMI，您必須先從自訂執行個體建立 AMI。您可以從兩個選項中選擇：

- 使用 Amazon EC2 主控台或 API，根據其中一個 AWS OpsWorks Stacks 支援的 AMIs 的 64 位元版本來建立和自訂執行個體。
- 對於 Linux AMIs，請使用 OpsWorks 根據其關聯層的組態來建立 Amazon EC2 執行個體。

建立自訂 Linux AMI 之前，請在`/tmp`分割區`noexec`上停用 ，以允許 AWS OpsWorks Stacks 在自訂 Linux 執行個體上安裝其代理程式。

> **Note**
>
> 請注意，AMI 可能無法使用所有執行個體類型，因此請確定您的啟動中 AMI 與您計劃使用的執行個體類型相容。具體而言，R3 執行個體類型需要硬體輔助虛擬化 (HVM) AMI。

然後，您可以使用 Amazon EC2 主控台或 API 從自訂執行個體建立自訂 AMI。將執行個體新增至 layer，並指定自訂 AMI，即可在相同區域的任何堆疊中使用自訂 AMI。如需如何建立利用自訂 AMI 之執行個體的詳細資訊，請參閱將執行個體新增至 Layer。

> **Note**
>
> 根據預設， AWS OpsWorks Stacks 會在開機時安裝所有 Amazon Linux 更新，為您提供最新版本。此外，Amazon Linux 大約每六個月會發行新版本，其中涉及大量變更。根據預設，當新的 Amazon Linux 版本發行時，以其為依據的自訂 AMI 會自動更新到該版本。建議作法是將自訂 AMI 鎖定為特定 Amazon Linux 版本，讓您可以將更新延遲到您測試過新版本為止。如需詳細資訊，請參閱如何將 AMI 鎖定為特定版本？。

主題

- 使用 Amazon EC2 建立自訂 AMI
- 從 Stacks AWS OpsWorks 執行個體建立自訂 Linux AMI
- 建立自訂 Windows AMI

使用 Amazon EC2 建立自訂 AMI

建立自訂 AMI 的最簡單方法是使用 Amazon EC2 主控台或 API 來執行整個任務，而 Windows AMIs 是唯一的選項。如需下列步驟的詳細資訊，請參閱建立自己的 AMI。

使用 Amazon EC2 主控台或 API 建立自訂 AMI

1. 使用其中一個 AWS OpsWorks Stacks 支援 AMI 的 64 位元版本，來建立執行個體。
2. 從步驟 1 自訂執行個體，方法是設定執行個體、安裝套件，以此類推。請記住，會根據 AMI 以在每個執行個體上重新產生您安裝的每個項目，因此請不要包括特定執行個體特有的項目。
3. 停止執行個體，並建立自訂 AMI。

從 Stacks AWS OpsWorks 執行個體建立自訂 Linux AMI

若要使用自訂的 AWS OpsWorks Stacks Linux 執行個體來建立 AMI，請注意 OpsWorks 建立的每個 Amazon EC2 執行個體都包含唯一的身分。如果您從此類執行個體建立自訂 AMI，則它包含該身分，且所有以 AMI 為基礎的執行個體都有相同的身分。為了確保根據您自訂 AMI 的執行個體具有唯一的身分，您必須先從自訂的執行個體移除身分，再建立 AMI。

從 Stacks AWS OpsWorks 執行個體建立自訂 AMI

1. 建立 Linux 堆疊，並新增一或多 layer，以定義自訂執行個體的組態。您可以使用內建 layer、適當地自訂，以及完全自訂 layer。如需詳細資訊，請參閱自訂 AWS OpsWorks Stacks。

2. 編輯 layer 並停用 AutoHealing。
3. 新增具有您慣用 Linux 發行版本的執行個體至一或多 layer，並啟動它。建議使用 Amazon EBS 支援的執行個體。開啟執行個體的詳細資訊頁面，並記錄其 Amazon EC2 ID 以供稍後使用。
4. 執行個體上線時，請使用 SSH 登入，並根據執行個體作業系統執行接下來四個步驟之一。
5. 對於 Chef 11 或 Chef 12 堆疊中的 Amazon Linux 執行個體，或 Chef 11 堆疊中的 Red Hat Enterprise Linux 7 執行個體，執行下列動作。

   a. `sudo /etc/init.d/monit stop`

   b. `sudo /etc/init.d/opsworks-agent stop`

   c. `sudo rm -rf /etc/aws/opsworks/ /opt/aws/opsworks/ /var/log/aws/opsworks/ /var/lib/aws/opsworks/ /etc/monit.d/opsworks-agent.monitrc /etc/monit/conf.d/opsworks-agent.monitrc /var/lib/cloud/ /etc/chef`

   > **Note**
   >
   > 對於 Chef 12 堆疊中的執行個體，請將下列兩個資料夾新增至此命令：
   >
   > - `/var/chef`
   > - `/opt/chef`

   d. `sudo rpm -e opsworks-agent-ruby`

   e. `sudo rpm -e chef`

6. 用於 Chef 12 堆疊中的 Ubuntu 16.04 LTS 或 18.04 LTS 執行個體時，請執行下列動作。

   a. `sudo systemctl stop opsworks-agent`

   b. `sudo rm -rf /etc/aws/opsworks/ /opt/aws/opsworks/ /var/log/aws/opsworks/ /var/lib/aws/opsworks/ /etc/monit.d/opsworks-agent.monitrc /etc/monit/conf.d/opsworks-agent.monitrc /var/lib/cloud/ /var/chef /opt/chef /etc/chef`

   c. `sudo apt-get -y remove chef`

   d. `sudo dpkg -r opsworks-agent-ruby`

   e. `systemctl stop apt-daily.timer`

   f. `systemctl stop apt-daily-upgrade.timer`

   g. `rm /var/lib/systemd/timers/stamp-apt-daily.timer`

h. `rm /var/lib/systemd/timers/stamp-apt-daily-upgrade.timer`

7. 如需 Chef 12 堆疊中其他支援的 Ubuntu 版本，執行下列動作。

   a. `sudo /etc/init.d/monit stop`

   b. `sudo /etc/init.d/opsworks-agent stop`

   c. `sudo rm -rf /etc/aws/opsworks/ /opt/aws/opsworks/ /var/log/aws/opsworks/ /var/lib/aws/opsworks/ /etc/monit.d/opsworks-agent.monitrc /etc/monit/conf.d/opsworks-agent.monitrc /var/lib/cloud/ /var/chef /opt/chef /etc/chef`

   d. `sudo apt-get -y remove chef`

   e. `sudo dpkg -r opsworks-agent-ruby`

8. 對於 Chef 12 堆疊中的 Red Hat Enterprise Linux 7 執行個體，執行下列動作。

   a. `sudo systemctl stop opsworks-agent`

   b. `sudo rm -rf /etc/aws/opsworks/ /opt/aws/opsworks/ /var/log/aws/opsworks/ /var/lib/aws/opsworks/ /etc/monit.d/opsworks-agent.monitrc /etc/monit/conf.d/opsworks-agent.monitrc /var/lib/cloud/ /etc/chef /var/chef`

   c. `sudo rpm -e opsworks-agent-ruby`

   d. `sudo rpm -e chef`

9. 此步驟取決於執行個體類型：

   - 對於 Amazon EBS 後端執行個體，請使用 AWS OpsWorks Stacks 主控台停止執行個體並建立 AMI，如建立 Amazon EBS 後端 Linux AMI 中所述。
   - 對於執行個體後端執行個體，如建立執行個體後端 Linux AMI 所述建立 AMI，然後使用 AWS OpsWorks Stacks 主控台停止執行個體。

     當您建立 AMI 時，請務必包括憑證檔案。例如，您可以呼叫 `ec2-bundle-vol`命令，並將`-i`引數設為 `-i $(find /etc /usr /opt -name '*.pem' -o -name '*.crt' -o -name '*.gpg' | tr '\n' ',')`。請不要在綁定時移除 apt 公有金鑰。預設 `ec2-bundle-vol` 命令會處理此任務。

10. 返回 Stacks AWS OpsWorks 主控台並從堆疊刪除執行個體，以清除您的堆疊。

建立自訂 Windows AMI

下列程序會為 Windows Server 2022 Base 建立自訂 AMIs。您可以在 Amazon EC2 管理主控台中選擇其他 Windows Server 作業系統。

> **Important**
>
> 目前，Stacks AWS OpsWorks 代理程式無法安裝在使用英文 - 美國 (en-US) 以外的系統使用者介面語言的 Windows 執行個體上，且 AWS OpsWorks Stacks 無法管理。

主題

- 使用 Sysprep 建立自訂 Windows AMI
- 不使用 Sysprep 建立自訂 Windows AMI
- 使用自訂 Windows AMI 新增執行個體

使用 **Sysprep** 建立自訂 Windows AMI

使用 Sysprep 建立自訂 Windows AMI 通常會導致較慢的執行個體啟動但較乾淨的程序。第一次啟動從建立的映像建立的執行個體`Sysprep`需要更多時間，因為`Sysprep`活動、重新啟動、 AWS OpsWorks 堆疊佈建和第一個 Stacks AWS OpsWorks 執行，包括設定和組態。完成在 Amazon EC2 主控台中建立自訂 Windows AMI 的步驟。

使用 Sysprep 建立自訂 Windows AMI

1. 在 Amazon EC2 主控台中，選擇啟動執行個體。
2. 尋找 Microsoft Windows Server 2022 Base，然後選擇選取。
3. 選擇您想要的執行個體類型，然後選擇 Configure Instance Details (設定執行個體詳細資訊)。變更 AMI 的組態 (包括機器名稱、儲存體和安全群組設定)。選擇啟動。
4. 執行個體啟動程序完成之後，請取得密碼，然後在 Windows Remote Desktop Connection (遠端桌面連線) 視窗中連線至執行個體。
5. 在 Windows Start (開始) 畫面上，選擇 Start (開始)，然後開始輸入 **`ec2configservice`**，直到結果顯示 EC2ConfigServiceSettings 主控台為止。開啟 主控台。
6. 在一般索引標籤上，確定已填入啟用 UserData 執行核取方塊 （ 雖然 不需要此選項`Sysprep`，但 Stacks AWS OpsWorks 需要安裝其代理程式 ）。清除 Set the computer name of the instance... (設定執行個體的電腦名稱...) 選項的核取方塊，因為此選項可能會導致 AWS OpsWorks Stacks 的重新啟動迴圈。

7. 在映像索引標籤上，將管理員密碼設定為 Random，以允許 Amazon EC2 自動產生您可以使用 SSH 金鑰擷取的密碼，或指定 以指定您自己的密碼。 會Sysprep儲存此設定。如果您指定自己的密碼，請將密碼存放在便利的位置。建議您不要選擇 Keep Existing (保留現有)。
8. 選擇 Apply (套用)，然後選擇 Shutdown with Sysprep (使用 Sysprep 關機)。出現確認提示時，選擇 Yes (是)。
9. 執行個體停止後，在 Amazon EC2 主控台中，在執行個體清單中的執行個體上按一下滑鼠右鍵，選擇映像，然後選擇建立映像。
10. 在 Create Image (建立映像) 頁面上，提供映像的名稱和描述，並指定磁碟區組態。完成之後，請選擇 Create Image (建立映像)。
11. 開啟 Images (映像) 頁面，並等待映像從 pending (待定) 階段變更為 available (可用)。新的 AMI 現在已可使用。

不使用 **Sysprep** 建立自訂 Windows AMI

完成在 Amazon EC2 主控台中建立自訂 Windows AMI 的步驟。

不使用 Sysprep 建立自訂 Windows AMI

1. 在 Amazon EC2 主控台中，選擇啟動執行個體。
2. 尋找 Microsoft Windows Server 2022 Base，然後選擇選取。
3. 選擇您想要的執行個體類型，然後選擇 Configure Instance Details (設定執行個體詳細資訊)。變更 AMI 的組態 (包括機器名稱、儲存體和安全群組設定)。選擇啟動。
4. 執行個體啟動程序完成之後，請取得密碼，然後在 Windows Remote Desktop Connection (遠端桌面連線) 視窗中連線至執行個體。
5. 在執行個體上，開啟 `C:\Program Files\Amazon\Ec2ConfigService\Settings\config.xml`，並變更下列兩種設定，然後儲存和關閉檔案：
   - `Ec2SetPassword` 至 `Enabled`
   - `Ec2HandleUserData` 至 `Enabled`
6. 中斷與遠端桌面工作階段的連線，並返回 Amazon EC2 主控台。
7. 在 Instances (執行個體) 清單中，停止執行個體。
8. 執行個體停止後，在 Amazon EC2 主控台中，在執行個體清單中的執行個體上按一下滑鼠右鍵，選擇映像，然後選擇建立映像。
9. 在 Create Image (建立映像) 頁面上，提供映像的名稱和描述，並指定磁碟區組態。完成之後，請選擇 Create Image (建立映像)。

10. 開啟 Images (映像) 頁面，並等待映像從 pending (待定) 階段變更為 available (可用)。新的 AMI 現在已可使用。

使用自訂 Windows AMI 新增執行個體

在您的映像變更為 available (可用) 狀態之後，即可建立根據自訂 Windows AMI 的新執行個體。當您從 Operating system (作業系統) 清單中選擇 Use custom Windows AMI (使用自訂 Windows AMI) 時，AWS OpsWorks Stacks 會顯示自訂 AMI 清單。

新增根據自訂 Windows AMI 的執行個體

1. 當您的新 AMI 可用時，請前往 AWS OpsWorks Stacks 主控台，開啟 Windows 堆疊的執行個體頁面，然後選擇頁面底部附近的 + 執行個體來新增執行個體。
2. 在 New (新增) 標籤上，選擇 Advanced (進階)。
3. 在 Operating system (作業系統) 下拉式清單上，選擇 Use custom Windows AMI (使用自訂 Windows AMI)。
4. 在 Custom AMI (自訂 AMI) 下拉式清單上，選擇您所建立的 AMI，然後選擇 Add Instance (新增執行個體)。

您現在可以啟動並執行執行個體。

手動啟動、停止和重新開機全年無休的執行個體

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 您可以 Linux 和 Windows 堆疊使用全年無休的執行個體。

將 24 小時全年無休執行個體新增至 layer 之後，您必須手動啟動執行個體以啟動對應的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體，並手動停止執行個體以終止 Amazon EC2 執行個

體。您也可以手動重新啟動無法正常運作的執行個體。 AWS OpsWorks Stacks 會自動啟動和停止以時間為基礎和以負載為基礎的執行個體。如需詳細資訊，請參閱使用時間型和負載型執行個體管理負載。

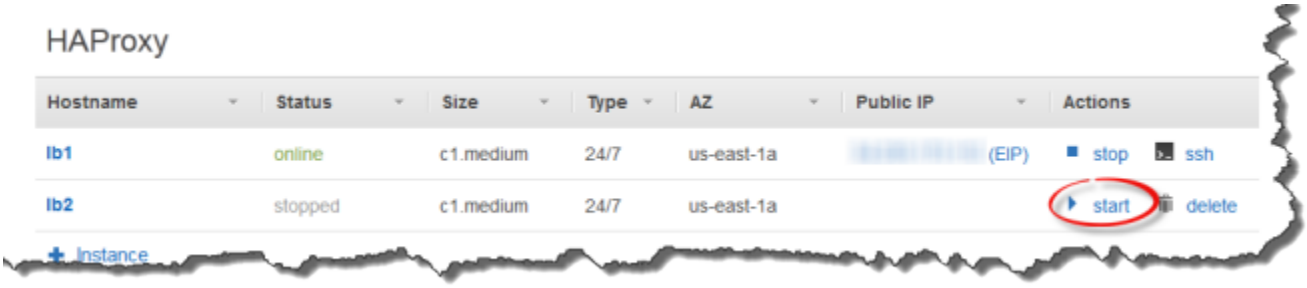
> **Important**
>
> AWS OpsWorks Stacks 執行個體必須僅在主控台中 AWS OpsWorks 啟動、停止和重新啟動。 AWS OpsWorks 無法辨識在 Amazon EC2 主控台中執行的啟動、停止或重新啟動操作。

主題

- 啟動或重新啟動執行個體
- 停止執行個體
- 重新啟動執行個體

啟動或重新啟動執行個體

若要啟動新的執行個體，請在 Instances (執行個體) 頁面上，按一下執行個體 Actions (動作) 欄中的 start (啟動)。



您也可以建立多個執行個體，然後按一下 Start all Instances (啟動所有執行個體) 同時將它們啟動。

啟動執行個體之後， AWS OpsWorks Stacks 會啟動 Amazon EC2 執行個體並啟動作業系統。啟動程序通常需要幾分鐘，而且 Windows 執行個體一般會比 Linux 執行個體慢。隨著啟動的進行，執行個體的 Status (狀態) 欄位會顯示下列一系列的值：

1. requested - AWS OpsWorks Stacks 已呼叫 Amazon EC2 服務來建立 Amazon EC2 執行個體。
2. 待定 - AWS OpsWorks Stacks 正在等待 Amazon EC2 執行個體啟動。
3. 開機 - Amazon EC2 執行個體正在開機。

4. running_setup - AWS OpsWorks Stacks 已觸發設定事件，並正在執行 layer 的Setup配方，後面接著其Deploy配方。如需詳細資訊，請參閱執行配方。如果您已將自訂技術指南新增至堆疊， AWS OpsWorks Stacks 會在執行 Setup和 Deploy配方之前，從您的儲存庫安裝最新版本。
5. online - 執行個體已準備就緒可供使用。

當 Status (狀態) 變更為 online 時，執行個體為完全運作狀態。

- 如果 layer 有連接的負載平衡器， AWS OpsWorks Stacks 會將執行個體新增至該 layer。
- AWS OpsWorks Stacks 會觸發事件，該Configure事件會執行每個執行個體的Configure配方。

  這些配方會視需要更新執行個體，以容納新的執行個體。
- AWS OpsWorks Stacks 會將執行個體的啟動動作取代為停止，您可以使用此動作來停止執行個體。

如果執行個體未能成功啟動或設定配方失敗，則狀態會分別設為 start_failed 或 setup_failed。您可以檢查日誌以確定原因。如需詳細資訊，請參閱偵錯和故障診斷指南。

已停止的執行個體仍留在堆疊中，並保留所有資源。例如，Amazon EBS 磁碟區和彈性 IP 地址仍與已停止的執行個體相關聯。您可以在執行個體的動作欄中選擇啟動，以重新啟動已停止的執行個體。重新啟動已停止的執行個體會執行下列動作：

- 執行個體存放區後端執行個體 – AWS OpsWorks Stacks 會啟動具有相同組態的新 Amazon EC2 執行個體。
- Amazon EBS 後端執行個體 – AWS OpsWorks Stacks 會重新啟動 Amazon EC2 執行個體，重新連接根磁碟區。

執行個體完成開機後， AWS OpsWorks Stacks 會安裝作業系統更新並執行 Setup和 Deploy配方，就像初始啟動一樣。 AWS OpsWorks Stacks 也會視情況對重新啟動的執行個體執行下列動作。

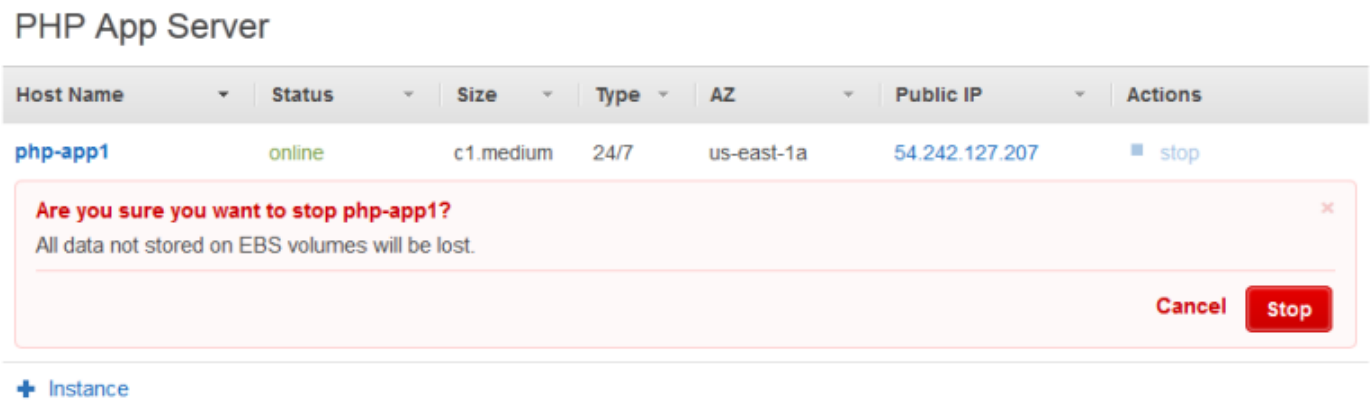
- 重新建立彈性 IP 地址的關聯。
- 重新連接 Amazon Elastic Block Store (Amazon EBS) 磁碟區。
- 針對執行個體後端執行個體，安裝最新的技術指南版本。

  Amazon EBS 後端執行個體會繼續使用存放在根磁碟區上的自訂技術指南。如果您的自訂技術指南在您停止執行個體後已變更，則您必須在這些執行個體上線後，手動更新它們。如需詳細資訊，請參閱更新自訂技術指南。

> **Note**
>
> 彈性 IP 地址與重新啟動的執行個體重新建立關聯可能需要幾分鐘的時間。請注意，執行個體的 Elastic IP (彈性 IP) 設定代表中繼資料，僅指出地址應與執行個體相關聯。Public IP (公有 IP) 設定反映執行個體的狀態，一開始可能是空的。當彈性 IP 地址與執行個體相關聯時，地址會指派給 Public IP (公有 IP) 設定，後面接著 (EIP)。

停止執行個體

在執行個體頁面上，按一下執行個體動作欄中的停止，通知 Stacks AWS OpsWorks 執行關機配方並終止 EC2 執行個體。



您也可以按一下 Stop All Instances (停止所有執行個體) 關機堆疊中的每個執行個體。

在您停止執行個體之後， AWS OpsWorks Stacks 會執行數個任務：

1. 如果執行個體的 layer 具有連接的 Elastic Load Balancing 負載平衡器， AWS OpsWorks Stacks 會取消註冊執行個體。

   若 layer 支援負載平衡器的連接耗盡功能， AWS OpsWorks Stacks 會延遲觸發 Shutdown 事件，直到連接耗盡完成。如需詳細資訊，請參閱Elastic Load Balancing Layer。
2. AWS OpsWorks Stacks 會觸發Shutdown事件，執行執行個體的Shutdown配方。
3. 觸發Shutdown事件後， AWS OpsWorks Stacks 會等待指定的時間，讓Shutdown配方有時間完成，然後執行下列動作：
   - 終止執行個體後端執行個體，這會刪除所有資料。
   - 停止 Amazon EBS 後端執行個體，保留根磁碟區上的資料。

   如需執行個體儲存體的詳細資訊，請參閱儲存體。

> **Note**
>
> 預設關機逾時設定為 120 秒。若您的 Shutdown 配方需要更多時間，您可以編輯 layer 組態來變更設定。

您可以查看執行個體的 Status (狀態) 欄來監控關機程序。隨著關機進度，其會顯示以下一系列的值：

1. 終止 - AWS OpsWorks Stacks 正在終止 Amazon EC2 執行個體。
2. shutting_down - AWS OpsWorks Stacks 正在執行 layer 的Shutdown配方。
3. 已終止 - Amazon EC2 執行個體已終止。
4. stopped - 執行個體已停止。

重新啟動執行個體

在 Instances (執行個體) 頁面上，按一下未作用的執行個體名稱，開啟 details (詳細資訊) 頁面，然後按一下 Reboot (重新開機)。

此命令會執行相關聯 Amazon EC2 執行個體的軟重新啟動。它不會刪除執行個體的資料，即使是執行個體後端執行個體的資料，也不會觸發任何生命週期事件。

> **Note**
>
> 若要讓 AWS OpsWorks Stacks 自動取代失敗的執行個體，請啟用自動修復。如需詳細資訊，請參閱使用自動修復。

使用時間型和負載型執行個體管理負載

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

隨著您的傳入流量變化，您的堆疊擁有的執行個體數可能會太少而無法正常處理負載，或是超過所需數太多。您可以透過使用時間式或負載式的執行個體，自動增加或減少 layer 的執行個體，來節省時間和金錢，以讓您總有足夠的執行個體來適當處理傳入流量，而無須為不需要的容量支付費用。您無須監控伺服器負載，也無須手動啟動或停止執行個體。此外，時間式和負載式的執行個體會自動分散、擴展和平衡區域內多個可用區域的應用程式，給予您備援及延展性。

自動擴展是以兩個執行個體類型為基礎，會根據不同的條件調整 layer 的線上執行個體。

- Time-based (時間式) 執行個體

  他們允許堆疊透過包含只在特定時間或特定幾天執行的執行個體，來處理遵循可預測模式的負載。例如，您可以在下午 6 點之後啟動一些執行個體，來執行每天晚上的備份任務，或是在週末流量較低時停止某些執行個體。

- Load-based (負載式) 執行個體

  他們允許堆疊透過在流量較高時啟動額外執行個體，並在流量較低時停止執行個體，來根據幾項負載指標中的任何指標來處理變動的負載。例如，當平均 CPU 使用率超過 80% 時，您可以讓 AWS OpsWorks Stacks 啟動執行個體，並在平均 CPU 負載低於 60% 時停止執行個體。

時間式和負載式執行個體都支援 Linux 堆疊，但 Windows 堆疊只能使用時間式執行個體。

與您必須手動啟動和停止的全年無休執行個體不同，您不會自行啟動或停止時間式或負載式執行個體。相反地，您可以設定執行個體，而 AWS OpsWorks Stacks 會根據其組態啟動或停止執行個體。例如，您可以設定以時間為基礎的執行個體，在指定的排程上啟動和停止。然後， AWS OpsWorks Stacks 會根據該組態啟動和停止執行個體。

常見的實務便是同時使用三種執行個體類型，如下所示。

- 一組全年無休的執行個體，處理基本負載。您通常只需啟動這些執行個體，並讓他們持續執行即可。

- 一組以時間為基礎的執行個體，Stacks AWS OpsWorks 會開始和停止處理可預測的流量變化。例如，或您的流量在工作時間內最高，您會將時間式執行個體設為在上午啟動，並在傍晚關機。
- 一組負載型執行個體，Stacks AWS OpsWorks 會開始和停止處理無法預測的流量變化。當負載接近堆疊的 24 小時全年無休執行個體和時間型執行個體的容量時， AWS OpsWorks Stacks 會啟動它們，並在流量恢復正常時停止它們。

如需如何使用這些擴展時間的詳細資訊，請參閱最佳化應用程式伺服器的數目。

> **Note**
>
> 如果您已為執行個體的 layer 建立應用程式或建立自訂技術指南，則當執行個體初次啟動時，AWS OpsWorks Stacks 會自動將最新版本部署到時間型執行個體和負載型執行個體。不過，AWS OpsWorks Stacks 不一定會部署最新的技術指南以重新啟動離線執行個體。如需詳細資訊，請參閱 編輯應用程式 和 更新自訂技術指南。

主題

- 使用自動以時間為基礎的擴展
- 使用自動負載型擴展
- 負載型擴展與自動修復有何不同

使用自動以時間為基礎的擴展

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

以時間為基礎的擴展可讓您控制 layer 在一週中的特定時間或某一天有多少執行個體應該上線，方法是依指定的排程啟動或停止執行個體。 AWS OpsWorks Stacks 會每幾分鐘檢查一次，並視需要啟動或停止執行個體。您會為每個執行個體分別指定排程，如下所示：

- 一天當中的時間。例如，您可以在白天執行比夜間更多的執行個體。
- 一週當中的天。例如，您可以在工作日執行比週末更多的執行個體。

> **Note**
>
> 您無法指定特定日期。

主題

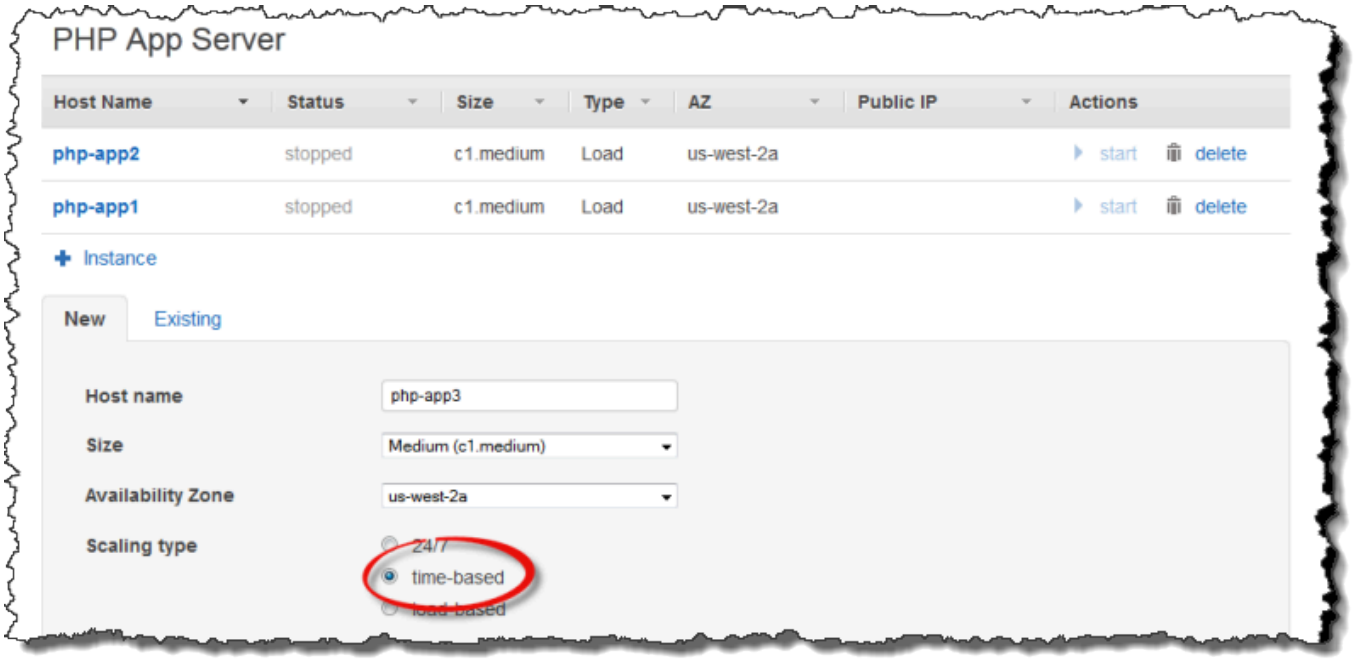
- 將以時間為基礎的執行個體新增至 layer
- 設定以時間為基礎的執行個體

將以時間為基礎的執行個體新增至 layer

您可以將新的時間式執行個體新增至 layer，或是使用現有的執行個體。

新增新的時間式執行個體

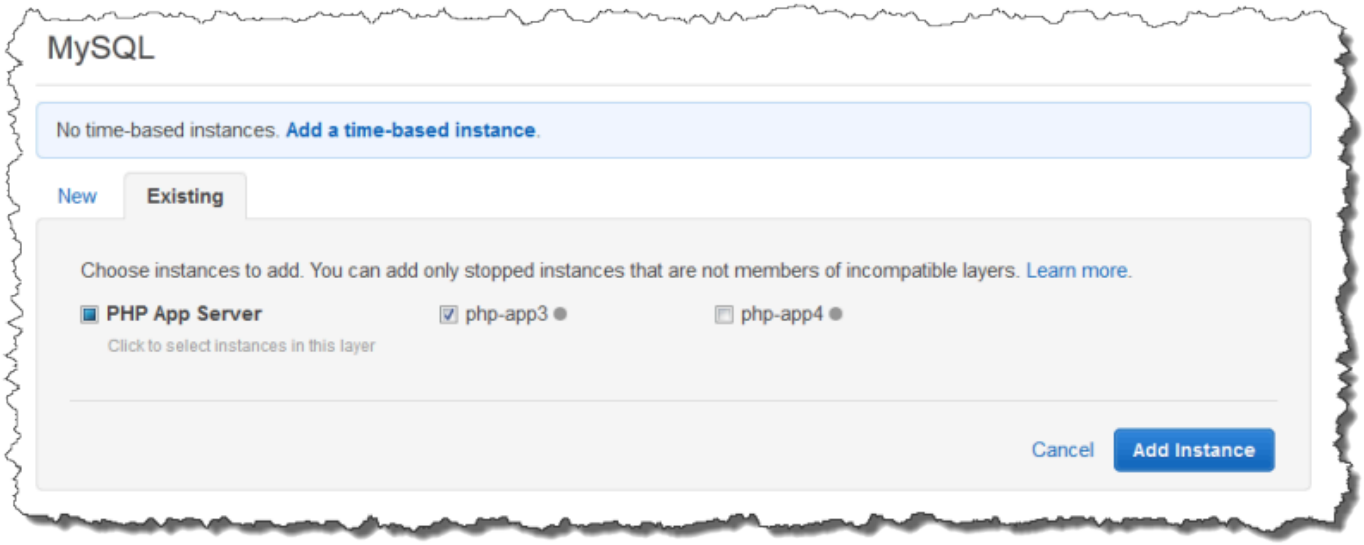
1. 在執行個體頁面上，選擇 + 執行個體以新增執行個體。在新增索引標籤上，選擇進階，然後選擇以時間為基礎的選項。



2. 設定執行個體。然後選擇新增執行個體，將執行個體新增至 layer。

將現有時間式執行個體新增至 layer

1. 在以時間為基礎的執行個體頁面上，如果 layer 已有以時間為基礎的執行個體，請選擇 + 執行個體。否則，請選擇新增以時間為基礎的執行個體。然後選擇現有索引標籤。



2. 在現有索引標籤上，從清單中選擇執行個體。清單只會顯示時間式執行個體。

> **Note**
>
> 如果您改變使用現有執行個體的想法，請在新增索引標籤上建立新的執行個體，如上述程序所述。

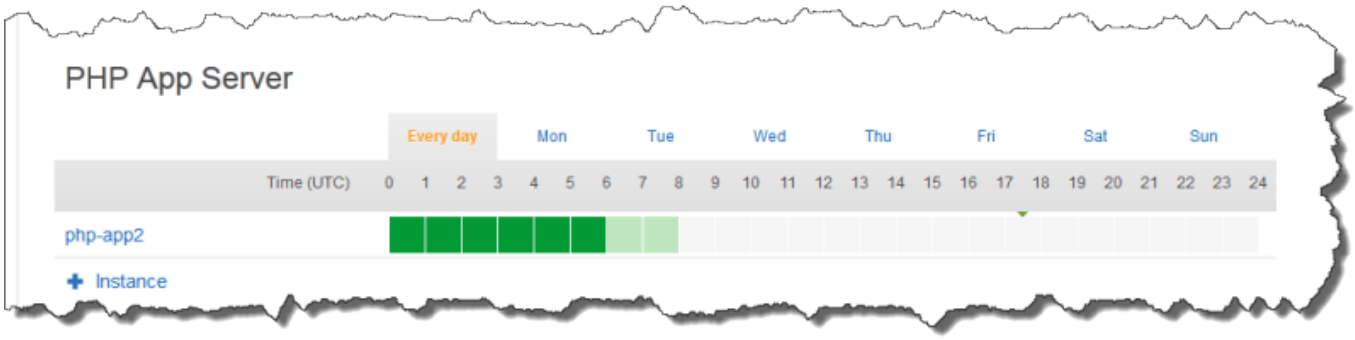
3. 選擇新增執行個體，將執行個體新增至 layer。

設定以時間為基礎的執行個體

在您將時間式執行個體新增至 layer 後，您會根據以下程序設定其排程。

設定時間式執行個體

1. 在導覽窗格中的執行個體下，選擇以時間為基礎的 。
2. 填寫所需小時數下方的適當方塊，以指定每個以時間為基礎的執行個體的線上期間。
   - 若要每天使用相同的排程，請選擇每天索引標籤，然後指定線上時段。
   - 若要在不同日期使用不同的排程，請選擇每天，然後選擇適當的時段。



> **Note**
>
> 請務必預留啟動執行個體所需的時間，且 Stacks AWS OpsWorks 只會每隔幾分鐘檢查一次，以查看執行個體是否應該啟動或停止。例如，若執行個體在 UTC 時間 1:00 時應處於執行狀態，請在 UTC 時間 0:00 啟動它。否則， AWS OpsWorks Stacks 可能無法啟動執行個體，直到超過 1：00 UTC 的幾分鐘，而且執行個體還需要幾分鐘才能上線。

您可以執行上述步驟，隨時變更執行個體的線上時段。下次 AWS OpsWorks Stacks 檢查時，它會使用新的排程來判斷要啟動還是停止執行個體。

> **Note**
>
> 您可以透過開啟時間型頁面，然後選擇新增時間型執行個體 （ 如果您尚未將時間型執行個體新增至 layer) 或 + 執行個體 （ 如果 layer 已有一或多個時間型執行個體 ），將新的時間型執行個體新增至 layer。然後，如上述程序所述設定執行個體。

使用自動負載型擴展

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

負載型執行個體可讓您快速啟動或停止執行個體，以回應傳入流量的變更。 AWS OpsWorks Stacks 使用 Amazon CloudWatch 資料來計算每個 layer 的下列指標，其代表所有 layer 執行個體的平均值：

- CPU：平均 CPU 消耗，例如 80%
- 記憶體：平均記憶體消耗，例如 60%
- 負載：系統在一分鐘內執行的平均運算工作。

您可以為任何或全部的指標定義「擴展」和「縮小」閾值。您也可以使用自訂 CloudWatch 警示做為閾值。

跨越閾值即會觸發「擴展事件」。您可以透過指定下列項目，來判斷 AWS OpsWorks Stacks 回應擴展事件的方式：

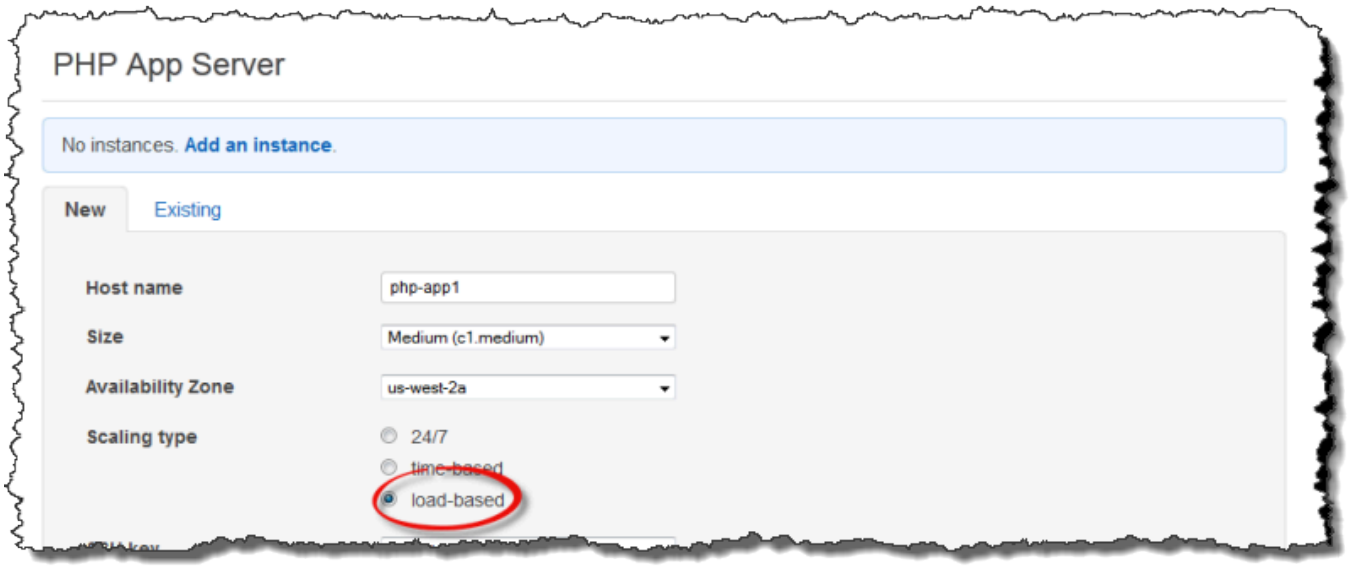
- 要啟動或停止的執行個體數。
- AWS OpsWorks Stacks 在超過閾值之後，應該等待多久，才能開始或刪除執行個體。例如，CPU 使用率必須超過閾值至少 15 分鐘。此值可讓您忽略短暫的流量波動。
- Stacks AWS OpsWorks 在啟動或停止執行個體後應等待多久，才能再次監控指標。您通常會希望允許足夠的時間，讓啟動的執行個體上線或停止的執行個體關機之後，再評定 layer 是否仍然超過閾值。

發生擴展事件時， AWS OpsWorks Stacks 只會啟動或停止負載型執行個體。它不會啟動或停止全年無休執行個體或時間式執行個體。

> **Note**
>
> 自動負載式擴展不會建立新的執行個體。它只會啟動和停止您已建立的執行個體。因此您必須事先佈建足夠的負載式執行個體，才能處理預期的最大負載。

建立負載式執行個體

1. 在執行個體頁面上，選擇 +Instance 以新增執行個體。選擇進階，然後選擇負載型。



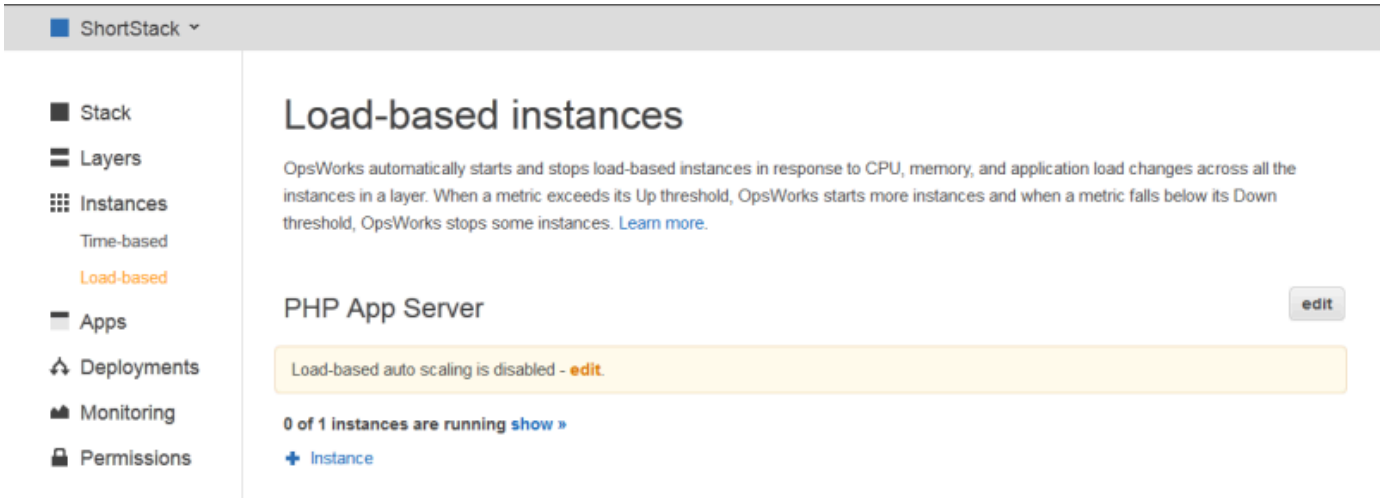
2. 設定執行個體，然後選擇新增執行個體，將執行個體新增至 layer。

重複這個程序，直到您已建立足夠數目的執行個體。您可以視需要於稍後新增或移除執行個體。

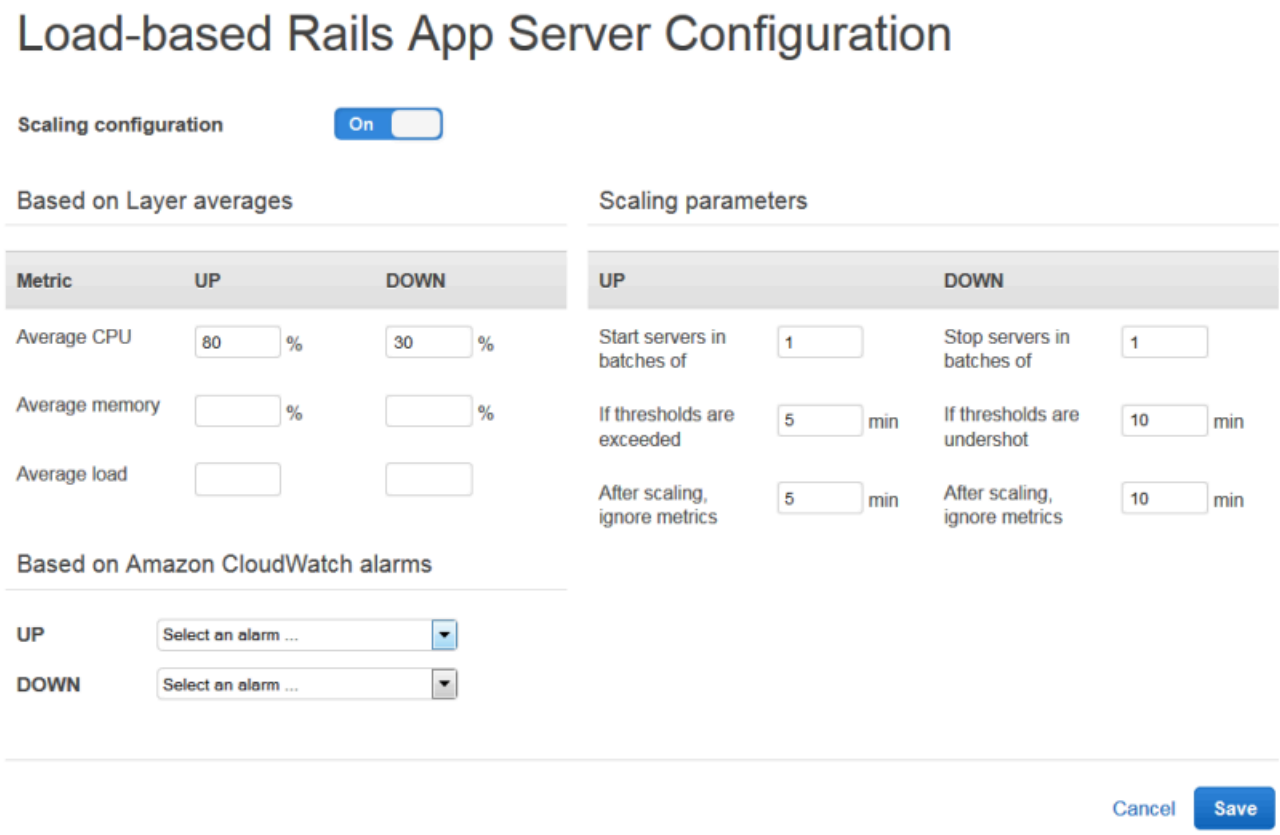
在您將負載式執行個體新增到 layer 後，您必須啟用負載式擴展並指定組態。負載式擴展組態為一種 layer 屬性，而非執行個體屬性，可指定 layer 應啟動或停止其負載式執行個體的時機。您必須為每個使用負載式執行個體的 layer 分別指定它。

啟用和設定自訂負載式擴展

1. 在導覽窗格的執行個體下，選擇以負載為基礎的，然後選擇編輯適當的 layer。



2. 將啟用負載型自動擴展設為開啟。然後設定閾值和擴展參數，定義新增或刪除執行個體的方式和時機。



Layer 平均閾值

您可以根據下列值 (所有 layer 執行個體的平均) 設定擴展的閾值。

- 平均 CPU – 層的平均 CPU 使用率，以總數的百分比表示。
- 平均記憶體 – 層的平均記憶體使用率，以總數的百分比表示。
- 平均負載 – layer 的平均負載。

  如需如何計算負載的詳細資訊，請參閱 Wikipedia 上的載入（運算）。

越過閾值會導致擴展事件，如果需要更多執行個體，則向上擴展，如果需要較少執行個體，則向下擴展。 AWS OpsWorks Stacks 接著會根據擴展參數新增或刪除執行個體。

自訂 CloudWatch 警示

您最多可以使用五個自訂 CloudWatch 警示作為上擴展或下擴展閾值。他們必須位於和堆疊相同的區域中。如需如何建立自訂警示的詳細資訊，請參閱建立 Amazon CloudWatch 警示。

> **Note**
>
> 若要使用自訂警示，您必須更新您的服務角色，允許 `cloudwatch:DescribeAlarms`。您可以在第一次使用此功能時讓 AWS OpsWorks Stacks 為您更新角色，也可以手動編輯角色。如需詳細資訊，請參閱允許 AWS OpsWorks Stacks 代表您。
>
> 當有多個警示設定為負載型組態時，如果警示處於INSUFFICIENT_DATA指標型警示狀態，即使另一個警示處於 ALARM 狀態，負載型執行個體也無法擴展。只有在所有警示都處於 OK或 ALARM 狀態時，才能繼續自動擴展。如需使用 Amazon CloudWatch 警示的詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的使用 Amazon CloudWatch 警示。 Amazon CloudWatch

擴展參數

下列參數控制 AWS OpsWorks Stacks 如何管理擴展事件。

- 批次啟動伺服器 – 發生擴展事件時要新增或移除的執行個體數量。
- 如果超過閾值 – 負載必須維持在向上擴展閾值或低於向下擴展閾值的時間量 （以分鐘為單位），Stacks AWS OpsWorks 才會觸發擴展事件。
- 擴展後，忽略指標 – 發生擴展事件後 Stacks AWS OpsWorks 應忽略指標並隱藏其他擴展事件的時間量 （以分鐘為單位）。

  例如， AWS OpsWorks Stacks 在升級事件之後新增執行個體，但執行個體在開機和設定之前不會開始減少負載。在新的執行個體上線並開始處理請求之前 (通常需要數分鐘)，引發其他擴展事件沒有意義。此設定可讓您命令 AWS OpsWorks Stacks 隱藏擴展事件足夠的時間，以讓新的執行個體上線。

  您可以增加此設定，以防止當層平均值，例如平均 CPU、平均記憶體或平均負載處於暫時分歧時，突然擺動擴展。

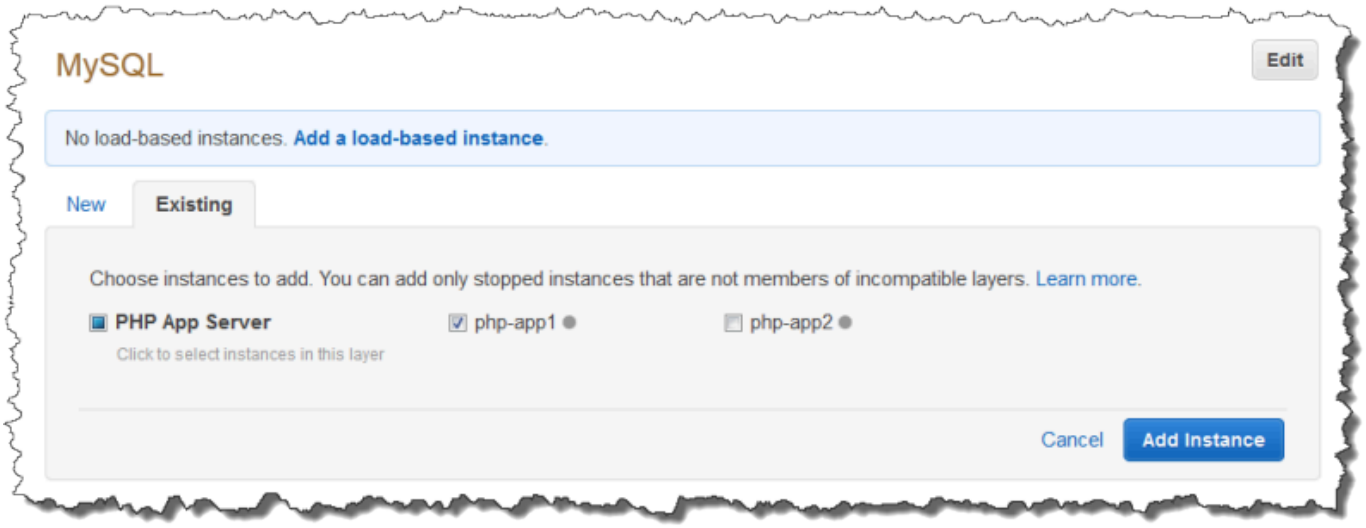
  例如，如果 CPU 用量超過限制，且記憶體用量接近縮減規模，執行個體向上擴展事件可能會立即接著記憶體縮減事件。若要避免這種情況，您可以增加擴展後忽略指標設定的分鐘數。在此範例中，會發生 CPU 擴展，但記憶體縮減事件不會發生。

3. 若要新增其他負載型執行個體，請選擇 + 執行個體、設定設定，然後選擇新增執行個體。重複執行此作業，直到您擁有足夠的負載式執行個體處理您的預期最大負載。然後選擇 Save (儲存)。

> **Note**
>
> 您也可以開啟負載型頁面，然後選擇新增負載型執行個體 （如果您尚未將負載型執行個體新增至 layer) 或 + 執行個體 （如果層已有一或多個負載型執行個體），將新的負載型執行個體新增至 layer。接著設定執行個體，如本節中先前部分所述。

將現有負載式執行個體新增至 layer

1. 在導覽窗格的執行個體下，選擇以負載為基礎的 。
2. 如果您已啟用層的負載型自動擴展，請選擇 + 執行個體。否則，請選擇新增負載型執行個體。選擇現有索引標籤。



3. 在現有索引標籤上，選擇執行個體。清單只會顯示負載式執行個體。

> **Note**
>
> 如果您改變使用現有執行個體的心意，請在新增索引標籤上，如上述程序所述建立新的執行個體。

4. 選擇新增執行個體，將執行個體新增至 layer。

您可以隨時修改自動負載式擴展的組態或停用它。

停用自動負載式擴展

1. 在導覽窗格的執行個體下，選擇以負載為基礎的，然後選擇編輯適當的 layer。

2. 啟用以負載為基礎的 Switch Load 自動擴展為 No。

負載型擴展與自動修復有何不同

自動負載式擴展使用所有執行中執行個體的平均負載指標。如果指標保持在指定的閾值之間， AWS OpsWorks Stacks 不會啟動或停止任何執行個體。另一方面，透過自動修復，當執行個體停止回應時， AWS OpsWorks Stacks 會自動啟動具有相同組態的新執行個體。執行個體可能會因網路問題或執行個體的一些問題而無法回應。

例如，假設您的 CPU 升級閾值為 80%，且一個執行個體停止回應。

- 如果停用自動修復，且剩餘的執行中執行個體可以將平均 CPU 使用率保持在 80% 以下， AWS OpsWorks 則Stacks 不會啟動新的執行個體。它只會在剩餘執行個體的平均 CPU 使用率超過 80% 時啟動取代用執行個體。
- 如果啟用自動修復，無論負載閾值為何， AWS OpsWorks Stacks 都會啟動替換執行個體。

使用在 AWS OpsWorks Stacks 之外建立的運算資源

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 只有 Linux 堆疊支援此功能。

執行個體 說明如何使用 AWS OpsWorks Stacks 來建立和管理 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體的群組。您也可以將 Linux 運算資源整合到在 Stacks AWS OpsWorks 外部建立的堆疊中：

- 您使用 Amazon EC2 主控台、CLI 或 API 直接建立的 Amazon EC2 執行個體。
- 在您自己的硬體上執行的「現場部署」執行個體，包括在虛擬機器中執行的執行個體。

這些運算資源會成為 AWS OpsWorks Stacks 受管執行個體，而且您可以像一般 Stacks AWS OpsWorks 執行個體一樣進行管理：

- 管理使用者許可 – 您可以使用 AWS OpsWorks Stacks 使用者管理來指定哪些使用者可存取您的堆疊、哪些使用者可在堆疊的執行個體上執行的動作，以及他們是否具有 SSH 存取和 sudo 權限。
- 自動化任務 – 您可以讓 AWS OpsWorks Stacks 執行自訂 Chef 配方來執行任務，例如使用單一命令在任何或所有堆疊執行個體上執行指令碼。

  如果您將執行個體指派給 layer， AWS OpsWorks Stacks 會在執行個體生命週期的重點自動執行一組指定的 Chef 配方，包括您的自訂配方。請注意，您只能將註冊的 Amazon EC2 執行個體指派給自訂層。
- 管理資源 – 堆疊可讓您在 中分組和管理資源 AWS 區域，而 OpsWorks 儀表板會顯示所有區域的堆疊狀態。
- 安裝套件 – 您可以使用 Chef 配方在堆疊中的任何執行個體上安裝套件。
- 更新作業系統 – AWS OpsWorks Stacks 提供在堆疊執行個體上安裝作業系統安全修補程式和更新的簡單方法。
- 部署應用程式 – AWS OpsWorks Stacks 會一致地將應用程式部署到堆疊的所有應用程式伺服器執行個體。
- 監控 – AWS OpsWorks Stacks 會建立自訂 CloudWatch 指標來監控堆疊的所有執行個體。

如需定價資訊，請參閱 AWS OpsWorks 定價。

以下是使用已註冊執行個體的基本程序。

1. 向堆疊註冊執行個體。

   執行個體現在是堆疊的一部分，並由 Stacks AWS OpsWorks 管理。
2. (選擇性) 將執行個體指派給某 layer。

   此步驟可讓您充分利用 AWS OpsWorks Stacks 管理功能。您可以將已註冊的內部部署執行個體指派給任何 layer；已註冊的 Amazon EC2 執行個體只能指派給自訂 layer。
3. 使用 AWS OpsWorks Stacks 管理執行個體。
4. 當您不再需要堆疊中的執行個體時，請將其取消註冊，這會從 Stacks AWS OpsWorks 中移除執行個體。

下列各節將詳細說明此程序。

主題

- 向 AWS OpsWorks Stacks Stack 註冊執行個體
- 管理已註冊的執行個體
- 將已註冊的執行個體指派給某 Layer
- 取消指派已註冊的執行個體
- 取消註冊已註冊的執行個體
- 已註冊執行個體的生命週期

向 AWS OpsWorks Stacks Stack 註冊執行個體

Important

AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Note

只有 Linux 堆疊支援此功能。

若要註冊 Stacks AWS OpsWorks 外部的執行個體，請執行 AWS CLI aws opsworks register命令。您可以從要註冊的執行個體或從另一部電腦執行此命令 您可以將 `AWSOpsWorksRegisterCLI_EC2`或 `AWSOpsWorksRegisterCLI_OnPremises`政策套用至使用者或群組，以授予 分別 AWS CLI 註冊 EC2 或內部部署執行個體所需的許可。這些政策需要 1.16.180 版 AWS CLI 或更新版本。

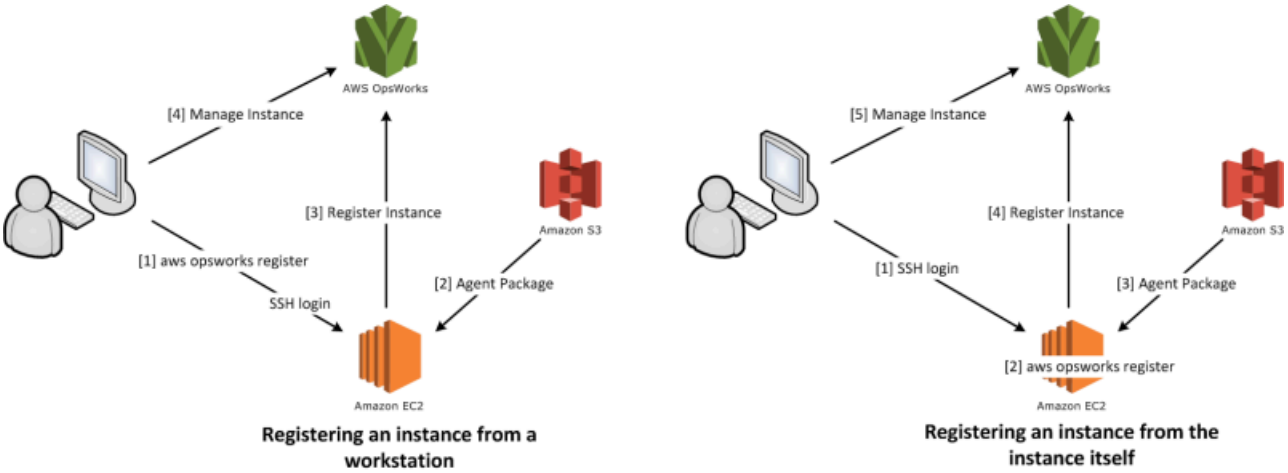
Note

若要防止使用者或角色註冊執行個體，請更新執行個體設定檔以拒絕存取 register命令。

註冊程序會使用 AWS OpsWorks Stacks 在您想要管理的執行個體上安裝代理程式，並使用您指定的 AWS OpsWorks 堆疊註冊執行個體。在註冊執行個體之後，執行個體會成為堆疊的一部分，並由 AWS OpsWorks Stacks 管理。如需詳細資訊，請參閱管理已註冊的執行個體。

> **Note**
>
> 雖然 AWS Tools for PowerShell 包含呼叫 `register` API 動作的 `Register-OpsInstance` cmdlet，但我們建議您改用 AWS CLI 執行`register`命令。

下圖顯示註冊 Amazon EC2 執行個體的兩種方法。您可以使用相同的方法來註冊現場部署執行個體。



> **Note**
>
> 您可以使用 AWS OpsWorks Stacks 主控台來管理已註冊的執行個體，但必須執行 AWS CLI `register` 命令來註冊執行個體。這樣做的原因是因為註冊程序必須從執行個體執行，無法透過主控台完成。

下列各節將詳細說明此程序。

主題

- 演練：從您的工作站註冊執行個體
- 註冊 Amazon EC2 和現場部署執行個體

演練：從您的工作站註冊執行個體

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 只有 Linux 堆疊支援此功能。

註冊程序支援數種案例。本節會逐步解說一個案例的end-to-end範例：如何使用工作站註冊 Amazon EC2 執行個體。其他註冊案例使用類似的程序。如需詳細資訊，請參閱註冊 Amazon EC2 和現場部署執行個體。

> **Note**
>
> 您通常想要註冊現有的 Amazon EC2 執行個體。不過，在本演練中，您可以直接建立新的執行個體和新的堆疊，並於完成時予以刪除。

主題

- 步驟 1：建立堆疊和執行個體
- 步驟 2：安裝並設定 AWS CLI
- 步驟 3：向 EC2Register 堆疊註冊執行個體

步驟 1：建立堆疊和執行個體

若要開始使用，您需要一個堆疊和一個 Amazon EC2 執行個體才能註冊該堆疊。

建立堆疊和執行個體

1. 使用 AWS OpsWorks Stacks 主控台來建立新的堆疊，並命名為 **EC2Register**。針對其他堆疊設定，您可以接受預設值。

2. 從 Amazon EC2 主控台啟動新的執行個體。請注意下列內容。

   - 執行個體必須與堆疊位於相同的區域和 VPC。

     如果您使用 VPC，請為本演練挑選一個公有子網路。
   - 如果您需要建立 SSH 金鑰，請將私有金鑰檔案儲存至您的工作站，並記錄名稱和檔案位置。

     如果您使用現有的金鑰，請記錄名稱和私有金鑰檔案位置。您稍後需要這些值。
   - 執行個體必須以其中一個支援的 Linux 作業系統為基礎。例如，如果您的堆疊位於美國西部（奧勒岡），您可以使用 `ami-35501205`在該區域中啟動 Ubuntu 14.04 LTS 執行個體。

   否則，請接受預設值。

在執行個體啟動期間，您可以繼續進行下一節。

步驟 2：安裝並設定 AWS CLI

使用 AWS CLI aws opsworks register 命令執行註冊。註冊第一個執行個體之前，您必須執行 1.16.180 版 AWS CLI 或更新版本。安裝詳細資訊取決於您工作站的作業系統。如需安裝 的詳細資訊 AWS CLI，請參閱安裝 AWS 命令列介面。若要檢查您正在執行的 AWS CLI 版本，請在 shell 工作階段中輸入 `aws --version`。

> **Note**
>
> 若要防止使用者或角色註冊執行個體，請更新執行個體設定檔以拒絕存取 register命令。

強烈建議您不要略過此步驟，即使您已在工作站 AWS CLI 上執行 。使用最新版 AWS CLI 是安全最佳實務。

您必須提供 `register` 一組具備適當許可的 AWS 登入資料。建議執行此作業的方式，是為了避免直接在執行個體上安裝登入資料，而是要使用執行個體描述檔註冊啟動的執行個體，然後將`--use-instance-profile`交換器新增至您的`register`命令。如果您透過執行個體描述檔取得登入資料，則請跳到本主題中的步驟 3：向 EC2Register 堆疊註冊執行個體。不過，如果未使用執行個體描述檔啟動您的執行個體，則可以建立 IAM 使用者。下列程序會建立具有適當許可的新使用者，在工作站上安裝使用者的登入資料，然後將這些登入資料傳遞給 `register`。

> **Warning**
>
> IAM 使用者具有長期憑證，這會造成安全風險。為了協助降低此風險，建議您只為這些使用者提供執行任務所需的許可，並在不再需要這些使用者時將其移除。

建立使用者

1. 在 IAM 主控台的導覽窗格中，選擇 Users (使用者)，然後選擇 Add user (新增使用者)。
2. 新增名為 **`EC2Register`** 的使用者。
3. 選擇 Next (下一步)。
4. 在設定許可頁面上，選擇直接連接政策。
5. **`OpsWorks`** 在許可政策篩選條件方塊中輸入 以顯示 AWS OpsWorks 政策，選取下列其中一個政策，然後選擇下一步：檢閱。此政策會授予您的使用者執行 `register` 所需的許可。
   - 選擇 `AWSOpsWorksRegisterCLI_EC2` 以許可使用者註冊使用執行個體設定檔的 EC2 執行個體。
   - 選擇 `AWSOpsWorksRegisterCLI_OnPremises` 以許可使用者註冊現場部署執行個體。
6. 選擇 Next (下一步)。
7. 在 Review (檢閱) 頁面上，選擇 Create user (建立使用者)。
8. 現在為您的使用者建立存取金鑰。從導覽窗格中，選擇使用者，然後選擇您要為其建立存取金鑰的使用者。
9. 選擇安全登入資料索引標籤，然後選擇建立存取金鑰。
10. 選擇最符合您任務的存取金鑰最佳實務和替代方案。
11. 選擇 Next (下一步)。
12. （選用）輸入標籤以識別存取金鑰。
13. 選擇 Next (下一步)。
14. 選擇下載 .csv 檔案，將登入資料檔案儲存至系統上方便的位置，然後選擇完成。

您需要將 IAM 使用者的登入資料提供給 `register`。本演練會透過安裝您工作站之 `credentials` 檔案中的 EC2Register 登入資料，來處理此任務。如需管理 登入資料的其他方法的資訊 AWS CLI，請參閱組態和登入資料檔案。

安裝使用者的登入資料

1. 建立或開啟您工作站的 `credentials` 檔案。該檔案位於 `~/.aws/credentials` (Linux、Unix 和 OS X) 或 `C:\Users\User_Name\.aws\credentials` (Windows 系統) 中。
2. 使用下列格式，將 EC2Register 使用者的描述檔新增至 `credentials` 檔案。

   ```
   [ec2register]
   aws_access_key_id = access_key_id
   aws_secret_access_key = secret_access_key
   ```

   將 *access_key_id* 和 *secret_access_key* 取代為您稍早下載的 EC2Register 金鑰。

步驟 3：向 EC2Register 堆疊註冊執行個體

您現在可以註冊執行個體。

註冊執行個體

1. 在 AWS OpsWorks Stacks 中，返回 EC2Register 堆疊，在導覽窗格中選擇執行個體，然後選擇註冊執行個體。
2. 選取 EC2 Instances (EC2 執行個體)，並選擇 Next: Select Instances (下一步：選取執行個體)，然後從清單中選取您的執行個體。
3. 選擇下一步：安裝 AWS CLI 和下一步：註冊執行個體。 AWS OpsWorks Stacks 會自動使用可用的資訊，例如堆疊 ID 和執行個體 ID 來建立`register`命令範本，該範本會顯示在註冊執行個體頁面上。在此範例中，您使用 `register` 透過 SSH 金鑰登入執行個體，並明確指定金鑰檔案，因此，請將 I use SSH keys to connect to my instances (我使用 SSH 金鑰連線到我的執行個體) 設為 Yes (是)。此命令範本如下所示。

   ```
   aws opsworks register --infrastructure-class ec2 --region region endpoint ID
     --stack-id 247be7ea-3551-4177-9524-1ff804f453e3 --ssh-username [username]
     --ssh-private-key [key-file] i-f1245d10
   ```

> **Note**
>
> 如果堆疊位於與區域端點相關聯的傳統區域中，您必須將區域設定為 AWS OpsWorks Stacks 服務的端點區域，而不是堆疊`us-east-1`的區域。 AWS OpsWorks Stacks 會從堆疊 ID 決定堆疊的區域。

4. 命令範本包含數個使用者特定的引數值，這些值會以括號表示且必須取代為適當的值。將命令範本複製到文字編輯器，並依照下列方式編輯。

> **Important**
>
> 在註冊程序期間建立的 IAM 使用者，在註冊執行個體的整個生命週期中都是必要的。刪除使用者會導致 AWS OpsWorks Stacks 代理程式無法與服務通訊。為了協助防止在使用者意外刪除時管理已註冊執行個體的問題，請將 `--use-instance-profile` 參數新增至您的`register`命令，以改用執行個體的內建執行個體描述檔。新增 `--use-instance-profile` 參數也可防止每 90 天輪換 AWS 帳戶存取金鑰時發生錯誤（建議的最佳實務），因為它可防止代理程式可用的 AWS OpsWorks 存取金鑰與必要的 IAM 使用者之間不相符。

- 將`####`取代為您建立執行個體時所儲存之 Amazon EC2 金鑰對的私有金鑰檔案完整路徑。

  如果您想要，也可以使用相對路徑。

- 將 *`username`* 取代為執行個體的使用者名稱。

  在此範例中，使用者名稱是 ubuntu (若是 Ubuntu 執行個體) 或 `ec2-user` (若是 Red Hat Enterprise Linux (RHEL) 或 Amazon Linux 執行個體)。

- 新增 `--use-instance-profile`，這會`register`使用執行個體設定檔執行，以防止金鑰輪換期間或意外刪除主體 IAM 使用者時發生錯誤。

您的命令應該如下所示。

```
aws opsworks register --use-instance-profile --infrastructure-class ec2 \
  --region us-west-2  --stack-id 247be7ea-3551-4177-9524-1ff804f453e3 --ssh-
username ubuntu \
  --ssh-private-key "./keys/mykeys.pem" i-f1245d10
```

5. 在您的工作站上開啟終端機視窗，從您的編輯器貼上 `register` 命令，然後執行命令。

   註冊通常需要約 5 分鐘。完成後，返回 Stacks AWS OpsWorks 主控台並選擇完成。然後，在導覽窗格中選擇 Instances (執行個體)。您的執行個體應該會列於 Unassigned Instances (未指派的執行個體) 下。然後，您可以將執行個體指派給某 layer 或保持不變，端視您預計管理執行個體的方式而定。

6. 完成後，請停止執行個體，然後使用 Stacks 主控台或命令將其刪除。 AWS OpsWorks 這會終止 Amazon EC2 執行個體，因此您不會產生任何進一步的費用。

註冊 Amazon EC2 和現場部署執行個體

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 只有 Linux 堆疊支援此功能。

本節說明如何向 AWS OpsWorks Stacks 堆疊註冊 Amazon EC2 或內部部署執行個體。

主題

- 準備執行個體
- 安裝並設定 AWS CLI
- 註冊執行個體
- 使用 register 命令列
- register 命令範例
- 執行個體註冊政策

準備執行個體

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 只有 Linux 堆疊支援此功能。

註冊執行個體之前，您必須確定其與 AWS OpsWorks Stacks 相容。詳細資訊取決於您是註冊現場部署還是 Amazon EC2 執行個體。

現場部署執行個體

內部部署執行個體必須符合下列條件：

- 執行個體必須執行其中一個支援的 Linux 作業系統。雖然可以使用從自訂或社群所產生 AMI 建立而成的其他作業系統 (例如 CentOS 6.x) 建立或註冊執行個體，但並未正式受到支援。

  您必須在執行個體上安裝 `libyaml` 套件。若是 Ubuntu 執行個體，套件名稱為 `libyaml-0-2`。若是 CentOS 和 Red Hat Enterprise Linux 執行個體，套件名稱為 `libyaml`。
- 執行個體必須具有支援的執行個體類型 (有時稱為執行個體大小)。支援的執行個體類型可能會因作業系統而異，並取決於您的堆疊是否在 VPC 中。如需支援的執行個體類型清單，請檢視當您嘗試在目標堆疊中建立新的執行個體時，Stacks AWS OpsWorks 主控台中顯示的大小下拉式清單值。如果執行個體類型呈現灰色且無法在您的目標堆疊中建立，則您無法註冊該類型的執行個體。
- 執行個體必須具有網際網路存取，允許其與 AWS OpsWorks Stacks 服務端點 通訊`opsworks.us-east-1.amazonaws.com (HTTPS)`。執行個體也必須支援對 Amazon S3 等 AWS 資源的傳出連線。
- 如果您預計從個別工作站註冊執行個體，已註冊的執行個體必須支援從工作站的 SSH 登入。

  如果您從執行個體執行註冊命令，則不需要 SSH 登入。
- AWS 存取金鑰用於從 AWS OpsWorks 代理程式到 Stacks AWS OpsWorks 服務的身分驗證。如果您依建議每 90 天輪換存取金鑰，請手動更新 AWS OpsWorks 代理程式以使用新的金鑰。在內

部部署電腦或執行個體上，使用新的存取金鑰和私密金鑰編輯/etc/aws/opsworks/instance-agent.yml檔案。下列命令顯示此檔案中的存取金鑰和秘密金鑰。使用舊金鑰的代理程式可能會造成錯誤。

```
cat /etc/aws/opsworks/instance-agent.yml | egrep "access_key|secret_key"
:access_key_id: AKIAIOSFODNN7EXAMPLE
:secret_access_key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

Amazon EC2 執行個體

Amazon EC2 執行個體必須符合下列條件：

- AMI 必須以其中一個支援的 Linux 作業系統為基礎。如需目前的清單，請參閱AWS OpsWorks Stacks 作業系統。

  如需詳細資訊，請參閱使用自訂 AMI。

  如果執行個體是以衍生自標準支援 AMI 的自訂 AMI 為基礎，或包含最少的安裝，您必須在執行個體上安裝 `libyaml` 套件。若是 Ubuntu 執行個體，套件名稱為 `libyaml-0-2`。對於 Amazon Linux 和 Red Hat Enterprise Linux 執行個體，套件名為 `libyaml`。
- 執行個體必須具有支援的執行個體類型 (有時稱為執行個體大小)。支援的執行個體類型可能會因作業系統而異，並取決於您的堆疊是否在 VPC 中。如需支援的執行個體類型清單，請檢視當您嘗試在目標堆疊中建立新的執行個體時，Stacks AWS OpsWorks 主控台中顯示的大小下拉式清單值。如果執行個體類型呈現灰色且無法在您的目標堆疊中建立，則您也無法註冊該類型的執行個體。
- 執行個體必須處於 `running` 狀態。
- 執行個體不應該是 Auto Scaling 群組的一部分。

  如需詳細資訊，請參閱從 Auto Scaling 群組分離 EC2 執行個體。
- 執行個體可以是 VPC 的一部分，但必須與堆疊位於相同的 VPC 中，且 VPC 必須設定為與 Stacks AWS OpsWorks 正常運作。
- 不支援 Spot 執行個體，因為這些執行個體不適用於自動修復。

當您註冊 Amazon EC2 執行個體時， AWS OpsWorks Stacks 不會修改執行個體的安全群組或規則。請確定執行個體的安全群組規則符合下列 Stacks AWS OpsWorks 需求。

Ingress Rules (輸入規則)

輸入規則應該允許下列項目。

- SSH 登入。
- 來自適當 layer 的流量。

  例如，資料庫伺服器通常會允許來自堆疊之應用程式伺服器 layer 的輸入流量。
- 連入適當連接埠的流量。

  例如，應用程式伺服器執行個體通常會允許連入連接埠 80 (HTTP) 和 443 (HTTPS) 的所有輸入流量。

Egress Rules (輸出規則)

輸出規則應該允許下列項目。

- 來自執行個體上執行之應用程式的 AWS OpsWorks Stacks 服務流量。
- 使用 AWS API 從應用程式存取 Amazon S3 等 AWS 資源的流量。

一個常用方法是不指定任何輸出規則，如此就不會對輸出流量設限。

安裝並設定 AWS CLI

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

註冊第一個執行個體之前，您必須在執行 的電腦上執行 1.16.180 版 AWS CLI 或更新版本`register`。安裝詳細資訊取決於您工作站的作業系統。如需安裝 的詳細資訊 AWS CLI，請參閱安裝 AWS Command Line Interface 和設定 AWS Command Line Interface。若要檢查您正在執行的 AWS CLI 版本，請在 shell 工作階段中輸入 `aws --version`。

> **Note**
>
> 雖然 AWS Tools for PowerShell 包含呼叫 `register` API 動作的 `Register-OpsInstance` cmdlet，但我們建議您改用 AWS CLI 執行`register`命令。

您必須以適當許可執行 `register`。您可以在要註冊的工作站或執行個體上安裝具有適當許可的使用者登入資料，藉此使用 IAM 角色取得許可，或取得最佳許可。然後，您可以使用這些登入資料來執行 `register`，如稍後所述。透過將 IAM 政策連接至使用者或角色來指定許可。對於 `register`，您可以使用 `AWSOpsWorksRegisterCLI_EC2`或 `AWSOpsWorksRegisterCLI_OnPremises`政策，分別授予註冊 Amazon EC2 或內部部署執行個體的許可。

> **Note**
>
> 如果您在 Amazon EC2 執行個體`register`上執行 ，最好使用 IAM 角色來提供憑證。如需如何將 IAM 角色連接至現有執行個體的詳細資訊，請參閱《Amazon EC2 使用者指南》中的將 IAM 角色連接至執行個體或取代 IAM 角色。

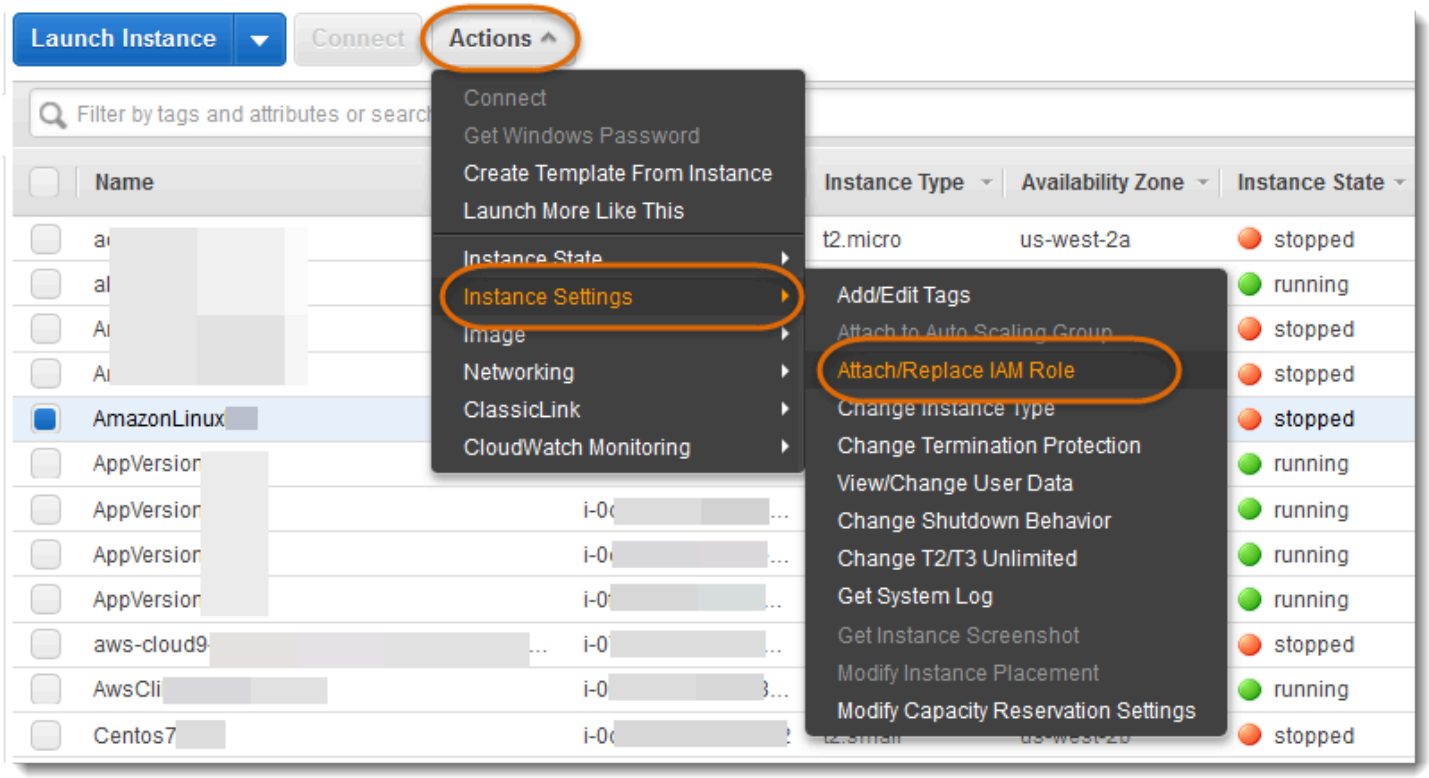
如需 `AWSOpsWorksRegisterCLI_EC2` 和 `AWSOpsWorksRegisterCLI_OnPremises` 政策的範例程式碼片段，請參閱執行個體註冊政策。如需建立及管理 AWS 登入資料的詳細資訊，請參閱 AWS 安全登入資料。

主題

- 使用 IAM 角色
- 使用安裝的登入資料

使用 IAM 角色

如果您從要註冊的 Amazon EC2 執行個體執行 命令，提供登入資料給 的偏好策略`register`是使用已連接`AWSOpsWorksRegisterCLI_EC2`政策或同等許可的 IAM 角色。此方法可讓您避免在執行個體上安裝您的登入資料。其中一種做法是在 EC2 主控台中使用 Attach/Replace IAM Role (連接/取代 IAM 角色) 命令，如下圖所述。



如需如何將 IAM 角色連接至現有執行個體的詳細資訊，請參閱《Amazon EC2 使用者指南》中的將 IAM 角色連接至執行個體或取代 IAM 角色。對於使用執行個體描述檔所啟動的執行個體 (建議)，將 `--use-instance-profile` 參數新增至 `register` 命令，以提供登入資料；請不要使用 `--profile` 參數。

如果執行個體正在執行且具有角色，您可以透過將 `AWSOpsWorksRegisterCLI_EC2` 政策連接至該角色來授予必要許可。該角色會提供一組預設登入資料給執行個體。只要您尚未在執行個體上安裝任何登入資料，`register` 就會自動擔任該角色並以其許可執行。

> **Important**
>
> 我們建議您不要在執行個體上安裝登入資料。除了建立安全風險之外，執行個體的角色也位於預設提供者鏈的結尾， AWS CLI 會用來尋找預設登入資料。安裝的登入資料可能會優先於角色，因此 `register` 可能沒有必要許可。如需詳細資訊，請參閱 AWS CLI入門。

如果執行中的執行個體沒有角色，您必須安裝在執行個體上安裝具備必要許可的登入資料，如使用安裝的登入資料中所述。建議使用透過執行個體描述檔所啟動的執行個體，而這種方式較為簡單，也較不容易出錯。

使用安裝的登入資料

有數種方法可在 系統上安裝使用者登入資料，並將其提供給 AWS CLI 命令。以下說明不再建議使用的方法，但此方法可以用於您註冊未使用執行個體描述檔所啟動的 EC2 執行個體時。只要連接的政策授予必要許可，您也可以使用現有 使用者的登入資料。如需詳細資訊，包括安裝登入資料之其他方式的說明，請參閱組態與登入資料檔案。

使用安裝的登入資料

1. 建立 IAM 使用者，並將存取金鑰 ID 和私密存取金鑰儲存在安全的位置。

   > **Warning**
   >
   > IAM 使用者具有長期憑證，這會造成安全風險。為了協助降低此風險，建議您只為這些使用者提供執行任務所需的許可，並在不再需要這些使用者時將其移除。

2. 連接 AWSOpsWorksRegisterCLI_OnPremises 政策至使用者。如果您想要，也可以連接授予更廣泛許可的政策，只要其中包含 `AWSOpsWorksRegisterCLI_OnPremises` 許可。
3. 在系統的 `credentials` 檔案中建立使用者的描述檔。該檔案位於 `~/.aws/credentials` (Linux、Unix 和 OS X) 或 `C:\Users\`*`User_Name`*`\.aws\credentials` (Windows 系統) 中。檔案包含以下格式的一或多個設定檔，每個設定檔都包含使用者的存取金鑰 ID 和私密存取金鑰。

   ```
   [profile_name]
   aws_access_key_id = access_key_id
   aws_secret_access_key = secret_access_key
   ```

   將先前儲存的 IAM 登入資料替換為 *`access_key_id`* 和 *`secret_access_key`* 值。您可以為描述檔名稱指定任何想要的名稱，但有兩項限制：該名稱必須是唯一的，而且預設描述檔必須命名為 `default`。您也可以使用現有的描述檔，只要該檔案具備必要許可。
4. 使用 `register` 命令的 `--profile` 參數來指定描述檔名稱。`register` 命令會以授予相關聯登入資料的許可執行。

   您也可以省略 `--profile`。在此情況下，`register` 會使用預設登入資料執行。請注意，這些不一定是預設描述檔的登入資料，因此您必須確定預設登入資料具備必要許可。如需 如何 AWS CLI 判斷預設登入資料的詳細資訊，請參閱設定 AWS 命令列介面。

註冊執行個體

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 只有 Linux 堆疊支援此功能。

您可以從工作站或執行個體執行 AWS CLI `register` 命令，來註冊執行個體。處理此操作的最簡單方式就是使用 AWS OpsWorks Stacks 主控台的註冊精靈，這會簡化建構命令字串的程序。熟悉註冊程序之後，如果您想要，也可以略過精靈，並執行 `register` 命令。

以下說明如何使用註冊精靈向現有的堆疊註冊執行個體。

> **Note**
>
> 若要向新堆疊註冊執行個體，您可以在 Stacks AWS OpsWorks 儀表板上選擇註冊執行個體。這會啟動與現有堆疊的精靈相同的精靈，但多了一個用來設定新堆疊的頁面。

使用註冊精靈註冊執行個體

1. 在 AWS OpsWorks Stacks 主控台中，建立堆疊或開啟現有的堆疊。
2. 在導覽窗格中，選擇 Instances (執行個體)，然後選擇 register an instance (註冊執行個體)。
3. 在選擇執行個體類型頁面上，指定您要註冊 Amazon EC2 還是現場部署執行個體：
   - 如果您要註冊 Amazon EC2 執行個體，請選擇下一步：選取執行個體。
   - 如果您要註冊現場部署執行個體，請選擇下一步：安裝 AWS CLI，然後前往步驟 5。
4. 如果您要註冊 Amazon EC2 執行個體，請開啟選取執行個體頁面，選取要註冊的執行個體。 AWS OpsWorks Stacks 會收集建置命令所需的資訊。完成時，選擇 Next:Install AWS CLI (下一步：安裝 AWS CLI)。

5. 您計劃執行的執行個體`register`必須執行 1.16.180 版 AWS CLI 或更新版本。為了安裝或更新 AWS CLI，註冊精靈頁面會提供安裝和設定指示的連結。驗證 AWS CLI 安裝之後，指定您會從要註冊的執行個體還是個別的工作站執行命令，然後選擇 Next: Register Instances (下一步：註冊執行個體)。

6. Register Instances (註冊執行個體) 頁面會顯示 `register` 命令字串的範本，其中包含您選取的選項。例如，如果您要從個別工作站註冊 Amazon EC2 執行個體，預設範本會如下所示。

```
aws opsworks register --infrastructure-class ec2 --region us-west-2
  --stack-id 247be7ea-3551-4177-9524-1ff804f453e3 --ssh-username [username] i-
f1245d10
```

> **Important**
>
> 在註冊程序期間建立的 IAM 使用者，在註冊執行個體的整個生命週期中都是必要的。刪除使用者會導致 AWS OpsWorks Stacks 代理程式無法與服務通訊。為了協助防止在使用者意外刪除時管理已註冊執行個體的問題，請將 `--use-instance-profile` 參數新增至您的`register`命令，以改用執行個體的內建執行個體描述檔。新增 `--use-instance-profile` 參數也可防止每 90 天輪換 AWS 帳戶存取金鑰時發生錯誤（建議的最佳實務），因為它可防止代理程式可用的 AWS OpsWorks 存取金鑰與必要的 IAM 使用者之間不相符。

如果您將我使用 SSH 金鑰設為是， AWS OpsWorks Stacks 會將`--ssh-private-key`引數新增至字串，您可以使用該字串來指定私有 SSH 金鑰檔案。

> **Note**
>
> 如果您想要使用密碼`register`登入，請將我使用 SSH 金鑰設為否。 當您執行時`register`，系統會提示您輸入密碼。

將此字串複製至文字編輯器，並視需要編輯。請注意下列內容。

- 括號文字代表您必須提供的資訊，例如您的 SSH 金鑰檔案位置。
- 此範本假設您使用預設 AWS 登入資料執行 `register`。否則，請將 `--profile` 引數新增至命令字串，並指定您要使用的登入資料描述檔名稱。

對於其他案例，您可能需要進一步變更命令。如需可用 `register` 引數的說明，以及建構命令字串的替代方式，請參閱使用 `register` 命令列。您也可以從命令列執行 `aws opsworks help register` 來顯示命令的文件。如需一些範例命令字串，請參閱register 命令範例。

7. 完成編輯命令字串之後，在您的工作站上開啟終端機視窗或使用 SSH 登入執行個體並執行命令。整個操作通常需要約五分鐘，在這段期間，執行個體會處於 Registering (正在註冊) 狀態。
8. 完成操作時，選擇 Done (完成)。執行個體現在處於 Registered (已註冊) 狀態，並在堆疊的 Instances (執行個體) 頁面上列為未指派的執行個體。

`register` 命令會執行下列動作。

1. 如果 `register` 正在工作站上執行，命令會先使用 SSH 登入要註冊的執行個體。

   其餘處理會在執行個體上進行，不論您在何處執行命令都一樣。
2. 從 AWS OpsWorks Amazon S3 下載 Stacks 代理程式套件。
3. 解壓縮並安裝代理程式及其相依性，例如適用於 Ruby 的 AWS 開發套件。
4. 建立下列項目：
   - 使用 Stacks 服務引導代理程式以提供安全通訊的 IAM AWS OpsWorks 使用者。

     使用者的許可只允許 `opsworks:RegisterInstance` 動作，並會在 15 分鐘後過期。
   - 堆疊的 IAM 群組，其中包含已註冊執行個體的使用者。
5. 建立 RSA 金鑰對，並將公有金鑰傳送至 AWS OpsWorks Stacks。

   此金鑰對可用來加密代理程式與 AWS OpsWorks Stacks 之間的通訊。
6. 向 Stacks AWS OpsWorks 註冊執行個體。堆疊接著會執行一組初始安裝配方來設定執行個體，其中包含下列項目。
   - 覆寫執行個體的主機檔案。

     透過註冊執行個體，您已將使用者管理移交給 AWS OpsWorks Stacks，而 Stacks 必須擁有自己的主機檔案才能控制 SSH 登入許可。
   - 對於 Amazon EC2 執行個體，初始設定也包含向堆疊註冊任何連接的 Amazon EBS 磁碟區或彈性 IP 地址。

     您必須確保 Amazon EBS 磁碟區未掛載至預留掛載點，包括執行個體圖層預留`/var/www`的任何掛載點。如需管理堆疊資源的詳細資訊，請參閱資源管理。如需 layer 掛載點的詳細資訊，請參閱AWS OpsWorks Stacks Layer 參考。

如需初始安裝組態變更的完整說明，請參閱初始安裝組態變更。

> **Note**
>
> 初始安裝不會更新已註冊執行個體的作業系統，您必須自行處理該任務。如需詳細資訊，請參閱管理安全性更新。

使用 **register** 命令列

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 只有 Linux 堆疊支援此功能。

若要註冊執行個體，請確定您正在執行至少 1.16.180 版的 AWS CLI。以下顯示 `register` 命令的一般語法。

```
aws opsworks register \
  [--profile profile_name] \
  [--region region_name] \
  --infrastructure-class instance_type \
  --stack-id stack ID \
  [--local] | [--ssh-private-key key_file --ssh-username username] | [--override-
ssh command_string] \
  [--override-hostname hostname] \
  [--debug] \
  [--override-public-ip public IP] \
  [--override-private-ip private IP] \
..[--use-instance-profile] \
  [ [IP address] | [hostname] | [instance ID]
```

下列引數對於所有 AWS CLI 命令都是常見的。

`--profile`

(選用) 登入資料的描述檔名稱。如果您省略此引數，命令會使用您的預設登入資料執行。如需 如何 AWS CLI 判斷預設登入資料的詳細資訊，請參閱設定 AWS 命令列介面。

`--region`

（選用） Stacks AWS OpsWorks 服務端點的區域。請勿`--region`設定為堆疊的區域。 AWS OpsWorks 堆疊會自動從堆疊 ID 決定堆疊的區域。

> **Note**
>
> 如果您的預設區域已設定，您可以省略此引數。如需如何指定預設區域的詳細資訊，請參閱設定 AWS 命令列界面。

針對 Amazon EC2 和內部部署執行個體使用以下引數。

`--infrastructure-class`

（必要） 此參數必須設定為 `ec2`或 `on-premises`，以指出您是分別註冊 Amazon EC2 還是內部部署執行個體。

`--stack-id`

(必要) 要用來註冊執行個體的堆疊 ID。

> **Note**
>
> 若要尋找堆疊 ID，請在 Stack (堆疊) 頁面上，選擇 Settings (設定)。堆疊 ID 會以 OpsWorks ID 標記，而且是看起來類似 `ad21bce6-7623-47f1-bf9d-af2affad8907` 的 GUID。

SSH 登入引數

使用下列引數指定 `register` 應該如何登入執行個體。

`--local`

(選用) 使用此引數註冊您執行命令的執行個體。

在此情況下，register 不需要登入執行個體。

--ssh-private-key 和 --ssh-username

(選用) 如果您想要從個別工作站註冊執行個體，並想要明確指定使用者名稱或私有金鑰檔案，請使用這些引數。

- --ssh-username – 使用此引數指定 SSH 使用者名稱。

  如果您省略 --ssh-username，ssh 會使用預設使用者名稱。

- --ssh-private-key – 使用此引數明確指定私有金鑰檔案。

  如果您省略 --ssh-private-key，ssh 會嘗試使用不需要密碼的身分驗證技術登入，包含使用預設私有金鑰。如果不支援上述任何技術，ssh 會詢問您的密碼。如需 ssh 如何處理身分驗證的詳細資訊，請參閱 The Secure Shell (SSH) Authentication Protocol。

--override-ssh

(選用) 如果您想要從個別工作站註冊執行個體，並想要指定自訂 ssh 命令字串，請使用此引數。register 命令使用此命令字串登入已註冊的執行個體。

如需 ssh 的詳細資訊，請參閱 SSH。

--override-hostname

（選用）指定執行個體的主機名稱，僅供 Stacks AWS OpsWorks 使用。預設值是執行個體的主機名稱。

--debug

(選用) 如果註冊程序失敗，則提供除錯資訊。如需故障診斷資訊，請參閱故障診斷執行個體註冊。

--use-instance-profile

（選用，但強烈建議 Amazon EC2 執行個體）讓register命令使用連接的執行個體描述檔，而不是建立 IAM 使用者。新增此參數有助於防止在意外刪除 IAM 使用者時嘗試管理已註冊執行個體時發生的錯誤。

> **Important**
>
> 在註冊程序期間建立的 IAM 使用者，在註冊執行個體的整個生命週期中都是必要的。刪除使用者會導致 AWS OpsWorks Stacks 代理程式無法與服務通訊。為了協助防止在使用者意外刪除時管理已註冊執行個體的問題，請將 --use-instance-profile 參數新增至

您的register命令，以改用執行個體的內建執行個體描述檔。新增 --use-instance-profile 參數也可防止每 90 天輪換 AWS 帳戶存取金鑰時發生錯誤（建議的最佳實務），因為它可防止 AWS OpsWorks 代理程式和必要使用者可用的存取金鑰不相符。

目標

(條件式) 如果您從工作站執行此命令，命令字串中的最終值會以下列其中一種方式指定註冊目標。

- 執行個體的公有 IP 地址。
- 執行個體的主機名稱。
- 對於 Amazon EC2 執行個體，則為執行個體 ID。

AWS OpsWorks Stacks 使用執行個體 ID 來取得執行個體組態，包括執行個體的公有 IP 地址。根據預設， AWS OpsWorks Stacks 使用此地址來建構用來登入執行個體的ssh命令字串。如果您需要連線至私有 IP 地址，您必須使用 --override-ssh 提供自訂命令字串。如需範例，請參閱「從工作站註冊現場部署執行個體」。

> **Note**
>
> 如果您指定主機名稱，ssh 需要 DNS 伺服器將名稱解析為特定執行個體。如果您不確定主機名稱是唯一的，請使用 ssh 確認主機名稱解析為正確的執行個體。

如果您從要註冊的執行個體執行此命令，請省略執行個體識別符，並改用 --local 引數。

下列引數僅適用於現場部署執行個體。

--override-public-ip

（選用） AWS OpsWorks Stacks 會將指定的地址顯示為執行個體的公有 IP 地址。它不會變更執行個體的公有 IP 地址。不過，如果使用者使用主控台連線至執行個體，例如在執行個體頁面上選擇地址， AWS OpsWorks Stacks 會使用指定的地址。 AWS OpsWorks Stacks 會自動判斷引數的預設值。

--override-private-ip

（選用） AWS OpsWorks Stacks 會將指定的地址顯示為執行個體的私有 IP 地址。它不會變更執行個體的私有 IP 地址。 AWS OpsWorks Stacks 會自動判斷引數的預設值。

register 命令範例

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 只有 Linux 堆疊支援此功能。

本節包含 `register` 命令字串的一些範例。

從工作站註冊 Amazon EC2 執行個體

下列範例會從工作站註冊 Amazon EC2 執行個體。命令字串使用預設登入資料，並透過其 Amazon EC2 執行個體 ID 識別執行個體。透過將 `ec2` 變更為 `on-premises`，即可將範例用於現場部署執行個體。

```
aws opsworks register \
  --region us-west-2 \
  --use-instance-profile \
  --infrastructure-class ec2 \
  --stack-id ad21bce6-7623-47f1-bf9d-af2affad8907 \
  --ssh-user-name my-sshusername \
  --ssh-private-key "./keys/mykeys.pem" \
  i-2422b9c5
```

從工作站註冊現場部署執行個體

下列範例會從個別工作站註冊現場部署執行個體。命令字串使用預設登入資料，並以指定的 `ssh` 命令字串登入執行個體。如果您的執行個體需要密碼，`register` 會提示您。您可以將 `on-premises`變更為 ，以使用 Amazon EC2 執行個體的範例`ec2`。

```
aws opsworks register \
  --region us-west-2 \
  --infrastructure-class on-premises \
```

```
  --stack-id ad21bce6-7623-47f1-bf9d-af2affad8907 \
  --override-ssh "ssh your-user@192.0.2.0"
```

> **Note**
>
> 您可以使用 `--override-ssh`指定任何自訂 SSH 命令字串。 AWS OpsWorks Stacks 接著會使用指定的字串登入執行個體，而不是建構命令字串。如需其他範例，請參閱使用自訂 SSH 命令字串註冊執行個體。

使用自訂 SSH 命令字串註冊執行個體

下列範例會從工作站註冊現場部署執行個體，並使用 `--override-ssh`引數來指定`register`使用登入執行個體的自訂 SSH 命令。此範例使用 `sshpass` 以使用者名稱和密碼登入，但您可以指定任何有效的 `ssh` 命令字串。

```
aws opsworks register \
  --region us-west-2 \
  --infrastructure-class on-premises \
  --stack-id 2f92ff9d-04f2-4728-879b-f4283b40783c \
  --override-ssh "sshpass -p 'mypassword' ssh your-user@192.0.2.0"
```

從執行個體執行 **`register`** 來註冊執行個體

下列範例示範如何`register`透過從執行個體本身執行 來註冊 Amazon EC2 執行個體。命令字串取決於其許可的預設登入資料。若要使用現場部署執行個體的範例，`--infrastructure-class`請變更為 `on-premises`。

```
aws opsworks register \
  --region us-west-2 \
  --infrastructure-class ec2 \
  --stack-id ad21bce6-7623-47f1-bf9d-af2affad8907 \
  --local
```

使用私有 IP 地址註冊執行個體

根據預設，`register` 使用執行個體的公有 IP 地址登入執行個體。若要使用私有 IP 地址註冊執行個體 (例如 VPC 之私有子網路中的執行個體)，您必須使用 `--override-ssh` 指定自訂 `ssh` 命令字串。

```
aws opsworks register \
```

```
--region us-west-2 \
--infrastructure-class ec2 \
--stack-id 2f92ff9d-04f2-4728-879b-f4283b40783c \
--override-ssh "ssh -i mykey.pem ec2-user@10.183.201.93" \
i-2422b9c5
```

執行個體註冊政策

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWSOpsWorksRegisterCLI_EC2 和 AWSOpsWorksRegisterCLI_OnPremises 政策分別提供適當的許可，來註冊 EC2 和現場部署執行個體。您可以將 AWSOpsWorksRegisterCLI_EC2 新增至您的 IAM 使用者以註冊 EC2 執行個體，但將 AWSOpsWorksRegisterCLI_OnPremises新增至您的使用者以註冊現場部署執行個體。若要使用這些政策，您至少必須執行 1.16.180 版 AWS CLI 或更新版本。

AWSOpsWorksRegisterCLI_EC2 政策

將 AWSOpsWorksRegisterCLI_EC2新增至您的使用者以註冊 EC2 執行個體。如果您計劃只註冊 EC2 執行個體，則應該使用此設定檔。當您使用此政策，許可是由 EC2 執行個體的執行個體設定檔提供。

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "opsworks:AssignInstance",
            "opsworks:CreateLayer",
            "opsworks:DeregisterInstance",
            "opsworks:DescribeInstances",
            "opsworks:DescribeStackProvisioningParameters",
            "opsworks:DescribeStacks",
            "opsworks:UnassignInstance"
```

```
        ],
        "Resource": [
          "*"
        ]
      },
      {
        "Effect": "Allow",
        "Action": [
          "ec2:DescribeInstances"
        ],
        "Resource": [
          "*"
        ]
      }
    ]
  }
```

AWSOpsWorksRegisterCLI_OnPremises 政策

AWSOpsWorksRegisterCLI_OnPremises 新增至您的使用者以註冊現場部署執行個體。此政策包含 IAM 許可，例如 AttachUserPolicy，但限制這些許可工作的資源。

```
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "opsworks:AssignInstance",
          "opsworks:CreateLayer",
          "opsworks:DeregisterInstance",
          "opsworks:DescribeInstances",
          "opsworks:DescribeStackProvisioningParameters",
          "opsworks:DescribeStacks",
          "opsworks:UnassignInstance"
        ],
        "Resource": [
          "*"
        ]
      },
      {
        "Effect": "Allow",
        "Action": [
```

```
            "ec2:DescribeInstances"
          ],
          "Resource": [
            "*"
          ]
        },
        {
          "Effect": "Allow",
          "Action": [
            "iam:CreateGroup",
            "iam:AddUserToGroup"
          ],
          "Resource": [
            "arn:aws:iam::*:group/AWS/OpsWorks/OpsWorks-*"
          ]
        },
        {
          "Effect": "Allow",
          "Action": [
            "iam:CreateUser",
            "iam:CreateAccessKey"
          ],
          "Resource": [
            "arn:aws:iam::*:user/AWS/OpsWorks/OpsWorks-*"
          ]
        },
        {
          "Effect": "Allow",
          "Action": [
            "iam:AttachUserPolicy"
          ],
          "Resource": [
            "arn:aws:iam::*:user/AWS/OpsWorks/OpsWorks-*"
          ],
          "Condition": {
            "ArnEquals":
              {
                "iam:PolicyARN": "arn:aws:iam::aws:policy/
AWSOpsWorksInstanceRegistration"
              }
            }
        }
      ]
```

```
    }
```

(已廢除) **AWSOpsWorksRegisterCLI** 政策

> **Important**
>
> AWSOpsWorksRegisterCLI 政策已廢除，因此無法用來註冊新的執行個體。它只適用於已註冊之執行個體的回溯相容性。此AWSOpsWorksRegisterCLI政策包含許多 IAM 許可CreateUser，包括 PutUserPolicy、 和 AddUserToGroup。由於這些是管理員層級許可，您應該只將 AWSOpsWorksRegisterCLI 政策指派給信任的管理使用者。

管理已註冊的執行個體

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 只有 Linux 堆疊支援此功能。

當您註冊執行個體時，它會成為 AWS OpsWorks Stacks 執行個體，而且您可以用與使用 Stacks AWS OpsWorks 建立的執行個體相同的方式來管理它。主要有兩點差異：

- 已註冊的執行個體不一定會指派給某 layer。
- 您可以取消註冊已註冊的執行個體，並將其交還給您直接控制。

註冊執行個體之後，執行個體即處於已註冊狀態。 AWS OpsWorks Stacks 為所有已註冊的執行個體提供下列管理功能：

- 運作狀態檢查 – AWS OpsWorks Stacks 會監控代理程式，以評估執行個體是否繼續運作。

如果執行個體未通過運作狀態檢查， AWS OpsWorks Stacks 會自動修復已註冊的 Amazon EC2 執行個體，並將已註冊現場部署執行個體的狀態變更為 `connection lost`。

- CloudWatch 監控 – 針對已註冊的執行個體啟用 CloudWatch 監控。

  您可以監控 CPU 使用率和可用記憶體等指標，並選擇性地在指標超過指定的閾值時收到通知。

- 使用者管理 – AWS OpsWorks Stacks 提供簡單的方法來指定哪些使用者可以存取執行個體，以及他們可以執行哪些操作。如需詳細資訊，請參閱管理使用者許可。
- 配方執行 – 您可以使用 Execute Recipes 堆疊命令在執行個體上執行 Chef 配方。
- 作業系統更新 – 您可以使用更新相依性堆疊命令來更新執行個體的作業系統。

若要充分利用 AWS OpsWorks Stacks 管理功能，您可以將執行個體指派給 layer。如需詳細資訊，請參閱將已註冊的執行個體指派給某 Layer。

Stacks AWS OpsWorks 管理 Amazon EC2 和內部部署執行個體的方式有所不同。

Amazon EC2 執行個體

- 如果您停止已註冊的 Amazon EC2 執行個體， AWS OpsWorks Stacks 會終止執行個體存放區後端執行個體，並停止 Amazon EBS 後端執行個體。

  執行個體仍處於已向堆疊註冊的狀態並已指派給其所在 layer，因此您可以視需要將它重新啟動。您必須取消註冊已註冊的執行個體來將它從堆疊中移除，可以明確進行或透過刪除執行個體，後者會自動將它取消註冊。

- 如果您重新啟動已註冊的 Amazon EC2 執行個體，或執行個體失敗且自動修復，則結果與使用 Amazon EC2 停止和重新啟動執行個體相同。請注意下列差異：
  - 執行個體存放區後端執行個體 – AWS OpsWorks Stacks 使用相同的 AMI 啟動新的執行個體。

    請注意，Stacks AWS OpsWorks 對您在執行個體註冊之前執行的任何操作一無所知，例如安裝軟體套件。如果您希望 AWS OpsWorks Stacks 在啟動時安裝套件或執行其他組態任務，您必須提供自訂 Chef 配方來執行所需的任務，並將它們指派給適當的 layer 設定事件。

  - Amazon EBS 後端執行個體 – AWS OpsWorks Stacks 會使用相同的 AMI 啟動新的執行個體，並重新連接根磁碟區，將執行個體還原為先前的組態。
- 如果您取消註冊已註冊的 Amazon EC2 執行個體，則會傳回為一般 Amazon EC2 執行個體。

現場部署執行個體

- AWS OpsWorks Stacks 無法停止或啟動已註冊的內部部署執行個體。

取消指派已註冊的內部部署執行個體會觸發關機事件。不過，該事件只會執行所指派 layer 的關機配方。它們會執行關閉服務等任務，但不會停止執行個體。

- AWS OpsWorks 如果已註冊的內部部署執行個體失敗， Stacks 無法自動修復，但執行個體會標記為連線遺失。
- 內部部署執行個體無法使用 Elastic Load Balancing、Amazon EBS 或 Elastic IP 地址服務。

將已註冊的執行個體指派給某 Layer

Important

AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Note

只有 Linux 堆疊支援此功能。

註冊執行個體之後，您可以將其指派給一或多 layer。將執行個體指派給 layer 而非未指派的優勢是，您可以將自訂配方指派給 layer 的生命週期事件。 AWS OpsWorks Stacks 接著會在 layer 的配方之後的適當時間自動執行。

- 您可以將已註冊的任何執行個體指派給自訂 layer。自訂 layer 包含不會安裝任何套件的一組最少配方，因此應該不會與執行個體的現有組態產生任何衝突。
- 您可以將內部部署執行個體指派給 AWS OpsWorks Stacks 內建層。

  每個內建 layer 包含自動安裝一或多個套件的配方。例如，Java App Server 設定配方會安裝 Apache 和 Tomcat。該 layer 的配方也可能會執行其他操作，例如重新啟動服務及部署應用程式。在將現場部署執行個體指派給內建 layer 之前，您應該確保 layer 的配方不會產生任何衝突，例如嘗試安裝與執行個體目前不同的應用程式伺服器版本。如需詳細資訊，請參閱 層 和 AWS OpsWorks Stacks Layer 參考。

將已註冊的執行個體指派給某 layer

1. 新增您要使用堆疊的 layer (如果您尚未這樣做)。
2. 在導覽窗格中選擇執行個體，然後在執行個體的動作欄中選擇指派。
3. 選取適當 layer，然後選擇 Save (儲存)。

當您將執行個體指派給 layer AWS OpsWorks Stacks 時，請執行下列動作。

- 執行該 layer 的安裝配方。
- 將任何連接的彈性 IP 地址或 Amazon EBS 磁碟區新增至堆疊的資源。

  然後，您可以使用 AWS OpsWorks Stacks 來管理這些資源。如需詳細資訊，請參閱資源管理。

完成之後，執行個體會處於線上狀態，並完全併入堆疊中。然後，當生命週期事件發生時， AWS OpsWorks Stacks 會執行 layer 的指派配方。

取消指派已註冊的執行個體

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

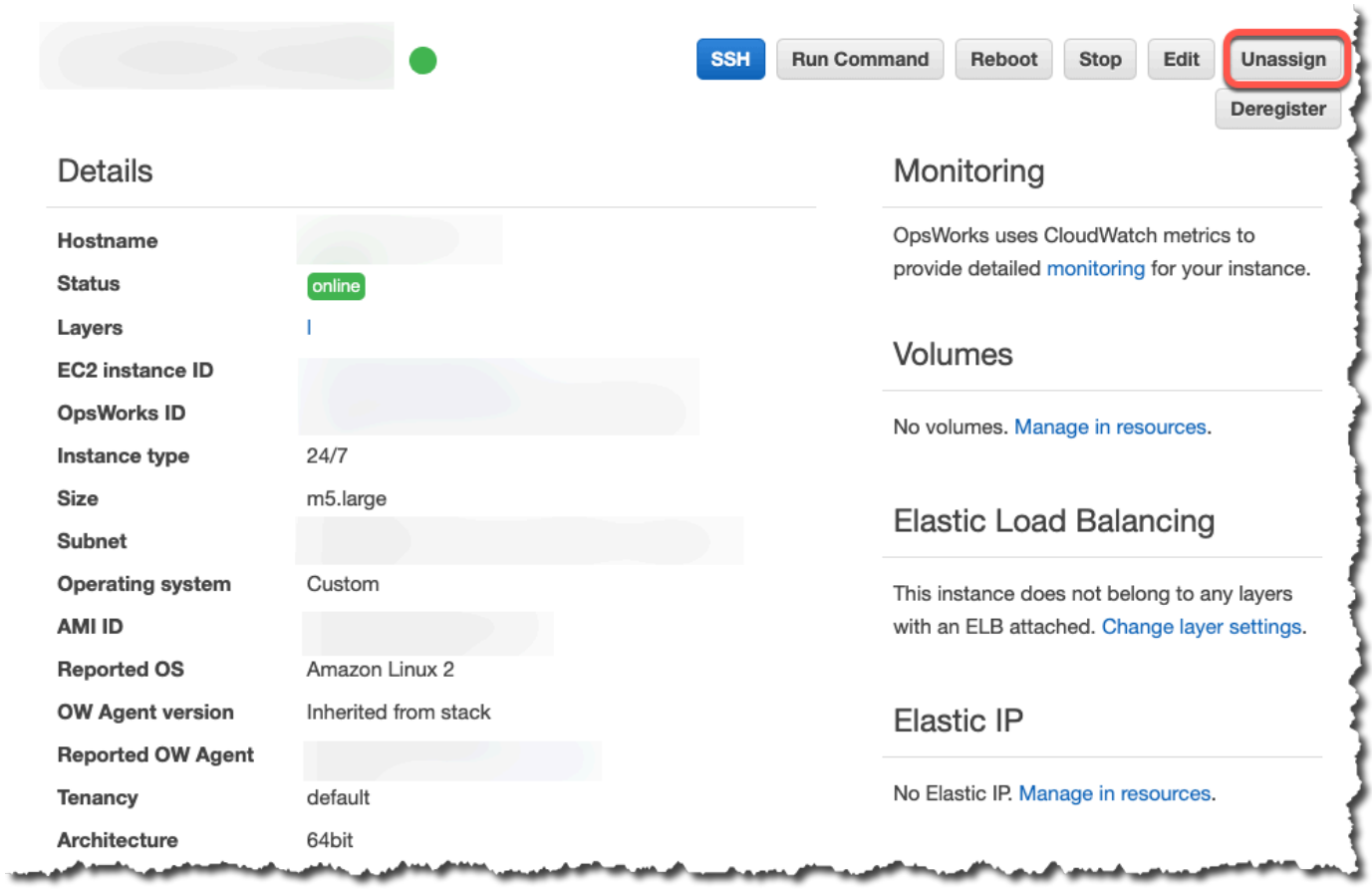
> **Note**
>
> 只有 Linux 堆疊支援此功能。

您可以使用 AWS OpsWorks 主控台或 SDK 操作 AWS CLI，從其層取消指派已註冊的執行個體。

當您取消指派執行個體時， AWS OpsWorks Stacks 會在執行個體上執行 layer 的關機配方。這些配方會執行關閉服務等任務，但不會停止執行個體。如果將執行個體指派給多 layer，取消指派會套用至每一 layer；您無法將執行個體從其一小部分 layer 取消指派。不過，執行個體仍處於已向堆疊註冊的狀態，因此您可以視需要將其指派給另一 layer。

使用主控台取消指派已註冊的執行個體

1. 在導覽窗格中，選擇 Instances (執行個體)。
2. 選擇您要取消指派的執行個體。
3. 在執行個體的詳細資訊頁面上，選擇取消指派。



使用 取消指派已註冊的執行個體 AWS CLI

執行 aws opsworks unassign-instance命令，從使用執行個體的所有層取消指派已註冊的執行個體。

```
aws opsworks unassign-instance --region region --instance-id instance-id
```

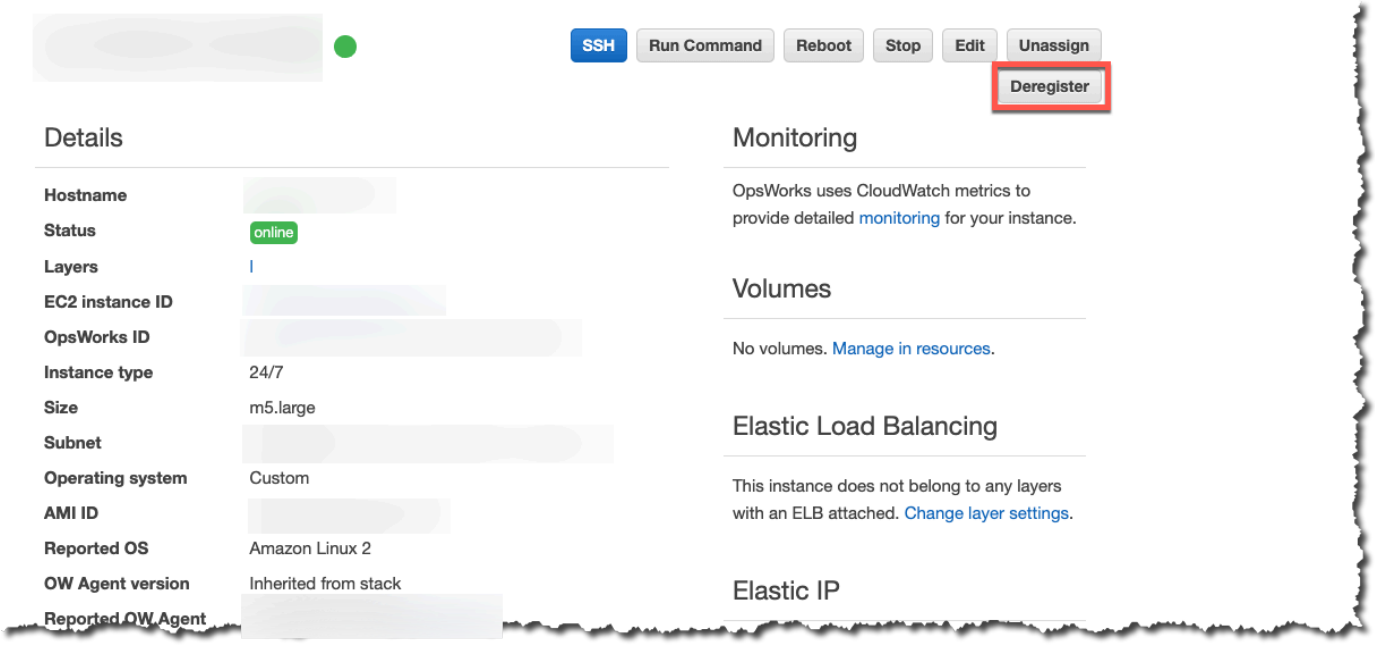
取消註冊已註冊的執行個體

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以使用 AWS OpsWorks 主控台 AWS CLI或 SDK 操作來取消註冊執行個體。

使用主控台取消註冊執行個體

1. 在導覽窗格中，選擇 Instances (執行個體)。
2. 選擇您要取消註冊的執行個體。
3. 在執行個體的詳細資訊頁面上，選擇取消註冊。



使用 取消註冊執行個體 AWS CLI

執行 aws opsworks deregister-instance命令，從其堆疊取消註冊執行個體。

```
aws opsworks deregister-instance --region region --instance-id instance-id
```

當您取消註冊執行個體時， AWS OpsWorks Stacks 會執行下列動作：

- 將執行個體從堆疊中移除。
- 將執行個體從任何指派 layer 取消指派。
- 關閉並解除安裝代理程式。
- 取消註冊任何連接的資源（彈性 IP 地址和 Amazon EBS 磁碟區）。

  此程序包含註冊前連接至執行個體的資源，以及您使用 Stacks AWS OpsWorks 在作為堆疊一部分時連接至執行個體的資源。取消註冊之後，資源將不再是堆疊資源的一部分，但會保持連接至執行個體。
- 停止收取現場部署執行個體的費用。
- 移除 OpsWorks 新增至執行個體的所有標籤。

執行個體會維持在執行中狀態，但由您直接控制，且不再由 Stacks AWS OpsWorks 管理。

> **Note**
>
> 只有在 Linux 堆疊中完全支援註冊和取消註冊電腦或執行個體。對於 Windows 堆疊，允許取消註冊執行個體，但不會從執行個體解除安裝 OpsWorks 代理程式。取消註冊不會移除所有變更的檔案，也不會完整還原為特定檔案的備份副本。此清單適用於 Chef 11.10 和 Chef 12 堆疊；以下說明這兩種版本之間的差異。
>
> - `/etc/hosts` 會備份至 `/var/lib/aws/opsworks/local-mode-cache/backup/etc/`，但不會還原。
> - `aws` 和 `opsworks` 項目會保留在 passwd、group 和 shadow 檔案等。
> - `/etc/sudoers` 包含 Stacks AWS OpsWorks 目錄的參考。
> - 下列檔案會安全留下；若要長期保留，請考慮刪除 `/var/lib/aws/opsworks`。
>   - `/var/log/aws/opsworks` 會保留在 Chef 11.10 堆疊中的執行個體上。
>   - `/var/lib/aws/opsworks` 會保留在 Chef 11.10 和 Chef 12 堆疊上。
>   - `/var/chef` 會保留在 Chef 12 堆疊中的執行個體上。
> - 其他留下的檔案：
>   - `/etc/logrotate.d/opsworks-agent`
>   - `/etc/cron.d/opsworks-agent-updater`
>   - `/etc/ld.so.conf.d/opsworks-user-space.conf`

- `/etc/motd.opsworks-static`
- `/etc/aws/opsworks`
- `/etc/sudoers.d/opsworks`
- `/etc/sudoers.d/opsworks-agent`

已註冊執行個體的生命週期

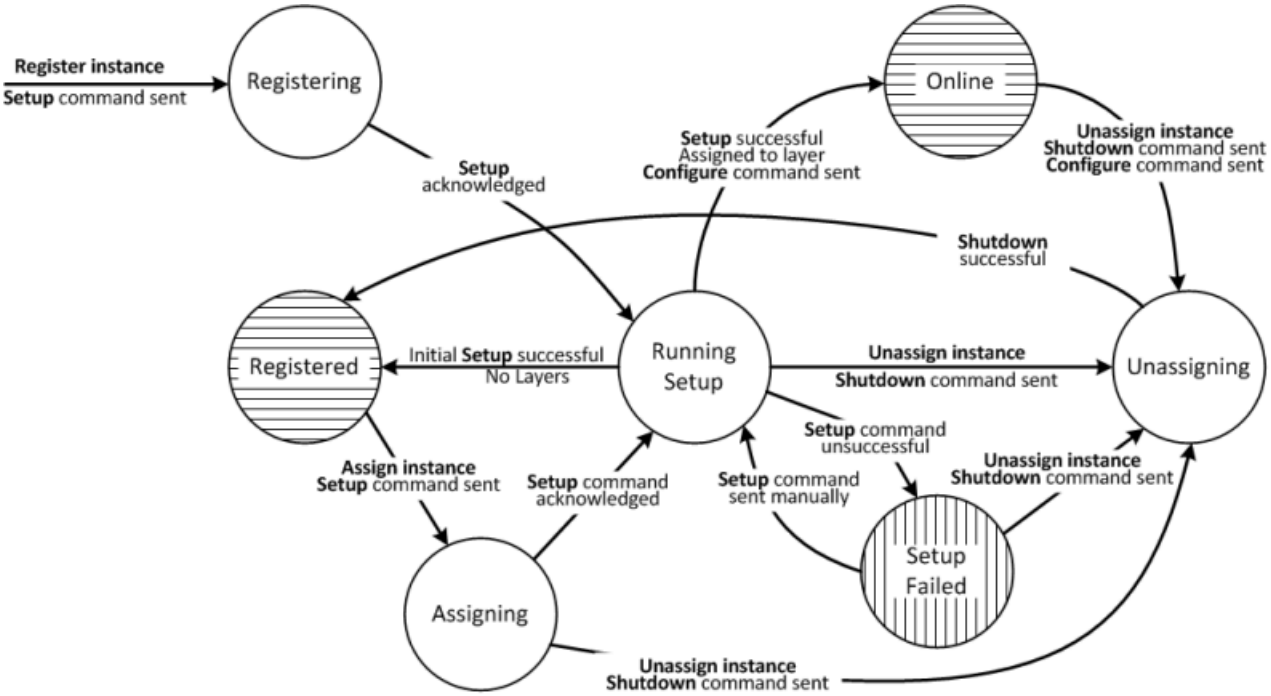
Important

AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Note

只有 Linux 堆疊支援此功能。

已註冊執行個體的生命週期是從安裝及執行代理程式之後開始。此時，它會指示 AWS OpsWorks Stacks 向堆疊註冊執行個體。下列狀態圖表摘要說明生命週期的主要元素。



每個狀態對應至一個執行個體狀態。邊緣代表下列其中一個 AWS OpsWorks Stacks 命令。下列各節將會詳細討論。

- 設定 – 此命令對應至設定生命週期事件，並執行執行個體的設定配方。
- 設定 – 此命令對應至設定生命週期事件。

  AWS OpsWorks 當執行個體進入或離開線上狀態時，Stacks 會在堆疊中的每個執行個體上觸發此事件。執行個體會執行其設定配方，這會進行任何必要變更，以配合新的執行個體。
- 關機 – 此命令對應至關機生命週期事件，該事件會執行執行個體的關機配方。

  這些配方會執行關閉服務等任務，但不會停止執行個體。
- 取消註冊 – 此命令會取消註冊執行個體，且不會對應至生命週期事件。

> **Note**
>
> 為求簡化，此圖表不會顯示 Deregistering (正在取消註冊) 和 Deleted (已刪除) 狀態。您可以取消註冊圖表中任何狀態的執行個體，這會將取消註冊命令傳送至執行個體，並將其移至 Deregistering (正在取消註冊) 狀態。
>
> - 如果您取消註冊線上執行個體， AWS OpsWorks Stacks 會將設定命令傳送至堆疊中的其餘執行個體，以通知他們執行個體正在離線。

- 確認取消註冊命令之後，執行個體仍在執行中，但會處於 Deleted (已刪除) 狀態，且不再是堆疊的一部分。如果您想要將執行個體再次納入堆疊，您必須重新註冊。

主題

- 正在註冊
- 安裝執行中
- 已登記
- 正在指派
- 線上
- 安裝失敗
- 正在取消指派
- 初始安裝組態變更

正在註冊

代理程式傳送註冊請求後， AWS OpsWorks Stacks 會透過將設定命令傳送至執行個體，使其處於註冊狀態，來啟動執行個體生命週期。執行個體確認安裝命令之後，便會移至 安裝執行中 狀態。

安裝執行中

Running Setup (安裝執行中) 狀態會執行執行個體的安裝配方。安裝運作與否，取決於先前狀態。

> **Note**
>
> 如果您在執行個體處於執行中設定狀態時取消指派執行個體， AWS OpsWorks Stacks 會傳送 Shutdown 命令，該命令會執行執行個體的 Shutdown 配方，但不會停止執行個體。執行個體會移至 正在取消指派 狀態。

主題

- 正在註冊
- 正在指派
- 安裝失敗

正在註冊

在註冊程序期間， 設定會建立 AWS OpsWorks Stacks 執行個體來代表堆疊中的已註冊執行個體，並在執行個體上執行一組核心設定配方。

初始安裝所執行的一項重要變更是覆寫執行個體的主機檔案。藉由註冊執行個體，您會將使用者管理移交給 AWS OpsWorks Stacks，其必須具有自己的主機檔案才能控制 SSH 登入許可。初始安裝也會建立或修改一些檔案，並在 Ubuntu 系統上修改套件來源及安裝一組套件。如需詳細資訊，請參閱 初始安裝組態變更。

在註冊期間， 程序會呼叫 IAM`AttachUserPolicy`，這是附加至您建立為先決條件之 IAM 使用者之許可的一部分。如果 `AttachUserPolicy` 不存在 (最可能的原因是您執行舊版 AWS CLI)，則程序會回到呼叫 `PutUserPolicy`。

> **Note**
>
> 為了保持一致性， AWS OpsWorks Stacks 會執行每個核心設定配方。不過，其中一些配方只會在執行個體已指派給至少一 layer 時執行其部分或所有任務，因此不一定會影響初始安裝。

- 如果安裝成功，執行個體會移至 已登記 狀態。
- 如果安裝失敗，執行個體會移至 安裝失敗 狀態。

正在指派

執行個體至少有一個指派的 layer。 AWS OpsWorks Stacks 會執行每個 layer 的設定配方，包括您指派給 layer 設定事件的任何自訂配方。

- 如果安裝成功，執行個體會移至 Online (線上) 狀態，而且 AWS OpsWorks Stacks 會在堆疊中的每個執行個體上觸發設定生命週期事件，以通知它們有此新的執行個體。
- 如果安裝失敗，執行個體會移至 Setup Failed (安裝失敗) 狀態。

> **Note**
>
> 此安裝程序會再次執行核心配方。不過，Chef 配方為等冪操作，因此不會重複已執行的任何任務。

安裝失敗

如果 正在指派 狀態之執行個體的安裝程序失敗，您可以使用安裝堆疊命令再試一次，來手動重新執行執行個體的安裝配方。

- 如果安裝成功，指派的執行個體會移至 線上 狀態，而且 AWS OpsWorks Stacks 會在堆疊中的每個執行個體上觸發設定生命週期事件，以通知它們有此新的執行個體。
- 如果安裝嘗試失敗，執行個體會移回 Setup Failed (安裝失敗) 狀態。

已登記

處於已註冊狀態的執行個體是堆疊的一部分，由 Stacks AWS OpsWorks 管理，但不會指派給 layer。它們可以無限期地保持在此狀態。

如果您將執行個體指派給一或多個圖層， AWS OpsWorks Stacks 會傳送設定命令給執行個體，並移至 正在指派 狀態。

正在指派

執行個體確認安裝命令之後，便會移至 安裝執行中 狀態。

如果您在執行個體處於指派狀態時取消指派執行個體， AWS OpsWorks Stacks 會終止設定程序並傳送 Shutdown 命令。執行個體會移至 正在取消指派 狀態。

線上

執行個體現在是至少一 layer 的成員，並會視為一般 AWS OpsWorks Stacks 執行個體。它可以無限期地保持在此狀態。

如果您在執行個體處於線上狀態時取消指派執行個體， AWS OpsWorks Stacks 會將 Shutdown 命令傳送至執行個體，並將 Configure 命令傳送至堆疊的其餘執行個體。執行個體會移至 正在取消指派 狀態。

安裝失敗

安裝命令失敗。

- 您可以執行安裝堆疊命令再試一次。

  執行個體會返回 安裝執行中 狀態。
- 如果您取消指派執行個體， AWS OpsWorks Stacks 會將 Shutdown 命令傳送至執行個體。

執行個體會移至 正在取消指派 狀態。

正在取消指派

關機命令完成之後，執行個體將不再指派給任一 layer，並會返回 已登記 狀態。

> **Note**
>
> 如果將執行個體指派給多 layer，取消指派會套用至每一 layer；您無法取消指派一小部分指派 layer。如果您想要一組不同的指派 layer，請取消指派執行個體，然後重新指派所需 layer。

初始安裝組態變更

初始安裝會建立或修改所有已註冊執行個體上的下列檔案和目錄。

建立的檔案

```
/etc/apt/apt.conf.d/99-no-pipelining
/etc/aws/
/etc/init.d/opsworks-agent
/etc/motd
/etc/motd.opsworks-static
/etc/sudoers.d/opsworks
/etc/sudoers.d/opsworks-agent
/etc/sysctl.d/70-opsworks-defaults.conf
/opt/aws/opsworks/
/usr/sbin/opsworks-agent-cli
/var/lib/aws/
/var/log/aws/
/vol/
```

修改的檔案

```
/etc/apt/apt.conf.d/99-no-pipelining
/etc/crontab
/etc/default/monit
/etc/group
/etc/gshadow
/etc/monit/monitrc
```

```
/etc/passwd
/etc/security/limits.conf (removing limits only for EC2 micro instances)
/etc/shadow
/etc/sudoers
```

初始設定也會在 Amazon EC2 微型執行個體上建立交換檔案。

初始安裝會對 Ubuntu 系統進行下列變更。

套件來源

初始安裝會將套件來源變更如下。

- deb http://archive.ubuntu.com/ubuntu/ ${code_name} main universe

  收件人：deb-src http://archive.ubuntu.com/ubuntu/ ${code_name} main universe
- deb http://archive.ubuntu.com/ubuntu/ ${code_name}-updates main universe

  收件人：deb-src http://archive.ubuntu.com/ubuntu/ ${code_name}-updates main universe
- deb http://archive.ubuntu.com/ubuntu ${code_name}-security main universe

  收件人：deb-src http://archive.ubuntu.com/ubuntu ${code_name}-security main universe
- deb http://archive.ubuntu.com/ubuntu/ ${code_name}-updates multiverse

  收件人：deb-src http://archive.ubuntu.com/ubuntu/ ${code_name}-updates multiverse
- deb http://archive.ubuntu.com/ubuntu ${code_name}-security multiverse

  收件人：deb-src http://archive.ubuntu.com/ubuntu ${code_name}-security multiverse
- deb http://archive.ubuntu.com/ubuntu/ ${code_name} multiverse

  收件人：deb-src http://archive.ubuntu.com/ubuntu/ ${code_name} multiverse

- `deb http://security.ubuntu.com/ubuntu ${code_name}-security multiverse`

  收件人：`deb-src http://security.ubuntu.com/ubuntu ${code_name}-security multiverse`

套件

初始安裝會解除安裝 `landscape` 並安裝下列套件。

autofs	libicu-dev	libopenssl-ruby
libssl-dev	libxml2-dev	libxslt-dev
libyaml-dev	monit	ntpd
procps	ruby	ruby-dev
rubygems	screen	sqlite
vim	xfs	

編輯執行個體組態

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以編輯執行個體組態，包括已註冊的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體，但有下列限制：

- 執行個體必須處於停止狀態。

  雖然您無法修改線上執行個體的屬性，但您可以透過編輯執行個體的 layer 來變更其組態的一部分。如需詳細資訊，請參閱編輯 OpsWorks Layer 的組態。
- 有些設定 (例如 Availability Zone (可用區域) 和 Scaling Type (擴展類型)) 是在您建立執行個體時決定的，之後便無法修改。

- 某些設定只能針對執行個體後端執行個體進行修改，不適用於 Amazon Elastic Block Store 後端執行個體。

  例如，您可以變更執行個體後端執行個體的作業系統。Amazon EBS 後端執行個體必須使用您在建立執行個體時指定的作業系統。如需執行個體儲存體的詳細資訊，請參閱儲存體。

- 根據預設，執行個體會繼承堆疊的代理程式版本設定。

  您可以使用 OpsWorks Agent Version (OpsWorks 代理程式版本) 覆寫堆疊的代理程式版本設定，並為執行個體指定特定的代理程式版本。如果您指定執行個體的代理程式版本， AWS OpsWorks 即使堆疊的代理程式版本設定為自動更新，Stacks 也不會在有新版本可用時自動更新代理程式。您必須透過編輯執行個體組態來手動更新執行個體的代理程式版本。然後， AWS OpsWorks Stacks 會在執行個體上安裝指定的代理程式版本。

> **Note**
>
> 您無法編輯已註冊現場部署執行個體的組態。

編輯執行個體組態

1. 停止執行個體 (若還未停止的話)。
2. 在 Instances (執行個體) 頁面上，按一下執行個體名稱以顯示 Details (詳細資訊) 頁面。
3. 按一下 Edit (編輯) 顯示編輯頁面。
4. 妥善編輯執行個體的組態。

如需 Host name (主機名稱)、Size (大小)、SSH key (SSH 金鑰) 和 Operating system (作業系統) 設定的說明，請參閱將執行個體新增至 Layer。Layers (Layer) 設定可讓您新增或移除 layer。執行個體目前的 layer 會出現在 layer 的清單之後。

- 若要新增另外一 layer，請從清單中選取它。
- 若要從其中一個 layer 移除執行個體，請按一下適當 layer 旁邊的 x。

  執行個體必須至少是一個 layer 的成員，因此您無法移除最後一個 layer。

當您重新啟動執行個體時， AWS OpsWorks Stacks 會使用更新後的組態啟動新的 Amazon EC2 執行個體。

刪除 AWS OpsWorks Stacks 執行個體

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以使用 AWS OpsWorks Stacks 來停止執行個體，包括已註冊的 Amazon EC2 執行個體。執行此作業會停止 EC2 執行個體，但執行個體仍會留在堆疊中。您可以按一下執行個體動作欄中的開始來重新啟動它。如果您不再需要執行個體，並想要從堆疊中移除執行個體，則可以將其刪除，這會從堆疊中移除執行個體，並終止相關聯的 Amazon EC2 執行個體。刪除執行個體也會刪除任何相關聯的日誌或資料，以及執行個體上的任何 Amazon Elastic Block Store (EBS) 磁碟區。

> **Important**
>
> 本主題僅適用於由 Stacks 管理的 Amazon EC2 AWS OpsWorks 執行個體。如需如何刪除由 Amazon EC2 主控台或 API 管理的執行個體的詳細資訊，請參閱終止執行個體。

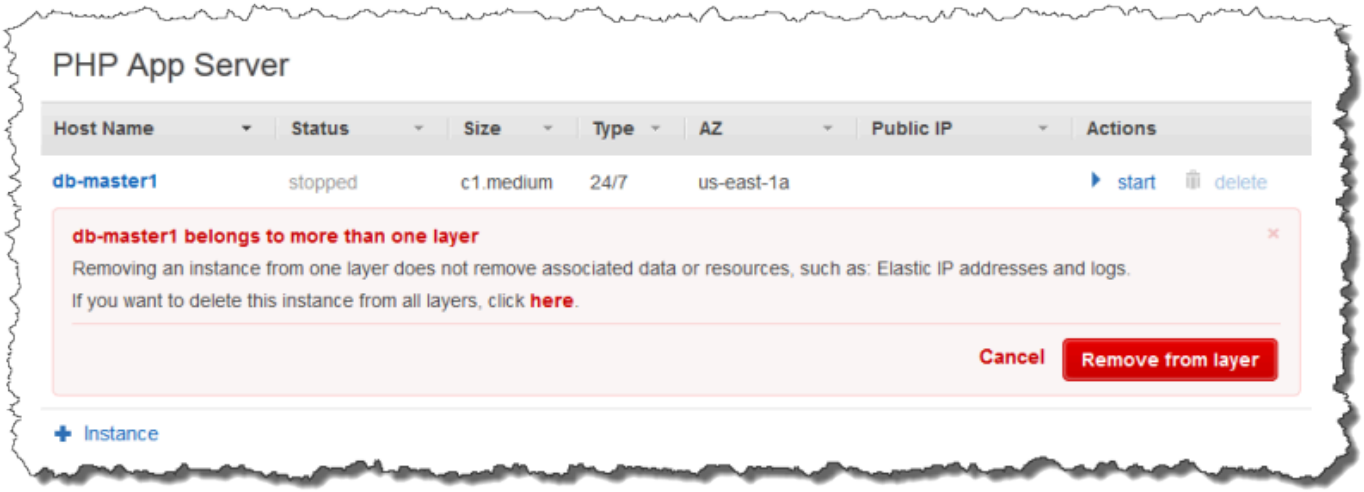
> **Note**
>
> 您無法使用 AWS OpsWorks Stacks 刪除已註冊的內部部署執行個體。

若執行個體屬於多個 layer，您可以從堆疊刪除執行個體，或是只移除特定的 layer。您也可以透過編輯執行個體組態，來從執行個體移除 layer，如編輯執行個體組態中所說明。

> **Important**
>
> 您只能使用 AWS OpsWorks Stacks 主控台或 API 刪除 AWS OpsWorks Stacks 執行個體。特別是，您不應使用 AWS OpsWorks Amazon EC2 主控台或 API 刪除 Stacks 執行個體，因為 Amazon EC2 AWS OpsWorks 動作不會自動與 Stacks 同步。例如，如果已啟用自動修復，而您使用 Amazon EC2 主控台終止執行個體， AWS OpsWorks Stacks 會將終止的執行個體視為失敗的執行個體，並啟動另一個 Amazon EC2 執行個體來取代它。如需詳細資訊，請參閱使用自動修復。

刪除執行個體

1. 在 Instances (執行個體) 頁面上，在適當的 layer 底下尋找執行個體。若執行個體正在執行中，請在 Actions (動作) 資料行中按一下 stop (停止)。
2. 在狀態變更為 stopped (已停止) 時，按一下 delete (刪除)。如果執行個體是多個 layer 的成員，layer AWS OpsWorks Stacks 會顯示下列區段。



   - 若要僅從選取的 layer 移除執行個體，請按一下 Remove from layer (從 layer 移除)。

     執行個體仍然會是其他 layer 的成員，可重新啟動。
   - 若要從所有 layer 刪除執行個體，並從堆疊中移除它，請按一下 here (這裡)。
3. 如果您選擇從堆疊中完全移除執行個體，或者執行個體只是一個 layer 的成員， AWS OpsWorks Stacks 會提示您確認刪除。

   選擇 Delete (刪除)，確認刪除。除了從堆疊刪除執行個體之外，此動作會刪除所有相關的日誌或資料，以及附加至執行個體的根磁碟區。若要移除所有執行個體磁碟區，請選擇刪除執行個體的 EBS 磁碟區 （ 快照將不會刪除 ），然後再選擇刪除。

使用 SSH 登入 Linux 執行個體

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以搭配內建的 MindTerm 用戶端或第三方用戶端 (例如 PuTTY)，使用 SSH 登入您的線上 Linux 執行個體。SSH 通常依存於身分驗證的 RSA 金鑰對。您會在執行個體上安裝公有金鑰，並將對應的私有金鑰提供給 SSH 用戶端。 AWS OpsWorks Stacks 會為您處理在堆疊的執行個體上安裝公有金鑰，如下所示。

- Amazon Elastic Compute Cloud (Amazon EC2) 金鑰對 – 如果堆疊的區域有一或多個 Amazon EC2 金鑰對，您可以指定堆疊的預設 SSH 金鑰對。

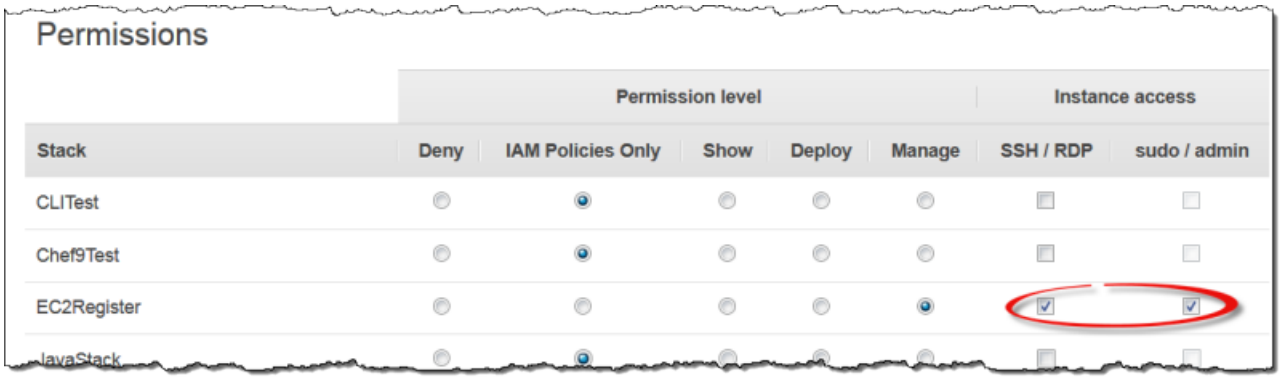
  您可以選擇性的在建立執行個體時覆寫預設金鑰對，及指定不同的金鑰對。無論哪種情況， AWS OpsWorks Stacks 都會在執行個體上安裝指定的金鑰對公有金鑰。如需如何建立 Amazon EC2 金鑰對的詳細資訊，請參閱 Amazon EC2 金鑰對。
- 個人金鑰對 – 每個使用者可以向 AWS OpsWorks Stacks 註冊個人金鑰對。

  使用者或管理員向 Stacks AWS OpsWorks 註冊公有金鑰，使用者將私有金鑰存放在本機。在設定堆疊的許可時，管理員會指定哪些使用者應具備堆疊執行個體的 SSH 存取。 AWS OpsWorks Stacks 會自動為每個授權使用者在堆疊的執行個體上建立系統使用者，並安裝使用者的個人公有金鑰。

使用者必須具備 SSH 授權，才能使用 MindTerm SSH 用戶端，或使用他們的個人金鑰對登入堆疊的執行個體。

授權使用者的 SSH

1. 在 AWS OpsWorks 堆疊導覽窗格中，按一下許可。
2. 為所需的 IAM 使用者選取 SSH/RDP，以授予必要的許可。如果您想要允許使用者使用 sudo 提升權限，例如執行代理程式 CLI 命令，請同時選取 sudo/admin。



如需如何使用 AWS OpsWorks Stacks 管理 SSH 存取的詳細資訊，請參閱 管理 SSH 存取。

主題

- 使用內建的 MindTerm SSH 用戶端
- 使用第三方 SSH 用戶端

使用內建的 MindTerm SSH 用戶端

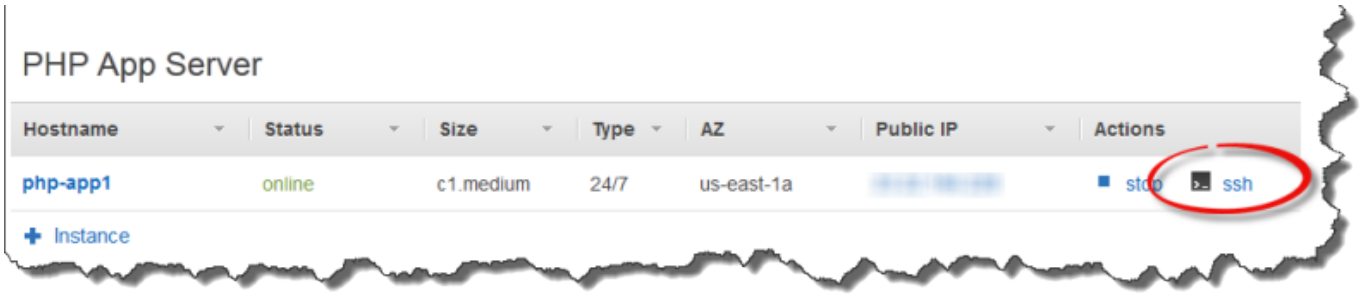
> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

登入 Linux 執行個體最簡單的方式，便是使用內建的 MindTerm SSH 用戶端。每個線上執行個體都包含一個 ssh 動作，可讓您用來啟動 MindTerm 用戶端。

> **Note**
>
> 您必須在您的瀏覽器中啟用 Java，才能使用 MindTerm 用戶端。

使用 MindTerm 用戶端登入

1. 若您尚未執行此作業，請授權要連線到執行個體之 IAM 使用者的 SSH 存取，如上一節所述。
2. 以使用者身分登入。
3. 在 Instances (執行個體) 頁面上，在適當執行個體的 Actions (動作) 資料行中選擇 ssh。



4. 對於私有金鑰，提供使用者個人私有金鑰或 Amazon EC2 私有金鑰的路徑，取決於您在執行個體上安裝的公有金鑰。
5. 選擇 Launch Mindterm (啟動 Mindterm)，然後使用終端機視窗在執行個體上執行命令。

使用第三方 SSH 用戶端

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您也可以使用第三方 SSH 用戶端 (例如 PuTTY) 連線到 Linux 執行個體。

使用第三方 SSH 用戶端

1. 請確定 AWS OpsWorks Stacks 已在執行個體上安裝 Amazon EC2 公有金鑰或 IAM 使用者的個人公有金鑰，如前所述。
2. 從其詳細資訊頁面，取得執行個體的公有 DNS 名稱或公有 IP 地址。
3. 根據作業系統，提供用戶端執行個體的主機名稱，如下所示：

   - Amazon Linux 和 Red Hat Enterprise Linux (RHEL) – `ec2-user@`*`DNSName/Address`*`.`
   - Ubuntu – `ubuntu@`*`DNSName/Address`*。

   將 *`DNSName/Address`* 取代為先前步驟中的公有 DNS 名稱或 IP 地址。
4. 提供用戶端對應至已安裝公有金鑰的私有金鑰。您可以根據執行個體上安裝的公有金鑰，使用 Amazon EC2 私有金鑰或 IAM 使用者的個人私有金鑰。

使用 RDP 登入 Windows 執行個體

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以使用 Windows 遠端桌面通訊協定 (RDP) 登入線上 Windows 執行個體，如下所示：

- 執行個體必須具有允許 RDP 存取之入站規則的安全群組。

如需使用安全群組的詳細資訊，請參閱使用安全群組。

- 一般使用者 – AWS OpsWorks Stacks 提供授權的一般使用者 RDP 密碼，有效期間有限，範圍從 30 分鐘到 12 小時不等。

  除了獲得授權之外，使用者必須至少擁有顯示許可層級，或其連接的 AWS Identity and Access Management (IAM) 政策必須允許 `opsworks:GrantAccess`動作。
- 管理員 – 您可以使用管理員密碼無限時間登入。

  如稍後所述，如果您已為執行個體指定 Amazon Elastic Compute Cloud (Amazon EC2) 金鑰對，您可以使用它來擷取管理員密碼。

> **Note**
>
> 本主題說明如何使用 Windows 遠端桌面連線用戶端從 Windows 工作站登入。您也可以使用其中一個 Linux 或 OS X 可用的 RDP 用戶端，但程序可能會有些不同。如需與 Microsoft Windows Server 2012 R2 相容之 RDP 用戶端的詳細資訊，請參閱 Microsoft 遠端桌面用戶端。

主題

- 提供允許 RDP 存取的安全群組
- 以一般使用者身分登入
- 以管理員身分登入

提供允許 RDP 存取的安全群組

在您使用 RDP 登入 Windows 執行個體前，執行個體的安全群組入站規則必須允許 RDP 連線。當您在區域中建立第一個堆疊時， AWS OpsWorks Stacks 會建立一組安全群組。其中包括一個名為 的物件，例如 `AWS-OpsWorks-RDP-Server`，Stacks AWS OpsWorks 會連接到所有 Windows 執行個體以允許 RDP 存取。不過，此安全群組預設沒有任何規則，因此您必須新增傳入規則來允許 RDP 存取您的執行個體。

允許 RDP 存取

1. 開啟 Amazon EC2 主控台，將其設定為堆疊的區域，然後從導覽窗格中選擇安全群組。
2. 選取 AWS-OpsWorks-RDP-Server，選擇 Inbound (入站) 標籤，然後選擇 Edit (編輯)。

3. 選擇 Add Rule (新增規則)，然後指定下列設定：

   - 類型 – RDP
   - 來源 – 允許的來源 IP 地址。

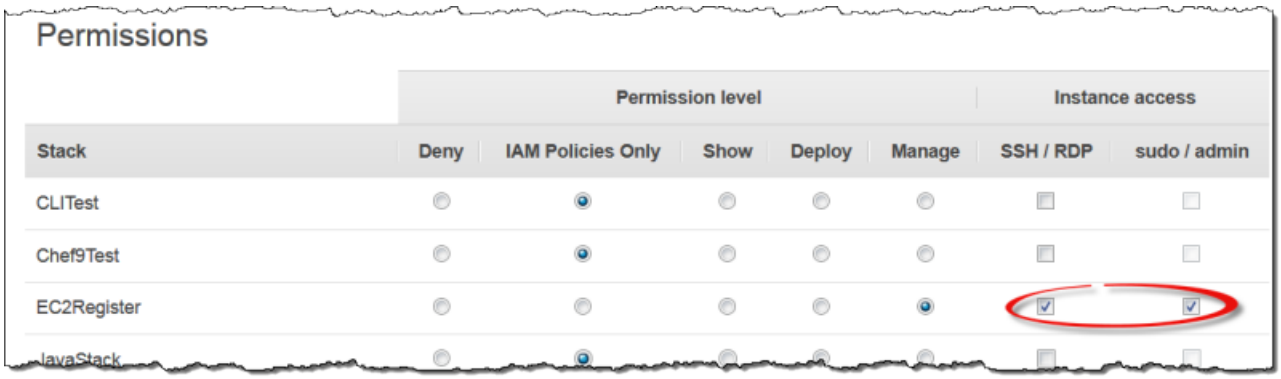
     通常，您會允許來自您 IP 地址或指定 IP 地址範圍 (通常是公司的 IP 地址範圍) 的傳入 RDP 請求。

以一般使用者身分登入

授權使用者可使用 AWS OpsWorks Stacks 提供的暫時密碼登入執行個體。

授權使用者的 RDP；

1. 在 AWS OpsWorks 堆疊導覽窗格中，按一下許可。
2. 選取所需使用者的 SSH/RDP 核取方塊，以授予必要的許可。若您希望使用者具有管理員許可，您也應選取 sudo/admin。



已授權的使用者可登入任何堆疊的線上執行個體，如下所示。

以一般 IAM 使用者身分登入

1. 以 IAM 使用者身分登入。
2. 在 Instances (執行個體) 頁面上，在適當執行個體的 Actions (動作) 資料行中選擇 rdp。
3. 指定工作階段長度，範圍可從 30 分鐘至 12 小時，然後選擇 Generate Password (產生密碼)。密碼只會在指定的工作階段期間內有效。
4. 記錄 public DNS name (公有 DNS 名稱)、username (使用者名稱) 及 password (密碼) 的值，然後選擇 Acknowledge and close (確認並關閉)。

5. 開啟 Windows 遠端桌面連線用戶端，選擇 Show Options (顯示選項)，並從您在步驟 4 記錄的資訊提供下列項目：

   - 電腦 – 執行個體的公有 DNS 名稱。

     若您希望的話，您也可以使用公有 IP 地址。選擇 Instances (執行個體) 並從執行個體的 Public IP (公有 IP) 資料行複製地址。
   - 使用者名稱 – 使用者名稱。
6. 當用戶端提示您提供登入資料時，輸入您在步驟 4 儲存的密碼。

> **Note**
>
> AWS OpsWorks Stacks 只會產生線上執行個體的使用者密碼。若您啟動執行個體並且假設您其中一個自訂安裝配方失敗，執行個體將會處於 `setup_failed` 狀態。即使執行個體在 Stacks AWS OpsWorks 方面並不在線上，EC2 執行個體正在執行中，而且通常有助於對問題進行故障診斷。在此情況下， AWS OpsWorks Stacks 不會為您產生密碼，但如果您已將 SSH 金鑰對指派給執行個體，您可以使用 EC2 主控台或 CLI 來擷取執行個體的管理員密碼，並以管理員身分登入。如需詳細資訊，請參閱下一節。

以管理員身分登入

您可以透過使用適當的密碼，以管理員身分登入執行個體。如果您已將 EC2 金鑰對指派給執行個體，Amazon EC2 會使用它在執行個體啟動時自動建立和加密管理員密碼。您接著便可搭配 EC2 主控台、API 或 CLI，使用金鑰對的私有金鑰擷取和解密密碼。

> **Note**
>
> 您無法使用個人 SSH 金鑰對擷取管理員密碼。您必須使用 EC2 金鑰對。

以下說明如何使用 EC2 主控台擷取管理員密碼並登入執行個體。若您偏好使用命令列工具，您也可以使用 AWS CLI 的 `get-password-data` 命令擷取密碼。

以管理員身分登入

1. 確認您已為執行個體指定 EC2 金鑰對。您可以在建立堆疊時為所有堆疊的執行個體指定預設金鑰對，或是在您建立執行個體時為特定執行個體指定金鑰對。

2. 開啟 EC2 主控台，將其設為堆疊的區域，然後從導覽窗格選擇 Instances (執行個體)。
3. 選取執行個體，選擇 Connect (連線)，然後選擇 Get Password (取得密碼)。
4. 提供您工作站上指向 EC2 金鑰對私有金鑰的路徑，然後選擇 Decrypt Password (解密密碼)。複製解密後的密碼，將於稍後使用。
5. 開啟 Windows 遠端桌面連線用戶端，選擇 Show Options (顯示選項)，並提供下列資訊：
   - 電腦 – 執行個體的公有 DNS 名稱或公有 IP 地址，您可以從執行個體的詳細資訊頁面取得。
   - 使用者名稱 – `Administrator`。
6. 當用戶端提示您提供登入資料時，提供您在步驟 4 取得的解密密碼。

應用程式

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 應用程式代表您想要在應用程式伺服器上執行的程式碼。程式碼本身位於儲存庫中，例如 Amazon S3 封存；應用程式包含將程式碼部署至適當應用程式伺服器執行個體所需的資訊。

當您部署應用程式時， AWS OpsWorks Stacks 會觸發部署事件，該事件會執行每一層的部署配方。AWS OpsWorks Stacks 也會安裝堆疊組態和部署屬性，其中包含部署應用程式所需的所有資訊，例如應用程式的儲存庫和資料庫連線資料。

您必須實作自訂配方，從堆疊組態和部署屬性擷取應用程式的部署資料，以及處理部署任務。

主題

- 新增應用程式
- 部署應用程式
- 編輯應用程式
- 將應用程式連線至資料庫伺服器
- 使用 環境變數
- 傳遞資料到應用程式

- 使用 Git 儲存庫 SSH 金鑰
- 使用自訂網域
- 使用 SSL

新增應用程式

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

若要將應用程式部署到您的應用程式伺服器，第一個步驟是將應用程式新增至堆疊。app 代表應用程式，並包含各式各樣的中繼資料，例如應用程式的名稱和類型，以及將應用程式部署至伺服器執行個體所需的資訊，例如儲存庫 URL。您必須具有管理許可才能將應用程式新增至堆疊。如需詳細資訊，請參閱管理使用者許可。

> **Note**
>
> 本節中的程序適用於 Chef 12 和更新的堆疊。如需如何在 Chef 11 堆疊中將應用程式新增至 layer 的資訊，請參閱步驟 2.4：建立和部署應用程式 - Chef 11。

將應用程式新增至堆疊

1. 將程式碼放在您偏好的儲存庫中：Amazon S3 封存、Git 儲存庫、Subversion 儲存庫或 HTTP 封存。如需詳細資訊，請參閱應用程式來源。
2. 按一下導覽窗格中的 Apps (應用程式)。針對您的第一個應用程式，在 Apps (應用程式) 頁面中，按一下 Add an app (新增應用程式)。針對後續的應用程式，按一下 +App (+應用程式)。
3. 使用 App New (新的應用程式) 頁面來設定應用程式，如下節中所述。

設定應用程式

Add App (新增應用程式) 頁面包含下列區段：Settings (設定)、Application source (應用程式來源)、Data Sources (資料來源)、Environment Variables (環境變數)、Add Domains (新增網域) 和 SSL Settings (SSL 設定)。

主題

- 設定
- 應用程式來源
- 資料來源
- 環境變數
- 網域和 SSL 設定

設定

名稱

應用程式名稱，用於在 UI 中代表應用程式。 AWS OpsWorks Stacks 也會使用此名稱來產生內部使用的應用程式的簡短名稱，並在堆疊組態和部署屬性中識別應用程式。將應用程式新增至堆疊之後，您可以按一下導覽窗格中的 Apps (應用程式)，然後按一下應用程式名稱開啟詳細資訊頁面，以查看短名。

Document root (文件根)

AWS OpsWorks Stacks 會將文件根設定指派給應用程式`[:document_root]`屬性中的屬性`deploy`。預設值為 `null`。您的部署配方可以使用標準 Chef 節點語法，從 `deploy` 屬性取得該值，並將指定的程式碼部署到伺服器的適當位置。如需如何部署應用程式的詳細資訊，請參閱部署配方。

應用程式來源

您可以從下列儲存庫類型部署應用程式：Git、Amazon S3 套件、HTTP 套件和其他。所有儲存庫類型都要求您必須指定儲存庫類型和儲存庫 URL。個別儲存庫類型都有自己的要求，如下所示。

> **Note**
>
> AWS OpsWorks Stacks 會自動將應用程式從標準儲存庫部署到內建伺服器層。如果您使用其他儲存庫類型，這是 Windows 堆疊的唯一選項， AWS OpsWorks Stacks 會將儲存庫資訊放在應用程式的`deploy`屬性中，但您必須實作自訂配方來處理部署任務。

主題

- HTTP 封存
- Amazon S3 封存

- Git 儲存庫
- 其他儲存庫

HTTP 封存

若要使用可公開存取的 HTTP 伺服器做為儲存庫：

1. 針對包含應用程式程式碼和任何相關聯檔案的資料夾，建立壓縮封存檔 -zip、gzip、bzip2、Java WAR 或 tarball。

   > **Note**
   >
   > AWS OpsWorks Stacks 不支援未壓縮的 tarball。

2. 將封存檔案上傳至伺服器。
3. 若要在主控台中指定儲存庫，請選取 HTTP Archive (HTTP 封存) 做為儲存庫類型並輸入 URL。

   如果封存受到密碼保護，請在應用程式來源下指定登入憑證。

Amazon S3 封存

若要使用 Amazon Simple Storage Service 儲存貯體做為儲存庫：

1. 建立公有或私有 Amazon S3 儲存貯體。如需詳細資訊，請參閱 Amazon S3 文件。
2. 若要讓 AWS OpsWorks Stacks 存取私有儲存貯體，您必須是對 Amazon S3 儲存貯體具有至少唯讀權限的使用者，而且您將需要存取金鑰 ID 和私密存取金鑰。如需詳細資訊，請參閱 AWS Identity and Access Management 文件。
3. 將程式碼和任何相關聯的檔案放入資料夾，並將該資料夾存放到壓縮封存 (zip、gzip、bzip2、Java WAR 或 tarball) 中。

   > **Note**
   >
   > AWS OpsWorks Stacks 不支援未壓縮的 tarball。

4. 將封存檔案上傳至 Amazon S3 儲存貯體並記錄 URL。
5. 若要在 Stacks AWS OpsWorks 主控台中指定儲存庫，請將儲存庫類型設定為 S3 Archive，然後輸入封存的 URL。若是私有封存，您也必須提供 AWS 存取金鑰 ID 和私密存取金鑰，其政策可授予存取儲存貯體的許可。若是公有封存，則將這些設定保留空白。

Git 儲存庫

Git 儲存庫提供來源控制和版本控制。 AWS OpsWorks Stacks 支援公有託管儲存庫網站，例如 GitHub 或 Bitbucket，以及私有託管的 Git 伺服器。若是應用程式和 Git 子模組，您用來在 Application Source (應用程式來源) 中指定儲存庫的 URL 格式則取決於儲存庫為公有或私有：

公有儲存庫 – 使用 HTTPS 或 Git 唯讀通訊協定。例如，Chef 11 Linux 堆疊入門 會使用公有 GitHub 儲存庫，其可透過下列任一 URL 格式來存取：

- Git 唯讀：**`git://github.com/amazonwebservices/opsworks-demo-php-simple-app.git`**
- HTTPS：**`https://github.com/amazonwebservices/opsworks-demo-php-simple-app.git`**

私有儲存庫 – 使用下列範例中顯示的 SSH 讀取/寫入格式：

- Github 儲存庫：**`git@github.com:`*`project`*`/`*`repository`*。
- 位於 Git 伺服器上的儲存庫：*`user`*`@`*`server`*`:`*`project`*`/`*`repository`*

在 Source Control (來源控制) 下方選取 Git 時，會顯示兩個額外的選擇性設定：

Repository SSH key (儲存庫 SSH 金鑰)

您必須指定部署 SSH 金鑰才能存取私有 Git 儲存庫。此欄位需要私有金鑰；公有金鑰會指派到您的 Git 儲存庫。針對 Git 子模組，指定的金鑰必須要能存取這些子模組。如需詳細資訊，請參閱使用 Git 儲存庫 SSH 金鑰。

> **Important**
>
> 部署的 SSH 金鑰不需要密碼； AWS OpsWorks Stacks 無法傳遞密碼。

Branch/Revision (分支/修訂)

如果儲存庫有多個分支， AWS OpsWorks Stacks 預設會下載主分支。若要指定特定分支，請輸入分支名稱、SHA1 雜湊或標籤名稱。若要指定特定的遞交，請輸入完整 40 個八進位碼的遞交識別符。

其他儲存庫

如果標準儲存庫不符合您的需求，您可以使用其他儲存庫，例如 Bazaar。不過， AWS OpsWorks Stacks 不會自動從這類儲存庫部署應用程式。您必須實作自訂配方來處理部署程序，並將這些配方指派給適當 layer 的部署事件。如需如何實作部署配方的範例，請參閱部署配方。

資料來源

此區段會將資料庫連接到應用程式。您有下列選項：

- RDS – 連接其中一個堆疊的 Amazon RDS 服務層。
- 無 – 請勿連接資料庫伺服器。

如果您選取 RDS，則必須指定下列項目。

資料庫執行個體

此清單包含每個 Amazon RDS 服務層。您也可以選取下列之一：

(必要) 指定哪部資料庫伺服器連接到應用程式。清單內容取決於資料來源而定。

- RDS – 堆疊的 Amazon RDS 服務層清單。

資料庫名稱

(選用) 指定資料庫名稱。

- Amazon RDS layer – 輸入您為 Amazon RDS 執行個體指定的資料庫名稱。

  您可以從 Amazon RDS 主控台取得資料庫名稱。

當您使用連接的資料庫部署應用程式時， AWS OpsWorks Stacks 會將資料庫執行個體的連線新增至應用程式的`deploy`屬性。

您可以編寫自訂配方，以從 `deploy` 屬性擷取資訊，並將資訊放入可供應用程式存取的檔案中。這是將資料庫連線資訊提供給其他應用程式類型的唯一選項。

如需如何處理資料庫連線的詳細資訊，請參閱連線至資料庫。

若要將資料庫伺服器與應用程式分離，請編輯應用程式的組態以指定不同的資料庫伺服器，或不指定任何伺服器。

環境變數

您可以為每個應用程式指定一組環境變數，以專屬於該應用程式。例如，如果您有兩個應用程式，則第二個應用程式無法使用您為第一個應用程式定義的環境變數，反之亦然。您也可以為多個應用程式定義相同的環境變數，並為每個應用程式指派不同的值。

> **Note**
>
> 環境變數的數目並沒有特定限制。不過，包含變數名稱、值和受保護標記值的相關聯資料結構大小不得超過 20 KB。此限制應該可以容納大多數的使用案例。超過此限制時會導致服務錯誤 (主控台)，或含下列訊息的例外狀況 (API)："Environment: is too large (maximum is 20KB)" (環境：太大 (上限為 10 KB))。

AWS OpsWorks Stacks 會將變數儲存為應用程式屬性中的`deploy`屬性。您可以使用標準 Chef 節點語法，讓您的自訂配方擷取這些值。如需如何存取應用程式環境變數的範例，請參閱使用 環境變數。

金鑰

變數名稱。它最多可以包含 64 個大小寫字母、數字和底線 (_)，但必須以字母或底線開頭。

Value

變數值。它最多可以包含 256 個字元，且全都必須為可列印字元。

受保護的值

值是否受保護。此設定可讓您隱藏密碼之類的敏感資訊。如果您在建立應用程式之後，為變數設定 Protected value (受保護的值)：

- 應用程式的詳細資訊頁面只會顯示變數名稱，而非值。
- 如果您有編輯應用程式的許可，則可以按一下 Update value (更新值) 來指定新的值，但您無法查看或編輯舊值。

> **Note**
>
> Chef 部署日誌有時可能包含環境變數。這表示受保護的變數可能會顯示在主控台。為了防止受保護的變數顯示在主控台中，我們建議您使用 Amazon S3 儲存貯體做為您不希望在主控台中顯示的受保護變數的儲存體。S3 儲存貯體如何用於此用途，請參閱本指南中的使用 Amazon S3 儲存貯體。

網域和 SSL 設定

對於其他應用程式類型， AWS OpsWorks Stacks 會將設定新增至應用程式的`deploy`屬性。您的配方可以從這些屬性擷取資料，並視需要設定伺服器。

網域設定

此區段有一個選用的 Add Domains (新增網域) 欄位，可用來指定網域。如需詳細資訊，請參閱使用自訂網域。

SSL 設定

此區段有一個 SSL Support (SSL 支援) 切換，可讓您用來啟用或停用 SSL。如果您按一下 Yes (是)，則需要提供 SSL 憑證資訊。如需詳細資訊，請參閱使用 SSL。

部署應用程式

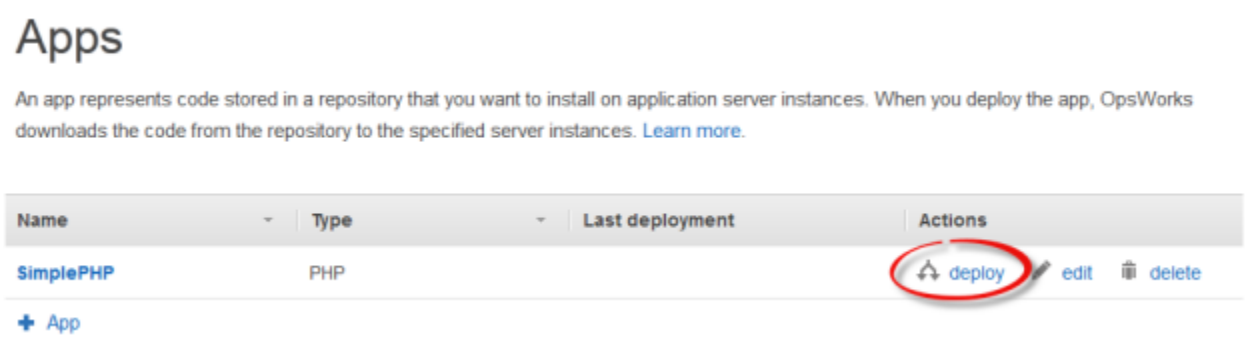
> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

部署的主要目的是將應用程式程式碼和相關檔案部署到應用程式伺服器執行個體。部署操作會由每個執行個體的部署配方處理，該配方則由執行個體的 layer 判斷。

當您啟動執行個體時，在設定配方完成後， AWS OpsWorks Stacks 會自動執行執行個體的部署配方。但是，當您新增或修改應用程式時，您必須手動部署到任何線上的執行個體。您必須具有管理或部署許可才能部署應用程式。如需詳細資訊，請參閱管理使用者許可。

部署應用程式

1. 在 Apps (應用程式) 頁面上，按一下應用程式的 deploy (部署) 動作。



> **Note**
>
> 您也可以透過按一下導覽窗格中的 Deployments (部署) 來部署應用程式。在 Deployments & Commands (部署及命令) 頁面上，按一下 Deploy an app (部署應用程式)。當您執行此作業時，您也可以選擇要部署的應用程式。

2. 指定下列內容：

   - (必要項目) 將 Command: (命令：) 設為 deploy (部署) (若尚未選取的話)。
   - (選用) 包含註解。

3. 按一下進階 >> 指定自訂 JSON. AWS OpsWorks Stacks，將一組堆疊組態和部署屬性新增至節點物件。`deploy` 屬性包含部署詳細資訊，可由部署配方用來處理安裝及組態。在 Linux 堆疊上，您可以使用自訂 JSON 欄位來覆寫預設的 Stacks AWS OpsWorks 設定，或將自訂設定傳遞至您的自訂配方。如需如何使用自訂 JSON 的詳細資訊，請參閱使用自訂 JSON。

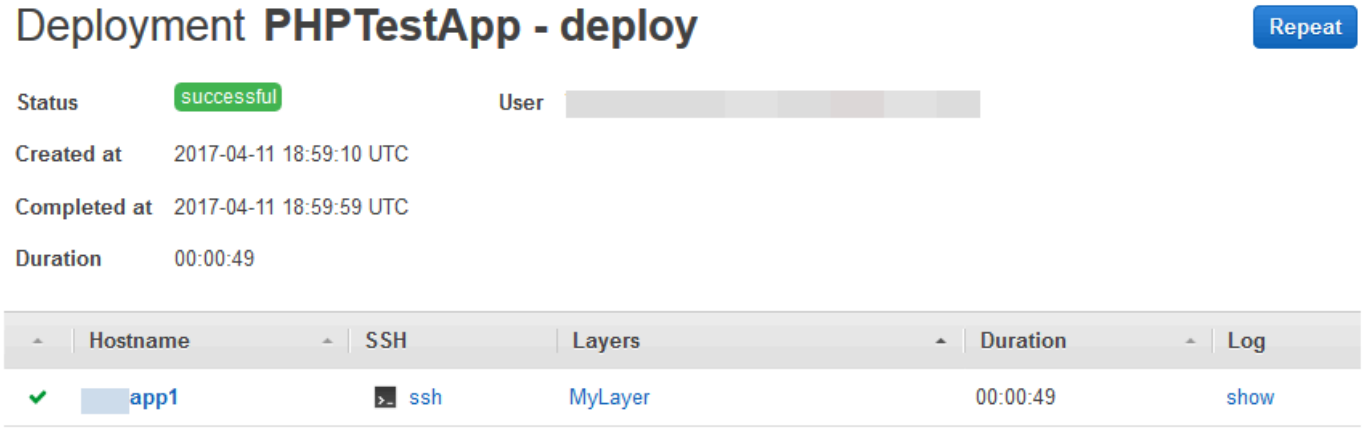
   > **Note**
   >
   > 若您在此指定自訂 JSON，它只會新增至此部署的堆疊組態和部署屬性。若您希望永久新增自訂 JSON，您必須將其新增至堆疊。自訂 JSON 的大小限制為 120 KB。如果您需要更多容量，建議您將部分資料儲存在 Amazon S3。您的自訂配方接著可以使用 AWS CLI 或適用於 Ruby 的 AWS 開發套件將資料從儲存貯體下載到您的執行個體。如需範例，請參閱「使用適用於 Ruby 的 SDK」。

4. 在 Instances (執行個體) 下方，按一下 Advanced >> (進階 >>) 並指定要在哪個執行個體上執行部署命令。

   部署命令會觸發部署事件，在選取的執行個體上執行部署配方。關聯應用程式伺服器的部署配方會從儲存庫下載程式碼和相關檔案，並將他們安裝在執行個體上，因此您通常會選取所有關聯的應用程式伺服器執行個體。但是，其他執行個體類型可能需要一些組態變更，才能適應新的應用程式，

因此通常在那些執行個體上也執行部署配方會非常有用。那些配方會視需要更新組態，但不會安裝應用程式的檔案。如需配方的詳細資訊，請參閱技術指南和配方。

5. 按一下 Deploy (部署) 以在指定的執行個體上執行部署配方，顯示部署頁面。程序完成時， AWS OpsWorks Stacks 會以綠色勾號標記應用程式，表示部署成功。如果部署失敗， AWS OpsWorks Stacks 會以紅色 X 標記應用程式。在這種情況下，您可以前往部署頁面並檢查部署日誌以取得詳細資訊。



> **Note**
>
> 在您將更新部署至 JSP 應用程式時，Tomcat 可能會無法識別更新，而繼續執行現有的應用程式版本。舉例來說，這可能會在您將應用程式做為僅包含 JSP 頁面的 .zip 檔案部署時發生。若要確保 Tomcat 執行的是最近部署的版本，專案的根目錄應包括內含 `web.xml` 檔案的 WEB-INF 目錄。`web.xml` 檔案可包含各種內容，但以下內容便足以確保 Tomcat 識別更新及執行目前部署的應用程式版本。您不需要為每次更新變更版本。若版本沒有變更，Tomcat 將會識別更新。
>
> ```
> <context-param>
>   <param-name>appVersion</param-name>
>   <param-value>0.1</param-value>
> </context-param>
> ```

其他部署命令

Deploy app (部署應用程式) 頁面包含數種其他用於管理您應用程式和關聯伺服器的命令。在下列命令中，只有 Undeploy (解除部署) 適用於位於 Chef 12 堆疊上的應用程式。

Undeploy (解除部署)

觸發解除部署生命週期事件，執行解除部署配方，從指定執行個體移除所有版本的應用程式。

轉返

還原先前部署的應用程式版本。例如，若您已部署應用程式三次，然後執行 Rollback (轉返)，伺服器會從第二次部署提供應用程式。若您再次執行 Rollback (轉返)，伺服器會從第一次部署提供應用程式。根據預設， AWS OpsWorks Stacks 會存放五個最近的部署，可讓您復原最多四個版本。若您超過存放版本的數目，命令會失敗，並留下最舊的版本。此命令無法在 Chef 12 堆疊中使用。

Start Web Server (啟動 Web 伺服器)

執行會在指定之執行個體上啟動應用程式伺服器的配方。此命令無法在 Chef 12 堆疊中使用。

Stop Web Server (停止 Web 伺服器)

執行會在指定之執行個體上停止應用程式伺服器的配方。此命令無法在 Chef 12 堆疊中使用。

Restart Web Server (重新啟動 Web 伺服器)

執行會在指定之執行個體上重新啟動應用程式伺服器的配方。此命令無法在 Chef 12 堆疊中使用。

Important

Start Web Server (啟動 Web 伺服器)、Stop Web Server (停止 Web 伺服器)、Restart Web Server (重新啟動 Web 伺服器) 和 Rollback (轉返) 基本上就是執行配方堆疊命令的自訂版本。他們會執行一組在指定之執行個體上執行任務的配方。

- 這些命令不會觸發生命週期事件，因此您無法引用命令執行自訂程式碼。
- 這些命令僅適用於內建的應用程式伺服器 layer。

  特別是，這些命令對於自訂 layer 沒有任何效果，即使他們支援應用程式伺服器也一樣。若要在自訂 layer 上啟動、停止或重新啟動伺服器，您必須實作自訂配方以執行這些任務，並使用執行配方堆疊命令執行他們。如需如何實作和安裝自訂配方的詳細資訊，請參閱技術指南和配方。

編輯應用程式

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以透過編輯應用程式來修改應用程式的組態。例如，如果您已準備好部署新版本，您可以編輯應用程式的 Stacks AWS OpsWorks 設定，以使用新的儲存庫分支。您必須具有管理或部署許可才能編輯應用程式組態。如需詳細資訊，請參閱管理使用者許可。

編輯應用程式

1. 在 Apps (應用程式) 頁面按一下應用程式名稱，開啟它的 details (詳細資訊) 頁面。
2. 按一下 Edit (編輯) 變更應用程式的組態。
   - 如果您修改應用程式的名稱， AWS OpsWorks Stacks 會使用新名稱在主控台中識別應用程式。

     變更名稱不會變更相關聯的簡稱。簡稱是在您將應用程式新增到堆疊後設定，以後無法修改。
   - 如已指定受保護的環境變數，您即無法查看或編輯值。不過，您可以按一下 Update value (更新值) 指定新的值。
3. 按一下 Save (儲存) 儲存新的組態，然後按一下 Deploy App (部署應用程式) 部署應用程式。

編輯應用程式會使用 Stacks AWS OpsWorks 變更設定，但不會影響堆疊的執行個體。當您第一次部署應用程式時，Deploy 配方會將程式碼及相關檔案下載到應用程式伺服器執行個體，然後這些執行個體會執行本機複本。如果您在儲存庫中修改應用程式，或變更任何其他設定，則必須部署應用程式，才能在應用程式伺服器執行個體上安裝更新，如下所示。當應用程式版本啟動時， AWS OpsWorks Stacks 會自動將目前的應用程式版本部署到新的執行個體。不過，現有執行個體的情況則不同：

- AWS OpsWorks Stacks 會在啟動時，自動將目前的應用程式版本部署到新的執行個體。
- AWS OpsWorks Stacks 會在重新啟動時，自動將最新的應用程式版本部署至離線執行個體，包括負載型和時間型執行個體。
- 您必須將更新的應用程式以手動方式部署到線上執行個體。

如需如何部署應用程式的詳細資訊，請參閱部署應用程式。

將應用程式連線至資料庫伺服器

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

當您建立應用程式時，您可以透過編輯應用程式，將 Amazon RDS 資料庫伺服器與應用程式建立關聯。然後，您的應用程式可以使用資料庫連線資訊 - 使用者名稱、密碼 ... - 來連線至資料庫伺服器。當您部署應用程式時， AWS OpsWorks Stacks 會以兩種方式將此資訊提供給應用程式：

- 對於 Linux 堆疊， AWS OpsWorks Stacks 會在每個內建應用程式伺服器執行個體上建立檔案，而此檔案包含應用程式可用來連線至資料庫伺服器的連線資料。
- AWS OpsWorks Stacks 包含堆疊組態中的連線資訊，以及安裝在每個執行個體上的部署屬性。

  您可以實作自訂配方以從這些屬性中擷取連線資訊，並將它放入您慣用格式的檔案中。如需詳細資訊，請參閱傳遞資料到應用程式。

> **Important**
>
> 對於 Linux 堆疊，如果您想要將 Amazon RDS 服務層與應用程式建立關聯，則必須將適當的驅動程式套件新增至關聯的應用程式伺服器層，如下所示：
>
> 1. 按一下導覽窗格中的 Layers (Layer)，然後開啟應用程式伺服器的 Recipes (配方) 標籤。
> 2. 按一下 Edit (編輯)，然後將適當的驅動程式套件新增至 OS Packages (OS 套件)。例如，如果 layer 包含 Amazon Linux 執行個體，則您應該指定 `mysql`；如果 layer 包含 Ubuntu 執行個體，則應該指定 `mysql-client`。
> 3. 儲存變更，並重新部署應用程式。

使用自訂配方

您可以實作自訂配方以從應用程式的 `deploy` 屬性中擷取連線資料，並將它儲存為應用程式可讀取的形式 (例如 YAML 檔案)。

當您建立應用程式或稍後編輯應用程式時，可以將資料庫伺服器連接至應用程式。當您部署應用程式時， AWS OpsWorks Stacks 會在包含資料庫連線資訊的每個執行個體上安裝堆疊組態和部署屬性。您的應用程式接著可以擷取適當的屬性。詳細資訊取決於您使用 Linux 還是 Windows 堆疊。

連線至 Linux 堆疊的資料庫伺服器

對於 Linux 堆疊，堆疊組態和部署屬性的 `deploy` 命名空間包括每個已部署應用程式的屬性 (命名為應用程式的簡短名稱)。當您將資料庫伺服器連接到應用程式時， AWS OpsWorks Stacks 會將連線資訊填入應用程式的`[:database]`屬性，並針對每個後續部署將其安裝在堆疊的執行個體上。屬性值由使用者所提供或由 AWS OpsWorks Stacks 所產生。

> **Note**
>
> AWS OpsWorks Stacks 可讓您將資料庫伺服器連接到多個應用程式，但每個應用程式只能有一個連接的資料庫伺服器。如果您想要將應用程式連線至多部資料庫伺服器，則請將其中一部伺服器連接至應用程式，並在應用程式的 `deploy` 屬性中使用此資訊來連線至該伺服器。使用自訂 JSON 將其他資料庫伺服器的連線資訊傳遞給應用程式。如需詳細資訊，請參閱傳遞資料到應用程式。

應用程式可以使用執行個體之 `deploy` 屬性中的連線資訊來連線至資料庫。不過，應用程式無法直接存取該資訊，只有配方可以存取`deploy`屬性。您可以實作自訂配方來解決此問題，而自訂配方會擷取 `deploy` 屬性中的連線資訊，並將它放入應用程式可讀取的檔案中。

使用 環境變數

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 本主題中的建議適用於 Chef 11.10 及更早版本的 Chef。若要在 Chef 12 及更新版本中取得環境變數，您必須使用應用程式資料包。如需詳細資訊，請參閱 AWS OpsWorks 資料包參考及應用程式資料包 (aws_opsworks_app)。

當您為應用程式指定環境變數時， AWS OpsWorks Stacks 會將變數定義新增至應用程式的deploy屬性。

自訂 layer 可透過使用標準節點語法，利用配方擷取變數的值，並以可由 layer 的應用程式存取的形式存放。

您必須實作從執行個體的 deploy 屬性取得環境變數的自訂配方。配方接著會將資料以可由應用程式存取的形式存放在執行個體上 (例如 YAML 檔案)。應用程式的環境變數定義會存放在應用程式 deploy 中的 environment_variables 屬性。以下範例顯示名為 simplephpapp 之應用程式內這些屬性的位置，並使用 JSON 代表屬性的結構。

```
{
  ...
  "ssh_users": {
  },
  "deploy": {
    "simplephpapp": {
      "application": "simplephpapp",
      "application_type": "php",
      "environment_variables": {
        "USER_ID": "168424",
        "USER_KEY": "somepassword"
      },
    ...
  }
}
```

配方可透過使用標準節點語法，取得變數的值。以下範例示範如何從先前的 JSON 中取得 USER_ID 的值，並將其置放於 Chef 日誌中。

```
Chef::Log.info("USER_ID: #{node[:deploy]['simplephpapp'][:environment_variables]
[:USER_ID]}")
```

如需如何從堆疊組態及部署 JSON 擷取資訊，並將其存放在執行個體上的更詳細說明，請參閱傳遞資料到應用程式。

傳遞資料到應用程式

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

通常將資料 (例如鍵/值對) 傳遞給伺服器上的應用程式很有用。若要執行此作業，請使用自訂 JSON 將資料新增至堆疊。 AWS OpsWorks Stacks 會將資料新增至每個生命週期事件的每個執行個體節點物件。

但是，請注意，雖然配方可透過 Chef 屬性從節點物件取得自訂 JSON 資料，應用程式則無法。其中一個將自訂 JSON 資料傳遞給一或多個應用程式的方式，便是實作自訂配方，從 node 物件擷取資料，然後將資料寫入應用程式可讀取的檔案中。本主題中的範例示範如何將資料寫入 YAML 檔案，但您可以針對其他格式 (例如 JSON 或 XML) 使用相同的基本方法。

若要將鍵/值資料傳遞給堆疊的執行個體，請新增如下的自訂 JSON 至堆疊。如需如何將自訂 JSON 新增至堆疊的詳細資訊，請參閱使用自訂 JSON。

```
{
  "my_app_data": {
    "app1": {
      "key1": "value1",
      "key2": "value2",
      "key3": "value3"
    },
    "app2": {
      "key1": "value1",
      "key2": "value2",
      "key3": "value3"
    }
  }
}
```

範例假設您有兩個應用程式，名稱分別為 app1 和 app2，並且每一個皆有三個資料值。隨附的配方假設您使用應用程式的短名識別關聯資料。其他名稱則為任意。如需應用程式短名的詳細資訊，請參閱設定。

下列範例中的配方示範如何為每個應用程式，從 deploy 屬性擷取資料，然後將其置放在 .yml 檔案中。配方假設您的自訂 JSON 包含每個應用程式的資料。

```
node[:deploy].each do |app, deploy|
  file File.join(deploy[:deploy_to], 'shared', 'config', 'app_data.yml') do
    content YAML.dump(node[:my_app_data][app].to_hash)
  end
end
```

deploy 屬性包含每個應用程式的屬性，以應用程式的短名命名。每個應用程式屬性皆包含一組屬性，代表應用程式的各種資訊。此範例使用應用程式的部署目錄，以 [:deploy][:*app_short_name*][:deploy_to] 屬性代表。如需 [:deploy] 的詳細資訊，請參閱deploy 屬性。

針對每個 deploy 中的應用程式，配方會執行下列作業：

1. 在應用程式的 app_data.yml 目錄中的 shared/config 子目錄內，建立名為 [:deploy_to] 的檔案。

   如需 Stacks AWS OpsWorks 如何安裝應用程式的詳細資訊，請參閱 部署配方。
2. 將應用程式的自訂 JSON 值轉換成 YAML，並將格式化後的資料寫入 app_data.yml。

將資料傳遞到應用程式

1. 將應用程式新增至堆疊，然後記下其短名。如需詳細資訊，請參閱新增應用程式。
2. 使用應用程式的資料，將自訂 JSON 新增至 deploy 屬性，如先前所述。如需如何將自訂 JSON 新增至堆疊的詳細資訊，請參閱使用自訂 JSON。
3. 建立技術指南，並將程式碼以先前範例做為基礎的配方新增至其中，針對您在自訂 JSON 中使用的屬性名稱，根據需求進行修改。如需如何建立技術指南和配方的詳細資訊，請參閱技術指南和配方。若您已有此堆疊的自訂技術指南，您也可以將配方新增至現有的技術指南，或是將程式碼新增至現有的部署配方。
4. 在您的堆疊上安裝技術指南。如需詳細資訊，請參閱安裝自訂技術指南。
5. 將配方指派給應用程式伺服器層的部署生命週期事件。 AWS OpsWorks 堆疊接著會在每個新執行個體啟動後執行配方。如需詳細資訊，請參閱執行配方。
6. 部署應用程式，同時也會安裝包含您資料的堆疊組態和部署屬性。

> **Note**
>
> 若資料檔案必須在應用程式部署前便存在，您也可以將配方指派給 layer 的安裝生命週期事件。該事件只會在執行個體完成開機後執行一次。不過， AWS OpsWorks Stacks 尚未建立部署目錄，因此您的配方應該在建立資料檔案之前明確建立所需的目錄。以下範例明確建立應用程式的 `/shared/config` 目錄，然後在該目錄中建立資料檔案。
>
> ```
> node[:deploy].each do |app, deploy|
>
>  directory "#{deploy[:deploy_to]}/shared/config" do
>       owner "deploy"
>       group "www-data"
>       mode 0774
>       recursive true
>       action :create
>     end
>
>   file File.join(deploy[:deploy_to], 'shared', 'config', 'app_data.yml') do
>     content YAML.dump(node[:my_app_data][app].to_hash)
>   end
> end
> ```

若要載入資料，您可以使用看起來像是下列 Sinatra 程式碼的內容：

```
#!/usr/bin/env ruby
# encoding: UTF-8
require 'sinatra'
require 'yaml'

get '/' do
  YAML.load(File.read(File.join('..', '..', 'shared', 'config', 'app_data.yml')))
End
```

您可以透過更新自訂 JSON 來隨時更新應用程式的資料值，如下所示。

更新應用程式資料

1. 編輯自訂 JSON 以更新資料值。

2. 再次部署應用程式，這會指示 AWS OpsWorks Stacks 在堆疊的執行個體上執行部署配方。配方會使用已更新堆疊組態及部署屬性的屬性，因此您的自訂配方會使用目前的值更新資料檔案。

使用 Git 儲存庫 SSH 金鑰

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Git 儲存庫 SSH 金鑰，有時候稱為部署 SSH 金鑰，為不帶密碼的 SSH 金鑰，可提供私有 Git 儲存庫的存取。此金鑰理應不屬於任何特定的開發人員。其目的是允許 AWS OpsWorks Stacks 從 Git 儲存庫非同步部署應用程式或技術指南，而不需要您進一步輸入。

下列說明建立儲存庫 SSH 金鑰的基本程序。如需詳細資訊，請參閱您儲存庫的文件。例如，管理部署金鑰說明如何建立 GitHub 儲存庫的儲存庫 SSH 金鑰，Bitbucket 上的部署金鑰則說明如何建立 Bitbucket 儲存庫的儲存庫 SSH 金鑰。請注意，某些文件會說明在伺服器上建立金鑰。對於 AWS OpsWorks Stacks，只需在說明中將 "server" 取代為 "workstation"。

建立儲存庫 SSH 金鑰

1. 在您的工作站上使用像是 `ssh-keygen` 等程式，建立您 Git 儲存庫的部署 SSH 金鑰對。

   > **Important**
   >
   > AWS OpsWorks Stacks 不支援 SSH 金鑰密碼片語。

2. 將公有金鑰指派給儲存庫，然後在您的工作站上存放私有金鑰。
3. 在您新增應用程式或指定技術指南儲存庫時，在 Repository SSH Key (儲存庫 SSH 金鑰) 方塊中輸入私有金鑰。如需詳細資訊，請參閱新增應用程式。

AWS OpsWorks Stacks 會將儲存庫 SSH 金鑰傳遞給每個執行個體，而內建配方接著會使用 金鑰連線至儲存庫並下載程式碼。金鑰會做為 `deploy` 存放於 `node[:deploy]['appshortname'][:scm][:ssh_key]` 屬性中，且僅能由根使用者存取。

使用自訂網域

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

如果您透過第三方託管網域名稱，您可將該網域名稱映射至應用程式。基本程序如下：

1. 透過 DNS 註冊商建立子網域，然後將其映射至負載平衡器的彈性 IP 地址，或應用程式伺服器的公有 IP 地址。
2. 更新應用程式的組態以指向子網域並重新部署應用程式。

> **Note**
>
> 請確定您將不合格網域名稱 (例如 myapp1.example.com) 轉送至合格網域名稱 (例如 www.myapp1.example.com)，以便兩者都能映射至您的應用程式。

當您為應用程式設定網域時，網域在伺服器的組態檔案中會做為伺服器別名列出。如果您使用負載平衡器，則負載平衡器會在請求傳入時檢查 URL 中的網域名稱，並根據網域來重新導向流量。

將子網域映射至 IP 地址

1. 如果您使用負載平衡器，請在 Instances (執行個體) 頁面上按一下負載平衡器執行個體，以開啟其詳細資訊頁面，並取得該執行個體的 Elastic IP (彈性 IP) 地址。否則請從應用程式伺服器執行個體的詳細資訊頁面取得公有 IP 地址。
2. 遵循 DNS 註冊商提供的指示來建立子網域，並將其映射至步驟 1 中的 IP 地址。

> **Note**
>
> 如果負載平衡器執行個體在某個時間點終止，會為您指派新的彈性 IP 地址。您需要更新 DNS 註冊設定，才能映射到新的彈性 IP 地址。

AWS OpsWorks Stacks 只會將網域設定新增至應用程式的deploy屬性。您必須實作自訂配方，才可從節點物件擷取資訊，並適當設定伺服器。如需詳細資訊，請參閱技術指南和配方。

在同一個應用程式伺服器上執行多個應用程式

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 本主題中的資訊不適用於 Node.js 應用程式。

如果您有多個相同類型的應用程式，在同一個應用程式伺服器執行個體上執行有時更符合成本效益。

在同一個伺服器上執行多個應用程式

1. 將應用程式新增至每個應用程式的堆疊。
2. 為每個應用程式取得個別的子網域，並將子網域映射至應用程式伺服器或負載平衡器的 IP 地址。
3. 編輯每個應用程式的組態以指定適當的子網域。

如需如何執行這些任務的詳細資訊，請參閱使用自訂網域。

> **Note**
>
> 如果您的應用程式伺服器執行多個 HTTP 應用程式，您可以使用 Elastic Load Balancing 進行負載平衡。對於多個 HTTPS 應用程式，您必須在負載平衡器終止 SSL 連線，或為每個應用程式建立個別的堆疊。HTTPS 請求經過加密，這表示如果您在伺服器終止 SSL 連線，負載平衡器就無法檢查網域名稱，判斷哪個應用程式應處理請求。

使用 SSL

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

若要搭配使用 SSL 與您的應用程式，您必須先從憑證授權機構 (CA) 取得數位伺服器憑證。為求簡化，本演練會建立憑證，然後對它自我簽署。自簽憑證適用於學習和測試，但您應該一律使用由 CA 針對生產堆疊所簽署的憑證。

在本演練中，您將會執行下列作業：

1. 安裝及設定 OpenSSL。
2. 建立私有金鑰。
3. 建立憑證簽署請求。
4. 產生自簽憑證。
5. 使用您的憑證資訊編輯應用程式。

> **Important**
>
> 如果您的應用程式使用 SSL，則建議您盡可能在應用程式伺服器 layer 中停用 SSLv3 來處理 CVE-2014-3566 中所述的漏洞。如果您的堆疊包含 Ganglia 層，您也應該停用該層的 SSL v3。詳細資訊取決於特定 layer；如需詳細資訊，請參閱下列項目。
>
> - Java 應用程式伺服器 AWS OpsWorks 堆疊層
> - Node.js 應用程式伺服器 AWS OpsWorks 堆疊層
> - PHP 應用程式伺服器 AWS OpsWorks 堆疊層
> - Rails 應用程式伺服器 AWS OpsWorks 堆疊層
> - 靜態 Web 伺服器 AWS OpsWorks 堆疊層
> - Ganglia 層

主題

- 步驟 1：安裝和設定 OpenSSL
- 步驟 2：建立私有金鑰
- 步驟 3：建立憑證簽署請求
- 步驟 4：將 CSR 提交至憑證授權機構
- 步驟 5：編輯應用程式

步驟 1：安裝和設定 OpenSSL

建立和上傳伺服器憑證需要支援 SSL 和 TLS 通訊協定的工具。OpenSSL 是一種開放原始碼工具，可提供建立 RSA 字符並使用私有金鑰簽署它所需的基本加密函數。

下列程序假設您的電腦尚未安裝 OpenSSL。

在 Linux 和 Unix 上安裝 OpenSSL

1. 前往 OpenSSL: Source, Tarballs。
2. 下載最新原始碼。
3. 建置套件。

在 Windows 安裝 OpenSSL

1. 如果您的系統上尚未安裝 Microsoft Visual C++ 2008 可轉散發套件，請下載套件。
2. 執行安裝程式，並遵循 Microsoft Visual C++ 2008 Redistributable Setup Wizard (Microsoft Visual C++ 2008 可轉散發套件設定精靈) 所提供的說明來安裝可轉散發套件。
3. 前往 OpenSSL: Binary Distributions，並按一下您環境適用的 OpenSSL 二進位檔版本，並在本機儲存安裝程式。
4. 執行安裝程式，並遵循 OpenSSL Setup Wizard (OpenSSL 設定精靈) 中的說明來安裝二進位檔。

開啟終端機或命令視窗，並使用下列命令列，以建立指向 OpenSSL 安裝點的環境變數。

- 在 Linux 和 Unix 上

```
export OpenSSL_HOME=path_to_your_OpenSSL_installation
```

- 在 Windows 上

```
set OpenSSL_HOME=path_to_your_OpenSSL_installation
```

開啟終端機或命令視窗，並使用下列命令列，以將 OpenSSL 二進位檔的路徑新增至您電腦的路徑變數。

- 在 Linux 和 Unix 上

```
export PATH=$PATH:$OpenSSL_HOME/bin
```

- 在 Windows 上

```
set Path=OpenSSL_HOME\bin;%Path%
```

> **Note**
>
> 您使用這些命令列對環境變數做出的任何變更，僅適用於目前命令列工作階段。

步驟 2：建立私有金鑰

您需要有唯一私有金鑰才能建立憑證簽署請求 (CSR)。使用下列命令列來建立金鑰：

```
openssl genrsa 2048 > privatekey.pem
```

步驟 3：建立憑證簽署請求

憑證簽署請求 (CSR) 是一種檔案，可傳送至憑證授權機構 (CA) 以申請數位伺服器憑證。使用下列命令列來建立 CSR。

```
openssl req -new -key privatekey.pem -out csr.pem
```

命令的輸出看起來與下列類似：

```
You are about to be asked to enter information that will be incorporated
 into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
```

下表可協助您建立憑證請求。

憑證請求資料

名稱	描述	範例
Country Name (國家/地區名稱)	兩個字母的 ISO 縮寫，用來代表您的國家/地區。	US = 美國
State or Province (州或省)	您組織位在的州名或省名。此名稱不得使用縮寫。	華盛頓州
Locality Name (地區名稱)	您組織所在城市的名稱。	西雅圖
Organization Name (組織名稱)	您組織的完整法定名稱。請不要使用您組織名稱的縮寫。	CorporationX
Organizational Unit (組織單位)	(選用) 這是額外的組織資訊。	行銷部門
Common Name (通用名稱)	您 CNAME 的完整網域名稱。如果這不是完全相符的項目，則您會收到憑證名稱檢查警告。	www.example.com
電子郵件地址	伺服器管理員的電子郵件地址。	someone@example.com

> **Note**
>
> Common Name (通用名稱) 欄位容易遭人誤解，因此完全不正確。通用名稱一般是您的主機加上網域名稱。它看起來像是 "www.example.com" 或 "example.com"。您需要使用正確的通用名稱來建立 CSR。

步驟 4：將 CSR 提交至憑證授權機構

針對生產用途，您可以將 CSR 提交至憑證授權機構 (CA) 來取得伺服器憑證，這可能需要其他登入資料或身分證明。如果您的應用程式成功，則 CA 會傳回數位簽署的身分憑證，因此可能是憑證鏈檔案。AWS 不建議使用特定 CA。如需可用 CA 的部分清單，請參閱 Wikipedia 上的 Certificate Authority - Providers。

您也可以產生自簽憑證，以用於測試目的。在此範例中，使用下列命令列產生自簽憑證。

```
openssl x509 -req -days 365 -in csr.pem -signkey privatekey.pem -out server.crt
```

輸出格式應類似以下內容：

```
Loading 'screen' into random state - done
Signature ok
subject=/C=us/ST=washington/L=seattle/O=corporationx/OU=marketing/CN=example.com/
emailAddress=someone@example.com
Getting Private key
```

步驟 5：編輯應用程式

在您產生並簽署憑證之後，請更新應用程式以啟用 SSL，並提供憑證資訊。在 Apps (應用程式) 頁面上，選擇應用程式以開啟詳細資訊頁面，然後按一下 Edit App (編輯應用程式)。若要啟用 SSL 支援，請將 Enable SSL (啟用 SSL) 設為 Yes (是)，以顯示下列組態選項。

SSL 憑證

將公有金鑰憑證 (.crt) 檔案的內容貼入方塊中。憑證看起來應該與下列類似：

```
-----BEGIN CERTIFICATE-----
MIICuTCCAiICCQCtqFKItVQJpzANBgkqhkiG9w0BAQUFADCBoDELMAkGA1UEBhMC
dXMxEzARBgNVBAgMCndhc2hpbmd0b24xEDAOBgNVBAcMB3NlYXR0bGUxDzANBgNV
BAoMBmFtYXpvbjEWMBQGA1UECwwNRGV2IGFuZCBUb29sczEdMBsGA1UEAwwUc3Rl
cGhhbmllYXBpZXJjZS5jb20xIjAgBgkqhkiG9w0BCQEWE3NhcGllcmNlQGFtYXpv
...
-----END CERTIFICATE-----
```

> **Note**
>
> 如果您使用 Nginx 並且具有憑證鏈檔案，則應該將內容附加到公有金鑰憑證檔案。

如果您更新現有憑證，則請執行下列動作：

- 選擇 Update SSL certificate (更新 SSL 憑證) 來更新憑證。
- 如果新憑證不符合現有私有金鑰，請選擇 Update SSL certificate key (更新 SSL 憑證金鑰)。
- 如果新憑證不符合現有憑證鏈，請選擇 Update SSL certificates (更新 SSL 憑證)。

SSL Certificate Key (SSL 憑證金鑰)

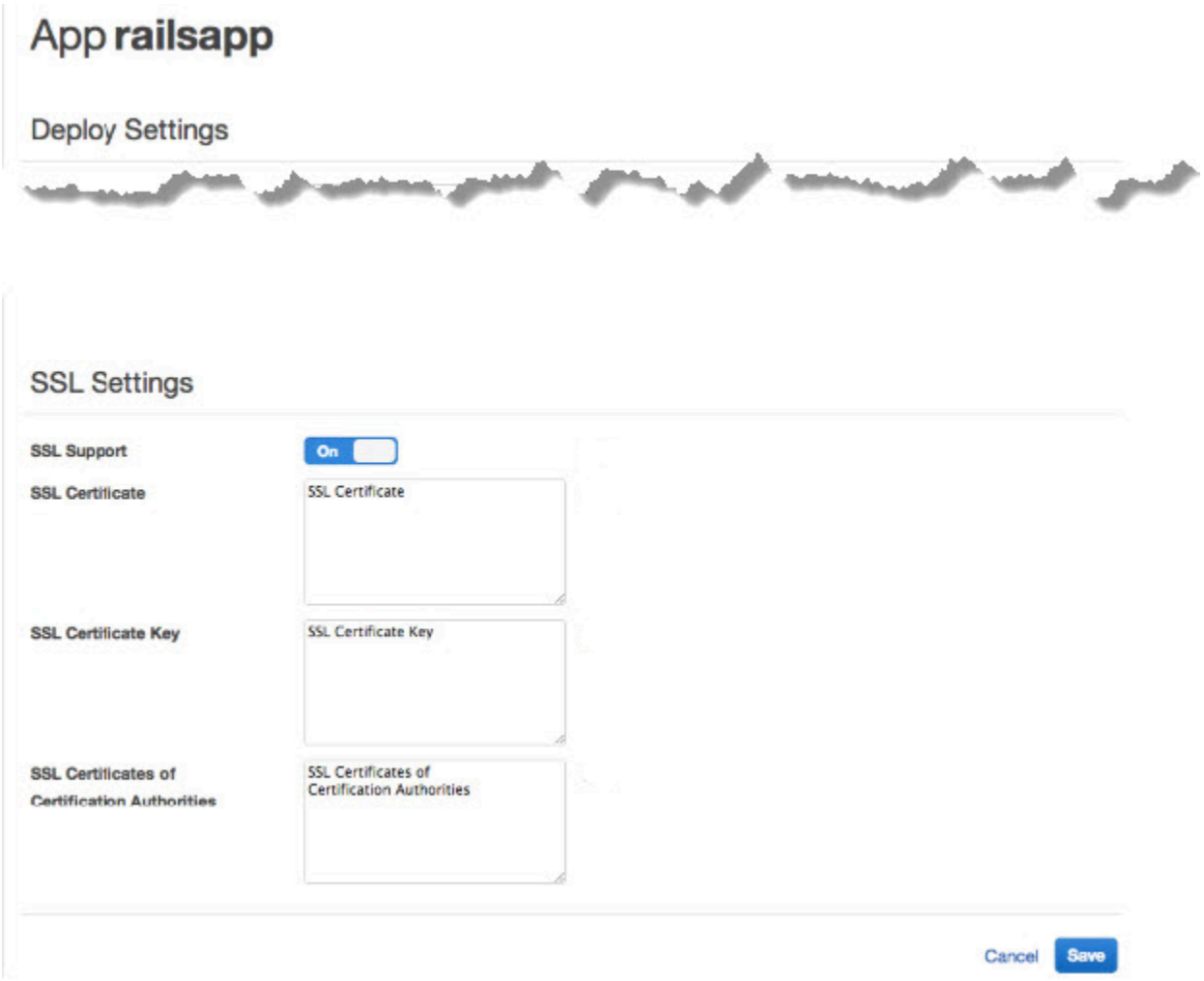
將私有金鑰檔案 (.pem 檔案) 的內容貼入方塊中。它看起來應該與下列類似：

```
----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQC0CYklJY5r4vV2NHQYEpwtsLuMMBhylMrgBShKq+HHVLYQQCL6
+wGIiRq5qXqZlRXje3GM5Jvcm6q0R71MfRIl1FuzKyqDtneZaAIEYniZibHiUnmO
/UNqpFDosw/6hY3ONk0fSBlU4ivD0Gjpf6J80jL3DJ4R23Ed0sdL4pRT3QIDAQAB
AoGBAKmMfWrNRqYVtGKgnWB6Tji9QrKQLMXjmHeGg95mppdJELiXHhpMvrHtpIyK
...
-----END RSA PRIVATE KEY-----
```

SSL certificates of Certification Authorities (憑證授權機構的 SSL 憑證)

如果您有憑證鏈檔案，則請將內容複製至方塊中。

> **Note**
>
> 如果您使用 Nginx，則應該將此方塊空白。如果您有憑證鏈檔案，請在 SSL Certificate (SSL 憑證) 中將它附加到公開金鑰憑證檔案。



在您按一下 Save (儲存) 之後，請重新部署應用程式以更新線上執行個體。

對於內建應用程式伺服器層， AWS OpsWorks Stacks 會自動更新伺服器組態。在部署完成之後，您可以驗證 OpenSSL 安裝已運作，如下所示。

驗證 OpenSSL 安裝

1. 前往 Instances (執行個體) 頁面。
2. 按一下應用程式伺服器執行個體的 IP 地址 (或者，如果您使用負載平衡器，則為負載平衡器的 IP 地址) 來執行應用程式。
3. 將 IP 地址字首從 **`http://`** 變更為 **`https://`**，並重新整理瀏覽器來使用 SSL 驗證頁面正確載入。

已設定要在 Mozilla Firefox 中執行之應用程式的使用者，有時會得到下列憑證錯誤：SEC_ERROR_UNKNOWN_ISSUER。造成此錯誤的原因可能是您組織的防毒和反惡意軟體中的憑證

更換功能、某些類型的網路流量監控和篩選軟體，或惡意軟體。如需如何排除此錯誤的詳細資訊，請參閱 Mozilla Firefox 支援網站上的如何排除安全網站上的安全錯誤碼。

針對所有其他 layer (包括自訂 layer)， AWS OpsWorks Stacks 只會將 SSL 設定新增至應用程式的 deploy 屬性。您必須實作自訂配方，才可從節點物件擷取資訊，並適當設定伺服器。

技術指南和配方

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 使用 Chef 技術指南來處理安裝和設定套件和部署應用程式等任務。本節說明如何搭配 Stacks AWS OpsWorks 使用技術指南。如需詳細資訊，請參閱 Chef。

> **Note**
>
> AWS OpsWorks Stacks 目前支援 Chef 版本 12、11.10.4、11.4.4 和 0.9.15.5。不過，Chef 0.9.15.5 即將移除，我們不建議您將它用於新的堆疊。為了方便起見，通常只以主要和次要版本號碼稱呼它們。執行 Chef 0.9 或 11.4 的堆疊使用 Chef Solo，執行 Chef 12 或 11.10 的堆疊則在本機模式中使用 Chef Client。針對 Linux 堆疊，您可以在您建立堆疊時，使用組態管理員指定要使用的 Chef 版本。Windows 堆疊必須使用 Chef 12.2。如需詳細資訊 (包含將堆疊遷移至更近 Chef 版本的準則)，請參閱 Chef 版本。

主題

- 技術指南儲存庫
- Chef 版本
- Ruby 版本
- 安裝自訂技術指南
- 更新自訂技術指南
- 執行配方

技術指南儲存庫

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您的自訂技術指南必須存放在線上儲存庫，如 .zip 檔案的封存或如 Git 的來源控制管理員。堆疊只能有一個自訂技術指南儲存庫，但儲存庫可以包含任意數目的技術指南。當您安裝或更新技術指南時，AWS OpsWorks Stacks 會在每個堆疊執行個體的本機快取中安裝整個儲存庫。例如，當執行個體需要執行一或多個配方時，會使用來自本機快取的程式碼。

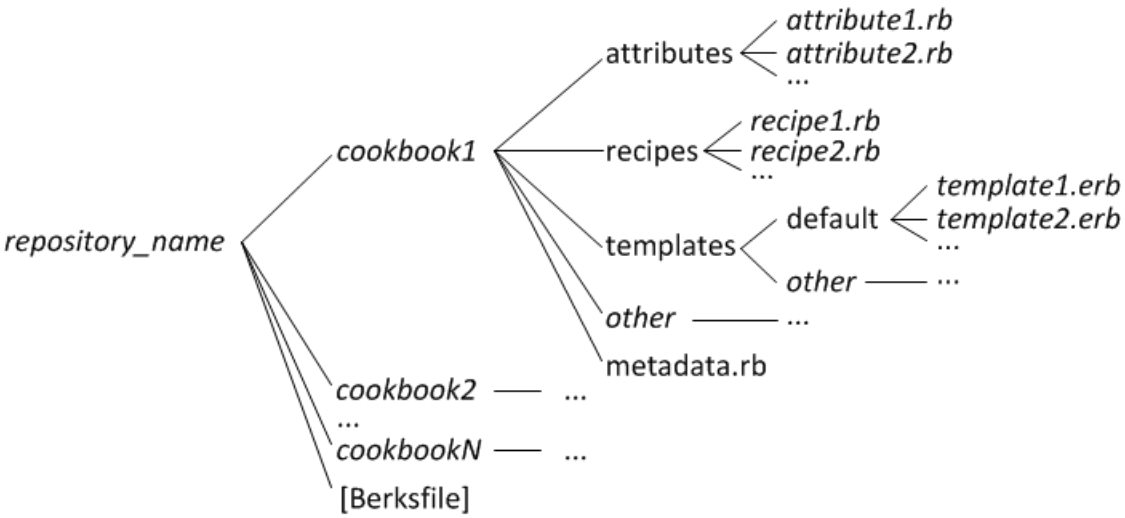
下列說明如何建構您的技術指南儲存庫，這取決於類型。圖中的斜體文字代表使用者定義的目錄和檔案名稱，包括儲存庫或封存名稱。

來源控制管理員

AWS OpsWorks Stacks 支援下列來源控制管理員：

- Linux 堆疊 – Git 和 Subversion
- Windows 堆疊 – Git

下列顯示所需的目錄和檔案結構：



- 技術指南目錄必須全在最上層。

存檔

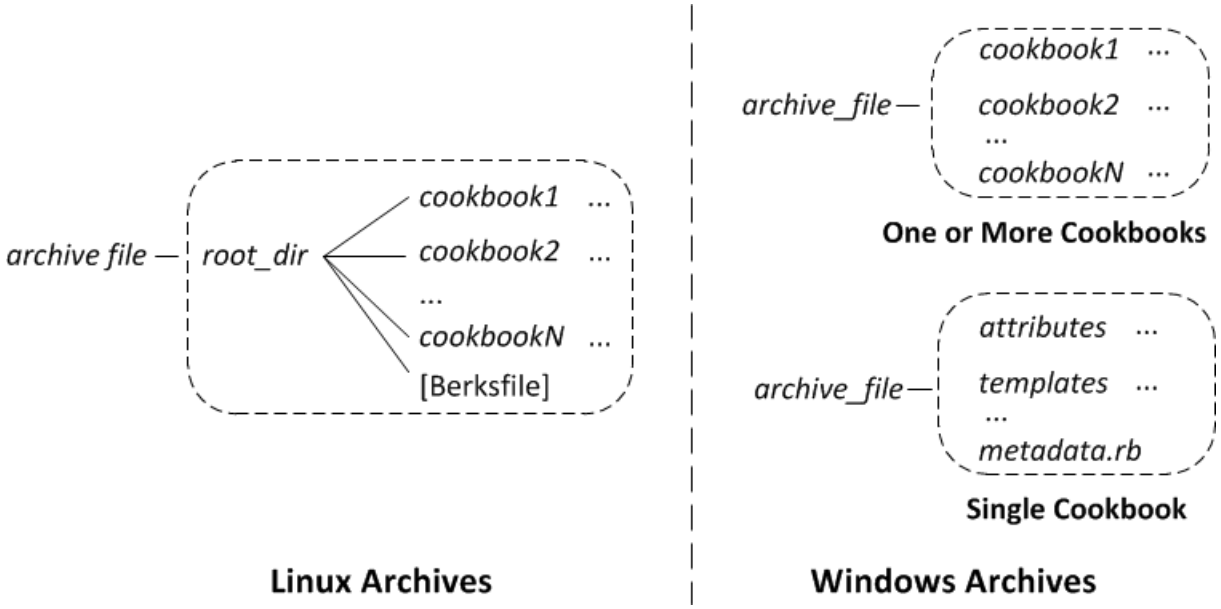
AWS OpsWorks Stacks 支援下列封存：

- Linux 堆疊 – zip、gzip、bzip2 或 tarball 檔案，存放在 Amazon S3 或網站 (HTTP 封存）。

  AWS OpsWorks Stacks 不支援未壓縮的 tarball。

- Windows 堆疊 – zip 和 tgz (gzip 壓縮 tar) 檔案，存放在 Amazon S3 上。

下列顯示所需的目錄和檔案結構，這取決於您執行的是 Linux 或 Windows 堆疊。技術指南結構和 SCM 儲存庫結構相同，所以使用省略號 (...) 表示。



- Linux 堆疊 – 技術指南目錄必須包含在根目錄中。
- Windows 堆疊 – 技術指南必須位於封存的頂層。

  如果您只有一個技術指南，您可以選擇略過技術指南目錄，將技術指南檔案放在最上層。在這種情況下， AWS OpsWorks Stacks 會從 metadata.rb 取得技術指南名稱。

每個技術指南目錄最少有下列標準目錄和檔案之一，一般全都有，它們必須使用標準名稱：

- `attributes` – 技術指南的屬性檔案。
- `recipes` – 技術指南的配方檔案。
- `templates` – 技術指南的範本檔案。
- `##` – 包含其他檔案類型的選用使用者定義目錄，例如定義或規格。
- `metadata.rb` – 技術指南的中繼資料。

針對 Chef 11.10 和更新版本，如果您的配方依賴其他技術指南，您必須在您技術指南的 `metadata.rb` 檔案中包含對應的 `depends` 陳述式。例如，若您的技術指南包含具有如 `include_recipe anothercookbook::somerecipe` 陳述式的配方，則您技術指南的 `metadata.rb` 檔案就必須包含下列內容：`depends "anothercookbook"`。如需詳細資訊，請參閱 About Cookbook Metadata。

範本必須在 `templates` 目錄的子目錄中，此目錄至少包含一或多個子目錄。這些子目錄也可以有子目錄。

- 範本通常有一個 `default` 子目錄，包含 Chef 預設使用的範本檔案。
- 「其他」代表可用於操作系統專屬範本的選用子目錄。
- 根據 File Specificity 中所述的命名慣例，Chef 會自動使用來自適當子目錄的範本。例如，在 Amazon Linux 和 Ubuntu 作業系統，您可以將作業系統專屬範本分別放在名為 `amazon` 或 `ubuntu` 的子目錄中。

如何處理自訂技術指南的詳細資訊，取決於您慣用的儲存庫類型。

使用封存

1. 使用上述章節顯示的資料夾結構實作您的技術指南。
2. 建立壓縮封存並將其上傳至 Amazon S3 儲存貯體或網站。

   如果您更新技術指南，您必須建立和上傳新的封存檔案。傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

使用 SCM

1. 使用前文顯示的結構設定 Git 或 Subversion 儲存庫。
2. 或者，使用儲存庫的版本控制功能來實作多個分支或版本。

   如果您更新技術指南，您可以在新的分支中如此做，只指示 OpsWorks 使用新版本。您也可以指定特定的標記版本。如需詳細資訊，請參閱指定自訂技術指南儲存庫。

安裝自訂技術指南 說明如何讓 AWS OpsWorks Stacks 在堆疊的執行個體上安裝您的技術指南儲存庫。

> **Important**
>
> 更新儲存庫中的現有技術指南後，您必須執行`update_cookbooks`堆疊命令，指示 AWS OpsWorks Stacks 更新每個線上執行個體的本機快取。如需詳細資訊，請參閱執行堆疊命令。

Chef 版本

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 支援多個版本的 Chef。您在建立堆疊時選取版本。 AWS OpsWorks 堆疊接著會在堆疊的所有執行個體上安裝該版本的 Chef，以及與該版本相容的一組內建配方。若您安裝任何自訂配方，他們必須和堆疊的 Chef 版本相容。

AWS OpsWorks Stacks 目前支援適用於 Linux 堆疊的 Chef 12、11.10、11.4 和 0.9 版，以及適用於 Windows 堆疊的 Chef 12.2 （目前為 Chef 12.22)。為了方便起見，通常只以主要和次要版本號碼稱呼它們。針對 Linux 堆疊，您可以在您建立堆疊時，使用組態管理員指定要使用的 Chef 版本。Windows 堆疊必須使用 Chef 12.2。如需詳細資訊 (包含將堆疊遷移至更近 Chef 版本的準則)，請參閱 Chef 版本。如需完整的版本資訊，請參閱AWS OpsWorks Stacks 作業系統。

Chef 12.2

Chef 12.2 支援於 2015 年 5 月推出，僅供 Windows 堆疊使用。Windows 堆疊上目前的 Chef 版本為 Chef 12.22。它會執行 Ruby 2.3.6，並使用本機模式中的 chef-client，以啟動名為 chef-zero 的本機記憶體內 Chef 伺服器。此伺服器的存在可讓配方使用 Chef search 和資料包。如實作配方：Chef 12.2中所說明，支援具有某些限制，但您可以執行許多社群技術指南，而無須進行修改。

Chef 12

Chef 12 支援於 2015 年 12 月推出，僅由 Linux 堆疊使用。它使用 Ruby 2.1.6 或 2.2.3 執行，並且使用可讓配方利用 Chef search 和資料包之本機模式中的 chef-client。如需詳細資訊，請參閱AWS OpsWorks Stacks 作業系統。

Chef 11.10

Chef 11.10 支援於 2014 年 3 月推出，僅由 Linux 堆疊使用。它使用 Ruby 2.0.0 執行，並且使用可讓配方利用 Chef search 和資料包之本機模式中的 chef-client。如實作配方：Chef 11.10中所說明，支援具有某些限制，但您可以執行許多社群技術指南，而無須進行修改。您也可以使用 Berkshelf 來管理您的技術指南依存項目。支援的 Berkshelf 版本取決於作業系統。如需詳細資訊，請參閱AWS OpsWorks Stacks 作業系統。您無法建立使用 Chef 11.10 的 CentOS 堆疊。

Chef 11.4

Chef 11.4 支援於 2013 年 7 月推出，僅由 Linux 堆疊使用。它使用 Ruby 1.8.7 執行，並且使用 chef-solo，不支援 Chef search 或資料包。您通常可以透過 Stacks AWS OpsWorks 使用相依於這些功能的社群技術指南，但您必須如 中所述進行修改遷移至新的 Chef 版本。您無法建立使用 Chef 11.4 的 CentOS 堆疊。美國東部（維吉尼亞北部）區域以外的區域端點不支援 Chef 11.4 堆疊。

Chef 0.9

Chef 0.9 僅由 Linux 堆疊使用，並且已不再受到支援。請注意以下詳細資訊：

- 您無法使用主控台建立新的 Chef 0.9 堆疊。

  您必須使用 CLI 或 API，或是使用不同版本的 Chef 建立堆疊之後，再修改堆疊的組態。
- 新的 AWS OpsWorks Stacks 功能不適用於 Chef 0.9 堆疊。
- 新的作業系統版本僅會針對 Chef 0.9 堆疊提供有限的支援。

  特別是，Amazon Linux 2014.09 和更新版本不支援使用依賴於 Ruby 1.8.7 的 Rails App Server 層的 Chef 0.9 堆疊。
- 包括歐洲（法蘭克福）在內的新 AWS 區域不支援 Chef 0.9 堆疊。

> **Note**
>
> 我們不建議針對新的堆疊使用 Chef 0.9。建議您將任何現有的堆疊盡快遷移至最新的 Chef 版本。

如果您想要搭配 Stacks AWS OpsWorks 使用社群技術指南，建議您為新的 Linux 堆疊指定 Chef 12，並將現有的 Linux 堆疊遷移至 Chef 12。您可以使用 AWS OpsWorks Stacks 主控台、API 或 CLI 將現有堆疊遷移至較新的 Chef 版本。如需詳細資訊，請參閱遷移至新的 Chef 版本。

主題

- 實作 Chef 12.2 Stacks 配方
- 實作 Chef 12 堆疊的配方
- 實作 Chef 11.10 堆疊的配方
- 實作 Chef 11.4 堆疊的配方
- 將現有 Linux 堆疊遷移至新的 Chef 版本

實作 Chef 12.2 Stacks 配方

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Chef 12.2 (目前為 Chef 12.22) 僅供 Windows 堆疊使用，且該堆疊也必須執行該 Chef 版本。

- 配方必須針對某些用途使用 Windows 限定屬性和資源。

  如需詳細資訊，請參閱 Chef for Microsoft Windows。
- Chef 執行使用 Ruby 2.3.6，因此您的配方可以使用新的 Ruby 語法。
- 配方可以使用 Chef search 和資料包。

  Chef 12.2 堆疊可以使用許多社群技術指南，無需修改。如需詳細資訊，請參閱使用 Chef 搜尋及使用資料包。
- 大部分在AWS OpsWorks Stacks Data Bag 參考和內建技術指南屬性中說明的堆疊組態及部署屬性都可供 Windows 配方使用。

  您可以使用 Chef search 取得這些屬性值。如需範例，請參閱「使用 Chef 搜尋取得屬性值」。如需屬性清單，請參閱AWS OpsWorks Stacks Data Bag 參考。

實作 Chef 12 堆疊的配方

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Chef 12 堆疊與 Chef 11.10 堆疊相比有下列優點：

- Chef 執行使用 Ruby 2.1.6，因此您的配方可以使用新的 Ruby 語法。
- Chef 12 堆疊可以使用更多社群技術指南，而無須修改。由於不再有任何內建的技術指南，因此再也不會發生內建技術指南與自訂技術指南的名稱發生衝突的情況。
- 您不再受限於 AWS OpsWorks Stacks 為其提供預先建置套件的 Berkshelf 版本。Berkshelf 不再安裝在 Chef 12 中的 AWS OpsWorks Stacks 執行個體上。您可以改為在您的本機工作站上使用任何 Berkshelf 版本。
- Stacks AWS OpsWorks 隨 Chef 12 (Elastic Load Balancing、Amazon RDS 和 Amazon ECS) 提供的內建技術指南與自訂技術指南之間現在有明確的區隔。這可讓針對失敗的 Chef 執行進行故障診斷更容易。

實作 Chef 11.10 堆疊的配方

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Chef 11.10 堆疊與 Chef 11.4 堆疊相比有下列優點：

- Chef 執行使用 Ruby 2.0.0，因此您的配方可以使用新的 Ruby 語法。
- 配方可以使用 Chef search 和資料包。

  Chef 11.10 堆疊可以使用許多社群技術指南，而無須修改。
- 您可以使用 Berkshelf 來管理技術指南。

Berkshelf 提供更多更靈活的方式來管理您的自訂技術指南，以及在堆疊中使用社群技術指南。

- 技術指南必須在 `metadata.rb` 中宣告依存項目。

  若您的技術指南依存於其他技術指南，您必須在您技術指南的 `metadata.rb` 檔案中包含該依存項目。例如，若您的技術指南包含具有如 `include_recipe anothercookbook::somerecipe` 陳述式的配方，則您技術指南的 `metadata.rb` 檔案就必須包含下列內容：`depends "anothercookbook"`。
- AWS OpsWorks 只有在堆疊包含 MySQL layer 時， Stacks 才會在堆疊的執行個體上安裝 MySQL 用戶端。
- AWS OpsWorks 只有在堆疊包含 Ganglia 層時， Stacks 才會在堆疊的執行個體上安裝 Ganglia 用戶端。
- 若部署執行 `bundle install` 但安裝失敗，部署也會失敗。

> **Important**
>
> 請勿針對自訂或社群技術指南重複使用內建技術指南的名稱。和內建技術指南具有相同名稱的自訂技術指南可能會失敗。如需 Chef 11.10、11.4 和 0.9 堆疊可用內建技術指南的完整清單，請參閱 GitHub 上的 opsworks-cookbooks 儲存庫。
>
> 帶有非 ASCII 字元並在 Chef 0.9 和 11.4 堆疊上可成功執行的技術指南，在 Chef 11.10 堆疊上可能會失敗。原因是 Chef 11.10 堆疊會針對 Chef run 使用 Ruby 2.0.0，該版本比 Ruby 1.8.7 具有更嚴格的編碼規範。為了確保這類技術指南能在 Chef 11.10 堆疊上成功執行，每個使用非 ASCII 字元的檔案都應在其頂端具備註解，提供關於編碼的提示。例如，針對 UTF-8 編碼，註解應為 `# encoding: UTF-8`。如需 Ruby 2.0.0 編碼的詳細資訊，請參閱 Encoding。

主題

- 技術指南安裝與優先順序
- 使用 Chef 搜尋
- 使用資料包
- 使用 Berkshelf

技術指南安裝與優先順序

安裝 AWS OpsWorks Stacks 技術指南的程序對 Chef 11.10 堆疊的運作方式與舊版 Chef 略有不同。對於 Chef 11.10 堆疊，在 AWS OpsWorks Stacks 安裝內建、自訂和 Berkshelf 技術指南之後，它會以下列順序將它們合併到通用目錄：

1. 內建技術指南。
2. Berkshelf 技術指南 (若有的話)。
3. 自訂技術指南 (若有的話)。

當 AWS OpsWorks Stacks 執行此合併時，它會複製目錄的完整內容，包括配方。若有任何重複項目，則會套用下列規則：

- Berkshelf 技術指南的內容會優先於內建技術指南。
- 自訂技術指南的內容會優先於 Berkshelf 技術指南。

為了示範此程序運作的方式，請考慮以下案例，其中三個技術指南目錄都包含名為 `mycookbook` 的技術指南：

- 內建技術指南 – `mycookbook`包含名為 的屬性檔案`someattributes.rb`、名為 的範本檔案`sometemplate.erb`，以及名為 的配方`somerecipe.rb`。
- Berkshelf 技術指南 – `mycookbook`包括 `sometemplate.erb`和 `somerecipe.rb`。
- 自訂技術指南 – `mycookbook`包含 `somerecipe.rb`。

合併的技術指南包含以下內容：

- 來自內建技術指南的 `someattributes.rb`。
- 來自 Berkshelf 技術指南的 `sometemplate.erb`。
- 來自自訂技術指南的 `somerecipe.rb`。

> **Important**
>
> 建議您不要透過將整個內建技術指南複製到您的儲存庫，然後修改技術指南的一部分，來自訂您的 Chef 11.10 堆疊。這樣做會覆寫整個內建技術指南 (包含配方)。如果 AWS OpsWorks

Stacks 更新了該技術指南，除非您手動更新私有複本，否則您的堆疊將不會受益於這些更新。如需如何自訂堆疊的詳細資訊，請參閱自訂 AWS OpsWorks Stacks。

使用 Chef 搜尋

您可以在您的配方中使用 Chef `search` 方法來查詢堆疊資料。您可以使用與 Chef 伺服器相同的語法，但 AWS OpsWorks Stacks 會從本機節點物件取得資料，而不是查詢 Chef 伺服器。此資料包括：

- 執行個體的堆疊組態及部署屬性。
- 執行個體的內建和自訂技術指南之屬性檔案的屬性。
- 由 Ohai 收集的系統資料。

堆疊組態和部署屬性包含配方通常透過搜尋取得的大部分資訊，包括堆疊中每個線上執行個體的主機名稱和 IP 地址等資料。 AWS OpsWorks 堆疊會為每個生命週期事件更新這些屬性，以確保它們準確反映目前的堆疊狀態。這表示您通常可以在您的堆疊中使用搜尋依存的社群配方，而無須修改。search 方法仍然會傳回適當的資料，只是該資料會來自堆疊組態及部署屬性，而非伺服器。

AWS OpsWorks Stacks 搜尋的主要限制是僅處理本機節點物件中的資料，特別是堆疊組態和部署屬性。因此，下列資料類型可能無法透過 search 取得：

- 在其他執行個體上的本機定義屬性。

  如果配方在本機定義 屬性，該資訊不會回報給 AWS OpsWorks Stacks 服務，因此您無法使用搜尋從其他執行個體存取該資料。
- 自訂 `deploy` 屬性。

  您可以在您部署應用程式時指定自訂 JSON，對應的屬性即會為該部署安裝在堆疊的執行個體上。但是，若您僅部署到選取的執行個體，屬性也會在那些執行個體上安裝。在所有其他執行個體上查詢那些自訂 JSON 屬性都會失敗。此外，自訂屬性也僅會包含在該特定部署的堆疊組態和部署 JSON 中。您僅能在下一次生命週期事件安裝新的一組堆疊組態及部署屬性之前存取他們。請注意，若您指定堆疊的自訂 JSON，屬性便會針對每個生命週期事件於每個執行個體上安裝，並且永遠可透過 search 進行存取。
- 來自其他執行個體的 Ohai 資料。

  Chef 的 Ohai 工具會取得執行個體上的各種系統資料，並新增到節點物件。這項資料會存放於本機，而不會報告回 AWS OpsWorks Stacks 服務，因此 search 無法從其他執行個體存取 Ohai 資料。但是，這項資料的其中一部分可能會包含在堆疊組態及部署屬性中。

- 離線執行個體。

  堆疊組態及部署屬性只包含線上執行個體的資料。

下列配方摘要會示範如何使用 search 取得 PHP layer 執行個體的私有 IP 地址。

```
appserver = search(:node, "role:php-app").first
Chef::Log.info("The private IP is '#{appserver[:private_ip]}'")
```

> **Note**
>
> 當 AWS OpsWorks Stacks 將堆疊組態和部署屬性新增至節點物件時，它實際上會建立兩組 layer 屬性，每個屬性都具有相同的資料。`layers` 命名空間中有一個集合，這是 Stacks AWS OpsWorks 存放資料的方式。另一組則位於 `role` 命名空間，即 Chef 伺服器存放對等資料的方式。`role` 命名空間的目的是允許為 Chef 伺服器實作的搜尋程式碼在 Stacks AWS OpsWorks 執行個體上執行。如果您專門為 AWS OpsWorks Stacks 編寫程式碼，您可以在上述範例中使用 `layers:php-app`或 `role:php-app search`，並傳回相同的結果。

使用資料包

您可以在您的配方中使用 Chef `data_bag_item` 方法來查詢資料包中的資訊。您會使用與您在使用 Chef 伺服器時相同的語法，但 AWS OpsWorks Stacks 會從執行個體的堆疊組態及部署屬性取得資料。不過， AWS OpsWorks Stacks 目前不支援 Chef 環境，因此 `node.chef_environment` 一律會傳回 `_default`。

您會透過使用自訂 JSON 建立資料包，來將一或多個屬性新增至 `[:opsworks][:data_bags]` 屬性。以下範例示範在自訂 JSON 中建立資料包的一般格式。

> **Note**
>
> 您無法透過將其新增至您的技術指南儲存庫，來建立資料包。您必須使用自訂 JSON。

```
{
  "opsworks": {
```

```
    "data_bags": {
      "bag_name1": {
        "item_name1: {
          "key1" : “value1”,
          "key2" : “value2”,
          ...
        }
      },
      "bag_name2": {
        "item_name1": {
          "key1" : “value1”,
          "key2" : “value2”,
          ...
        }
      },
      ...
    }
  }
}
```

您通常會指定堆疊的自訂 JSON，針對每個接下來的生命週期事件，於每個執行個體上安裝自訂屬性。您也可以在您部署應用程式時指定自訂 JSON，但那些屬性只會針對該部署進行安裝，並且可能只會安裝到選取的執行個體。如需詳細資訊，請參閱部署應用程式。

下列自訂 JSON 範例會建立名為 myapp 的資料包。其中有一個項目 (mysql)，以及兩個鍵/值對。

```
{ "opsworks": {
    "data_bags": {
      "myapp": {
        "mysql": {
          "username": "default-user",
          "password": "default-pass"
        }
      }
    }
  }
}
```

若要在您的配方中使用資料，您可以呼叫 data_bag_item 並將資料包和值名稱傳遞給它，如下列摘要所示。

```
mything = data_bag_item("myapp", "mysql")
Chef::Log.info("The username is '#{mything['username']}' ")
```

若要修改資料包中的資料，您只需修改自訂 JSON，它便會針對下一次的生命週期事件安裝在堆疊的執行個體上。

使用 Berkshelf

在 Chef 0.9 和 Chef 11.4 堆疊中，您只能安裝一個自訂技術指南儲存庫。在 Chef 11.10 堆疊中，您可以使用 Berkshelf 來管理您的技術指南和其依存項目，讓您可以從多個儲存庫安裝技術指南。(如需詳細資訊，請參閱 本機封裝技術指南依存性。) 特別是，使用 Berkshelf，您可以直接從儲存庫安裝 AWS OpsWorks 與 Stacks 相容的社群技術指南，而不必將其複製到自訂技術指南儲存庫。支援的 Berkshelf 版本取決於作業系統。如需詳細資訊，請參閱AWS OpsWorks Stacks 作業系統。

若要使用 Berkshelf，您必須明確啟用它，如安裝自訂技術指南中所說明。然後，在您的技術指南儲存庫根目錄中包含一個 `Berksfile` 檔案，指定要安裝的技術指南。

若要在 Berksfile 中指定外部技術指南來源，請在檔案的頂端包含一個來源屬性，指定預設儲存庫 URL。Berkshelf 會在來源 URL 中尋找技術指南，除非您明確指定儲存庫。然後，在每個您希望安裝的技術指南中包含一行文字，格式如下：

```
cookbook 'cookbook_name', ['>= cookbook_version'], [cookbook_options]
```

cookbook 之後的欄位會指定特定技術指南。

- *cookbook_name* – （必要）指定技術指南的名稱。

  若您沒有包含任何其他欄位，Berkshelf 會從指定來源 URL 安裝技術指南。
- *cookbook_version* – （選用）指定技術指南版本。

  您可以使用像是 = 或 >= 等前綴，來指定特定版本或可接受的版本範圍。若您並未指定版本，Berkshelf 會安裝最新的版本。
- *cookbook_options* – （選用）最後一個欄位是包含一或多個索引鍵/值對的雜湊，指定儲存庫位置等選項。

  例如，您可以包含一個 `git` 鍵，指定特定的 Git 儲存庫，以及一個 `tag` 鍵，指定特定的儲存庫分支。指定儲存庫分支通常是確保安裝您偏好之技術指南的最佳方式。

> **Important**
>
> 請不要透過在您的 Berksfile 中包含 `metadata` 並在 `metadata.rb` 中宣告技術指南依存項目，來宣告技術指南。若要使其正常執行，兩個檔案都必須位於相同的目錄中。使用 AWS OpsWorks Stacks 時，Berksfile 必須位於儲存庫的根目錄中，但`metadata.rb`檔案必須位於各自的技術指南目錄中。您應改為在 Berksfile 中明確宣告外部技術指南。

以下為 Berksfile 的範例，示範指定技術指南的不同方式。如需如何建立 Berksfile 的詳細資訊，請參閱 Berkshelf。

```
source "https://supermarket.chef.io"

cookbook 'apt'
cookbook 'bluepill', '>= 2.3.1'
cookbook 'ark', git: 'git://github.com/opscode-cookbooks/ark.git'
cookbook 'build-essential', '>= 1.4.2', git: 'git://github.com/opscode-cookbooks/build-essential.git', tag: 'v1.4.2'
```

此檔案會安裝以下技術指南：

- 來自社群技術指南儲存庫的最新版本 `apt`。
- 來自社群技術指南的最新版本 `bluepill` (只要其版本為 2.3.1 或更新版本)。
- 來自指定儲存庫的最新版本 `ark`。

  此範例的 URL 為 GitHub 上公有社群技術指南儲存庫的 URL，但您可以從其他儲存庫安裝技術指南，包含私有儲存庫。如需詳細資訊，請參閱 Berkshelf。
- 來自指定儲存庫 v1.4.2 分支的 `build-essential` 技術指南。

除了 Berksfile，自訂技術指南儲存庫可以包含自訂技術指南。在這種情況下， AWS OpsWorks Stacks 會安裝兩組技術指南，這表示執行個體最多可以有三個技術指南儲存庫。

- 內建的技術指南會安裝到 `/opt/aws/opsworks/current/cookbooks`。
- 或您的自訂技術指南儲存庫包含技術指南，他們會安裝到 `/opt/aws/opsworks/current/site-cookbooks`。
- 若您已啟用 Berkshelf，且您的自訂技術指南儲存庫包含 Berksfile，指定的技術指南便會安裝到 `/opt/aws/opsworks/current/berkshelf-cookbooks`。

內建技術指南和您的自訂技術指南會在設定期間安裝在每個執行個體上，除非您手動執行更新自訂技術指南堆疊命令，否則後續不會更新。 AWS OpsWorks 堆疊`berks install`會在每次 Chef 執行時執行，因此您的 Berkshelf 技術指南會根據下列規則，針對每個生命週期事件更新：

- 若您在儲存庫中有新的技術指南版本，此操作會從儲存庫更新技術指南。
- 否則，此操作會從本機快取更新 Berkshelf 技術指南。

> **Note**
>
> 操作會覆寫 Berkshelf 技術指南，因此若您已修改任何技術指南的本機複本，變更將會遭到覆寫。如需詳細資訊，請參閱 Berkshelf

您也可以透過執行更新自訂技術指南堆疊命令來更新您的 Berkshelf 技術指南，該命令會同時更新 Berkshelf 技術指南和您的自訂技術指南。

實作 Chef 11.4 堆疊的配方

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Important**
>
> 請勿針對自訂或社群技術指南重複使用內建技術指南的名稱。和內建技術指南具有相同名稱的自訂技術指南可能會失敗。如需 Chef 11.10、11.4 和 0.9 堆疊可用內建技術指南的完整清單，請參閱 GitHub 上的 opsworks-cookbooks 儲存庫。

Chef 11.4 堆疊的主要限制為配方無法使用 Chef search 或資料包。不過， AWS OpsWorks Stacks 會在每個執行個體上安裝堆疊組態和部署屬性，其中包含您可以透過搜尋取得的大部分資訊，包括下列項目：

- 來自主控台的使用者定義資料，例如主機或應用程式名稱。

- Stacks AWS OpsWorks 服務產生的堆疊組態資料，例如堆疊的 layer、應用程式和執行個體，以及 IP 地址等每個執行個體的詳細資訊。
- 包含由使用者提供之資料的自訂 JSON 屬性，其用途與資料包幾乎相同。

AWS OpsWorks Stacks 會在每個生命週期事件的每個執行個體上安裝目前版本的堆疊組態和部署屬性，然後再開始事件的 Chef 執行。資料會透過標準 `node[:attribute][:child_attribute][...]` 語法，供配方使用。例如，堆疊組態及部署屬性包含堆疊名稱，`node[:opsworks][:stack][:name]`。

以下來自其中一個內建配方的摘要會取得堆疊名稱，並用它來建立組態檔案。

```
template '/etc/ganglia/gmetad.conf' do
  source 'gmetad.conf.erb'
  mode '0644'
  variables :stack_name => node[:opsworks][:stack][:name]
  notifies :restart, "service[gmetad]"
end
```

許多堆疊組態及部署屬性值包含多個屬性。您必須逐一查看這些屬性，才能取得您需要的資訊。以下範例顯示來自堆疊組態及部署屬性的摘要。為了方便，以下以 JSON 物件表示。它包含一個最上層屬性 `deploy`，其針對每個堆疊的應用程式都包含一個屬性，以該應用程式的短名命名。

```
{
  ...
  "deploy": {
    "app1_shortname": {
      "document_root": "app1_root",
      "deploy_to": "deploy_directory",
      "application_type": "php",
      ...
    },
    "app2_shortname": {
      "document_root": "app2_root",
      ...
    }
  },
  ...
}
```

每個應用程式屬性都包含一組描述應用程式的屬性。例如，deploy_to 屬性表示應用程式的部署目錄。以下摘要會設定每個應用程式部署目錄的使用者、群組及路徑。

```
node[:deploy].each do |application, deploy|
  opsworks_deploy_dir do
    user deploy[:user]
    group deploy[:group]
    path deploy[:deploy_to]
  end
  ...
end
```

如需堆疊組態及部署屬性的詳細資訊，請參閱自訂 AWS OpsWorks Stacks。如需部署目錄的詳細資訊，請參閱部署配方。

Chef 11.4 堆疊不支援資料包，但您可以透過指定自訂 JSON，將任意資料新增到堆疊組態及部署屬性。您的配方接著便能透過使用標準 Chef 節點語法來存取資料。如需詳細資訊，請參閱使用自訂 JSON。

如果您需要加密資料包的功能，其中一個選項是將敏感屬性存放在安全的位置，例如私有 Amazon S3 儲存貯體。然後，您的配方可以使用安裝在所有 Stacks AWS OpsWorks 執行個體上的 AWS Ruby 開發套件，從儲存貯體下載資料。

> **Note**
>
> 每個 AWS OpsWorks Stacks 執行個體都有一個執行個體描述檔。相關聯的 IAM 角色會指定執行個體上執行的應用程式可存取哪些 AWS 資源。若要讓您的配方存取 Amazon S3 儲存貯體，角色的政策必須包含類似以下的陳述式，這會授予從指定儲存貯體擷取檔案的許可。
>
> ```
> "Action": ["s3:GetObject"],
> "Effect": "Allow",
> "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
> ```
>
> 如需執行個體描述檔的詳細資訊，請參閱指定在 EC2 執行個體上執行之應用程式的許可。

將現有 Linux 堆疊遷移至新的 Chef 版本

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以使用 AWS OpsWorks Stacks 主控台、API 或 CLI 將 Linux 堆疊遷移至較新的 Chef 版本。但是，您的配方可能需要修改，才能和更新的版本相容。在準備遷移堆疊時，請考慮以下事項。

- 您無法透過編輯或複製堆疊，將 AWS OpsWorks Stacks 堆疊版本從 Chef 11 變更為 Chef 12。Chef 主要版本的升級無法透過本節中的程序執行。如需從 Chef 11.10 轉換至 Chef 12 的詳細資訊，請參閱實作配方：Chef 12。
- 從一個版本的 Chef 轉換至另一個版本，會涉及幾項變更，其中也有一些重大變更。

  如需從 Chef 0.9 轉換至 Chef 11.4 的詳細資訊，請參閱遷移至新的 Chef 版本。如需從 Chef 11.4 轉換至 Chef 11.10 的詳細資訊，請參閱實作配方：Chef 11.10。如需從 Chef 11.10 轉換至 Chef 12 的詳細資訊，請參閱實作配方：Chef 12。
- Chef 執行在 Chef 0.9 和 Chef 11.4 堆疊 (Ruby 1.8.7)、Chef 11.10 堆疊 (Ruby 2.0.0)，以及 Chef 12 堆疊 (Ruby 2.1.6) 上皆使用了不同的 Ruby 版本。

  如需詳細資訊，請參閱Ruby 版本。
- Chef 11.10 堆疊處理技術指南安裝的方式與 Chef 0.9 或 Chef 11.4 堆疊不同。

  這項差異可能會導致在遷移使用自訂技術指南的堆疊至 Chef 11.10 時發生問題。如需詳細資訊，請參閱技術指南安裝與優先順序。

以下是將 Chef 堆疊遷移至更新 Chef 版本的建議準則：

將堆疊遷移至更新的 Chef 版本

1. 複製您的生產堆疊。在 Clone Stack (複製堆疊) 頁面上，按一下 Advanced>> (進階>>) 以顯示 Configuration Management (組態管理) 區段，然後將 Chef version (Chef 版本) 變更至下一個更高的版本。

> **Note**
>
> 若您是使用 Chef 0.9 堆疊開始的，您無法直接升級至 Chef 11.10。您必須先升級到 Chef 11.4。若您希望在測試您的配方前先將您的堆疊遷移至 Chef 11.10，請等待 20 分鐘讓更新執行，再將堆疊從 11.4 升級到 11.10。

2. 將執行個體新增到 layer，然後在測試或預備系統上測試複製堆疊的應用程式和技術指南。如需詳細資訊，請參閱 All about Chef ...。
3. 當您滿意測試結果時，請執行下列其中一項作業：

   - 若這是您需要的 Chef 版本，您可以直接使用複製堆疊做為您的生產堆疊，或在您的生產堆疊上重設 Chef 版本。
   - 若您正在兩階段式的將 Chef 0.9 堆疊遷移至 Chef 11.10，請重複程序，將堆疊從 Chef 11.4 遷移至 Chef 11.10。

> **Note**
>
> 當您在測試配方時，您可以使用 SSH 連線到執行個體，然後使用執行個體代理程式 CLI run_command 命令，來執行與各種生命週期事件關聯的配方。代理程式 CLI 在測試安裝配方時特別有用，因為即使安裝失敗，執行個體並未達到線上狀態，您也可以使用它。您也可以使用安裝堆疊命令來重新執行安裝配方，但該命令只有在安裝成功且執行個體處於線上狀態時，才能使用。

您也可以將執行中的堆疊更新至新的 Chef 版本。

將執行中的堆疊更新至新的 Chef 版本

1. 編輯堆疊，變更 Chef version (Chef 版本) 堆疊設定。
2. 儲存新設定並等待 AWS OpsWorks Stacks 更新執行個體，這通常需要 15 - 20 分鐘。

> **Important**
>
> AWS OpsWorks Stacks 不會同步 Chef 版本更新與生命週期事件。若您希望在生產堆疊上更新 Chef 版本，您必須小心，確保更新在下一次的生命週期事件發生前完成。如果事件發生，通常是部署或設定事件，執行個體代理程式會更新您的自訂技術指南，並執行事件指派的配方，無

論版本更新是否完成。沒有直接的方式能判斷版本更新是否已完成，但部署日誌中會包含 Chef 版本。

Ruby 版本

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Linux 堆疊中的所有執行個體都已安裝 Ruby。 AWS OpsWorks Stacks 會在每個執行個體上安裝 Ruby 套件，用來執行 Chef 配方和執行個體代理程式。 AWS OpsWorks Stacks 會根據堆疊執行的 Chef 版本來決定 Ruby 版本。請勿嘗試修改此版本，這樣做可能會停用執行個體代理程式。

AWS OpsWorks Stacks 不會在 Windows 堆疊上安裝應用程式 Ruby 可執行檔。Chef 12.2 用戶端隨附 Ruby 2.0.0 p451，但 Ruby 可執行檔不會新增至執行個體的 PATH 環境變數。如果您想要使用此可執行檔執行 Ruby 程式碼，它位在您 Windows 磁碟機的 `\opscode\chef\embedded\bin\ruby.exe`。

下表摘要說明 AWS OpsWorks Stacks Ruby 版本。可用的應用程式 Ruby 版本也取決於執行個體的作業系統。如需詳細資訊，包括可用的修補程式版本，請參閱 AWS OpsWorks Stacks 作業系統。

Chef 版本	Chef Ruby 版本	可用的應用程式 Ruby 版本
0.9 (c)	1.8.7	1.8.7(a)、1.9.3(e)、2.0.0
11.4 (c)	1.8.7	1.8.7(a)、1.9.3(e)、2.0.0、2.1、2.2.0、2.3
11.10	2.0.0-p481	1.9.3(c、e)、2.0.0、2.1、2.2.0、2.3、2.6.1
12 (b)	2.1.6、2.2.3	無
12.22 (d)	2.3.6	無

(a) 不適用於 Amazon Linux 2014.09 及更新版本、Red Hat Enterprise Linux (RHEL)，或 Ubuntu 14.04 LTS。

(b) 僅適用於 Linux 堆疊。

(c) 不適用於 RHEL。

(d) 僅適用於 Windows 堆疊。主要版本為 12.2。目前的次要版本為 12.22。

(e) 棄用已完成；支援已結束。

安裝位置取決於 Chef 版本：

- 應用程式對所有 Chef 版本都使用 `/usr/local/bin/ruby` 可執行檔。
- 若為 Chef 0.9 和 11.4，執行個體代理程式和 Chef 配方使用 `/usr/bin/ruby` 可執行檔。
- 若為 Chef 11.10，執行個體代理程式和 Chef 配方使用 `/opt/aws/opsworks/local/bin/ruby` 可執行檔。

安裝自訂技術指南

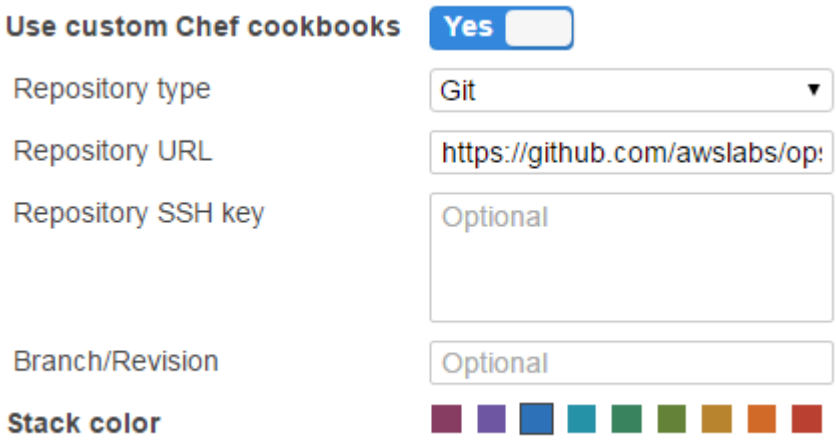
> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

若要使堆疊安裝及使用自訂技術指南，您必須設定堆疊，啟用自訂技術指南 (若還未設定的話)。您接著必須提供儲存庫 URL 和任何相關資訊 (例如密碼)。

> **Important**
>
> 將堆疊設定為支援自訂技術指南之後， AWS OpsWorks Stacks 會在啟動時在所有新執行個體上自動安裝技術指南。不過，您必須執行更新自訂技術指南堆疊命令，明確指示 AWS OpsWorks Stacks 在任何現有的執行個體上安裝新的或更新的技術指南。 如需詳細資訊，請參閱更新自訂技術指南。在您於堆疊上啟用 Use custom Chef cookbooks (使用自訂 Chef 技術指南) 前，請確認您執行的自訂和社群技術指南支援您堆疊使用的 Chef 版本。

若要為自訂技術指南設定堆疊

1. 在您的堆疊頁面上，按一下 Stack Settings (堆疊設定) 顯示其 Settings (設定) 頁面，按一下 Edit (編輯) 以編輯設定。
2. 將 Use custom Chef cookbooks (使用自訂 Chef 技術指南) 切換為 Yes (是)。



3. 設定您的自訂技術指南。

當您完成後，按一下 Save (儲存) 以儲存更新的堆疊。

指定自訂技術指南儲存庫

Linux 堆疊可以從下列任何儲存庫類型安裝自訂技術指南：

- HTTP 或 Amazon S3 封存。

  它們可以是公有或私有，但 Amazon S3 通常是私有封存的偏好選項。
- Git 和 Subversion 儲存庫提供來源控制和具有多個版本的能力。

Windows 堆疊可以從 Amazon S3 封存和 Git 儲存庫安裝自訂技術指南。

所有儲存庫類型都有以下必要欄位。

- 儲存庫類型 – 儲存庫類型
- 儲存庫 URL – 儲存庫 URL

AWS OpsWorks Stacks 支援公有託管的 Git 儲存庫網站，例如 GitHub 或 Bitbucket，以及私有託管的 Git 伺服器。針對 Git 儲存庫，您必須使用下列其中一個 URL 格式，取決於儲存庫為公有或私有。針對 Git 子模組遵循相同的 URL 準則。

針對公有 Git 儲存庫，使用 HTTPS 或 Git 唯讀通訊協定：

- Git 唯讀 – `git://github.com/amazonwebservices/opsworks-example-cookbooks.git`。
- HTTPS – `https://github.com/amazonwebservices/opsworks-example-cookbooks.git`。

針對私有 Git 儲存庫，您必須使用 SSH 讀取/寫入格式，如下列範例所示：

- Github 儲存庫 – `git@github.com:`*`project`*`/`*`repository`*。
- Git 伺服器上的儲存庫 – *`user`*`@`*`server`*`:`*`project`*`/`*`repository`*

其餘設定則會根據儲存庫類型而有所不同，如以下章節所說明。

HTTP 封存

針對 Repository type (儲存庫類型) 選取 Http Archive (Http 封存) 會顯示兩個額外設定，若封存受到密碼保護，您便必須完成這些設定。

- 使用者名稱 – 您的使用者名稱
- 密碼 – 您的密碼

Amazon S3 封存

選取 S3 Archive for Repository 類型會顯示下列額外的選用設定。 AWS OpsWorks 無論您使用 Stacks API 或主控台， Stacks 都可以使用 Amazon EC2 角色（主機作業系統管理員身分驗證） AWS OpsWorks 存取您的儲存庫。

- 存取金鑰 ID – AWS 存取金鑰 ID，例如 AKIAIOSFODNN7EXAMPLE。
- 私密存取金鑰 – 對應的 AWS 私密存取金鑰，例如 wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY。

Git 儲存庫

在 Source Control (來源控制) 下方選取 Git 會顯示下列額外選擇性設定：

Repository SSH key (儲存庫 SSH 金鑰)

您必須指定部署 SSH 金鑰才能存取私有 Git 儲存庫。針對 Git 子模組，指定的金鑰必須要能存取這些子模組。如需詳細資訊，請參閱使用 Git 儲存庫 SSH 金鑰。

> **Important**
>
> 部署的 SSH 金鑰不需要密碼； AWS OpsWorks Stacks 無法傳遞密碼。

Branch/Revision (分支/修訂)

如果儲存庫有多個分支， AWS OpsWorks Stacks 預設會下載主分支。若要指定特定分支，請輸入分支名稱、SHA1 雜湊或標籤名稱。若要指定特定的遞交，請輸入完整 40 個八進位碼的遞交 ID。

Subversion 儲存庫

在 Source Control (來源控制) 下方選取 Subversion 會顯示下列額外選擇性設定：

- 使用者名稱 – 私有儲存庫的使用者名稱。
- 密碼 – 您的密碼，適用於私有儲存庫。
- Revision–【選用】 如果您有多個修訂，則為修訂名稱。

若要指定分支或標籤，您必須修改儲存庫 URL，例如：**`http://repository_domain/repos/myapp/branches/my-apps-branch`** 或 **`http://repository_domain_name/repos/calc/myapp/my-apps-tag`**。

更新自訂技術指南

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

當您為 AWS OpsWorks Stacks 提供自訂技術指南時，內建的設定配方會在每個新啟動的執行個體上建立本機快取，並將技術指南下載至快取。然後， AWS OpsWorks Stacks 會從快取執行配方，而非儲存庫。如果您修改儲存庫中的自訂技術指南，您必須確保更新的技術指南已安裝在執行個體的本機快取上。當新的技術指南啟動時， AWS OpsWorks Stacks 會自動將最新的技術指南部署到新的執行個體。不過，現有執行個體的情況則不同：

- 您必須將更新的自訂技術指南手動部署到線上執行個體。
- 您不必將更新的自訂技術指南部署到離線的執行個體後端執行個體，包括以負載為基礎和以時間為基礎的執行個體。

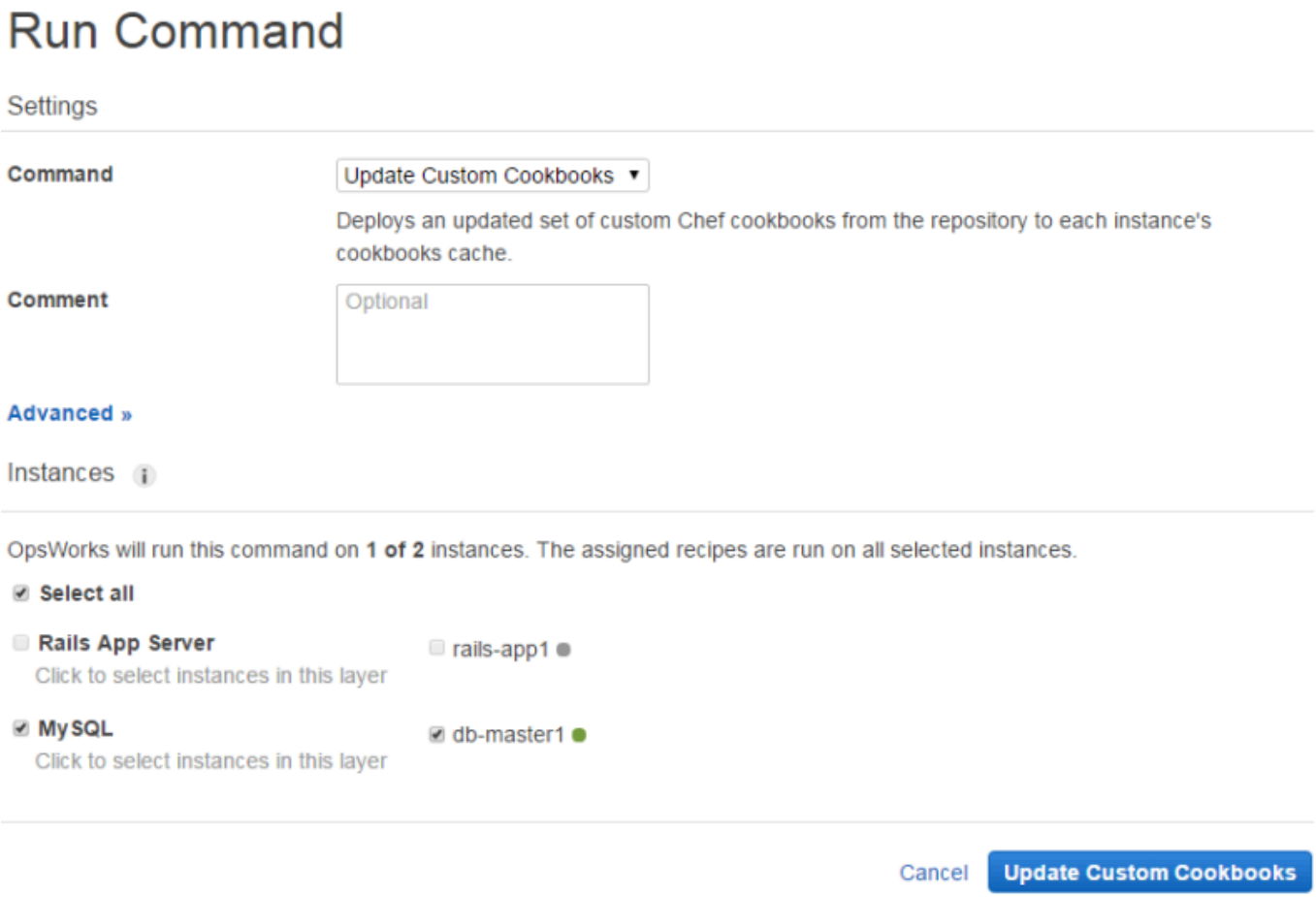
  AWS OpsWorks Stacks 會在執行個體重新啟動時自動部署目前的技術指南。
- 您必須啟動不以負載為基礎或以時間為基礎的全天候離線 EBS 後端執行個體。
- 您不能啟動以負載為基礎和以時間為基礎的離線 EBS 後端執行個體，因此最簡單的方法是刪除離線執行個體，並新增執行個體加以取代。

  由於它們現在是新的執行個體，因此當執行個體啟動時， AWS OpsWorks Stacks 會自動部署目前的自訂技術指南。

若要手動更新自訂技術指南

1. 使用修改過的技術指南更新儲存庫。 AWS OpsWorks Stacks 會使用您最初安裝技術指南時提供的快取 URL，因此技術指南根檔案名稱、儲存庫位置和存取權不應變更。

   - 對於 Amazon S3 或 HTTP 儲存庫，請以具有相同名稱的新 .zip 檔案取代原始 .zip 檔案。
   - 若是 Git 或 Subversion 儲存庫，編輯您的堆疊設定以將 Branch/Revision (分支/修訂) 欄位變更為新版本。
2. 在堆疊的頁面上，按一下 Run Command (執行命令)，並選取 Update Custom Cookbooks (更新自訂技術指南) 命令。



3. 視需要新增註解。
4. 或者，指定命令的自訂 JSON 物件，將自訂屬性新增至 AWS OpsWorks Stacks 在執行個體上安裝的堆疊組態和部署屬性。如需詳細資訊，請參閱使用自訂 JSON及覆寫屬性。
5. 根據預設， AWS OpsWorks Stacks 會更新每個執行個體上的技術指南。若要指定更新哪些執行個體，請從頁面底端的清單選取適當的執行個體。若要選取 layer 中的每個執行個體，請在左欄選取適當的 layer 核取方塊。
6. 按一下更新自訂技術指南以安裝更新的技術指南。 AWS OpsWorks Stacks 會刪除指定執行個體上快取的自訂技術指南，並從儲存庫安裝新的技術指南。

> **Note**
>
> 僅現有執行個體需要進行此程序，其快取中含有舊版技術指南。如果您之後將執行個體新增至 layer， AWS OpsWorks Stacks 會部署目前在儲存庫中的技術指南，讓他們自動取得最新版本。

執行配方

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可使用兩種方式執行配方：

- 自動執行，方法是將配方指派給適當 layer 的生命週期事件。
- 手動執行，方法是執行執行配方堆疊命令或使用代理程式 CLI。

主題

- AWS OpsWorks Stacks 生命週期事件
- 自動執行配方
- 手動執行配方

AWS OpsWorks Stacks 生命週期事件

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

每個 layer 都有一組五個生命週期事件，而每個事件都有一組特別與該 layer 相關聯的配方。當事件在 layer 的執行個體上發生時， AWS OpsWorks Stacks 會自動執行該組適當的配方。若要提供這些事件的自訂回應，請實作自訂配方，並將其指派給每個 layer 的適當事件。 AWS OpsWorks Stacks 會在事件的內建配方之後執行這些配方。

Setup

此事件會在已啟動的執行個體完成開機後發生。您也可以使用設定堆疊命令手動觸發Setup事件。AWS OpsWorks Stacks 會執行配方，根據執行個體的 layer 設定執行個體。例如，如果執行個體是

Rails App Server layer 的成員，Setup配方會安裝 Apache、Ruby Enterprise Edition、 Passenger 和 Ruby on Rails。

> **Note**
>
> Setup (設定) 事件會導致執行個體服務中斷。由於執行個體在 Setup (設定) 生命週期事件執行時，並未處於 Online (線上) 狀態，因此會從負載平衡器移除您執行 Setup (設定) 事件的執行個體。

Configure

發生以下其中一種情況時，此事件會在堆疊的所有執行個體上發生：

- 執行個體進入或離開線上狀態。
- 您將彈性 IP 地址與執行個體建立關聯，或將某個彈性 IP 地址從執行個體解除關聯。
- 您可以將 Elastic Load Balancing 負載平衡器連接到 layer，或從 layer 分離。

例如，假設您的堆疊有執行個體 A、B 和 C，而且您啟動新的執行個體 D。在 D 完成執行其設定配方之後， AWS OpsWorks Stacks 會在 A、B、C 和 D 上觸發Configure事件。如果您隨後停止 A，AWS OpsWorks Stacks 會在 B、C 和 D 上觸發Configure事件。 AWS OpsWorks Stacks 會透過執行每個 layer 的Configure配方來回應Configure事件，以更新執行個體的組態以反映目前的線上執行個體集。Configure 事件是重新產生組態檔案的好機會。例如，HAProxy Configure配方會重新設定負載平衡器，以適應線上應用程式伺服器執行個體集中的任何變更。

您也可以使用設定堆疊命令來手動觸發設定事件。

Deploy

此事件會在您執行 Deploy (部署) 命令時發生，通常發生於將應用程式部署至一組應用程式伺服器執行個體的情況下。執行個體會執行將應用程式和任何相關檔案從其儲存庫部署到 layer 之執行個體的配方。例如，針對 Rails 應用程式伺服器執行個體，Deploy 配方會檢查指定的 Ruby 應用程式，並告知 Phusion Passenger 重新載入它。您也可以在其他執行個體上執行 Deploy，讓它們 (舉例來說) 可更新其組態以適應新部署的應用程式。

> **Note**
>
> 安裝包含部署，其會在安裝完成後執行部署配方。

Undeploy

此事件會在您刪除應用程式或執行 Undeploy 命令，藉此從一組應用程式伺服器執行個體中移除應用程式時發生。指定的執行個體會執行配方，以移除所有應用程式版本，並執行所有必要的清除。

Shutdown

此事件會在您指示 AWS OpsWorks Stacks 關閉執行個體，但在關聯的 Amazon EC2 執行個體實際終止之前發生。 AWS OpsWorks Stacks 會執行配方來執行清除任務，例如關閉服務。

如果您已將 Elastic Load Balancing 負載平衡器連接至 layer，並啟用連線耗盡的支援，則 AWS OpsWorks Stacks 會等到連線耗盡完成，再觸發Shutdown事件。

觸發Shutdown事件後， AWS OpsWorks Stacks 允許Shutdown配方有指定的時間執行其任務，然後停止或終止 Amazon EC2 執行個體。預設 Shutdown 逾時值為 120 秒。若您的 Shutdown 配方可能需要更多時間，您可以編輯 layer 組態來變更逾時值。如需執行個體 Shutdown 的詳細資訊，請參閱停止執行個體。

> **Note**
>
> 重新啟動執行個體不會觸發任何生命週期事件。

如需更多有關 Deploy 和 Undeploy 應用程式命令的討論，請參閱部署應用程式。

在已啟動的執行個體完成開機後，其餘的啟動順序如下：

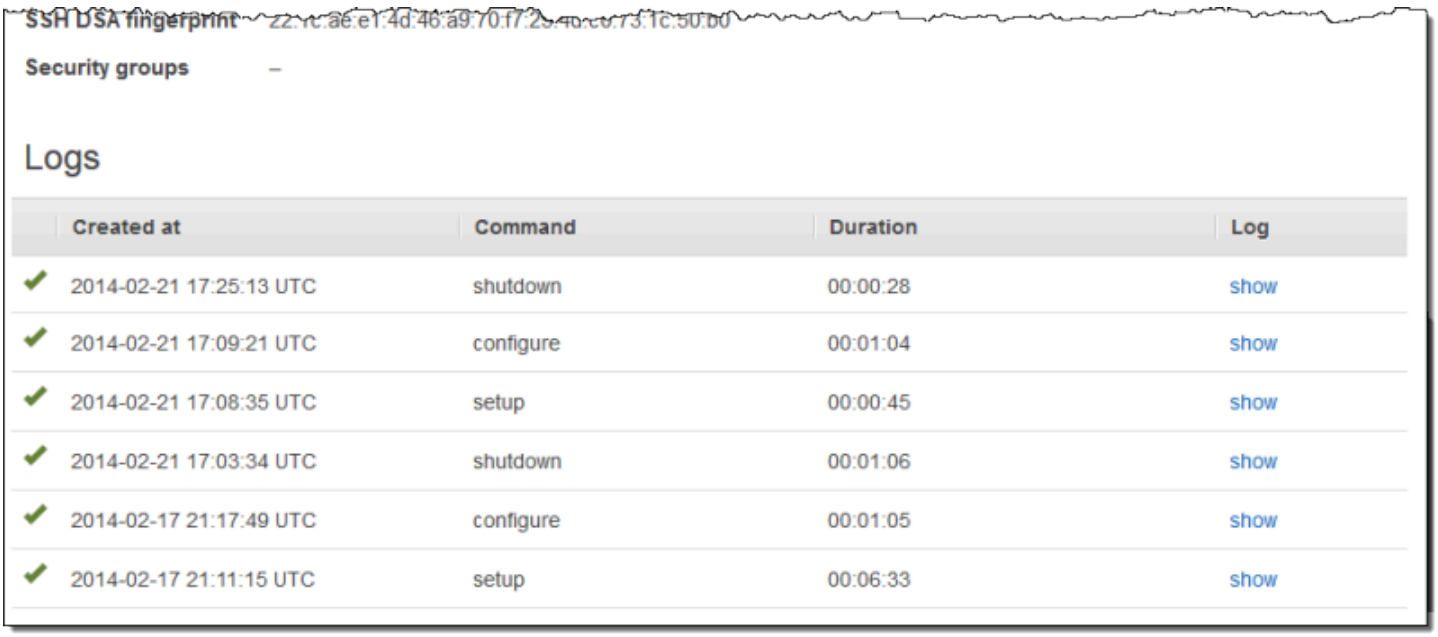
1. AWS OpsWorks Stacks 會執行執行個體的內建Setup配方，後面接著任何自訂Setup配方。
2. AWS OpsWorks Stacks 會執行執行個體的內建Deploy配方，後面接著任何自訂Deploy配方。

   執行個體現在已上線。
3. AWS OpsWorks Stacks 會在堆疊中的所有執行個體上觸發Configure事件，包括新啟動的執行個體。

   AWS OpsWorks Stacks 會執行執行個體的內建Configure配方，後面接著任何自訂Configure配方。

> **Note**
>
> 若要查看發生在特定執行個體上的生命週期事件，請前往 Instances (執行個體) 頁面，然後按一下該執行個體的名稱以開啟其詳細資訊頁面。事件清單位於頁面底部的 Logs (日誌) 區段。

您可以按一下日誌欄中的顯示，以檢查事件的 Chef 日誌。該日誌會提供事件處理方式的詳細資訊，包括哪些配方已執行。如需如何解讀 Chef 日誌的詳細資訊，請參閱 Chef 日誌。



對於每個生命週期事件， AWS OpsWorks Stacks 會在每個執行個體上安裝一組堆疊組態和部署屬性，其中包含目前堆疊狀態，以及Deploy事件的部署相關資訊。這些屬性包括哪些執行個體可用、其 IP 地址等資訊。如需詳細資訊，請參閱堆疊組態及部署屬性。

> **Note**
>
> 同時啟動或停止大量的執行個體，可能會快速產生大量的 Configure 事件。為了避免不必要的處理， AWS OpsWorks Stacks 只會回應最後一個事件。該事件的堆疊組態和部署屬性包含為堆疊之執行個體更新整組變更所需的所有資訊。這樣就不需要處理先前的Configure事件。AWS OpsWorks Stacks 會將未處理Configure的事件標記為已取代。

自動執行配方

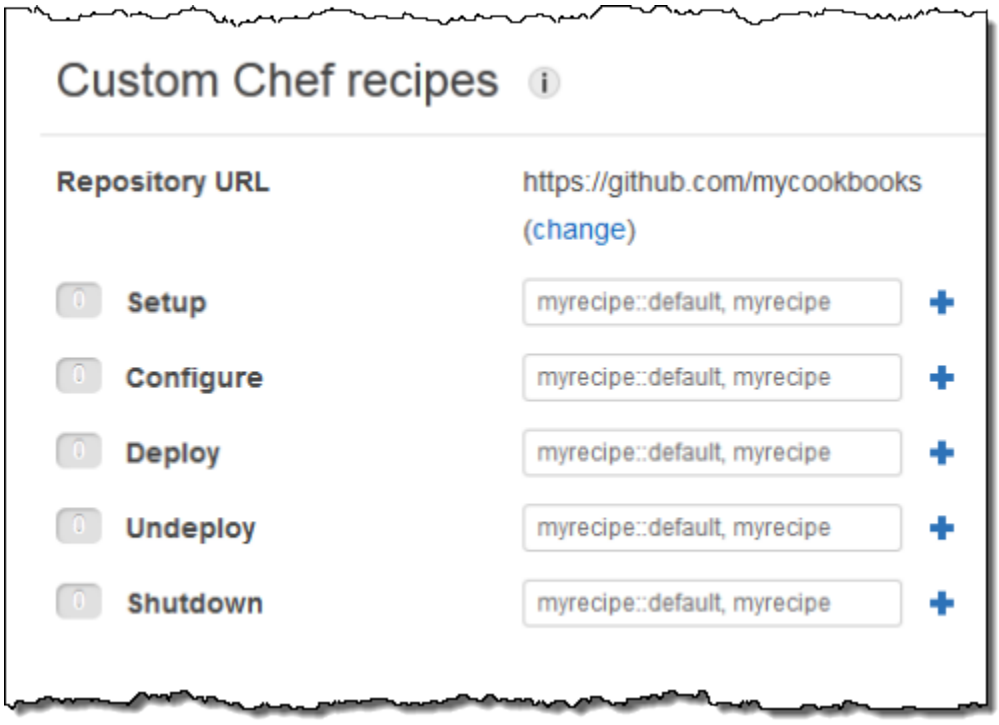
> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

每個 layer 都各有一組指派給每個生命週期事件的內建配方，不過部分 layer 缺少解除部署配方。當 執行個體上發生生命週期事件時， AWS OpsWorks Stacks 會為相關聯的 layer 執行適當的配方集。

如果您已安裝自訂技術指南，您可以將每個配方指派給 layer 的生命週期事件，讓 AWS OpsWorks Stacks 自動執行部分或全部配方。事件發生後， AWS OpsWorks Stacks 會在 layer 的內建配方之後執行指定的自訂配方。

將自訂配方指派給 layer 事件

1. 在 Layers 頁面上，對適當的 layer 按一下 Recipes (配方)，然後按一下 Edit (編輯)。如果您尚未啟用自訂技術指南，請按一下設定技術指南，以開啟堆疊的 Settings (設定) 頁面。將 Use custom Chef Cookbooks (使用自訂 Chef 技術指南) 切換到 Yes (是)，並提供技術指南的儲存庫資訊。然後按一下 Save (儲存)，並導覽回 Recipes (配方) 標籤的編輯頁面。如需詳細資訊，請參閱安裝自訂技術指南。
2. 在 Recipes (配方) 標籤上，在適當的事件欄位中輸入每個自訂配方，並按一下 + 將其新增至清單。配方的指定如下：*cookbook*::*somerecipe* (省略 `.rb` 副檔名)。



當您啟動新的執行個體時， AWS OpsWorks Stacks 會在執行標準配方之後，為每個事件自動執行自訂配方。

> **Note**
>
> 自訂配方會按照您在主控台中輸入的順序執行。控制執行順序的另一個方法，是實作按正確順序執行配方的中繼配方。

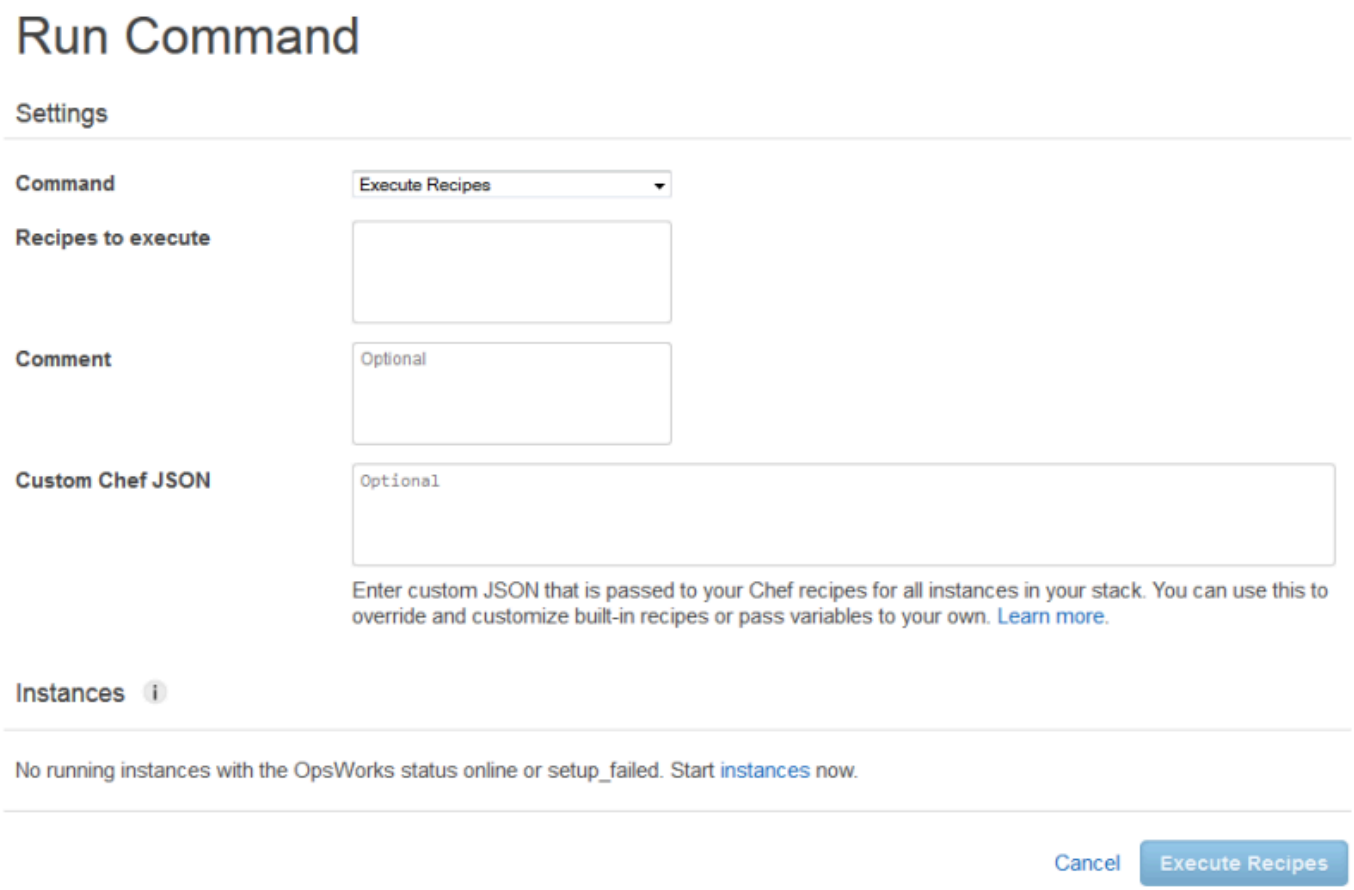
手動執行配方

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

雖然配方通常會自動執行以回應生命週期事件，您仍可隨時在任何或所有堆疊執行個體上手動執行配方。這項功能通常用於未良好映射到生命週期事件的任務，例如備份執行個體。若要手動執行自訂配方，該配方必須位於您其中一個自訂技術指南中，但不需要指派給生命週期事件。當您手動執行配方時， AWS OpsWorks Stacks 會安裝與部署事件相同的`deploy`屬性。

若要在堆疊執行個體上手動執行配方

1. 在 Stack (堆疊) 頁面上，按一下 Run command (執行命令)。針對 Command (命令)，選取 Execute Recipes (執行配方)。



2. 使用標準`######`：：*recipename* 格式，在執行方塊的配方中輸入要執行的配方。使用逗號分隔多個配方，這些配方將按您列出的順序執行。
3. 您可以選擇性使用 Custom Chef JSON (自訂 Chef JSON) 方塊來新增自訂 JSON 物件，以定義要合併到安裝在執行個體上之堆疊組態和部署屬性的自訂屬性。如需使用自訂 JSON 物件的詳細資訊，請參閱使用自訂 JSON 和覆寫屬性。
4. 在執行個體下，選取 Stacks AWS OpsWorks 應執行配方的執行個體。

當生命週期事件發生時， Stacks AWS OpsWorks 代理程式會收到執行相關聯配方的命令。您可以在特定執行個體上手動執行這些命令，方法為使用適當的堆疊命令或使用代理程式 CLI 的 run_command 命令。如需如何使用代理程式 CLI 的詳細資訊，請參閱 AWS OpsWorks Stacks 代理程式 CLI。

資源管理

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

資源頁面可讓您在 Stacks 堆疊中使用帳戶的彈性 IP 地址、Amazon EBS 磁碟區或 Amazon RDS AWS OpsWorks 執行個體資源。您可使用 Resources (資源) 執行下列作業：

- 使用讓您將資源連接到其中一個堆疊執行個體的堆疊註冊資源。
- 連接資源到其中一個堆疊的執行個體。
- 在執行個體之間移動資源。
- 從執行個體分離資源。資源維持註冊狀態，並可連接到另一個執行個體。
- 取消註冊資源。未註冊的資源無法供 AWS OpsWorks Stacks 使用，但除非您刪除，否則仍會保留在您的 帳戶中，並且可以使用另一個堆疊註冊。

請注意下列限制條件：

- 您無法將已註冊的 Amazon EBS 磁碟區連接至 Windows 執行個體。
- 資源頁面會管理標準、PIOPS、輸送量最佳化 HDD、冷 HDD 或一般用途 (SSD) Amazon EBS 磁碟區，但不管理 RAID 陣列。
- Amazon EBS 磁碟區必須是 xfs 格式。

  AWS OpsWorks Stacks 不支援其他檔案格式，例如 ext4。如需準備 Amazon EBS 磁碟區的詳細資訊，請參閱讓 Amazon EBS 磁碟區可供使用。
- 您無法將 Amazon EBS 磁碟區連接至執行中的執行個體，也無法將其從中分離。

  您只能在離線執行個體上操作。例如，您可以向堆疊註冊使用中的磁碟區，然後將它連接到離線執行個體，但您先必須停止原執行個體並分離磁碟區，再啟動新的執行個體。否則，啟動程序會失敗。
- 所有已註冊的資源都會僅管理 AWS OpsWorks。這可以覆寫資源生命週期屬性，例如適用於 EC2 磁碟區的 `DeleteOnTermination`。
- 您可以連接和分離彈性 IP 地址與執行中的執行個體。

您可以在線上或離線執行個體上操作。例如，您可以註冊使用中的地址，並將其指派給執行中的執行個體，而 AWS OpsWorks Stacks 會自動重新指派該地址。

- 若要註冊和取消註冊資源，您的 IAM 政策必須授予下列動作的許可：

Amazon EBS 磁碟區	彈性 IP 地址	Amazon RDS 執行個體
RegisterVolume	RegisterElasticIp	RegisterRdsDbInstance
UpdateVolume	UpdateElasticIp	UpdateRdsDbInstance
DeregisterVolume	DeregisterElasticIp	DeregisterRdsDbInstance

管理許可等級授予所有這些動作的許可。為避免管理使用者註冊或取消註冊特定資源，請編輯他們的 IAM 政策以拒絕適當動作的許可。如需詳細資訊，請參閱安全與許可。

主題

- 向堆疊註冊資源
- 連接與移動資源
- 分離資源
- 取消註冊資源

向堆疊註冊資源

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Amazon EBS 磁碟區或彈性 IP 地址必須先向堆疊註冊，才能將其連接至執行個體。當 AWS OpsWorks Stacks 為堆疊建立資源時，它們會自動向該堆疊註冊。如果您想要使用外部建立的資源，您必須明確註冊它們。注意下列事項：

- 您一次只能向一個堆疊註冊資源。

- 當您刪除堆疊時，AWS OpsWorks Stacks 會取消註冊所有資源。

主題

- 使用 Stack 註冊 Amazon EBS 磁碟區
- 向堆疊註冊彈性 IP 地址
- 使用 Stack 註冊 Amazon RDS 執行個體

使用 Stack 註冊 Amazon EBS 磁碟區

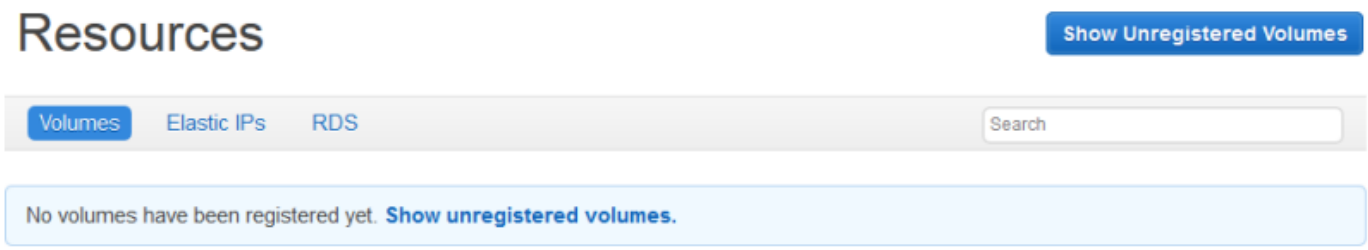
> **Note**
>
> 此資源只搭配 Linux 堆疊使用。雖然您可以使用 Windows 堆疊註冊 Amazon EBS 磁碟區，但無法將其連接至執行個體。

您可以使用 資源頁面向堆疊註冊 Amazon EBS 磁碟區，但需受下列限制條件約束：

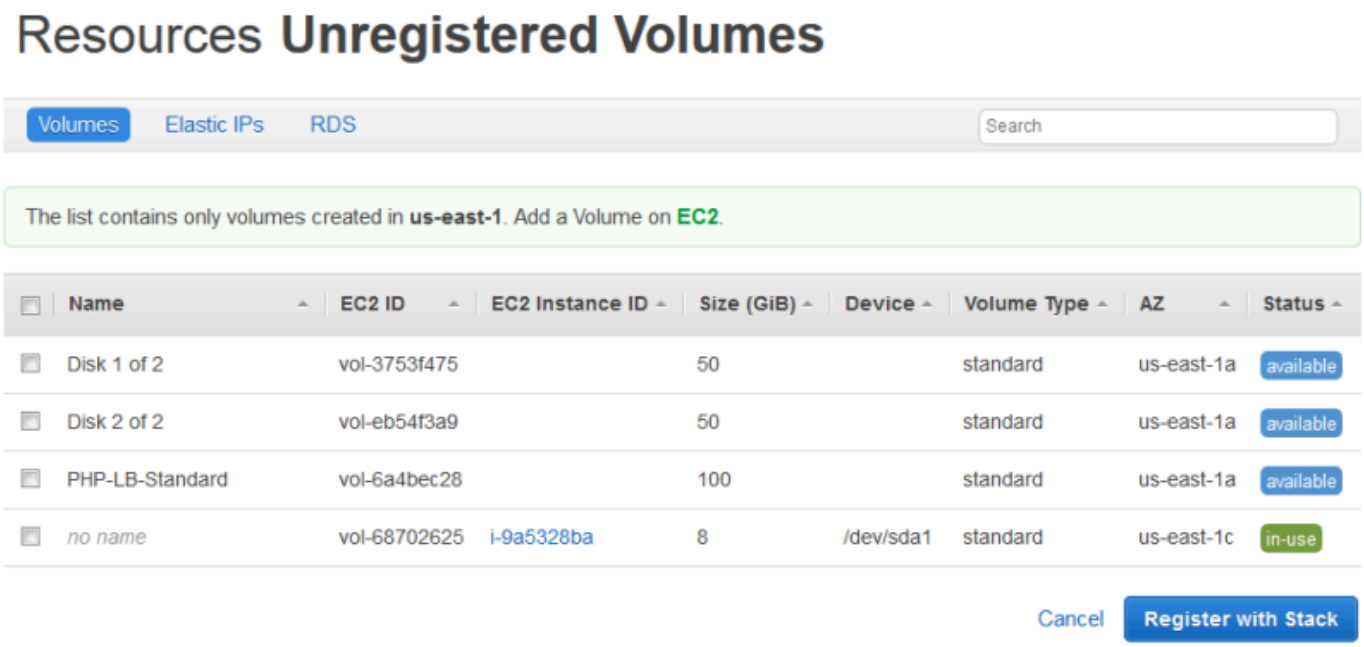
- 連接的非根 Amazon EBS 磁碟區必須是標準、輸送量最佳化 HDD、冷 HDD、PIOPS 或一般用途 (SSD)，但不是 RAID 陣列。如需磁碟區大小之上限和下限的資訊，請參閱本指南的EBS 磁碟區。
- 磁碟區必須是 XFS 格式。
- AWS OpsWorks Stacks 不支援非根 Amazon EBS 磁碟區的其他檔案格式，例如第四個擴充檔案系統 (ext4)。如需準備 Amazon EBS 磁碟區的詳細資訊，請參閱讓 Amazon EBS 磁碟區可供使用。請注意，該主題中的範例說明如何建立 ext4 型磁碟區，但您可以遵循相同的步驟建立 XFS 型磁碟區。

註冊 Amazon EBS 磁碟區

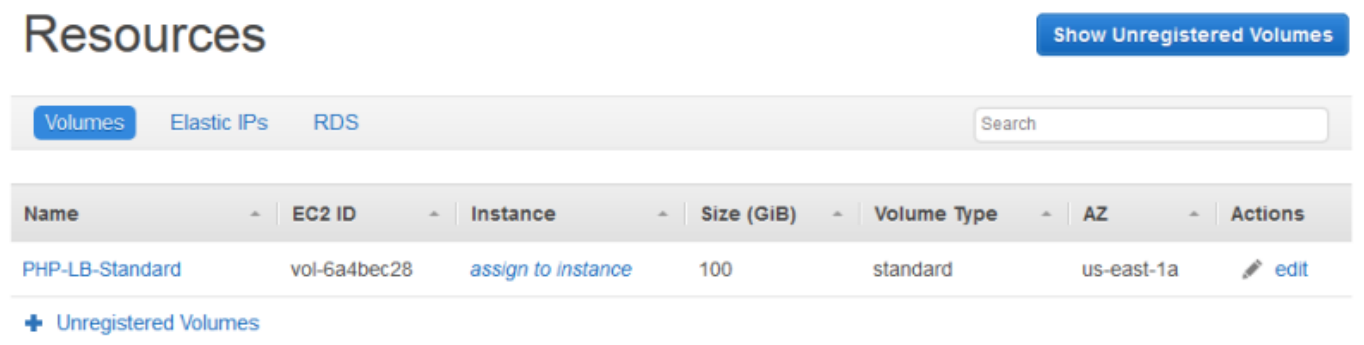
1. 開啟所需的堆疊，然後按一下導覽窗格中的 Resources (資源)。
2. 按一下磁碟區以顯示可用的 Amazon EBS 磁碟區。堆疊一開始沒有任何註冊的磁碟區，如下圖所示。



3. 按一下顯示未註冊的磁碟區，以顯示您帳戶中位於堆疊區域中的 Amazon EBS 磁碟區，以及適用的堆疊 VPC。Status (狀態) 欄指示磁碟區可供使用。Volume Type (磁碟區類型) 指出磁碟區是標準 (`standard`)、一般用途 SSD (`gp2`)、PIOPS (`io1`，後面跟著以括號括住的每個磁碟值 IOPS)、輸送量最佳化 HDD (`st1`) 或 Cold HDD (`sc1`)。



4. 選取適當的磁碟區，然後按一下 Register to Stack (向堆疊註冊)。Resources (資源) 頁面現可列出新註冊的磁碟區。



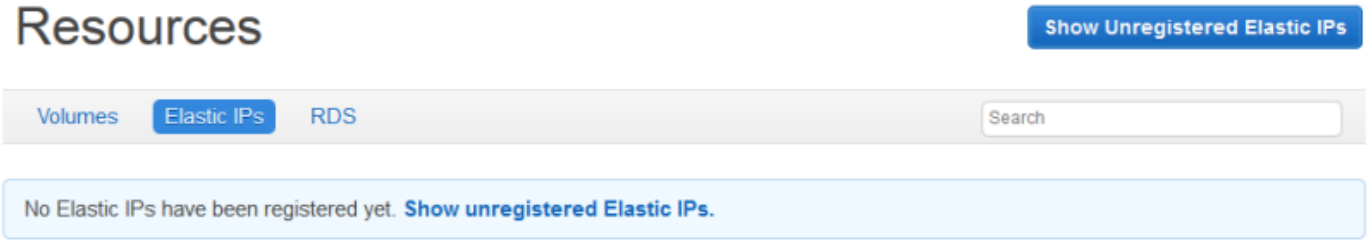
若要註冊其他磁碟區，請按一下 Show Unregistered Volumes (顯示未註冊的磁碟區) 或 + Unregistered Volumes (+ 未註冊的磁碟區)，並重複此程序。

向堆疊註冊彈性 IP 地址

使用下列程序註冊彈性 IP 地址。

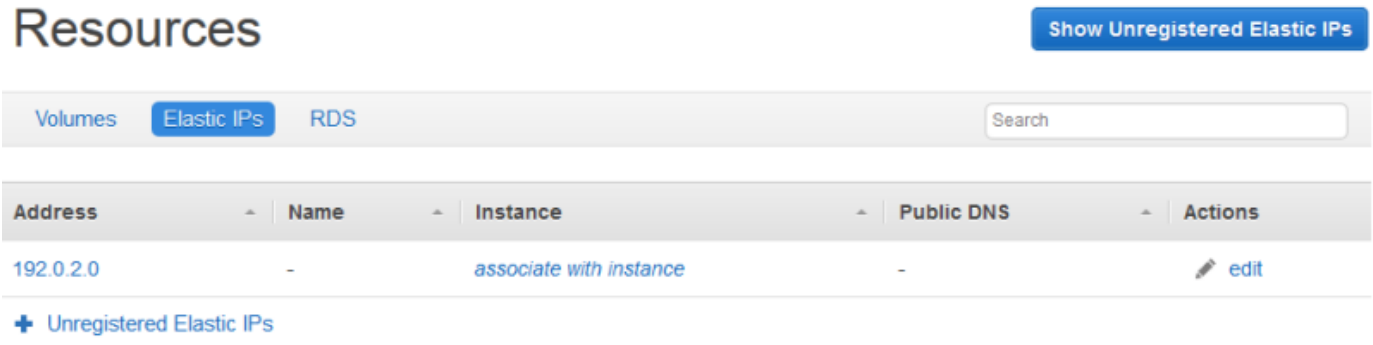
註冊彈性 IP 地址

1. 開啟堆疊的 Resources (資源) 頁面，然後按一下 Elastic IPs (彈性 IP) 顯示可用的彈性 IP 地址。堆疊一開始沒有任何註冊的地址，如下圖所示。



2. 按一下 Show Unregistered Elastic IPs (顯示未註冊的彈性 IP) 顯示您帳戶中該堆疊區域的可用彈性 IP 地址。

3. 選取適當的地址，然後按一下 Register to Stack (向堆疊註冊)。這會讓您回到 Resources (資源) 頁面，現可列出新註冊的地址。



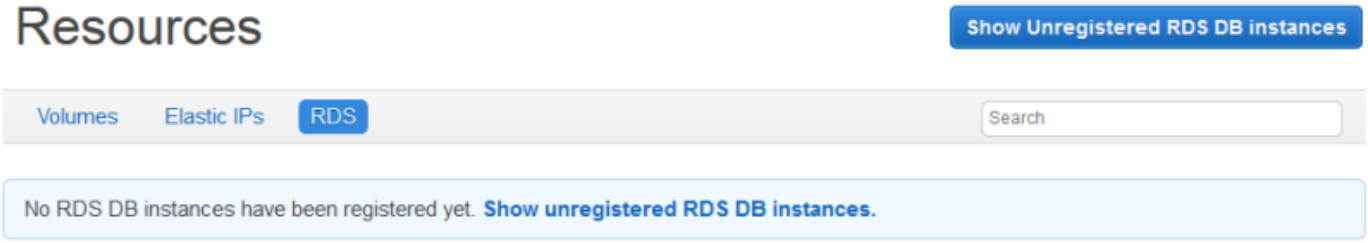
若要註冊其他地址，請按一下 Show Unregistered Elastic IPs (顯示未註冊的彈性 IP) 或 + Unregistered Elastic IPs (+ 未註冊的彈性 IP)，並重複此程序。

使用 Stack 註冊 Amazon RDS 執行個體

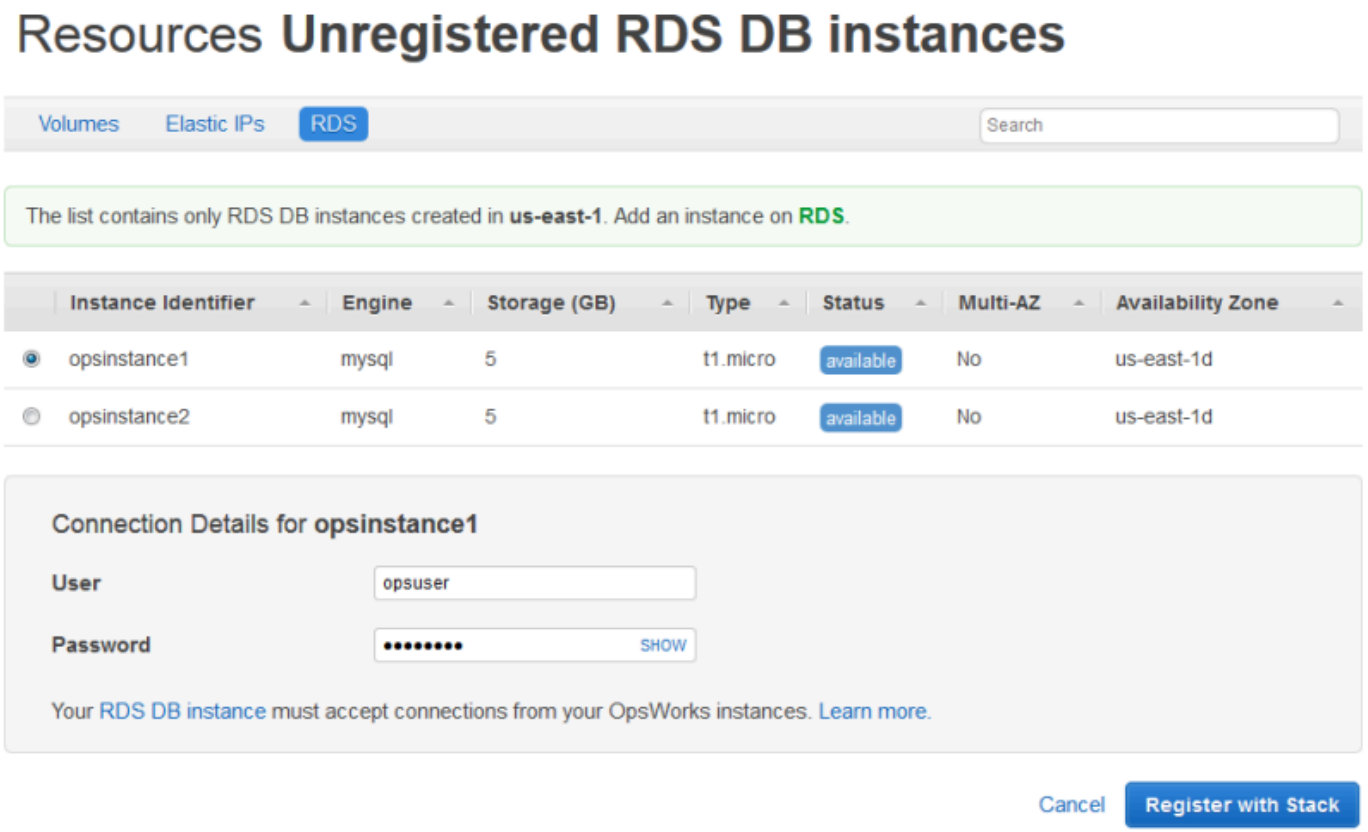
使用下列程序註冊 Amazon RDS 執行個體。

註冊 Amazon RDS 執行個體

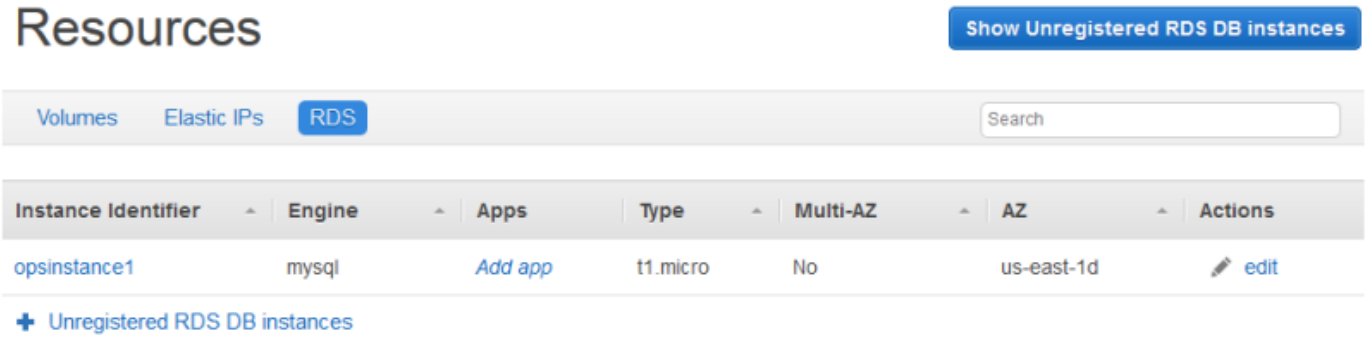
1. 開啟堆疊的資源頁面，然後按一下 RDS 以顯示可用的 Amazon RDS 執行個體。堆疊一開始沒有任何註冊的執行個體，如下圖所示。



2. 按一下顯示未註冊的 RDS 資料庫執行個體，以顯示您帳戶中位於堆疊區域中的可用 Amazon RDS 執行個體。



3. 選取適當的執行個體，針對 User (使用者) 和 Password (密碼) 輸入其主要使用者和主密碼值，然後按一下 Register to Stack (向堆疊註冊)。這會讓您回到 Resources (資源) 頁面，現可會註冊的執行個體。



> **Important**
>
> 您必須確保您用來註冊 Amazon RDS 執行個體的使用者和密碼對應至有效的使用者和密碼。否則，您的應用程式將無法連線至執行個體。

若要註冊其他地址，請按一下 Show Unregistered RDS DB instances (顯示未註冊的 RDS 資料庫執行個體) 或 + Unregistered RDS DB instances (+ 未註冊的 RDS 資料庫執行個體)，並重複此程序。如需如何搭配 Stacks 使用 Amazon RDS AWS OpsWorks 執行個體的詳細資訊，請參閱Amazon RDS 服務層。

> **Note**
>
> 您也可以透過 Layers 頁面註冊 Amazon RDS 執行個體。如需詳細資訊，請參閱Amazon RDS 服務層。

連接與移動資源

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

在您向堆疊註冊資源之後，您就可以將它連接到堆疊的其中一個執行個體。您也可以將已連接的資源從某個執行個體移動到另一個。注意下列事項：

- 當您連接或移動 Amazon EBS 磁碟區時，操作中涉及的執行個體必須離線。如果您有興趣的執行個體不在 Resources (資源) 頁面中，請移至 Instances (執行個體) 頁面停止執行個體。停止後，您可以返回 Resources (資源) 頁面並連接或移動資源。
- 當您連接或移動彈性 IP 地址時，執行個體可以為線上或離線。
- 如果您刪除執行個體，任何連接的資源都保持向堆疊註冊。然後，您可以將資源連接到另一個執行個體；或者，如果您不再需要它，請取消註冊此資源。

主題

- 將 Amazon EBS 磁碟區指派給執行個體
- 建立彈性 IP 地址與執行個體的關聯
- 將 Amazon RDS 執行個體連接至應用程式

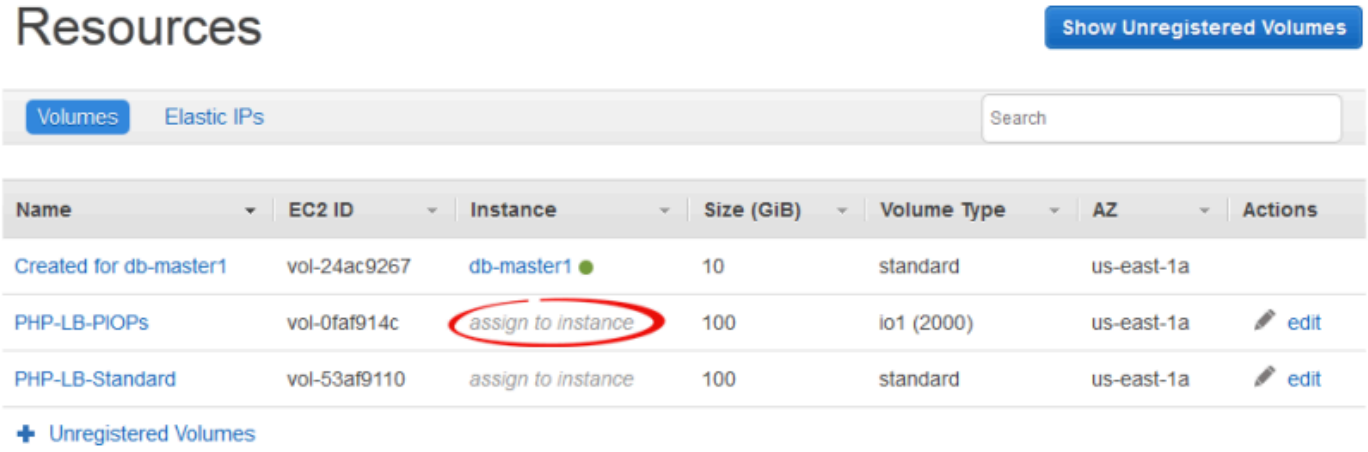
將 Amazon EBS 磁碟區指派給執行個體

> **Note**
>
> 您無法將 Amazon EBS 磁碟區指派給 Windows 執行個體。

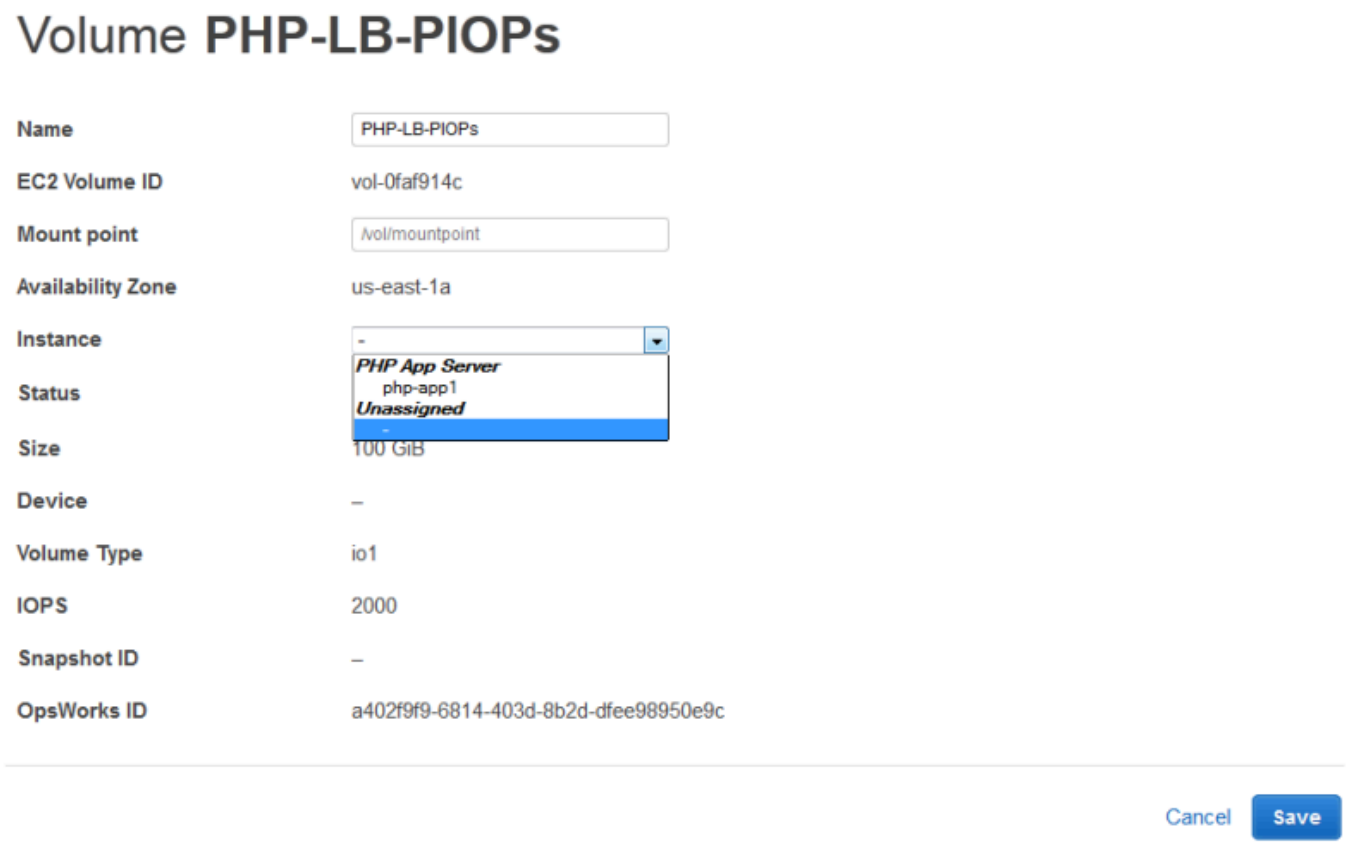
您可以將已註冊的 Amazon EBS 磁碟區指派給執行個體，並將其從一個執行個體移至另一個執行個體，但兩個執行個體都必須離線。

將 Amazon EBS 磁碟區指派給執行個體

1. 在 Resources (資源) 頁面，在適當磁碟區的 Instance (執行個體) 欄中按一下 assign to instance (指派給執行個體)。



2. 在磁碟區的 details (詳細資訊) 頁面，選取適當的執行個體，指定磁碟區的名稱和掛載點，然後按一下 Save (儲存) 將磁碟區連接到執行個體。



> **Important**
>
> 如果您已將外部使用中磁碟區指派給執行個體，則必須使用 Amazon EC2 主控台、API 或 CLI 從原始執行個體取消指派，否則啟動程序將會失敗。

您也可以使用詳細資訊頁面，將指派的 Amazon EBS 磁碟區移至堆疊中的另一個執行個體。

將 Amazon EBS 磁碟區移至另一個執行個體

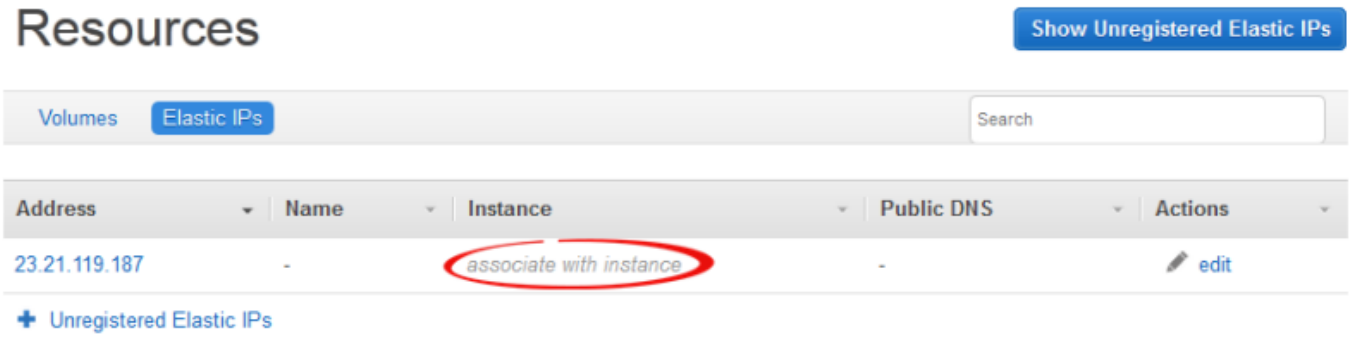
1. 確定兩個執行個體都是離線狀態。
2. 在 Resources (資源) 頁面中按一下 Volumes (磁碟區)，然後在磁碟區的 Actions (動作) 欄中按一下 edit (編輯)。
3. 執行以下任意一項：
   - 若要將磁碟區移到堆疊的另一個執行個體，請從 Instance (執行個體) 清單中選取適當的執行個體，然後按一下 Save (儲存)。
   - 若要將磁碟區移至另一個堆疊的執行個體，請取消註冊磁碟區、向新的堆疊註冊磁碟區，然後連接到新的執行個體。

建立彈性 IP 地址與執行個體的關聯

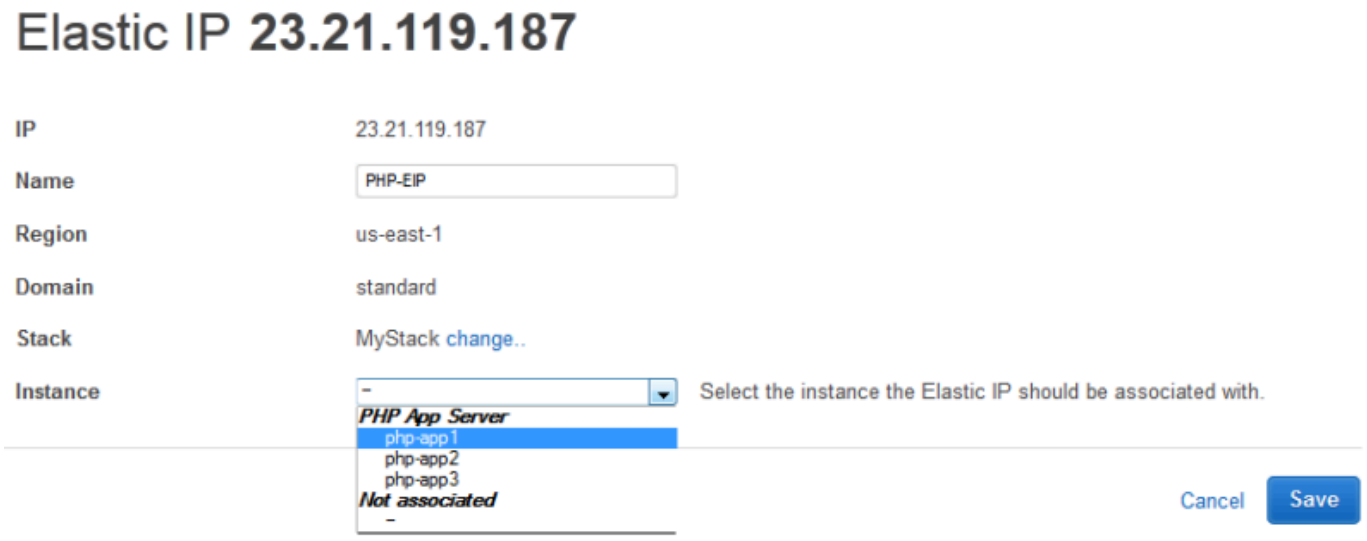
您可以建立已註冊彈性 IP 地址與執行個體的關聯，然後將它從一個執行個體移到另一個執行個體，包括其他堆疊中的執行個體。這些執行個體可以是線上或離線。

建立彈性 IP 位址與執行個體的關聯

1. 在 Resources (資源) 頁面，在適當地址的 Instance (執行個體) 欄中按一下 associate with instance (與執行個體建立關聯)。



2. 在地址的 details (詳細資訊) 頁面，選取適當的執行個體，指定地址的名稱，然後按一下 Save (儲存) 建立地址與執行個體的關聯。

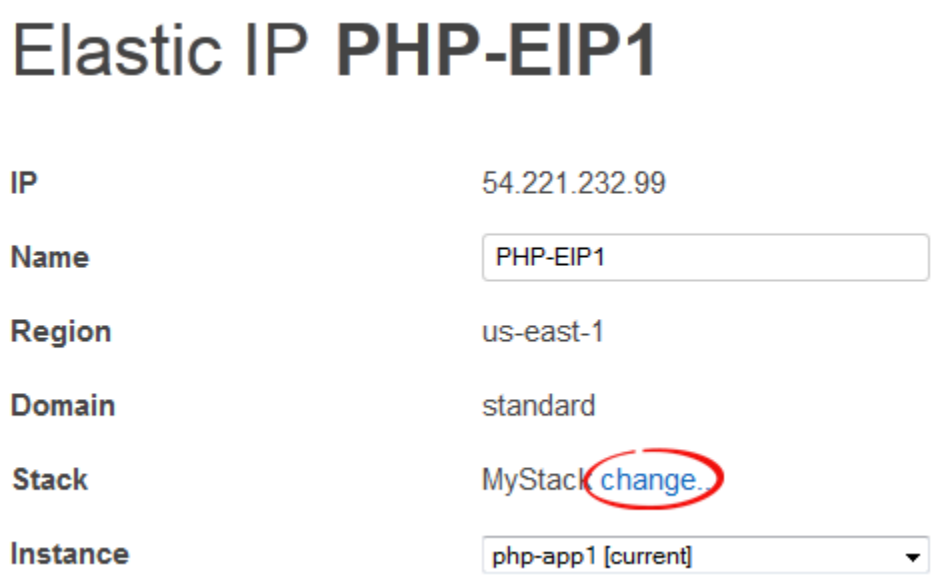


> **Note**
>
> 如果彈性 IP 地址目前與另一個線上執行個體相關聯， AWS OpsWorks Stacks 會自動將地址重新指派給新的執行個體。

您也可以使用 details (詳細資訊) 頁面將相關聯彈性 IP 地址移到另一個執行個體。

將彈性 IP 地址移到另一個執行個體

1. 在 Resources (資源) 頁面中按一下 Elastic IPs (彈性 IP)，然後在地址的 Actions (動作) 欄中按一下 edit (編輯)。
2. 執行以下任意一項：
   - 若要將地址移到堆疊的另一個執行個體，請從 Instance (執行個體) 清單中選取適當的執行個體，然後按一下 Save (儲存)。
   - 若要將地址移至另一個堆疊中的執行個體，請按一下堆疊設定中的變更，以查看可用堆疊的清單。從 Stack (堆疊) 清單中選取堆疊，從 Instance (執行個體) 清單中選取執行個體。然後按一下 Save (儲存)。



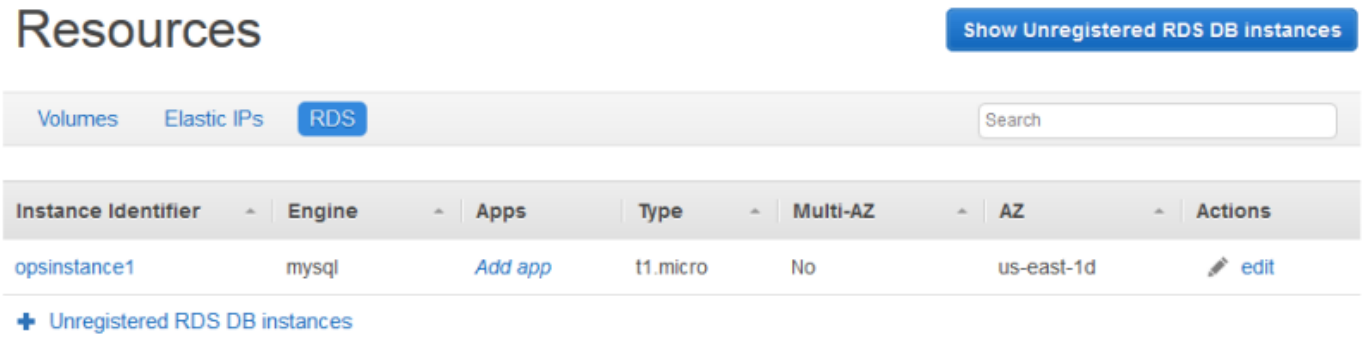
在您連接或移動地址之後，AWS OpsWorks Stacks 會觸發設定生命週期事件，以通知堆疊的執行個體變更。

將 Amazon RDS 執行個體連接至應用程式

您可以將 Amazon RDS 執行個體連接至一或多個應用程式。

將 Amazon RDS 執行個體連接至應用程式

1. 在 Resources (資源) 頁面，在適當執行個體的 Apps (應用程式) 欄中按一下 Add app (新增應用程式)。



2. 使用新增應用程式頁面來連接 Amazon RDS 執行個體。如需詳細資訊，請參閱新增應用程式。

由於 Amazon RDS 可以連接到多個應用程式，因此沒有將執行個體從一個應用程式移動到另一個應用程式的特殊程序。只要編輯第一個應用程式移除 RDS 執行個體，或編輯第二個應用程式新增 RDS 執行個體即可。如需詳細資訊，請參閱編輯應用程式。

分離資源

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

當您不再需要連接的資源時，您可以分離它。此資源會保持向堆疊註冊，並可連接到其他位置。

主題

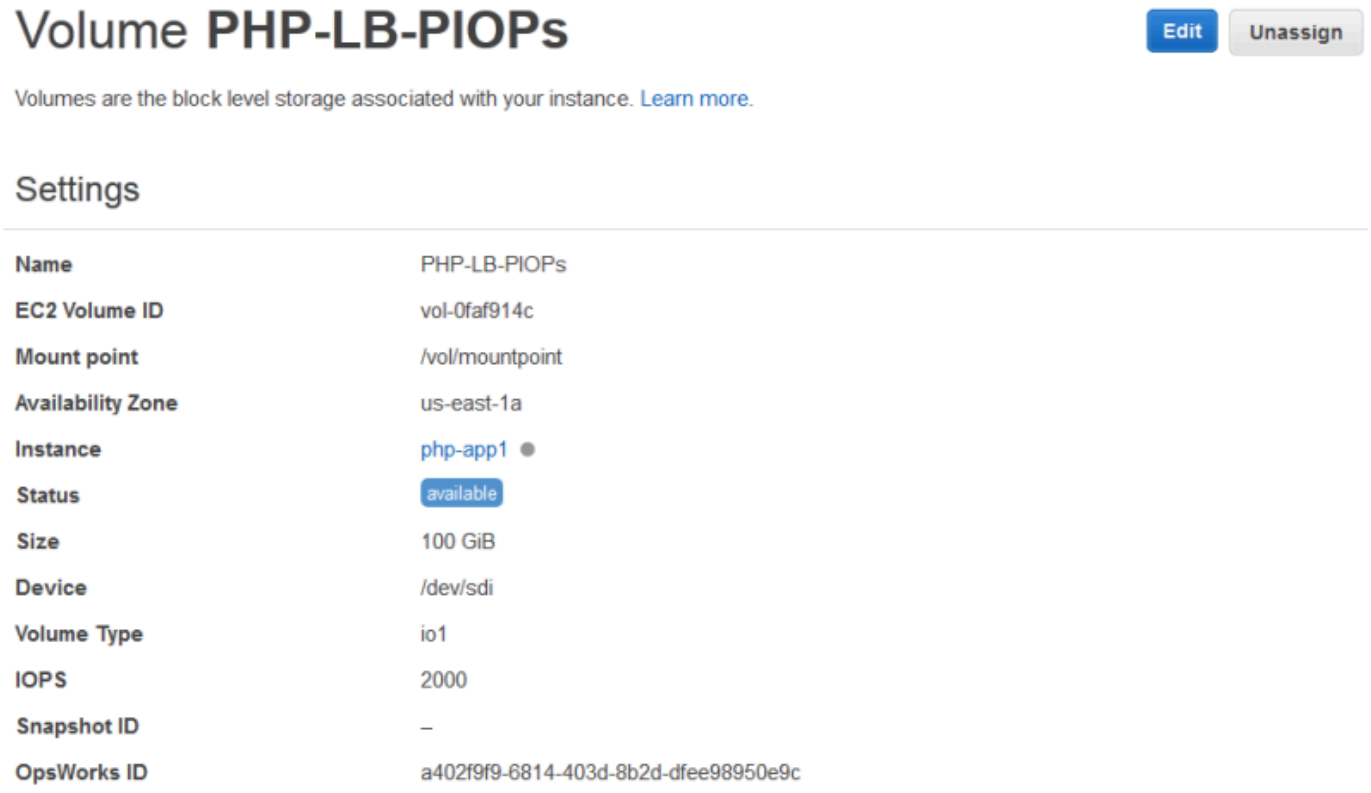
- 取消指派 Amazon EBS 磁碟區
- 取消與彈性 IP 地址的關聯
- 分離 Amazon RDS 執行個體

取消指派 Amazon EBS 磁碟區

使用下列程序從執行個體取消指派 Amazon EBS 磁碟區。

取消指派 Amazon EBS 磁碟區

1. 確定此執行個體為離線狀態。
2. 在 Resources (資源) 頁面中按一下 Volumes (磁碟區)，然後按一下磁碟區名稱。
3. 在磁碟區的 details (詳細資訊) 頁面，按一下 Unassign (取消指派)。

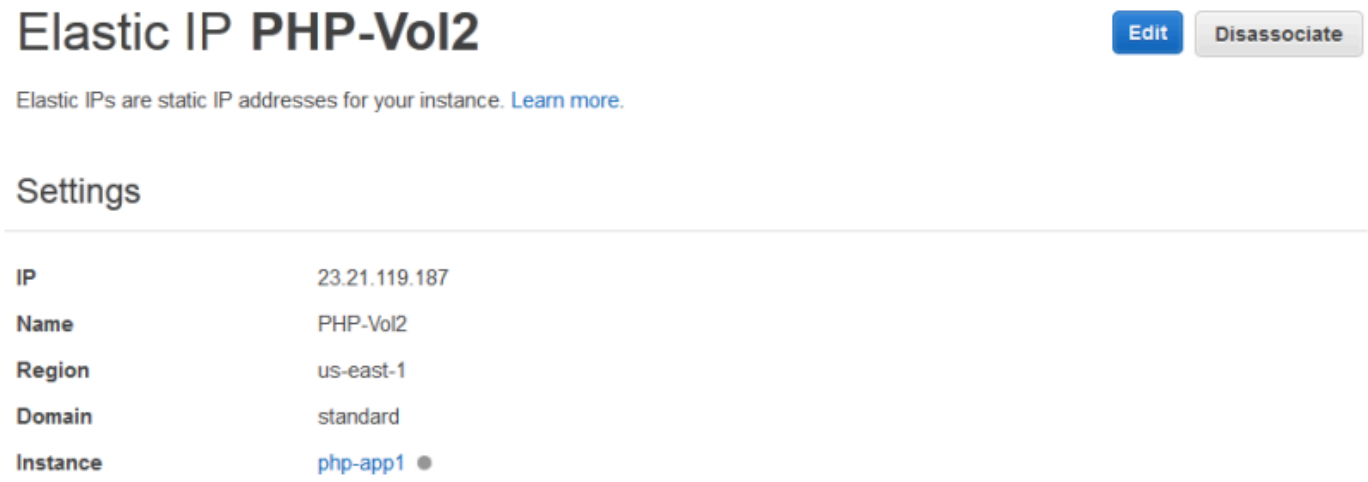


取消與彈性 IP 地址的關聯

使用下列程序從其執行個體取消與彈性 IP 地址的關聯。

取消與彈性 IP 位址的關聯

1. 在 Resources (資源) 頁面中按一下 Elastic IPs (彈性 IP)，然後在地址的 Actions (動作) 欄中按一下 edit (編輯)。
2. 在地址的 details (詳細資訊) 頁面，按一下 Disassociate (取消關聯)。



取消與地址的關聯之後， AWS OpsWorks Stacks 會觸發設定生命週期事件，以通知堆疊的執行個體變更。

分離 Amazon RDS 執行個體

使用下列程序將 Amazon RDS 從應用程式分離。

分離 Amazon RDS 執行個體

1. 在 Resources (資源) 頁面中按一下 RDS，然後在 Apps (應用程式) 欄中按一下適當的應用程式。
2. 按一下 Edit (編輯)，編輯應用程式組態以分離執行個體。如需詳細資訊，請參閱編輯應用程式。

> **Note**
>
> 此程序會從單一應用程式分離 Amazon RDS。如果執行個體連接到多個應用程式，您必須為每個應用程式重複此程序。

取消註冊資源

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

如果您不再需要向堆疊註冊資源，您可以取消註冊。取消註冊不會從您的帳戶刪除資源；資源會保留在那裡，並且可以使用另一個堆疊註冊或在 Stacks AWS OpsWorks 外部使用。如果您要刪除整個資源，您有兩個選擇：

- 如果彈性 IP 或 Amazon EBS 資源已連接至執行個體，您可以在刪除執行個體時刪除資源。

  移至 Instances (執行個體) 頁面，按一下執行個體 Actions (動作) 欄中的 delete (刪除)，然後選取 Delete instance's EBS volumes (刪除執行個體的 EBS 磁碟區) 或 Delete the instance's Elastic IP (刪除執行個體的彈性 IP)。
- 取消註冊資源，然後使用 Amazon EC2 或 Amazon RDS 主控台、API 或 CLI 將其刪除。

主題

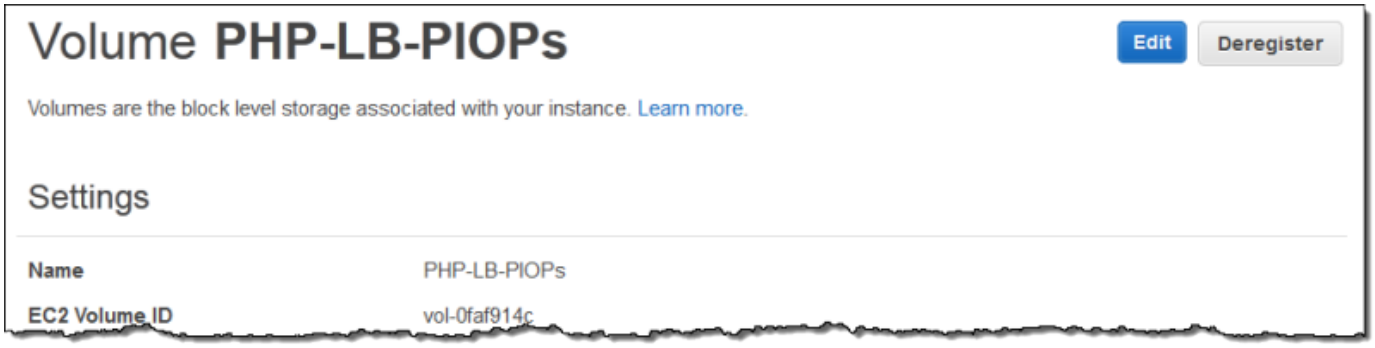
- 取消註冊 Amazon EBS 磁碟區
- 取消註冊彈性 IP 地址
- 取消註冊 Amazon RDS 執行個體

取消註冊 Amazon EBS 磁碟區

使用下列程序取消註冊 Amazon EBS 磁碟區。

取消註冊 Amazon EBS 磁碟區

1. 如果磁碟區已連接到執行個體，請如取消指派 Amazon EBS 磁碟區中所述取消指派它。
2. 在 Resources (資源) 頁面，按一下 Name (名稱) 欄的磁碟區名稱。
3. 在磁碟區的 details (詳細資訊) 頁面，按一下 Deregister (取消註冊)。

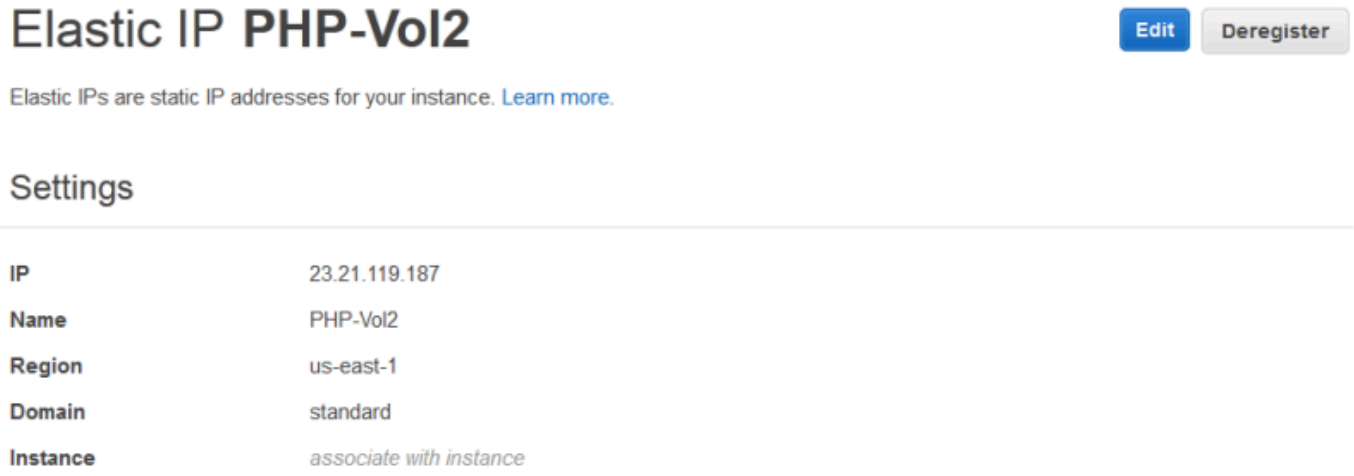


取消註冊彈性 IP 地址

使用下列程序取消註冊彈性 IP 地址。

取消註冊彈性 IP 地址

1. 如果地址與執行個體相關聯，請如取消與彈性 IP 地址的關聯中所述取消關聯。
2. 在 Resources (資源) 頁面中按一下 Elastic IPs (彈性 IP)，然後按一下 Address (地址) 欄中的 IP 地址。
3. 在地址的 details (詳細資訊) 頁面，按一下 Deregister (取消註冊)。



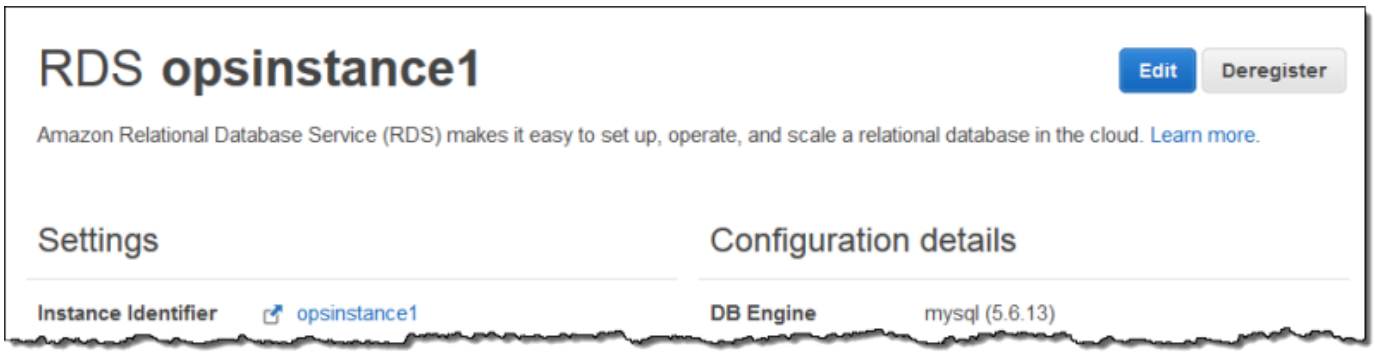
> **Note**
>
> 如果您只想向不同的堆疊註冊 Elastic IP 地址，您必須從它目前的堆疊取消註冊，然後向新的堆疊註冊。不過，您可以將已連接的彈性 IP 地址直接移到另一個堆疊的執行個體。如需詳細資訊，請參閱連接與移動資源。

取消註冊 Amazon RDS 執行個體

使用下列程序取消註冊 Amazon RDS 執行個體。

取消註冊 Amazon RDS 執行個體

1. 如果執行個體與應用程式相關聯，請如分離資源中所述分離它。
2. 在 Resources (資源) 頁面中按一下 RDS，然後按一下執行個體的名稱。
3. 在執行個體的 details (詳細資訊) 頁面，按一下 Deregister (取消註冊)。



標籤

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

標籤可以協助您對 Chef 11.10、Chef 12 和 Chef 12.2 堆疊中的資源進行分組，並追蹤在 AWS 帳單與成本管理 中使用這些資源的成本。

您可以在堆疊和 layer 層級套用標籤。建立標籤時，您會將標籤套用於已加上標籤的結構中每項資源。例如，如果您將標籤套用至 layer，則您要將標籤套用至 layer 中的每個執行個體、Amazon EBS 磁碟區 （根除外） 或 Elastic Load Balancing 負載平衡器。標籤目前無法套用至執行個體根或預設的 EBS 磁碟區。

標籤是您指派給 Stacks AWS OpsWorks 中堆疊或圖層的鍵/值對。建立標籤後，請開啟 Billing and Cost Management 主控台以啟用使用者定義的標籤。如需如何啟用標籤並使用它們來追蹤和管理 AWS OpsWorks Stacks 資源成本的詳細資訊，請參閱 帳單和成本管理使用者指南中的使用成本分配標籤和啟用使用者定義的成本分配標籤。

標籤的運作方式類似於 Stacks AWS OpsWorks 中的自訂屬性。您套用到堆疊的標籤，會由該堆疊中的每個 layer 繼承。在圖層層級，您可以覆寫繼承標籤的值 （但不是金鑰名稱），並新增新的圖層特定標籤。 會將產生的標籤集 AWS OpsWorks 套用至圖層中的所有資源。當您建立新資源或將現有資源指派到某個 layer 時，該 layer 中的新資源將使用相同的標籤集進行標記。

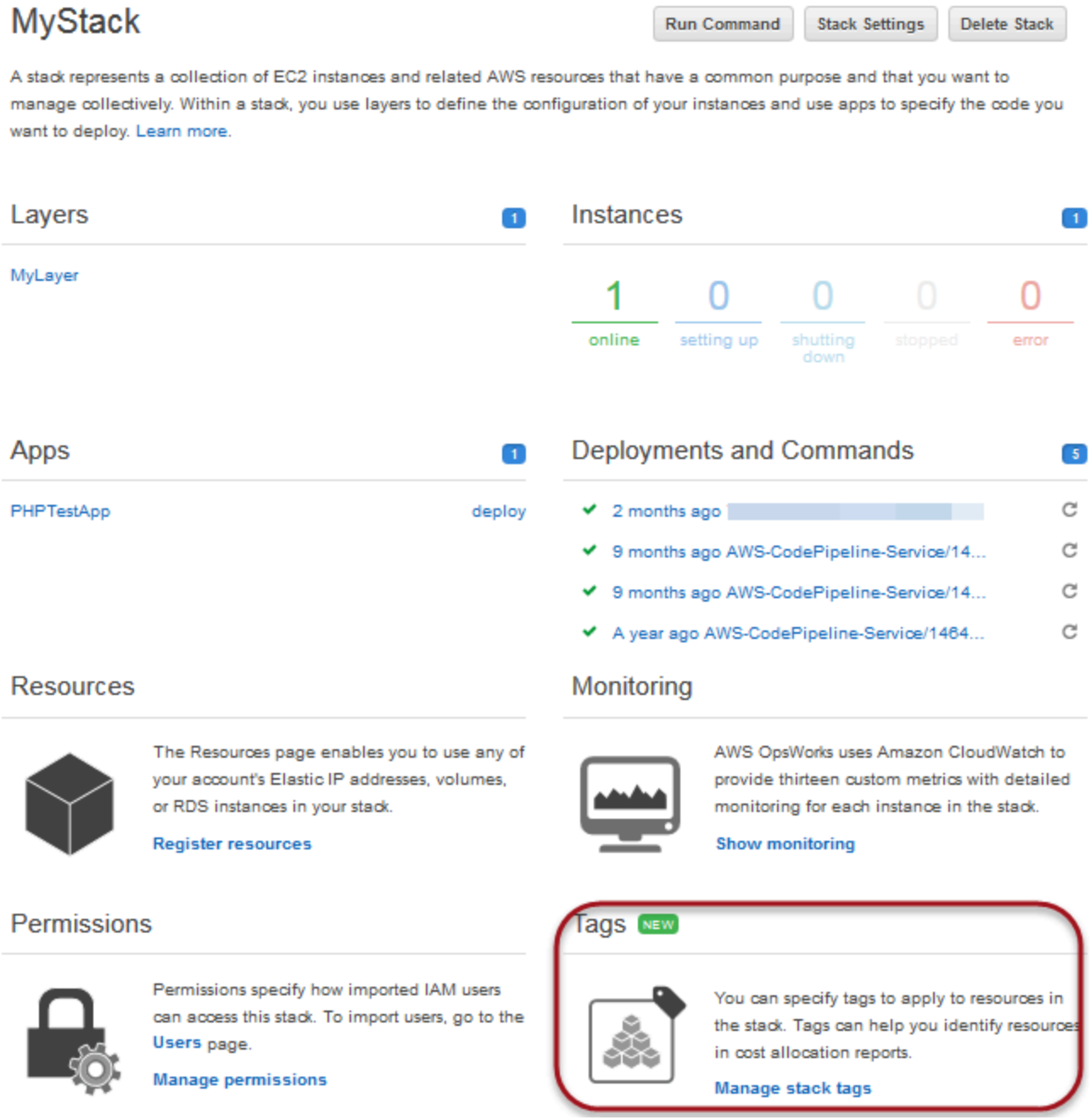
主題

- 在堆疊層級設定標籤

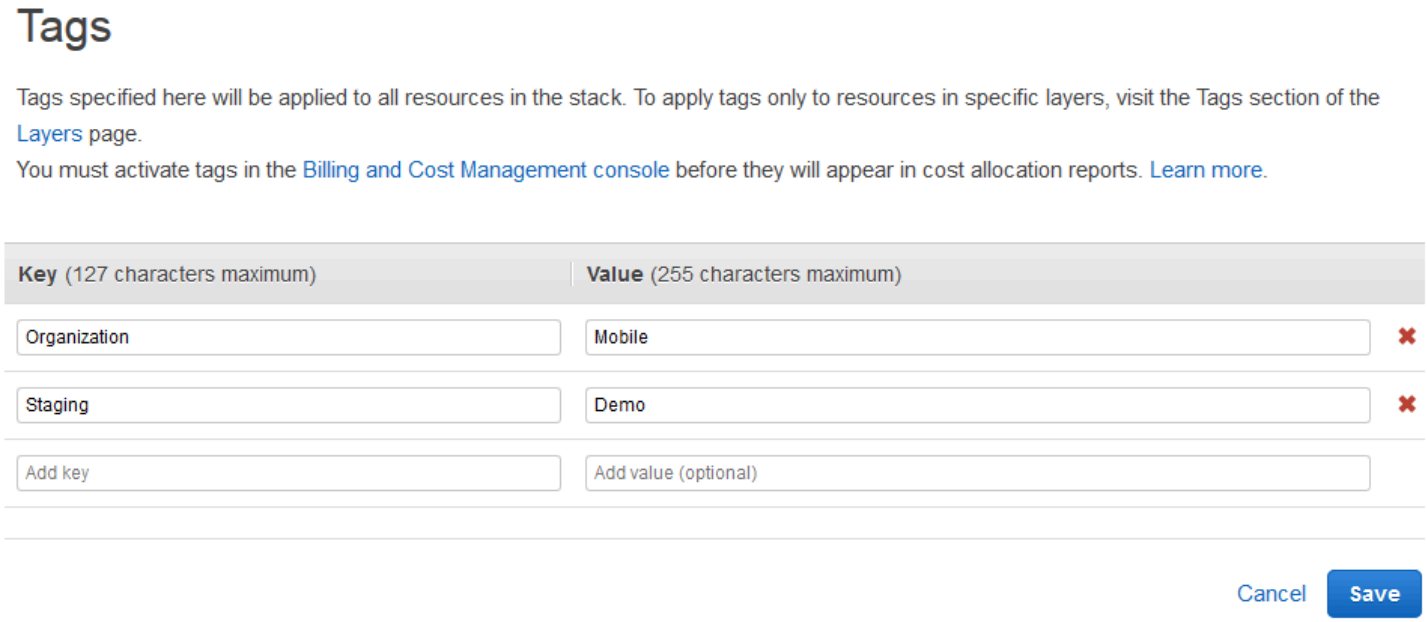
- 在 Layer 層級設定標籤
- 使用 管理標籤 AWS CLI
- 標籤限制

在堆疊層級設定標籤

在堆疊層級，您可以在堆疊的首頁上選擇 Tags (標籤) 來新增和管理標籤。

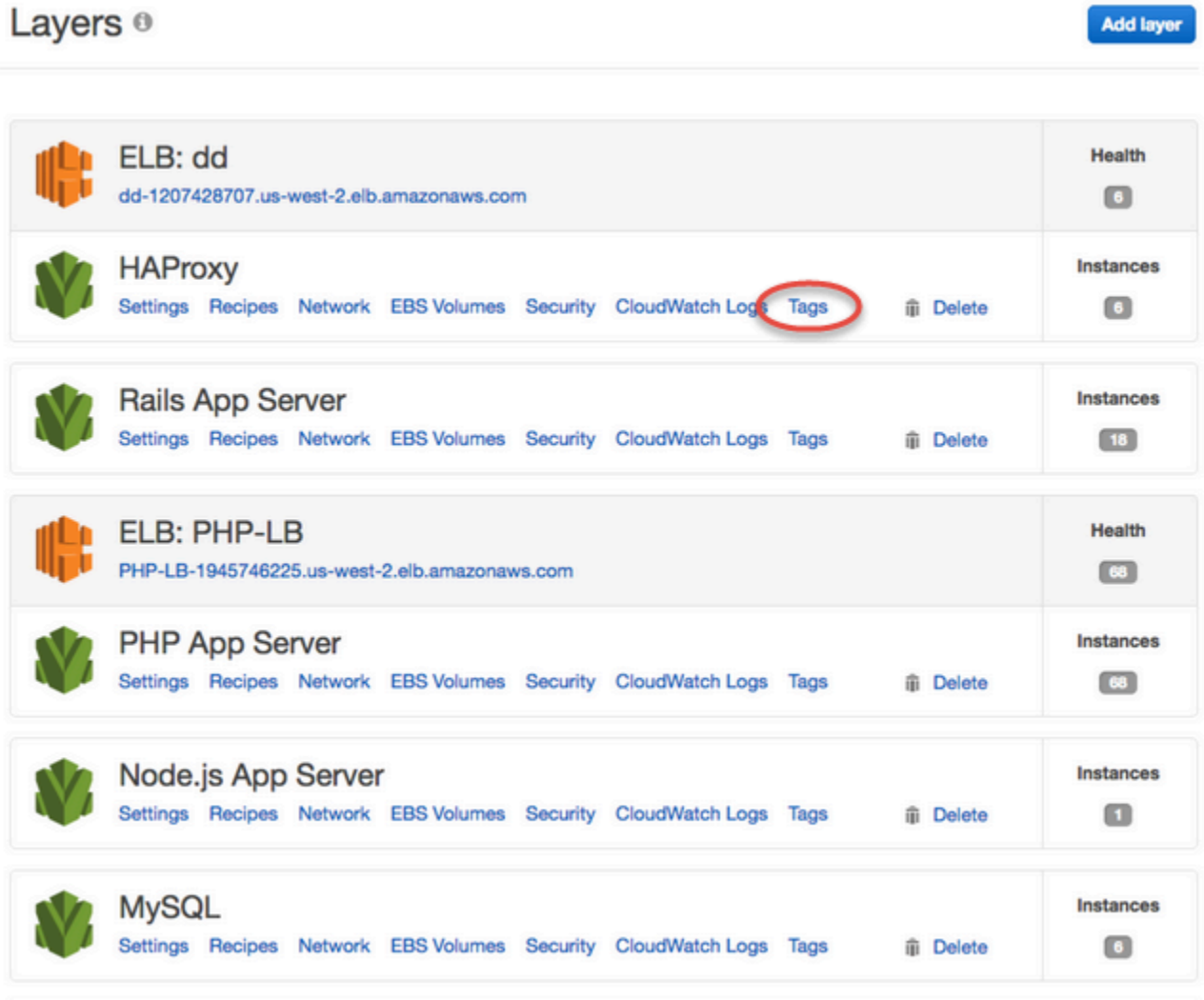


在 Tags (標籤) 頁面上，將標籤新增為鍵/值對。下列螢幕擷取畫面示範一些範例標籤。您可以選擇鍵/值對右邊的紅色 X 來刪除標籤。

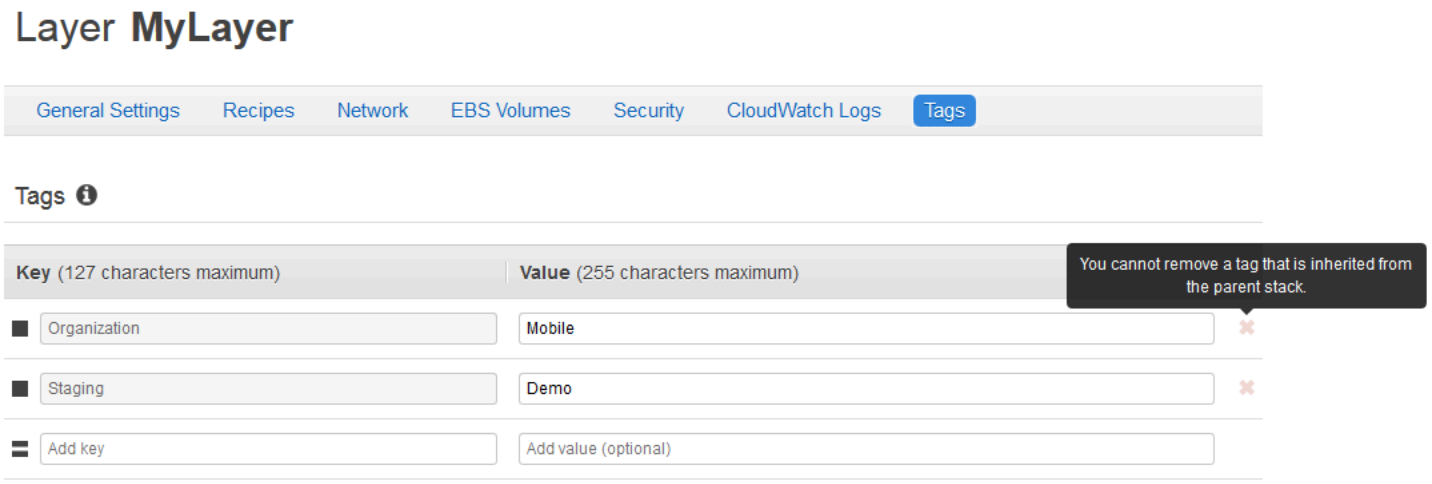


在 Layer 層級設定標籤

在 layer 層級，選擇 Tags (標籤) 標籤來設定標籤。您可以在 Layers (Layer) 首頁以及每個 layer 的首頁上找到此標籤。



當您在 layer 層級變更或新增標籤時，請注意，已在父堆疊層級新增的標籤將由該 layer 及其資源繼承。雖然您可以變更繼承標籤的值，但無法變更鍵名稱或刪除繼承標籤。請在堆疊設定中變更鍵名稱或刪除從父堆疊繼承的標籤。下列螢幕擷取畫面示範從堆疊層級繼承的標籤。繼承的標籤呈現灰色。



如需將標籤新增至堆疊的詳細資訊，請參閱建立新的堆疊。如需如何將標籤新增至 layer 的詳細資訊，請參閱編輯 OpsWorks Layer 的組態。

使用 管理標籤 AWS CLI

您也可以使用 AWS CLI 命令來新增和移除堆疊和層層級的標籤。如需下載和安裝 的詳細資訊 AWS CLI，請參閱安裝 AWS 命令列界面。如果要標記的堆疊不在您的預設區域內，請記得將 `--region` 參數新增到您的命令中。Layer ARNs目前未出現在 中 AWS Management Console。若要取得 layer 的 ARN，請執行 describe-layers 命令。

使用 新增標籤 AWS CLI

- 在 AWS CLI 命令提示中，輸入下列命令，取代 *stack_or_layer_ARN* 並指定您的鍵/值對標籤，然後按 Enter。雙引號會以反斜線逸出。

```
aws opsworks tag-resource --resource-arn stack_or_layer_ARN --tags "{\"key\":
\"value\",\"key\":\"value\"}"
```

以下是範例。

```
aws opsworks tag-resource --resource-arn arn:aws:opsworks:us-
east-2:800000000003:stack/500b99c0-ec00-4cgg-8a0d-1000000jjd1b --tags "{\"Stage\":
\"Production\",\"Organization\":\"Mobile\"}"
```

使用 移除標籤 AWS CLI

- 在 AWS CLI 命令提示字元中輸入以下內容，然後按 Enter。

```
aws opsworks untag-resource --resource-arn stack_or_layer_ARN --tag-keys "[\"key\",
\"key\"]"
```

若要移除標籤，您只需指定您想要移除的標籤鍵即可。以下是範例。

```
aws opsworks untag-resource --resource-arn arn:aws:opsworks:us-
east-2:800000000003:stack/500b99c0-ec00-4cgg-8a0d-1000000jjd1b --tag-keys "[\"Stage
\",\"Organization\"]"
```

> **Note**
>
> 您無法從 layer 中移除繼承的標籤 (在父堆疊層級中新增的標籤)。請從堆疊移除繼承的標籤。

標籤限制

建立標籤時，請謹記下列限制：

- AWS OpsWorks Stacks 會將堆疊和 layer 層級的使用者定義標籤數量限制為 40，包括從父層級繼承的使用者定義標籤。這會保留 10 個可用插槽，用於以 開頭的預設標籤`opsworks:`，以及由其他 AWS 程序設定的標籤。資源上最多允許 50 個標籤，包括由 建立的使用者定義和預設標籤 AWS。
- 標籤鍵不能以 **`aws:`**、**`opsworks:`** 或 **`rds:`** 起始。請勿使用 **`name`**或 **`Name`**作為標籤索引鍵，因為 Stacks AWS OpsWorks **`Name`**已預留。
- 一個鍵最多可包含 127 個字元，並且只能包含 Unicode 字母、數字或分隔符號，或是下列特殊字元：`+ - = . _ : /` 。
- 一個值最多可包含 255 個字元，並且只能包含 Unicode 字母、數字或分隔符號，或是下列特殊字元：`+ - = . _ : /` 。

監控

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可採用以下方式來監控您的堆疊。

- AWS OpsWorks Stacks 使用 Amazon CloudWatch 為堆疊中的每個執行個體提供十三個具有詳細監控的自訂指標。
- AWS OpsWorks Stacks 與 整合 AWS CloudTrail ，以記錄每個 AWS OpsWorks Stacks API 呼叫，並將資料存放在 Amazon S3 儲存貯體中。

- 您可以使用 Amazon CloudWatch Logs 來監控堆疊的系統、應用程式和自訂日誌。

主題

- 使用 Amazon CloudWatch 監控堆疊
- 使用 AWS OpsWorks 記錄 Stacks API 呼叫 AWS CloudTrail
- 搭配 Stacks AWS OpsWorks 使用 Amazon CloudWatch Logs
- 使用 Amazon CloudWatch Events 監控堆疊

使用 Amazon CloudWatch 監控堆疊

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 使用 Amazon CloudWatch (CloudWatch) 來監控堆疊。

- 對於 Linux 堆疊， AWS OpsWorks Stacks 支援十三個自訂指標，以提供堆疊中每個執行個體的詳細監控，並在監控頁面上摘要資料，以方便您使用。
- 對於 Windows 堆疊，您可以使用 CloudWatch 主控台監控執行個體的標準 Amazon EC2 指標。

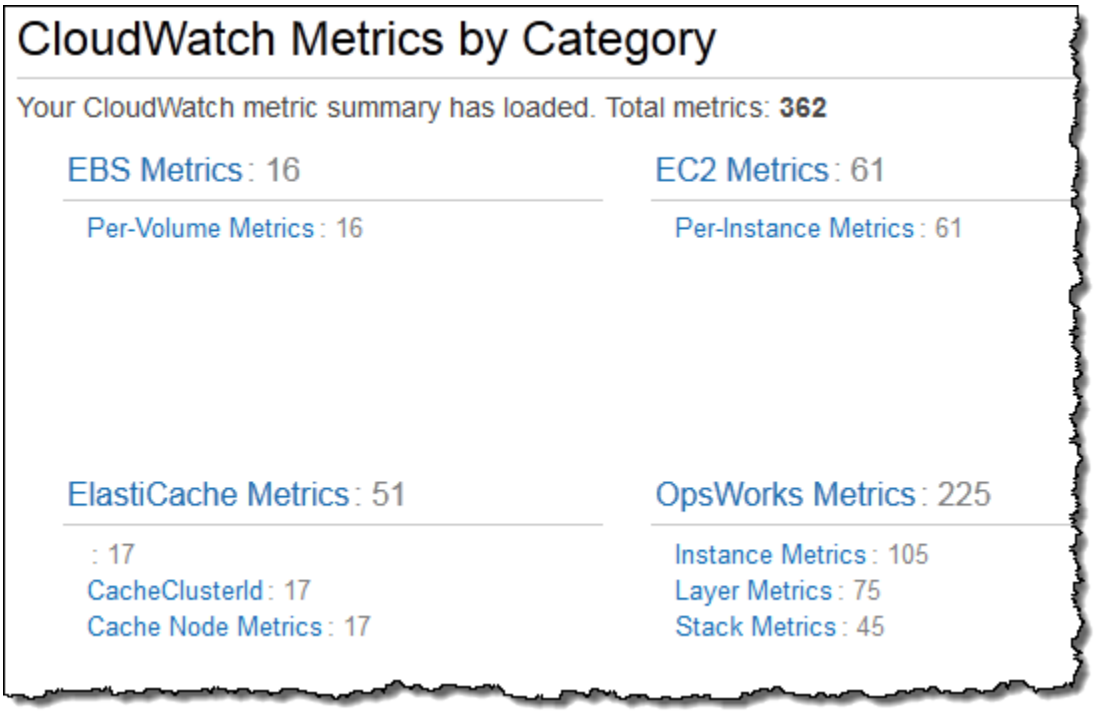
  Monitoring (監控) 頁面不會顯示 Windows 指標。

監控頁面會顯示整個堆疊、 layer 或執行個體的指標。 AWS OpsWorks 堆疊指標與 Amazon EC2 指標不同。您也可以透過 CloudWatch 主控台啟用其他指標，但它們通常需要額外付費。您也可以在 CloudWatch 主控台上檢視基礎資料，如下所示：

在 CloudWatch 中檢視 OpsWorks 自訂指標

1. 透過 https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台。
2. 在導覽列中，選取堆疊的區域。
3. 在導覽窗格中，選擇 指標。

4. 在 OpsWorks Metrics (OpsWorks 指標) 中，選擇 Instance Metrics (執行個體指標)、Layer Metrics (Layer 指標) 或 Stack Metrics (堆疊指標)。



> **Note**
>
> AWS OpsWorks Stacks 透過在每個執行個體（執行個體代理程式）上執行程序來收集指標。由於 CloudWatch 使用 Hypervisor 以不同方式收集指標，因此 CloudWatch 主控台中的值可能與 Stacks 主控台中監控頁面上 AWS OpsWorks 的對應值略有不同。

您也可以使用 CloudWatch 主控台來設定警示。如需如何建立警示的詳細資訊，請參閱建立 Amazon CloudWatch 警示。如需 CloudWatch 自訂指標的清單，請參閱 AWS OpsWorks 指標和維度。如需詳細資訊，請參閱 Amazon CloudWatch。

主題

- AWS OpsWorks 堆疊指標
- AWS OpsWorks 堆疊指標的維度
- 堆疊指標
- Layer 指標
- 執行個體指標

AWS OpsWorks 堆疊指標

AWS OpsWorks Stacks 每五分鐘將下列指標傳送至 CloudWatch。

CPU 指標

指標	描述
cpu_idle	CPU 閒置時間的百分比。 有效維度：您要檢視指標之個別資源IDs：StackId、LayerId 或 InstanceId。 有效統計資料：Average、Minimum、Sum、Maximum或 Data Samples。 單位：無
cpu_nice	CPU 處理具有正nice值之程序的時間百分比，其排程優先順序較低。如需這項措施的詳細資訊，請參閱良好 (Unix)。 有效維度：您要檢視指標之個別資源IDs：StackId、LayerId 或 InstanceId。 有效統計資料：Average、Minimum、Sum、Maximum或 Data Samples。 單位：無
cpu_steal	隨著 AWS 在越來越多的執行個體中配置 Hypervisor CPU 資源，虛擬化負載會增加，並可能影響 Hypervisor 在執行個體上執行請求工作的頻率。會cpu_steal 測量執行個體等待 Hypervisor 配置實體 CPU 資源的時間百分比。 有效維度：您要檢視指標之個別資源IDs：StackId、LayerId 或 InstanceId。 有效統計資料：Average、Minimum、Sum、Maximum或 Data Samples。

指標	描述
	單位：無
cpu_system	CPU 正在處理系統操作的時間百分比。 有效維度：您要檢視指標之個別資源IDs：StackId、LayerId 或 InstanceId。 有效統計資料：Average、Minimum、Sum、Maximum或 Data Samples。 單位：無
cpu_user	CPU 處理使用者操作的時間百分比。 有效維度：您要檢視指標之個別資源IDs：StackId、LayerId 或 InstanceId。 有效統計資料：Average、Minimum、Sum、Maximum或 Data Samples。 單位：無
cpu_waitio	CPU 等待輸入/輸出操作的時間百分比。 有效維度：您要檢視指標之個別資源IDs：StackId、LayerId 或 InstanceId。 有效統計資料：Average、Minimum、Sum、Maximum或 Data Samples。 單位：無

記憶體指標

指標	描述
memory_buffers	緩衝記憶體的數量。

指標	描述
	有效維度：您要檢視指標之個別資源IDs：StackId、LayerId 或 InstanceId。 有效統計資料：Average、Minimum、Sum、Maximum或 Data Samples。 單位：無
memory_cached	快取記憶體的數量。 有效維度：您要檢視指標之個別資源IDs：StackId、LayerId 或 InstanceId。 有效統計資料：Average、Minimum、Sum、Maximum或 Data Samples。 單位：無
memory_free	可用記憶體的數量。 有效維度：您要檢視指標之個別資源IDs：StackId、LayerId 或 InstanceId。 有效統計資料：Average、Minimum、Sum、Maximum或 Data Samples。 單位：無
memory_swap	交換空間的數量。 有效維度：您要檢視指標之個別資源IDs：StackId、LayerId 或 InstanceId。 有效統計資料：Average、Minimum、Sum、Maximum或 Data Samples。 單位：無

指標	描述
memory_total	記憶體總量。 有效維度：您要檢視指標之個別資源IDs：StackId、LayerId 或 InstanceId。 有效統計資料：Average、Minimum、Sum、Maximum或 Data Samples。 單位：無
memory_used	使用的記憶體量。 有效維度：您要檢視指標之個別資源IDs：StackId、LayerId 或 InstanceId。 有效統計資料：Average、Minimum、Sum、Maximum或 Data Samples。 單位：無

負載指標

指標	描述
load_1	一分鐘時段的平均負載。 有效維度：您要檢視指標之個別資源IDs：StackId、LayerId 或 InstanceId。 有效統計資料：Average、Minimum、Sum、Maximum或 Data Samples。 單位：無
load_5	五分鐘時段的平均負載。 有效維度：您要檢視指標之個別資源IDs：StackId、LayerId 或 InstanceId。

指標	描述
	有效統計資料：Average、Minimum、Sum、Maximum或 Data Samples。 單位：無
load_15	在 15 分鐘的時段內平均負載。 有效維度：您要檢視指標之個別資源IDs：StackId、LayerId 或 InstanceId。 有效統計資料：Average、Minimum、Sum、Maximum或 Data Samples。 單位：無

程序指標

指標	描述
procs	作用中程序的數量。 有效維度：您要檢視指標之個別資源IDs：StackId、LayerId 或 InstanceId。 有效統計資料：Average、Minimum、Sum、Maximum或 Data Samples。 單位：無

AWS OpsWorks 堆疊指標的維度

AWS OpsWorks Stacks 指標使用 AWS OpsWorks Stacks 命名空間，並提供下列維度的指標：

維度	描述
StackId	堆疊的平均值。

維度	描述
LayerId	分層的平均值。
InstanceId	執行個體的平均值。

堆疊指標

若要檢視整個堆疊的指標摘要，請在堆疊 AWS OpsWorks 儀表板中選取堆疊，然後在導覽窗格中按一下監控。下列範例為使用 PHP 和資料庫 layer 的堆疊。

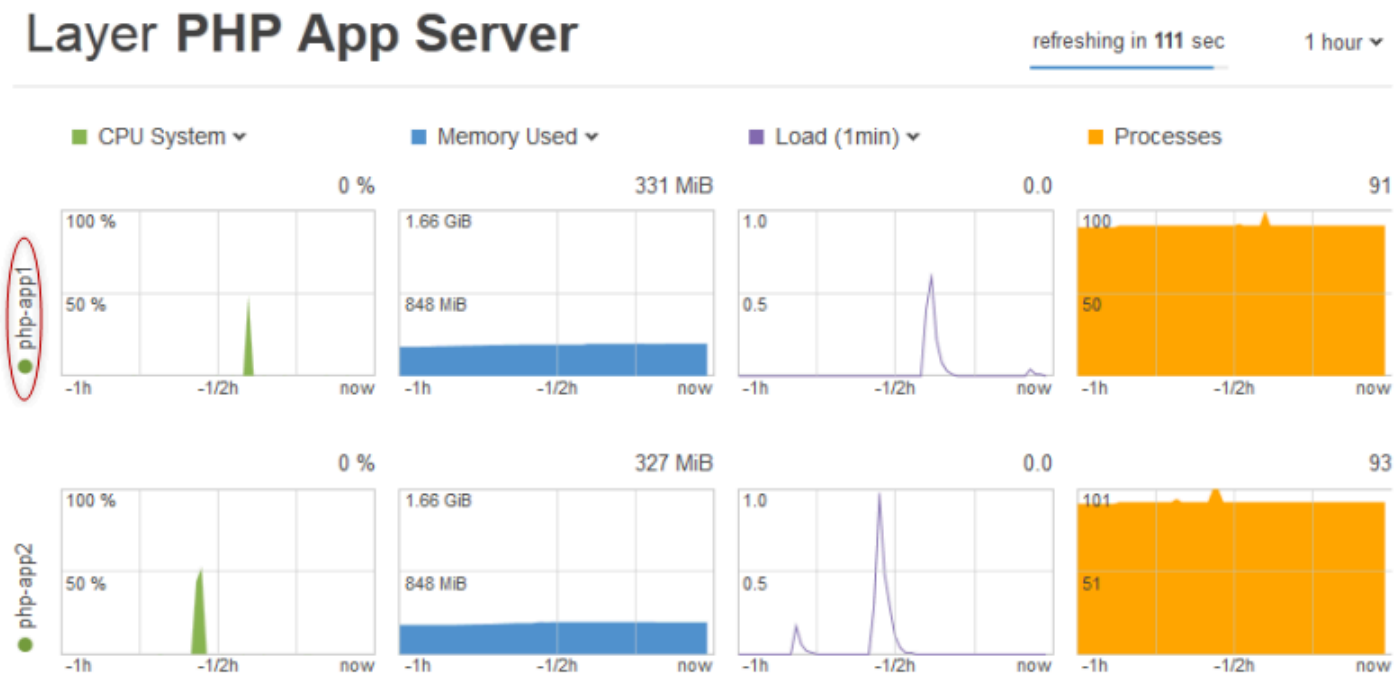
堆疊檢視會為每個 layer 顯示經過一段指定時間後四類指標的圖表：1 小時、8 小時、24 小時、1 週或 2 週。注意下列事項：

- AWS OpsWorks Stacks 會定期更新圖形；右上角的倒數計時器會指出下次更新之前剩餘的時間，
- 如果 layer 有多個執行個體，圖表會顯示 layer 的平均值。
- 您可以按一下右上的清單，然後選取您慣用的值指定期間。

您可以使用圖表上方的清單，為每個指標類型選取您要檢視的特定指標。

Layer 指標

若要查看特定 layer 的指標，請按一下 Monitoring Layers (監控 Layers) 檢視中的 layer 名稱。下列範例顯示 PHP layer 的指標，它有兩個執行個體。

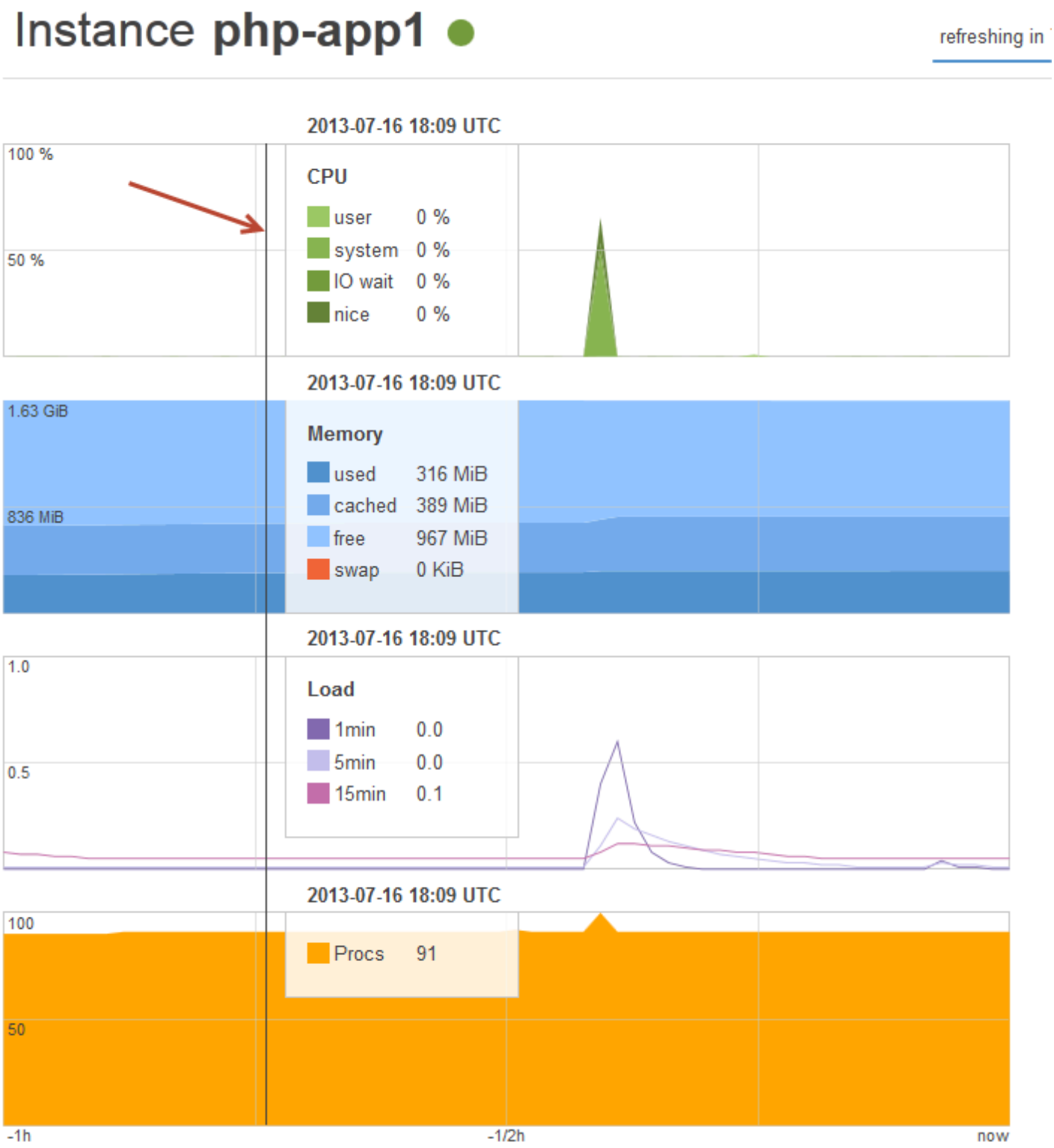


指標類型和堆疊指標的類型相同，您可以使用圖表上方的清單，為每個類型選取您要檢視的特定指標。

> **Note**
>
> 您也可以移至 layer 的 details (詳細資訊) 頁面，按一下右上角的 Monitoring (監控)，顯示 layer 指標。

執行個體指標

若要檢視特定執行個體的指標，請按一下 layer 監控檢視中的執行個體名稱。下列範例顯示 PHP layer php-app1 執行個體的指標。



圖表摘要每個指標類型所有可用的指標。若要取得特定時間點的確切值，請使用滑鼠將滑桿 (上圖中紅色箭頭所指處) 移至適當的位置。

> **Note**
>
> 您也可以移至執行個體的 details (詳細資訊) 頁面，選擇右上角的 Monitoring (監控)，來顯示執行個體指標。

使用 AWS OpsWorks 記錄 Stacks API 呼叫 AWS CloudTrail

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 已與 整合 AWS CloudTrail，此服務提供 IAM 身分所採取動作的記錄，或 Stacks AWS OpsWorks 中的 AWS 服務。CloudTrail AWS OpsWorks 會將 Stacks 的所有 API 呼叫擷取為事件，包括從 Stacks AWS OpsWorks 主控台以及從程式碼呼叫到 AWS OpsWorks Stacks APIs 呼叫。如果您建立線索，則可以啟用 CloudTrail 事件持續交付至 Amazon S3 儲存貯體，包括 Stacks AWS OpsWorks 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。使用 CloudTrail 收集的資訊，您可以判斷對 AWS OpsWorks Stacks 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 OpsWorks CloudTrail 中的堆疊資訊

當您建立 AWS 帳戶時，您的帳戶上會啟用 CloudTrail。當活動在 AWS OpsWorks Stacks 中發生時，該活動會記錄於 CloudTrail 事件，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 Stacks AWS OpsWorks 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，當您在主控台建立權杖時，權杖會套用到所有區域。追蹤會記錄 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 CloudTrail 日誌中收集的事件資料並對其採取行動。如需詳細資訊，請參閱：

- 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，以及從多個帳戶接收 CloudTrail 日誌檔案

所有 AWS OpsWorks Stacks 動作都會由 CloudTrail 記錄，並記錄在 AWS OpsWorks Stacks API 參考中。例如，對 `CreateLayer`、`DescribeInstances` 以及 `StartInstance` 動作發出的呼叫會在 CloudTrail 日誌檔案中產生項目。

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：

- 該請求是否使用根或 IAM 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 OpsWorks Stacks 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 日誌檔案並非依公有 API 呼叫追蹤記錄的堆疊排序，因此不會以任何特定順序出現。

以下範例顯示的是展示 `CreateLayer` 動作的 CloudTrail 日誌項目。

```
  {
"Records": [
    {
        "awsRegion": "us-west-2",
        "eventID": "342cd1ec-8214-4a0f-a68f-8e6352feb5af",
        "eventName": "CreateLayer",
        "eventSource": "opsworks.amazonaws.com",
        "eventTime": "2014-05-28T16:05:29Z",
        "eventVersion": "1.01"ed,
        "requestID": "e3952a2b-e681-11e3-aa71-81092480ee2e",
        "requestParameters": {
            "attributes": {},
            "customRecipes": {},
            "name": "2014-05-28 16:05:29 +0000 a073",
```

```
                "shortname": "customcf4571d5c0d6",
                "stackId": "a263312e-f937-4949-a91f-f32b6b641b2c",
                "type": "custom"
            },
            "responseElements": null,
            "sourceIPAddress": "198.51.100.0",
            "userAgent": "aws-sdk-ruby/2.0.0 ruby/2.1 x86_64-linux",
            "userIdentity": {
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "accountId": "111122223333",
                "arn": "arn:aws:iam::111122223333:user/A-User-Name",
                "principalId": "AKIAI44QH8DHBEXAMPLE",
                "type": "IAMUser",
                "userName": "A-User-Name"
            }
        },
        {
            "awsRegion": "us-west-2",
            "eventID": "a860d8f8-c1eb-449b-8f55-eafc373b49a4",
            "eventName": "DescribeInstances",
            "eventSource": "opsworks.amazonaws.com",
            "eventTime": "2014-05-28T16:05:31Z",
            "eventVersion": "1.01",
            "requestID": "e4691bfd-e681-11e3-aa71-81092480ee2e",
            "requestParameters": {
                "instanceIds": [
                    "218289c4-0492-473d-a990-3fbe1efa25f6"
                ]
            },
            "responseElements": null,
            "sourceIPAddress": "198.51.100.0",
            "userAgent": "aws-sdk-ruby/2.0.0 ruby/2.1x86_64-linux",
            "userIdentity": {
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "accountId": "111122223333",
                "arn": "arn:aws:iam::111122223333:user/A-User-Name",
                "principalId": "AKIAI44QH8DHBEXAMPLE",
                "type": "IAMUser",
                "userName": "A-User-Name"
            }
        }
    ]
}
```

搭配 Stacks AWS OpsWorks 使用 Amazon CloudWatch Logs

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

為了簡化監控多個執行個體日誌的程序， AWS OpsWorks Stacks 支援 Amazon CloudWatch Logs。您可以在 Stacks 的 layer AWS OpsWorks 層級啟用 CloudWatch Logs。CloudWatch Logs 整合適用於 Chef 11.10 和 Chef 12 Linux 型堆疊。當您啟用 CloudWatch Logs 時，會產生額外費用，因此請在開始之前檢閱 Amazon CloudWatch 定價。

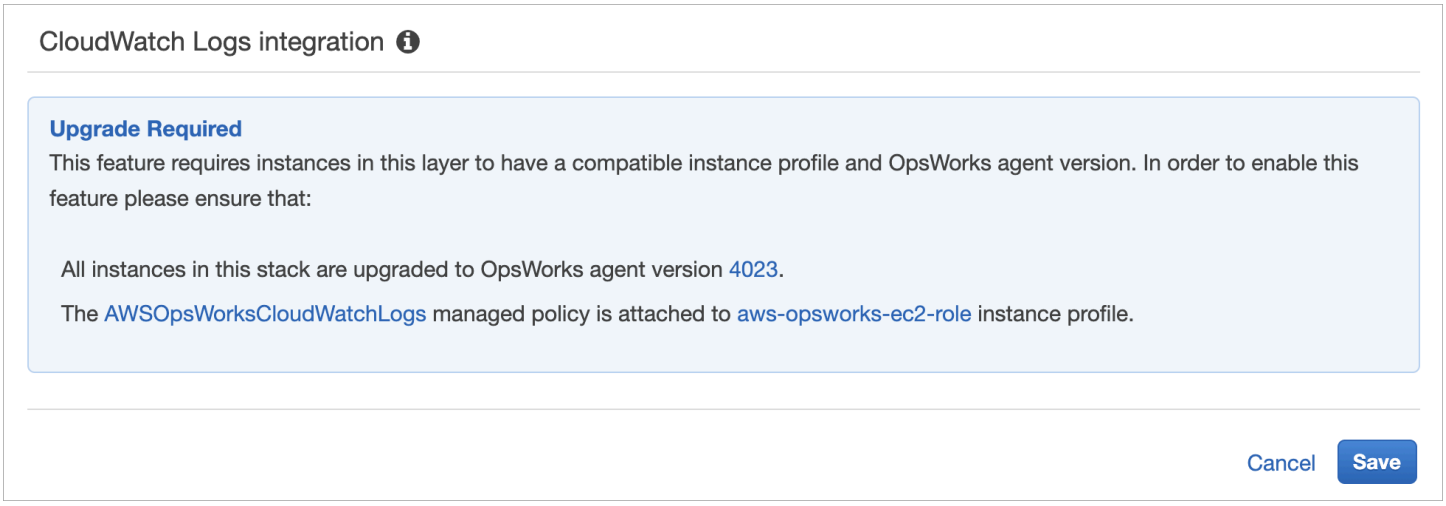
CloudWatch Logs 會監控所選日誌是否存在使用者指定的模式。例如，您可以監控出現 `NullReferenceException` 等文字的日誌，或計算這類事件的數目。在 Stacks AWS OpsWorks 中啟用 CloudWatch Logs 之後，Stacks AWS OpsWorks 代理程式會將日誌傳送至 CloudWatch Logs。如需 CloudWatch Logs 的詳細資訊，請參閱 CloudWatch Logs 入門。

先決條件

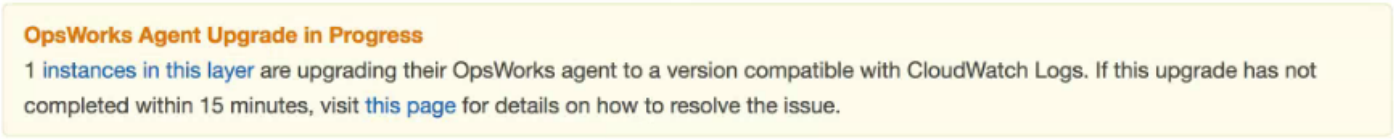
在啟用 CloudWatch Logs 之前，您的執行個體必須在 Chef 11.10 堆疊中執行 Stacks 代理程式的 3444 版或更新版本 AWS OpsWorks ，並在 Chef 12 堆疊中執行 4023 版或更新版本。您也必須為使用 CloudWatch Logs 監控的任何執行個體使用相容的執行個體描述檔。

如果您使用的是自訂執行個體描述檔 AWS OpsWorks (Stacks 在您建立堆疊時未提供的描述檔）， AWS OpsWorks Stacks 就無法自動升級執行個體描述檔。您必須使用 IAM 手動將 AWSOpsWorksCloudWatchLogs 政策連接至您的設定檔。如需詳細資訊，請參閱《IAM 使用者指南》中的管理 IAM 政策。

如果您需要升級代理程式版本或執行個體描述檔，當您開啟 Layer 頁面上的 CloudWatch Logs 索引標籤時， AWS OpsWorks Stacks 會顯示類似下列螢幕擷取畫面的提醒。



更新 layer 中所有執行個體上的代理程式可能需要一些時間。如果您在客服人員升級完成之前嘗試在 layer 上啟用 CloudWatch Logs，您會看到類似下列的訊息。



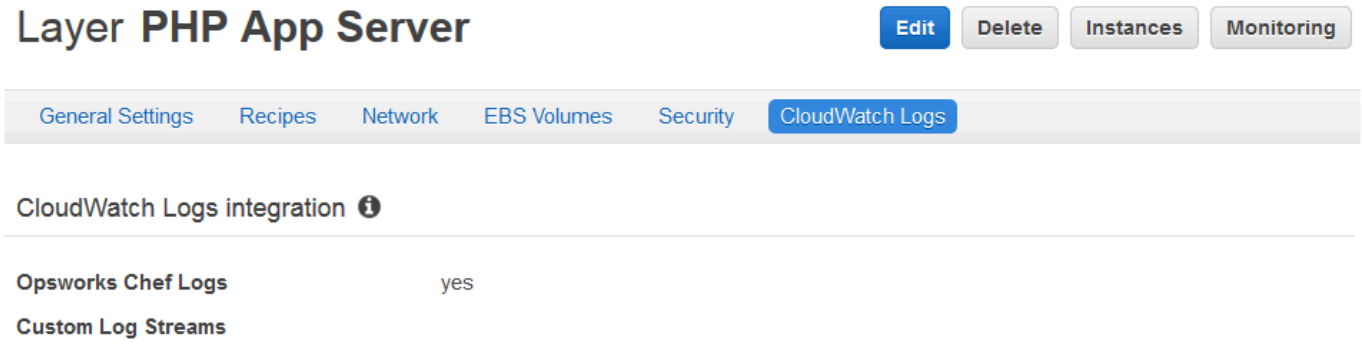
啟用 CloudWatch Logs

1. 完成任何必要的代理程式和執行個體設定檔升級後，您可以透過將 CloudWatch Logs 標籤上的滑桿控制項設定為開啟來啟用 CloudWatch Logs。

2. 若要串流命令日誌，請將 Stream command logs (串流命令日誌) 滑桿設定為 On (開啟)。這會將您 layer 執行個體上的 Chef 活動和使用者啟動命令的日誌傳送至 CloudWatch Logs。

   當您開啟日誌 URL 的目標時，這些日誌中包含的資料與您在 DescribeCommands 操作結果中看到的內容密切相符。它包含有關 `setup`、`configure`、`deploy`、`undeploy`、`start`、`stop` 和配方執行命令的資料。

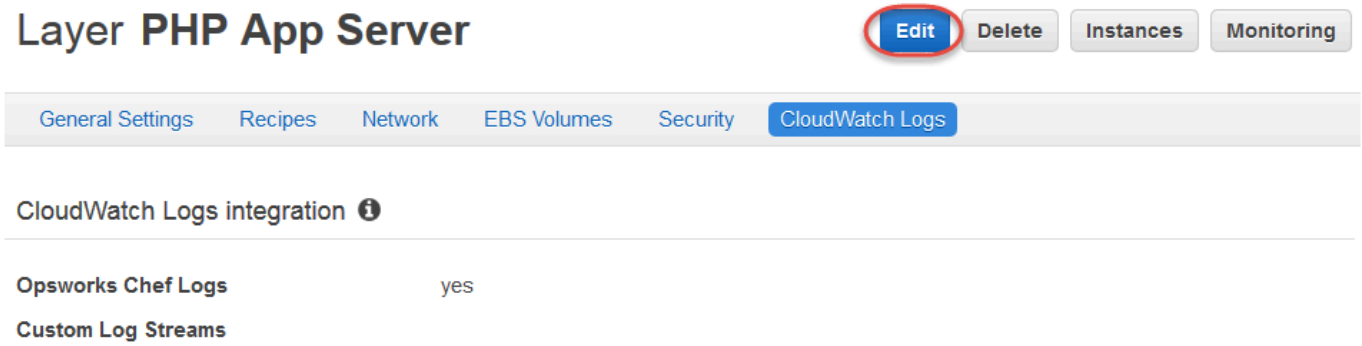
3. 若要串流存放在 layer 執行個體上之自訂位置的活動日誌，例如 `/var/log/apache/myapp/mylog*`，請在 Stream custom logs (串流自訂日誌) 字串方塊中輸入自訂位置，然後選擇 Add (新增) (+)。
4. 選擇 Save (儲存)。在幾分鐘內， AWS OpsWorks Stacks 日誌串流應該會顯示在 CloudWatch Logs 主控台中。



關閉 CloudWatch Logs

若要關閉 CloudWatch Logs，請編輯您的 layer 設定。

1. 在您 layer 的 properties (屬性) 頁面上，選擇 Edit (編輯)。



2. 在編輯頁面上，選擇 CloudWatch Logs (CloudWatch 日誌) 標籤。
3. 在 CloudWatch Logs (CloudWatch 日誌) 區域中，關閉 Stream command logs (串流命令日誌)。選擇自訂日誌的 X 將它們從日誌串流刪除 (如果適用)。
4. 選擇 Save (儲存)。

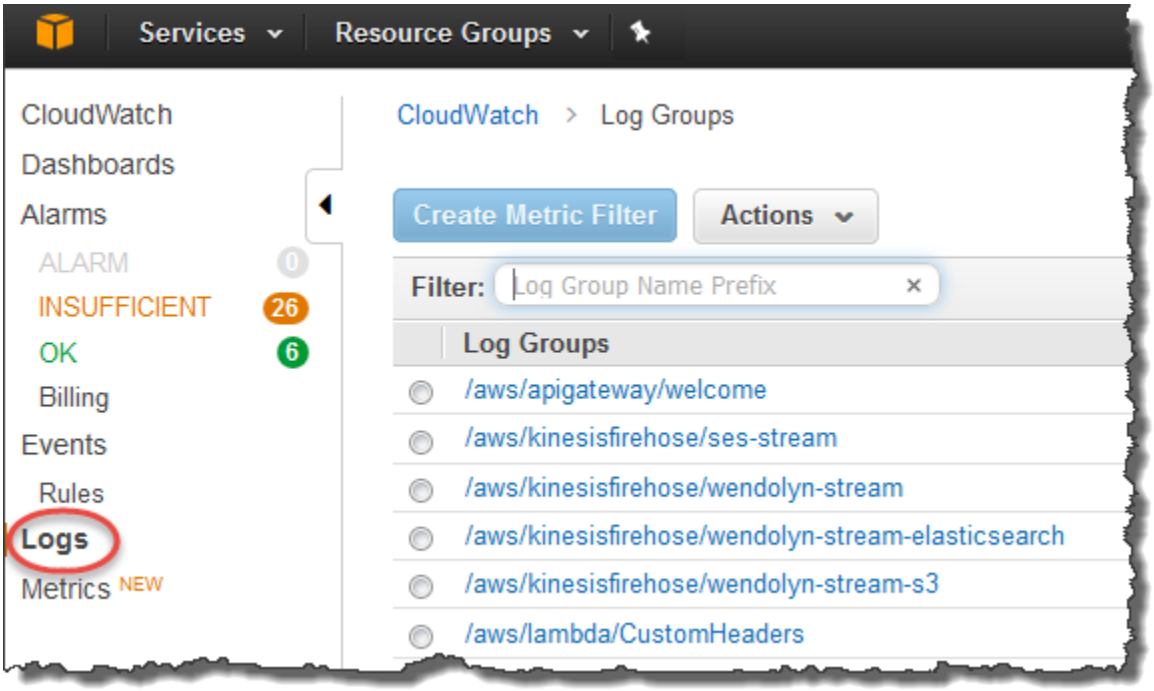
從 CloudWatch Logs 刪除串流日誌

從 Stacks 關閉 CloudWatch Logs AWS OpsWorks 串流後，CloudWatch Logs 管理主控台中仍可使用現有的日誌。除非您將日誌匯出至 Amazon S3 或刪除日誌，否則仍會產生儲存日誌的費用。如需將日誌匯出至 S3 的詳細資訊，請參閱將日誌資料匯出至 Amazon S3。

您可以在 CloudWatch Logs 管理主控台中刪除日誌串流和日誌群組，或執行 `delete-log-stream`和 `delete-log-group` AWS CLI 命令。如需變更日誌保留期間的詳細資訊，請參閱在 CloudWatch Logs 中變更日誌資料保留。

在 CloudWatch Logs 中管理您的日誌

您正在串流的日誌會在 CloudWatch Logs 主控台中管理。



AWS OpsWorks 會自動建立預設日誌群組和日誌串流。 AWS OpsWorks Stacks 資料的日誌群組名稱符合以下模式：

stack_name/*layer_name*/*chef_log_name*

自訂日誌名稱符合下列模式：

/stack_name/layer_short_name/file_path_name。路徑名稱在移除星號 (*) 等特殊字元後，更容易閱讀。

當您在 CloudWatch Logs 中找到日誌時，您可以將日誌組織成群組、建立指標篩選條件來搜尋和篩選日誌，以及建立自訂警示。

設定 Chef 12.2 Windows Layer 使用 CloudWatch Logs

Windows 執行個體不支援 CloudWatch Logs 自動整合。無法在 Chef 12.2 堆疊中的 layer 上使用 CloudWatch Logs (CloudWatch 日誌) 標籤。若要手動啟用串流至適用於 Windows 執行個體的 CloudWatch Logs，請執行下列動作。

- 更新 Windows 執行個體的執行個體設定檔，以便 CloudWatch Logs 代理程式具有適當的許可。AWSOpsWorksCloudWatchLogs 政策陳述式會顯示必要的許可。

  此任務一般只需要執行一次。然後，您可以為 layer 中所有的 Windows 執行個體使用更新的執行個體描述檔。
- 在每個執行個體上編輯下列 JSON 組態檔案。這個檔案包含日誌串流偏好設定，例如要監控哪些日誌。

  ```
  %PROGRAMFILES%\Amazon\Ec2ConfigService\Settings
  \AWS.EC2.Windows.CloudWatch.json
  ```

您可以建立自訂配方處理必要任務，將它們指派給 Chef 12.2 layer 的 Setup (設定) 事件，來自動化前兩項任務。每次在這些層上啟動新的執行個體時， AWS OpsWorks Stacks 都會在執行個體完成開機後自動執行您的配方，以啟用 CloudWatch Logs。

若要關閉 Windows 執行個體上的 CloudWatch Logs，請反轉程序。清除 EC2 Service Properties 對話方塊中的啟用 CloudWatch Logs 整合核取方塊，從 `AWS.EC2.Windows.CloudWatch.json` 檔案刪除日誌串流偏好設定；並停止執行任何自動將 CloudWatch Logs 許可指派給 Chef 12.2 層中新執行個體的 Chef 配方。

使用 Amazon CloudWatch Events 監控堆疊

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以在 Amazon CloudWatch Events AWS OpsWorks 中設定規則，以提醒您 Stacks 資源的變更，並指示 CloudWatch Events 根據事件內容採取動作。如需有關如何開始使用 CloudWatch Events 和設定規則的詳細資訊，請參閱 CloudWatch Events 使用者指南中的 CloudWatch Events 入門。 CloudWatch

CloudWatch Events AWS OpsWorks 支援下列 Stacks 事件類型。

Instance state change (執行個體狀態變更)

表示 Stacks AWS OpsWorks 執行個體的狀態變更。

Command state change (命令狀態變更)

表示 Stacks AWS OpsWorks 命令的狀態發生變更。

Deployment state change (部署狀態變更)

表示 AWS OpsWorks Stacks 部署的狀態發生變更。

Alerts (提醒)

表示已引發 AWS OpsWorks Stacks 服務錯誤。

如需 CloudWatch Events AWS OpsWorks 支援之 Stacks 事件類型的詳細資訊，請參閱 CloudWatch Events 使用者指南中的 AWS OpsWorks Stacks Events。 CloudWatch

安全與許可

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您的每個使用者都必須有適當的 AWS 登入資料，才能存取您帳戶 AWS 的資源。向使用者提供登入資料的建議方法是使用 AWS Identity and Access Management(IAM)。 AWS OpsWorks Stacks 與 IAM 整合，可讓您控制下列項目：

- 個別使用者如何與 Stacks AWS OpsWorks 互動。

例如，您可以允許某些使用者在任何堆疊部署應用程式，但不能修改堆疊本身，同時授予其他使用者特定堆疊的完整存取權等等。

- Stacks AWS OpsWorks 如何代表您存取堆疊資源，例如 Amazon EC2 執行個體和 Amazon S3 儲存貯體。

  AWS OpsWorks Stacks 提供的服務角色可授予這些任務的許可。

- 在 Stacks 控制之 Amazon EC2 AWS OpsWorks 執行個體上執行的應用程式如何存取其他 AWS 資源，例如存放在 Amazon S3 儲存貯體上的資料。

  您可以將執行個體描述檔指派給 layer 的執行個體，將許可授予在這些執行個體上執行的應用程式，以存取其他 AWS 資源。

- 如何管理以使用者為基礎的 SSH 金鑰，以及使用 SSH 或 RDP 連線到執行個體。

  針對每個堆疊，管理使用者都可以指派每個 使用者個人的 SSH 金鑰，或授權使用者指定自己的金鑰。您也可以在每個使用者的堆疊執行個體上授權 SSH 或 RDP 存取權以及 sudo 或管理員權限。

其他安全方面包括下列項目：

- 如何使用最新的安全性修補程式管理更新您的執行個體作業系統。

  如需詳細資訊，請參閱管理安全性更新。

- 如何設定 Amazon EC2 安全群組，以控制往返執行個體的網路流量。

  如何指定自訂安全群組，而不是 AWS OpsWorks Stacks 預設安全群組。如需詳細資訊，請參閱使用安全群組。

主題

- 管理 AWS OpsWorks Stacks 使用者許可
- 允許 AWS OpsWorks Stacks 代表您
- Stacks AWS OpsWorks 中的跨服務混淆代理預防
- 指定在 EC2 執行個體上執行之應用程式的許可
- 管理 SSH 存取
- 管理 Linux 安全性更新
- 使用安全群組

管理 AWS OpsWorks Stacks 使用者許可

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

最佳實務是，將 AWS OpsWorks Stacks 使用者限制為一組指定的動作或一組堆疊資源。您可以透過兩種方式控制 AWS OpsWorks Stacks 使用者許可：使用 AWS OpsWorks Stacks 許可頁面，以及套用適當的 IAM 政策。

OpsWorks 許可頁面，或同等的 CLI 或 API 動作，可讓您透過將多個許可層級之一指派給每個使用者，以控制多使用者環境中的使用者許可。每個層級都會授予特定堆疊資源之標準動作組的許可。使用 Permissions (許可) 頁面，您可以控制下列項目：

- 誰可以存取每個堆疊。
- 每個使用者在每個堆疊上允許執行的動作有哪些。

  例如，您可以允許一部分的使用者僅能檢視堆疊，其他使用者則能部署應用程式、新增執行個體等。
- 誰可以管理每個堆疊。

  您可以將每個堆疊的管理委派給一或多個指定的使用者。
- 在每個堆疊的 Amazon EC2 執行個體上，擁有使用者層級 SSH 存取和 sudo 權限 (Linux) 或 RDP 存取和管理員權限 (Windows) 的人員。

  您可以隨時為每個使用者分別授予或移除這些許可。

> **Important**
>
> 拒絕 SSH/RDP 存取不一定能防止使用者登入執行個體。如果您為執行個體指定 Amazon EC2 金鑰對，則任何具有對應私有金鑰的使用者都可以登入或使用 金鑰來擷取 Windows 管理員密碼。如需詳細資訊，請參閱管理 SSH 存取。

您可以使用 IAM 主控台、CLI 或 API 將政策新增至您的使用者，以授予各種 AWS OpsWorks Stacks 資源和動作的明確許可。

- 使用 IAM 政策來指定許可比使用許可層級更靈活。
- 您可以設定 IAM 身分 （使用者、使用者群組和角色），將許可授予 IAM 身分，例如使用者和使用者群組，或定義可與聯合身分使用者相關聯的角色。
- IAM 政策是授予特定金鑰 Stacks AWS OpsWorks 動作許可的唯一方法。

  例如，您必須使用 IAM 來授予 `opsworks:CreateStack`和 的許可`opsworks:CloneStack`，這些許可分別用於建立和複製堆疊。

雖然無法在主控台中明確匯入聯合身分使用者，但聯合身分使用者可以透過選擇 Stacks AWS OpsWorks 主控台右上角的我的設定，然後選擇使用者，也可以選擇右上角的 來隱含地建立使用者設定檔。在 使用者 頁面上，聯合身分使用者 - 其帳戶是使用 API 或 CLI 建立，或隱含地透過主控台 - 可以與非聯合身分使用者類似地管理其帳戶。

兩種方式並非互斥關係，有時候合併使用兩者也會非常有用。 AWS OpsWorks Stacks 接著便會評估這兩組許可。例如，假設您希望允許使用者新增或刪除執行個體，但不希望其新增或刪除 layer。Stacks AWS OpsWorks 許可層級都不會授予該特定許可集。不過，您可以使用許可頁面來授予使用者管理許可層級，這可讓使用者執行大多數堆疊操作，然後套用 IAM 政策來拒絕新增或移除層的許可。如需詳細資訊，請參閱使用 政策控制對 資源的 AWS 存取。

以下是管理使用者許可的典型模型。在每個案例中，皆預設讀者 (即您) 具備管理員身分。

1. 使用 IAM 主控台將 AWSOpsWorks_FullAccess 政策套用至一或多個管理使用者。
2. 使用不授予 AWS OpsWorks Stacks 許可的政策，為每個非管理使用者建立使用者。

   如果使用者只需要存取 AWS OpsWorks Stacks，您可能完全不需要套用政策。您可以改為使用 AWS OpsWorks Stacks 許可頁面管理其許可。
3. 使用 AWS OpsWorks Stacks 使用者頁面將非管理使用者匯入 AWS OpsWorks Stacks。
4. 針對每個堆疊，使用堆疊的 Permissions (許可) 頁面，將許可層級指派給每個使用者。
5. 視需要套用適當設定的 IAM 政策，以自訂使用者的許可層級。

如需有關管理使用者的詳細資訊，請參閱 最佳實務：管理許可。

如需 IAM 最佳實務的詳細資訊，請參閱《IAM 使用者指南》中的 IAM 中的安全最佳實務。

主題

- 管理 AWS OpsWorks Stacks 使用者

- 授予 Stacks AWS OpsWorks 使用者每個堆疊的許可
- 連接 AWS OpsWorks IAM 政策來管理 Stacks 許可
- 範例政策
- AWS OpsWorks Stacks 許可層級

管理 AWS OpsWorks Stacks 使用者

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您必須先為每個人建立使用者，才能將使用者匯入 AWS OpsWorks Stacks 並授予許可。若要建立 IAM 使用者，請先以已授予 IAMFullAccess 政策中定義許可的使用者 AWS 身分登入 。然後，您可以使用 IAM 主控台為需要存取 Stacks 的每個人建立 IAM 使用者。 AWS OpsWorks 然後，您可以將這些使用者匯入 Stacks AWS OpsWorks ，並授予使用者許可，如下所示：

一般 Stacks AWS OpsWorks 使用者

一般使用者不需要連接政策。如果他們確實有一個，則通常不包含任何 AWS OpsWorks Stacks 許可。反之，請使用 AWS OpsWorks Stacks 許可頁面，以stack-by-stack方式將下列其中一個許可層級指派給一般使用者。

- Show (顯示) 許可允許使用者檢視堆疊，但不允許其執行任何操作。
- Deploy (部署) 許可包含 Show (顯示) 許可，並且也會允許使用者部署及更新應用程式。
- Manage (管理) 許可包含 Deploy (部署) 許可，但也會允許使用者執行堆疊管理操作 (例如新增 layer 或執行個體、使用 Permissions (許可) 頁面設定使用者許可，以及啟用其自身的 SSH/RDP 和 sudo/admin 權限)。
- Deny (拒絕) 許可會拒絕存取堆疊。

如果這些許可層級不如您想要的特定使用者，您可以套用 IAM 政策來自訂使用者的許可。例如，您可能想要使用 AWS OpsWorks Stacks 許可頁面來指派管理許可層級給使用者，以授予他們執行所有堆疊管理操作的許可，但不能建立或複製堆疊。然後，您可以套用政策，透過拒絕這些許可來限制這些許可，以新增或刪除圖層，或透過允許它們建立或複製堆疊來增強這些許可。如需詳細資訊，請參閱連接 AWS OpsWorks IAM 政策來管理 Stacks 許可。

AWS OpsWorks Stacks 管理使用者

管理使用者是帳戶擁有者或具有 AWSOpsWorks_FullAccess 政策所定義許可的 IAM 使用者。除了授予給 Manage (管理) 使用者的許可之外，此政策還包含無法透過 Permissions (許可) 頁面授予的動作許可，例如以下項目：

- 將使用者匯入 AWS OpsWorks Stacks
- 建立和複製堆疊

如需取得完整的政策，請參閱範例政策。如需只能透過套用 IAM 政策授予使用者之許可的詳細清單，請參閱 AWS OpsWorks Stacks 許可層級。

主題

- 使用者和區域
- 建立 AWS OpsWorks Stacks 管理使用者
- 為 Stacks 建立 IAM AWS OpsWorks 使用者
- 將使用者匯入 AWS OpsWorks Stacks
- 編輯 AWS OpsWorks Stacks 使用者設定

使用者和區域

AWS OpsWorks Stacks 使用者可在其建立所在的區域端點中使用。您可以在下列任何區域中建立使用者。

- 美國東部 (俄亥俄) 區域
- 美國東部 (維吉尼亞北部) 區域
- 美國西部 (奧勒岡) 區域
- 美國西部 (加利佛尼亞北部) 區域
- 加拿大 （ 中部 ） 區域 （ 僅限 API；不適用於 AWS Management Console
- 亞太 (孟買) 區域
- 亞太區域 (新加坡) 區域
- 亞太 (雪梨) 區域
- 亞太 (東京) 區域
- 亞太區域 (首爾) 區域

- 歐洲 (法蘭克福) 區域
- 歐洲 (愛爾蘭) 區域
- 歐洲 (倫敦) 區域
- 歐洲 (巴黎) 區域
- 南美洲 (聖保羅) 區域

當您將使用者匯入 Stacks AWS OpsWorks 時，您可以將他們匯入其中一個區域端點；如果您希望使用者可在多個區域中使用，則必須將使用者匯入該區域。您也可以將 AWS OpsWorks Stacks 使用者從一個區域匯入到另一個區域；如果您將使用者匯入到已有相同名稱使用者的區域，則匯入的使用者會取代現有的使用者。如需匯入使用者的詳細資訊，請參閱匯入使用者。

建立 AWS OpsWorks Stacks 管理使用者

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以透過將`AWSOpsWorks_FullAccess`政策新增至使用者來建立 AWS OpsWorks Stacks 管理使用者，這會授予該使用者 AWS OpsWorks Stacks 完整存取許可。如需建立管理使用者的詳細資訊，請參閱建立管理使用者。

> **Note**
>
> AWSOpsWorks_FullAccess 政策允許使用者建立和管理 AWS OpsWorks Stacks 堆疊，但使用者無法為堆疊建立 IAM 服務角色；他們必須使用現有角色。建立堆疊的第一個使用者必須具有額外的 IAM 許可，如中所述 管理許可。當此使用者建立第一個堆疊時， AWS OpsWorks Stacks 會建立具有所需許可的 IAM 服務角色。之後，任何具有 `opsworks:CreateStack` 許可的使用者皆能使用該角色建立額外的堆疊。如需詳細資訊，請參閱允許 AWS OpsWorks Stacks 代表您。

建立使用者時，您可以視需要新增其他客戶管理政策來微調使用者的許可。例如，您可能希望管理使用者能夠建立或刪除堆疊，但無法匯入新的使用者。如需詳細資訊，請參閱連接 AWS OpsWorks IAM 政策來管理 Stacks 許可。

如果您有多個管理使用者，而不是為每個使用者分別設定許可，您可以將 AWSOpsWorks_FullAccess 政策新增至 IAM 群組，並將使用者新增至該群組。

如需建立群組的詳細資訊，請參閱建立 IAM 使用者群組。當您建立群組時，請新增 AWSOpsWorks_FullAccess 政策。您也可以新增 AdministratorAccess 政策，其中包含 AWSOpsWorks_FullAccess 許可。

如需將許可新增至現有群組的詳細資訊，請參閱將政策連接至 IAM 使用者群組。

為 Stacks 建立 IAM AWS OpsWorks 使用者

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您必須先建立 IAM 使用者，才能將他們匯入 AWS OpsWorks Stacks。您可以使用 IAM 主控台、命令列或 API 來執行此操作。如需完整說明，請參閱在 AWS 帳戶中建立 IAM 使用者。

請注意，與管理使用者不同，您不需要連接政策以定義許可。您可以在將使用者匯入 AWS OpsWorks Stacks 之後設定許可，如管理使用者許可中所解釋。

如需建立 IAM 使用者和群組的詳細資訊，請參閱 IAM 入門。

將使用者匯入 AWS OpsWorks Stacks

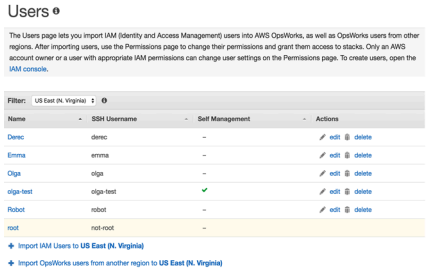
> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

管理使用者可以將使用者匯入 AWS OpsWorks Stacks；也可以將 AWS OpsWorks Stacks 使用者從一個區域端點匯入至另一個區域端點。當您將使用者匯入至 AWS OpsWorks Stacks 時，您可以將他們匯入至其中一個 AWS OpsWorks Stacks 區域端點。如果您希望使用者可在多個區域中使用，您必須將使用者匯入該區域。

雖然無法在主控台中明確匯入聯合身分使用者，但聯合身分使用者可以透過選擇 Stacks AWS OpsWorks 主控台右上角的我的設定，然後選擇右上角的使用者，來隱含地建立使用者設定檔。在 使用者 頁面上，聯合身分使用者 - 其帳戶是使用 API 或 CLI 建立，或隱含地透過主控台 - 可以與非聯合身分使用者類似地管理其帳戶。

將使用者匯入 AWS OpsWorks Stacks

1. 以管理使用者或帳戶擁有者身分登入 AWS OpsWorks Stacks。
2. 選擇位於右上方的 Users (使用者) 以開啟 Users (使用者) 頁面。



3. 選擇將 IAM 使用者匯入 **<*region name*>** ，以顯示可用但尚未匯入的使用者。

4. 填滿 Select all (選取全部) 核取方塊，或是選取一或多個個別使用者。當您完成時，請選擇 Import to OpsWorks (匯入 OpsWorks)。

> **Note**
>
> 將使用者匯入 AWS OpsWorks Stacks 之後，如果您使用 IAM 主控台或 API 將使用者從您的帳戶中刪除，使用者不會自動失去您透過 AWS OpsWorks Stacks 授予的 SSH 存取權。您還必須開啟使用者頁面，然後在使用者的動作欄中選擇刪除，從 AWS OpsWorks Stacks 刪除使用者。

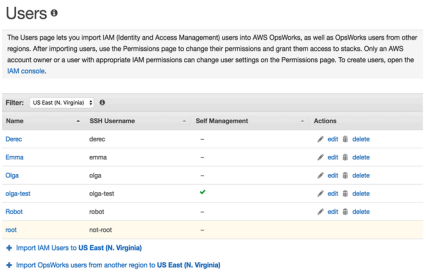
將 AWS OpsWorks Stacks 使用者從一個區域匯入到另一個區域

AWS OpsWorks Stacks 使用者可在其建立所在的區域端點中使用。您可以在 中顯示的區域中建立使用者使用者和區域。

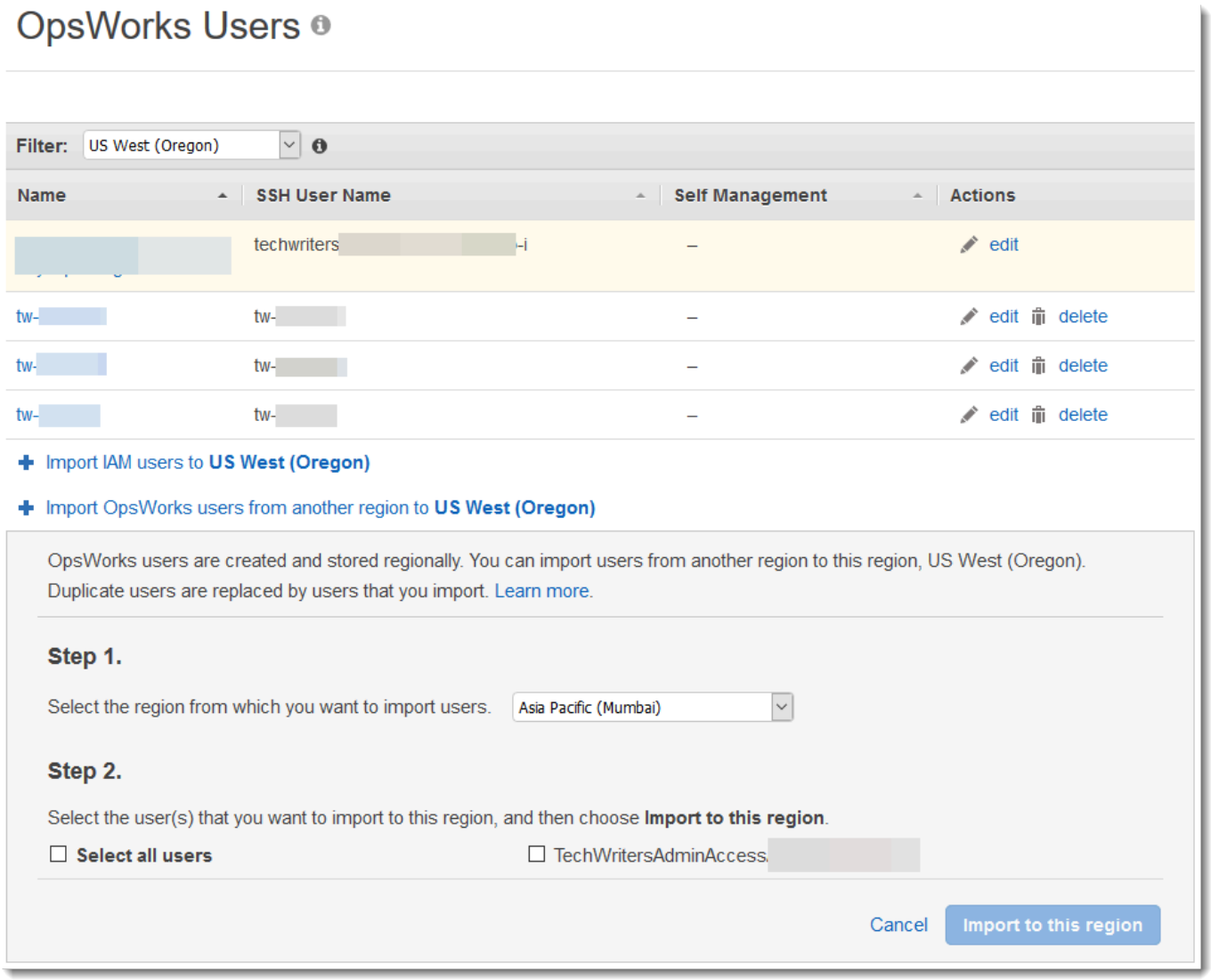
您可以將 AWS OpsWorks Stacks 使用者從一個區域匯入目前篩選使用者清單的區域。如果您將使用者匯入已有相同名稱使用者的 區域，則匯入的使用者會取代現有的使用者。

1. 以管理使用者或帳戶擁有者身分登入 AWS OpsWorks Stacks。

2. 選擇位於右上方的 Users (使用者) 以開啟 Users (使用者) 頁面。如果您在多個區域中有 AWS OpsWorks Stacks 使用者，請使用篩選條件控制項來篩選您要匯入使用者的 區域。



3. 選擇從另一個區域將 AWS OpsWorks Stacks 使用者匯入至 <***current region***>。



4. 選取您要從中匯入 Stacks AWS OpsWorks 使用者的 區域。

5. 選取要匯入的一或多個使用者，或是選取所有使用者，然後選擇 Import to this region (匯入此區域)。等待 AWS OpsWorks Stacks 在使用者清單中顯示匯入的使用者。

Unix 外部堆疊建立IDs AWS OpsWorks 和使用者

AWS OpsWorks 在 AWS OpsWorks 2000 到 4000 之間指派 Stacks 執行個體 Unix ID (UID) 值上的使用者。由於 AWS OpsWorks 會保留 2000-4000 範圍的 UIDs，因此您在 AWS OpsWorks （例如，透過使用技術指南配方，或從 IAM AWS OpsWorks 將使用者匯入至）之外建立的使用者，可以擁有由 AWS OpsWorks Stacks 為其他使用者覆寫的 UIDs。這可能會導致您在 AWS OpsWorks Stacks 外部建立的使用者未出現在資料包搜尋結果中，或被排除在 AWS OpsWorks Stacks 內建`sync_remote_users`操作之外。

外部程序也可以使用 Stacks 可以覆寫UIDs AWS OpsWorks 建立使用者。舉例來說，有些作業系統套件可能會建立使用者，做為後安裝程序的一部分。當您或軟體程序在 Linux 作業系統上建立使用者時，而不明確指定 UID，這是預設值，Stacks AWS OpsWorks 指派的 UID 是 *##### AWS OpsWorks UID>* + 1。

最佳實務是，在 AWS OpsWorks Stacks 主控台中 AWS CLI或使用 AWS SDK 建立 AWS OpsWorks Stacks 使用者並管理其存取權。如果您在外部的 AWS OpsWorks Stacks 執行個體上建立使用者 AWS OpsWorks，請使用大於 4000 的 *UnixID* 值。

編輯 AWS OpsWorks Stacks 使用者設定

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

在您匯入使用者之後，您可以編輯其設定，如下所示：

編輯使用者設定

1. 在 Users (使用者) 頁面上，在使用者的 Actions (動作) 資料行中選擇 edit (編輯)。
2. 您可以指定下列設定。

   自我管理

   選取 Yes (是) 以允許使用者使用 MySettings 頁面指定自身的 SSH 金鑰。

> **Note**
>
> 您也可以將 IAM 政策新增至授予 DescribeMyUserProfile 和 UpdateMyUserProfile 動作許可的 IAM 身分，以啟用自我管理。

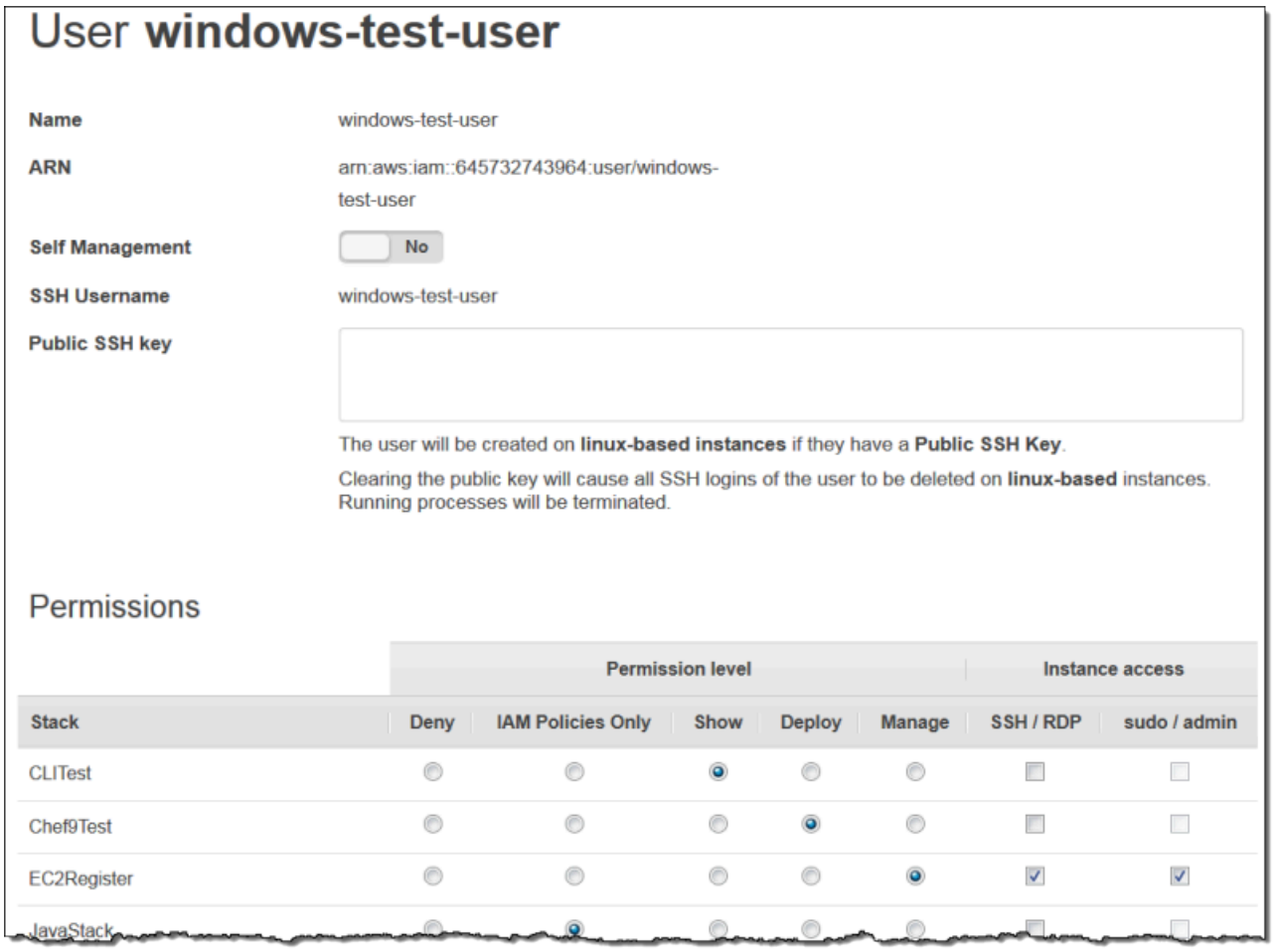
公有 SSH 金鑰

(選用) 輸入使用者的公有 SSH 金鑰。此金鑰會顯示在使用者的 My Settings (我的設定) 頁面上。若您啟用自我管理，使用者可編輯 My Settings (我的設定) 並指定其自身的金鑰。如需詳細資訊，請參閱註冊使用者的公有 SSH 金鑰。

AWS OpsWorks Stacks 會在所有 Linux 執行個體上安裝此金鑰；使用者可以使用相關聯的私有金鑰登入。如需詳細資訊，請參閱使用 SSH 登入。您無法搭配 Windows 堆疊使用此金鑰。

許可

(選擇性) 在同一處設定使用者每個堆疊的許可層級，而非透過使用每個堆疊的 Permissions (許可) 頁面分別設定他們。如需許可層級的詳細資訊，請參閱授予每個堆疊的許可。



授予 Stacks AWS OpsWorks 使用者每個堆疊的許可

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

管理 AWS OpsWorks Stacks 使用者許可的最簡單方法是使用堆疊的許可頁面。每個堆疊都有自己的頁面，可授予該堆疊的許可。

您必須以管理使用者身分或 Manage (管理) 使用者登入，才能修改任何許可設定。此清單只會顯示已匯入 Stacks AWS OpsWorks 的使用者。如需如何建立和匯入使用者的資訊，請參閱管理使用者。

預設許可層級是僅限 IAM 政策，僅授予使用者在其 IAM 政策中的那些許可。

- 當您從 IAM 或其他區域匯入使用者時，系統會將使用者新增至具有 IAM 僅政策許可層級之所有現有堆疊的清單。
- 根據預設，您剛從另一個區域匯入的使用者無法存取目的地區域中的堆疊。如果您從另一個區域匯入使用者，若要讓他們管理目的地區域中的堆疊，則必須在您匯入使用者後將許可指派給這些堆疊。
- 當您建立新的堆疊時，所有目前的使用者都會新增至清單，並帶有 IAM Policies Only (僅限 IAM 政策) 許可層級。

主題

- 設定使用者的許可
- 檢視您的許可
- 使用 IAM 條件金鑰驗證暫時登入資料

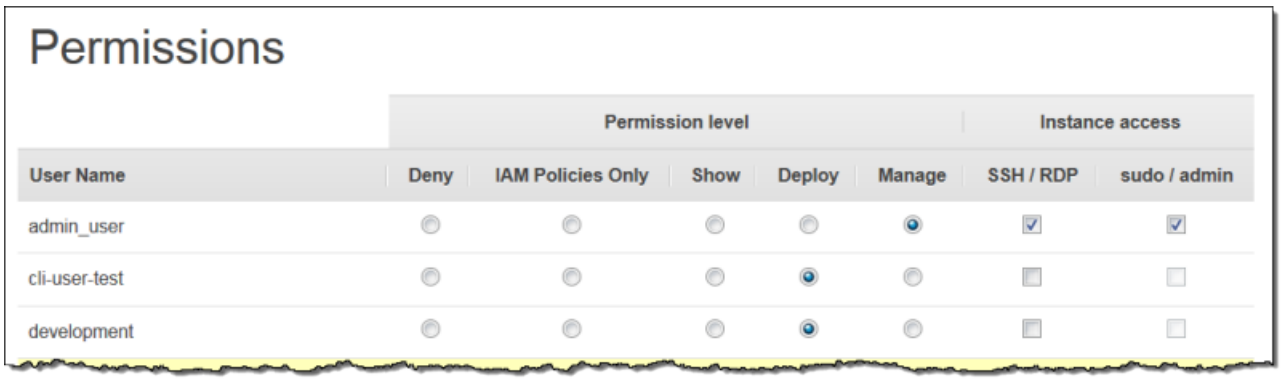
設定使用者的許可

設定使用者的許可

1. 在導覽窗格中，選擇 Permissions (許可)。
2. 在 Permissions (許可) 頁面上，選擇 Edit (編輯)。
3. 變更 Permission level (許可層級) 和 Instance access (執行個體存取) 設定。

    - 使用 Permissions level (許可層級) 設定指派其中一個標準許可層級給每個使用者。該許可層級會判斷使用者是否可以存取堆疊，以及使用者可執行的動作為何。如果使用者具有 IAM 政策，AWS OpsWorks Stacks 會評估兩組許可。如需範例，請參閱範例政策。
    - Instance access (執行個體存取) SSH/RDP 設定會指定使用者是否具有堆疊執行個體的 SSH (Linux) 或 RDP (Windows) 存取。

      若您授權 SSH/RDP 存取，您可以選擇性的選取 sudo/admin，授予使用者堆疊執行個體上的 sudo (Linux) 或管理 (Windows) 權限。



您可以將每個使用者指派給下列其中一個許可層級。如需每個層級允許動作的清單，請參閱[AWS OpsWorks Stacks 許可層級]。

拒絕

使用者無法在堆疊上執行任何 AWS OpsWorks Stacks 動作，即使他們擁有授予 Stacks 完整存取許可的 IAM AWS OpsWorks 政策。舉例來說，您可能會使用此層級來拒絕某些使用者存取未發行產品的堆疊。

僅限 IAM 政策

預設層級，這是指派給所有新匯入使用者，以及新建立堆疊中所有使用者的層級。使用者的許可取決於其 IAM 政策。如果使用者沒有 IAM 政策，或其政策沒有明確的 AWS OpsWorks Stacks 許可，則無法存取堆疊。管理使用者通常會獲指派此層級，因為其 IAM 政策已授予完整存取許可。

Show (顯示)

使用者可檢視堆疊，但無法執行任何操作。例如，管理人員可能希望監控帳戶的堆疊，但不需要部署應用程式或以任何方式修改堆疊。

部署

包含 Show (顯示) 許可，同時也會允許使用者部署應用程式。例如，應用程式開發人員可能需要將更新部署到堆疊的執行個體，但不需要將 layer 或執行個體新增至堆疊。

Manage (管理)

包含 Deploy (部署) 許可，但同時也會允許使用者執行各種堆疊管理操作，包含：

- 新增或刪除 layer 和執行個體。
- 使用堆疊的 Permissions (許可) 頁面將許可層級指派給使用者。

- 註冊或取消註冊資源。

例如，每個堆疊可以有一名指定管理人員，負責確認堆疊具有適當數目及類型的執行個體、處理套件和作業系統更新等。

> **Note**
>
> 管理層級無法讓使用者建立或複製堆疊。這些許可必須由 IAM 政策授予。如需範例，請參閱「管理許可」。

如果使用者也有 IAM 政策， AWS OpsWorks Stacks 會評估兩組許可。這可讓您將許可層級指派給使用者，然後套用政策來限制或擴增層級的允許動作。例如，您可以套用允許管理使用者建立或複製堆疊的政策，或拒絕該使用者註冊或取消註冊資源的能力。如需這類政策的一些範例，請參閱範例政策。

> **Note**
>
> 若使用者的政策允許額外的動作，其結果可能會覆寫 Permissions (許可) 頁面設定。例如，如果使用者的政策允許 CreateLayer 動作，但您使用許可頁面指定部署許可，則仍然允許使用者建立層。此規則的例外是拒絕選項，即使使用者使用 AWSOpsWorks_FullAccess 政策，此選項也會拒絕堆疊存取。如需詳細資訊，請參閱使用 政策控制對 AWS 資源的存取。

檢視您的許可

若已啟用 self-management，使用者可透過選擇位於右上角的 My Settings (我的設定)，來檢視其在每個堆疊上的許可層級。如果使用者的政策授予 DescribeMyUserProfile 和 UpdateMyUserProfile 動作的許可，使用者也可以存取 My Settings。

使用 IAM 條件金鑰驗證暫時登入資料

AWS OpsWorks Stacks 具有內建的授權層，可支援其他授權案例 （例如簡化管理個別使用者的唯讀或讀寫堆疊存取）。此授權 layer 依存於使用暫時登入資料。因此，您無法使用 `aws:TokenIssueTime`條件來驗證使用者是否使用長期登入資料，或封鎖使用臨時登入資料的使用者採取的動作，如 IAM 文件中的 IAM JSON 政策元素參考所述。

連接 AWS OpsWorks IAM 政策來管理 Stacks 許可

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以連接 IAM AWS OpsWorks 政策來指定使用者的 Stacks 許可。連接的政策為某些許可的必要項目：

- 管理使用者許可 (例如匯入使用者)。
- 某些動作的許可 (例如建立或複製堆疊)。

如需需要已連接政策的完整動作清單，請參閱AWS OpsWorks Stacks 許可層級。

您也可以使用政策來自訂透過許可頁面授予的許可層級。本節提供如何將 IAM 政策套用至使用者以指定 Stacks AWS OpsWorks 許可的簡短摘要。如需詳細資訊，請參閱存取 AWS 資源的管理。

IAM 政策是包含一或多個陳述式的 JSON 物件。每個陳述式元素皆有一個許可清單，其具備自身三個基本元素：

Action

許可影響的動作。您可以將 AWS OpsWorks Stacks 動作指定為 `opsworks:action`。Action 可設為特定動作，例如 `opsworks:CreateStack`，這會指定是否允許使用者呼叫 `CreateStack`。您也可以使用萬用字元指定動作群組。例如，`opsworks:Create*` 會指定所有建立動作。如需 Stacks AWS OpsWorks 動作的完整清單，請參閱 AWS OpsWorks Stacks API 參考。

效果

允許或拒絕指定動作。

Resource

許可影響 AWS 的資源。 AWS OpsWorks 堆疊有一個資源類型，即堆疊。若要指定特定堆疊資源的許可，請將 Resource 設為堆疊的 ARN，其格式如下：`arn:aws:opsworks:region:account_id:stack/stack_id/`。

您也可以使用萬用字元。例如，將 Resource 設為 * 會授予每個資源的許可。

例如，下列政策會拒絕使用者停止 ID 為 2860-2f18b4cb-4de5-4429-a149-ff7da9f0d8ee 之堆疊上的執行個體。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "opsworks:StopInstance",
      "Effect": "Deny",
      "Resource": "arn:aws:opsworks:*:*:stack/2f18b4cb-4de5-4429-a149-ff7da9f0d8ee/"
    }
  ]
}
```

如需將許可新增至 IAM 使用者的詳細資訊，請參閱https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_change-permissions.html#users_change_permissions-add-console。

如需如何建立或修改 IAM 政策的詳細資訊，請參閱 IAM 中的政策和許可。如需 Stacks AWS OpsWorks 政策的一些範例，請參閱 範例政策。

範例政策

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節說明可套用至 Stacks 使用者的範例 IAM AWS OpsWorks 政策。

- 管理許可 說明用來授予許可給管理使用者的政策。
- 管理許可 和 部署許可顯示可套用至使用者以增強或限制管理和部署許可層級的政策範例。

AWS OpsWorks Stacks 透過評估 IAM 政策授予的許可以及許可頁面授予的許可，來決定使用者的許可。如需詳細資訊，請參閱使用 政策控制對 AWS 資源的存取。如需 Permissions (許可) 頁面許可的詳細資訊，請參閱AWS OpsWorks Stacks 許可層級。

管理許可

使用 IAM 主控台 https://console.aws.amazon.com/iam/：// 來存取 AWSOpsWorks_FullAccess 政策，將此政策連接至使用者，以授予他們執行所有 Stacks AWS OpsWorks 動作的許可。IAM 許可是允許管理使用者匯入使用者的必要許可。

您必須建立 IAM 角色，允許 AWS OpsWorks Stacks 代表您存取其他 AWS 資源，例如 Amazon EC2 執行個體。您通常會讓管理使用者建立第一個堆疊，並讓 Stacks AWS OpsWorks 為您建立角色，藉此處理此任務。您接著可以使用針對所有後續的堆疊使用該角色。如需詳細資訊，請參閱允許 AWS OpsWorks Stacks 代表您。

建立第一個堆疊的管理使用者必須具有 AWSOpsWorks_FullAccess 政策中未包含的一些 IAM 動作的許可。將下列許可新增至政策的 `Actions`區段。為了取得適當的 JSON 語法，請務必在動作之間新增逗號，並在動作清單結尾移除結尾逗號。

```
"iam:PutRolePolicy",
"iam:AddRoleToInstanceProfile",
"iam:CreateInstanceProfile",
"iam:CreateRole"
```

管理許可

Manage (管理) 許可 layer 級允許使用者執行各種堆疊管理動作，包含新增或刪除 layer。本主題說明數個政策，可用來管理使用者以增強或限制標準許可。

拒絕 Manage (管理) 使用者新增或刪除 layer。

您可以限制管理許可層級，以允許使用者執行所有管理動作，但使用下列 IAM 政策新增或刪除圖層除外。將 *region*、*account_id* 和 *stack_id* 取代為您的組態適用的值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "opsworks:CreateLayer",
        "opsworks:DeleteLayer"
      ],
      "Resource": "arn:aws:opsworks:region:account_id:stack/stack_id/"
    }
```

```
  ]
}
```

允許 Manage (管理) 使用者建立或複製堆疊

管理許可層級不允許使用者建立或複製堆疊。您可以套用下列 IAM 政策，變更管理許可以允許使用者建立或複製堆疊。將 *region* 和 *account_id* 取代為您的組態適用的值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetRolePolicy",
        "iam:ListRoles",
        "iam:ListInstanceProfiles",
        "iam:ListUsers",
        "opsworks:DescribeUserProfiles",
        "opsworks:CreateUserProfile",
        "opsworks:DeleteUserProfile"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:opsworks::account_id:stack/*/",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "opsworks.amazonaws.com"
        }
      }
    }
  ]
}
```

拒絕 Manage (管理) 使用者註冊或取消註冊資源。

管理許可層級允許使用者向堆疊註冊和取消註冊 Amazon EBS 和彈性 IP 地址資源。您可以限制管理許可，以允許使用者執行所有管理動作，但透過套用下列政策來註冊資源。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "opsworks:RegisterVolume",
        "opsworks:RegisterElasticIp"
      ],
      "Resource": "*"
    }
  ]
}
```

允許 Manage (管理) 使用者匯入使用者

管理許可層級不允許使用者將使用者匯入 AWS OpsWorks Stacks。您可以透過套用下列 IAM 政策來增強管理許可，以允許使用者匯入和刪除使用者。將 *region* 和 *account_id* 取代為您的組態適用的值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetRolePolicy",
        "iam:ListRoles",
        "iam:ListInstanceProfiles",
        "iam:ListUsers",
        "iam:PassRole",
        "opsworks:DescribeUserProfiles",
        "opsworks:CreateUserProfile",
        "opsworks:DeleteUserProfile"
      ],
      "Resource": "arn:aws:iam:region:account_id:user/*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "opsworks.amazonaws.com"
        }
      }
    }
  ]
```

```
}
```

部署許可

Deploy (部署) 許可層級不允許使用者建議或刪除應用程式。您可以透過套用下列 IAM 政策來增強部署許可，以允許使用者建立和刪除應用程式。將 *region*、*account_id* 和 *stack_id* 取代為您的組態適用的值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "opsworks:CreateApp",
        "opsworks:DeleteApp"
      ],
      "Resource": "arn:aws:opsworks:region:account_id:stack/stack_id/"
    }
  ]
}
```

AWS OpsWorks Stacks 許可層級

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節列出 AWS OpsWorks Stacks 許可頁面上顯示、部署和管理許可層級所允許的動作。它還包括一個動作清單，您只能透過將 IAM 政策套用至使用者來授予許可。

Show (顯示)

Show (顯示) 層級允許 `Describe`*XYZ* 命令，除了以下例外：

```
DescribePermissions
DescribeUserProfiles
```

```
DescribeMyUserProfile
DescribeStackProvisioningParameters
```

若管理使用者已啟用使用者的自我管理，Show (顯示) 使用者也可以使用 `DescribeMyUserProfile` 和 `UpdateMyUserProfile`。如需自我管理的詳細資訊，請參閱編輯使用者設定。

部署

除了 Show (顯示) 層級允許的動作之外，Deploy (部署) 層級還允許以下動作。

```
CreateDeployment
UpdateApp
```

Manage (管理)

除了 Deploy (部署) 和 Show (顯示) 層級允許的動作之外，Manage (管理) 層級還允許以下動作。

```
AssignInstance
AssignVolume
AssociateElasticIp
AttachElasticLoadBalancer
CreateApp
CreateInstance
CreateLayer
DeleteApp
DeleteInstance
DeleteLayer
DeleteStack
DeregisterElasticIp
DeregisterInstance
DeregisterRdsDbInstance
DeregisterVolume
DescribePermissions
DetachElasticLoadBalancer
DisassociateElasticIp
GrantAccess
GetHostnameSuggestion
RebootInstance
RegisterElasticIp
RegisterInstance
RegisterRdsDbInstance
RegisterVolume
```

```
SetLoadBasedAutoScaling
SetPermission
SetTimeBasedAutoScaling
StartInstance
StartStack
StopInstance
StopStack
UnassignVolume
UpdateElasticIp
UpdateInstance
UpdateLayer
UpdateRdsDbInstance
UpdateStack
UpdateVolume
```

需要 IAM 政策的許可

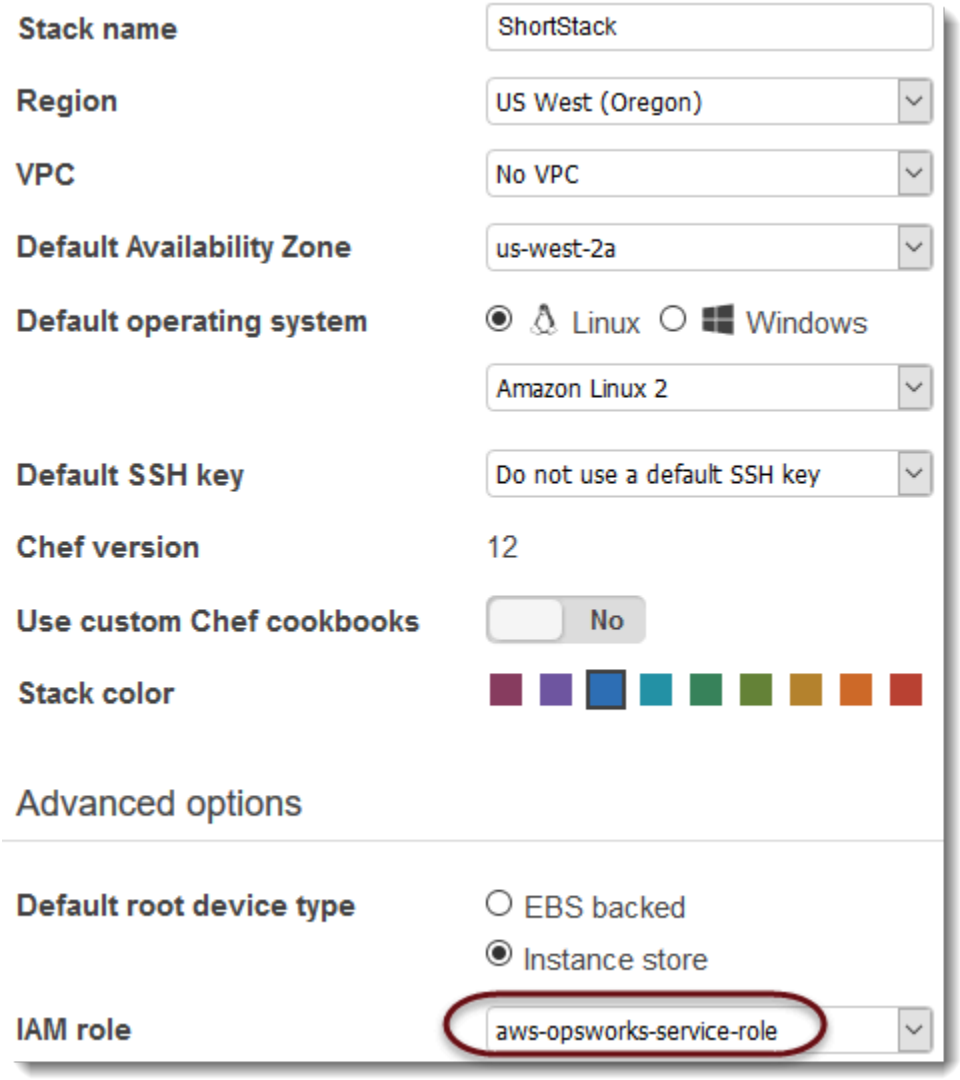
您必須套用適當的 IAM 政策給使用者，以授予下列動作的許可。如需一些範例，請參閱範例政策。

```
CloneStack
CreateStack
CreateUserProfile
DeleteUserProfile
DescribeUserProfiles
UpdateUserProfile
```

允許 AWS OpsWorks Stacks 代表您

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 需要代表您與各種 AWS 服務互動。例如， AWS OpsWorks Stacks 會與 Amazon EC2 互動以建立執行個體，並與 Amazon CloudWatch 互動以取得監控統計資料。當您建立堆疊時，您可以指定 IAM 角色，通常稱為服務角色，以授予 AWS OpsWorks Stacks 適當的許可。



當您指定新堆疊的服務角色時，您可以執行下列其中一項作業：

- 指定您先前建立的標準服務角色。

  您通常可以在建立您第一個堆疊時建立一個標準服務角色，然後針對所有後續的堆疊使用該角色。
- 指定您使用 IAM 主控台或 API 建立的自訂服務角色。

  如果您想要授予 AWS OpsWorks Stacks 比標準服務角色更多的有限許可，此方法很有用。

> **Note**
>
> 若要建立您的第一個堆疊，您必須在 IAM AdministratorAccess 政策範本中定義許可。這些許可會允許 AWS OpsWorks Stacks 建立新的 IAM 服務角色，允許您匯入使用者，如先前所述。

針對所有後續堆疊，使用者可選取為第一個堆疊建立的服務角色。他們不需要完整的管理許可也能建立堆疊。

標準服務角色會授予下列許可：

- 執行所有 Amazon EC2 動作 (`ec2:*`)。
- 取得 CloudWatch 統計資料 (`cloudwatch:GetMetricStatistics`)。
- 使用 Elastic Load Balancing 將流量分散到伺服器 (`elasticloadbalancing:*`)。
- 使用 Amazon RDS 執行個體做為資料庫伺服器 (`rds:*`)。
- 使用 IAM 角色 (`iam:PassRole`) AWS OpsWorks 在 Stacks 和 Amazon EC2 執行個體之間提供安全通訊。

如果您建立自訂服務角色，則必須確保其授予 Stacks AWS OpsWorks 管理堆疊所需的所有許可。下列 JSON 範本是標準服務角色的政策聲明；自訂服務角色應該在其政策聲明中至少包含以下許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:*",
                "iam:PassRole",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:DescribeAlarms",
                "ecs:*",
                "elasticloadbalancing:*",
                "rds:*"
            ],
            "Effect": "Allow",
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "ec2.amazonaws.com"
                }
            }
        }
    ]
```

```
}
```

服務角色也具有信任關聯。Stacks AWS OpsWorks 建立的服務角色具有下列信任關係。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "StsAssumeRole",
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

服務角色必須具有此信任關係， Stacks AWS OpsWorks 才能代表您採取行動。若您使用預設服務角色，請不要修改信任關聯。如果您要建立自訂服務角色，請執行下列其中一項動作來指定信任關係：

- 如果您在 IAM 主控台中使用建立角色精靈，請在選擇使用案例中選擇 Opsworks。此角色具有適當的信任關係，但不會隱含連接任何政策。若要授予 AWS OpsWorks Stacks 代表您行事的許可，請建立包含下列項目的客戶受管政策，並將其連接至新角色。

  ```
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "cloudwatch:DescribeAlarms",
          "cloudwatch:GetMetricStatistics",
          "ec2:*",
          "ecs:*",
          "elasticloadbalancing:*",
          "iam:GetRolePolicy",
          "iam:ListInstanceProfiles",
          "iam:ListRoles",
          "iam:ListUsers",
          "rds:*"
  ```

```
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "ec2.amazonaws.com"
        }
      }
    }
  ]
}
```

- 如果您使用的是 AWS CloudFormation 範本，您可以將類似下列內容新增至範本的資源區段。

```
"Resources": {
  "OpsWorksServiceRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
            "Statement": [ {
              "Effect": "Allow",
              "Principal": {
                  "Service": [ "opsworks.amazonaws.com" ]
              },
              "Action": [ "sts:AssumeRole" ]
            } ]
        },
        "Path": "/",
        "Policies": [ {
            "PolicyName": "opsworks-service",
            "PolicyDocument": {
              ...
            } ]
        }
      },
   }
```

```
}
```

Stacks AWS OpsWorks 中的跨服務混淆代理預防

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

混淆代理人問題屬於安全性議題，其中沒有執行動作許可的實體可以強制具有更多許可的實體執行該動作。在 中 AWS，跨服務模擬可能會導致混淆代理人問題。在某個服務 (呼叫服務) 呼叫另一個服務 (被呼叫服務) 時，可能會發生跨服務模擬。可以操縱呼叫服務來使用其許可，以其不應有存取許可的方式對其他客戶的資源採取動作。為了預防這種情況， AWS 提供的工具可協助您保護所有服務的資料，而這些服務主體已獲得您帳戶中資源的存取權。

我們建議在堆疊存取政策中使用 `aws:SourceArn`和 `aws:SourceAccount`全域條件內容索引鍵，以限制 AWS OpsWorks Stacks 為堆疊提供另一個服務的權限。如果 `aws:SourceArn` 值不包含帳戶 ID (例如 Amazon S3 儲存貯體 ARN)，您必須使用這兩個全域條件內容金鑰來限制許可。如果同時使用這兩個全域條件內容金鑰，且 `aws:SourceArn` 值包含帳戶 ID，則在相同政策陳述式中使用 `aws:SourceAccount` 值和 `aws:SourceArn` 值中的帳戶時，必須使用相同的帳戶 ID。`aws:SourceArn` 如果您希望只有一個堆疊與跨服務存取相關聯，請使用 。`aws:SourceAccount` 如果您想要允許該帳戶中的任何堆疊與跨服務使用相關聯，請使用 。

的值`aws:SourceArn`必須是 AWS OpsWorks 堆疊的 ARN。

防範混淆代理人問題的最有效方法是使用`aws:SourceArn`全域條件內容索引鍵搭配 Stacks 堆疊的完整 ARN AWS OpsWorks 。如果您不知道完整的 ARN，或如果您指定多個堆疊 ARNs，請將`aws:SourceArn`全域內容條件索引鍵與萬用字元 (*) 用於 ARN 的未知部分。例如：`arn:aws:`*`servicename`*`:*:`*`123456789012`*`:*`。

下一節說明如何使用 Stacks 中的 AWS OpsWorks `aws:SourceArn`和 `aws:SourceAccount`全域條件內容索引鍵來防止混淆代理人問題。

防止 Stacks AWS OpsWorks 中的混淆代理人入侵

本節說明如何協助防止 Stacks 中的 AWS OpsWorks 混淆代理人入侵，並包含許可政策範例，您可以將這些政策連接到您用來存取 AWS OpsWorks Stacks 的 IAM 角色。作為安全最佳實務，我們建議您

將 aws:SourceArn和 aws:SourceAccount條件金鑰新增至 IAM 角色與其他 服務的信任關係。信任關係可讓 AWS OpsWorks Stacks 擔任角色，在建立或管理 Stacks AWS OpsWorks 堆疊所需的其他服務中執行動作。

編輯信任關係以新增aws:SourceArn和aws:SourceAccount條件索引鍵

1. 開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在左側導覽窗格中，選擇 Roles (角色)。
3. 在搜尋方塊中，搜尋您用來存取 Stacks AWS OpsWorks 的角色。 AWS 受管角色為 aws-opsworks-service-role。
4. 在角色的摘要頁面上，選擇信任關係索引標籤。
5. 在信任關係索引標籤上，選擇編輯信任政策。
6. 在編輯信任政策頁面上，將至少一個 aws:SourceArn或 aws:SourceAccount條件索引鍵新增至政策。使用 aws:SourceArn將跨服務 （ 例如 Amazon EC2) 和 AWS OpsWorks Stacks 之間的信任關係限制為更嚴格的特定 AWS OpsWorks Stacks 堆疊。新增 aws:SourceAccount 以限制跨服務與 Stacks AWS OpsWorks 之間在特定帳戶中堆疊的信任關係，這較不嚴格。以下是範例。請注意，如果您使用兩個條件索引鍵，帳戶 IDs相同。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "arn:aws:opsworks:us-east-2:123456789012:stack/
EXAMPLEd-5699-40a3-80c3-22c32EXAMPLE/"
        }
      }
    }
  ]
}
```

7. 新增條件金鑰完成後，請選擇更新政策。

以下是使用 aws:SourceArn和 限制對堆疊存取的其他角色範例aws:SourceAccount。

主題

- 範例：存取特定區域中的堆疊
- 範例：將多個堆疊 ARN 新增至 aws:SourceArn

範例：存取特定區域中的堆疊

下列角色信任關係陳述式會存取美國東部 （俄亥俄） 區域 () 中的任何 AWS OpsWorks Stacks 堆疊us-east-2。請注意，區域是在 ARN 值 中指定aws:SourceArn，但堆疊 ID 值是萬用字元 (*)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:opsworks:us-east-2:123456789012:stack/*"
        }
      }
    }
  ]
}
```

範例：將多個堆疊 ARN 新增至 **aws:SourceArn**

下列範例限制存取帳戶 ID AWS OpsWorks 123456789012 中兩個 Stacks 堆疊的陣列。

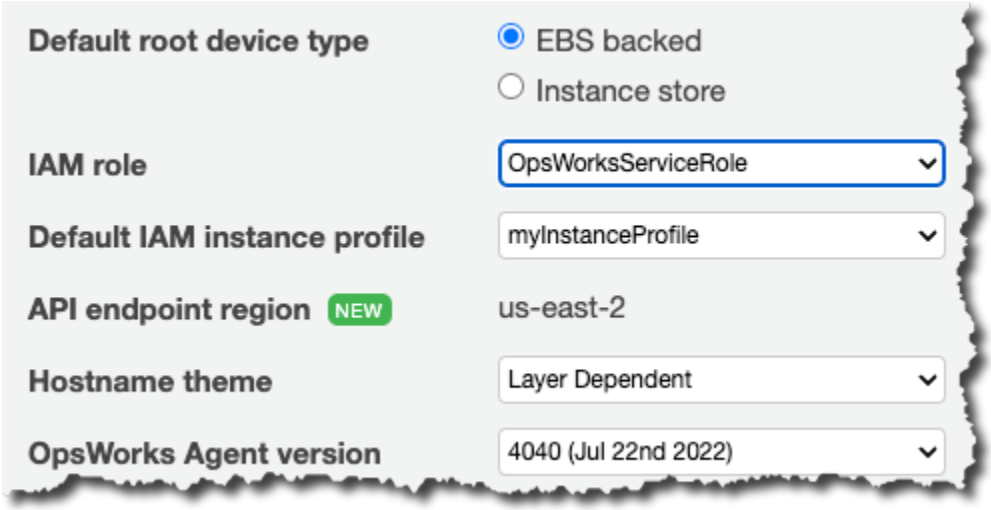
```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "opsworks.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnEquals": {
          "aws:SourceArn": [
             "arn:aws:opsworks:us-east-2:123456789012:stack/unique_ID1",
             "arn:aws:opsworks:us-east-2:123456789012:stack/unique_ID2"
           ]
       }
      }
    }
  ]
}
```

指定在 EC2 執行個體上執行之應用程式的許可

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

如果在您堆疊的 Amazon EC2 執行個體上執行的應用程式需要存取其他 AWS 資源，例如 Amazon S3 儲存貯體，則必須具有適當的許可。若要授予那些許可，您可以使用執行個體描述檔。您可以在建立 Stacks AWS OpsWorks 堆疊時為每個執行個體指定執行個體描述檔。



您也可以透過編輯 layer 組態，來指定 layer 執行個體的描述檔。

執行個體描述檔指定 IAM 角色。在執行個體上執行的應用程式可取得該角色來存取 AWS 資源，並受角色政策授予的許可約束。如需應用程式取得角色之方式的詳細資訊，請參閱使用 API 呼叫取得角色。

您可以透過下列任何一種方式建立執行個體描述檔：

- 使用 IAM 主控台或 API 來建立設定檔。

  如需詳細資訊，請參閱角色 (委派及聯合)。
- 使用 AWS CloudFormation 範本建立設定檔。

  如需如何在範本中包含 IAM 資源的一些範例，請參閱 Identity and Access Management (IAM) 範本片段。

執行個體描述檔必須具備信任關聯及授予存取 AWS 資源許可的連接政策。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
```

```
    }
  ]
}
```

執行個體描述檔必須具有此信任關係， Stacks AWS OpsWorks 才能代表您採取行動。若您使用預設服務角色，請不要修改信任關聯。若您建立自訂服務角色，請指定信任關聯如下：

- 若您使用 IAM 主控台中的 Create Role (建立角色) 精靈，請在精靈第二頁的 AWS Service Roles (AWS 服務角色) 下方輸入 Amazon EC2 角色。
- 如果您使用的是 AWS CloudFormation 範本，您可以將類似下列內容新增至範本的資源區段。

```
"Resources": {
      "OpsWorksEC2Role": {
         "Type": "AWS::IAM::Role",
         "Properties": {
            "AssumeRolePolicyDocument": {
               "Statement": [ {
                  "Effect": "Allow",
                  "Principal": {
                     "Service": [ "ec2.amazonaws.com" ]
                  },
                  "Action": [ "sts:AssumeRole" ]
               } ]
            },
            "Path": "/"
         }
      },
      "RootInstanceProfile": {
         "Type": "AWS::IAM::InstanceProfile",
         "Properties": {
            "Path": "/",
            "Roles": [ {
               "Ref": "OpsWorksEC2Role"
            }
         ]
      }
   }
}
```

當您建立執行個體描述檔時，您可以將適當的政策連接到該描述檔當時的角色。建立堆疊之後，您必須使用 IAM 主控台或 API 將適當的政策連接至設定檔的角色。例如，下列政策會授予 Amazon S3 儲

存貯體中名為 amzn-s3-demo-bucket 的所有物件的完整存取權。將 *region* 和 amzn-s3-demo-bucket 取代為您的組態適用的值。

```
{
  "Version": "2012-10-17",
  "Statement": [ {
    "Effect": "Allow",
    "Action": "s3:*",
    "Resource": "arn:aws:s3:region::amzn-s3-demo-bucket/*"
    }
  ]
}
```

如需如何建立和使用執行個體描述檔的範例，請參閱使用 Amazon S3 儲存貯體。

如果您的應用程式使用執行個體描述檔從 EC2 執行個體呼叫 AWS OpsWorks Stacks API，則除了 Stacks 和其他 AWS AWS OpsWorks 服務的適當`iam:PassRole`動作之外，政策還必須允許 動作。`iam:PassRole` 許可允許 AWS OpsWorks Stacks 代您取得服務角色。如需 AWS OpsWorks Stacks API 的詳細資訊，請參閱 AWS OpsWorks API 參考。

以下是 IAM 政策的範例，可讓您從 EC2 執行個體呼叫任何 AWS OpsWorks Stacks 動作，以及任何 Amazon EC2 或 Amazon S3 動作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:*",
        "s3:*",
        "opsworks:*",
        "iam:PassRole"
      ],
      "Resource": "arn:aws:ec2:region:account_id:instance/*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "opsworks.amazonaws.com"
        }
      }
    }
  ]
```

```
}
```

> **Note**
>
> 如果您不允許 `iam:PassRole`，任何呼叫 AWS OpsWorks Stacks 動作的嘗試都會失敗，並出現如下錯誤：
>
> ```
> User: arn:aws:sts::123456789012:federated-user/Bob is not authorized
> to perform: iam:PassRole on resource:
> arn:aws:sts::123456789012:role/OpsWorksStackIamRole
> ```

如需針對許可在 EC2 執行個體上使用角色的詳細資訊，請參閱《使用者指南》中的AWS Identity and Access Management 授予在 Amazon EC2 執行個體上執行的應用程式存取 AWS 資源。

管理 SSH 存取

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 支援 Linux 和 Windows 堆疊的 SSH 金鑰。

- 若是 Linux 執行個體，您可以使用 SSH 登入執行個體以執行代理程式 CLI 命令。

  如需詳細資訊，請參閱使用 SSH 登入。
- 若是 Windows 執行個體，您可以使用 SSH 金鑰取得該執行個體的管理員密碼，以用來登入 RDP。

  如需詳細資訊，請參閱使用 RDP 登入。

身分驗證是根據 SSH 金鑰對來進行；金鑰對中包含公有金鑰和私有金鑰：

- 您會在執行個體上安裝公有金鑰。

  位置取決於特定的作業系統，但 AWS OpsWorks Stacks 會為您處理詳細資訊。

- 您可將私有金鑰存放在本機，並將其提供給 SSH 用戶端 (例如 `ssh.exe`)，以存取執行個體。

  SSH 用戶端會使用私有金鑰連線到該執行個體。

若要將 SSH 存取權提供給堆疊的使用者，您需要一種可以建立 SSH 金鑰對、在堆疊的執行個體上安裝公有金鑰，以及安全地管理私有金鑰的方式。

Amazon EC2 提供在執行個體上安裝公有 SSH 金鑰的簡單方法。您可以使用 Amazon EC2 主控台或 API，為您計劃使用的每個 AWS 區域建立一或多個金鑰對。Amazon EC2 會將公有金鑰存放在 AWS 上，而您將私有金鑰存放在本機。當您啟動執行個體時，您可以指定其中一個區域的金鑰對，Amazon EC2 會自動將其安裝在執行個體上。然後，您即可使用對應的私有金鑰登入執行個體。如需詳細資訊，請參閱 Amazon EC2 金鑰對。

使用 AWS OpsWorks Stacks，您可以在建立堆疊時指定其中一個區域的 Amazon EC2 金鑰對，並在建立每個執行個體時選擇性地使用不同的金鑰對覆寫它。當 AWS OpsWorks Stacks 啟動對應的 Amazon EC2 執行個體時，它會指定金鑰對，Amazon EC2 會在執行個體上安裝公有金鑰。然後，您可以使用私有金鑰登入或擷取管理員密碼，就像使用標準 Amazon EC2 執行個體一樣。如需詳細資訊，請參閱安裝 Amazon EC2 金鑰。

使用 Amazon EC2 金鑰對很方便，但有兩個重大限制：

- Amazon EC2 金鑰對與特定 AWS 區域綁定。

  如果您在多個區域中工作，就必須管理多組金鑰對。
- 您只能在執行個體上安裝一個 Amazon EC2 金鑰對。

  如果您想要允許多位使用者登入，則所有使用者都必須具備私有金鑰的複本；這不是建議的安全做法。

對於 Linux 堆疊， AWS OpsWorks Stacks 提供更簡單且更靈活的方法來管理 SSH 金鑰對。

- 每位使用者可註冊個人金鑰對。

  它們會在本機存放私有金鑰，並向 AWS OpsWorks Stacks 註冊公有金鑰，如中所述註冊使用者的公有 SSH 金鑰。
- 在設定堆疊的許可時，您可以指定哪些使用者應具備堆疊執行個體的 SSH 存取權。

  AWS OpsWorks Stacks 會自動為每個授權使用者在堆疊的執行個體上建立系統使用者，並安裝其公有金鑰。使用者即可使用對應的私有金鑰登入，如使用 SSH 登入中所述。

使用個人 SSH 金鑰有下列優勢。

- 不需要手動設定執行個體上的金鑰； AWS OpsWorks Stacks 會自動在每個執行個體上安裝適當的公有金鑰。
- AWS OpsWorks Stacks 只會安裝授權使用者的個人公有金鑰。

  未經授權的使用者不能使用其個人私有金鑰來存取執行個體。使用 Amazon EC2 金鑰對，任何具有對應私有金鑰的使用者都可以登入，無論是否具有授權的 SSH 存取。
- 如果使用者不再需要 SSH 存取權，您可以使用 Permissions (許可) 頁面，撤銷使用者的 SSH/RDP 許可。

  AWS OpsWorks Stacks 會立即從堆疊的執行個體解除安裝公有金鑰。
- 您可以為任何 AWS 區域使用相同的金鑰。

  使用者只需要管理一個私有金鑰。
- 您不需要共享私有金鑰。

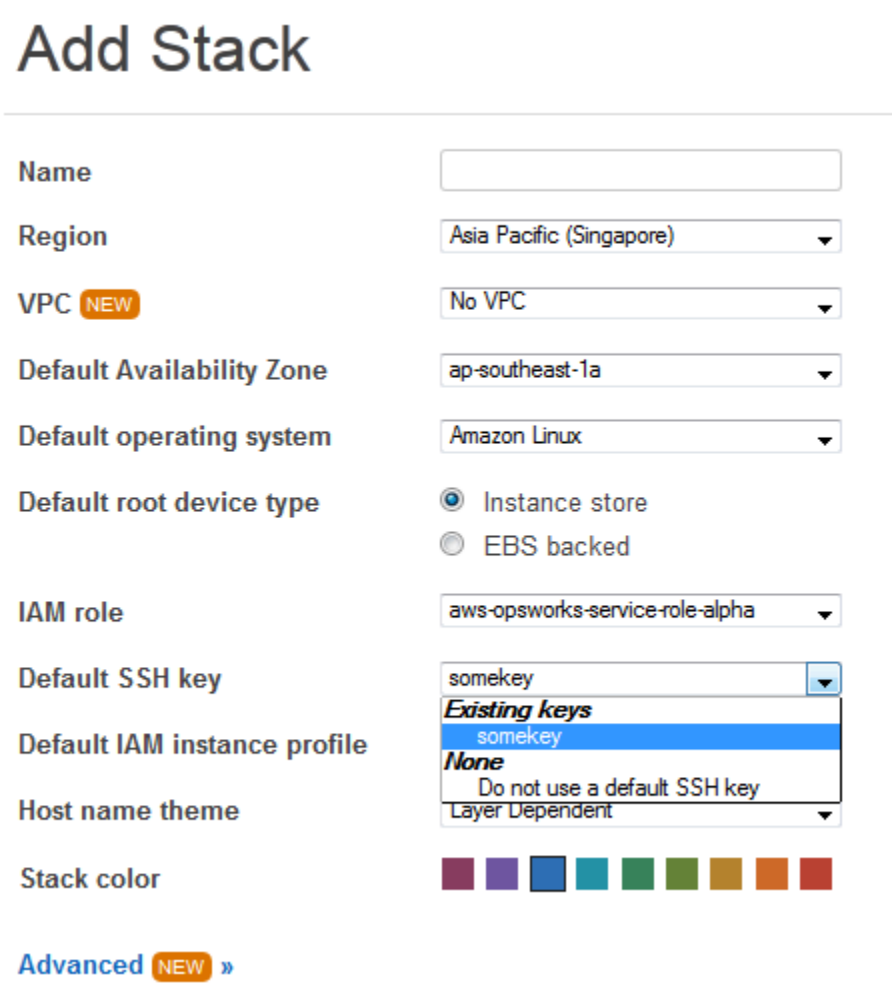
  每位使用者都有自己的私有金鑰。
- 輪換金鑰非常簡單。

  您或使用者可以在 My Settings (我的設定) 中更新公有金鑰，而 AWS OpsWorks Stacks 即會自動更新執行個體。

安裝 Amazon EC2 金鑰

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

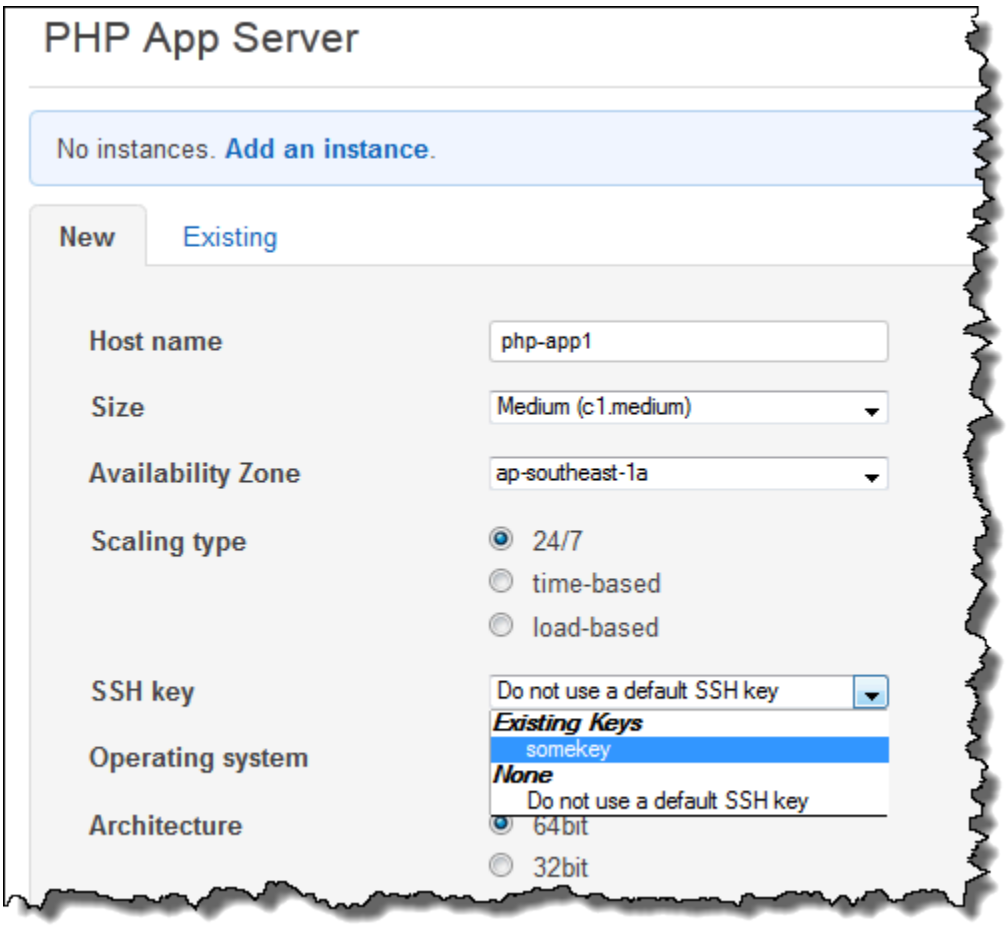
建立堆疊時，您可以指定 Amazon EC2 SSH 金鑰，該金鑰預設安裝在堆疊中的每個執行個體上。



預設 SSH 金鑰清單會顯示您 AWS 帳戶的 Amazon EC2keys。您可以執行下列任一作業：

- 從清單中選取適當的金鑰。
- 若不指定任何金鑰，請選取 Do not use a default SSH key (不使用預設 SSH 金鑰)。

如果您已選取 Do not use a default SSH key (不使用預設 SSH 金鑰)，或是您希望覆寫堆疊的預設金鑰，則可以在您建立執行個體時指定金鑰。



當您啟動執行個體時，Stacks AWS OpsWorks 會在 `authorized_keys` 檔案中安裝公有金鑰。

註冊使用者的公有 SSH 金鑰

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您有兩種方式可以註冊使用者的公有 SSH 金鑰：

- 管理使用者可以將公有 SSH 金鑰指派給一或多位使用者，並提供對應的私有金鑰給他們。
- 管理使用者可以啟用一或多位使用者的自我管理。

  這些使用者可以指定自己的公有 SSH 金鑰。

如需了解管理使用者如何啟用自行管理或將公有金鑰指派給使用者的詳細資訊，請參閱 編輯使用者設定。

若要在 PuTTY 終端機中使用 SSH 連線至 Linux 式執行個體，會需要額外的步驟。如需詳細資訊，請參閱 AWS 文件中的使用 PuTTY 從 Windows 連線至您的 Linux 執行個體和對您的執行個體連線進行故障診斷。

以下說明啟用自我管理的使用者如何指定其公有金鑰。

指定您的 SSH 公有金鑰

1. 建立 SSH 金鑰對。

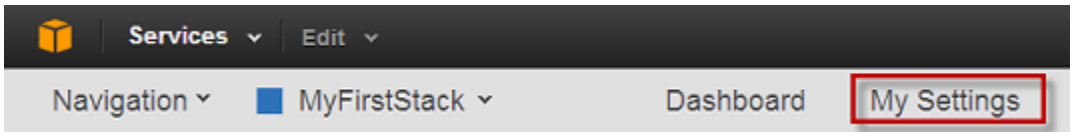
   最簡單的方法是本機產生金鑰對。如需詳細資訊，請參閱如何產生自己的金鑰並將它匯入 Amazon EC2。

   > **Note**
   >
   > 如果您使用 PuTTYgen 產生金鑰對，請從公有金鑰複製公有金鑰，以貼入 OpenSSH authorized_keys 檔案方塊。按一下 Save Public Key (儲存公有金鑰) 時，會將公有金鑰儲存為 MindTerm 不支援的格式。

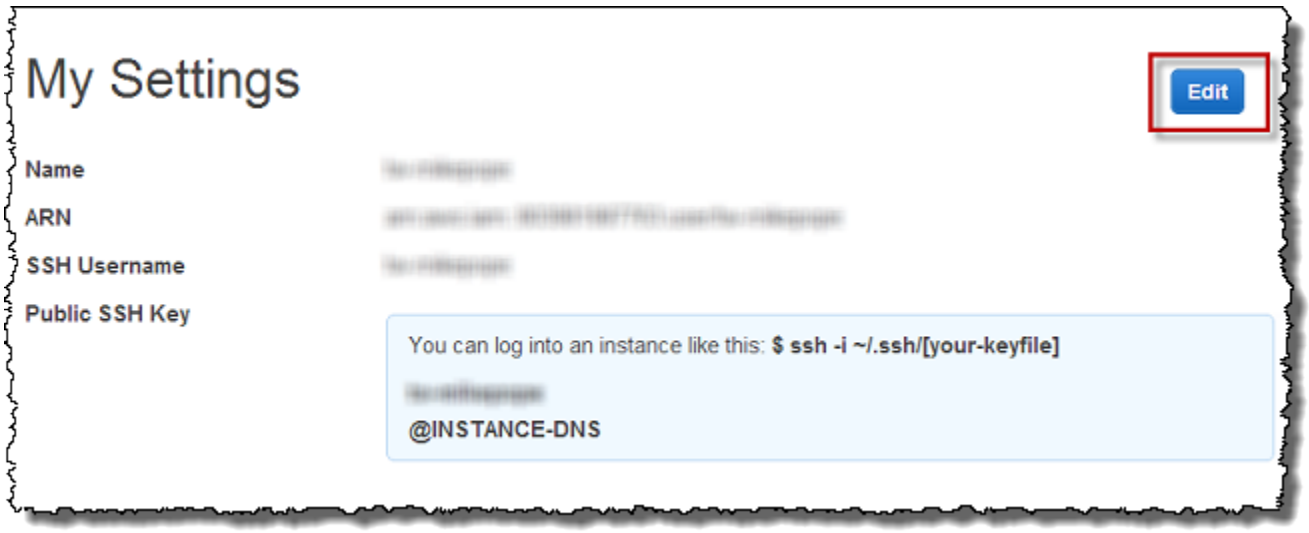
2. 以啟用自我管理的 IAM 使用者身分登入 AWS OpsWorks Stacks 主控台。

   > **Important**
   >
   > 如果您以帳戶擁有者身分登入，或是未啟用自我管理的 IAM 使用者身分登入， AWS OpsWorks Stacks 不會顯示我的設定。如果您是管理使用者或帳戶擁有者，則可以前往 Users (使用者) 頁面，並編輯使用者設定，來改為指定 SSH 金鑰。

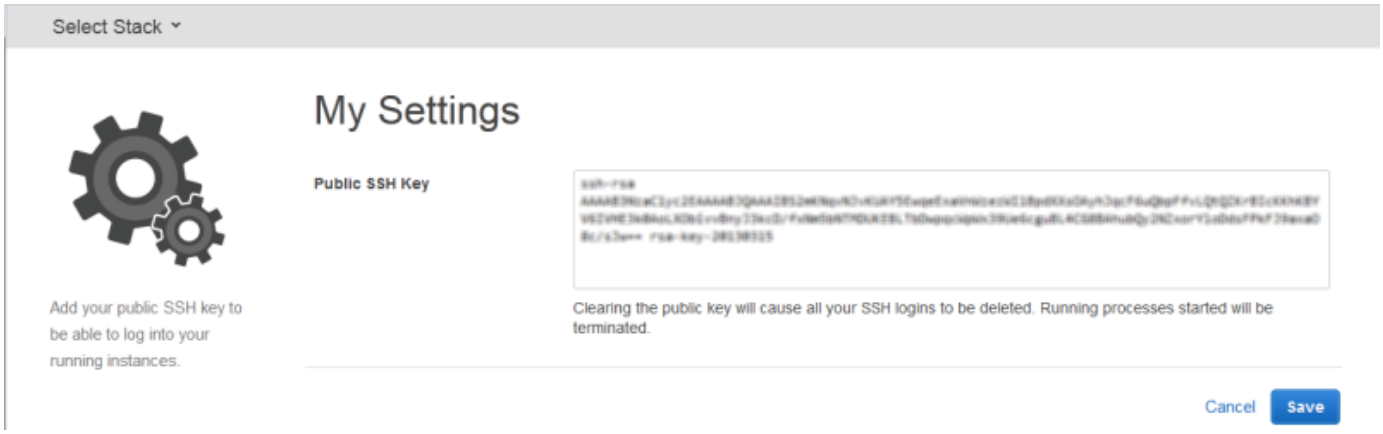
3. 選取我的設定，顯示登入使用者的設定。

4. 在 My Settings (我的設定) 頁面中，按一下 Edit (編輯)。



5. 在 Public SSH Key (公有 SSH 金鑰) 方塊中，輸入您的 SSH 公有金鑰，然後按一下 Save (儲存)。



> **Important**
>
> 若要使用內建的 MindTerm SSH 用戶端連線至 Amazon EC2 執行個體，使用者必須以 IAM 使用者身分登入，並使用 Stacks 註冊公有 SSH AWS OpsWorks 金鑰。如需詳細資訊，請參閱使用內建的 MindTerm SSH 用戶端。

管理 Linux 安全性更新

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

安全性更新

Linux 作業系統供應商會提供定期更新；其中大部分都是作業系統安全性修補程式，但也可能包含已安裝套件的更新。您應該確保您的執行個體作業系統處於最新狀態，並含有最新的安全性修補程式。

根據預設， AWS OpsWorks Stacks 會在安裝期間，在執行個體完成開機後自動安裝最新的更新。AWS OpsWorks Stacks 不會在執行個體上線後自動安裝更新，以避免中斷，例如重新啟動應用程式伺服器。反之，您可以自行管理線上執行個體的更新，將任何中斷降至最低。

我們建議您使用下列其中一個項目，來更新線上執行個體。

- 建立並啟動新的執行個體，以取代目前的線上執行個體。然後刪除目前的執行個體。

  新的執行個體會在設定期間安裝最新的一組安全性修補程式。
- 在 Chef 11.10 或較舊版本堆疊中的 Linux 類型執行個體上，執行更新相依性堆疊命令，以在指定的執行個體上安裝最新一組安全性修補程式和其他更新。

對於這兩種方法， AWS OpsWorks Stacks 會透過執行 `yum update` Amazon Linux 和 Red Hat Enterprise Linux (RHEL) 或 Ubuntu `apt-get update` 來執行更新。每個分佈處理更新的方式略有不同，因此您應該查看相關連結中的資訊，以確切了解更新會對您的執行個體造成哪些影響：

- Amazon Linux – Amazon Linux 更新會安裝安全修補程式，也可能安裝功能更新，包括套件更新。

  如需詳細資訊，請參閱 Amazon Linux AMI 常見問答集。
- Ubuntu – Ubuntu 更新主要僅限於安裝安全修補程式，但也可能針對有限數量的關鍵修正安裝套件更新。

  如需詳細資訊，請參閱 LTS - Ubuntu Wiki。
- CentOS – CentOS 更新通常會維持與舊版的二進位相容性。

- RHEL – RHEL 更新通常會維持與舊版的二進位相容性。

  如需詳細資訊，請參閱 Red Hat Enterprise Linux Life Cycle。

如果您想要對更新進行更多控制，例如指定特定的套件版本，您可以使用 CreateInstance、UpdateInstance、CreateLayer 或 UpdateLayer 動作，或同等的 AWS SDK 方法或 AWS CLI 命令，將 `InstallUpdatesOnBoot` 參數設定為 `false`，來停用自動更新。下列範例說明如何使用 AWS CLI 來停用 `InstallUpdatesOnBoot`，以做為現有 layer 的預設設定。

```
aws opsworks update-layer --layer-id layer ID --no-install-updates-on-boot
```

接著，您必須自行管理更新。例如，您可以採用下列其中一個策略：

- 實作自訂配方，其可執行適當的 shell 命令以安裝您慣用的更新。

  由於系統更新不會自然對應到生命週期事件，因此請將配方納入您的自訂技術指南中，但以手動方式執行。針對套件更新，您也可以使用 yum_package (Amazon Linux) 或 apt_package (Ubuntu) 資源，而不是 shell 命令。
- 使用 SSH 登入每個執行個體，並手動執行適當的命令。

使用安全群組

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

安全群組

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

每個 Amazon EC2 執行個體都有一或多個相關聯的安全群組，可管理執行個體的網路流量，就像防火牆一樣。安全群組有一或多個「規則」，每一個都會指定允許流量的特定類別。指定下列項目的規則：

- 允許流量的類型，例如 SSH 或 HTTP
- 此流量的通訊協定，例如 TCP 或 UDP
- 流量來源的 IP 地址範圍
- 流量的允許連接埠範圍

安全群組有兩種規則類型：

- 傳入規則管理傳入的網路流量。

  例如，應用程式伺服器執行個體通常有一個傳入規則允許來自任何 IP 地址的 HTTP 流量傳入連接埠 80，另一個傳入規則允許來自指定 IP 地址集的 SSH 流量傳入連接埠 22。
- 傳出規則管理傳出的網路流量。

  常用實務是使用允許任何傳出流量的預設設定。

如需安全群組的詳細資訊，請參閱 Amazon EC2 安全群組。

當您第一次在區域中建立堆疊時， AWS OpsWorks Stacks 會為每個 layer 建立內建的安全群組，並具有一組適當的規則。所有的群組都有允許所有傳出流量的預設傳出規則。一般而言，傳入規則允許下列項目：

- 從適當的 Stacks 層傳入 TCP、UDP AWS OpsWorks 和 ICMP 流量
- 連接埠 22 上的傳入 TCP 流量 (SSH 登入)

> **Warning**
>
> 預設的安全群組組態會向任何網路位置 (0.0.0.0/0) 開放 SSH (連接埠 22)。這可讓所有 IP 地址使用 SSH 存取您的執行個體。對於生產環境，您必須使用只允許特定 IP 地址或地址範圍之 SSH 存取的組態。在它們建立後立即更新預設的安全群組，或改用自訂的安全群組。

- 對於 Web 伺服器 layer，所有的傳入 TCP 以及 UDP 流量流向連接埠 80 (HTTP) 和 443 (HTTPS)

> **Note**
>
> 內建的 `AWS-OpsWorks-RDP-Server` 安全群組會指派給所有的 Windows 執行個體，以允許 RDP 存取。不過，根據預設，它沒有任何規則。如果您執行的是 Windows 堆疊，並想要使用 RDP 存取執行個體，您必須新增允許 RDP 存取的傳入規則。如需詳細資訊，請參閱使用 RDP 登入。

若要查看每個群組的詳細資訊，請前往 Amazon EC2 主控台，在導覽窗格中選取安全群組，然後選取適當的 layer 安全群組。例如，AWS-OpsWorks-Default-Server 是所有堆疊的預設內建安全群組，而 AWS-OpsWorks-WebApp 是 Chef 12 範例堆疊的預設內建安全群組。

> **Note**
>
> 如果您不小心刪除 Stacks AWS OpsWorks 安全群組，重新建立的偏好方法是讓 AWS OpsWorks Stacks 為您執行任務。只要在相同的 AWS 區域和 VPC 建立新的堆疊，如果有的話，Stacks AWS OpsWorks 就會自動重新建立所有內建安全群組，包括您刪除的安全群組。您接著可以刪除您不再需要使用的堆疊，安全群組仍會留下。如果您想要手動重新建立安全群組，它必須是和原始安全群組完全一致 (包含群組名稱大小寫) 的複本。
> 此外，如果發生下列任何情況， AWS OpsWorks Stacks 將嘗試重新建立所有內建安全群組：
>
> - 您可以在 Stacks AWS OpsWorks 主控台中對堆疊的設定頁面進行任何變更。
> - 您啟動其中一個堆疊的執行個體。
> - 您建立新的堆疊。

您可以使用以下任一種方法指定安全群組。您使用 Use OpsWorks security groups (使用 OpsWorks 安全群組) 設定在您建立堆疊時指定偏好。

- 是 （預設設定） – AWS OpsWorks Stacks 會自動將適當的內建安全群組與每個 layer 建立關聯。

  您可以新增自訂安全群組與您偏好的設定，微調 layer 的內建安全群組。不過，當 Amazon EC2 評估多個安全群組時，會使用限制性最低的規則，因此您無法使用此方法指定比內建群組更嚴格的規則。
- 否 – AWS OpsWorks Stacks 不會將內建安全群組與 layer 建立關聯。

您必須建立適當的安全群組，並建立至少一個安全群組與您所建之每個 layer 的關聯。使用此方法指定比內建群組更嚴格的規則。請注意，只要您想要，您仍然可以手動建立內建安全群組與 layer 的關聯；只有需要自訂設定的 layer 才需要自訂的安全群組。

> **Important**
>
> 如果您使用內建的安全群組，您即無法透過手動修改群組設定來建立更嚴格的規則。每次建立堆疊時， AWS OpsWorks Stacks 都會覆寫內建安全群組的組態，因此您所做的任何變更都會在下次建立堆疊時遺失。如果 layer 需要比內建安全群組更嚴格的安全群組設定，請將 Use OpsWorks security groups (使用 OpsWorks 安全群組) 設為 No (否)，並使用您偏好的設定建立自訂安全群組，然後在建立時將其指派給 layer。

AWS OpsWorks Chef 12 Linux 的 Stacks 支援

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節提供 AWS OpsWorks Stacks for Chef 12 Linux 的簡短概觀。如需 Windows 上 Chef 12 的資訊，請參閱「入門：Windows」。如需 Linux 上舊版 Chef 的資訊，請參閱「適用於 Linux 的 Chef 11.10 和較舊版本」。

概觀

AWS OpsWorks Stacks 支援適用於 Linux 堆疊的 Chef 最新版本 Chef 12。如需詳細資訊，請參閱 Learn Chef。

AWS OpsWorks Stacks 持續支援適用於 Linux 堆疊的 Chef 11.10。不過，如果您是進階的 Chef 使用者，並希望享有廣泛的社群 cookbook (技術指南) 選擇或撰寫您自己的自訂 cookbook (技術指南)，建議您使用 Chef 12。Chef 12 堆疊與 Chef 11.10 和較舊版 Linux 堆疊相比有下列優點：

- 兩個單獨的 Chef 執行 - 當在執行個體上執行命令時，Stacks AWS OpsWorks 代理程式現在會執行兩個隔離的 Chef 執行：一個執行任務，將執行個體與其他 AWS 服務整合，例如 AWS Identity and

Access Management (IAM)，另一個執行您的自訂技術指南。第一個 Chef 執行會在執行個體上安裝 AWS OpsWorks Stacks 代理程式，並執行系統任務，例如使用者設定和管理、磁碟區設定和組態、CloudWatch 指標的組態等。第二個執行則專門執行您在 AWS OpsWorks Stacks 生命週期事件方面的自訂配方。第二個執行可讓您使用自己的 Chef cookbook (技術指南) 或社群 cookbook (技術指南)。

- 解決命名空間衝突 - 在 Chef 12 以前的版本中， AWS OpsWorks Stacks 會在共享環境中執行系統任務，以及執行內建和自訂配方。這會導致命名空間衝突，並不清楚 Stacks AWS OpsWorks 已執行哪些配方。您必須手動覆寫不需要的預設組態，這項任務耗時又容易出錯。在適用於 Linux 的 Chef 12 中， AWS OpsWorks Stacks 不再支援 PHP、Node.js 或 Rails 等應用程式伺服器環境的內建 Chef 技術指南。透過消除內建配方， AWS OpsWorks Stacks 消除了命名內建配方與自訂配方之間的衝突問題。
- 對 Chef 社群技術指南的強大支援 – AWS OpsWorks Stacks Chef 12 Linux 為 Chef 超級市場的社群技術指南提供更高的相容性和支援。您現在可以使用優於 Stacks AWS OpsWorks 先前提供的內建技術指南的社群技術指南，這些技術指南旨在與最新的應用程式伺服器環境和架構搭配使用。您可以在適用於 Linux 的 Chef 12 上執行大部分 cookbook (技術指南)，而不需修改。如需詳細資訊，請前往 Learn Chef 網站上的 Chef Supermarket、Chef Supermarket 網站，以及 GitHub 上的 Chef Cookbooks 儲存庫。
- Timely Chef 12 更新 - AWS OpsWorks Stacks 會在每個 Chef 版本不久後將其 Chef 環境更新為最新的 Chef 12 版本。使用 Chef 12，次要 Chef 更新和新的 AWS OpsWorks Stacks 代理程式版本將同時發生。這可讓您直接測試新的 Chef 版本，並讓您的 Chef 配方和應用程式利用最新的 Chef 功能。

如需 Chef 12 以前之 Chef 支援版本的詳細資訊，請參閱「適用於 Linux 的 Chef 11.10 和較舊版本」。

移至 Chef 12

與舊版 Chef 11.10、11.4 AWS OpsWorks 和 0.9 的支援相比，Chef 12 Linux 的 Key Stacks 變更如下所示：

- 適用於 Linux 堆疊的 Chef 12 已不再提供或支援內建堆疊層。由於只會執行您的自訂配方，因此移除此支援可讓您全面了解執行個體的設定情況，並更輕鬆地撰寫及維護自訂 cookbook (技術指南)。例如，不再需要覆寫內建 Stacks AWS OpsWorks 配方的屬性。移除內建圖層也可讓 AWS OpsWorks Stacks 更好地支援 Chef 社群開發和維護的技術指南，以便您可以充分利用它們。適用於 Linux 的 Chef 12 不再提供的內建堆疊層類型包括：AWS Flow (Ruby)、Ganglia、HAProxy、Java 應用程式伺服器、Memcached、MySQL、Node.js 應用程式伺服器、PHP 應用程式伺服器、Rails 應用程式伺服器和靜態 Web 伺服器。

  - 由於 AWS OpsWorks Stacks 正在執行您提供的配方，因此不再需要透過執行自訂技術指南來覆寫內建 AWS OpsWorks 的 Stacks 屬性。若要覆寫您自己或社群配方中的屬性，請遵循 Chef 12 文件內 About Attributes 中的說明和範例。
- AWS OpsWorks Stacks 持續為 Chef 12 Linux 堆疊提供下列層的支援：
  - 自訂 Layer
  - Amazon RDS 服務層
  - ECS 叢集層
- Chef 12 Linux 的堆疊組態和資料包已變更，看起來與 Chef 12.2 Windows 的堆疊組態和資料包非常類似。這可讓您更輕鬆地查詢、分析這些資料包及排解其問題 (尤其如果您使用作業系統類型不同的堆疊)。請注意，Stacks AWS OpsWorks 不支援加密的資料包。若要以加密形式存放機密資料 (例如密碼或憑證)，建議您將它存放在私有 S3 儲存貯體中。然後，您可以建立自訂配方，定義其使用適用於 Ruby 的 Amazon 開發套件擷取資料。如需範例，請參閱「使用適用於 Ruby 的 SDK」。如需詳細資訊，請參閱「AWS OpsWorks Stacks Data Bag 參考」。
- 在 Chef 12 Linux 中，Berkshelf 不會再安裝到堆疊執行個體。相反地，建議您在本機開發機器上使用 Berkshelf，以在本機封裝您的 cookbook (技術指南) 相依性。然後將包含相依性的套件上傳至 Amazon Simple Storage Service。最後，修改您的 Chef 12 Linux 堆疊，使其使用上傳的套件做為技術指南來源。如需詳細資訊，請參閱本機封裝技術指南依存性。
- 不再支援 EBS 磁碟區的 RAID 組態。為了提高效能，您可以使用 Amazon Elastic Block Store (Amazon EBS) 的佈建 IOPS。
- 不再支援 autofs。
- 不再支援 Subversion 儲存庫。
- 各堆疊層的 OS 套件安裝現在必須透過自訂配方完成。如需詳細資訊，請參閱個別 layer 套件安裝。

支援的作業系統

Chef 12 支援與舊版 Chef 相同的 Linux 作業系統。如需 Chef 12 Linux 堆疊可以使用的 Linux 作業系統類型和版本清單，請參閱「Linux 作業系統」。

支援的執行個體類型

AWS OpsWorks Stacks 支援 Chef 12 Linux 堆疊的所有執行個體類型，除了特殊執行個體類型，例如高效能運算 (HPC) 叢集運算、叢集 GPU 和高記憶體叢集執行個體類型。

詳細資訊

若要進一步了解如何使用適用於 Linux 堆疊的 Chef 12，請參閱下列各章：

- 入門：範例

  透過引導您使用 AWS OpsWorks Stacks 主控台進行簡短的實作練習，以建立 Node.js 應用程式環境，來介紹 AWS OpsWorks Stacks。
- 入門：Linux

  向您介紹 AWS OpsWorks Stacks 和 Chef 12 Linux，方法是引導您使用 Stacks 主控台進行實作練習，以建立基本的 Chef 12 Linux AWS OpsWorks 堆疊，其中包含使用 Node.js 應用程式提供流量的簡單層。
- 自訂 Layer

  提供指導，用以將包含 cookbook (技術指南) 和配方的堆疊層新增至 Chef 12 Linux 堆疊。您可以使用 Chef 社群提供之現成可用的 cookbook (技術指南) 和配方，也可以自行建立。
- 移至資料包

  比較及比對執行 Chef 11 以前 (含) 版本及 Chef 12 之 Linux 堆疊使用的執行個體 JSON。同時提供 Chef 12 執行個體 JSON 格式的參考文件指標。

將堆疊設定屬性移至資料包

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 會向 Chef 配方公開各種堆疊設定。這些堆疊設定包含的值如：

- 堆疊技術指南來源 URL
- Layer 磁碟區組態
- 執行個體主機名稱
- Elastic Load Balancing DNS 名稱

- 應用程式來源 URL
- 使用者名稱

與直接在配方中硬編碼堆疊設定相比，參考配方中的堆疊設定能讓配方程式碼更強大且較不容易出錯。本主題說明如何存取這些堆疊設定，以及如何將它們從適用於 Linux 之 Chef 11.10 和舊版的屬性中移至 Chef 12 Linux 的資料包。

在適用於 Linux 的 Chef 11.10 和舊版中，堆疊設定可提供為 Chef 屬性，並可透過 Chef node 物件或透過 Chef 搜尋存取。這些屬性存放在 `/var/lib/aws/opsworks/chef` 目錄中一組 JSON 檔案的 AWS OpsWorks Stacks 執行個體上。如需詳細資訊，請參閱堆疊組態及部署屬性：Linux。

在 Chef 12 Linux 中，堆疊設定可提供為 Chef 資料包，但只能透過 Chef 搜尋存取。資料包存放在 `/var/chef/runs/run-ID/data_bags`目錄中一組 JSON 檔案的 AWS OpsWorks Stacks 執行個體上，其中 *`run-ID`* 是 AWS OpsWorks Stacks 指派給執行個體上每個 Chef 的唯一 ID。堆疊設定不再提供為 Chef 屬性，因此堆疊設定再也不能透過 Chef node 物件存取。如需詳細資訊，請參閱 AWS OpsWorks Stacks Data Bag 參考。

例如，在適用於 Linux 的 Chef 11.10 和舊版中，以下配方程式碼使用 Chef node 物件取得代表應用程式簡稱和來源 URL 的屬性。接著使用 Chef 日誌寫入這兩個屬性值：

```
Chef::Log.info ("********** The app's short name is '#{node['opsworks']
['applications'].first['slug_name']}' **********")
Chef::Log.info("********** The app's URL is '#{node['deploy']['simplephpapp']['scm']
['repository']}' **********")
```

在 Chef 12 Linux 中，以下配方程式碼使用 `aws_opsworks_app` 搜尋索引取得 `aws_opsworks_app` 資料包中第一個資料包項目的內容。然後，程式碼將兩個訊息寫入 Chef 日誌，一個有應用程式簡稱資料包內容，另一個有應用程式來源 URL 資料包內容：

```
app = search("aws_opsworks_app").first

Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}' **********")
```

若要將您存取堆疊設定配方程式碼，從適用於 Linux 的 Chef 11.10 和舊版遷移至 Chef 12 Linux，您必須修改您的程式碼，以：

- 存取 Chef 資料包，而非 Chef 屬性。
- 使用 Chef 搜尋，而非 Chef node 物件。

- 使用 AWS OpsWorks Stacks 資料包名稱，例如 `aws_opsworks_app`，而不是使用 AWS OpsWorks Stacks 屬性名稱，例如 `opsworks`和 `deploy`。

如需詳細資訊，請參閱 AWS OpsWorks Stacks Data Bag 參考。

在 Stacks 中支援先前的 Chef AWS OpsWorks 版本

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節提供先前 Chef AWS OpsWorks 版本的 Stacks 文件的簡短概觀。

適用於 Linux 的 Chef 11.10 和較舊版本

提供適用於 Linux AWS OpsWorks 堆疊之 Chef 11.10、11.4 和 0.9 的 Stacks 支援文件。

適用於 Linux 的 Chef 11.10 和較舊版本

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節提供 Chef AWS OpsWorks 11.10、11.4 和 0.9 for Linux Stacks 文件的簡短概觀。

Chef 11 Linux 堆疊入門

提供示範如何建立簡易但具有功能之 PHP 應用程式伺服器堆疊的演練。

建立您的第一個 Node.js 堆疊

說明如何建立支援 Node.js 應用程式伺服器的 Linux 堆疊，以及如何部署簡易應用程式。

[自訂 AWS OpsWorks Stacks]

說明如何自訂 AWS OpsWorks Stacks 以符合您的特定需求。

[技術指南 101]

說明如何實作 Stacks AWS OpsWorks 執行個體的配方。

[平衡 Layer 的負載]

說明如何使用可用的 AWS OpsWorks Stacks 負載平衡選項。

[在 VPC 中執行堆疊]

描述如何在 virtual private cloud 中建立和執行堆疊。

[從 Chef Server 遷移]

提供從 Chef Server 遷移至 Stacks AWS OpsWorks 的指導方針。

[AWS OpsWorks Stacks Layer 參考]

描述可用的 AWS OpsWorks Stacks 內建層。

[技術指南元件]

說明下列三個標準技術指南元件：屬性、範本和配方。

[堆疊組態及部署屬性：Linux]

說明適用於 Linux 的堆疊組態和部署屬性。

[內建技術指南屬性]

說明如何使用內建配方屬性控制已安裝軟體的組態。

[對適用於 Linux 的 Chef 11.10 和舊版故障診斷]

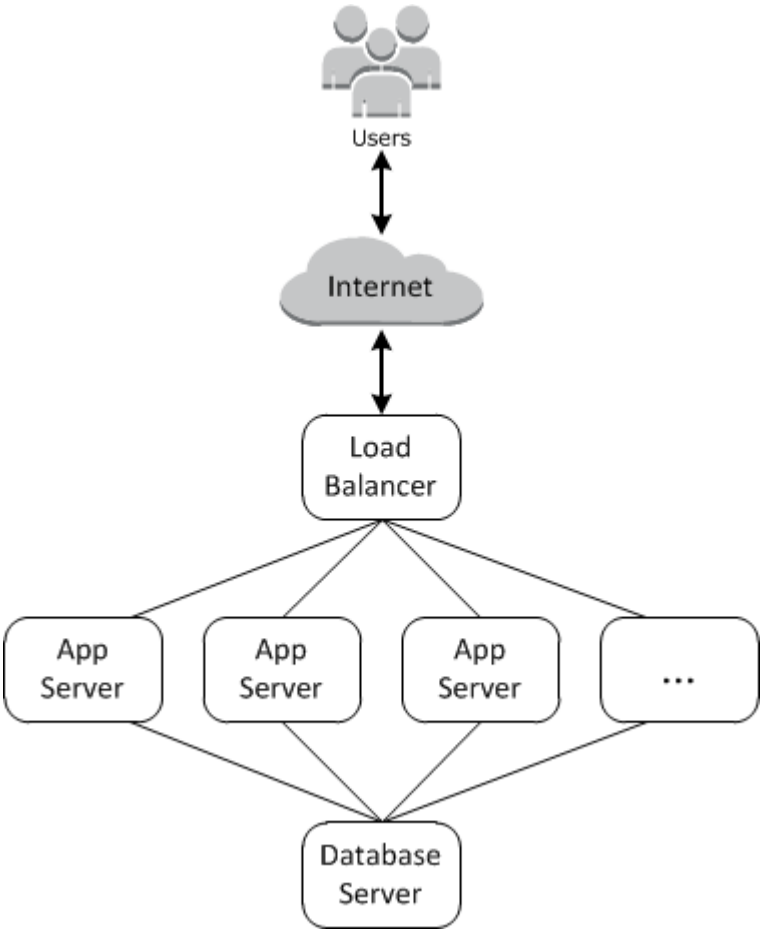
描述疑難排解 Stacks AWS OpsWorks 中各種問題的方法。

Chef 11 Linux 堆疊入門

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 本節說明如何開始搭配 Chef 11 使用 Linux 堆疊。如需 Chef 12 Linux 堆疊入門的資訊，請參閱入門：Linux。如需 Chef 12 Windows 堆疊入門的資訊，請參閱入門：Windows。

雲端型應用程式通常需要一組必須共同建立和管理的相關資源，例如應用程式伺服器、資料庫伺服器等。此執行個體的集合稱為「堆疊」。簡易的應用程式堆疊看起來可能如下。



基本架構由下列項目組成：

- 將來自使用者的傳入流量平均分散至應用程式伺服器的負載平衡器。
- 一組數量足以處理流量的應用程式伺服器執行個體。
- 提供應用程式伺服器後端資料存放區的資料庫伺服器。

此外，您通常需要一種將應用程式分散至應用程式伺服器、監控堆疊等的方式。

AWS OpsWorks Stacks 提供簡單且直接的方式來建立和管理堆疊及其相關聯的應用程式和資源。本章介紹 AWS OpsWorks Stacks 的基本概念，以及一些更複雜的功能，引導您完成圖表中建立應用程式伺服器堆疊的程序。它使用 Stacks AWS OpsWorks 易於遵循的增量開發模型：設定基本堆疊，並在正常運作後新增元件，直到您達到功能完整的實作為止。

- 步驟 1：完成事前準備示範如何進行設定以開始演練。
- 步驟 2：建立簡易應用程式伺服器堆疊 - Chef 11 示範如何建立只由單一應用程式伺服器組成的極簡堆疊。
- 步驟 3：新增後端資料存放區示範如何建立資料庫伺服器，並將其連線至應用程式伺服器。
- 步驟 4：橫向擴展 MyStack示範如何透過新增更多應用程式伺服器來擴展堆疊以處理增加的負載，以及分配傳入流量的負載平衡器。

主題

- 步驟 1：完成事前準備
- 步驟 2：建立簡易應用程式伺服器堆疊 - Chef 11
- 步驟 3：新增後端資料存放區
- 步驟 4：橫向擴展 MyStack
- 步驟 5：刪除 MyStack

步驟 1：完成事前準備

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

完成下列步驟，才能開始演練。這些設定步驟包括註冊 AWS 帳戶、建立管理使用者，以及將存取許可指派給 AWS OpsWorks Stacks。

若您已完成任何 Stacks AWS OpsWorks 入門演練，您便完成本演練的事前準備，可直接跳到步驟 2：建立簡易應用程式伺服器堆疊 - Chef 11。

主題

- 註冊 AWS 帳戶
- 建立具有管理存取權的使用者
- 將服務存取許可指派給您的使用者

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。

將服務存取許可指派給您的使用者

將 AWS OpsWorks `AWSOpsWorks_FullAccess`和 `AmazonS3FullAccess`許可新增至您的角色或使用者，以啟用 Stacks 服務的存取權 （以及 AWS OpsWorks Stacks 依賴的相關服務）。

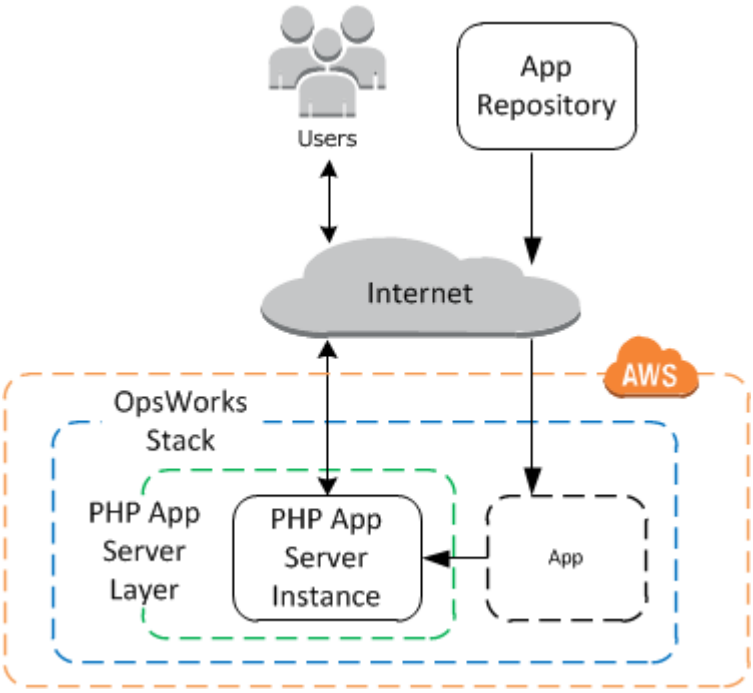
如需新增許可的詳細資訊，請參閱新增 IAM 身分許可 （主控台）。

您現已完成所有設定步驟，可開始本演練。

步驟 2：建立簡易應用程式伺服器堆疊 - Chef 11

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

基本應用程式伺服器堆疊包含一個具有公有 IP 地址的應用程式伺服器執行個體，以接收使用者請求。應用程式程式碼和任何相關檔案均存放於單獨的儲存庫中，並從該位置部署到伺服器。下圖說明此類堆疊。



該堆疊具有下列元件：

- layer 代表執行個體的群組，並指定設定它們的方式。

  此範例中的 layer 代表一組 PHP App Server 執行個體。

- 執行個體，代表 Amazon EC2 執行個體。

  在此範例中，執行個體已設定為執行 PHP 應用程式伺服器。Layers 可以有任意數量的執行個體。AWS OpsWorks Stacks 也支援數個其他應用程式伺服器。如需詳細資訊，請參閱應用程式伺服器 Layer。

- 「應用程式」包含在應用程式伺服器上安裝應用程式的必要資訊。

  程式碼存放在遠端儲存庫中，例如 Git 儲存庫或 Amazon S3 儲存貯體。

下列各節說明如何使用 AWS OpsWorks Stacks 主控台來建立堆疊並部署應用程式。您也可以使用 AWS CloudFormation 範本來佈建堆疊。如需本主題中說明之佈建堆疊的範例範本，請參閱 AWS OpsWorks 程式碼片段。

主題

- 步驟 2.1：建立堆疊 - Chef 11
- 步驟 2.2：新增 PHP 應用程式伺服器層 - Chef 11
- 步驟 2.3：將執行個體新增至 PHP App Server Layer - Chef 11
- 步驟 2.4：建立和部署應用程式 - Chef 11

步驟 2.1：建立堆疊 - Chef 11

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以透過建立堆疊來啟動 AWS OpsWorks Stacks 專案，堆疊可做為執行個體和其他資源的容器。堆疊組態會指定某些由堆疊的所有執行個體共享之基本設定，如 AWS 區域和預設作業系統。

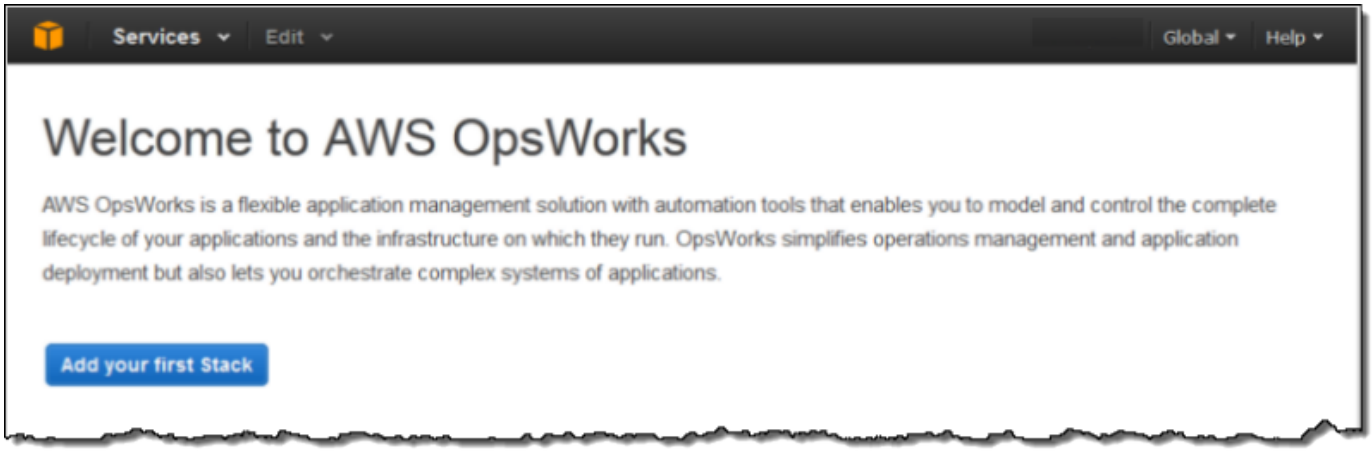
> **Note**
>
> 此頁面可協助您建立 Chef 11 堆疊。如需如何建立 Chef 12 堆疊的資訊，請參閱建立堆疊。

此頁面可協助您在 Chef 11 中建立堆疊。

建立新堆疊

1. 新增堆疊

   登入 AWS OpsWorks Stacks 主控台。若帳戶沒有堆疊，您會看到 Welcome to AWS OpsWorks (歡迎使用 AWS OpsWorks) 頁面。按一下 Add your first stack (新增您的第一個堆疊)。否則，您會看到 AWS OpsWorks Stacks 儀表板，其中列出您帳戶的堆疊；按一下新增堆疊。



2. 設定堆疊

   在 Add Stack (新增堆疊) 頁面上，選擇 Chef 11 stack (Chef 11 堆疊) 並指定下列設定：

   Stack name (堆疊名稱)

   輸入堆疊的名稱，其中可包含英數字元 (a–z、A–Z 和 0–9) 和連字號 (-)。本演練的範例堆疊名為 **MyStack**。

   區域

   選取美國西部 （奧勒岡） 做為堆疊的區域。

   接受其他設定的預設值，然後按一下 Add Stack (新增堆疊)。如需各種堆疊設定的詳細資訊，請參閱 建立新的堆疊。

步驟 2.2：新增 PHP 應用程式伺服器層 - Chef 11

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

儘管堆疊基本上是執行個體的容器，但您無法直接將執行個體新增到堆疊。您會新增 layer，其代表相關執行個體的群組，然後將執行個體新增至 layer。

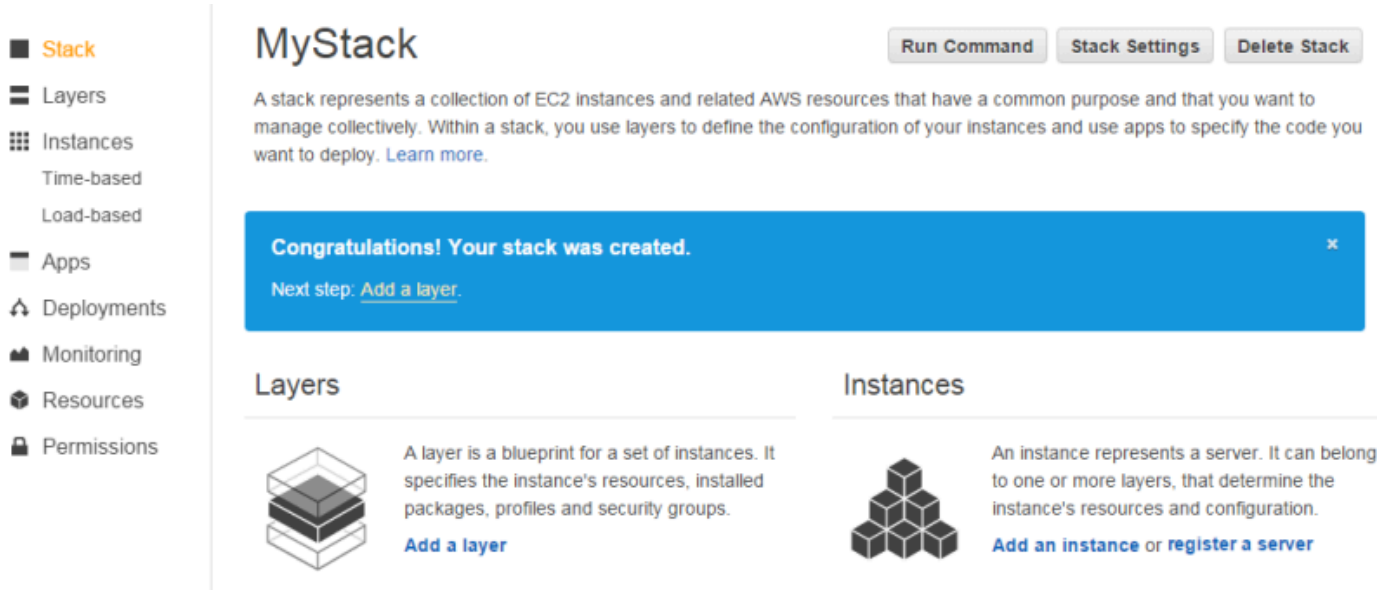
layer 基本上是 Stacks AWS OpsWorks 用來建立具有相同組態之 Amazon EC2 執行個體集的藍圖。您會為每個相關執行個體群組的堆疊新增一個 layer。 AWS OpsWorks Stacks 包含一組內建 layer，代表執行標準軟體套件的執行個體群組，例如 MySQL 資料庫伺服器或 PHP 應用程式伺服器。此外，您可以建立部分或完整自訂的 layer，以符合您的特定需求。如需詳細資訊，請參閱自訂 AWS OpsWorks Stacks。

MyStack 有一個 layer，即內建的 PHP App Server layer，代表一組充當 PHP 應用程式伺服器的執行個體。如需詳細資訊，包含內建 layer 的描述，請參閱層。

將 PHP App Server layer 新增至 MyStack

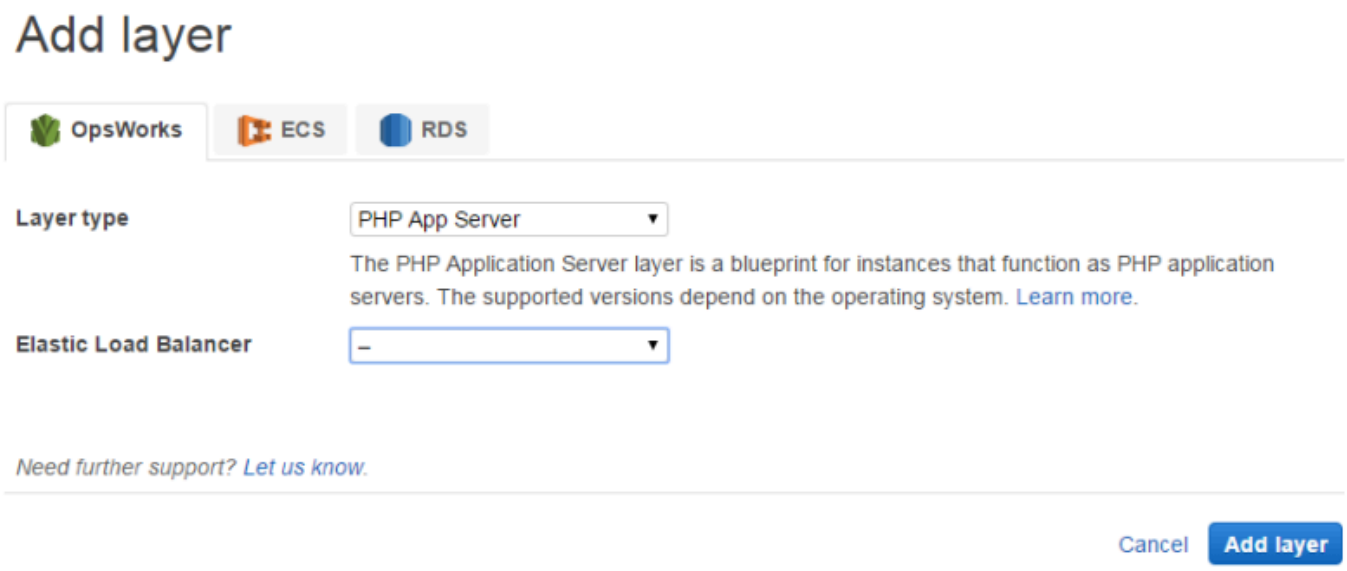
1. 開啟新增 Layer 頁面

   建立堆疊完成後， AWS OpsWorks Stacks 會顯示堆疊頁面。按一下 Add a layer (新增 layer) 以新增您的第一個 layer。



2. 指定 Layer 類型及設定 Layer

   在層類型方塊中，選取 PHP App Server，接受預設的 Elastic Load Balancer 設定，然後按一下新增層。在您建立 layer 後，您可以藉由編輯 layer 來指定其他屬性，例如：EBS 磁碟區組態。



步驟 2.3：將執行個體新增至 PHP App Server Layer - Chef 11

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Stacks AWS OpsWorks 執行個體代表特定的 Amazon EC2 執行個體：

- 執行個體的組態會指定一些基本概念，例如 Amazon EC2operating系統和大小；它會執行，但不會執行太多操作。
- 執行個體的 layer 可透過判斷要安裝何種套件、執行個體是否具有彈性 IP 地址等，為執行個體新增功能。

AWS OpsWorks Stacks 會在與服務互動的每個執行個體上安裝 代理程式。若要將 layer 的功能新增至執行個體， AWS OpsWorks Stacks 會指示代理程式執行稱為 Chef 配方的小型應用程式，這些應用程式可以安裝應用程式和套件、建立組態檔案等。 AWS OpsWorks Stacks 會在執行個體生命週期中的關鍵點執行配方。例如，OpsWorks 會在執行個體完成開機後執行安裝配方，處理像是安裝軟體等任務，然後在您部署應用程式時執行部署配方安裝程式碼和相關檔案。

> **Note**
>
> 如果您對配方的運作方式感到好奇，所有 AWS OpsWorks Stacks 內建配方都在公有 GitHub 儲存庫中：OpsWorks 技術指南。您也可以建立您自己的自訂配方，讓 AWS OpsWorks Stacks 執行他們，如稍後所說明。

若要將 PHP 應用程式伺服器新增至 MyStack，請將執行個體新增至您在上一個步驟中建立的 PHP App Server layer。

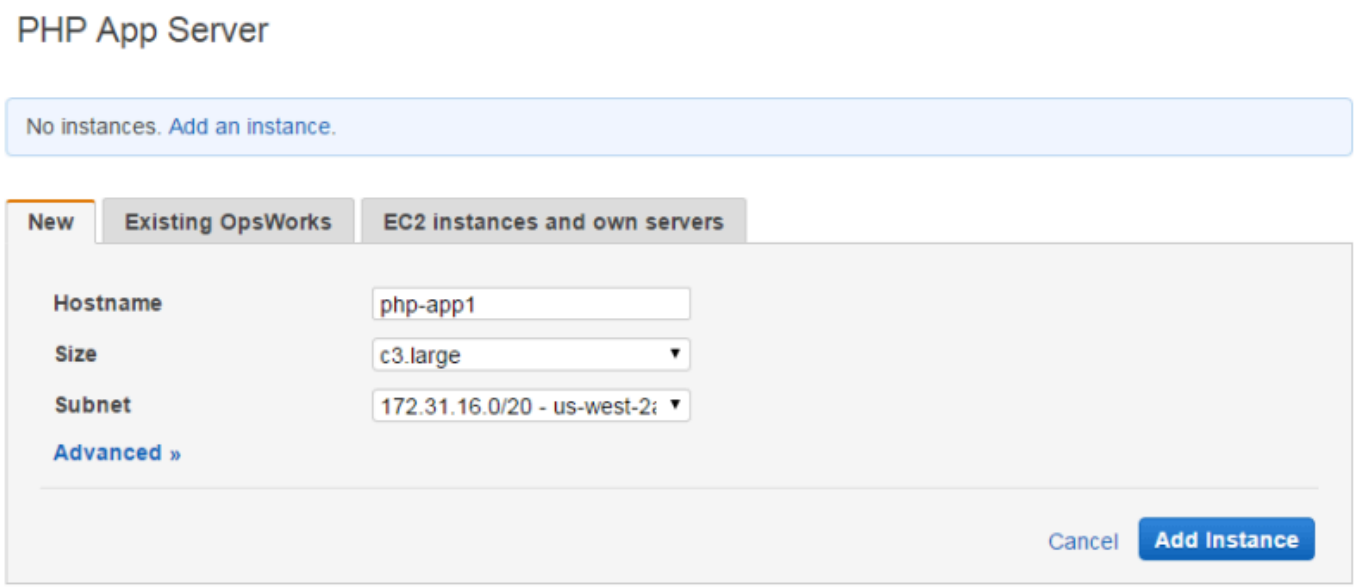
將執行個體新增至 PHP App Server layer

1. 開啟新增執行個體

   新增 layer 完成後， AWS OpsWorks Stacks 會顯示 Layers 頁面。在導覽窗格中按一下執行個體，然後在 PHP App Server 下按一下新增執行個體。

2. 設定執行個體

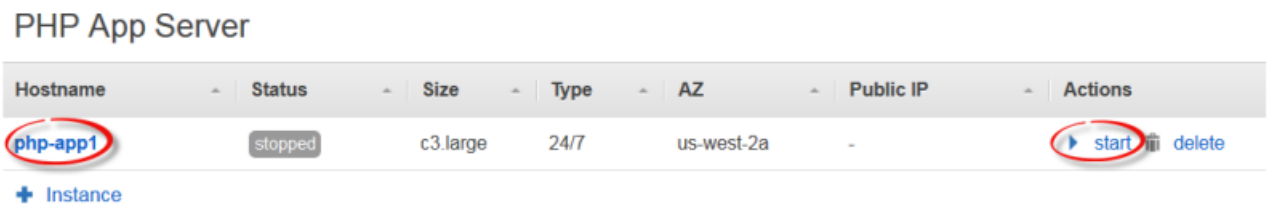
每個執行個體都有 Stacks AWS OpsWorks 為您產生的預設主機名稱。在此範例中， AWS OpsWorks Stacks 只會將數字新增至 layer 的簡短名稱。您可以個別設定每個執行個體，包含覆寫您在建立堆疊時指定的部分預設設定，例如可用區域或作業系統。針對本演練，請接受預設設定，然後按一下 Add Instance (新增執行個體) 將執行個體新增至 layer。如需詳細資訊，請參閱執行個體。



3. 啟動執行個體

到目前為止，您完成了執行個體組態的指定。您必須啟動執行個體，才能建立執行中的 Amazon EC2 執行個體。然後， AWS OpsWorks Stacks 會使用組態設定在指定的可用區域中啟動 Amazon EC2 執行個體。您啟動執行個體之方式的詳細資訊取決於執行個體的「擴展類型」。在先前的步驟中，您使用預設擴展類型「全年無休」建立執行個體，該類型必須手動啟動，並且會持續執行直到手動停止。您也可以建立時間型和負載型擴展類型， AWS OpsWorks 堆疊會根據排程或目前的負載自動啟動和停止。如需詳細資訊，請參閱使用時間型和負載型執行個體管理負載。

前往 PHP App Server 下的 php-app1，然後按一下列的動作欄中的開始以啟動執行個體。

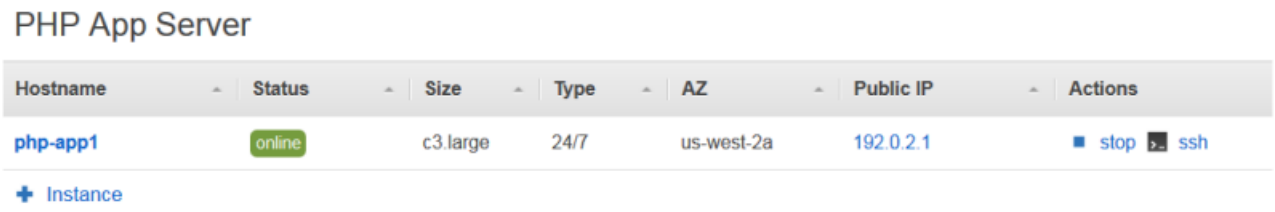


4. 在啟動時監控執行個體的狀態

   通常需要幾分鐘的時間來啟動 Amazon EC2 執行個體並安裝套件。隨著啟動的進行，執行個體的 Status (狀態) 欄位會顯示下列一系列的值：

   1. requested - AWS OpsWorks Stacks 已呼叫 Amazon EC2 服務來建立 Amazon EC2 執行個體。
   2. 待定 - AWS OpsWorks Stacks 正在等待 Amazon EC2 執行個體啟動。
   3. 開機 - Amazon EC2 執行個體正在開機。
   4. running_setup - Stacks AWS OpsWorks 代理程式正在執行 layer 的設定配方，處理設定和安裝套件等任務，以及部署配方，將任何應用程式部署到執行個體。
   5. online - 執行個體已準備就緒可供使用。

   在 php-app1 上線後，Instances (執行個體) 頁面看起來應該會像是這樣：

   頁面的開頭為所有您堆疊執行個體的快速摘要。現在，它會顯示一個線上的執行個體。在 php-app1 的 Actions (動作) 資料行中，請注意會停止執行個體的 stop (停止) 已取代 start (啟動) 和 delete (刪除)。

步驟 2.4：建立和部署應用程式 - Chef 11

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

若要讓 MyStack 更實用，您需要將應用程式部署到 PHP App Server 執行個體。您將應用程式的程式碼和任何相關檔案存放在儲存庫中 (例如 Git)。您需要幾個步驟，才能使那些檔案進入到您的應用程式伺服器內。

> **Note**
>
> 本節中的程序適用於 Chef 11 堆疊。如需如何在 Chef 12 堆疊中將應用程式新增至 layer 的資訊，請參閱新增應用程式。

1. 建立應用程式。

   應用程式包含 AWS OpsWorks Stacks 從儲存庫下載程式碼和相關檔案所需的資訊。您也可以指定額外的資訊，例如應用程式的網域。

2. 將應用程式部署到您的應用程式伺服器。

   當您部署應用程式時， AWS OpsWorks Stacks 會觸發部署生命週期事件。代理程式接著會執行執行個體的部署配方，將檔案下載到適當的目錄並執行相關任務，例如設定伺服器、重新啟動服務等。

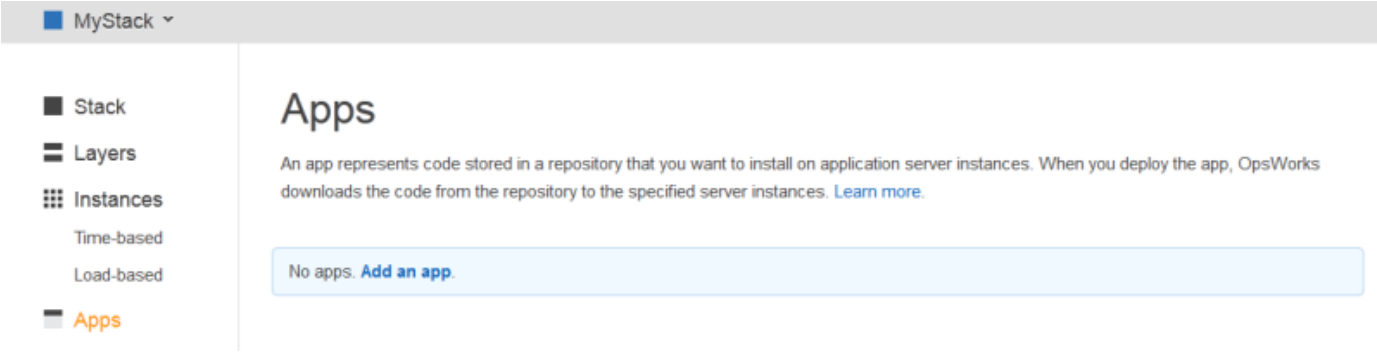
> **Note**
>
> 當您建立新的執行個體時， AWS OpsWorks Stacks 會自動將任何現有的應用程式部署到執行個體。但是，當您建立新的應用程式或更新現有的應用程式時，您必須手動部署應用程式或更新所有現有的執行個體。

此步驟顯示如何手動從公有 Git 儲存庫將範例應用程式部署到應用程式伺服器。若您想要檢查應用程式，請前往 https://github.com/amazonwebservices/opsworks-demo-php-simple-app。此範例中使用的應用程式位於版本 1 分支中。 AWS OpsWorks Stacks 也支援數種其他儲存庫類型。如需詳細資訊，請參閱應用程式來源。

建立和部署應用程式

1. 開啟應用程式頁面

   在導覽窗格中，按一下 Apps (應用程式)，然後在 Apps (應用程式) 頁面上，按一下 Add an app (新增應用程式)。



2. 設定應用程式。

   在 App (應用程式) 頁面上，指定下列值：

   名稱

   應用程式的名稱， Stacks AWS OpsWorks 用於顯示目的。範例應用程式名為 **SimplePHPApp**。 AWS OpsWorks Stacks 也會為此範例產生簡短名稱 -simplephpapp，供內部和部署配方使用，如下所述。

   Type

   應用程式的類型，判斷部署應用程式的位置。此範例使用 PHP，會將應用程式部署到 PHP App Server 執行個體。

   Data source type (資料來源類型)

   關聯的資料庫伺服器。現階段請先選取 None (無)。我們將於步驟 3：新增後端資料存放區介紹資料庫伺服器。

   儲存庫類型

   應用程式的儲存庫類型。範例應用程式存放於 Git 儲存庫。

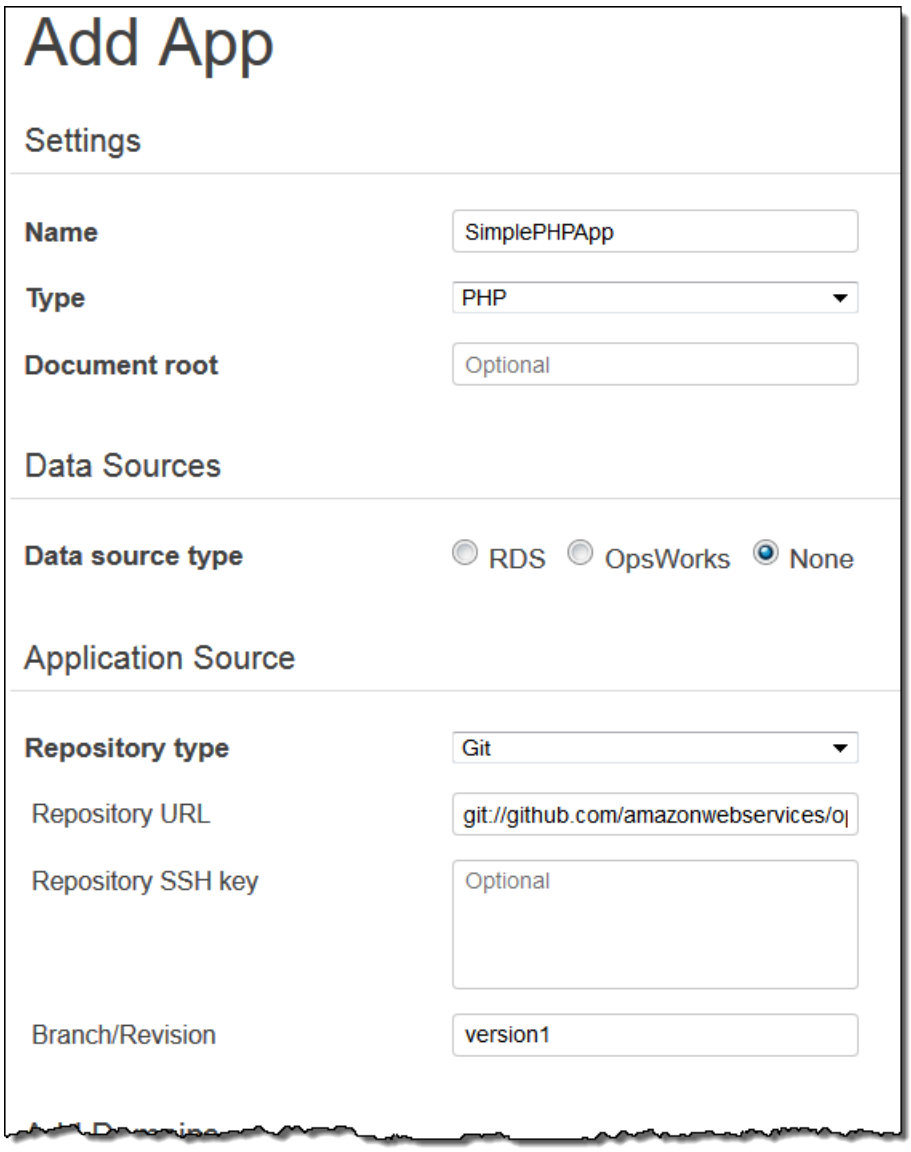
   Repository URL (儲存庫 URL)

   應用程式的儲存庫 URL。範例 URL 為 **git://github.com/awslabs/opsworks-demo-php-simple-app.git**

   Branch/Revision (分支/修訂)

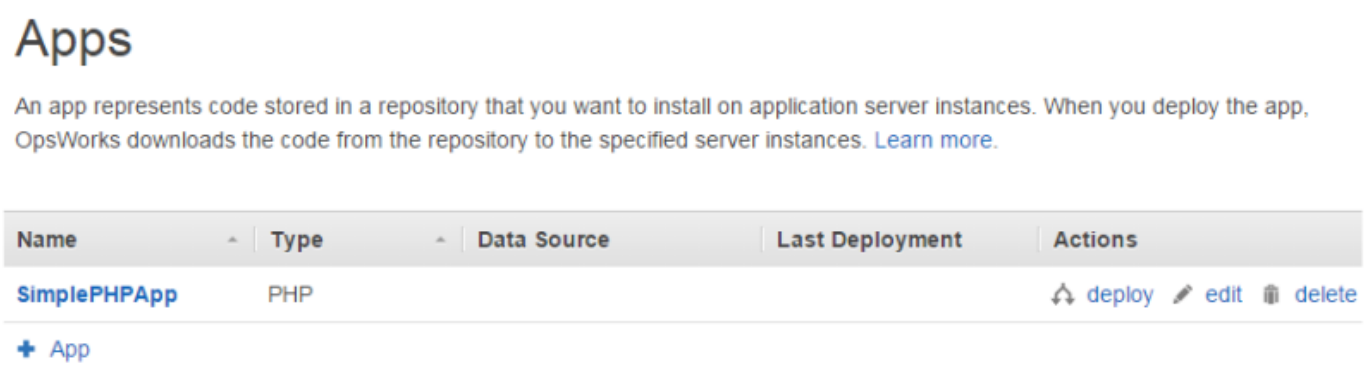
   應用程式的分支或版本。本演練的此部分使用 **version1** 分支。

   保留剩餘設定的預設值，然後按一下 Add App (新增應用程式)。如需詳細資訊，請參閱新增應用程式。



3. 開啟部署頁面

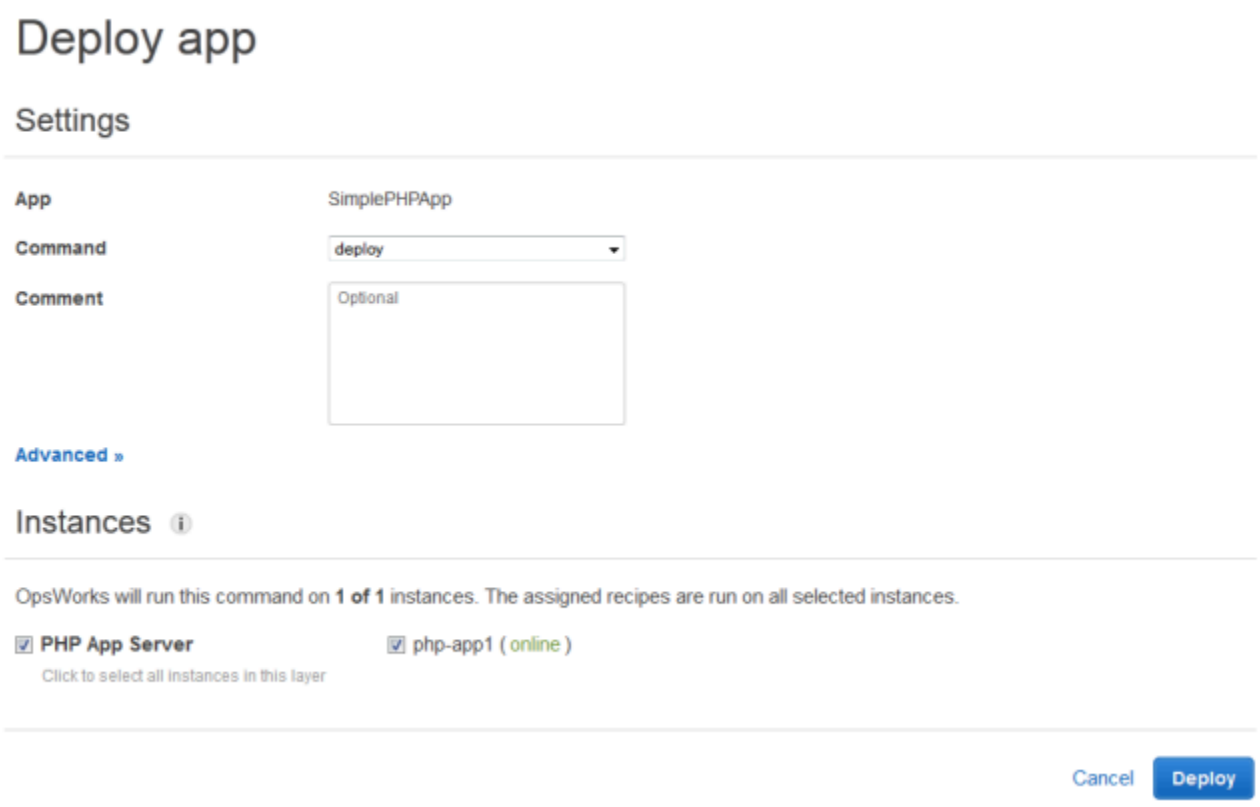
   若要在伺服器上安裝程式碼，您必須「部署」應用程式。若要執行此作業，請按一下 SimplePHPApp 中 Actions (動作) 資料行中的 deploy (部署)。



4. 部署應用程式

   當您部署應用程式時，代理程式會在 PHP App Server 執行個體上執行部署配方，以下載和設定應用程式。

   Command (命令) 應該已設為 deploy (部署)。保持其他設定的預設值，然後按一下 Deploy (部署) 部署應用程式。

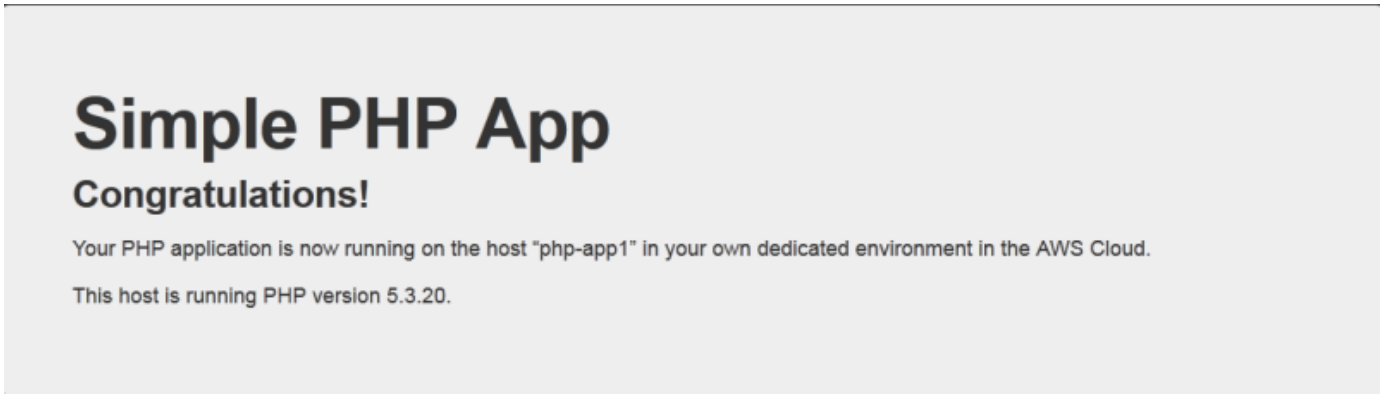


   部署完成後，Deployment (部署) 頁面便會顯示 Status (狀態) 為 Successful (成功)，並且 php-app1 一旁會有綠色的核取記號。

5. 執行 SimplePHPApp

SimplePHPApp 現在已安裝並準備好可供使用。若要執行它，請按一下導覽窗格中的 Instances (執行個體) 以前往 Instances (執行個體) 頁面。然後按一下 php-app1 執行個體的公有 IP 地址。

您應該會在您的瀏覽器中看到如下的頁面。

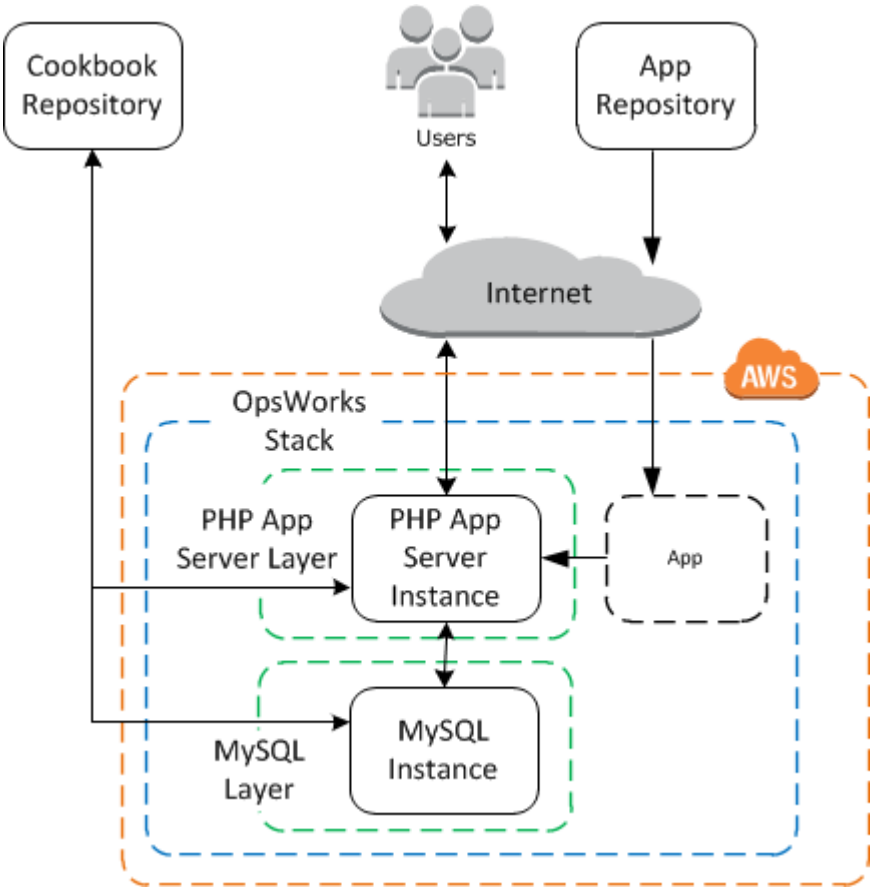


> **Note**
>
> 本演練假設您會繼續前進到下一節，並最終會於一個工作階段中完成整個演練。如果您願意，您可以隨時停止，稍後再登入 Stacks AWS OpsWorks 並開啟堆疊以繼續。但是，您必須為任何您使用的 AWS 資源支付費用 (例如線上執行個體)。為避免不必要的費用，您可以停止您的執行個體，如此便會停止對應的 EC2 執行個體。當您準備好繼續時，您可以再次啟動執行個體。

步驟 3：新增後端資料存放區

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

步驟 2.1：建立堆疊 - Chef 11 向您示範如何建立提供 PHP 應用程式的堆疊。不過，這是非常簡單的應用程式，作用只比顯示一些靜態文字多一點。生產應用程式通常使用後端資料存放區，產生類似下圖的堆疊組態。



本節示範如何擴展 MyStack 以包含後端 MySQL 資料庫伺服器。雖然您需要做的不只是新增 MySQL 伺服器到堆疊。您也必須將應用程式設定為與資料庫伺服器正確通訊。 AWS OpsWorks Stacks 不會為您執行此操作；您將需要實作一些自訂配方來處理該任務。

主題

- 步驟 3.1：新增後端資料庫
- 步驟 3.2：更新 SimplePHPApp
- 簡短分類：技術指南、配方和 AWS OpsWorks Stacks 屬性
- 步驟 3.3：將自訂技術指南新增至 MyStack
- 步驟 3.4：執行配方
- 步驟 3.5：部署 SimplePHPApp 2 版
- 步驟 3.6：執行 SimplePHPApp

步驟 3.1：新增後端資料庫

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

SimplePHPApp 的新版本將其資料存放在後端資料庫中。 AWS OpsWorks Stacks 支援兩種類型的資料庫伺服器：

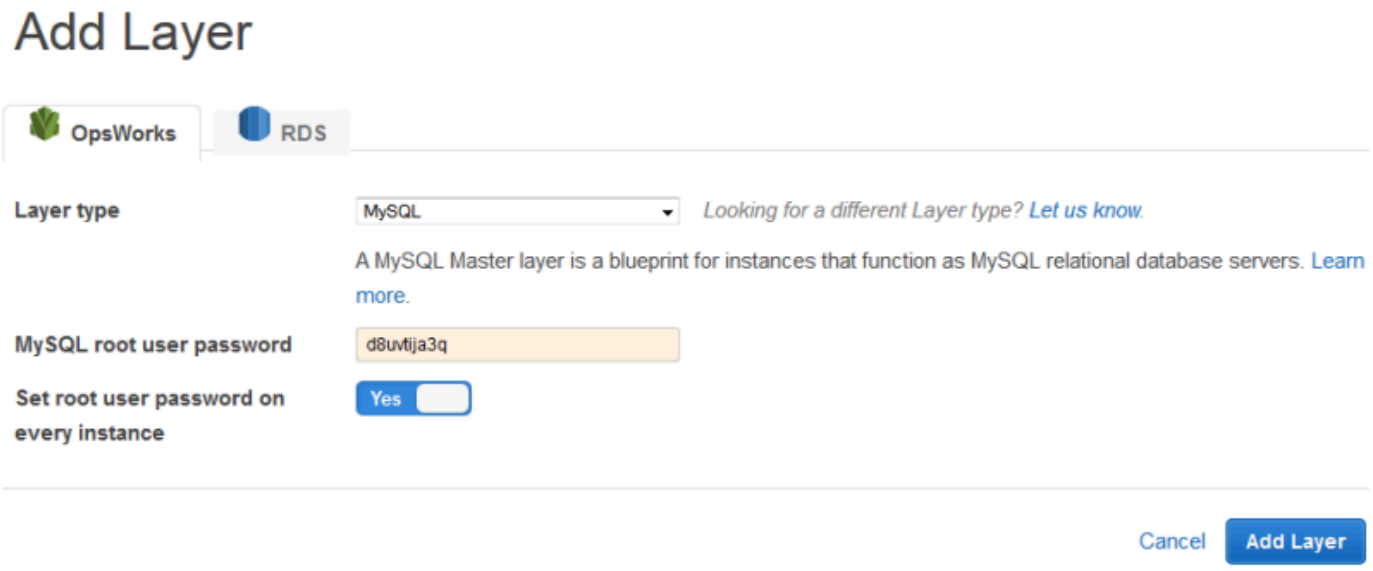
- MySQL AWS OpsWorks Stacks layer 是建立託管 MySQL 資料庫主節點的 Amazon EC2 執行個體的藍圖。
- Amazon RDS 服務層提供將 Amazon RDS 執行個體併入堆疊的方法。

您也可以使用其他資料庫，例如 Amazon DynamoDB，或建立自訂 layer 以支援資料庫，例如 MongoDB。如需詳細資訊，請參閱the section called “使用後端資料存放區”。

此範例使用 MySQL layer。

將 MySQL layer 新增至 MyStack

1. 在 Layers (Layer) 頁面上，按一下 + Layer。
2. 在 Add Layer (新增 Layer) 頁面上，針對 Layer type (Layer 類型)，選取 MySQL，接受預設設定，然後按一下 Add Layer (新增 Layer)。



將執行個體新增至 MySQL layer

1. 在 Layers (Layer) 頁面的 MySQL 資料列上，按一下 Add an instance (新增執行個體)。
2. 在 Instances (執行個體) 頁面上，在 MySQL 下，按一下 Add an instance (新增執行個體)。
3. 接受預設值，然後按一下 Add instance (新增執行個體)，但先不啟動它。

> **Note**
>
> AWS OpsWorks Stacks 會自動建立名為 的資料庫，在此範例中使用應用程式的簡短名稱 simplephpapp。如果您想要使用 Chef 配方與資料庫互動，您會需要此名稱。

步驟 3.2：更新 SimplePHPApp

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

若要開始，您必須使用後端資料存放區的新版 SimplePHPApp。使用 AWS OpsWorks Stacks，您可以輕鬆更新應用程式。如果您使用 Git 或 Subversion 儲存庫，每個應用程式版本都可以有單獨的儲存庫分支。範例應用程式會將使用後端資料庫之版本的應用程式存放在 Git 儲存庫的 version2 分支。您只需要更新應用程式的組態來指定新分支以及重新部署應用程式。

更新 SimplePHPApp

1. 開啟應用程式的 Edit (編輯) 頁面

   在導覽窗格中，按一下 Apps (應用程式)，然後按一下 SimplePHPApp 資料列中 Actions (動作) 欄的 edit (編輯)。
2. 更新應用程式組態

   變更下列設定。

Branch/Revision (分支/修訂)

此設定指出應用程式的儲存庫分支。第一版的 SimplePHPApp 未連線到資料庫。若要使用已啟用資料庫的應用程式版本，請將此值設定為 **`version2`**。

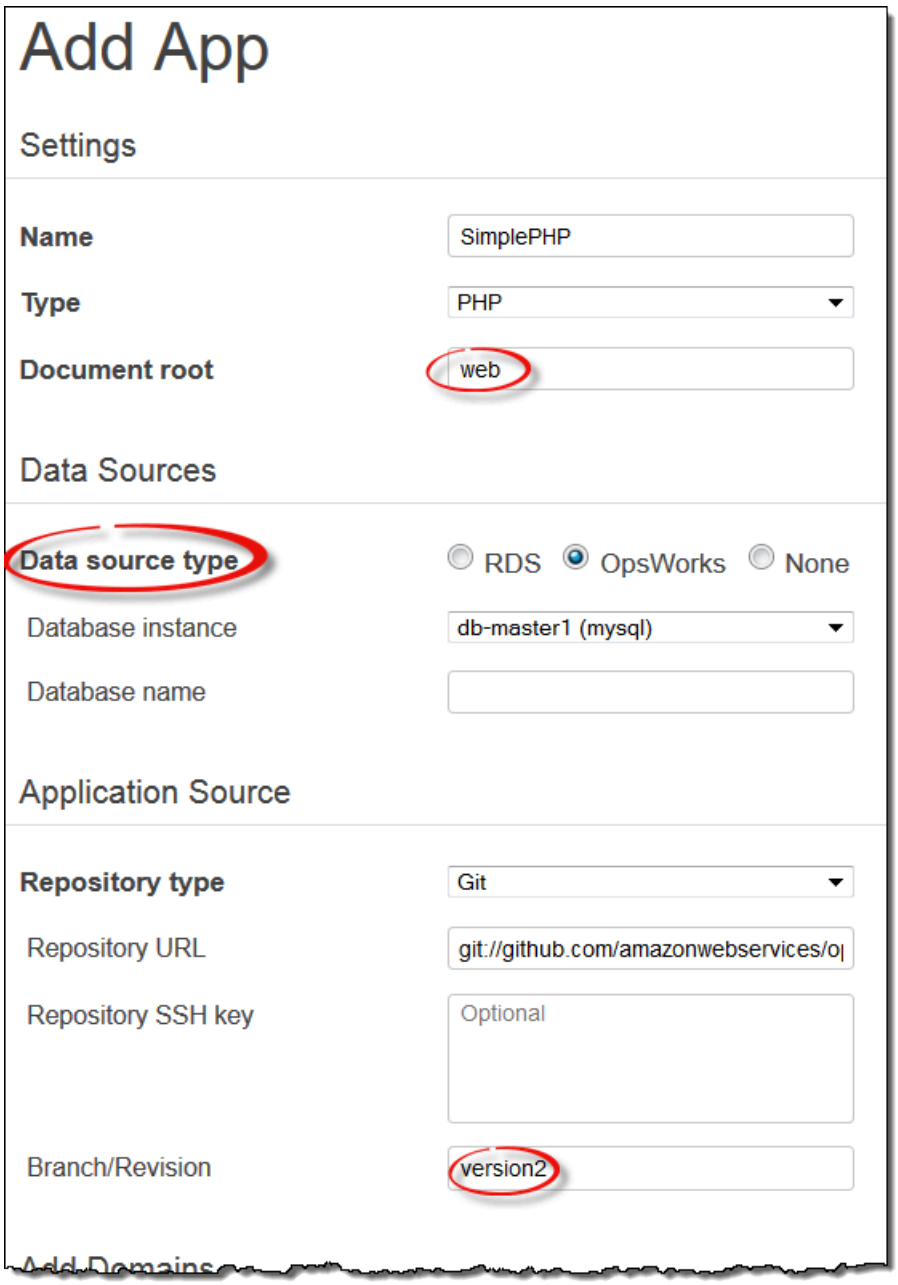
Document root (文件根)

此設定指定應用程式的根資料夾。第一版的 SimplePHPApp 使用預設設定，將 `index.php` 安裝在伺服器的標準根資料夾中 (PHP 應用程式為 `/srv/www`)。如果您在此處指定子資料夾 — 僅名稱，沒有前置 '/' —AWS OpsWorks Stacks 會將它附加到標準資料夾路徑。SimplePHPApp 版本 2 應該在 `/srv/www/web`，因此請將 Document root (文件根) 設為 **`web`**。

Data source type (資料來源類型)

此設定會建立資料庫伺服器與應用程式的關聯。此範例使用您在上一個步驟中建立的 MySQL 執行個體，因此請將資料來源類型設定為 OpsWorks 和資料庫執行個體，設定為您在上一個步驟 db-master1 (mysql) 中建立的執行個體。將資料庫名稱保留空白； AWS OpsWorks Stacks 會在名為 的伺服器上，以應用程式的簡短名稱 simplephpapp 建立資料庫。

然後按一下 Save (儲存)，儲存新的組態。



3. 啟動 MySQL 執行個體。

更新應用程式之後，當您啟動應用程式伺服器執行個體時， AWS OpsWorks Stacks 會自動將新的應用程式版本部署到任何新的應用程式伺服器執行個體。不過， AWS OpsWorks Stacks 不會自動將新的應用程式版本部署到現有的伺服器執行個體；您必須手動執行此操作，如中所述步驟 2.4：建立和部署應用程式 - Chef 11。您現在可以部署已更新的 SimplePHPApp，但在此範例中，最好再等等。

簡短分類：技術指南、配方和 AWS OpsWorks Stacks 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您現在有應用程式和資料庫伺服器，但它們還不能使用。您仍然需要設定資料庫並設定應用程式的連線設定。 AWS OpsWorks Stacks 不會自動處理這些任務，但支援 Chef 技術指南、配方和動態屬性。您可以實作一對配方，一個用於設定資料庫，另一個用於設定應用程式的連線設定，並讓 AWS OpsWorks Stacks 為您執行它們。

包含必要配方的 phpapp 技術指南已實作可供使用，如果想要，您可以直接跳到步驟 3.3：將自訂技術指南新增至 MyStack。如果想知道更多，本節會提供一些有關技術指南和配方的背景，說明配方的運作方式。若要查看技術指南本身，請參閱 phpapp 技術指南。

主題

- 配方和屬性
- 設定資料庫
- 將應用程式連線到資料庫

配方和屬性

Chef 配方基本上是專門在執行個體上執行任務 (例如安裝套件、建立組態檔案、執行 shell 命令等等) 的 Ruby 應用程式。相關配方的群組會組織成「技術指南」，其也包含支援的檔案，例如建立組態檔案的範本。

AWS OpsWorks Stacks 有一組支援內建 layer 的技術指南。您也可以使用自己的配方建立自訂技術指南，在您的執行個體上執行自訂任務。本主題提供配方的簡介，並示範如何使用它們設定資料庫以及設定應用程式的連線設定。如需技術指南和配方的詳細資訊，請參閱技術指南和配方或自訂 AWS OpsWorks Stacks。

配方通常取決於輸入資料的 Chef「屬性」：

- 這些屬性有部分由 Chef 定義，提供執行個體的基本資訊，例如作業系統。

- AWS OpsWorks Stacks 會定義一組屬性，其中包含堆疊的相關資訊，例如 layer 組態，以及已部署應用程式的相關資訊，例如應用程式儲存庫。

  您可以將自訂 JSON 指派給堆疊或部署，將自訂的屬性新增至此集合。
- 您的技術指南也可以定義技術指南專用的屬性。

  phpapp 技術指南屬性在 `attributes/default.rb` 中定義。

如需 AWS OpsWorks Stacks 屬性的完整清單，請參閱 堆疊組態及部署屬性：Linux和 內建技術指南屬性。如需詳細資訊，請參閱覆寫屬性。

屬性是以階層式結構組織，可以 JSON 物件表示。

您使用 Chef 節點語法將此資料併入您的應用程式，如下所示：

```
[:deploy][:simplephpapp][:database][:username]
```

`deploy` 節點有單一應用程式節點 `simplephpapp`，其包含應用程式資料庫、Git 儲存庫等等的資訊。此範例呈現資料庫使用者名稱的值，解析成 `root`。

設定資料庫

MySQL layer 的內建設定配方會自動為名為 的應用程式建立資料庫，並使用應用程式的短名，因此在此範例中，您已有一個名為 simplephpapp 的資料庫。不過，您需要建立表格供應用程式存放其資料，以完成設定。您可以手動建立資料表，但更好的方法是實作自訂配方來處理任務，並讓 AWS OpsWorks Stacks 為您執行。本節說明如何實作配方 `dbsetup.rb`。稍後將說明讓 AWS OpsWorks Stacks 執行配方的程序。

若要查看儲存庫中的配方，請前往 dbsetup.rb。以下範例顯示 `dbsetup.rb` 程式碼。

`execute` 是執行指定命令的「Chef 資源」。在此範例中，它是建立表格的 MySQL 命令。如果指定的表格已存在，`not_if` 指令會確保命令不會執行。如需 Chef 資源的詳細資訊，請參閱關於資源和提供者。

配方使用先前討論過的節點語法，將屬性值插入命令字串。例如，以下內容會插入資料庫的使用者名稱。

```
#{deploy[:database][:username]}
```

讓我們解釋這個有點隱晦的程式碼：

- 每次重複，deploy 都設為目前的應用程式節點，所以它會解析成 [:deploy][:*app_name*]。在此範例中，它解析成 [:deploy][:simplephpapp]。
- 使用先前顯示的部署屬性值，整個節點會解析成 root。
- 您用 #{ } 包裝節點，將它插入字串中。

其他大部分的節點也以類似方式解析。但 #{node[:phpapp][:dbtable]} 是例外，它是由自訂技術指南的屬性檔案所定義，解析成表格名稱 urler。因此，在 MySQL 執行個體上執行的實際命令為：

```
"/usr/bin/mysql
    -uroot
    -pvjud1hw5v8
    simplephpapp
    -e'CREATE TABLE urler(
       id INT UNSIGNED NOT NULL AUTO_INCREMENT,
       author VARCHAR(63) NOT NULL,
       message TEXT,
       PRIMARY KEY (id))'
"
```

此命令會使用來自部署屬性的登入資料和資料庫名稱，建立有 ID、作者和訊息欄位的表格，名為 urler。

將應用程式連線到資料庫

第二塊拼圖是應用程式，它需要連線資訊，例如存取表格的資料庫密碼。SimplePHPApp 其實只有一個工作檔案 app.php，而 index.php 所做的只是載入 app.php。

app.php 包含處理資料庫連線的 db-connect.php，但此檔案不在儲存庫中。您不能事先建立 db-connect.php，因為它會根據特定的執行個體定義資料庫。反之，appsetup.rb 配方使用來自部署屬性的連線資料產生 db-connect.php。

若要查看儲存庫中的配方，請前往 appsetup.rb。以下範例顯示 appsetup.rb 程式碼。

如同 dbsetup.rb， appsetup.rb 會反覆運算deploy節點中的應用程式，只需再次使用 simplephpapp。它會執行有 script 資源和 template 資源的程式碼區塊。

`script` 資源會安裝 Composer，這是 PHP 應用程式的相依性管理員。然後執行 Composer 的 `install` 命令，將範例應用程式的相依性安裝到應用程式的根目錄。

`template` 資源會產生 `db-connect.php`，並將它放在 `/srv/www/simplephpapp/current`。注意下列事項：

- 此配方使用條件式陳述式指定檔案擁有者，這取決於執行個體的作業系統。
- `only_if` 指令通知 Chef 只在指定目錄存在時產生範本。

`template` 資源在基本上與相關聯檔案有相同的內容和結構，但包含各種資料值的預留位置的範本上操作。`source` 參數指定範本 `db-connect.php.erb`，它在 phpapp 技術指南的 `templates/default` 目錄中，包含下列內容：

當 Chef 處理範本時，它會以範本資源中的對應變數值取代 `<%= =>` 預留位置，因此取自部署屬性。因此產生的檔案為：

步驟 3.3：將自訂技術指南新增至 MyStack

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您將自訂技術指南存放在儲存庫中，很類似應用程式。每個堆疊都有一個包含一組自訂技術指南的儲存庫。然後，指示 AWS OpsWorks Stacks 在堆疊的執行個體上安裝自訂技術指南。

1. 按一下導覽窗格中的 Stack (堆疊) 查看目前堆疊的頁面。
2. 按一下 Stack Settings (堆疊設定)，然後按一下 Edit (編輯)。
3. 如下修改堆疊組態：
   - 使用自訂 Chef 技術指南 – 是
   - 儲存庫類型 – Git
   - 儲存庫 URL – **`git://github.com/amazonwebservices/opsworks-example-cookbooks.git`**
4. 按一下 Save (儲存) 更新堆疊組態。

AWS OpsWorks Stacks 接著會在堆疊的所有執行個體上安裝技術指南儲存庫的內容。如果您建立新的執行個體， AWS OpsWorks Stacks 會自動安裝技術指南儲存庫。

> **Note**
>
> 如果您需要更新任何技術指南，或將新的技術指南新增至儲存庫，您可以這樣做，而無需接觸堆疊設定。 AWS OpsWorks Stacks 會自動在所有新的執行個體上安裝更新的技術指南。不過， AWS OpsWorks Stacks 不會自動在堆疊的線上執行個體上安裝更新的技術指南。您必須執行堆疊`Update Cookbooks`命令，明確指示 AWS OpsWorks Stacks 更新技術指南。如需詳細資訊，請參閱執行堆疊命令。

步驟 3.4：執行配方

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

有了自訂技術指南之後，您需要在適當的執行個體上執行配方。您可以手動執行它們。不過，配方一般需要在執行個體生命週期的可預測點上執行，例如在執行個體開機後，或部署應用程式時。本節說明更簡單的方法：讓 AWS OpsWorks Stacks 在適當的時間自動為您執行。

AWS OpsWorks Stacks 支援一組生命週期事件，可簡化執行中的配方。例如，執行個體開機完成後會發生 Setup (設定) 事件，當您部署應用程式時則會發生 Deploy (部署) 事件。每個 layer 都有一組與每個生命週期事件相關聯的內建配方。當執行個體上發生生命週期事件時，代理程式會為每個執行個體

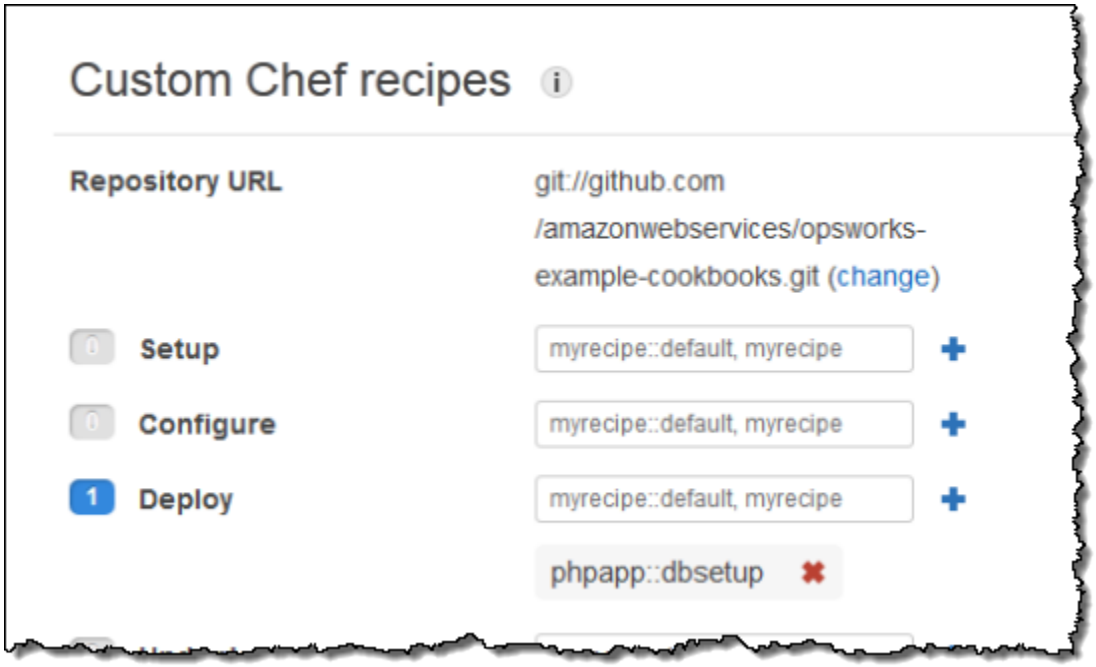
layer 執行相關聯的配方。若要讓 AWS OpsWorks Stacks 自動執行自訂配方，請將它新增至適當 layer 上的適當生命週期事件，代理程式會在內建配方完成後執行配方。

在此範例中，您需要`dbsetup.rb`在 MySQLinstance 和 PHP App Server 執行個體`appsetup.rb`上執行兩個配方。

> **Note**
>
> 您使用 *cookbook_name*::*recipe_name* 格式在主控台上指定配方，*recipe_name* 不包含 .rb 副檔名。例如，您參考 `dbsetup.rb` 為 **phpapp::dbsetup**。

將自訂配方指派給生命週期事件

1. 在 Layers 頁面上，針對 MySQL，按一下配方，然後按一下編輯。
2. 在自訂 Chef 配方區段中，輸入 部署**phpapp::dbsetup**。



3. 按一下 + 圖示，將配方指派給事件，然後按一下 Save (儲存) 儲存新的 layer 組態。
4. 返回 Layers (Layer) 頁面，重複此程序將 **phpapp::appsetup** 指派給 PHP App Server (PHP 應用程式伺服器) layer 的 Deploy (部署) 事件。

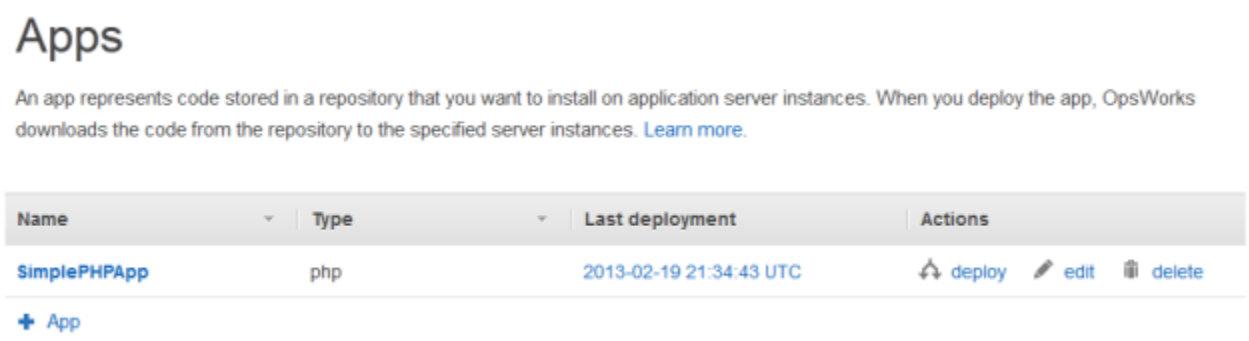
步驟 3.5：部署 SimplePHPApp 2 版

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

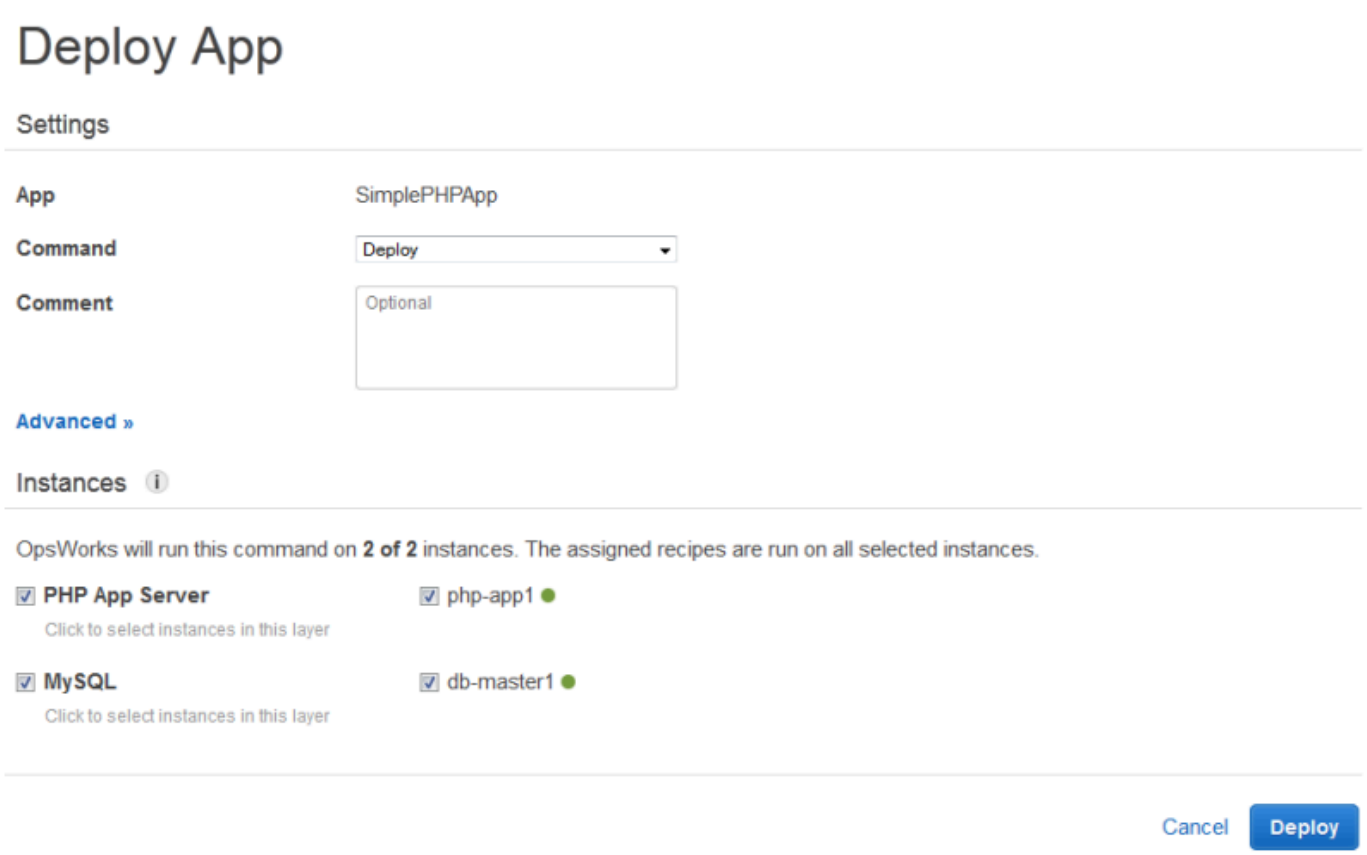
最終步驟是部署新版本的 SimplePHPApp。

部署 SimplePHPApp

1. 在 Apps (應用程式) 頁面上，按一下 SimplePHPApp 應用程式 Actions (動作) 中的 deploy (部署)。



2. 接受預設值，然後按一下 Deploy (部署)。



當您在部署應用程式頁面上按一下部署時，會觸發部署生命週期事件，通知客服人員執行其部署配方。根據預設，您會觸發所有堆疊的執行個體上之事件。內建的部署配方只會將應用程式部署到應用程式類型的適當執行個體，在此情況下為 PHP App Server 執行個體。不過，觸發其他執行個體上的 Deploy (部署) 事件，讓它們回應應用程式部署，通常很有用。在此情況下，您也想要觸發 MySQL 執行個體上的部署來設定資料庫。

注意下列事項：

- PHP App Server 執行個體上的代理程式會執行 layer 的內建配方，後面接著 `appsetup.rb`，這會設定應用程式的資料庫連線。
- MySQL 執行個體上的代理程式不會安裝任何項目，但會執行 `dbsetup.rb`來建立 urler 資料表。

當部署完成後，Deployment (部署) 頁面上的 Status (狀態) 會變更為 successful (成功)。

步驟 3.6：執行 SimplePHPApp

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

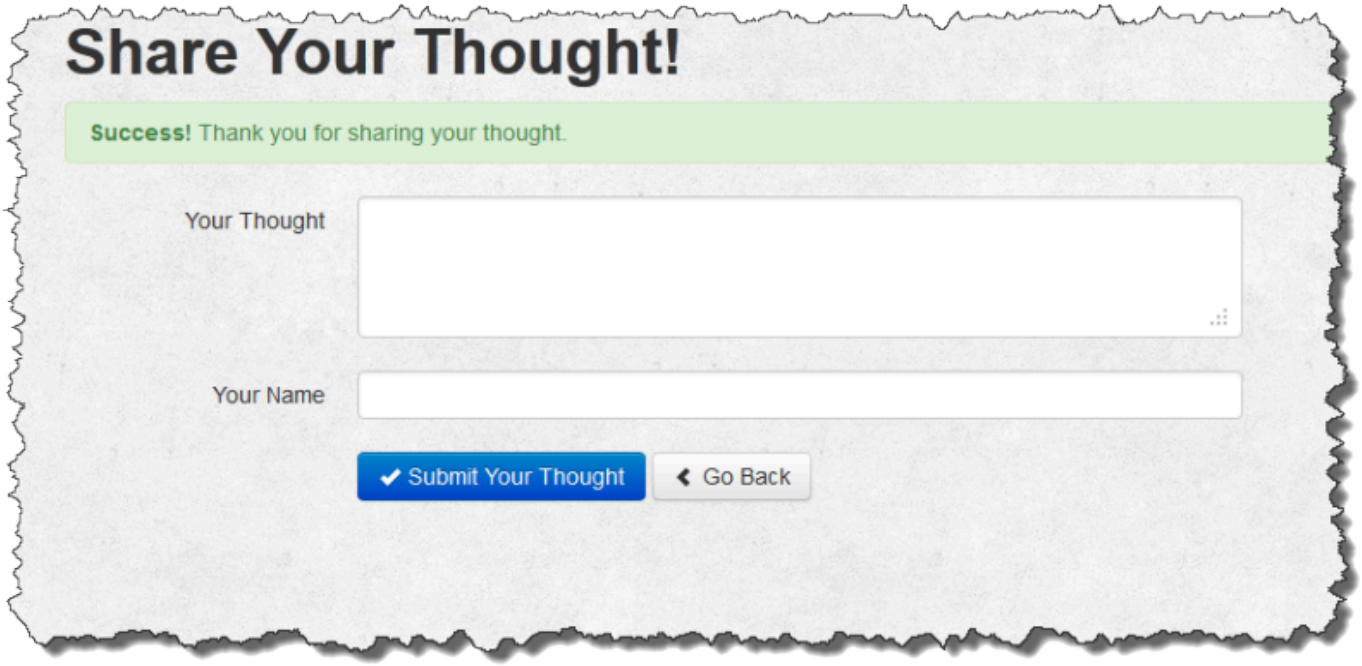
部署狀態變更為 successful (成功) 後，您就可以執行新的 SimplePHPApp 版本，如下所示。

執行 SimplePHPApp

1. 在 Instances (執行個體) 頁面上，按一下在 php-app1 資料列中的公有 IP 地址。

   您應該會在瀏覽器中看到如下頁面。

2. 按一下 Share Your Thought (分享您的想法)，輸入類似 **`Hello world!`** (針對 Your Thought (您的想法)) 和您的姓名 (針對 Your Name (您的姓名))。然後按一下 Submit Your Thought (提交您的想法) 將訊息新增到資料庫。

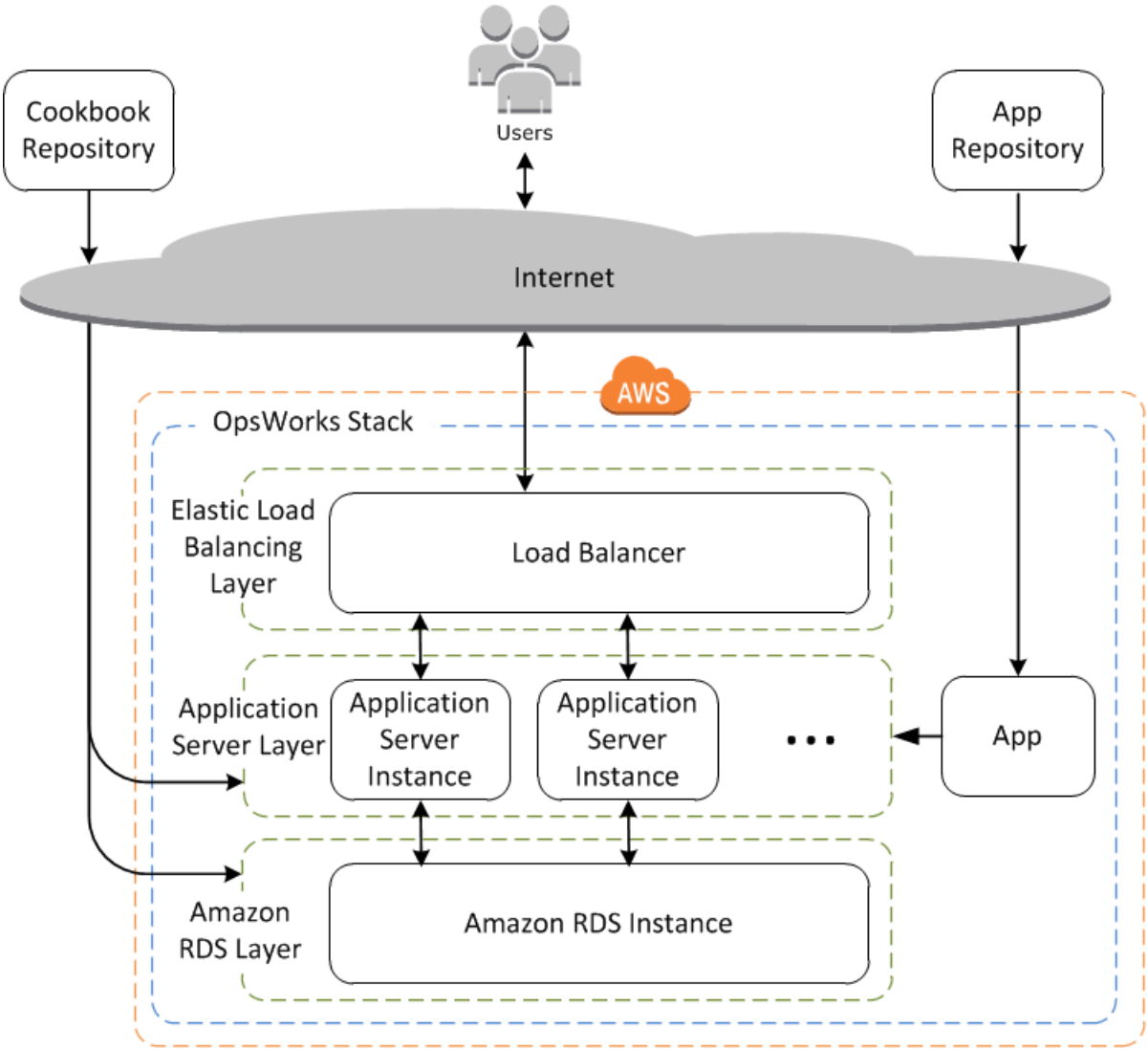


3. 按一下 Go Back (返回) 檢視資料庫中的所有訊息。

步驟 4：橫向擴展 MyStack

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

MyStack 目前只有一部應用程式伺服器。生產堆疊可能需要多部應用程式伺服器才能處理傳入流量，以及一個負載平衡器才能將傳入流量平均配送至應用程式伺服器。此架構類似下列內容。



AWS OpsWorks Stacks 可讓您輕鬆地向外擴展堆疊。本節說明如何將第二個全年無休 PHP App Server 執行個體新增至 MyStack，並將兩個執行個體放在 Elastic Load Balancing 負載平衡器後方，以擴展堆疊的基本概念。您可以輕鬆擴展程序來新增 24/7 執行個體的任意數量，也可以使用時間型或負載型執行個體，讓 AWS OpsWorks Stacks 自動擴展您的堆疊。如需詳細資訊，請參閱使用時間型和負載型執行個體管理負載。

步驟 4.1：新增負載平衡器

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Elastic Load Balancing 是一種 AWS 服務，可在多個 Amazon EC2 執行個體之間自動分配傳入的應用程式流量。除了分佈流量之外，Elastic Load Balancing 還會執行下列動作：

- 偵測運作狀態不佳的 Amazon EC2 執行個體。

  它會將流量重新路由至狀況良好的執行個體，直到狀況不良的執行個體恢復為止。
- 自動擴展處理容量的請求，以回應傳入的流量

> **Note**
>
> 負載平衡器有兩種用途。其中一個顯而易見的用途，是使應用程式伺服器上的負載達到均衡。此外，許多網站都偏好隔離它們的應用程式伺服器和資料庫，讓使用者無法直接存取。使用 AWS OpsWorks Stacks，您可以透過在具有公有和私有子網路的虛擬私有雲端 (VPC) 中執行堆疊來執行此操作，如下所示。
>
> - 將應用程式伺服器和資料庫放入私有子網路中，而 VPC 中的其他執行個體可以在其中存取它們，但使用者無法存取。
> - 將使用者流量導向公有子網路中的負載平衡器，接著將流量轉遞給私有子網路中的應用程式伺服器，並將回應傳回給使用者。
>
> 如需詳細資訊，請參閱在 VPC 中執行堆疊。如需延伸本演練中範例以在 VPC 中執行的 AWS CloudFormation 範本，請下載 `OpsWorksVPCtemplates.zip` 檔案。

雖然 Elastic Load Balancing 通常稱為 layer，但其運作方式與其他內建 layer 略有不同。您可以使用 Amazon EC2 主控台建立 Elastic Load Balancing 負載平衡器，然後將其連接到其中一個現有 layer，通常是應用程式伺服器 layer。 AWS OpsWorks Stacks 接著會向 服務註冊 layer 的現有執行個體，並自動新增任何新執行個體。下列程序說明如何將負載平衡器新增至 MyStack 的 PHP App Server layer。

> **Note**
>
> AWS OpsWorks Stacks 不支援 Application Load Balancer。您只能搭配 Stacks AWS OpsWorks 使用 Classic Load Balancer。

將負載平衡器連接至 PHP App Server layer

1. 使用 Amazon EC2 主控台為 MyStack 建立新的負載平衡器。詳細資訊取決於您的帳戶是否支援 EC2 Classic。如需詳細資訊，請參閱 Elastic Load Balancing 入門。當您執行 Create Load Balancer (建立負載平衡器) 精靈時，請設定負載平衡器，如下所示：

   Define Load Balancer (定義負載平衡器)

   指派可輕鬆辨識的名稱給負載平衡器，例如 PHP-LB，讓您更輕鬆地在 Stacks AWS OpsWorks 主控台中找到 。然後選擇 Continue (繼續) 接受其餘設定的預設值。

   如果您從 Create LB Inside (於內部建立 LB) 選單中選擇具有一或多個子網路的 VPC，則必須針對您想要負載平衡器路由流量的每個可用區域選取子網路。

   Assign Security Groups (指派安全群組)

   如果您的帳戶支援預設 VPC，精靈會顯示此頁面以決定負載平衡器的安全群組。但不會為 EC2 Classic 顯示此頁面。

   在本演練中，選擇 default VPC security group (預設 VPC 安全群組)。

   Configure Security Settings (設定安全設定)

   如果您在定義負載平衡器頁面上選擇 HTTPS 做為負載平衡器通訊協定，請在此頁面設定憑證、密碼和 SSL 通訊協定設定。 Load Balancer Load Balancer 在本演練中，接受預設值，然後選擇 Configure Health Check (設定運作狀態檢查)。

   Configure Health Check (設定運作狀態檢查)

   將 ping 路徑設定為 **/**，並接受其餘設定的預設值。

   Add EC2 Instances (新增 EC2 執行個體)

   選擇繼續； AWS OpsWorks Stacks 會自動向負載平衡器註冊執行個體。

   新增標籤

   新增標籤，協助您尋找。每個標籤都是鍵/值對；例如，您可以指定 **Description** 做為鍵並指定 **Test LB** 做為值，以用於演練。

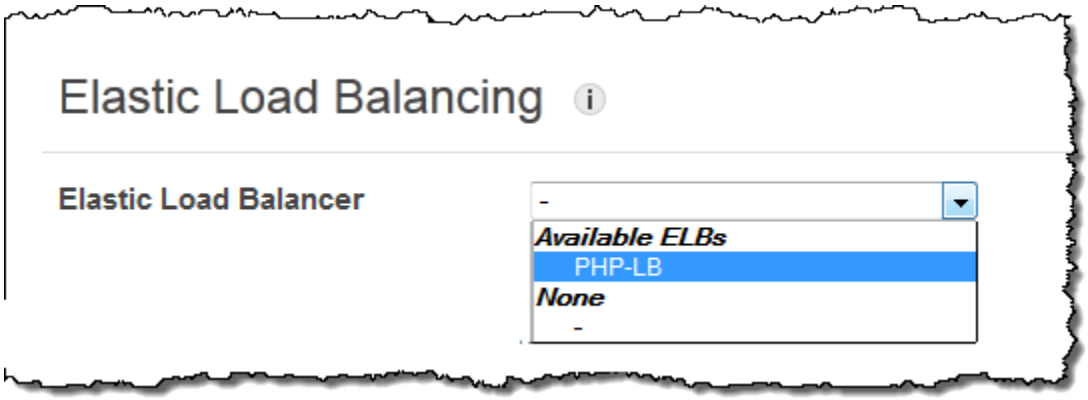
   檢閱

   檢閱您的選擇，並選擇 Create (建立)，然後選擇 Close (關閉)，以啟動負載平衡器。

2. 如果您的帳戶支援預設 VPC，則在您啟動負載平衡器之後，必須確保其安全群組具有適當的傳入規則。預設規則不接受任何傳入流量。

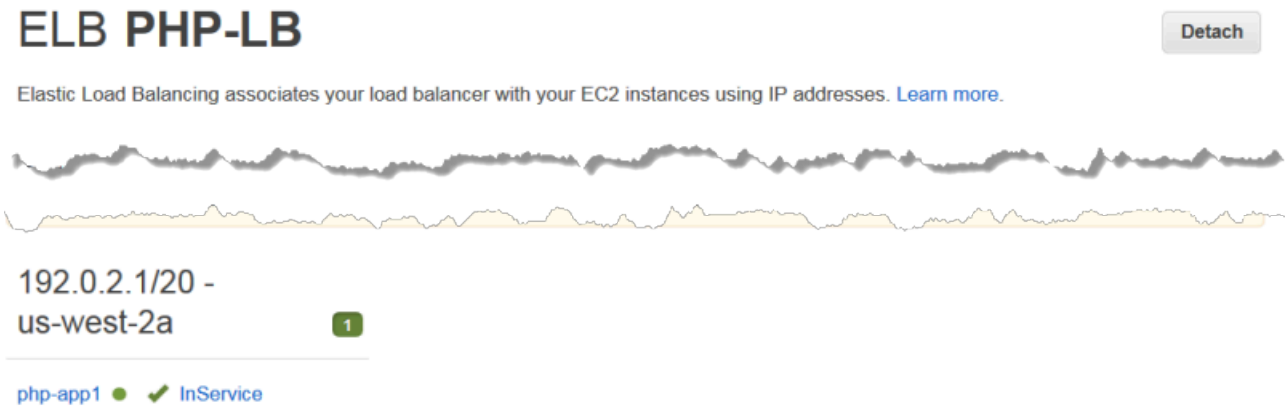
1. 在 Amazon EC2 導覽窗格中選擇安全群組。
2. 選取 default VPC security group (預設 VPC 安全群組)
3. 在 Inbound (傳入) 標籤上，選擇 Edit (編輯)。
4. 在本演練中，將 Source (來源) 設定為 Anywhere (隨處)，以指示負載平衡器接受來自任何 IP 地址的傳入流量。

3. 返回 AWS OpsWorks Stacks 主控台。在 Layers (Layer) 頁面上，選擇 layer 的 Network (網路) 連結，然後選擇 Edit (編輯)。
4. 在 Elastic Load Balancing 下，選擇您在步驟 1 建立的負載平衡器，然後選擇 Save (儲存)。



將負載平衡器連接到 layer 之後， AWS OpsWorks Stacks 會自動註冊 layer 的目前執行個體，並在新執行個體上線時新增執行個體。

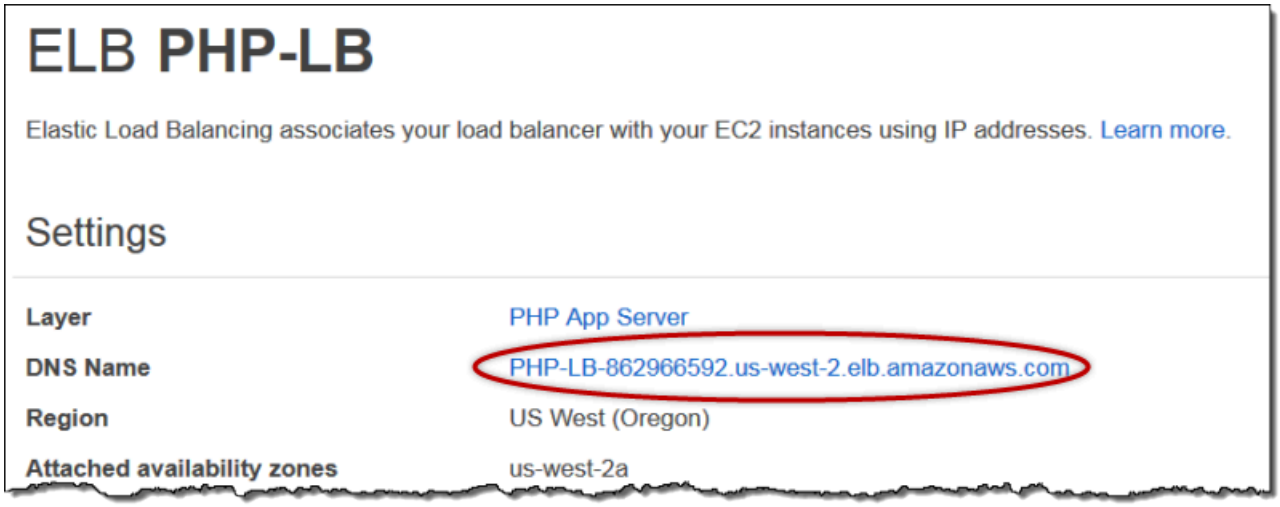
5. 在 Layers (Layer) 頁面上，按一下負載平衡器的名稱，以開啟其詳細資訊頁面。當註冊完成且執行個體通過運作狀態檢查時， AWS OpsWorks Stacks 會在負載平衡器頁面上的執行個體旁顯示綠色核取記號。



您現在可以將請求傳送到負載平衡器來執行 SimplePHPApp。

透過負載平衡器執行 SimplePHPApp

1. 再次開啟負載平衡器的詳細資訊頁面 (若尚未開啟)。
2. 在屬性頁面上，驗證執行個體的運作狀態檢查狀態，然後按一下負載平衡器的 DNS 名稱來執行 SimplePHPApp。負載平衡器會將請求轉送到 PHP App Server 執行個體，並傳回回應，這看起來應該與您按一下 PHP App Server 執行個體的公有 IP 地址時收到的回應完全相同。



> **Note**
>
> AWS OpsWorks Stacks 也支援 HAProxy 負載平衡器，這對於某些應用程式可能具有優勢。如需詳細資訊，請參閱HAProxy AWS OpsWorks Stacks Layer。

步驟 4.2：新增 PHP 應用程式伺服器執行個體

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

現在負載平衡器已就緒，您可以將更多執行個體新增至 PHP App Server layer 來擴展堆疊。從您的觀點而言，操作會無縫進行。每次新的 PHP App Server 執行個體上線時， AWS OpsWorks Stacks 會自動向負載平衡器註冊它，並部署 SimplePHPApp，讓伺服器可以立即開始處理傳入流量。為了簡潔起

見，本主題說明如何新增一個額外的 PHP App Server 執行個體，但您可以使用相同的方法來新增所需的數量。

將另一個執行個體新增至 PHP App Server layer

1. 在執行個體頁面上，按一下 PHP App Server 下的 + 執行個體。
2. 接受預設設定，然後按一下 Add Instance (新增執行個體)。
3. 按一下 start (啟動) 以啟動執行個體。

步驟 4.3：監控 MyStack

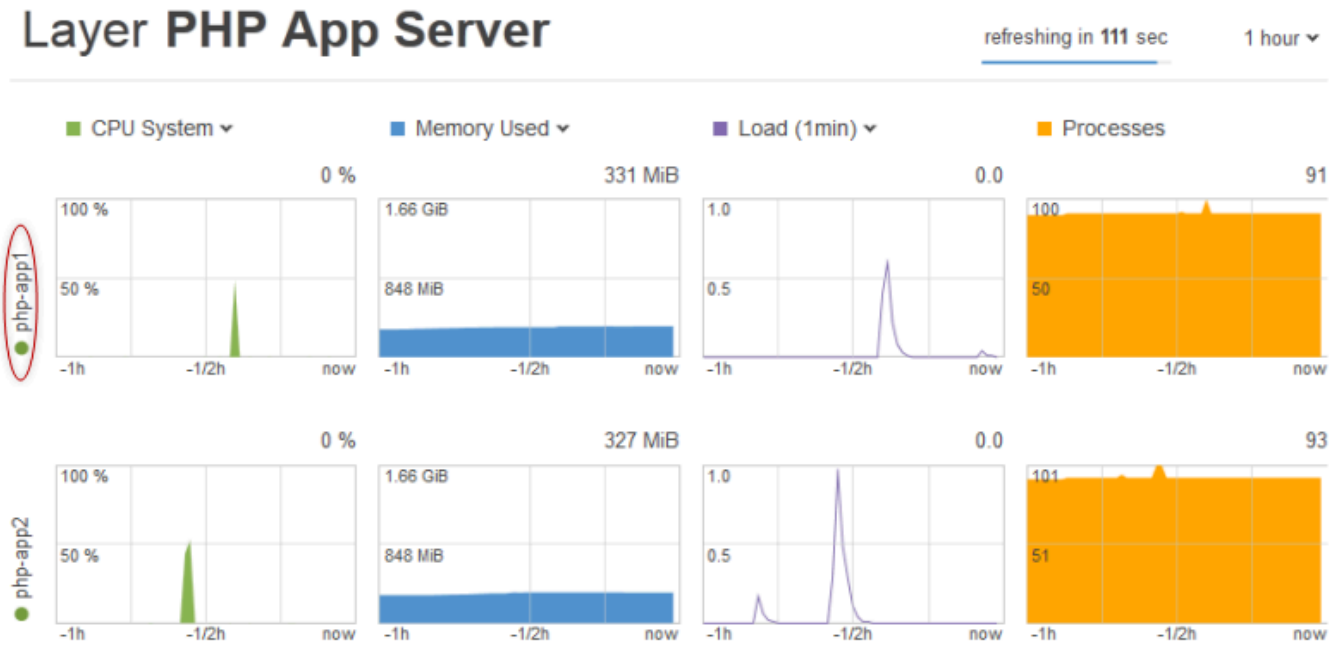
> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 使用 Amazon CloudWatch 來提供堆疊的指標，並在監控頁面上摘要這些指標，方便您使用。您可以檢視整個堆疊、指定 layer 或指定執行個體的指標。

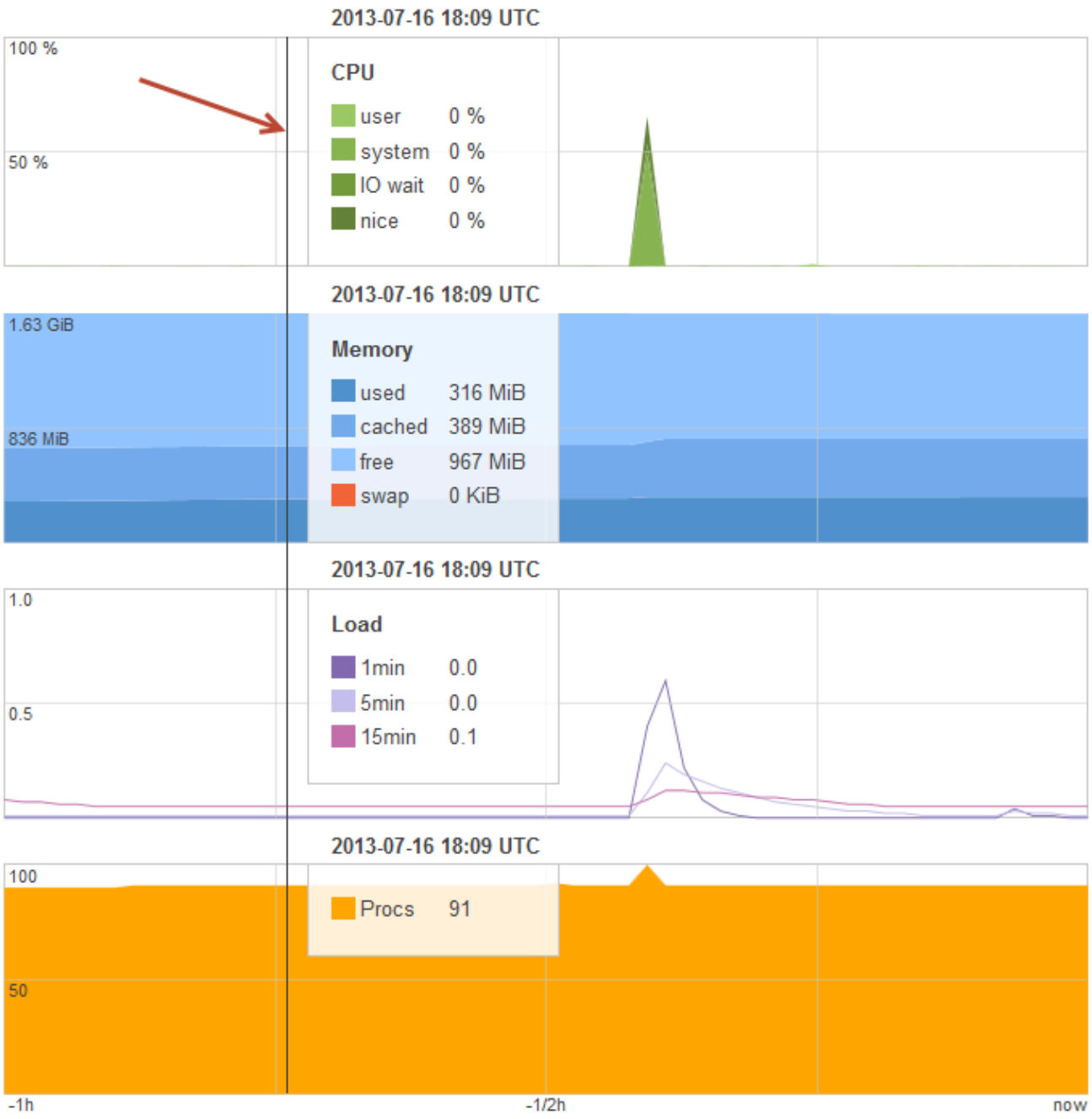
監控 MyStack

1. 在導覽窗格中，按一下 Monitoring (監控)，以顯示一組具有每 layer 平均指標的圖形。您可以使用 CPU System (CPU 系統)、Memory Used (已使用記憶體) 和 Load (負載) 的選單來顯示不同的相關指標。

2. 按一下 PHP App Server，查看該 layer 每個執行個體的指標。



3. 按一下 php-app1，查看該執行個體的指標。您可以移動滑桿來查看任何特定時間點的指標。


Instance php-app1
refreshing in
2013-07-16 18:09 UTC
100 %
50 %
CPU
user 0 %
system 0 %
IO wait 0 %
nice 0 %
2013-07-16 18:09 UTC
1.63 GiB
836 MiB
Memory
used 316 MiB
cached 389 MiB
free 967 MiB
swap 0 KiB
2013-07-16 18:09 UTC
1.0
0.5
Load
1min 0.0
5min 0.0
15min 0.1
2013-07-16 18:09 UTC
100
50
Procs 91
-1h
-1/2h
now

> **Note**
>
> AWS OpsWorks Stacks 也支援 Ganglia 監控伺服器，這對於某些應用程式可能具有優勢。如需詳細資訊，請參閱Ganglia 層。

步驟 5：刪除 MyStack

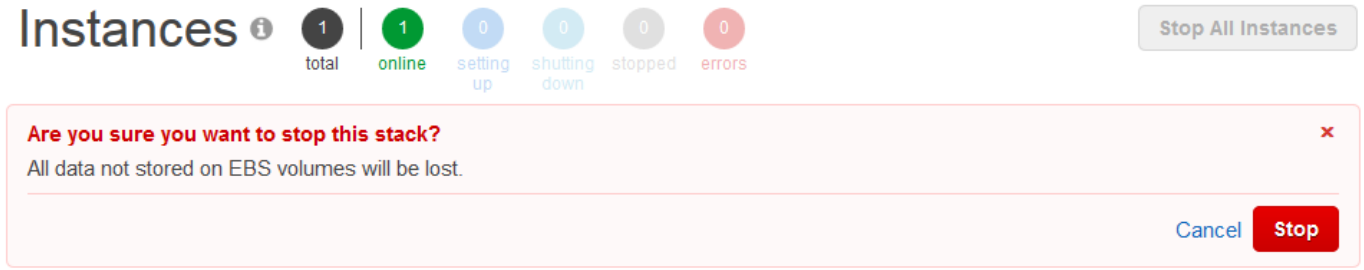
> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

一旦您開始使用 AWS 資源，例如 Amazon EC2 執行個體，就會根據用量收費。若您目前暫時不需要使用，建議您停止執行個體，以避免產生任何不必要的費用。若您不再需要堆疊，您可以刪除它。

刪除 MyStack

1. 停止所有執行個體

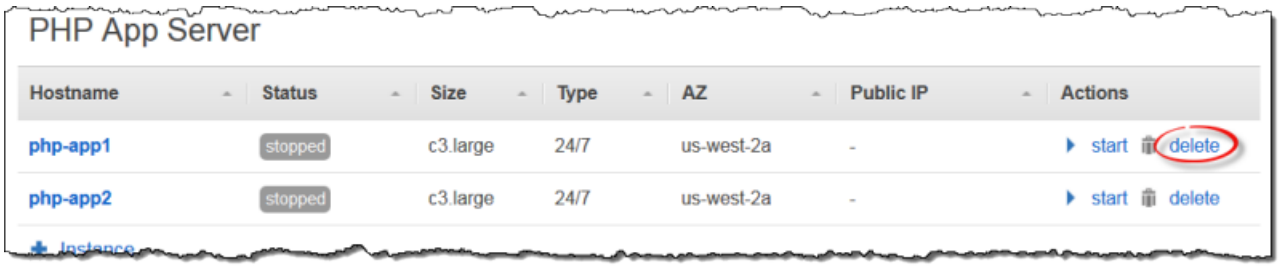
   在 Instances (執行個體) 頁面上，按一下 Stop All Instances (停止所有執行個體)，然後在出現提示確認操作時，按一下 Stop (停止)。



   按一下停止後， AWS OpsWorks Stacks 會終止相關聯的 Amazon EC2 執行個體，但不會終止任何相關聯的資源，例如彈性 IP 地址或 Amazon EBS 磁碟區。

2. 刪除所有執行個體

   停止執行個體只會終止相關聯的 Amazon EC2 執行個體。在執行個體狀態顯示已停止時，您必須刪除每個執行個體。在 PHP App Server (PHP 應用程式伺服器) layer 中，在 php-app1 執行個體的 Actions (動作) 資料行中按一下 delete (刪除)。

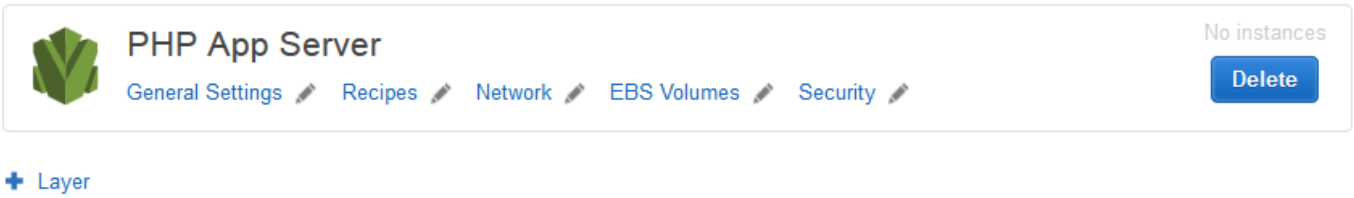


AWS OpsWorks Stacks 接著會要求您確認刪除，並顯示任何相依的資源。您可以選擇保留任何或全部的資源。此範例沒有依存資源，因此只需按一下 Delete (刪除)。

針對 php-app2、MySQL 執行個體、db-master1 重複此程序。請注意，db-master1 具有相關聯的 Amazon Elastic Block Store 磁碟區，這是預設選取的。將其維持在選取狀態，以伴隨執行個體一同刪除它。

3. 刪除 Layer。

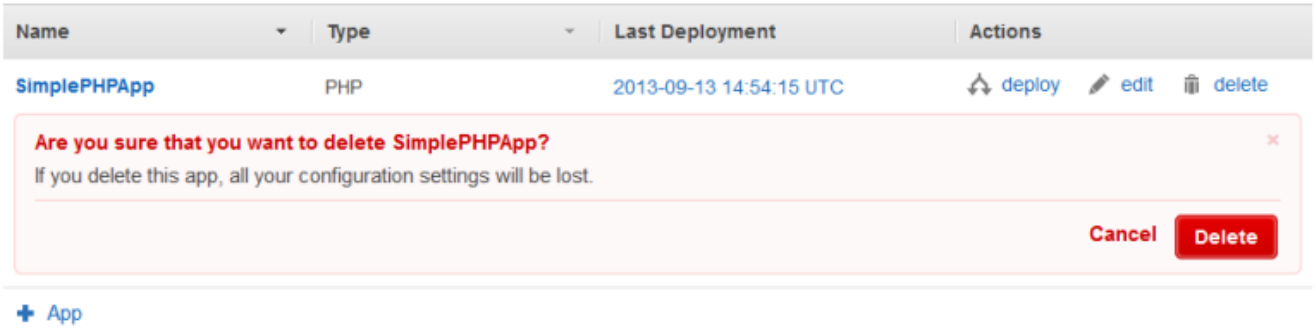
   在 Layers (Layer) 頁面上，按一下 Delete (刪除)，然後按一下 Delete (刪除) 確認。



   針對 MySQL layer 重複此程序。

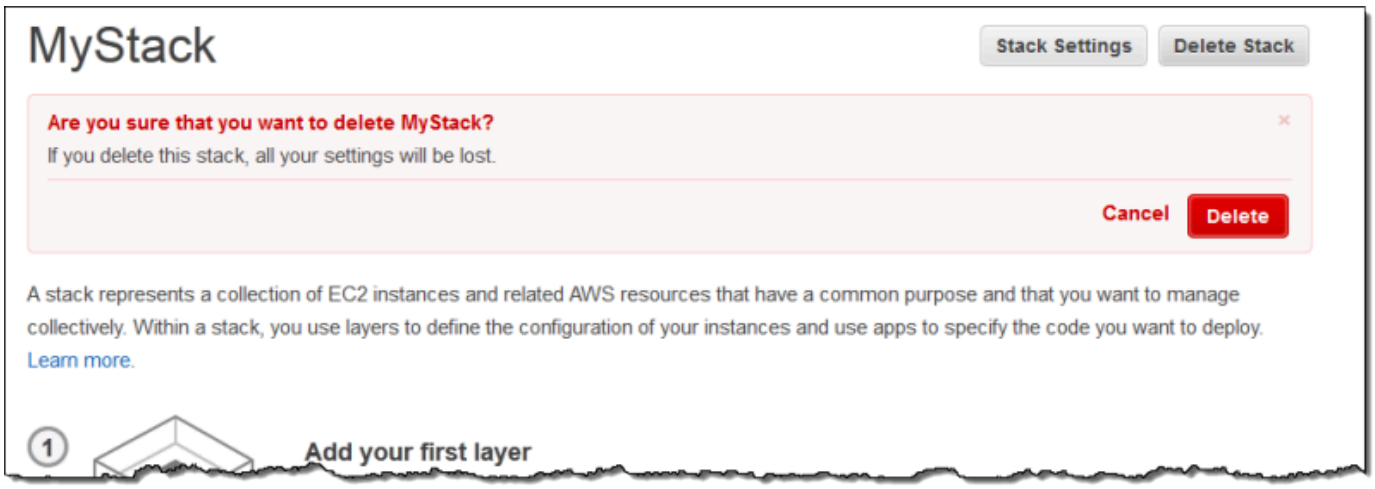
4. 刪除應用程式

   在 Apps (應用程式) 頁面上，按一下 SimplePHPApp 應用程式的 Actions (動作) 資料行中的 delete (刪除)，然後按一下 Delete (刪除) 確認。



5. 刪除 MyStack

   在 Stack (堆疊) 頁面上，按一下 Delete Stack (刪除堆疊)，然後按一下 Delete (刪除) 確認。



這份演練到此結束。

建立您的第一個 Node.js 堆疊

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

此範例說明如何建立支援 Node.js 應用程式伺服器的 Linux 堆疊，以及如何部署簡易應用程式。堆疊包含下列元件：

- 具有兩個執行個體的 Node.js App Server layer
- Elastic Load Balancing 負載平衡器可將流量分配到應用程式伺服器執行個體
- 提供後端資料庫的 Amazon Relational Database Service (Amazon RDS) 服務層

主題

- 先決條件
- 實作應用程式
- 建立資料庫伺服器和負載平衡器
- 建立堆疊
- 部署應用程式

- 後續步驟？

先決條件

本演練的假設如下：

- 您有一個 AWS 帳戶，並對如何使用 AWS OpsWorks Stacks 有基本的了解。

  如果您是初次使用 AWS OpsWorks Stacks 或 AWS，請完成 中的簡介教學課程來學習基本知識Chef 11 Linux 堆疊入門。
- 您具有如何實作 Node.js 應用程式的基本理解。

  若您是初次使用 Node.js，請完成簡介教學 (例如 Node: Up and Running) 來學習基本概念。
- 您已在您計劃於此範例中使用的 AWS 區域內建立至少一個堆疊。

  當您在區域中建立第一個堆疊時， AWS OpsWorks Stacks 會為每個 layer 類型建立 Amazon Elastic Compute Cloud (Amazon EC2) 安全群組。您需要這些安全群組才能建立 Amazon RDS 資料庫 (DB) 執行個體。如果您是初次使用 AWS OpsWorks Stacks，建議您在此範例中使用與您在 中遵循教學課程時相同的區域Chef 11 Linux 堆疊入門。若您希望使用新的區域，請在區域內建立新的堆疊。堆疊不需要具備任何 layer 或執行個體。建立堆疊後， AWS OpsWorks Stacks 會自動將一組安全群組新增至區域。
- 您會在預設 VPC 中建立您的堆疊。

  您可以針對本演練使用 EC2-Classic，但一部分的詳細資訊可能會有所不同。例如，使用 EC2-Classic 時，您必須指定執行個體的可用區域 (AZ) 而非其子網路。
- 您的 IAM 使用者具有 Stacks AWS OpsWorks 的完整存取許可。

  基於安全考量，我們強烈建議您不要在本演練中使用您帳戶的根登入資料。反之，請建立具有 AWS OpsWorks Stacks 完整存取許可的使用者，並將這些登入資料與 Stacks AWS OpsWorks 搭配使用。如需詳細資訊，請參閱建立 管理使用者。

實作應用程式

本演練使用連接至 Amazon RDS 資料庫執行個體的簡單 Express 應用程式，並列出執行個體的資料庫。

若要實作應用程式，請在您工作站上方便的位置建立名為 nodedb 的目錄，並將下列三個檔案新增至其中。

主題

- 套件描述項
- 配置檔案
- 程式碼檔案

套件描述項

若要建立應用程式的套件描述項，請在 nodedb 目錄中新增名為 package.json 的檔案，其中包含以下內容。package.json 為 Express 應用程式的必要項目，因此必須位於應用程式的根目錄中。

```
{
  "name": "Nodejs-DB",
  "description": "Node.js example application",
  "version": "0.0.1",
  "dependencies": {
    "express": "*",
    "ejs": "*",
    "mysql": "*"
  }
}
```

此 package.json 範例相當精簡。它定義了所需的 name 及 version 屬性，並列出依存套件：

- express 參考 Express 套件。
- ejs 參考 EJS 套件，應用程式會用它來將文字插入 HTML 配置檔案。
- mysql 參考 node-mysql 套件，應用程式會用它來連線至 RDS 執行個體。

如需套件描述項檔案的詳細資訊，請參閱 package.json。

配置檔案

若要建立應用程式的配置檔案，請將 views 目錄新增至 nodedb 目錄，然後將名為 index.html 的檔案新增至 views，其中內容如下：

```
<!DOCTYPE html>
<html>
<head>
```

```
  <title>AWS Opsworks Node.js Example</title>
</head>
<body>
  <h1>AWS OpsWorks Node.js Example</h1>
    <p>Amazon RDS Endpoint: <i><%= hostname %></i></p>
    <p>User: <i><%= username %></i></p>
    <p>Password: <i><%= password %></i></p>
    <p>Port: <i><%= port %></i></p>
    <p>Database: <i><%= database %></i></p>

    <p>Connection: <%= connectionerror %></p>
    <p>Databases: <%= databases %></p>
</body>
</html>
```

在此範例中，配置檔案是簡單的 HTML 文件，會顯示來自 Amazon RDS 的一些資料。每個 <%= ... => 元素都代表在應用程式程式碼檔案中定義的變數值。我們會在接下來的步驟中建立。

程式碼檔案

若要建立應用程式的程式碼檔案，請在 nodedb 目錄中新增名為 server.js 的檔案，其中包含以下內容。

> **Important**
>
> 透過 AWS OpsWorks Stacks，Node.js 應用程式的主要程式碼檔案必須命名server.js並位於應用程式的根資料夾中。

```
var express = require('express');
var mysql = require('mysql');
var dbconfig = require('opsworks'); //[1] Include database connection data
var app = express();
var outputString = "";

app.engine('html', require('ejs').renderFile);

//[2] Get database connection data
app.locals.hostname = dbconfig.db['host'];
app.locals.username = dbconfig.db['username'];
app.locals.password = dbconfig.db['password'];
```

```
app.locals.port = dbconfig.db['port'];
app.locals.database = dbconfig.db['database'];
app.locals.connectionerror = 'successful';
app.locals.databases = '';

//[3] Connect to the Amazon RDS instance
var connection = mysql.createConnection({
    host: dbconfig.db['host'],
    user: dbconfig.db['username'],
    password: dbconfig.db['password'],
    port: dbconfig.db['port'],
    database: dbconfig.db['database']
});

connection.connect(function(err)
{
    if (err) {
        app.locals.connectionerror = err.stack;
        return;
    }
});

// [4] Query the database
connection.query('SHOW DATABASES', function (err, results) {
    if (err) {
        app.locals.databases = err.stack;
    }

    if (results) {
        for (var i in results) {
            outputString = outputString + results[i].Database + ', ';
        }
        app.locals.databases = outputString.slice(0, outputString.length-2);
    }
});

connection.end();

app.get('/', function(req, res) {
    res.render('./index.html');
});

app.use(express.static('public'));
```

```
//[5] Listen for incoming requests
app.listen(process.env.PORT);
```

範例顯示資料庫連線資訊，並查詢資料庫伺服器及顯示伺服器的資料庫。您可以輕易的將其一般化，以和資料庫互動 (若需要的話)。下列備註指向先前程式碼中具有編號的註解。

[1] 包含資料庫連線資料

`require` 陳述式包含資料庫連線資料。如稍後所述，當您將資料庫執行個體連接至應用程式時，AWS OpsWorks Stacks 會將連線資料放入名為 的檔案中`opsworks.js`，看起來類似以下內容：

```
exports.db = {
  "host":"nodeexample.cdlqlk5uwd0k.us-west-2.rds.amazonaws.com",
  "database":"nodeexampledb",
  "port":3306,
  "username":"opsworksuser",
  "password":"your_pwd",
  "reconnect":true,
  "data_source_provider":"rds",
  "type":"mysql"}
```

`opsworks.js` 位於應用程式的 `shared/config` 目錄，`/srv/www/app_shortname/shared/config` 中。不過， AWS OpsWorks Stacks `opsworks.js`會將符號連結放入應用程式的根目錄中，因此您可以只使用 來包含物件`require 'opsworks'`。

[2] 取得資料庫連線資料

這一組陳述式會將 db 物件的值指派給一組 `app.locals` 屬性，其中每一項都會對應到 `index.html` 檔案中的其中一個 <%= ... %> 元素，以顯示 `opsworks.js` 的連線資料。轉譯後的文件會使用對應的屬性值取代 <%= ... %> 元素。

[3] 連線至 Amazon RDS 執行個體

範例使用 `node-mysql` 存取資料庫。為連線到資料庫，範例會透過將連線資料傳遞給 `connection`，然後呼叫 `createConnection` 建立連線，來建立 `connection.connect` 物件。

[4] 查詢資料庫

在建立連線後，範例會呼叫 `connection.query` 查詢資料庫。此範例只會查詢伺服器的資料庫名稱。`query` 會傳回 `results` 物件的陣列，每個資料庫一個，並且資料庫名稱會指派給 `Database`

屬性。範例會串連名稱，並將他們指派給 `app.locals.databases,`，在轉譯後的 HTML 頁面上顯示清單。

在此範例中，有五個資料庫，您在建立 RDS 執行個體時指定的`nodeexampledb`資料庫，以及由 Amazon RDS 自動建立的四個其他資料庫。

[5] 接聽傳入請求

最後一個陳述式會在指定連接埠上接聽傳入請求。您不必指定明確的連接埠值。當您將應用程式新增至堆疊時，您可以指定應用程式是否支援 HTTP 或 HTTPS 請求。 AWS OpsWorks Stacks 接著會將`PORT`環境變數設定為 80 (HTTP) 或 443 (HTTPS)，而且您可以在應用程式中使用該變數。

您可以監聽其他連接埠，但 Node.js App Server 層的內建安全群組 AWS-OpsWorks-nodejs-App-Server 僅允許傳入使用者流量至連接埠 80、443 和 22 (SSH)。若要允許傳入使用者流量到其他連接埠，請使用適當的傳入規則建立安全群組，並將其指派給 Node.js App Server layer。請勿透過編輯內建安全群組來修改傳入規則。每次建立堆疊時， AWS OpsWorks Stacks 都會以標準設定覆寫內建的安全群組，因此您所做的任何變更都會遺失。

> **Note**
>
> 您可以在您建立或更新關聯應用程式時，將自訂環境變數與您的應用程式建立關聯。您也可以使用自訂 JSON 和自訂配方將資料傳遞到您的應用程式。如需詳細資訊，請參閱傳遞資料到應用程式。

建立資料庫伺服器和負載平衡器

此範例使用 Amazon RDS 資料庫伺服器和 Elastic Load Balancing 負載平衡器執行個體。您必須分別建立每個執行個體，然後將其併入您的堆疊。本節說明如何建立新的資料庫和負載平衡器執行個體。雖然您可以改為使用現有的執行個體，但我們建議您閱讀整個程序，以確保那些執行個體的設定正確。

以下說明如何建立精簡設定，足以用於此範例的 RDS 資料庫執行個體。如需詳細資訊，請參閱 Amazon RDS 使用者指南。

建立 RDS 資料庫執行個體

1. 開啟 主控台。

   開啟 Amazon RDS 主控台，並將區域設定為美國西部（奧勒岡）。在導覽窗格中，選擇 RDS Dashboard (RDS 儀表板)，然後選擇 Launch DB Instance (啟動資料庫執行個體)。

2. 指定資料庫引擎。

   選擇 MySQL Community Edition 做為資料庫引擎。

3. 拒絕異地同步備份部署。

   選擇 No, this instance... (否，此執行個體...)，然後選擇 Next (下一步)。您在此範例中不需要使用異地同步備份部署。

4. 設定基本設定。

   在 DB Instance Details (資料庫執行個體詳細資訊) 頁面上，指定下列設定：

   - DB Instance Class (資料庫執行個體類別)：db.t2.micro。
   - Multi-AZ Deployment (異地同步備份部署)：No (否)
   - Allocated Storage (配置儲存體)：**5** GB
   - DB Instance Identifier (資料庫執行個體識別符)：**nodeexample**
   - Master Username (主要使用者名稱)：**opsworksuser**
   - Master Password (主要密碼)：您選擇的密碼

   記錄執行個體識別符、使用者名稱、密碼以供稍後使用，接受其他選項的預設設定，然後選擇 Next (下一步)。

5. 設定進階設定。

   在 Configure Advanced Settings (設定進階設定) 頁面上，指定下列設定：

   - 資料庫名稱：**nodeexampledb**
   - DB Security Group(s) (資料庫安全群組)：AWS-OpsWorks-DB-Master-Server

   > **Note**
   >
   > AWS-OpsWorks-DB-Master-Server 安全群組只會允許您堆疊的執行個體存取資料庫。若您希望直接存取資料庫，請將額外的安全群組連接到 RDS 資料庫執行個體，並搭配適當的傳入規則。如需詳細資訊，請參閱 Amazon RDS 安全群組。您也可以透過將執行個體置放在 VPC 中，來控制存取。如需詳細資訊，請參閱在 VPC 中執行堆疊。

記錄資料庫名稱以供稍後使用，接受其他設定的預設值，然後選擇 Launch DB Instance (啟動資料庫執行個體)。

下列程序說明如何為此範例建立 Elastic Load Balancing 負載平衡器。如需詳細資訊，請參閱《Elastic Load Balancing 使用者指南》。

建立負載平衡器

1. 開啟 Amazon EC2 主控台。

   開啟 Amazon EC2 主控台，並確保區域設定為美國西部 （奧勒岡）。在導覽窗格中，選擇 Load Balancers (負載平衡器)，然後選擇 Create Load Balancer (建立負載平衡器)。

2. 定義負載平衡器。

   在 Define Load Balancer (定義負載平衡器) 頁面上，指定下列設定。

   - 名稱 – **`Node-LB`**
   - 在內部建立 LB – 我的預設 VPC

   接受其他選項的預設設定，然後選擇 Next (下一步)。

3. 指派安全群組。

   在 Assign Security Groups (指派安全群組) 頁面上，指定下列群組：

   - default VPC security group (預設 VPC 安全群組)
   - AWS-OpsWorks-nodejs-App-Server

   選擇下一步。在 Configure Security Settings (設定安全設定) 頁面上，選擇 Next (下一步)。您在此範例中不需要使用安全接聽程式。

4. 設定運作狀態檢查。

   在 Configure Health Check (設定運作狀態檢查) 頁面上，將 Ping Path (Ping 路徑) 設為 **/**，並接受其他設定的預設值。選擇下一步。在 Add EC2 Instances (新增 EC2 執行個體) 頁面上，選擇 Next (下一步)。在新增標籤頁面上，選擇檢閱和建立。 AWS OpsWorks Stacks 會處理將 EC2 執行個體新增至負載平衡器的任務，在此範例中您不需要標籤。

5. 建立負載平衡器。

   在 Review (頁面) 上，選擇 Create (建立) 以建立負載平衡器。

建立堆疊

您現在已具備所有需要用來建立堆疊的元件。

建立堆疊

1. 登入 AWS OpsWorks Stacks 主控台。

   登入 AWS OpsWorks Stacks 主控台，然後選擇 Add Stack (新增堆疊)。

2. 建立堆疊。

   若要建立新的堆疊，請選擇 Chef 11 stack (Chef 11 堆疊)，然後指定下列設定。

   - – **`NodeStack`**
   - 區域 – 美國西部 （ 奧勒岡 ）

     您可以在任何 AWS 區域中建立堆疊，但我們建議美國西部 （ 奧勒岡 ） 進行教學課程。

   選擇 Add Stack (新增堆疊)。如需堆疊組態設定的詳細資訊，請參閱建立新的堆疊。

3. 使用連接的負載平衡器新增 Node.js App Server layer。

   在 NodeStack 頁面上，選擇 Add a layer (新增 layer)，然後指定下列設定：

   - Layer 類型 – Node.js 應用程式伺服器
   - Elastic Load Balancer – Node-LB

   接受其他設定的預設值，然後選擇 Add Layer (新增 Layer)。

4. 將執行個體新增至 layer 並啟動。

   在導覽窗格中選擇 Instances (執行個體)，然後新增兩個執行個體至 Rails 應用程式伺服器 layer，如下所示。

   1. 在 Node.js App Server 下，選擇新增執行個體。

將 Size (大小) 設為 t2.micro，接受其他設定的預設值，然後選擇 Add Instance (新增執行個體)。

2. 選擇 +Instance (+執行個體)，然後新增第二個 t2.micro 執行個體至位於不同子網路中的 layer。

   這會將執行個體置放在不同的可用區域 (AZ) 中。

3. 選擇 Add instance (新增執行個體)。
4. 若要啟動兩個執行個體，請選擇 Start All Instances (啟動所有執行個體)。

您已將 Elastic Load Balancing 負載平衡器指派給此層。當執行個體進入或離開線上狀態時，AWS OpsWorks Stacks 會自動向負載平衡器註冊或取消註冊執行個體。

> **Note**
>
> 針對生產堆疊，我們建議您將您的應用程式伺服器執行個體分散到多個 AZ 中。若使用者無法連線到其中一個 AZ，負載平衡器會將傳入流量路由至剩餘區域中的執行個體，讓您的網站可繼續運作。

5. 向堆疊註冊 RDS 資料庫執行個體。

在導覽窗格中選擇 Resources (資源) 並向堆疊註冊 RDS 資料庫執行個體，如下所示。

1. 選擇 RDS 標籤，然後選擇 Show Unregistered RDS DB (顯示未註冊的 RDS 資料庫) 執行個體。
2. 選擇 nodeexampledb 執行個體，然後指定下列設定：
   - 使用者 – 您在建立執行個體時指定的主要使用者名稱；在此範例中為 **`opsworksuser`**。
   - 密碼 – 您在建立執行個體時指定的主要密碼。
3. 選擇向 Stack 註冊，將 RDS 資料庫執行個體新增至堆疊做為 Amazon RDS 服務層。

> **Warning**
>
> AWS OpsWorks Stacks 不會驗證使用者或密碼值，只會將值傳遞給應用程式。若您輸入不正確，您的應用程式將無法連線至資料庫。

若要將 RDS 資料庫執行個體新增至堆疊做為 Amazon RDS 服務層，請選擇向 Stack 註冊。

部署應用程式

您必須將應用程式存放在遠端儲存庫中。當您部署它時，AWS OpsWorks Stacks 會將程式碼和相關檔案從儲存庫部署到應用程式伺服器執行個體。為了方便起見，此範例使用公有 Amazon Simple Storage Service (Amazon S3) 封存做為儲存庫，但您也可以使用數個其他儲存庫類型，包括 Git 和 Subversion。如需詳細資訊，請參閱應用程式來源。

部署應用程式

1. 將應用程式封裝在封存檔中。

   建立 `.zip` 目錄和子目錄的 nodedb 封存，命名為 nodedb.zip。您也可以使用其他類型的封存檔，包括 gzip、bzip2 和 tarball。請注意，AWS OpsWorks Stacks 不支援未壓縮的 tarball。如需詳細資訊，請參閱應用程式來源。

2. 將封存檔案上傳至 Amazon S3。

   `nodedb.zip` 上傳至 Amazon S3 儲存貯體，將檔案設為公有，然後複製檔案的 URL 以供日後使用。如需如何建立儲存貯體以及上傳檔案的詳細資訊，請參閱開始使用 Amazon Simple Storage Service。

   > **Note**
   >
   > AWS OpsWorks Stacks 也可以從 Amazon S3 儲存貯體部署私有檔案，但為了簡單起見，此範例會使用公有檔案。如需詳細資訊，請參閱應用程式來源。

3. 建立 AWS OpsWorks Stacks 應用程式。

   返回 AWS OpsWorks Stacks 主控台，在導覽窗格中，選擇應用程式，然後選擇新增應用程式。指定下列設定：

   - Name (名稱) – `NodeDB`。

     此字串為應用程式的顯示名稱。大多數情況下，您需要應用程式的簡短名稱，Stacks AWS OpsWorks 會透過將所有字元轉換為小寫並移除標點符號，從顯示名稱產生該名稱。針對此範例，短名為 nodedb。若要驗證應用程式的短名，在建立應用程式之後，請在 Apps (應用程式) 頁面上選擇應用程式以顯示其詳細資訊頁面。

   - 類型 – `Node.js`.
   - Data source type (資料來源類型) – `RDS`。
   - 資料庫執行個體 – 選擇您先前註冊的 Amazon RDS 資料庫執行個體。

- Database name (資料庫名稱) – 指定您先前建立的資料庫名稱，此範例中為 `nodeexampledb`。
- Repository type (儲存庫類型) – `Http Archive`。

  您必須將此儲存庫類型用於公有 Amazon S3 檔案。`S3 Archive` 類型僅用於私有封存。
- 儲存庫 URL – 封存檔案的 Amazon S3 URL。

針對剩餘設定使用預設值，然後按一下 Add App (新增應用程式) 以建立應用程式。

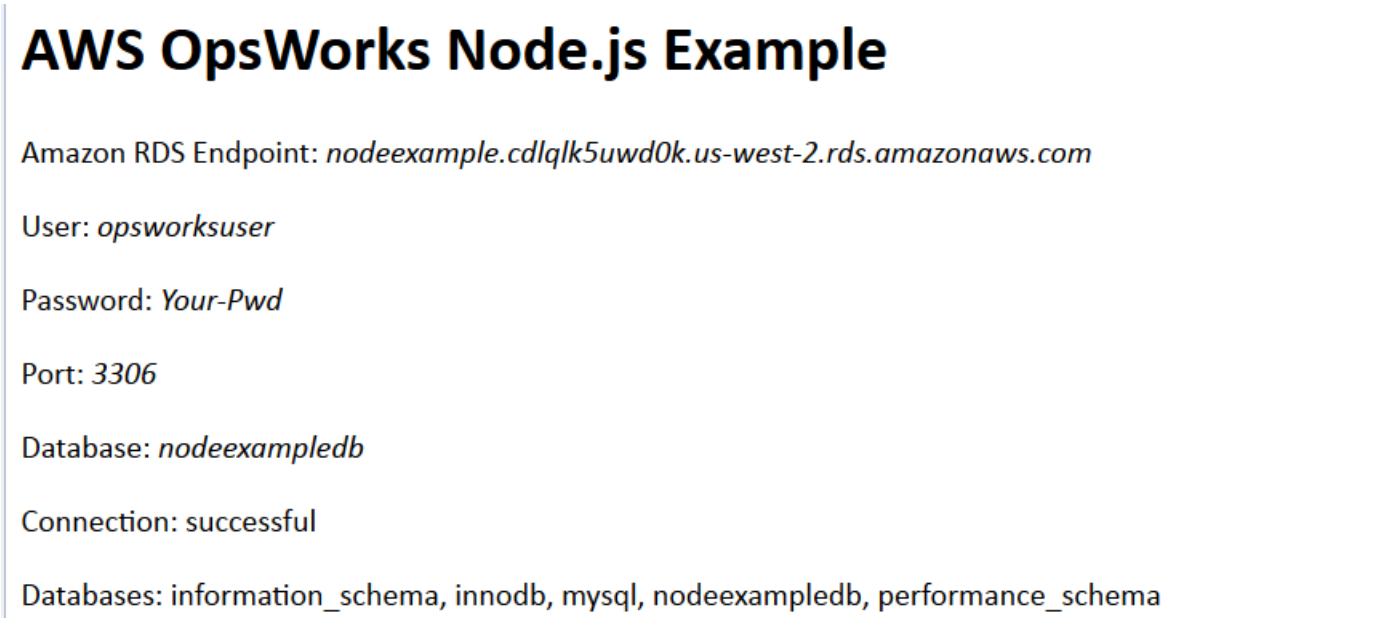
4. 部署應用程式。

   前往 Apps (應用程式) 頁面，然後在 NodeDB 應用程式的 Actions (動作) 資料行中，選擇 deploy (部署)。然後選擇部署，將應用程式部署到伺服器執行個體。 AWS OpsWorks Stacks 會在每個執行個體上執行部署配方，從儲存庫下載應用程式並重新啟動伺服器。當每個執行個體都有綠色的核取記號，並且 Status (狀態) 為 successful (成功) 時，部署便已完成，應用程式已準備好開始處理請求。

   > **Note**
   >
   > 如果部署失敗，請選擇日誌欄中的顯示以顯示部署的 Chef 日誌。錯誤資訊位於接近底部的位置。

5. 開啟 應用程式。

   若要開啟應用程式，請選擇 Layer (Layer)，選擇負載平衡器，然後選擇負載平衡器的 DNS 名稱，將 HTTP 請求傳送到負載平衡器。您應該會看到類似下列的內容。



> **Note**
>
> AWS OpsWorks Stacks 會在設定期間自動將應用程式部署到新的執行個體。只需要針對線上執行個體進行手動部署。如需詳細資訊，請參閱部署應用程式。如需部署的一般討論，包含一些更複雜的部署策略，請參閱管理和部署應用程式與技術指南。

後續步驟？

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本演練帶領您完成設定簡易 Node.js 應用程式伺服器堆疊的基本操作。以下是有關後續作業的一些建議。

檢查 Node.js 的內建技術指南

如果您想要詳細了解執行個體的設定方式，請參閱 layer 的內建技術指南 opsworks_nodejs，其中包含 Stacks AWS OpsWorks 用來安裝和設定軟體的配方和相關檔案，以及內建部署技術指南，其中包含 AWS OpsWorks Stacks 用來部署應用程式的配方。

自訂伺服器組態

範例堆疊相當基本。針對生產用途，您可以會希望自訂堆疊。如需詳細資訊，請參閱自訂 AWS OpsWorks Stacks。

新增 SSL 支援

您可以為您的應用程式啟用 SSL 支援，並在建立應用程式時為 AWS OpsWorks Stacks 提供適當的憑證。 AWS OpsWorks Stacks 接著會在適當的目錄中安裝憑證。如需詳細資訊，請參閱使用 SSL。

新增記憶體內快取

生產層級網站通常會透過在記憶體內鍵/值存放區 (例如 Redis 或 Memcache) 中快取資料來改善效能。您可以搭配 Stacks AWS OpsWorks 堆疊使用 。如需詳細資訊，請參閱ElastiCache Redis及Memcached。

使用更複雜的部署策略

範例使用簡易的應用程式部署策略，將更新同時部署到每個執行個體。這種方法非常簡單且快速，但沒有錯誤的餘地。若部署失敗或更新發生任何問題，每個您生產堆疊中的執行個體都會受到影響，可能中斷或停用您的網站，直到您修正問題為止。如需部署策略的詳細資訊，請參閱管理和部署應用程式與技術指南。

擴展 Node.js 應用程式伺服器層

您可以透過各種方式延伸 layer。例如，您可以實作配方，以在執行個體上執行指令碼，或實作 Chef 部署勾點以自訂應用程式部署。如需詳細資訊，請參閱擴充 Layer。

定義環境變數

您可以透過定義關聯應用程式的環境變數，來將資料傳遞給您的應用程式。當您部署應用程式時，AWS OpsWorks Stacks 會匯出這些變數，以便您可以從應用程式存取它們。如需詳細資訊，請參閱使用 環境變數。

自訂 AWS OpsWorks Stacks

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 內建 layer 提供的標準功能足以用於許多用途。不過，您可能會遇到下列一或多種情況：

- 內建 layer 的標準組態雖足夠但不理想，您希望針對您的特定需求最佳化。

  例如，您可能想要透過指定自己的值來調整靜態 Web 伺服器層的 Nginx 伺服器組態，例如工作者程序數目上限或 `keepalivetimeout`值。
- 內建 layer 的功能很好，但您想要安裝額外的套件或執行一些自訂安裝程式碼擴展它。

  例如，您可能也想要安裝 Redis 伺服器來擴展 PHP App Server layer。
- 您有任何內建 layer 都不處理的要求。

  例如， AWS OpsWorks Stacks 不包含某些熱門資料庫伺服器的內建層。您可以建立自訂 layer，在 layer 的執行個體上安裝這些伺服器。
- 您執行的是 Windows 堆疊，它只支援自訂 layer。

AWS OpsWorks Stacks 提供各種自訂 layer 的方式，以符合您的特定需求。下列範例依增加複雜度和功能的順序列出：

> **Note**
>
> 這些方法有部分只適用於 Linux 堆疊。請參閱以下主題以了解詳細資訊。

- 使用自訂 JSON AWS OpsWorks 覆寫預設的 Stacks 設定。
- 使用覆寫預設 Stacks 設定的屬性檔案實作自訂 Chef AWS OpsWorks 技術指南。
- 使用覆寫或擴展預設 Stacks 範本的範本實作自訂 Chef AWS OpsWorks 技術指南。
- 使用執行 shell 指令碼的簡單配方實作自訂的 Chef 技術指南。

- 使用執行任務 (例如建立和設定目錄、安裝套件、建立組態檔案、部署應用程式等等) 的配方，實作自訂的 Chef 技術指南。

您也可以覆寫配方，視堆疊的 Chef 版本和作業系統而定。

- 使用 Chef 0.9 和 11.4 堆疊，您無法透過以相同的技術指南和配方名稱實作自訂配方，來覆寫內建配方。

  對於每個生命週期事件， AWS OpsWorks Stacks 一律會先執行內建配方，接著執行任何自訂配方。由於這些 Chef 版本不會執行兩次有相同技術指南和配方名稱的配方，所以內建配方優先，不執行自訂配方。
- 您可以在 Chef 11.10 堆疊上覆寫內建配方。

  如需詳細資訊，請參閱技術指南安裝與優先順序。
- 您無法在 Windows 堆疊上覆寫內建配方。

  Stacks 處理 Chef AWS OpsWorks 為 Windows 堆疊執行的方式不允許覆寫內建配方。

> **Note**
>
> 由於許多技術都使用自訂技術指南，技術指南和配方如果您尚未熟悉技術指南實作，應先閱讀。 技術指南基本概念 提供實作自訂技術指南的詳細教學簡介，並實作 Stacks AWS OpsWorks 的技術指南涵蓋如何實作 Stacks AWS OpsWorks 執行個體技術指南的一些詳細資訊。

主題

- 透過覆寫屬性自訂 AWS OpsWorks Stacks 組態
- 使用自訂範本擴展 AWS OpsWorks Stacks 組態檔案
- 擴充 Layer
- 建立自訂 Tomcat 伺服器 Layer
- 堆疊組態及部署屬性

透過覆寫屬性自訂 AWS OpsWorks Stacks 組態

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 對於 Windows 堆疊和 Chef 12 Linux 堆疊， AWS OpsWorks Stacks 會針對內建配方和自訂配方使用個別的 Chef 執行。這表示您無法使用本節討論的技術覆寫 Windows 堆疊和 Chef 12 Linux 堆疊的內建屬性。

配方和範本取決於執行個體的各種 Chef 屬性，或堆疊限定的資訊 (例如 layer 組態或應用程式伺服器設定)。這些屬性有幾個來源：

- 自訂 JSON – 您可以在建立、更新或複製堆疊或部署應用程式時選擇性地指定自訂 JSON 屬性。
- 堆疊組態屬性 –AWS OpsWorks 堆疊會定義這些屬性以保留堆疊組態資訊，包括您透過主控台設定指定的資訊。
- 部署屬性 – AWS OpsWorks 會定義部署事件的部署相關屬性。
- 技術指南屬性 – 內建和自訂技術指南通常包含一或多個屬性檔案，其中包含代表技術指南特定值的屬性，例如應用程式伺服器組態設定。
- Chef–Chef 的 Ohai 工具定義了代表各種系統組態設定的屬性，例如 CPU 類型和已安裝的記憶體。

如需堆疊組態、部署屬性和內建技術指南屬性的完整清單，請參閱堆疊組態及部署屬性：Linux 和內建技術指南屬性。如需 Ohai 屬性的詳細資訊，請參閱 Ohai。

當部署或設定等生命週期事件發生時，或是您執行像是 或 等`execute_recipes`堆疊命令`update_packages`時， AWS OpsWorks Stacks 會執行下列作業：

- 將對應的命令傳送到每個受影響之執行個體的代理程式。

  代理程式會執行適當的配方。例如，針對部署事件，代理程式會執行內建的部署配方，其後跟隨任何自訂部署配方。

- 將任何自訂 JSON 和部署屬性與堆疊組態屬性合併，然後在執行個體上安裝他們。

來自自訂 JSON、堆疊組態和部署屬性、技術指南屬性和 Ohai 屬性的屬性會合併成一個「節點物件」，該物件會將屬性值提供給配方。針對堆疊組態屬性，執行個體基本上是無狀態的，包含任何自訂 JSON。當您執行部署或堆疊命令時，關聯的配方會使用以命令下載的堆疊組態屬性。

主題

- 屬性優先順序
- 使用自訂 JSON 覆寫屬性
- 使用自訂技術指南屬性覆寫 AWS OpsWorks Stacks 屬性

屬性優先順序

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

若屬性的定義是唯一的，Chef 會直接將其併入節點物件。但是，任何屬性來源都可以定義任何屬性，因此相同的屬性可能會有值不同的多個定義。例如，內建的 apache2 技術指南定義了 node[:apache][:keepalive]，但您也可以在自訂 JSON 或自訂技術指南中定義該屬性。若屬性具有多個定義，他們會按照稍後說明的順序進行評估，並且節點物件會接收到具有最高優先順序的定義。

一個屬性定義如下：

```
node.type[:attribute][:sub_attribute][:...]=value
```

如果屬性有多個定義， 類型會決定哪個定義具有優先順序，且該定義會併入節點物件中。 AWS OpsWorks Stacks 會使用下列屬性類型：

- default – 這是最常見的類型，基本上表示「如果尚未定義屬性，請使用此值」。若所有屬性的定義皆為 default 類型，評估順序中的第一個定義便具有優先順序，其餘值則會遭到忽略。請注意，Stacks AWS OpsWorks 會將所有堆疊組態和部署屬性定義設定為 default類型。

- normal – 此類型的屬性會覆寫先前在評估順序中定義的任何 `default`或 `normal` 屬性。例如，若第一個屬性是來自內建的技術指南，並且具有 `default` 類型，而第二個則是具有 `normal` 類型的使用者定義屬性，則第二個定義便具有優先順序。
- set – 這是您在舊版技術指南中可能看到的已棄用類型。它已由具有相同優先順序的 `normal` 取代。

Chef 支援數種額外的屬性類型，包含優先順序高於任何其他屬性定義的 `automatic` 類型。由 Chef 的 Ohai 工具產生的屬性定義全部皆為 `automatic` 類型，因此他們基本上等同於唯讀。這通常不是問題，因為沒有理由覆寫它們，而且它們與 AWS OpsWorks Stacks 的屬性不同。不過，建議您小心地為自訂技術指南屬性命名，使其與 Ohai 屬性相異。如需詳細資訊，請參閱關於屬性。

> **Note**
>
> Ohai 工具為您可以從命令列執行的可執行檔。若要列出執行個體的 Ohai 屬性，請登入執行個體並在終端機視窗中執行 `ohai`。請注意，它會產生非常長的輸出。

以下為將各種屬性定義併入節點物件的步驟：

1. 將任何自訂堆疊組態屬性併入堆疊組態和部署屬性。

   您可為堆疊或特定部署設定自訂 JSON 屬性。他們會位於評估的第一優先順序，基本上等同於 `normal` 類型。若一或多個堆疊組態屬性也在自訂 JSON 中定義，則自訂 JSON 的值會具有優先順序。否則 AWS OpsWorks Stacks 會直接將自訂 JSON 屬性併入堆疊組態。
2. 將任何部署自訂 JSON 屬性併入堆疊組態和部署屬性。

   部署自訂 JSON 屬性也等同於 `normal` 類型，因此他們會優先於內建和自訂堆疊組態 JSON 及內建部署 JSON。
3. 將堆疊組態和部署屬性合併成執行個體的節點物件。
4. 將執行個體的內建技術指南屬性合併成節點物件。

   內建技術指南屬性全部皆為 `default` 類型。如果堆疊組態和部署屬性中也定義了一或多個內建技術指南屬性，通常是因為您使用自訂 JSON 定義這些屬性，則堆疊組態定義優先於內建技術指南定義。所有其他內建技術指南屬性也會直接併入節點物件。
5. 將執行個體的自訂技術指南屬性合併成節點物件。

自訂技術指南屬性通常不是 normal 就是 default 類型。唯一屬性會併入節點物件。如果也在步驟 1–3 中定義任何自訂技術指南屬性（通常是因為您使用自訂 JSON 定義它們），優先順序取決於自訂技術指南屬性的類型：

- 步驟 1-3 中定義的屬性優先於自訂技術指南default屬性。
- 自訂 c ookbook normal 屬性優先於步驟 1–3 的定義。

> **Important**
>
> 請勿使用自訂技術指南的 default 屬性覆寫堆疊組態或內建技術指南屬性。由於自訂技術指南屬性會在最後評估，default 屬性具有最低的優先順序，因此無法覆寫任何內容。

使用自訂 JSON 覆寫屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 由於 AWS OpsWorks Stacks 處理 Chef 的執行方式與 Linux 堆疊不同，因此您無法針對 Windows 堆疊使用本節中討論的技術。

覆寫 AWS OpsWorks Stacks 屬性的最簡單方法是在自訂 JSON 中定義它，其優先順序高於堆疊組態和部署屬性，以及內建和自訂技術指南default屬性。如需詳細資訊，請參閱屬性優先順序。

> **Important**
>
> 建議您小心覆寫堆疊組態和部署屬性。例如：覆寫 opsworks 命名空間中的屬性可能會影響內建配方。如需詳細資訊，請參閱堆疊組態及部署屬性。

您也可以使用自訂 JSON 定義唯一屬性，通常是將資料傳遞到您的自訂配方。屬性會直接併入節點物件，並且配方可以透過使用標準 Chef 節點語法參考他們。

如何指定自訂 JSON

若要使用自訂 JSON 覆寫屬性值，您必須先判斷屬性的完整屬性名稱。您接著會建立包含您希望覆寫之屬性的 JSON 物件，將其設為您喜好的值。為了方便起見，堆疊組態及部署屬性：Linux 和內建技術指南屬性文件常會使用堆疊組態、部署和內建技術指南屬性，包含其完整名稱。

物件的父系和子系關聯必須對應到適當的完整 Chef 節點。例如，假設您希望變更以下 Apache 屬性：

- 節點為 並且預設值為 keepalivetimeout 的 node[:apache][:keepalivetimeout]3 屬性。
- 節點為 logrotate 並且預設值為 的 schedulenode[:apache][:logrotate][:schedule]"daily" 屬性。

若要覆寫屬性並將值分別設為 5 和 "weekly"，您會使用下列自訂 JSON：

```
{
  "apache" : {
    "keepalivetimeout" : 5,
    "logrotate" : {
       "schedule" : "weekly"
    }
  }
}
```

何時指定自訂 JSON

您可以為下列任務指定自訂 JSON：

- 建立新的堆疊
- 更新堆疊
- 執行堆疊命令
- 複製堆疊
- 部署應用程式

對於每個任務， AWS OpsWorks Stacks 會將自訂 JSON 屬性與堆疊組態和部署屬性合併，並將其傳送至執行個體，以合併至節點物件。但是，請注意以下內容：

- 若您在建立、複製或更新堆疊時指定自訂 JSON，針對後續生命週期事件和堆疊命令，屬性會合併至堆疊組態和部署屬性。
- 若您為部署指定自訂 JSON，屬性只會針對對應的事件合併至堆疊組態和部署屬性。

  若您希望針對後續部署使用那些自訂屬性，您必須再次明確指定自訂 JSON。

請務必記得，屬性只有在由配方使用時才會影響執行個體。若您覆寫屬性值，但是沒有任何後續的配方參考該屬性，則變更便不具有效果。您必須確保在關聯配方執行前傳送自訂 JSON，或是重新執行適當的配方。

自訂 JSON 最佳實務

您可以使用自訂 JSON 來覆寫任何 AWS OpsWorks Stacks 屬性，但手動輸入資訊有些麻煩，而且不在任何類型的來源控制下。自訂 JSON 最適合用於以下目的：

- 當您希望覆寫少量的屬性，並且不需要使用自訂技術指南時。

  透過自訂 JSON，您可以免於僅為了覆寫幾個屬性而設定及維護技術指南儲存庫的額外負荷。
- 敏感性值，例如密碼或身分驗證金鑰。

  技術指南屬性存放於儲存庫中，因此任何敏感性資訊都會暴露在洩漏的風險中。相反的，請使用 dummy 值定義屬性，並使用自訂 JSON 設定真正的值。
- 值應不同。

  例如，建議的實務是使用獨立的開發和預備堆疊支援您的生產堆疊。假設這些堆疊支援接受付款的應用程式。若您使用自訂 JSON 指定付款端點，您可以為您的預備堆疊指定測試 URL。當您準備好將更新後的堆疊遷移至您的生產堆疊時，您可以使用相同的技術指南，並使用自訂 JSON 將付款端點設為生產 URL。
- 特定堆疊或部署命令專屬的值。

使用自訂技術指南屬性覆寫 AWS OpsWorks Stacks 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 對於 Windows 堆疊， AWS OpsWorks Stacks 會針對內建配方和自訂配方使用個別的 Chef 執行。這表示您無法使用本節討論的技術覆寫 Windows 堆疊的內建屬性。

自訂 JSON 是覆寫 AWS OpsWorks Stacks 堆疊組態和內建技術指南屬性的便利方式，但有一些限制。特別是您必須為每一次的使用手動輸入自訂 JSON，因此您可用來管理定義的強固方法。較佳的方法通常是使用自訂技術指南屬性檔案覆寫內建的屬性。這樣做可讓您將定義置放在來源控制之下。

使用自訂屬性檔案覆寫 AWS OpsWorks Stacks 定義的程序非常簡單。

覆寫 AWS OpsWorks Stacks 屬性定義

1. 設定技術指南儲存庫，如技術指南和配方中所述。
2. 使用與內建技術指南相同的名稱建立技術指南，其中包含您希望覆寫的屬性。例如，若要覆寫 Apache 屬性，技術指南的名稱應為 apache2。
3. 將一個 `attributes` 資料夾新增至技術指南，並在該資料夾中新增一個名為 `customize.rb` 的檔案。
4. 針對每一個內建技術指南中您希望覆寫的屬性，將屬性定義新增至檔案，並設為您喜好的值。屬性必須是 `normal` 類型或更高版本，並且與對應的 AWS OpsWorks Stacks 屬性具有完全相同的節點名稱。如需 AWS OpsWorks Stacks 屬性的詳細清單，包括節點名稱，請參閱 堆疊組態及部署屬性：Linux和 內建技術指南屬性。如需屬性和屬性檔案的詳細資訊，請參閱關於屬性檔案。

   > **Important**
   >
   > 您的屬性必須是`normal`類型才能覆寫 AWS OpsWorks Stacks 屬性；`default`類型沒有優先順序。例如，若您的 `customize.rb` 檔案包含 `default[:apache]`

`[:keepalivetimeout] = 5` 屬性定義，內建 `apache.rb` 屬性檔案中的對應屬性會先受到評估，並取得優先順序。如需詳細資訊，請參閱覆寫屬性。

5. 為每個內建技術指南重複步驟 2 – 4，其中包含您要覆寫的屬性。
6. 為您的堆疊啟用自訂技術指南，並提供 Stacks AWS OpsWorks 將技術指南下載至堆疊執行個體所需的資訊。如需詳細資訊，請參閱安裝自訂技術指南。

> **Note**
>
> 如需此程序的完整演練，請參閱覆寫內建屬性。

後續生命週期事件、部署命令和堆疊命令所使用的節點物件現在將包含您的屬性定義，而不是 Stacks AWS OpsWorks 值。

例如，若要覆寫`keepalivetimeout` 中討論的內建 Apache `logrotate schedule` 和 如何指定自訂 JSON 設定，請將 `apache2` 技術指南新增至您的儲存庫，然後將 `customize.rb` 檔案新增至技術指南的 `attributes` 資料夾，並帶有以下內容。

```
normal[:apache][:keepalivetimeout] = 5
normal[:apache][:logrotate][:schedule] = 'weekly'
```

> **Important**
>
> 您不應透過修改相關聯的內建屬性檔案複本來覆寫 AWS OpsWorks Stacks 屬性。假設您將 `apache.rb` 複製到您的 `apache2/attributes` 資料夾並修改其部分設定，您基本上便已覆寫內建檔案中的每個屬性。配方會使用您複本中的屬性定義，並忽略內建檔案。若 AWS OpsWorks Stacks 稍後修改了內建屬性檔案，配方將無法存取變更，除非您手動更新您的複本。
>
> 為避免這種情況，所有內建的技術指南都包含一個空白的 `customize.rb` 屬性檔案，為所有透過 `include_attribute` 指示詞之模組的必要項目。透過覆寫您 `customize.rb` 複本中的屬性，您只會影響那些特定的屬性。配方會從內建屬性檔案取得任何其他的屬性值，並自動取得任何您未覆寫之屬性的目前值。
>
> 這種方法可協助您將您技術指南儲存庫中的屬性維持在較小的數目，減少您的維護額外負荷，並使未來的升級更容易管理。

使用自訂範本擴展 AWS OpsWorks Stacks 組態檔案

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 由於 AWS OpsWorks Stacks 處理 Chef 的執行方式與 Linux 堆疊不同，因此您無法針對 Windows 堆疊使用本節中討論的技術。

AWS OpsWorks Stacks 使用 範本來建立組態檔案等檔案，這些檔案通常取決於許多設定的屬性。如果您使用自訂 JSON 或自訂技術指南屬性來覆寫 AWS OpsWorks Stacks 定義，您偏好的設定會併入組態檔案中，以取代 AWS OpsWorks Stacks 設定。不過， AWS OpsWorks Stacks 不一定會為每個可能的組態設定指定屬性；它接受某些設定的預設值，並直接在範本中硬式編碼其他設定。如果沒有對應的 Stacks 屬性，則無法使用自訂 JSON AWS OpsWorks 或自訂技術指南屬性來指定偏好的設定。

您可以藉由建立自訂範本，來擴展組態檔案以包含額外組態設定。然後，您可以將所需的任何組態設定或其他內容新增至該檔案，並覆寫任何硬式編碼的設定。如需範本的詳細資訊，請參閱範本。

> **Note**
>
> 您可以覆寫任何內建範本，但 opsworks-agent.monitrc.erb「除外」。

建立自訂範本

1. 使用與內建技術指南相同的結構和目錄名稱來建立技術指南。接著，在適當的目錄中，使用您想要自訂之內建範本的相同結構和目錄名稱，來建立範本檔案。例如，若要使用自訂範本以擴展 Apache `httpd.conf` 組態檔案，您必須在儲存庫中實作 `apache2` 技術指南，且您的範本檔案必須為 `apache2/templates/default/apache.conf.erb`。使用完全相同的名稱可讓 AWS OpsWorks Stacks 識別自訂範本，並使用它，而不是內建範本。

   最簡單的方法是從內建技術指南的 GitHub 儲存庫，將內建的範本檔案複製到您的技術指南，並視需要修改。

> **Important**
>
> 除了您想要自訂的範本檔案以外，請不要從內建的技術指南複製任何檔案。其他類型的技術指南檔案 (例如配方) 複本會建立重複的 Chef 資源，並可能造成錯誤。

技術指南也可以包含自訂屬性、配方和相關的檔案，但它們的檔案名稱不應與內建的檔案名稱重複。

2. 自訂範本檔案以產生符合您要求的組態檔案。您可以新增更多設定、刪除現有的設定、取代硬式編碼的屬性等。
3. 如果您尚未執行，請編輯堆疊設定以啟用自訂技術指南，並指定您的技術指南儲存庫。如需詳細資訊，請參閱安裝自訂技術指南。

> **Note**
>
> 如需此程序的完整演練，請參閱覆寫內建範本。

您不需要實作任何配方或將配方新增至 layer 組態，即可覆寫範本。 AWS OpsWorks Stacks 一律會執行內建配方。當您執行配方以建立組態檔案時，它會自動使用您的自訂範本，而不是使用內建的範本。

> **Note**
>
> 如果 AWS OpsWorks Stacks 對內建範本進行任何變更，您的自訂範本可能會變得不同步且無法正常運作。例如，假設您的範本參考相依檔案，且檔案名稱變更。 AWS OpsWorks Stacks 不會經常進行此類變更，當範本確實變更時，它會列出變更，並讓您選擇升級至新版本。您應該監控 AWS OpsWorks Stacks 儲存庫的變更，並視需要手動更新範本。

擴充 Layer

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

針對透過修改 AWS OpsWorks Stacks 屬性或自訂範本處理的項目以外的項目，您有時需要自訂內建 layer。例如，假設您需要建立符號連結、設定檔案或資料夾模式、安裝額外的套件，以此類推。您必須擴充自訂 layer，以提供高於最少功能的功能。在該情況下，您需要實作一或多個具有配方的自訂技術指南來處理自訂任務。本主題提供如何使用配方來擴充 layer 的一些範例。

如果您是第一次使用 Chef，建議您先閱讀技術指南 101，該教學介紹如何實作技術指南以執行各種常見任務的基本概念。如需如何實作自訂 layer 的詳細範例，請參閱建立自訂 Tomcat 伺服器 Layer。

主題

- 使用配方執行指令碼
- 使用 Chef 部署勾點
- 在 Linux 執行個體上執行 Cron 任務
- 在 Linux 執行個體上安裝和設定套件

使用配方執行指令碼

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

如果您的現有指令碼執行所需的自訂任務，則擴充 layer 的最簡單方式實作通常是實作簡單配方來執行指令碼。您接著可以將配方指派給適當的生命週期事件 (一般是安裝或部署)，或使用 `execute_recipes` 堆疊命令手動執行配方。

下列範例會在 Linux 執行個體上執行 Shell 指令碼，但您可以將相同的方式用於其他類型的指令碼 (包括 Windows PowerShell 指令碼)。

```
cookbook_file "/tmp/lib-installer.sh" do
  source "lib-installer.sh"
  mode 0755
end

execute "install my lib" do
  command "sh /tmp/lib-installer.sh"
```

```
end
```

cookbook_file 資源代表存放在技術指南 files 目錄之子目錄中的檔案，並將檔案傳輸至執行個體上的指定位置。此範例會將 Shell 指令碼 lib-installer.sh 傳輸至執行個體的 /tmp 目錄，並將檔案的模式設定為 0755。如需詳細資訊，請參閱 cookbook_file。

execute 資源代表命令 (如 Shell 命令)。此範例執行 lib-installer.sh。如需詳細資訊，請參閱 execute。

您也可以將指令碼併入配方來執行指令碼。下列範例會執行 bash 指令碼，但 Chef 也支援 Csh、Perl、Python 和 Ruby。

```
script "install_something" do
  interpreter "bash"
  user "root"
  cwd "/tmp"
  code <<-EOH
    #insert bash script
  EOH
end
```

script 資源代表指令碼。此範例指定 bash 解譯器、將使用者設定為 "root"，並將工作目錄設定為 /tmp。它接著會執行 code 區塊中的 bash 指令碼，可視需要包括多行。如需詳細資訊，請參閱 script。

如需如何使用配方執行指令碼的詳細資訊，請參閱範例 7：執行命令和指令碼。如需如何在 Windows 執行個體上執行 PowerShell 指令碼的範例，請參閱執行 Windows PowerShell 指令碼。

使用 Chef 部署勾點

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

部署的自訂方式是實作自訂配方來執行所需任務，並將它指派給適當 layer 的部署事件。替代且有時更簡單的方法，特別是當您不需要實作技術指南用於其他用途時，就是使用 Chef 部署勾點來執行自訂程

式碼。此外，自訂部署配方會在內建配方已執行部署之後執行。部署勾點可讓您在部署期間互動，例如，從儲存庫簽出應用程式碼之後，但在重新啟動 Apache 之前。

Chef 會以四個階段部署應用程式：

- 結帳 – 從儲存庫下載檔案
- Migrate – 視需要執行遷移
- Symlink – 建立符號連結
- 重新啟動 - 重新啟動應用程式

Chef 部署勾點提供一種簡單的方法，選擇性地在每個階段完成之後執行使用者所提供的 Ruby 應用程式來自訂部署。若要使用部署勾點，請實作一或多個 Ruby 應用程式，並將它們放入您應用程式的 `/deploy` 目錄中 (如果您的應用程式沒有 `/deploy` 目錄，則請在 `APP_ROOT` 層級建立該目錄)。應用程式必須具有下列其中一個名稱，以決定其何時執行。

- `before_migrate.rb` 是在 Checkout (簽出) 階段完成之後但在 Migrate (遷移) 之前執行。
- `before_symlink.rb` 是在 Migrate (遷移) 階段完成之後但在 Symlink (符號連結) 之前執行。
- `before_restart.rb` 是在 Symlink (符號連結) 階段完成之後但在 Restart (重新啟動) 之前執行。
- `after_restart.rb` 是在 Restart (重新啟動) 階段完成之後執行。

Chef 部署勾點使用標準節點語法，即可存取節點物件，就像配方一樣。部署勾點也可以存取您所指定之任何應用程式環境變數的值。不過，您必須使用 `new_resource.environment["VARIABLE_NAME"]` 存取變數的值，而不是 `ENV["VARIABLE_NAME"]`。

在 Linux 執行個體上執行 Cron 任務

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Linux Cron 任務會指示 Cron 協助程式依指定的排程執行一或多個命令。例如，假設您的堆疊支援 PHP 電子商務應用程式。您可以設定 cron 任務，讓伺服器在每週的指定時間將銷售報告傳送給您。如需 cron 的詳細資訊，請參閱 Wikipedia 上的 cron。如需如何直接在 Linux 電腦或執行個體上執行 cron

任務的詳細資訊，請參閱 Indiana University 知識庫網站上的 What are cron and crontab, and how do I use them? (什麼是 cron 和 crontab 以及如何使用它們？)。

雖然您可以使用 SSH 連線 cron 任務並編輯其 crontab 項目以在個別 Linux 執行個體上手動設定它們，但是 AWS OpsWorks Stacks 的主要優點是您可以指示它跨整 layer 的執行個體執行任務。下列程序說明如何在 PHP App Server layer 的執行個體上設定cron任務，但您可以將相同的方法與任何 layer 搭配使用。

若要在 layer 執行個體上設定 **cron** 任務

1. 實作配方具有已設定任務之 cron 資源的技術指南。此範例假設配方命名為 cronjob.rb；稍後會說明實作詳細資訊。如需技術指南和配方的詳細資訊，請參閱技術指南和配方。
2. 在您的堆疊上安裝技術指南。如需詳細資訊，請參閱安裝自訂技術指南。
3. 讓 AWS OpsWorks Stacks 透過將配方指派給下列生命週期事件，在 layer 的執行個體上自動執行配方。如需詳細資訊，請參閱自動執行配方。

   - 設定 – 指派給cronjob.rb此事件會指示 AWS OpsWorks Stacks 在所有新執行個體上執行配方。
   - 部署 – 指派給cronjob.rb此事件會指示 AWS OpsWorks Stacks 在您將應用程式部署或重新部署至 layer 時，在所有線上執行個體上執行配方。

   您也可以使用 Execute Recipes 堆疊命令手動在線上執行個體上執行配方。如需詳細資訊，請參閱執行堆疊命令。

以下是 cronjob.rb 範例，可設定 Cron 任務一週一次執行使用者實作的 PHP 應用程式，以從伺服器收集銷售資料，並透過郵件傳送報告。如需如何使用 cron 資源的更多範例，請參閱 cron。

```
cron "job_name" do
  hour "1"
  minute "10"
  weekday "6"
  command "cd /srv/www/myapp/current && php .lib/mailing.php"
end
```

cron 是代表 cron 任務的 Chef 資源。當 AWS OpsWorks Stacks 在執行個體上執行配方時，相關聯的提供者會處理設定任務的詳細資訊。

- *job_name* 是 cron 任務的使用者定義名稱，例如 weekly report。
- hour/minute/weekday 指定何時應該執行命令。此範例會在每週六的上午 1:10 執行命令。
- command 指定要執行的命令。

此範例執行兩個命令。第一個導覽至 /srv/www/myapp/current 目錄。第二個執行使用者實作的 mailing.php 應用程式，以收集銷售資料並傳送報告。

> **Note**
>
> bundle 命令預設不會與 cron 任務搭配運作。原因是 AWS OpsWorks Stacks 在 /usr/local/bin目錄中安裝 Bundler。若要搭配使用 bundle 與 cron 任務，您必須將路徑 /usr/local/bin 明確地新增至 Cron 任務。此外，因為 $PATH 環境變數可能不會在 cron 任務中擴展，所以最佳實務是將任何必要的路徑資訊明確地新增至任務，而不依賴擴展 $PATH 變數。下列範例顯示兩種在 cron 任務中使用 bundle 的方式。
>
> ```
> cron "my first task" do
>   path "/usr/local/bin"
>   minute "*/10"
>   command "cd /srv/www/myapp/current && bundle exec my_command"
> end
> ```
>
> ```
> cron_env = {"PATH" => "/usr/local/bin"}
> cron "my second task" do
>   environment cron_env
>   minute "*/10"
>   command "cd /srv/www/myapp/current && /usr/local/bin/bundle exec my_command"
> end
> ```

如果您的堆疊有多個應用程式伺服器，cronjob.rb則指派給 PHP App Server layer 的生命週期事件可能不是理想的方法。例如，配方會在 layer 的所有執行個體上執行，因此您會收到多份報告。較佳的方式是使用自訂 layer，確保只有一部伺服器傳送報告。

只在 layer 的其中一個執行個體上執行配方

1. 建立自訂 layer (例如稱為 PHPAdmin)，並將 `cronjob.rb` 指派給其安裝和部署事件。自訂 layer 不一定需要執行很多操作。在此情況下，PHPAdmin 只會在其執行個體上執行一個自訂配方。
2. 將其中一個 PHP App Server 執行個體指派給 AdminLayer。如果執行個體屬於多個 layer， AWS OpsWorks Stacks 會執行每個 layer 的內建和自訂配方。

由於只有一個執行個體屬於 PHP App Server 和 PHPAdmin layer，因此 `cronjob.rb`只會在該執行個體上執行，而且您只會收到一個報告。

在 Linux 執行個體上安裝和設定套件

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

內建 layer 只支援特定套件。如需詳細資訊，請參閱層。您可以安裝其他套件 (例如 Redis 伺服器)，方法是實作自訂配方來處理相關的安裝、組態和部署任務。在某些情況下，最佳方式是擴充內建 layer，讓它在其執行個體上安裝套件以及 layer 的標準套件。例如，如果您的堆疊支援 PHP 應用程式，而且您想要包含 Redis 伺服器，則除了 PHP 應用程式伺服器之外，您還可以擴展 PHP App Server layer 在 layer 的執行個體上安裝和設定 Redis 伺服器。

套件安裝配方一般需要執行任務，如下所示：

- 建立一或多個目錄，並設定其模式。
- 從範本建立組態檔案。
- 在執行個體上執行安裝程式來安裝套件。
- 啟動一或多個服務。

如需如何安裝 Tomcat 伺服器的範例，請參閱建立自訂 Tomcat 伺服器 Layer。本主題說明如何設定自訂 Redis layer，但您可以使用更多相同的程式碼，以在內建 layer 上安裝和設定 Redis。如需如何安裝其他套件的範例，請參閱內建技術指南，網址為 https：//https://github.com/aws/opsworks-cookbooks。

建立自訂 Tomcat 伺服器 Layer

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 本主題說明如何實作 Linux 堆疊的自訂 layer。不過，基本原則和部分程式碼經改寫後也能用於實作 Windows 堆疊的自訂 layer，特別是應用程式部署一節的內容。

在 Stacks AWS OpsWorks 執行個體上使用非標準套件的最簡單方法是擴展現有的 layer。但是，此方法會在 layer 的執行個體上同時安裝及執行標準和非標準套件，而這並非是您所想要的。有一種要求較高也更強大的方式，便是實作自訂 layer，讓您幾乎可以完全控制 layer 的執行個體，包含下列項目：

- 要安裝的套件
- 每個套件的設定方式
- 從儲存庫將應用程式部署到執行個體的方式

無論您是使用主控台還是 API，您建立和管理自訂 layer 的方式與其他 layer 幾乎一模一樣，如自訂 Layer所述。但是，自訂 layer 的內建配方會執行一些非常基本的任務，例如安裝 Ganglia 用戶端以將指標報告給 Ganglia 主機。若要使自訂 layer 執行個體不僅具有最低限度的功能，您必須使用 Chef 配方和相關檔案實作一或多個自訂技術指南，以處理安裝及設定套件、部署應用程式等任務。但您不一定需要從頭開始實作所有東西。例如，若您在其中一個標準儲存庫中存放應用程式，您可以使用內建的部署配方處理在 layer 的執行個體上安裝應用程式的大部分工作。

> **Note**
>
> 如果您是第一次使用 Chef，建議您先閱讀技術指南 101，該教學介紹如何實作技術指南以執行各種常見任務的基本概念。

下列演練說明如何實作支援 Tomcat 應用程式伺服器的自訂 layer。layer 是以名為 Tomcat 的自訂技術指南為基礎，其中包含處理套件安裝、部署等的配方。演練包含摘錄自 Tomcat 技術指南的摘要。您可以從其 GitHub 儲存庫下載完整的技術指南。若您不熟悉 Opscode Chef，建議您先閱讀技術指南和配方。

> **Note**
>
> AWS OpsWorks Stacks 包含用於生產用途的完整功能 Java App Server layer。Tomcat 技術指南的目的是示範如何實作自訂 layer，使其支援 Tomcat 限制版本 (不包含諸如 SSL 等功能)。如需完整實作的範例，請參閱內建的 opsworks_java 技術指南。

Tomcat 技術指南支援執行個體具有下列特性的自訂 layer：

- 他們支援使用 Apache 前端的 Tomcat Java 應用程式伺服器。
- Tomcat 設定為允許應用程式使用 JDBC `DataSource` 物件連線到單獨的 MySQL 執行個體，做為後端資料存放區。

此專案的技術指南涉及幾項主要元件：

- 屬性檔案包含各種配方會使用的組態設定。
- 安裝配方會指派給 layer 的安裝生命週期事件。他們會在執行個體啟動並執行像是安裝套件和建立組態檔案等任務後執行。
- 設定配方會指派給 layer 的設定生命週期事件。它們會在堆疊的組態變更後執行，主要是在執行個體上線或離線時，並處理任何必要的組態變更。
- 部署配方會指派給 layer 的部署生命週期事件。他們會在安裝配方之後，以及您手動部署應用程式，然後在 layer 的執行個體上安裝程式碼和相關檔案時執行，並會處理相關任務 (例如重新啟動服務)。

最後一節 說明如何建立堆疊，其中包含以 Tomcat 技術指南為基礎的自訂 layer建立堆疊和執行應用程式，以及如何部署和執行簡單的 JSP 應用程式，以顯示在屬於個別 MySQL layer 的執行個體上執行之 MySQL 資料庫的資料。

> **Note**
>
> Tomcat 技術指南配方取決於一些 AWS OpsWorks Stacks 內建配方。為使每個配方的來源更為清晰，本主題會使用 Chef cookbookname::recipename 慣例識別配方。

主題

- 屬性檔案
- 安裝配方
- 設定配方
- 部署配方
- 建立堆疊和執行應用程式

屬性檔案

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

在查看配方前，先檢查 Tomcat 技術指南的屬性檔案 (包含各種配方使用的組態設定) 會非常有用。屬性並非必要項目。您可以直接以硬式編碼的方式在您的配方或範本中撰寫這些值。不過，如果您使用屬性定義組態設定，您可以使用 AWS OpsWorks Stacks 主控台或 API 透過定義自訂 JSON 屬性來修改值，這比每次變更設定時重寫配方或範本程式碼更簡單、更靈活。舉例來說，這種方法可讓您針對多個堆疊使用相同的技術指南，但為每一個堆疊分別設定不同的 Tomcat 伺服器。如需屬性和其覆寫方式的詳細資訊，請參閱覆寫屬性。

下列範例顯示完整的屬性檔案 (`default.rb`)，位於 Tomcat 技術指南的 `attributes` 目錄。

```
default['tomcat']['base_version'] = 6
default['tomcat']['port'] = 8080
default['tomcat']['secure_port'] = 8443
default['tomcat']['ajp_port'] = 8009
default['tomcat']['shutdown_port'] = 8005
default['tomcat']['uri_encoding'] = 'UTF-8'
default['tomcat']['unpack_wars'] = true
default['tomcat']['auto_deploy'] = true
case node[:platform]
when 'centos', 'redhat', 'fedora', 'amazon'
  default['tomcat']['java_opts'] = ''
when 'debian', 'ubuntu'
```

```
  default['tomcat']['java_opts'] = '-Djava.awt.headless=true -Xmx128m -XX:
+UseConcMarkSweepGC'
end
default['tomcat']['catalina_base_dir'] = "/etc/tomcat#{node['tomcat']['base_version']}"
default['tomcat']['webapps_base_dir'] = "/var/lib/tomcat#{node['tomcat']
['base_version']}/webapps"
default['tomcat']['lib_dir'] = "/usr/share/tomcat#{node['tomcat']['base_version']}/lib"
default['tomcat']['java_dir'] = '/usr/share/java'
default['tomcat']['mysql_connector_jar'] = 'mysql-connector-java.jar'
default['tomcat']['apache_tomcat_bind_mod'] = 'proxy_http' # or: 'proxy_ajp'
default['tomcat']['apache_tomcat_bind_config'] = 'tomcat_bind.conf'
default['tomcat']['apache_tomcat_bind_path'] = '/tc/'
default['tomcat']['webapps_dir_entries_to_delete'] = %w(config log public tmp)
case node[:platform]
when 'centos', 'redhat', 'fedora', 'amazon'
  default['tomcat']['user'] = 'tomcat'
  default['tomcat']['group'] = 'tomcat'
  default['tomcat']['system_env_dir'] = '/etc/sysconfig'
when 'debian', 'ubuntu'
  default['tomcat']['user'] = "tomcat#{node['tomcat']['base_version']}"
  default['tomcat']['group'] = "tomcat#{node['tomcat']['base_version']}"
  default['tomcat']['system_env_dir'] = '/etc/default'
end
```

設定本身會在稍後的相關章節中討論。下列注意事項一概適用：

- 所有的節點定義都是 `default` 類型，讓您可以使用自訂 JSON 屬性予以覆寫。
- 檔案使用 `case` 陳述式，根據執行個體的作業系統，有條件的設定一些屬性值。

  `platform` 節點是由 Chef 的 Ohai 工具產生的，代表執行個體的作業系統。

安裝配方

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

安裝配方會指派給 layer 的安裝生命週期事件，並會在執行個體啟動後執行。他們會執行像是安裝套件、建立組態檔案和啟動服務等任務。安裝配方完成執行後， AWS OpsWorks Stacks 會執行部署配方，將任何應用程式部署到新的執行個體。

主題

- tomcat::setup
- tomcat::install
- tomcat::service
- tomcat::container_config
- tomcat::apache_tomcat_bind

tomcat::setup

`tomcat::setup` 配方用於指派給 layer 的安裝生命週期事件。

```
include_recipe 'tomcat::install'
include_recipe 'tomcat::service'

service 'tomcat' do
  action :enable
end

# for EBS-backed instances we rely on autofs
bash '(re-)start autofs earlier' do
  user 'root'
  code <<-EOC
    service autofs restart
  EOC
  notifies :restart, resources(:service => 'tomcat')
end

include_recipe 'tomcat::container_config'
include_recipe 'apache2'
include_recipe 'tomcat::apache_tomcat_bind'
```

`tomcat::setup` 配方基本上就是一個中繼配方。它包含一組處理大部分安裝和設定 Tomcat 之詳細資訊和相關套件的依存配方。`tomcat::setup` 的第一個部分會執行下列配方。下列配方會在稍後進行討論：

- tomcat::install 配方會安裝 Tomcat 伺服器套件。
- tomcat::service 配方會設定 Tomcat 服務。

`tomcat::setup` 的中間部分會啟用並執行 Tomcat 服務：

- Chef service 資源會在開機時啟用 Tomcat 服務。
- Chef bash 資源會執行 Bash 指令碼來啟動 autofs 協助程式，這是 Amazon EBS 後端執行個體的必要項目。資源接著會通知 `service` 資源重新啟動 Tomcat 服務。

  如需詳細資訊，請參閱 autofs (Amazon Linux 適用) 或 Autofs (Ubuntu 適用)。

`tomcat::setup` 的最後部分會建立組態檔案，安裝及設定前端 Apache 伺服器：

- tomcat::container_config 配方會建立組態檔案。
- `apache2` 配方 (的速記`apache2::default`) 是安裝和設定 Apache 伺服器的 AWS OpsWorks Stacks 內建配方。
- tomcat::apache_tomcat_bind 配方會設定 Apache 伺服器，做為 Tomcat 伺服器的前端。

> **Note**
>
> 您通常可以透過使用內建配方執行一部分的必要任務，省下時間和精力。此配方使用內建的 `apache2::default` 配方安裝 Apache，而非從零開始實作。如需如何使用內建配方的另一個範例，請參閱部署配方。

下列章節會更詳細的說明 Tomcat 技術指南的安裝配方。如需 `apache2` 配方的詳細資訊，請參閱 opsworks-cookbooks/apache2。

tomcat::install

`tomcat::install` 配方會安裝 Tomcat 伺服器、OpenJDK 和處理 MySQL 伺服器連線的 Java 連接器程式庫。

```
tomcat_pkgs = value_for_platform(
  ['debian', 'ubuntu'] => {
    'default' => ["tomcat#{node['tomcat']['base_version']}", 'libtcnative-1',
 'libmysql-java']
```

```
  },
  ['centos', 'redhat', 'fedora', 'amazon'] => {
    'default' => ["tomcat#{node['tomcat']['base_version']}", 'tomcat-native', 'mysql-
connector-java']
  },
  'default' => ["tomcat#{node['tomcat']['base_version']}"]
)

tomcat_pkgs.each do |pkg|
  package pkg do
    action :install
  end
end

link ::File.join(node['tomcat']['lib_dir'], node['tomcat']['mysql_connector_jar']) do
  to ::File.join(node['tomcat']['java_dir'], node['tomcat']['mysql_connector_jar'])
  action :create
end

# remove the ROOT webapp, if it got installed by default
include_recipe 'tomcat::remove_root_webapp'
```

配方會執行下列任務：

1. 根據執行個體的作業系統建立要安裝的套件清單。
2. 安裝清單中的每個套件。

   Chef 套件資源使用適用於 Amazon Linux 和 `apt-get` Ubuntu 的適當提供者`yum`來處理安裝。套件提供者會將 OpenJDK 做為 Tomcat 的依存安裝，但 MySQL 連接器程式庫必須明確安裝。
3. 使用 Chef link 資源，在 Tomcat 伺服器的 lib 目錄中建立連結到 JDK 中 MySQL 連接器程式庫的 symlink。

   使用預設屬性值，Tomcat lib 目錄為 `/usr/share/tomcat6/lib`，MySQL 連接器程式庫 (`mysql-connector-java.jar`) 則位於 `/usr/share/java/`。

`tomcat::remove_root_webapp` 配方會移除 ROOT web 應用程式 (預設為 `/var/lib/tomcat6/webapps/ROOT`)，避免某些安全問題。

```
ruby_block 'remove the ROOT webapp' do
  block do
```

```
    ::FileUtils.rm_rf(::File.join(node['tomcat']['webapps_base_dir'], 'ROOT'), :secure
 => true)
  end
  only_if { ::File.exists?(::File.join(node['tomcat']['webapps_base_dir'], 'ROOT'))
 && !::File.symlink?(::File.join(node['tomcat']['webapps_base_dir'], 'ROOT')) }
end
```

only_if 陳述式會確保只有在檔案存在時，配方才會移除他們。

> **Note**
>
> Tomcat 的版本以 ['tomcat']['base_version'] 屬性指定，其在屬性檔案中已設為 6。若要安裝 Tomcat 7，您可以使用自訂 JSON 屬性覆寫屬性。您只需編輯您的堆疊設定並在 Custom Chef JSON (自訂 Chef JSON) 方塊中輸入下列 JSON，或是將其新增至任何現有的自訂 JSON：
>
> ```
> {
>   'tomcat' : {
>     'base_version' : 7
>   }
> }
> ```
>
> 自訂 JSON 屬性會覆寫預設屬性，並將 Tomcat 的版本設為 7。如需覆寫屬性的詳細資訊，請參閱覆寫屬性。

tomcat::service

tomcat::service 配方會建立 Tomcat 服務定義。

```
service 'tomcat' do
  service_name "tomcat#{node['tomcat']['base_version']}"

  case node[:platform]
  when 'centos', 'redhat', 'fedora', 'amazon'
    supports :restart => true, :reload => true, :status => true
  when 'debian', 'ubuntu'
    supports :restart => true, :reload => false, :status => true
  end
```

```
  action :nothing
end
```

配方會使用 Chef service 資源指定 Tomcat 服務名稱 (預設為 tomcat6)，並設定 `supports` 屬性，以定義 Chef 管理服務重新啟動、重新載入和在不同作業系統上狀態命令的方式。

- `true` 表示 Chef 可使用 init 指令碼或其他服務提供者執行命令。
- `false` 表示 Chef 必須嘗試使用其他方式執行命令。

請注意，`action` 已設為 `:nothing`。對於每個生命週期事件， AWS OpsWorks Stacks 會啟動 Chef 執行，以執行適當的配方集。Tomcat 技術指南遵循讓配方建立服務定義，但不重新啟動服務的常見模式。Chef 執行中的其他配方會處理重新啟動，其方式通常為藉由在用來建立組態檔案的 `notifies` 資源中包含 `template` 命令。通知為重新啟動服務的一種便利方式，因為他們只會在組態變更時才會執行此作業。此外，當 Chef 執行具有多個服務的重新啟動通知時，Chef 最多只會重新啟動服務一次。這種做法可避免在嘗試重新啟動未完全運作的服務時可能發生的問題。此為 Tomcat 錯誤的常見來源。

必須為任何使用重新啟動通知的 Chef run 定義 Tomcat 服務。因此，`tomcat::service` 會包含在數個配方中，確保為每一個 Chef run 定義服務。即使 Chef 執行包含多個 `tomcat::service` 的執行個體，也不會有任何損失，因為 Chef 會確保每個配方只會在每次執行時執行一次，無論其包含多少次在內。

tomcat::container_config

`tomcat::container_config` 配方會從技術指南範本檔案建立組態檔案。

```
include_recipe 'tomcat::service'

template 'tomcat environment configuration' do
  path ::File.join(node['tomcat']['system_env_dir'], "tomcat#{node['tomcat']
['base_version']}")
  source 'tomcat_env_config.erb'
  owner 'root'
  group 'root'
  mode 0644
  backup false
  notifies :restart, resources(:service => 'tomcat')
end

template 'tomcat server configuration' do
```

```
  path ::File.join(node['tomcat']['catalina_base_dir'], 'server.xml')
  source 'server.xml.erb'
  owner 'root'
  group 'root'
  mode 0644
  backup false
  notifies :restart, resources(:service => 'tomcat')
end
```

配方首先會呼叫 tomcat::service，其將於必要時定義服務。大量的配方皆由兩個 template 資源組成，這兩個資源會各自從其中一個技術指南範本檔案建立組態檔案、設定檔案屬性，並通知 Chef 重新啟動服務。

Tomcat 環境資訊檔案

第一個 template 資源會使用 tomcat_env_config.erb 範本檔案建立 Tomcat 環境資訊檔案，用於設定像是 JAVA_HOME 等環境變數。預設檔案名稱為 template 資源的引數。tomcat::container_config 會使用 path 屬性覆寫預設值及命名組態檔 /etc/sysconfig/tomcat6 (Amazon Linux) 或 /etc/default/tomcat6 (Ubuntu)。template 資源也會指定檔案的擁有者、群組和模式設定，並指示 Chef 不要建立備份檔案。

若您查看來源碼，通常會有三種版本的 tomcat_env_config.erb，每一種都位於 templates 目錄中的不同子目錄內。ubuntu 和 amazon 目錄包含其各自作業系統的範本。default 資料夾則包含僅有一行註解的 dummy 範本，僅會在您嘗試在使用不支援作業系統的執行個體上執行此配方時使用。tomcat::container_config 配方不需要指定要使用何種 tomcat_env_config.erb。Chef 會自動根據檔案精確性中描述的規則，為執行個體的作業系統挑選適當的目錄。

此範例的 tomcat_env_config.erb 檔案大部分由註解組成。若要設定其他環境變數，只需要取消註解適當的行，並提供您喜好的值即可。

> **Note**
>
> 任何可能變更的組態設定都應定義為屬性，而非在範本中硬式編碼。如此一來，您便不需要重新撰寫範本以變更設定；您可以直接覆寫屬性。

Amazon Linux 範本只會設定一個環境變數，如下列摘要所示。

```
...
```

```
# Use JAVA_OPTS to set java.library.path for libtcnative.so
#JAVA_OPTS="-Djava.library.path=/usr/lib"

JAVA_OPTS="${JAVA_OPTS} <%= node['tomcat']['java_opts'] %>"

# What user should run tomcat
#TOMCAT_USER="tomcat"
...
```

JAVA_OPTS 可用來指定 Java 選項 (例如程式庫路徑)。若使用預設屬性值，範本便不會為 Amazon Linux 設定任何 Java 選項。您可以藉由覆寫 ['tomcat']['java_opts'] 屬性 (例如：使用自訂 JSON 屬性)，來設定您自己的 Java 選項。如需範例，請參閱「建立堆疊」。

Ubuntu 範本會設定數個環境變數，如下列範本摘要所示。

```
# Run Tomcat as this user ID. Not setting this or leaving it blank will use the
# default of tomcat<%= node['tomcat']['base_version'] %>.
TOMCAT<%= node['tomcat']['base_version'] %>_USER=tomcat<%= node['tomcat']
['base_version'] %>
...
# Run Tomcat as this group ID. Not setting this or leaving it blank will use
# the default of tomcat<%= node['tomcat']['base_version'] %>.
TOMCAT<%= node['tomcat']['base_version'] %>_GROUP=tomcat<%= node['tomcat']
['base_version'] %>
...
JAVA_OPTS="<%= node['tomcat']['java_opts'] %>"

<% if node['tomcat']['base_version'].to_i < 7 -%>
# Unset LC_ALL to prevent user environment executing the init script from
# influencing servlet behavior.  See Debian bug #645221
unset LC_ALL
<% end -%>
```

若使用預設屬性值，範本便會設定 Ubuntu 環境變數，如下所示：

- 代表 Tomcat 使用者和群組的 TOMCAT6_USER 和 TOMCAT6_GROUP 都已設為 tomcat6。

  若您將 ['tomcat']['base_version'] 設為 tomcat7，變數名稱會解析為 TOMCAT7_USER 和 TOMCAT7_GROUP，並且都會設為 tomcat7。
- JAVA_OPTS 已設定為 -Djava.awt.headless=true -Xmx128m -XX:+UseConcMarkSweepGC：

- 將 `-Djava.awt.headless` 設為 `true` 會向圖形引擎通知執行個體沒有標題並且也沒有主控台，表示特定圖形應用程式的錯誤行為。
- `-Xmx128m` 可確保 JVM 具有足夠的記憶體資源，此範例為 128MB。
- `-XX:+UseConcMarkSweepGC` 指定同時標記整理記憶體回收，協助限制記憶體回收造成的暫停。

  如需詳細資訊，請參閱 Concurrent Mark Sweep Collector Enhancements。
- 若 Tomcat 的版本低於 7，範本會取消設定處理 Ubuntu 錯誤的 `LC_ALL`。

> **Note**
>
> 若使用預設屬性，部分環境變數便會設為其預設值。但是，明確將環境變數設為屬性表示您可以定義自訂 JSON 屬性覆寫預設屬性，並提供自訂的值。如需覆寫屬性的詳細資訊，請參閱覆寫屬性。

如需完整的範本檔案，請參閱來源碼。

Server.xml 組態檔案

第二個 `template` 資源會使用 `server.xml.erb` 建立 `system.xml` 組態檔，設定 servlet/JSP 容器。`server.xml.erb` 沒有包含任何作業系統限定設定，因此它會位於 `template` 目錄的 `default` 子目錄中。

範本使用者使用標準設定，但它也可以為 Tomcat 6 或 Tomcat 7 建立 `system.xml` 檔案。例如，下列來自範本的伺服器區段的程式碼會為指定的版本適當設定接聽程式。

```
<% if node['tomcat']['base_version'].to_i > 6 -%>
  <!-- Security listener. Documentation at /docs/config/listeners.html
  <Listener className="org.apache.catalina.security.SecurityListener" />
  -->
<% end -%>
  <!--APR library loader. Documentation at /docs/apr.html -->
  <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />
  <!--Initialize Jasper prior to webapps are loaded. Documentation at /docs/jasper-
howto.html -->
  <Listener className="org.apache.catalina.core.JasperListener" />
  <!-- Prevent memory leaks due to use of particular java/javax APIs-->
```

```
  <Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />
<% if node['tomcat']['base_version'].to_i < 7 -%>
  <!-- JMX Support for the Tomcat server. Documentation at /docs/non-existent.html -->
  <Listener className="org.apache.catalina.mbeans.ServerLifecycleListener" />
<% end -%>
  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
<% if node['tomcat']['base_version'].to_i > 6 -%>
  <Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />
<% end -%>
```

範本使用屬性，而非硬式編碼設定，讓您可以藉由定義自訂 JSON 屬性，輕易變更設定。例如：

```
<Connector port="<%= node['tomcat']['port'] %>" protocol="HTTP/1.1"
           connectionTimeout="20000"
           URIEncoding="<%= node['tomcat']['uri_encoding'] %>"
           redirectPort="<%= node['tomcat']['secure_port'] %>" />
```

如需詳細資訊，請參閱來源碼。

tomcat::apache_tomcat_bind

`tomcat::apache_tomcat_bind` 配方會啟用 Apache 伺服器做為 Tomcat 的前端，接受傳入的請求，並將他們轉遞至 Tomcat 並將回應傳回用戶端。此範例使用 mod_proxy 做為 Apache 代理/閘道。

```
execute 'enable mod_proxy for apache-tomcat binding' do
  command '/usr/sbin/a2enmod proxy'
  not_if do
    ::File.symlink?(::File.join(node['apache']['dir'], 'mods-enabled', 'proxy.load'))
 || node['tomcat']['apache_tomcat_bind_mod'] !~ /\Aproxy/
  end
end

execute 'enable module for apache-tomcat binding' do
  command "/usr/sbin/a2enmod #{node['tomcat']['apache_tomcat_bind_mod']}"
  not_if {::File.symlink?(::File.join(node['apache']['dir'], 'mods-enabled',
 "#{node['tomcat']['apache_tomcat_bind_mod']}.load"))}
end

include_recipe 'apache2::service'

template 'tomcat thru apache binding' do
```

```
  path ::File.join(node['apache']['dir'], 'conf.d', node['tomcat']
['apache_tomcat_bind_config'])
  source 'apache_tomcat_bind.conf.erb'
  owner 'root'
  group 'root'
  mode 0644
  backup false
  notifies :restart, resources(:service => 'apache2')
end
```

若要啟用 mod_proxy，您必須啟用 proxy 模組及通訊協定式模組。您有兩個通訊協定模組選項：

- HTTP: proxy_http
- Apache JServ 通訊協定 (ASP)：proxy_ajp

  AJP 是一種內部 Tomcat 通訊協定。

兩種配方的 execute 資源都會執行 a2enmod 命令，透過建立必要的 symlink 啟用指定的模組：

- 第一個 execute 資源會啟用 proxy 模組。
- 第二個 execute 資源會啟用通訊協定模組，根據預設設為 proxy_http。

  若您要使用 AJP，您可以定義自訂 JSON 覆寫 apache_tomcat_bind_mod 屬性，將其設為 proxy_ajp。

apache2::service 配方是定義 Apache AWS OpsWorks 服務的 Stacks 內建配方。如需詳細資訊，請參閱 Stacks GitHub AWS OpsWorks 儲存庫中的配方。

template 資源使用 apache_tomcat_bind.conf.erb 建立根據預設名為 tomcat_bind.conf 的組態檔案。它會將檔案置放在 ['apache']['dir']/.conf.d 目錄中。['apache']['dir'] 屬性會在內建的 apache2 屬性檔案中定義，並根據預設設為 /etc/httpd (Amazon Linux) 或 /etc/apache2 (Ubuntu)。若 template 資源建立或變更組態檔案，notifies 命令會排程 Apache 服務重新啟動。

```
<% if node['tomcat']['apache_tomcat_bind_mod'] == 'proxy_ajp' -%>
ProxyPass <%= node['tomcat']['apache_tomcat_bind_path'] %> ajp://localhost:<%=
 node['tomcat']['ajp_port'] %>/
ProxyPassReverse <%= node['tomcat']['apache_tomcat_bind_path'] %> ajp://localhost:<%=
 node['tomcat']['ajp_port'] %>/
```

```
<% else %>
ProxyPass <%= node['tomcat']['apache_tomcat_bind_path'] %> http://localhost:<%=
 node['tomcat']['port'] %>/
ProxyPassReverse <%= node['tomcat']['apache_tomcat_bind_path'] %> http://localhost:<%=
 node['tomcat']['port'] %>/
<% end -%>
```

範本使用 ProxyPass 和 ProxyPassReverse 指示詞設定用來在 Apache 和 Tomcat 間傳遞流量的連接埠。因為兩個伺服器都位於相同的執行個體上，他們可以使用 localhost URL，並且根據預設都會設為 `http://localhost:8080`。

設定配方

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

設定配方會指派給 layer 的設定生命週期事件，該事件會在執行個體進入或離開線上狀態時，在所有堆疊的執行個體上發生。您可使用設定配方調整執行個體的組態，以適當回應變更。當您實作設定配方時，請記得堆疊組態變更可能會涉及與這一 layer 無關的執行個體。配方必須要能適當地進行回應，在某些案例中，可能不會進行任何行為。

tomcat::configure

`tomcat::configure` 配方用於 layer 的設定生命週期事件。

```
include_recipe 'tomcat::context'
# Optional: Trigger a Tomcat restart in case of a configure event, if relevant
# settings in custom JSON have changed (e.g. java_opts/JAVA_OPTS):
#include_recipe 'tomcat::container_config'
```

`tomcat::configure` 配方基本上就是執行兩個依存配方的中繼配方。

1. `tomcat::context` 配方會建立 web 應用程式內容組態檔案。

   此檔案會設定應用程式用來與 MySQL 執行個體通訊的 JDBC 資源，如下一節所討論。執行此配方以回應設定事件，可讓 layer 在資料庫 layer 變更時更新 web 應用程式內容組態檔案。

2. `tomcat::container_config` 安裝配方會再次執行，以擷取任何容器組態中的變更。

`include` 的 `tomcat::container_config` 在此範例中已標示為註解。若您希望使用自訂的 JSON 修改 Tomcat 設定，您可以移除註解。設定生命週期事件接著會執行 `tomcat::container_config` 更新 Tomcat 的相關組態檔案 (如 tomcat::container_config 所述) 並重新啟動 Tomcat 服務。

tomcat::context

Tomcat 技術指南可透過使用 J2EE DataSource 物件，讓應用程式存取可能在分離的執行個體上執行的 MySQL 資料庫伺服器。透過 Tomcat，您可以藉由建立和安裝每個應用程式的 web 應用程式內容組態檔案，來啟用連線。此檔案定義應用程式和應用程式用來與資料庫通訊之 JDBC 資源間的關聯。如需詳細資訊，請參閱內容容器。

`tomcat::context` 配方的主要用途是建立此組態檔案。

```
include_recipe 'tomcat::service'

node[:deploy].each do |application, deploy|
  context_name = deploy[:document_root].blank? ? application : deploy[:document_root]

  template "context file for #{application} (context name: #{context_name})" do
    path ::File.join(node['tomcat']['catalina_base_dir'], 'Catalina', 'localhost',
 "#{context_name}.xml")
    source 'webapp_context.xml.erb'
    owner node['tomcat']['user']
    group node['tomcat']['group']
    mode 0640
    backup false
    only_if { node['datasources'][context_name] }
    variables(:resource_name => node['datasources'][context_name], :webapp_name =>
 application)
    notifies :restart, resources(:service => 'tomcat')
  end
end
```

除了 Tomcat 技術指南屬性之外，此配方還使用 AWS OpsWorks Stacks 隨設定事件一起安裝的堆疊組態和部署屬性。Stacks AWS OpsWorks 服務會將屬性新增至每個執行個體的節點物件，其中包含配方通常會使用資料包或搜尋取得的資訊，並在每個執行個體上安裝屬性。屬性包含堆疊組態、部署應用程式和任何使用者希望包含之自訂資料的詳細資訊。配方可透過使用標準 Chef 節點語法，從堆疊組態和

部署屬性取得資料。如需詳細資訊，請參閱堆疊組態及部署屬性。透過 Chef 11.10 堆疊，您也可以使用 Chef 搜尋取得堆疊組態和部署資料。如需詳細資訊，請參閱使用 Chef 搜尋。

deploy 屬性是指[:deploy]命名空間，其中包含透過主控台或 API 定義的部署相關屬性，或由 Stacks AWS OpsWorks 服務產生的屬性。deploy 屬性包含每個部署應用程式的屬性，以應用程式的短名命名。每個應用程式屬性都包含一組描述應用程式的屬性，例如文件根 ([:deploy][:*appname*][:document_root])。

context 配方首先會透過呼叫 tomcat::service，確認已為此 Chef 執行定義服務。它接著會定義 context_name 變數，代表組態檔案的名稱 (排除 .xml 副檔名)。若您使用預設文件根，context_name 會設為應用程式的短名。否則，它會設為指定的文件根。建立堆疊和執行應用程式中討論的範例會將文件根設為 "ROOT"，因此內容為 ROOT，組態檔案則名為 ROOT.xml。

大量的配方都會穿過每個應用程式的部署應用程式清單，使用 webapp_context.xml.erb 範本建立內容組態檔案。範例只會部署一個應用程式，但 deploy 屬性的定義仍會要求您將其視為應用程式清單處理。

webapp_context.xml.erb 範本並非作業系統限定，因此它會位於 templates 目錄的 default 子目錄中。

配方會建立組態檔案如下：

- 當使用預設屬性值時，組態檔案名稱會設為 *context_name*.xml，並安裝在 /etc/tomcat6/Catalina/localhost/ 目錄中。

  堆疊組態屬性的 ['datasources'] 節點包含一或多個屬性，每一種都會將一個內容名稱映射到關聯應用程式用來與資料庫通訊的 JDBC 資料來源。節點和其內容都會在您建立堆疊時，使用自訂 JSON 定義，如稍後的建立堆疊和執行應用程式所述。範例具有將 ROOT 內容名稱與名為 jdbc/mydb 的 JDBC 資源建立關聯的單一屬性。
- 使用預設屬性值，檔案的使用者和群組都會設為由 Tomcat 套件定義的值：tomcat (Amazon Linux) 或 tomcat6 (Ubuntu)。
- template 資源只會在 ['datasources'] 節點存在且包含 context_name 屬性值建立組態檔案。
- template 資源會定義兩個變數：resource_name 和 webapp_name。

  resource_name 已設為與 context_name 關聯的資源名稱，webapp_name 則已設為應用程式的短名。
- 範本資源會重新啟動 Tomcat 服務，載入及啟用變更。

webapp_context.xml.erb 範本由包含具有自己屬性組之 Context 元素的 Resource 元素組成。

Resource 屬性會描述內容組態的特性：

- name – JDBC 資源名稱，設定為 中定義的resource_name值tomcat::context。

  例如，資源名稱已設為 jdbc/mydb。
- 驗證和類型 – 這些是 JDBC DataSource連線的標準設定。
- maxActive、maxIdle 和 maxWait – 作用中和閒置連線的數量上限，以及傳回連線的等待時間上限。
- username， and password – 從deploy屬性取得的資料庫使用者名稱和密碼。
- driverClassName – JDBC 驅動程式的類別名稱，設定為 MySQL 驅動程式。
- url – 連線 URL。

  前綴取決於資料庫。其應設為 jdbc:mysql (MySQL)、jdbc:postgresql (Postgres) 和 jdbc:sqlserver (SQL Server)。範例將 URL 設為 jdbc:mysql://*host_IP_Address*:3306:simplejsp，其中 *simplejsp* 為應用程式的短名。
- factory – MySQL 資料庫需要的 DataSource factory。

如需此組態檔案的詳細資訊，請參閱 Tomcat 維基的 Using DataSources 主題。

部署配方

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

部署配方會指派給 layer 的部署生命週期事件。每當您部署應用程式時，它通常會發生在堆疊的所有執行個體上，但您可以選擇將事件限制為僅指定執行個體。安裝配方完成後， AWS OpsWorks 堆疊也會在新執行個體上執行部署配方。部署配方的主要用途是從儲存庫將程式碼和相關檔案部署到應用程式伺服器 layer 的執行個體。但是，您通常也需要在其他 layer 執行部署配方。這可讓那些 layer 的執行個體進行像是更新其組態以適應新部署之應用程式等行為。當您實作部署配方時，請記得部署事件不一定表示應用程式正部署到執行個體。它可能只是一項針對應用程式正在部署到堆疊中其他執行個體的通知，讓執行個體進行任何必要的更新。配方必須要能適當地進行回應，其可能不會進行任何行為。

AWS OpsWorks Stacks 會自動將標準應用程式類型的應用程式部署至對應的內建應用程式伺服器層。若要將應用程式部署到自訂 layer，您必須實作自訂部署配方，從儲存庫將應用程式檔案下載到執行個體上的適當位置。但是，您通常可以藉由使用內建的部署技術指南限制您必須撰寫的程式碼數量，處理一部分的部署。例如，若您將您的檔案存放在其中一個支援的儲存庫，內建技術指南可以處理從儲存庫將檔案下載到 layer 的執行個體的詳細資訊。

`tomcat::deploy` 技術指南用於指派給部署生命週期事件。

```
include_recipe 'deploy'

node[:deploy].each do |application, deploy|
  opsworks_deploy_dir do
    user deploy[:user]
    group deploy[:group]
    path deploy[:deploy_to]
  end

  opsworks_deploy do
    deploy_data deploy
    app application
  end
...
```

`tomcat::deploy` 配方針對部署中非應用程式限定的一部分使用內建的部署技術指南。`deploy` 配方 (為內建 `deploy::default` 配方的速記) 是一種內建的配方，可根據 `deploy` 屬性的資料處理設定使用者、群組等詳細資訊。

配方使用兩個內建 Chef 定義 (`opsworks_deploy_dir` 和 `opworks_deploy`) 安裝應用程式。

`opsworks_deploy_dir` 定義會根據應用程式部署 JSON 的資料設定目錄結構。定義基本上是封裝資源定義的便利方式，位於技術指南的 `definitions` 目錄中。配方可像資源一樣使用定義，但定義本身不具有關聯的提供者，只有包含在定義內的資源。您可以在配方中定義變數，該變數會傳遞到基礎資源定義。`tomcat::deploy` 配方會根據部署 JSON 的資料設定 `user`、`group` 和 `path` 變數。他們會傳遞到定義的 directory 資源，該資源會管理目錄。

> **Note**
>
> 您部署應用程式的使用者和群組由 `[:opsworks][:deploy_user][:user]` 和 `[:opsworks][:deploy_user][:group]` 屬性決定，這兩個屬性的定義位於內建部署技術指南的 `deploy.rb` 屬性檔案中。`[:opsworks][:deploy_user][:user]` 的預設值為

deploy。[:opsworks][:deploy_user][:group] 的預設值視執行個體的作業系統而定。

- 若是 Ubuntu 執行個體，預設群組為 www-data。
- 對於使用 Nginx 和 Unicorn 的 Rails App Server layer 成員的 Amazon Linux 執行個體，預設群組為 nginx。
- 若是其他所有 Amazon Linux 執行個體，預設群組為 apache。

您可以透過使用自訂 JSON 或自訂屬性檔案覆寫適當的屬性，來變更設定。如需詳細資訊，請參閱覆寫屬性。

其他定義 (opsworks_deploy) 會根據 deploy 屬性的資料，處理檢查來自儲存庫的應用程式程式碼和相關檔案，以及將他們部署到執行個體的詳細資訊。您可以針對任何應用程式類型使用此定義。部署詳細資訊 (例如目錄名稱) 會在主控台中或透過 API 指定，並置放於 deploy 屬性中。但是，opsworks_deploy 僅適用於四種支援的儲存庫類型：Git、Subversion、S3 和 HTTP。若您希望使用不同的儲存庫類型，您必須自行實作程式碼。

您會在 Tomcat webapps 目錄中安裝應用程式的檔案。典型的做法是將檔案直接複製到 webapps。不過， AWS OpsWorks Stacks 部署的設計可在執行個體上保留最多五個版本的應用程式，因此您可以視需要轉返至較早版本。 AWS OpsWorks Stacks 因此會執行下列動作：

1. 將應用程式部署到名稱包含時間戳記的相異目錄，例如 /srv/www/my_1st_jsp/releases/20130731141527。
2. 建立連結到此唯一目錄，名為 current 的 symlink，例如 /srv/www/my_1st_jsp/current。
3. 若尚未存在，請從 webapps 目錄建立連結到步驟 2 中建立之 current symlink 的 symlink。

若您需要轉返至較早的版本，請修改 current symlink 指向包含適當時間戳記的相異目錄 (例如，變更 /srv/www/my_1st_jsp/current 的連結目標)。

tomcat::deploy 中間的區段會設定 symlink。

```
  ...
  current_dir = ::File.join(deploy[:deploy_to], 'current')
  webapp_dir = ::File.join(node['tomcat']['webapps_base_dir'],
deploy[:document_root].blank? ? application : deploy[:document_root])
```

```
  # opsworks_deploy creates some stub dirs, which are not needed for typical webapps
  ruby_block "remove unnecessary directory entries in #{current_dir}" do
    block do
      node['tomcat']['webapps_dir_entries_to_delete'].each do |dir_entry|
        ::FileUtils.rm_rf(::File.join(current_dir, dir_entry), :secure => true)
      end
    end
  end

  link webapp_dir do
    to current_dir
    action :create
  end
  ...
```

配方首先會建立兩個變數 (current_dir 和 webapp_dir) 分別代表 current 和 webapp。它接著會使用 link 資源將 webapp_dir 連結到 current_dir。Stacks AWS OpsWorks deploy::default配方會建立此範例不需要的一些 stub 目錄，因此摘錄的中間部分會將其移除。

tomcat::deploy 的最終部分會重新啟動 Tomcat 服務 (若必要的話)。

```
  ...
  include_recipe 'tomcat::service'

  execute 'trigger tomcat service restart' do
    command '/bin/true'
    not_if { node['tomcat']['auto_deploy'].to_s == 'true' }
    notifies :restart, resources(:service => 'tomcat')
  end
end

include_recipe 'tomcat::context'
```

配方首先會執行 tomcat::service，確保已為此 Chef 執行定義服務。它接著會使用 execute 資源通知服務重新啟動，但只有在 ['tomcat']['auto_deploy'] 設為 'true' 時才會這麼做。否則，Tomcat 會接聽其 webapps 目錄中的變更，使明確重新啟動 Tomcat 服務為非必要事項。

> **Note**
>
> `execute` 資源不會執行任何實際操作。`/bin/true` 僅是一個 dummy shell 指令碼，只會傳回成功代碼。在此使用它的目的只是為了方便產生重新啟動通知。如先前所述，使用通知可確保服務不會太頻繁的重新啟動。

最後，`tomcat::deploy` 會執行 `tomcat::context`，更新 web 應用程式內容組態檔案 (若您有變更後端資料庫的話)。

建立堆疊和執行應用程式

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節說明如何使用 Tomcat 技術指南實作基本堆疊設定，執行名為 SimpleJSP 的簡易 Java 伺服器頁面 (JSP) 應用程式。堆疊包含名為 TomCustom 的 Tomcat 型自訂層和 MySQL 層。SimpleJSP 會部署到 TomCustom，並顯示 MySQL 資料庫中的一些資訊。如果您尚未熟悉如何使用 AWS OpsWorks Stacks 的基本概念，您應該先閱讀 Chef 11 Linux 堆疊入門。

SimpleJSP 應用程式

SimpleJSP 應用程式會示範如何設定資料庫連線和從堆疊的 MySQL 資料庫擷取資料的基本操作。

```
<html>
  <head>
    <title>DB Access</title>
  </head>
  <body>
    <%@ page language="java" import="java.sql.*,javax.naming.*,javax.sql.*" %>
    <%
      StringBuffer output = new StringBuffer();
      DataSource ds = null;
      Connection con = null;
      Statement stmt = null;
      ResultSet rs = null;
```

```
      try {
        Context initCtx = new InitialContext();
        ds = (DataSource) initCtx.lookup("java:comp/env/jdbc/mydb");
        con = ds.getConnection();
        output.append("Databases found:<br>");
        stmt = con.createStatement();
        rs = stmt.executeQuery("show databases");
        while (rs.next()) {
          output.append(rs.getString(1));
          output.append("<br>");
        }
      }
      catch (Exception e) {
        output.append("Exception: ");
        output.append(e.getMessage());
        output.append("<br>");
      }
      finally {
        try {
          if (rs != null) {
            rs.close();
          }
          if (stmt != null) {
            stmt.close();
          }
          if (con != null) {
            con.close();
          }
        }
        catch (Exception e) {
          output.append("Exception (during close of connection): ");
          output.append(e.getMessage());
          output.append("<br>");
        }
      }
    %>
    <%= output.toString() %>
  </body>
</html>
```

SimpleJSP 使用 DataSource 物件與 MySQL 資料庫通訊。Tomcat 使用 Web 應用程式內容組態檔案中的資料建立和初始化 DataSource 物件，並將其繫結到邏輯名稱。它接著會使用 Java 命

名及目錄界面 (JNDI) 命名服務註冊邏輯名稱。為取得適當 DataSource 物件的執行個體，您會建立 InitialContext 物件，並將資源的邏輯名稱傳遞給物件的 lookup 方法，擷取適當的物件。SimpleJSP 範例的邏輯名稱 (java:comp/env/jdbc/mydb) 包含下列元件：

- 根命名空間 (java)，以冒號 (:) 與名稱的剩餘部分分開。
- 任何額外的命名空間，都會以斜線 (/) 分開。

  Tomcat 會自動將資源新增到 comp/env 命名空間。
- 資源名稱，定義於 web 應用程式內容組態檔案中，並以斜線與命名空間分隔。

  此範例的資源名稱為 jdbc/mydb。

為建立資料庫的連線，SimpleJSP 會執行下列作業：

1. 呼叫 DataSource 物件的 getConnection 方法，傳回 Connection 物件。
2. 呼叫 Connection 物件的 createStatement 方法，建立 Statement 物件，您會使用此物件與資料庫通訊。
3. 透過呼叫適當的 Statement 方法與資料庫通訊。

   SimpleJSP 會呼叫 executeQuery 執行 SHOW DATABASES 查詢，列出伺服器的資料庫。

executeQuery 方法會傳回 ResultSet 物件，其中包含查詢的結果。SimpleJSP 會從傳回的 ResultSet 物件取得資料庫名稱，然後串連它們以建立輸出字串。最後，範例會關閉 ResultSet、Statement 和 Connection 物件。如需 JSP 和 JDBC 的詳細資訊，請分別參考 JavaServer Pages Technology 和 JDBC Basics。

若要搭配堆疊使用 SimpleJSP，您必須將其置放在儲存庫中。您可以使用任何支援的儲存庫，但若要搭配下列章節中討論的範例堆疊使用 SimpleJSP，您必須將其置放在公有 S3 封存中。如需如何使用其他標準儲存庫的資訊，請參閱技術指南儲存庫。

將 SimpleJSP 置放在 S3 封存儲存庫中

1. 將範例程式碼複製到名為 simplejsp.jsp 的檔案，然後將檔案置放在名為 simplejsp 的目錄中。
2. 建立 simplejsp 目錄的 .zip 存檔。
3. 建立公有 Amazon S3 儲存貯體、上傳至simplejsp.zip儲存貯體，並將檔案設為公有。

   如需如何執行此任務的說明，請參閱 Amazon Simple Storage Service 入門。

建立堆疊

若要執行 SimpleJSP，您需要具有下列 layer 的堆疊。

- 支援後端 MySQL 資料庫的 MySQL layer。
- 使用 Tomcat 技術指南支援 Tomcat 伺服器執行個體的自訂 layer。

建立堆疊

1. 在 AWS OpsWorks Stacks 儀表板上，按一下新增堆疊以建立新的堆疊，然後按一下進階 **>>** 以顯示所有選項。設定堆疊如下。

   - 名稱 – 使用者定義的堆疊名稱；此範例使用 TomStack。
   - 使用自訂 Chef 技術指南 – 將切換設定為是，這會顯示一些額外的選項。
   - 儲存庫類型 – Git。
   - 儲存庫 URL–`git://github.com/amazonwebservices/opsworks-example-cookbooks.git`。
   - 自訂 Chef JSON – 新增下列 JSON：

   ```
   {
     "tomcat": {
       "base_version": 7,
       "java_opts": "-Djava.awt.headless=true -Xmx256m"
     },
     "datasources": {
       "ROOT": "jdbc/mydb"
     }
   }
   ```

   對於剩餘的選項，您可以接受預設值。

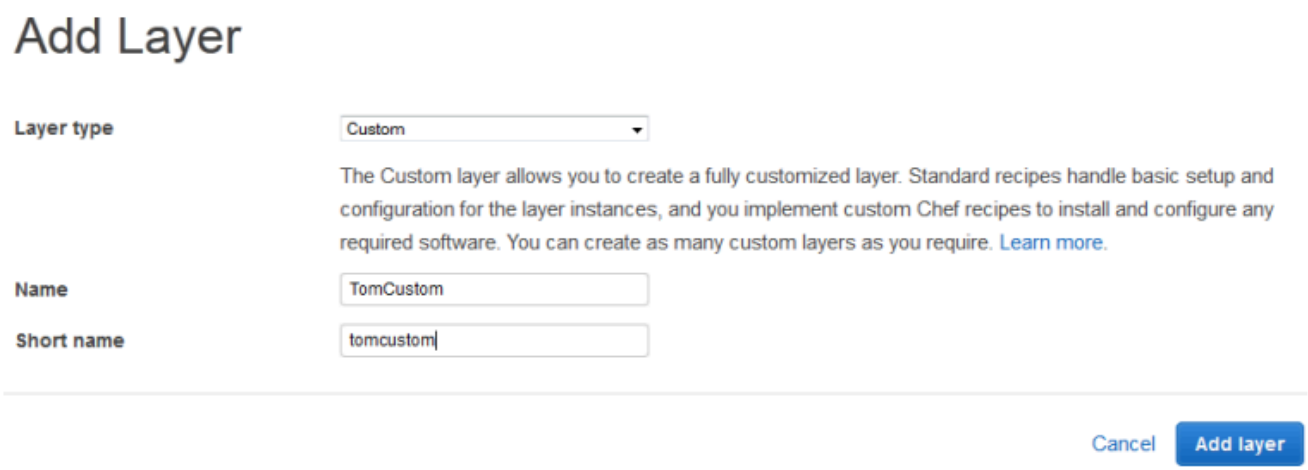
   自訂 JSON 會執行下列操作：

   - 覆寫 Tomcat 技術指南的 `['base_version']` 屬性，將 Tomcat 的版本設為 7。預設值為 6。
   - 覆寫 Tomcat 技術指南的 `['java_opts']` 屬性，指定執行個體沒有標題，並將 JVM 最大堆積大小設為 256MB。預設值不會為執行 Amazon Linux 的執行個體設定任何選項。

- 指定 `['datasources]` 屬性值，指派 JDBC 資源名稱 (jdbc/mydb) 給 web 應用程式內容名稱 (ROOT)，如 tomcat::context 中所討論。

  這個最後一個屬性沒有任何預設值。您必須使用自訂 JSON 設定它。

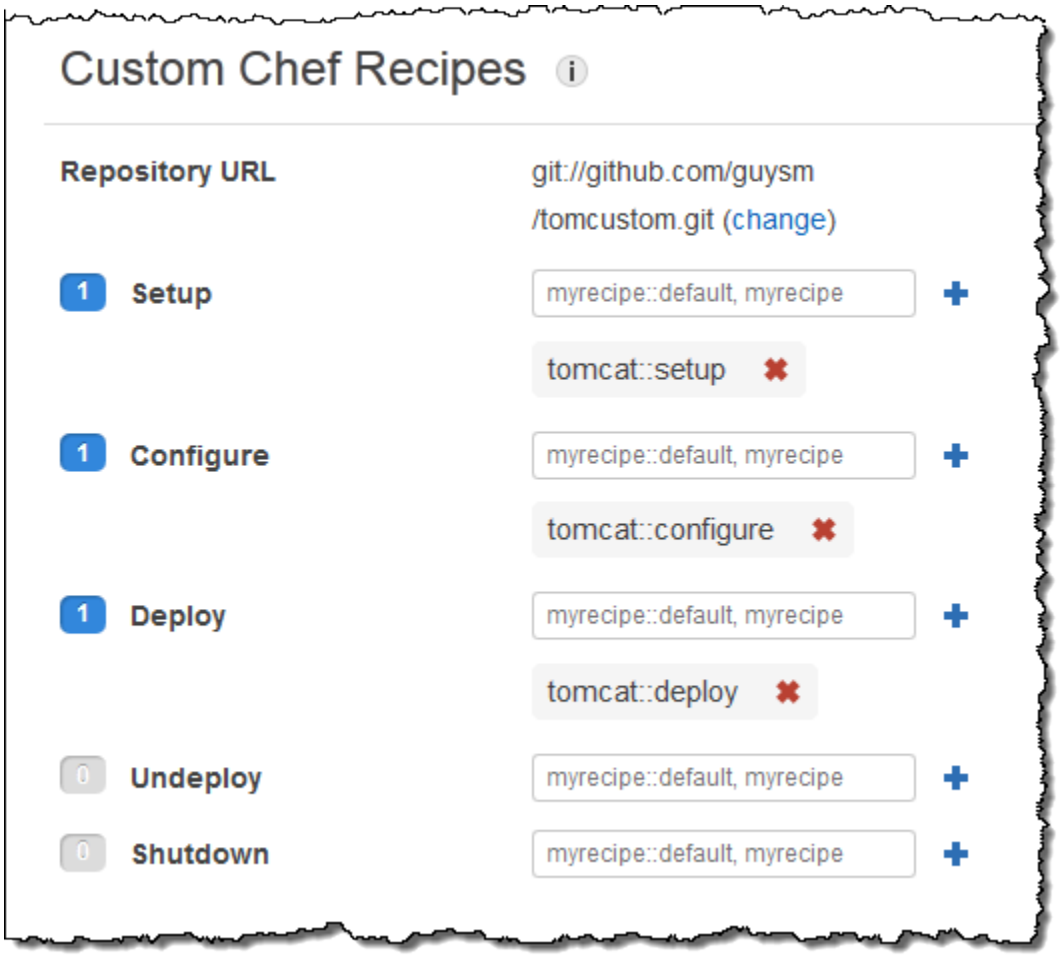
2. 按一下 Add a layer (新增 layer)。針對 Layer type (Layer 類型)，選取 MySQL。然後按一下 Add Layer (新增 Layer)。
3. 按一下導覽窗格中的 Instances (執行個體)，然後按一下 Add an instance (新增執行個體)。按一下 Add Instance (新增執行個體) 接受預設值。在執行個體的列上，按一下 start (啟動)。
4. 返回 Layers (Layer) 頁面然後按一下 + Layer (+Layer) 以新增 layer。針對 Layer type (Layer 類型)，按一下 Custom (自訂)。範例使用 **`TomCustom`** 和 **`tomcustom`** 分別做為 layer 的名稱和短名。



5. 在 Layers (Layer) 頁面上，針對自訂 layer，按一下 Recipes (配方) 然後按一下 Edit (編輯)。在 Custom Chef Recipes (自訂 Chef 配方) 下，將 Tomcat 技術指南配方指派給 layer 的生命週期事件，如下所示：

   - 針對 Setup (安裝)，輸入 **`tomcat::setup`** 然後按一下 +。
   - 針對 Configure (設定)，輸入 **`tomcat::configure`** 然後按一下 +。
   - 針對 Deploy (部署)，輸入 **`tomcat::deploy`** 然後按一下 +。然後按一下 Save (儲存)。

   .



6. 按一下導覽窗格中的 Apps (應用程式)，然後按一下 Add an app (新增應用程式)。指定下列選項，然後按一下 Add App (新增應用程式)：

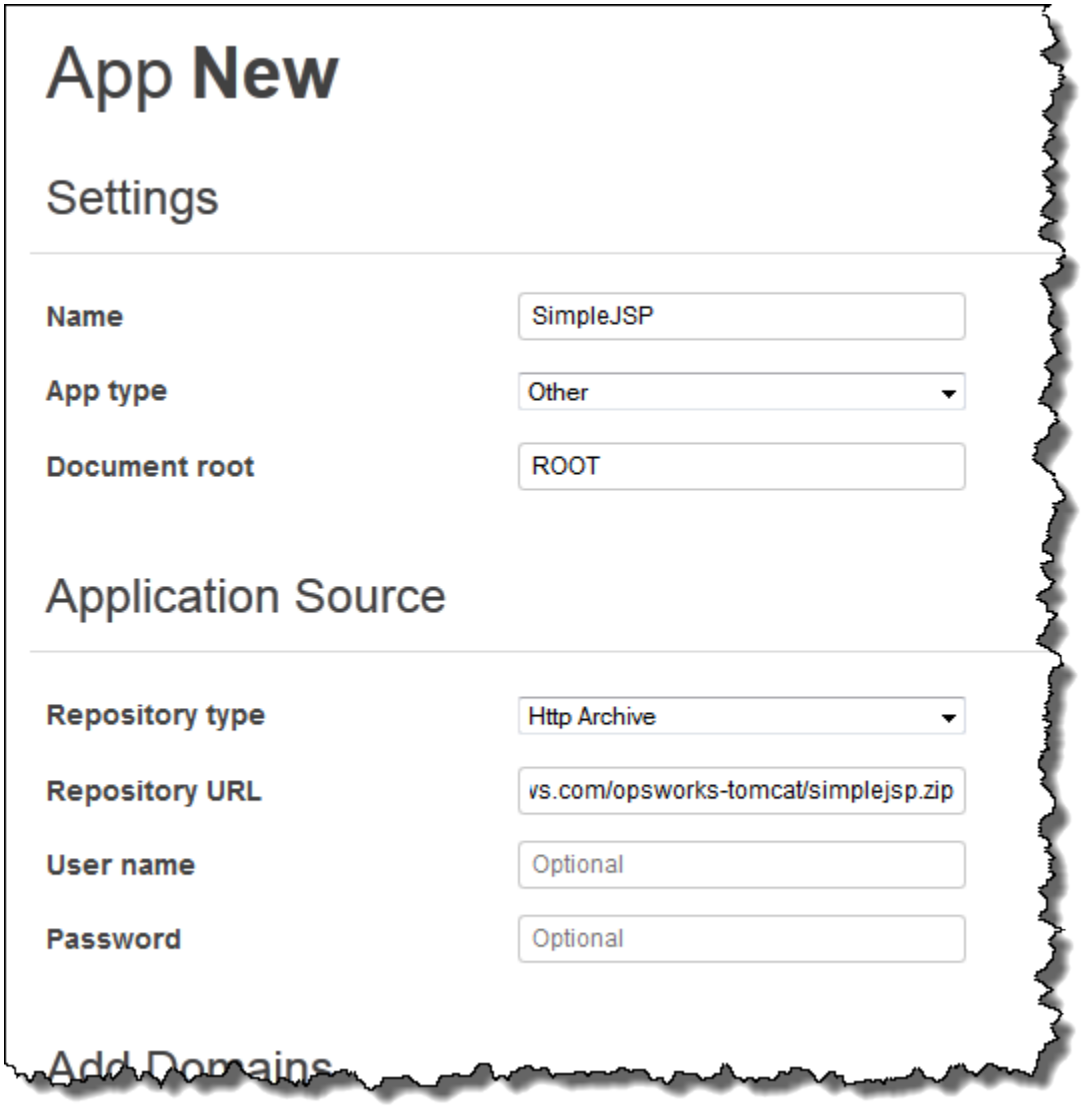
   - 名稱 – 應用程式的名稱；範例使用 SimpleJSP，而 Stacks AWS OpsWorks 產生的簡短名稱將是 simplejsp。
   - 應用程式類型 – 將此選項設定為其他。

AWS OpsWorks Stacks 會自動將標準應用程式類型部署至相關聯的伺服器執行個體。若您將 App type (應用程式類型) 設為「其他」，AWS OpsWorks Stacks 便只會執行部署配方，並讓它們處理部署。

- 文件根 – 將此選項設定為 **ROOT**。

  Document root (文件根) 的值指定內容名稱。
- 儲存庫類型 – 將此選項設定為 S3 Archive。
- 儲存庫 URL – 將此設定為您先前建立的應用程式 Amazon S3 URL。

其他選項請使用預設設定。



7. 使用執行個體頁面將執行個體新增至 TomCustom 層並啟動。安裝配方完成後， AWS OpsWorks Stacks 會自動在新執行個體上執行部署配方，因此啟動執行個體也會部署 SimpleJSP。
8. 當 TomCustom 執行個體上線時，在 Instances (執行個體) 頁面上按一下執行個體名稱以查看詳細資訊。複製公有 IP 地址。然後建構 URL 如下：http://*publicIP*/tc/*appname.jsp*。例如，此 URL 看起來會像這樣：**`http://50.218.191.172/tc/simplejsp.jsp`**。

   > **Note**
   >
   > 轉遞請求至 Tomcat 的 Apache URL 已設為預設 `['tomcat']['apache_tomcat_bind_path']` 屬性，`/tc/`。SimpleJSP 文件根已設為 `ROOT`，為解析至 `/` 的特殊值。URL 因此為 ".../tc/simplejsp.jsp"。

9. 將先前步驟中的 URL 於您的瀏覽器內貼上。請查看下列事項：

```
Databases found:
information_schema
simplejsp
test
```

   > **Note**
   >
   > 如果您的堆疊有 MySQL 執行個體， AWS OpsWorks Stacks 會自動為每個應用程式建立資料庫，以應用程式的簡短名稱命名。

堆疊組態及部署屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

當 AWS OpsWorks Stacks 在執行個體上執行命令時，例如，回應部署生命週期事件的部署命令，它會將一組屬性新增至描述堆疊目前組態的執行個體節點物件。對於部署事件和執行配方堆疊命令，AWS OpsWorks Stacks 會安裝部署屬性，以提供一些額外的部署資訊。如需節點物件的詳細資訊，請

參閱覆寫屬性。如需常用堆疊組態及部署屬性的清單 (包含完整的節點名稱)，請參閱堆疊組態及部署屬性：Linux 和內建技術指南屬性。

> **Note**
>
> 在 Linux 堆疊上，您可以透過使用代理程式 CLI 的 get_json 命令取得這些屬性的完整清單，其格式為 JSON 物件。

下列各節顯示與設定事件關聯的屬性，以及一個簡易堆疊的部署事件，由下列項目組成：

- 具有兩個執行個體的 PHP App Server layer
- 具有一個執行個體的 HAProxy layer

這些範例來自其中一個 PHP App Server 執行個體 php-app1。為了方便，屬性已格式化為 JSON 物件。物件的結構會映射到屬性的完整名稱。例如，`node[:opsworks][:ruby_version]` 屬性會在 JSON 表示中以下列方式呈現。

```
{
  "opsworks": {
    ...
    "ruby_version": "1.8.7",
    ...
  }
}
```

主題

- 設定屬性
- 部署屬性

設定屬性

下列 JSON 物件顯示設定事件的屬性，其會在執行個體上線或離線時，在堆疊中的每個執行個體上發生，屬性包含內建堆疊組態屬性和任何在事件之前為堆疊定義的自訂 JSON 屬性 (在此範例中則無)。它已針對長度進行編輯。如需各種屬性的詳細描述，請參閱堆疊組態及部署屬性：Linux 和內建技術指南屬性。

```
{
  "opsworks": {
    "layers": {
      "php-app": {
        "id": "4a2a56c8-f909-4b39-81f8-556536d20648",
        "instances": {
          "php-app2": {
            "elastic_ip": null,
            "region": "us-west-2",
            "booted_at": "2013-02-26T20:41:10+00:00",
            "ip": "192.0.2.0",
            "aws_instance_id": "i-34037f06",
            "availability_zone": "us-west-2a",
            "instance_type": "c1.medium",
            "private_dns_name": "ip-10-252-0-203.us-west-2.compute.internal",
            "private_ip": "10.252.0.203",
            "created_at": "2013-02-26T20:39:39+00:00",
            "status": "online",
            "backends": 8,
            "public_dns_name": "ec2-192-0-2-0.us-west-2.compute.amazonaws.com"
          },
          "php-app1": {
            ...
          }
        },
        "name": "PHP Application Server"
      },
      "lb": {
        "id": "15c86142-d836-4191-860f-f4d310440f14",
        "instances": {
          "lb1": {
           ...
          }
        },
        "name": "Load Balancer"
      }
    },
    "agent_version": "104",
    "applications": [

    ],
    "stack": {
```

```
      "name": "MyStack"
    },
    "ruby_version": "1.8.7",
    "sent_at": 1361911623,
    "ruby_stack": "ruby_enterprise",
    "instance": {
      "layers": [
        "php-app"
      ],
      "region": "us-west-2",
      "ip": "192.0.2.0",
      "id": "45ef378d-b87c-42be-a1b9-b67c48edafd4",
      "aws_instance_id": "i-32037f00",
      "availability_zone": "us-west-2a",
      "private_dns_name": "ip-10-252-84-253.us-west-2.compute.internal",
      "instance_type": "c1.medium",
      "hostname": "php-app1",
      "private_ip": "10.252.84.253",
      "backends": 8,
      "architecture": "i386",
      "public_dns_name": "ec2-192-0-2-0.us-west-2.compute.amazonaws.com"
    },
    "activity": "configure",
    "rails_stack": {
      "name": null
    },
    "deployment": null,
    "valid_client_activities": [
      "reboot",
      "stop",
      "setup",
      "configure",
      "update_dependencies",
      "install_dependencies",
      "update_custom_cookbooks",
      "execute_recipes"
    ]
  },
  "opsworks_custom_cookbooks": {
    "recipes": [

    ],
    "enabled": false
  },
```

```
  "recipes": [
    "opsworks_custom_cookbooks::load",
    "opsworks_ganglia::configure-client",
    "ssh_users",
    "agent_version",
    "mod_php5_apache2::php",
    "php::configure",
    "opsworks_stack_state_sync",
    "opsworks_custom_cookbooks::execute",
    "test_suite",
    "opsworks_cleanup"
  ],
  "opsworks_rubygems": {
    "version": "1.8.24"
  },
  "ssh_users": {
  },
  "opsworks_bundler": {
    "manage_package": null,
    "version": "1.0.10"
  },
  "deploy": {
  }
}
```

大多數的資訊都位於 opsworks 屬性下方，通常其名為命名空間。以下清單說明關鍵屬性：

- layers 屬性 – 一組屬性，每個屬性都會描述其中一個堆疊層的組態。

  這些 layer 會根據其短名識別，此範例中為 php-app 和 lb。如需其他 layer 之短名的詳細資訊，請參閱 AWS OpsWorks Stacks Layer 參考。

- instances 屬性 – 每個 layer 都有 instances元素，其中包含每個 layer 線上執行個體的屬性，以執行個體的簡短名稱命名。

  PHP App Server layer 有兩個執行個體 php-app1和 php-app2。HAProxy layer 有一個執行個體 lb1。

> **Note**
>
> instances 元素只包含那些在建立特定堆疊及部署屬性時處於線上狀態的執行個體。

- 執行個體屬性 – 每個執行個體屬性都包含一組屬性來描述執行個體，例如執行個體的私有 IP 地址和私有 DNS 名稱。為了簡化，此範例只顯示 php-app2 屬性的詳細資訊。其他的也包含相似的資訊。
- `applications` – 部署的應用程式清單，不在此範例中使用。
- `stack` – 堆疊名稱；在此範例中`MyStack`為 。
- `instance` – 已安裝這些屬性的執行個體；在此範例中`php-app1`為 。配方可使用此屬性取得執行中執行個體的資訊，例如執行個體的公有 IP 地址。
- `activity` – 產生屬性的活動；此範例中的設定事件。
- `rails_stack` – 包含 Rails App Server layer 之堆疊的 Rails 堆疊。
- `deployment` – 這些屬性是否與部署相關聯。此範例中已設為 `null`，因為他們是與設定事件關聯。
- `valid_client_activities` – 有效用戶端活動的清單。

`opsworks` 屬性其後跟隨幾個其他最上層屬性，包含下列項目：

- `opsworks_custom_cookbooks` – 是否啟用自訂技術指南。若為是，則屬性會包含自訂配方的清單。
- `recipes` – 此活動執行的配方。
- `opsworks_rubygems` – 執行個體的 RubyGems 版本。
- `ssh_users` – SSH 使用者清單；此範例中沒有。
- `opsworks_bundler` – 綁定器版本及其是否已啟用。
- `deploy` – 部署活動的相關資訊；此範例中沒有。

部署屬性

部署事件或執行配方堆疊命令的屬性由內建的堆疊組態及部署屬性，以及任何自訂堆疊或部署屬性組成 (此範例中則無)。下列 JSON 物件顯示與部署事件關聯之 php-app1 的屬性。該事件會將 SimplePHP 應用程式部署到堆疊的 PHP 執行個體。物件的大部分都是由與先前章節中說明的設定事件屬性相似的堆疊組態屬性組成，因此範例主要會聚焦在部署限定的屬性。如需各種屬性的詳細描述，請參閱堆疊組態及部署屬性：Linux 和內建技術指南屬性。

```
{
   ...
  "opsworks": {
    ...
```

```
    "activity": "deploy",
    "applications": [
      {
        "slug_name": "simplephp",
        "name": "SimplePHP",
        "application_type": "php"
      }
    ],
    "deployment": "5e6242d7-8111-40ee-bddb-00de064ab18f",
    ...
  },
  ...
{
  "ssh_users": {
  },
  "deploy": {
    "simplephpapp": {
      "application": "simplephpapp",
      "application_type": "php",
      "environment_variables": {
        "USER_ID": "168424",
        "USER_KEY": "somepassword"
      },
      "auto_bundle_on_deploy": true,
      "deploy_to": "/srv/www/simplephpapp",
      "deploying_user": "arn:aws:iam::123456789012:user/guysm",
      "document_root": null,
      "domains": [
        "simplephpapp"
      ],
      "migrate": false,
      "mounted_at": null,
      "rails_env": null,
      "restart_command": "echo 'restarting app'",
      "sleep_before_restart": 0,
      "ssl_support": false,
      "ssl_certificate": null,
      "ssl_certificate_key": null,
      "ssl_certificate_ca": null,
      "scm": {
        "scm_type": "git",
        "repository": "git://github.com/amazonwebservices/opsworks-demo-php-simple-
app.git",
        "revision": "version1",
```

```
        "ssh_key": null,
        "user": null,
        "password": null
      },
      "symlink_before_migrate": {
        "config/opsworks.php": "opsworks.php"
      },
      "symlinks": {
      },
      "database": {
      },
      "memcached": {
        "host": null,
        "port": 11211
      },
      "stack": {
        "needs_reload": false
      }
    }
  },
}
```

opsworks 屬性大致上與先前章節中的範例相同。以下區段是與部署最為相關的部分：

- activity – 與這些屬性相關聯的事件；此範例中的部署事件。
- applications – 包含每個應用程式一組屬性，提供應用程式的名稱、slug 名稱和類型。

  slug 名稱是應用程式的簡短名稱，由 AWS OpsWorks Stacks 從應用程式名稱產生。SimplePHP 的動態資料欄位名稱為 simplephp。
- deployment – 部署 ID，可唯一識別部署。

deploy 屬性包含正在部署之應用程式的相關資訊。例如，內建的部署配方會使用 deploy 屬性中的資料，來在適當的目錄中安裝檔案及建立資料庫連線檔案。deploy 屬性包含每個部署應用程式的一個屬性，以應用程式的短名命名。每個應用程式的屬性都包含以下屬性：

- environment_variables – 包含您已為應用程式定義的任何環境變數。如需詳細資訊，請參閱環境變數。
- domains – 根據預設，網域是應用程式的簡短名稱，在此範例中為 simplephpapp。若您已指派自訂網域，他們也會出現在這裡。如需詳細資訊，請參閱使用自訂網域。

- `application` – 應用程式的簡短名稱。
- `scm` – 此元素包含從其儲存庫下載應用程式檔案所需的資訊；在此範例中為 Git 儲存庫。
- `database` – 資料庫資訊，如果堆疊包含資料庫層。
- `document_root` – 文件根，`null`在此範例中設定為 ，表示根為公有。
- `ssl_certificate_ca`、`ssl_support`、 `ssl_certificate_key` – 指出應用程式是否支援 SSL。若有的話，`ssl_certificate_key` 和 `ssl_certificate_ca` 屬性會設為對應的憑證。
- `deploy_to` – 應用程式根目錄。

技術指南 101

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

生產層級 AWS OpsWorks 的 Stacks 堆疊通常需要一些自訂，這通常表示使用一或多個配方、屬性檔案或範本檔案實作自訂 Chef 技術指南。本主題是實作 Stacks AWS OpsWorks 技術指南的教學簡介。

如需 Stacks AWS OpsWorks 如何使用技術指南的詳細資訊，其中包括技術指南的簡短一般簡介，請參閱 技術指南和配方。如需如何實作和測試 Chef 配方的額外資訊，請參閱 Test-Driven Infrastructure with Chef, 2nd Edition。

教學範例分為兩個部分：

- 技術指南基本概念是一組範例演練，適合不熟悉 Chef 的使用者。經驗豐富的 Chef 使用者可以跳過此節。

  範例會帶您演練如何實作技術指南以執行常見任務 (例如安裝套件或建立目錄) 的基礎。為了簡化程序，您會使用一對有用的工具，Vagrant 和 Test Kitchen 來在虛擬機器中本機執行大多數的範例。在開始技術指南基本概念前，建議您先閱讀 Vagrant 和 Test Kitchen，了解如何安裝和使用這些工具。由於 Test Kitchen 尚不支援 Windows，因此所有的範例都只適用於 Linux，但會有附註，指示如何改寫以供 Windows 使用。
- 實作 Stacks AWS OpsWorks 的技術指南 說明如何實作 Stacks AWS OpsWorks 的配方，包括適用於 Windows 堆疊的配方。

它還包括一些更進階的 ，例如如何使用 Berkshelf 來管理外部技術指南。範例是針對新的 Chef 使用者撰寫，與技術指南基本概念中的範例相似。不過 AWS OpsWorks ，Stacks 的運作方式與 Chef 伺服器略有不同，因此我們建議經驗豐富的 Chef 使用者至少閱讀本節。

Vagrant 和 Test Kitchen

若您使用適用於 Linux 執行個體的配方，Vagrant 和 Test Kitchen 對於了解和初始開發及測試來說是非常有用的工具。這提供 Vagrant 和 Test Kitchen 的簡短描述，並引導您參閱安裝指示和演練，協助您設定並熟悉如何使用工具的基本概念。雖然 Vagrant 支援 Windows，但 Test Kitchen 不支援，因此針對這些工具只提供 Linux 範例。

Vagrant

Vagrant 提供在虛擬機器上執行和測試程式碼的一致環境。它支援各種環境，稱為 Vagrant 方塊，每個環境都代表設定的作業系統。對於 AWS OpsWorks Stacks，感興趣的環境是以 Ubuntu、Amazon 或 Red Hat Enterprise Linux (RHEL) 發行版本為基礎，因此範例主要使用名為 的 Vagrant 方塊`opscode-ubuntu-12.04`。

Vagrant 可供 Linux、Windows 和 Macintosh 系統使用，因此您可以使用您偏好的工作站，在任意支援的作業系統上實作和測試配方。本章的範例是在 Ubuntu Linux 系統上建立的，但將程序翻譯為 Windows 或 Macintosh 系統非常簡單。

Vagrant 基本上就是虛擬化提供者的包裝函式。大多數的範例使用 VirtualBox 提供者。VirtualBox 是免費的，可用於 Linux、Windows 和 Macintosh 系統。若您的系統上還沒有 VirtualBox，Vagrant 演練提供了安裝說明。請注意，您可以在 VirtualBox 上執行 Ubuntu 型環境，但 Amazon Linux 僅適用於 Amazon EC2 執行個體。但是，您可以在 VirtualBox 上執行相似的作業系統 (例如 CentOS)，這對初始開發和測試來說非常有用。

如需其他提供者的資訊，請參閱 Vagrant 文件。特別是，`vagrant-aws`外掛程式提供者可讓您將 Vagrant 與 Amazon EC2 執行個體搭配使用。此供應商在 Amazon Linux 上測試配方特別有用，僅適用於 Amazon EC2 執行個體。`vagrant-aws` 提供者是免費的，但您必須擁有 AWS 帳戶並為您使用的任何 AWS 資源支付費用。

此時，建議您參閱 Vagrant 的入門演練。其內容說明了如何在您的工作站上安裝 Vagrant，並教您了解使用 Vagrant 的基礎。請注意，本章內的範例沒有使用 Git 儲存庫，因此若您需要的話，您可以省略演練中的該部分。

Test Kitchen

Test Kitchen 可簡化您在 Vagrant 上執行和測試技術指南的程序。實際上來說，您很少需要直接使用 Vagrant。Test Kitchen 會執行大多數的常見任務，包含：

- 在 Vagrant 中啟動執行個體。
- 將技術指南傳輸到執行個體。
- 在執行個體上執行技術指南的配方。
- 在執行個體上測試技術指南的配方。
- 使用 SSH 來登入執行個體。

相較於直接安裝 Test Kitchen gem，我們建議您安裝 Chef DK。除了 Chef 本身之外，此套件還包含了 Test Kitchen、Berkshelf、ChefSpec 和其他幾項有用的工具。

此時，建議您參閱 Test Kitchen 的入門演練。其內容會教您如何使用 Test Kitchen 執行和測試配方的基礎。

> **Note**
>
> 本章中的範例會使用 Test Kitchen 以方便執行配方。若您偏好的話，您可以在完成手動驗證一節後停止入門演練，因為屆時其內容已涵蓋所有您針對範例需要了解的部分。但是，Test Kitchen 主要是一個支援像是 Bash 自動化測試系統 (BATS) 等測試框架的測試平台。建議您在之後找時間完成演練的剩餘部分，以了解如何使用 Test Kitchen 測試您的配方。

技術指南基本概念

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以使用技術指南來完成各式各樣的任務。下列主題假設您不熟悉 Chef，所以會說明如何使用技術指南完成一些常見的任務。由於 Test Kitchen 尚不支援 Windows，因此所有的範例都只適用於 Linux，但會有附註，指示如何改寫以供 Windows 使用。如果您不熟悉 Chef，即使您使用的是

Windows，我們仍建議您演練這些範例。本主題的大部分範例稍加變更都可以在 Windows 執行個體上使用，範例中會註明。所有的範例都是在虛擬機器中執行，因此您甚至都不需要有 Linux 電腦。只需在您平常的工作站上安裝 Vagrant 和 Test Kitchen 即可。

> **Note**
>
> 如果您想要在 Windows 執行個體上執行這些配方，最簡單的方法是建立 Windows 堆疊，在其中一個堆疊的執行個體上執行這些配方。如需如何在 AWS OpsWorks Stacks Windows 執行個體上執行配方的詳細資訊，請參閱 在 Windows 執行個體上執行配方。

請先確認您已安裝 Vagrant 和 Test Kitchen，並已完成它們的入門演練，再繼續操作。如需詳細資訊，請參閱Vagrant 和 Test Kitchen。

主題

- 配方結構
- 範例 1：安裝套件
- 範例 2：管理使用者
- 範例 3：建立目錄
- 範例 4：新增流程控制
- 範例 5：使用屬性
- 範例 6：建立檔案
- 範例 7：執行命令和指令碼
- 範例 8：管理服務
- 範例 9：使用 Amazon EC2 執行個體
- 後續步驟

配方結構

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

技術指南主要是一組可在執行個體上執行各種任務的「配方」。若要釐清如何實作配方，查看簡單的範例會有所幫助。以下是內建 HAProxy layer 的設定配方。這個時候只要關注整體結構，不必太過擔心詳細資訊，後續範例會有相關說明。

```
package 'haproxy' do
  action :install
end

if platform?('debian','ubuntu')
  template '/etc/default/haproxy' do
    source 'haproxy-default.erb'
    owner 'root'
    group 'root'
    mode 0644
  end
end

include_recipe 'haproxy::service'

service 'haproxy' do
  action [:enable, :start]
end

template '/etc/haproxy/haproxy.cfg' do
  source 'haproxy.cfg.erb'
  owner 'root'
  group 'root'
  mode 0644
  notifies :restart, "service[haproxy]"
end
```

> **Note**
>
> 如需此範例和其他範例的工作配方和相關檔案，請參閱 AWS OpsWorks Stacks 內建配方。

此範例強調關鍵的配方元素，下列各節中會詳加說明。

主題

- 資源

- 流量控制
- 隨附的配方

資源

配方的組成主要是一組 Chef「資源」。它們每一個都會指定特定方面的執行個體最終狀態，例如要安裝的套件或要啟動的服務。此範例有四項資源：

- package 資源，在本範例中代表已安裝的套件，即 HAProxy 伺服器。
- service 資源，在本範例中代表服務，即 HAProxy 服務。
- 兩個 template 資源，在本範例中代表要從指定的範本建立的檔案，即兩個 HAProxy 組態檔案。

資源以宣告的方式指定執行個體狀態。在幕後，每項資源都有一個相關聯的「提供者」，執行所需的任務，例如安裝套件、建立及設定目錄、啟動服務等等。如果任務的詳細資訊取決於特定的作業系統，則資源會有多個提供者，並會使用最適合系統的那一個。例如，在 Red Hat Linux 系統中，package 提供者會使用 yum 安裝套件。在 Ubuntu Linux 系統中，package 提供者會使用 apt-get。

您要將資源實作為具有下列一般格式的 Ruby 程式碼區塊。

```
resource_type "resource_name" do
  attribute1 'value1'
  attribute2 'value2'
  ...
  action :action_name
  notifies : action 'resource'
end
```

這些元素是：

資源類型

(必要) 本範例包括三種資源類型：package、service 和 template。

資源名稱

(必要) 此名稱可識別特定的資源，有時用為其中一個屬性的預設值。在此範例中，package 代表名為 haproxy 的套件資源，而第一個 template 資源代表名為 /etc/default/haproxy 的組態檔案。

Attributes

(選用) 這些屬性會指定資源組態，並會隨著資源類型及您設定資源的方式而不同。

- 此範例的 `template` 資源會明確定義一組屬性，指定已建立的檔案來源、擁有者、群組和模式。
- 此範例的 `package` 和 `service` 資源不會明確定義任何屬性。

  資源名稱一般是必要屬性的預設值，有時候只需要它就夠了。例如，資源名稱是 `package` 資源之 `package_name` 屬性的預設值，它是唯一需要的屬性。

也有一些特殊化的屬性，稱之為保護屬性，它們會指定資源提供者何時採取動作。例如，`only_if` 屬性只有在符合指定條件時，才會指示資源提供者採取動作。HAProxy 配方不使用保護屬性，但下列幾個範例會使用它們。

動作和通知

(選用) 動作和通知會指定提供者要執行的任務。

- `action` 指示提供者採取指定的動作，例如安裝或建立。

  每項資源都有一組取決於特定資源的動作，其中一個是預設動作。在此範例中，`package` 資源的動作是 `install`，它會指示提供者安裝套件。第一項 `template` 資源沒有 `action` 元素，所以提供者採取預設的 `create` 動作。
- `notifies` 指示另一項資源的提供者執行動作，但只有在資源狀態變更後。

  `notifies` 通常搭配 `template` 和 `file` 等資源使用，執行的任務如在修改組態檔案後重新啟動服務。資源沒有預設的通知。如果您想要有通知，資源必須有明確的 `notifies` 元素。在 HAProxy 配方中，如果相關聯的組態檔案已變更，第二項 `template` 資源會通知 haproxy `service` 資源重新啟動 HAProxy 服務。

資源有時取決於作業系統。

- 有些資源只能用於 Linux 或 Windows 系統。

  例如，package 會在 Linux 系統上安裝套件，而在 Windows 系統上安裝套件要使用 windows_package。
- 有些資源可搭配任何作業系統，但需要特定系統的專屬屬性。

  例如，file 資源可用於 Linux 或 Windows 系統，但設定許可時使用不同的屬性集。

如需標準資源的說明，包括每項資源的可用屬性、動作和通知，請參閱關於資源和提供者。

流量控制

因為配方是 Ruby 應用程式，所以您可以使用 Ruby 控制結構在配方中納入流程控制。例如，您可以使用 Ruby 條件邏輯，讓配方在不同的系統中有不同的行為。HAProxy 配方包含的 `if` 區塊可使用 `template` 資源建立組態檔案，但只有當配方在 Debian 或 Ubuntu 系統中執行時才可以。

另一個常見的情況是使用迴圈以不同的屬性設定來多次執行資源。例如，您可以使用迴圈以不同的目錄名稱多次執行 `directory` 資源，來建立一組目錄。

> **Note**
>
> 如果您不熟悉 Ruby，請參閱僅足夠適用於 Chef 的 Ruby，其中包含多數配方需要知道的內容。

隨附的配方

`include_recipe` 在您的程式碼中包含其他配方，讓您模組化您的配方，並在多種配方中重複使用相同的程式碼。當您執行主機配方時，Chef 會先使用指定的配方程式碼取代每個 `include_recipe` 元素，再執行主機配方。您使用標準的 Chef `cookbook_name::recipe_name` 語法來納入配方，其中 `recipe_name` 會省略 `.rb` 副檔名。此範例包含的配方 `haproxy::service`，代表 HAProxy 服務。

> **Note**
>
> 如果您在於 Chef 11.10 和更新版本上執行的配方中使用 `include_recipe` 來納入其他技術指南的配方，即必須使用 `depends` 陳述式在技術指南的 `metadata.rb` 檔案中宣告相依性。如需詳細資訊，請參閱實作配方：Chef 11.10。

範例 1：安裝套件

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

套件安裝是較常用的配方之一，相當簡單，視套件而定。例如，下列配方將 Git 安裝在 Linux 系統中。

```
package 'git' do
  action :install
end
```

package 資源處理套件安裝。在本範例中，您不需要指定任何屬性。資源名稱是 package_name 屬性的預設值，它可以識別套件。install 動作指示提供者安裝套件。略過 install 可讓程式碼變得更簡單，它是 package 資源的預設動作。當您執行配方時，Chef 會使用適當的提供者安裝套件。在本範例要使用的 Ubuntu 系統中，提供者透過呼叫 apt-get 來安裝 Git。

> **Note**
>
> 在 Windows 系統上安裝軟體需要的程序略有不同。如需詳細資訊，請參閱安裝 Windows 軟體。

若要使用 Test Kitchen 在 Vagrant 中執行此配方，您必須先設定技術指南，然後初始化及設定 Test Kitchen。下列內容適用於 Linux 系統，但 Windows 和 Macintosh 系統所用的程序基本類似。一開始先開啟終端機視窗，本章的所有範例都使用命令列工具。

準備技術指南

1. 在您的主目錄中建立名為 opsworks_cookbooks 的子目錄，它會包含本章所有的技術指南。然後，為此技術指南建立名為 installpkg 的子目錄中，導覽至它。
2. 在 installpkg 中建立名為 metadata.rb 的檔案，包含下列程式碼。

   ```
   name "installpkg"
   version "0.1.0"
   ```

   為求簡化，本章的範例只指定技術指南名稱和版本，但 metadata.rb 可以包含各種技術指南中繼資料。如需詳細資訊，請參閱 About Cookbook Metadata。

   > **Note**
   >
   > 初始化 Test Kitchen 之前請務必先建立 metadata.rb，它會使用資料建立預設的組態檔案。

3. 在 installpkg 中執行 kitchen init，這會初始化 Test Kitchen 並安裝預設的 Vagrant 驅動程式。

4. kitchen init 命令會在 installpkg 中建立名為 .kitchen.yml 的 YAML 組態檔案。在您偏好的文字編輯器中開啟 檔案。.kitchen.yml 檔案包含 platforms 區段，指定執行配方的系統。Test Kitchen 會在每個平台上建立執行個體並執行指定的配方。

   > **Note**
   >
   > Test Kitchen 預設一次只在一個平台上執行配方。如果您在建立執行個體的任何命令中新增 -p 參數，Test Kitchen 就會以平行方式在每個平台上執行配方。

   針對本範例，單一平台即已足夠，所以請編輯 .kitchen.yml 以移除 centos-6.4 平台。您的 .kitchen.yml 檔案現在看起來應如下：

   ```
   ---
   driver:
     name: vagrant

   provisioner:
     name: chef_solo

   platforms:
     - name: ubuntu-12.04

   suites:
     - name: default
       run_list:
         - recipe[installpkg::default]
       attributes:
   ```

   Test Kitchen 只執行 .kitchen.yml 執行清單中的配方。您使用 [*cookbook_name*::*recipe_name*] 格式識別配方，其中 *recipe_name* 省略 .rb 副檔名。.kitchen.yml 執行清單一開始會包含技術指南的預設配方 installpkg::default。這是您要實作的配方，因此您不需要修改執行清單。

5. 建立 installpkg 的子目錄，名為 recipes。

   如果技術指南包含配方，大多數情況下，它們必須位於 recipes子目錄中。

您現在可以將配方新增到技術指南，使用 Test Kitchen 在執行個體上執行它。

執行配方

1. 建立包含區段開頭處之 Git 安裝範例程式碼名為 `default.rb` 的檔案，並將它儲存到 `recipes` 子目錄中。
2. 在 `installpkg` 目錄中執行 `kitchen converge`。此命令會在 Vagrant 中啟動新的 Ubuntu 執行個體、將技術指南複製到執行個體，並啟動 Chef 執行以執行`.kitchen.yml`執行清單中的配方。
3. 若要驗證配方是否成功，請執行 `kitchen login` 開啟執行個體的 SSH 連線。然後，`git --version` 驗證是否已成功安裝 Git。若要返回您的工作站，請執行 `exit`。
4. 完成後，請執行 `kitchen destroy` 關機執行個體。下一個範例會使用不同的技術指南。

此範例是非常好的入門方法，但它非常簡單。其他套件的安裝比較複雜，您可能需要執行下列任一或所有作業：

- 建立並設定使用者。
- 建立一或多個資料、日誌等等的目錄。
- 安裝一或多個組態檔案。
- 為不同的作業系統指定不同的套件名稱或屬性值。
- 啟動服務，然後視需要再重新啟動它。

下列範例說明如何解決這些問題，以及其他一些有用的操作。

範例 2：管理使用者

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

另一個簡單的任務是管理執行個體上的使用者。下列配方會將新的使用者新增到 Linux 執行個體。

```
user "myuser" do
  home "/home/newuser"
  shell "/bin/bash"
end
```

您會使用 user 資源來管理 Linux 和 Windows 系統上的使用者，不過有些屬性只適用一個系統。範例建立名為 myuser 的使用者並指定其主目錄和 shell。不指定任何動作，所以資源會使用預設的 create 動作。您可以將屬性新增至 user 以指定各種其他設定，例如其密碼或群組 ID。您也可以為修改使用者設定或刪除使用者等相關的使用者管理任務使用 user。如需詳細資訊，請參閱 user。

執行配方

1. 在 opsworks_cookbooks 內建立並導覽至名為 newuser 的目錄。
2. 建立包含下列程式碼的 metadata.rb 檔案，並儲存至 newuser。

   ```
   name "newuser"
   version "0.1.0"
   ```
3. 初始化及設定 Test Kitchen (如範例 1：安裝套件中所述)，並在 recipes 目錄中新增 newuser 目錄。
4. 將具有範例配方的 default.rb 檔案新增至技術指南的 recipes 目錄。
5. 執行 kitchen converge 以執行配方。
6. 使用 kitchen login 登入執行個體，執行 cat /etc/passwd 驗證新的使用者是否存在。myuser 使用者應位在檔案底部。

範例 3：建立目錄

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

當您在執行個體上安裝套件時，通常需要建立一些組態檔案並將它們放置在適當的目錄中。不過，這些目錄可能還不存在。您可能還需要建立資料、日誌檔等等的目錄。例如，您首先啟動用於大多數範例的 Ubuntu 系統，/srv目錄沒有子目錄。如果您要安裝應用程式伺服器，您可能希望有資料檔案、日誌等等的 /srv/www/ 目錄，可能還有一些子目錄。下列配方在執行個體上建立 /srv/www/。

```
directory "/srv/www/" do
  mode 0755
  owner 'root'
  group 'root'
  action :create
end
```

您使用 directory 資源在 Linux 和 Windows 系統上建立並設定目錄，但某些屬性的用法不盡相同。資源名稱是用於將資源的 path 屬性預設值，因此本範例會建立 /srv/www/ 並指定其 mode、owner 和 group 屬性。

執行配方

1. 在 opsworks_cookbooks 中建立名為 createdir 的目錄，導覽至它。
2. 初始化及設定 Test Kitchen (如範例 1：安裝套件中所述)，並在 recipes 中新增 createdir 目錄。
3. 將具有配方程式碼的 default.rb 檔案新增至技術指南的 recipes 子目錄。
4. 執行 kitchen converge 以執行配方。
5. 執行 kitchen login，導覽至 /srv 並驗證它是否有 www 子目錄。
6. Run exit 返回到工作站，但保持執行個體執行。

> **Note**
>
> 在執行個體中建立主目錄的相對目錄，使用 #{ENV['HOME']} 表示主目錄。例如，下列內容會建立 ~/shared 目錄。
>
> ```
> directory "#{ENV['HOME']}/shared" do
>   ...
> end
> ```

假設您想要建立更深的巢狀目錄，例如 /srv/www/shared。您可以修改前述配方，如下所示。

```
directory "/srv/www/shared" do
```

```
  mode 0755
  owner 'root'
  group 'root'
  action :create
end
```

執行配方

1. 以前述配方取代 default.rb 中的程式碼。
2. 從 kitchen converge 目錄執行 createdir。
3. 若要確認已確實建立目錄，請執行 kitchen login、導覽至 /srv/www，然後驗證它是否包含 shared 子目錄。
4. 執行 kitchen destroy 關閉執行個體。

您會發現 kitchen converge 命令執行速度更快了。這是因為執行個體已在執行，所以不需要開機執行個體、安裝 Chef 等等。Test Kitchen 只是將更新的技術指南複製到執行個體，並啟動 Chef 執行。

現在再次執行 kitchen converge，這會在全新的執行個體中執行配方。您現在會看到下列結果。

```
Chef Client failed. 0 resources updated in 1.908125788 seconds
[2014-06-20T20:54:26+00:00] ERROR: directory[/srv/www/shared] (createdir::default line
 1) had an error: Chef::Exceptions::EnclosingDirectoryDoesNotExist: Parent directory /
srv/www does not exist, cannot create /srv/www/shared
[2014-06-20T20:54:26+00:00] FATAL: Chef::Exceptions::ChildConvergeError: Chef run
 process exited unsuccessfully (exit code 1)
>>>>>> Converge failed on instance <default-ubuntu-1204>.
>>>>>> Please see .kitchen/logs/default-ubuntu-1204.log for more details
>>>>>> ------Exception-------
>>>>>> Class: Kitchen::ActionFailed
>>>>>> Message: SSH exited (1) for command: [sudo -E chef-solo --config /tmp/kitchen/
solo.rb --json-attributes /tmp/kitchen/dna.json  --log_level info]
>>>>>> ----------------------
```

發生了什麼？ 問題是，directory 資源預設一次只能建立一個目錄，無法建立目錄鏈。此配方之前可以運作是因為您在執行個體上執行的最初配方已建立 /srv/www，因此建立 /srv/www/shared 只會建立一個子目錄。

> **Note**
>
> 當您執行 `kitchen converge` 時，請確定您知道您是在新的或現有的執行個體上執行您的配方。您可能會得到不同的結果。

若要建立子目錄鏈，請將 `recursive` 屬性新增至 `directory` 並將它設為 `true`。下列配方會在乾淨的執行個體上建立 `/srv/www/shared` 目錄。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end
```

範例 4：新增流程控制

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

有些配方只是一系列的 Chef 資源。在這種情況下，當您執行配方時，它僅會依序執行每個資源提供者。不過，有更精密的執行路徑通常會很有幫助。下列是兩個常見情況：

- 您想要某個配方以不同的屬性設定多次執行相同的資源。
- 您想要在不同的作業系統上使用不同的屬性設定。

您可以將 Ruby 控制結構併入配方以處理類似的情況。本節示範如何修改範例 3：建立目錄中的配方來處理這兩種情況。

主題

- 重複

- 條件式邏輯

重複

範例 3：建立目錄示範如何使用 directory 資源來建立目錄或目錄鏈。不過，假設您要建立兩個不同的目錄 /srv/www/config 和 /srv/www/shared。您可以分別為每個目錄實作目錄資源，但如果您想要建立非常多的目錄，這種方法太過繁瑣。下列配方示範處理此任務的較簡單方式。

```
[ "/srv/www/config", "/srv/www/shared" ].each do |path|
  directory path do
    mode 0755
    owner 'root'
    group 'root'
    recursive true
    action :create
  end
end
```

配方使用包含子目錄路徑的字串集合，而不是為每個子目錄分別使用目錄資源。Ruby each 方法從第一個元素開始，為每個集合元素執行一次資源。資源中以 path 變數表示元素，在本例中表示目錄路徑。您可以輕鬆地調整此範例建立任意數目的子目錄。

執行配方

1. 留在 createdir 目錄中，接下來數個範例都會使用該技術指南。
2. 若尚未完成，請執行 kitchen destroy 以從乾淨的執行個體開始。
3. 以範例取代 default.rb 中的程式碼並執行 kitchen converge。
4. 登入執行個體，您會在 /srv 下看到新建立的目錄。

您可以使用雜湊表為每次重複指定兩個值。下列配方建立 /srv/www/config 和 /srv/www/shared，每個都有不同的模式。

```
{ "/srv/www/config" => 0644, "/srv/www/shared" => 0755 }.each do |path, mode_value|
  directory path do
    mode mode_value
    owner 'root'
    group 'root'
```

```
    recursive true
    action :create
  end
end
```

執行配方

1. 若尚未完成，請執行 `kitchen destroy` 以從乾淨的執行個體開始。
2. 以範例取代 `default.rb` 中的程式碼並執行 `kitchen converge`。
3. 登入執行個體，您會在指定模式的 `/srv` 下看到新建立的目錄。

> **Note**
>
> AWS OpsWorks Stacks 配方通常會使用此方法從堆疊組態和部署 JSON 擷取值，基本上是大型雜湊資料表，並將其插入資源。如需範例，請參閱「部署配方」。

條件式邏輯

您也可以使用 Ruby 條件式邏輯來建立多個執行分支。下列配方使用 `if-elsif-else` 邏輯來擴展前一個範例，以便建立名為 `/srv/www/shared` 的子目錄，但只限於 Debian 和 Ubuntu 系統。針對所有其他系統，其會記錄顯示在 Test Kitchen 輸出中的錯誤訊息。

```
if platform?("debian", "ubuntu")
  directory "/srv/www/shared" do
    mode 0755
    owner 'root'
    group 'root'
    recursive true
    action :create
  end
else
  log "Unsupported system"
end
```

執行範例配方

1. 如果您的執行個體仍在執行中，請執行 `kitchen destroy` 關閉它。

2. 以範例程式碼取代 `default.rb` 中的程式碼。
3. 編輯 `.kitchen.yml` 將 CentOS 6.4 系統新增到平台清單。檔案的 `platforms` 區段看起來應該如下：

```
...
platforms:
  - name: ubuntu-12.04
  - name: centos-6.4
...
```

4. 執行 `kitchen converge`，這會建立執行個體，並依序為 `.kitchen.yml` 中的每個平台執行配方。

> **Note**
>
> 如果您想收斂成僅只一個執行個體，請將執行個體名稱新增為參數。例如，若要將配方收斂成僅在 Ubuntu 平台，請執行 `kitchen converge default-ubuntu-1204`。如果您忘記了平台名稱，只要執行 `kitchen list` 即可。

您應該會在 Test Kitchen 的 CentOS 部分看到您的日誌訊息，它看起來類似如下：

```
...
Converging 1 resources
Recipe: createdir::default
* log[Unsupported system] action write[2014-06-23T19:10:30+00:00] INFO: Processing
 log[Unsupported system] action write (createdir::default line 12)
[2014-06-23T19:10:30+00:00] INFO: Unsupported system

[2014-06-23T19:10:30+00:00] INFO: Chef Run complete in 0.004972162 seconds
```

您現在可以登入執行個體並驗證是否已建立目錄。不過，現在無法僅執行 `kitchen login`。您必須透過附加平台名稱來指定執行個體，例如 `kitchen login default-ubuntu-1204`。

> **Note**
>
> 如果 Test Kitchen 命令接受執行個體名稱，您就不需要輸入完整的名稱。Test Kitchen 會將執行個體名稱視為 Ruby 規則表達式，因此您只需要可提供唯一相符的足夠字元即可。例如，您

可以透過執行 `kitchen converge ub` 收斂成僅 Ubuntu 執行個體，或透過執行 `kitchen login 64` 登入 CentOS 執行個體。

此時可能有的疑問，是配方如何知道它在哪個平台上執行。Chef 為收集系統資料 (包括平台) 的每個回合都執行名為 Ohai 的工具，在稱為「節點物件」的結構中用一組屬性表示此資料。Chef `platform?` 方法使用括號來比較系統和 Ohai 平台值，如果其中一項符合即傳回 true。

您可以使用 `node['attribute_name']` 直接參考您程式碼中的節點屬性值。例如，平台值以 `node['platform']` 表示。例如，您可能將前述範例撰寫如下。

```
if node[:platform] == 'debian' or node[:platform] == 'ubuntu'
  directory "/srv/www/shared" do
    mode 0755
    owner 'root'
    group 'root'
    recursive true
    action :create
  end
else
  log "Unsupported system"
end
```

在配方中包含條件邏輯的常見原因，是為了因應不同的 Linux 系列有時使用不同的套件名稱、目錄等等。例如，Apache 套件名稱在 CentOS 系統為 `httpd`，在 Ubuntu 系統為 `apache2`。

如果只是因為不同的系統需要不同的字串，Chef `value_for_platform` 方法是比 `if-elsif-else` 更簡單的解決方案。下列配方在 CentOS 系統中建立 `/srv/www/shared` 目錄，在 Ubuntu 系統中建立 `/srv/www/data` 目錄，在所有其他系統中建立 `/srv/www/config`。

```
data_dir = value_for_platform(
  "centos" => { "default" => "/srv/www/shared" },
  "ubuntu" => { "default" => "/srv/www/data" },
  "default" => "/srv/www/config"
)
directory data_dir do
  mode 0755
  owner 'root'
  group 'root'
```

```
  recursive true
  action :create
end
```

value_for_platform 將適當的路徑指派給 data_dir，而 directory 資源則使用該值建立目錄。

執行範例配方

1. 如果您的執行個體仍在執行中，請執行 kitchen destroy 關閉它。
2. 以範例程式碼取代 default.rb 中的程式碼。
3. 執行 kitchen converge，然後登入每個執行個體，以驗證是否有適當的目錄。

範例 5：使用屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

除平台之外，前面各節中的配方針對所有項目都使用硬式編碼值。這種方法很麻煩，例如，當您想要在多個配方中使用相同的值時。您可以在技術指南中包含屬性檔案，從配方分別定義值。

屬性檔案是將值指派給一或多個屬性的 Ruby 應用程式。它必須位在技術指南的 attributes 資料夾。Chef 將屬性併入到節點物件，而所有配方可以透過參考屬性使用這些屬性值。本主題示範如何修改重複中的配方來使用這些屬性。下列是供參考的原始配方。

```
[ "/srv/www/config", "/srv/www/shared" ].each do |path|
  directory path do
    mode 0755
    owner 'root'
    group 'root'
    recursive true
    action :create
  end
end
```

下列定義子目錄名稱、模式、擁有者和群組值的屬性。

```
default['createdir']['shared_dir'] = 'shared'
default['createdir']['config_dir'] = 'config'
default['createdir']['mode'] = 0755
default['createdir']['owner'] = 'root'
default['createdir']['group'] = 'root'
```

注意下列事項：

- 每個定義都是以「屬性類型」開始。

  如果屬性定義超過一次，可能是在不同屬性檔案中，屬性類型會指定屬性的優先順序，這會決定將哪個定義併入節點物件。如需詳細資訊，請參閱屬性優先順序。本範例中的所有定義均有 `default` 屬性類型，這是此用途的一般類型。
- 屬性有巢狀名稱。

  節點物件基本上是可任意深度嵌套的雜湊表，所以屬性名稱可以是、也普遍是巢狀的。此屬性檔案遵循使用巢狀名稱的標準實務，以技術指南名稱 `createdir` 為第一個元素。

使用 createdir 為屬性第一個元素的原因是，當您執行 Chef 執行時，Chef 會將每個技術指南的屬性納入節點物件。使用 AWS OpsWorks Stacks，除了您定義的任何屬性之外，節點物件還包含來自內建技術指南的大量屬性。在屬性名稱中包含技術指南名稱，可降低與其他技術指南屬性名稱發生衝突的風險，特別是當您的屬性名稱類似 `port` 或 `user` 時。除非您想要覆寫該屬性的值，否則請不要將屬性命名為類似 `[:apache2][:user]`。如需詳細資訊，請參閱使用自訂技術指南屬性。

下列範例顯示使用屬性的原始配方，不是使用硬式編碼值。

```
[ "/srv/www/#{node['createdir']['shared_dir']}", "/srv/www/#{node['createdir']
['config_dir']}" ].each do |path|
  directory path do
    mode node['createdir']['mode']
    owner node['createdir']['owner']
    group node['createdir']['group']
    recursive true
    action :create
  end
end
```

> **Note**
>
> 如果您想要將屬性值納入字串中，請用 #{} 括住它。在上述範例中，#{node['createdir']['shared_dir']} 會將 "shared" 附加到 "/srv/www/" 的後面。

執行配方

1. 執行 `kitchen destroy` 以從乾淨的執行個體開始。
2. 以前述的配方範例取代 `recipes/default.rb` 中的程式碼。
3. 建立名為 `createdir` 的 `attributes` 子目錄，並新增包含屬性定義之名為 `default.rb` 的檔案。
4. 編輯 `.kitchen.yml` 將 CentOS 從平台清單中移除。
5. 執行 `kitchen converge`，然後登入執行個體，驗證 `/srv/www/shared` 和 `/srv/www/config` 是否在此。

> **Note**
>
> 透過 AWS OpsWorks Stacks，將值定義為屬性可提供額外的優點；您可以使用自訂 JSON 來覆寫每個堆疊或甚至每個部署的這些值。這是適用於各種用途，包括：
>
> - 您可以自訂您的配方行為，例如組態設定或使用者名稱，不必修改技術指南。
>
> 例如，您可以在不同的堆疊使用相同的技術指南，使用自訂 JSON 指定特定堆疊的關鍵組態設定。這可節省您修改技術指南或為每個堆疊使用不同技術指南所需的時間和心力。
> - 您不必將資料庫密碼等可能的敏感資訊放在技術指南儲存庫。
>
> 您可以改用屬性定義預設值，然後使用自訂的 JSON 以真實的值覆寫該值。
>
> 如需如何使用自訂 JSON 覆寫屬性的詳細資訊，請參閱覆寫屬性。

如果是相當簡單的名稱，此屬性檔案名為 `default.rb`，因為它是 Ruby 應用程式。例如，這表示您可以根據作業系統，使用條件式邏輯指定屬性值。在條件式邏輯中，您已在配方中為不同的 Linux 系列指定不同的子目錄名稱。使用屬性檔案，您可以改將條件式邏輯放在屬性檔案中。

下列屬性檔案使用 value_for_platform 指定不同的 ['shared_dir'] 屬性值，視作業系統而定。其他條件可以使用 Ruby if-elsif-else 邏輯或 case 陳述式。

```
data_dir = value_for_platform(
  "centos" => { "default" => "shared" },
  "ubuntu" => { "default" => "data" },
  "default" => "user_data"
)
default['createdir']['shared_dir'] = data_dir
default['createdir']['config_dir'] = "config"
default['createdir']['mode'] = 0755
default['createdir']['owner'] = 'root'
default['createdir']['group'] = 'root'
```

執行配方

1. 執行 kitchen destroy 從全新的執行個體開始。
2. 以前述範例取代 attributes/default.rb 中的程式碼。
3. 編輯 .kitchen.yml 將 CentOS 平台新增到平台部分，如條件式邏輯中所述。
4. 執行 kitchen converge，然後登入執行個體，驗證目錄是否在此。

完成後，請執行 kitchen destroy 終止執行個體。下一個範例會使用新的技術指南。

範例 6：建立檔案

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

在您建立目錄之後，您通常需要在目錄中放入組態檔案、資料檔案等等。本主題顯示在執行個體上安裝檔案的兩種方式。

主題

- 從技術指南安裝檔案

- 從範本建立檔案

從技術指南安裝檔案

在執行個體上安裝檔案的最簡單方法是使用 cookbook_file 資源，將檔案從技術指南複製到 Linux 和 Windows 系統之執行個體上的指定位置。此範例會擴展範例 3：建立目錄中的配方，在目錄建立之後將資料檔案新增到 /srv/www/shared。下列的原始配方提供您參考。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end
```

設定技術指南

1. 在 opsworks_cookbooks 目錄中建立名為 createfile 的目錄，導覽至它。
2. 將 metadata.rb 檔案新增至具有以下內容的 createfile。

   ```
   name "createfile"
   version "0.1.0"
   ```

3. 初始化及設定 Test Kitchen (如範例 1：安裝套件中所述)，並從 platforms 清單中移除 CentOS。
4. 將 recipes 子目錄新增至 createfile。

要安裝的檔案包含下列 JSON 資料。

```
{
  "my_name" : "myname",
  "your_name" : "yourname",
  "a_number" : 42,
  "a_boolean" : true
}
```

設定資料檔案

1. 將 files 子目錄新增至 createfile，default 子目錄新增至 files。所有使用 cookbook_file 安裝的檔案都必須位在 files 的子目錄中，例如本範例中的 files/default。

   > **Note**
   >
   > 如果您想要為不同的系統指定不同的檔案，您可以將每個系統專屬的檔案放在以系統為名的子資料夾中，例如 files/ubuntu。cookbook_file 資源會複製適當的系統專屬檔案 (如果有的話)，否則使用 default 檔案。如需詳細資訊，請參閱 cookbook_file。

2. 使用上述範例中的 JSON 建立名為 example_data.json 的檔案，然後將它新增至 files/default。

下列配方會將 example_data.json 複製到指定的位置。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end

cookbook_file "/srv/www/shared/example_data.json" do
  source "example_data.json"
  mode 0644
  action :create_if_missing
end
```

在目錄資源建立 /srv/www/shared 之後，cookbook_file 資源會將 example_data.json 複製到該目錄，並且設定檔案的使用者、群組和模式。

> **Note**
>
> cookbook_file 資源引入新的動作：create_if_missing。您也可以使用 create 動作，但這會覆寫現有的檔案。如果您不想覆寫任何內容，請使用 create_if_missing，當 example_data.json 還不存在時才安裝。

執行配方

1. 執行 kitchen destroy 從全新的執行個體開始。
2. 建立包含前述配方的 default.rb 檔案，並儲存至 recipes。
3. 執行 kitchen converge，然後登入執行個體，驗證 /srv/www/shared 是否包含 example_data.json。

從範本建立檔案

cookbook_file 資源適合某些用途，但只會安裝技術指南中已有的檔案。template 透過以動態方式從範本建立檔案，讓您用更有彈性的方式在 Windows 或 Linux 執行個體上安裝檔案。然後，您可以在執行時間判斷檔案的詳細資訊，視需要變更它們。例如，當您啟動執行個體時，您可能希望組態檔案有特定的設定，稍後當您在堆疊中新增更多執行個體時，修改此設定。

本範例修改 createfile 技術指南以使用 template 資源安裝稍為修改的 example_data.json 版本。

已安裝的檔案看起來如下：

```
{
  "my_name" : "myname",
  "your_name" : "yourname",
  "a_number" : 42,
  "a_boolean" : true,
  "a_string" : "some string",
  "platform" : "ubuntu"
}
```

Template 資源通常搭配屬性檔案使用，因此範例會使用一個屬性檔案來定義下列值。

```
default['createfile']['my_name'] = 'myname'
default['createfile']['your_name'] = 'yourname'
default['createfile']['install_file'] = true
```

設定技術指南

1. 刪除 createfile 技術指南的 files 目錄及其內容。
2. 將 attributes 子目錄新增至 createfile，然後將 default.rb 檔案新增至包含前述屬性定義的 attributes。

範本是一種 .erb 檔案，基本上是最終檔案的複本，某些內容由預留位置表示。當 template 資源建立檔案時，會將範本的內容複製到指定的檔案，以其獲指派的值覆寫預留位置。下列為 example_data.json 範本。

```
{
  "my_name" : "<%= node['createfile']['my_name'] %>",
  "your_name" : "<%= node['createfile']['your_name'] %>",
  "a_number" : 42,
  "a_boolean" : <%= @a_boolean_var %>,
  "a_string" : "<%= @a_string_var %>",
  "platform" : "<%= node['platform'] %>"
}
```

這些 <%=...%> 值是預留位置。

- <%=node[...]%> 表示節點屬性值。

  在此範例中，"your_name" 值是表示技術指南屬性檔案中其中一個屬性值的預留位置。
- 如前文所述，<%=@...%> 表示範本資源中定義的變數值。

建立範本檔案

1. 將 templates 子目錄新增至 createfile 技術指南，default 子目錄新增至 templates。

> **Note**
>
> templates 目錄的作用如同 files 目錄。您可以將系統專屬範本放入以系統為名的子目錄，例如 ubuntu。template 資源會使用適當的系統專屬範本 (如果有的話)，否則使用 default 範本。

2. 建立名為 example_data.json.erb 的檔案，放入 templates/default 目錄。範本可使用任意名稱，但通常會在檔案名稱後附加 .erb，包括任何副檔名。

下列配方使用 template 資源建立 /srv/www/shared/example_data.json。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end

template "/srv/www/shared/example_data.json" do
  source "example_data.json.erb"
  mode 0644
  variables(
    :a_boolean_var => true,
    :a_string_var => "some string"
  )
  only_if {node['createfile']['install_file']}
end
```

此 template 資源從範本建立 example_data.json，然後將它安裝在 /srv/www/shared。

- 範本名稱 /srv/www/shared/example_data.json，指定已安裝檔案的路徑和名稱。
- source 屬性指定建立檔案所使用的範本。
- mode 屬性指定已安裝檔案的模式。
- 資源會定義兩個變數：a_boolean_var 和 a_string_var。

  當資源建立 example_data.json 時，會使用資源中的對應值覆寫範本中的變數預留位置。

- only_if guard 屬性指示資源只有當 ['createfile']['install_file'] 設為 true 時才建立檔案。

執行配方

1. 執行 kitchen destroy 從全新的執行個體開始。
2. 以前述範例取代 recipes/default.rb 中的程式碼。
3. 執行 kitchen converge，然後登入執行個體，驗證檔案是否位於 /srv/www/shared 且具有正確的內容。

完成後，請執行 kitchen destroy 關機執行個體。下一節使用新的技術指南。

範例 7：執行命令和指令碼

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Chef 資源可以處理執行個體上的各式任務，但有時使用 shell 命令或指令碼會更好。例如，您可能已有用以達成特定任務的指令碼，而且繼續使用它們比實作新程式碼更容易。本節示範如何在執行個體上執行命令或指令碼。

主題

- 執行命令
- 執行指令碼

執行命令

script 資源執行一或多個命令。它支援 csh、bash、Perl、Python 和 Ruby 命令解譯器，因此只要安裝適當的轉譯器，就可以用於 Linux 或 Windows 系統。本主題示範如何在 Linux 執行個體上執行簡單的 bash 命令。Chef 也支援 powershell_script 和 batch 資源在 Windows 上執行指令碼。如需詳細資訊，請參閱執行 Windows PowerShell 指令碼。

開始使用

1. 在 opsworks_cookbooks 目錄中建立名為 script 的目錄，導覽至它。
2. 將 metadata.rb 檔案新增至具有以下內容的 script。

```
name "script"
version "0.1.0"
```

3. 初始化及設定 Test Kitchen (如範例 1：安裝套件中所述)，並從 platforms 清單中移除 CentOS。
4. 在 script 中建立名為 recipes 的目錄。

您可以使用 script 資源本身執行命令，但 Chef 也支援資源的一組命令解譯器特定版本，以解譯器為名。下列配方使用 bash 資源執行簡單的 bash 指令碼。

```
bash "install_something" do
  user "root"
  cwd "/tmp"
  code <<-EOH
    touch somefile
  EOH
  not_if do
    File.exists?("/tmp/somefile")
  end
end
```

bash 資源設定如下。

- 它使用預設動作 run，在 code 區塊中執行命令。

  此範例有一個命令 touch somefile，但 code 區塊可以包含多個命令。
- user 屬性指定執行命令的使用者。
- cwd 屬性指定工作目錄。

  在此範例中，touch 會在 /tmp 目錄中建立檔案。
- 如果檔案已存在，not_if 保護屬性指示資源不採取任何動作。

執行配方

1. 建立包含前述範例程式碼的 default.rb 檔案，並將它儲存至 recipes。
2. 執行 kitchen converge，然後登入執行個體，驗證檔案是否位於 /tmp。

執行指令碼

script 資源很方便，尤其是當您只需要執行一或兩個命令時，但通常將指令碼存放在檔案中並執行該檔案會更好。execute 資源在 Linux 或 Windows 上執行指定的可執行檔，包括指令碼檔案。本主題修改前述範例中的 script 技術指南，使用 execute 執行簡單的 shell 指令碼。您可以輕鬆將此範例擴展為較複雜的指令碼，或其他類型的可執行檔。

設定指令碼檔案

1. 將 files 子目錄新增至 script，default 子目錄新增至 files。
2. 建立包含下列內容且名為 touchfile 的檔案，將它新增至 files/default。本範例中使用常見的 Bash 解譯器程式碼，但如有需要可取代為您 shell 環境中使用的解譯器。

   ```
   #!/usr/bin/env bash
   touch somefile
   ```

   指令碼檔案可以包含任意數目的命令。為了方便起見，此範例指令碼只有單一 touch 命令。

下列配方執行指令碼。

```
cookbook_file "/tmp/touchfile" do
  source "touchfile"
  mode 0755
end

execute "touchfile" do
  user "root"
  cwd "/tmp"
  command "./touchfile"
end
```

cookbook_file 資源將指令碼檔案複製到 /tmp，並設定模式讓檔案變成可執行檔。然後 execute 資源執行此檔案，如下所示：

- `user` 屬性指定命令的使用者 (本範例中為 `root`)。
- `cwd` 屬性指定工作目錄 (本範例中為 `/tmp`)。
- 此 `command` 屬性指定要執行的指令碼 (本範例中為 `touchfile`)，位於工作目錄。

執行配方

1. 以前述範例取代 `recipes/default.rb` 中的程式碼。
2. 執行 `kitchen converge`，然後登入執行個體以驗證 `/tmp` 現在包含模式設定為 0755 的指令碼檔案以及 `somefile`。

完成後，請執行 `kitchen destroy` 關機執行個體。下一節使用新的技術指南。

範例 8：管理服務

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

應用程式伺服器等套件，通常會有相關聯的服務，必須啟動、停止、重新啟動等等。例如，在安裝套件或執行個體完成開機之後，您需要啟動 Tomcat 服務，每次修改組態檔案後也要重新啟動服務。本主題以 Tomcat 應用程式伺服器為例，討論如何在 Linux 執行個體上管理服務的基本概念。細節上雖有一些差異，但 service 資源的作用在 Windows 執行個體上極其相似。如需詳細資訊，請參閱`service`。

> **Note**
>
> 此範例執行 Tomcat 最小安裝，僅足以示範如何使用 `service` 資源的基本概念。如需如何實作配方以取得更多 Tomcat 伺服器功能的範例，請參閱建立自訂 Tomcat 伺服器 Layer。

主題

- 定義及啟動服務
- 使用通知來啟動或重新啟動服務

定義及啟動服務

本節說明如何定義及啟動服務的基本概念。

開始使用

1. 在 opsworks_cookbooks 目錄中建立名為 tomcat 的目錄，導覽至它。
2. 將 metadata.rb 檔案新增至具有以下內容的 tomcat。

```
name "tomcat"
version "0.1.0"
```

3. 初始化及設定 Test Kitchen (如範例 1：安裝套件中所述)，並從 platforms 清單中移除 CentOS。
4. 將 recipes 子目錄新增至 tomcat。

您要使用 service 資源管理服務。下列預設配方安裝 Tomcat 並啟動服務。

```
execute "install_updates" do
  command "apt-get update"
end

package "tomcat7" do
    action :install
end

include_recipe 'tomcat::service'

service 'tomcat' do
  action :start
end
```

配方會執行下列動作：

- execute 資源執行 apt-get update 安裝最新的系統更新。

  對於此範例中使用的 Ubuntu 執行個體，您必須先安裝更新，才能安裝 Tomcat。其他系統可能會有不同的需求。
- package 資源安裝 Tomcat 7。

- 包含在內的 `tomcat::service` 配方會定義服務並在後文中討論。
- `service` 資源啟動 Tomcat 服務。

  您也可以使用此資源發出其他命令，例如停止和重新啟動服務。

下列範例顯示 `tomcat::service` 配方。

```
service 'tomcat' do
  service_name "tomcat7"
  supports :restart => true, :reload => false, :status => true
  action :nothing
end
```

此配方會建立 Tomcat 服務定義，如下所示：

- 其他配方使用資源名稱 `tomcat` 參考服務。

  例如，`default.rb` 參考 `tomcat` 以啟動服務。
- `service_name` 資源指定服務名稱。

  當您列出執行個體上的服務時，Tomcat 服務即命名為 tomcat7。
- `supports` 指定 Chef 如何管理服務的 `restart`、`reload` 和 `status` 命令。
  - `true` 表示 Chef 可使用 init 指令碼或其他服務提供者執行命令。
  - `false` 表示 Chef 必須嘗試使用其他方式執行命令。

請注意，`action` 設為 `:nothing`，指示資源不採取任何動作。服務資源不支援 `start` 和 `restart` 等動作。不過，此技術指南會遵循使用服務定義的標準實務，不採取任何動作，在其他位置啟動或重新啟動服務。每個啟動或重新啟動服務的配方，都必須先行定義，以便使用最簡單的方法將服務定義放在不同的配方中，並視需要將其包含在其他配方中。

> **Note**
>
> 為求簡化，此範例的預設配方會在執行服務定義後，使用 `service` 資源啟動服務。生產實作通常會使用 `notifies` 啟動或重新啟動服務，如後文所述。

執行配方

1. 建立包含預設配方範例的 default.rb 檔案，並將它儲存至 recipes。
2. 建立包含服務定義範例的 service.rb 檔案，並將它儲存至 recipes。
3. 執行 kitchen converge，然後登入執行個體，執行下列命令驗證服務是否正在執行。

```
sudo service tomcat7 status
```

> **Note**
>
> 如果您從 default.rb 分別執行 service.rb，您即必須編輯 .kitchen.yml 將 tomcat::service 新增到執行清單。不過，當您包含配方時，其程式碼會先納入父配方，再執行配方。因此，service.rb 基本上已是 default.rb 的一部分，而不需要另一個執行清單項目。

使用通知來啟動或重新啟動服務

生產實作通常不使用 service 啟動或重新啟動服務。而是將 notifies 新增到數個資源之中的任一個。例如，如果您想要在修改組態檔案後重新啟動服務，請將 notifies 包含在相關聯的 template 資源中。使用 notifies 明確重新啟動服務，比使用 service 資源的更有優勢，如下所示。

- notifies 元素只會在相關聯的組態檔案變更時才會重新啟動服務，所以沒有重新啟動不必要服務的風險。
- 無論回合包含多少個 notifies，Chef 在每個回合結束時最多重新啟動服務一次。

  例如，Chef 執行可能包含多項範本資源，它們每一個都會修改不同的組態檔案，檔案變更後需要重新啟動服務。不過，在 Chef 執行結束後，您通常只會想要重新啟動服務一次。否則，您可能嘗試重新啟動在前面重新啟動中尚未完全運作的服務，這會導致錯誤。

此範例會修改 tomcat::default 以包含 template 資源，使用 notifies 重新啟動服務。實際範例會使用範本資源建立其中一個 Tomcat 組態檔案的自訂版本，但這些檔案又長又複雜。為求簡化，此範例只使用從範本建立檔案中的範本資源。它和 Tomcat 沒有任何關係，但會提供簡單的方式，示範如何使用 notifies。如需如何使用範本建立 Tomcat 組態檔案的範例，請參閱安裝配方。

設定技術指南

1. 將 templates 子目錄新增至 tomcat，default 子目錄新增至 templates。
2. 將 example_data.json.erb 範本從 createfile 技術指南複製到 templates/default 目錄。
3. 將 attributes 子目錄新增至 tomcat。
4. 將 default.rb 屬性檔案從 createfile 技術指南複製到 attributes 目錄。

下列配方使用 notifies 重新啟動 Tomcat 服務。

```
execute "install_updates" do
  command "apt-get update"
end

package "tomcat7" do
    action :install
end

include_recipe 'tomcat::service'

service 'tomcat' do
  action :enable
end

directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end

template "/srv/www/shared/example_data.json" do
  source "example_data.json.erb"
  mode 0644
  variables(
    :a_boolean_var => true,
    :a_string_var => "some string"
  )
  only_if {node['createfile']['install_file']}
```

```
  notifies :restart, resources(:service => 'tomcat')
end
```

此範例將從範本建立檔案中的配方合併到上一節的配方，有兩大變更：

- service 資源仍然存在，但用途略有不同。

  :enable 動作可在開機時啟用 Tomcat 服務。
- 範本資源現在包含 notifies，如果 example_data.json 有所變更，則會重新啟動 Tomcat 服務。

  這可確保每次組態變更之後，Tomcat 一經安裝和重新啟動即啟動服務。

執行配方

1. 執行 kitchen destroy 以從乾淨的執行個體開始。
2. 以前述範例取代 default.rb 中的程式碼。
3. 執行 kitchen converge，然後登入執行個體，驗證服務是否正在執行。

> **Note**
>
> 如果您想要重新啟動服務，但配方不包含支援 template 的 notifies 等資源，您可改用虛擬 execute 資源。例如
>
> ```
> execute 'trigger tomcat service restart' do
>   command 'bin/true'
>   notifies :restart, resources(:service => 'tomcat')
> end
> ```
>
> execute 資源必須有 command 屬性，即使您只是使用資源來執行 notifies。此範例透過執行 /bin/true 規避需求，這個 shell 命令只會傳回成功代碼。

範例 9：使用 Amazon EC2 執行個體

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

此時，您一直在本機的 VirtualBox 中執行執行個體。雖然這快速又簡單，但您最終會想要在 Amazon EC2 執行個體上測試配方。特別是，如果您想要在 Amazon Linux 上執行配方，它只能在 Amazon EC2 上使用。您可以使用類似的系統，例如 CentOS 進行初步實作和測試，但在 Amazon Linux 上完整測試配方的唯一方法是使用 Amazon EC2 執行個體。

本主題說明如何在 Amazon EC2 執行個體上執行配方。Test Kitchen 和 Vagrant 的使用方法和前面各節差不多，但有兩點不同：

- 驅動程式是 `kitchen-ec2`，不是 Vagrant。
- 技術指南`.kitchen.yml`的檔案必須設定啟動 Amazon EC2 執行個體所需的資訊。

> **Note**
>
> 另一種方法是使用 `vagrant-aws` Vagrant 外掛程式。如需詳細資訊，請參閱 Vagrant AWS 提供者。

您需要 AWS 登入資料才能建立 Amazon EC2 執行個體。如果您沒有 AWS 帳戶，您可以取得一個，如下所示。

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。

您應該建立具有存取 Amazon EC2 許可的 IAM 使用者，並將使用者的存取和私密金鑰儲存至工作站上的安全位置。 Amazon EC2 Test Kitchen 會使用這些登入資料建立執行個體。向 Test Kitchen 提供登入資料，最好是將金鑰指派給您工作站的下列環境變數。

> **Warning**
>
> IAM 使用者具有長期憑證，這會造成安全風險。為了協助降低此風險，建議您只為這些使用者提供執行任務所需的許可，並在不再需要這些使用者時將其移除。

- AWS_ACCESS_KEY – 使用者的存取金鑰，看起來像 AKIAIOSFODNN7EXAMPLE。
- AWS_SECRET_KEY – 使用者的私密金鑰，看起來像 wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY。

此方法可以減少意外危害您帳戶的機率，例如，將包含您登入資料的專案上傳到公有儲存庫。

設定技術指南

1. 若要使用 `kitchen-ec2` 驅動程式，您必須在您的系統上安裝 `ruby-dev` 套件。下列範例命令說明如何使用 `aptitude` 將套件安裝在 Ubuntu 系統。

   ```
   sudo aptitude install ruby1.9.1-dev
   ```
2. `kitchen-ec2` 驅動程式是 Gem 套件，安裝方式如下：

```
gem install kitchen-ec2
```

視您的工作站而定，此命令可能需要 sudo，或者您也可以使用 Ruby 環境管理員，例如 RVM。此程序已經 kitchen-ec2 驅動程式 0.8.0 版測試，但現有更新版本。若要安裝特定版本，請執行 gem install kitchen-ec2 -v <*version number*>。

3. 您必須指定 Test Kitchen 可用來連線至執行個體的 Amazon EC2 SSH 金鑰對。如果您沒有 Amazon EC2 金鑰對，請參閱 Amazon EC2 金鑰對，以取得如何建立金鑰對的資訊。請注意，金鑰對必須屬於與執行個體相同的 AWS 區域。此範例使用美國西部（加利佛尼亞北部）。

   在您選取金鑰對後，請建立名為 opsworks_cookbooks 的 ec2_keys 子目錄，將金鑰對的私有金鑰 (.pem) 檔案複製到該子目錄。請注意，將私有金鑰放在 ec2_keys 中，只為方便略微簡化程式碼，它可在您系統上的任何位置。

4. 建立並導覽至名為 createdir-ec2 的 opsworks_cookbooks 子目錄。

5. 將 metadata.rb 檔案新增至具有以下內容的 createdir-ec2。

   ```
   name "createdir-ec2"
   version "0.1.0"
   ```

6. 初始化 Test Kitchen，如範例 1：安裝套件中所述。下一節說明如何設定 .kitchen.yml，這對於 Amazon EC2 執行個體來說更為複雜。

7. 將 recipes 子目錄新增至 createdir-ec2。

為 Amazon EC2 設定 .kitchen.yml

您可以使用kitchen-ec2驅動程式啟動適當設定.kitchen.yml之 Amazon EC2 執行個體所需的資訊進行設定。以下是美國西部（加利佛尼亞北部）區域中 Amazon Linux 執行個體.kitchen.yml的檔案範例。

```
driver:
  name: ec2
  aws_ssh_key_id: US-East1
  region: us-west-1
  availability_zone: us-west-1c
  require_chef_omnibus: true
  security_group_ids: sg........
  subnet_id: subnet-.........
```

```
  associate_public_ip: true
  interface: dns

provisioner:
  name: chef_solo

platforms:
  -name: amazon
  driver:
    image_id: ami-xxxxxxxx
  transport:
    username: ec2-user
    ssh_key: ../ec2_keys/US-East1.pem

suites:
  - name: default
    run_list:
      - recipe[createdir-ec2::default]
    attributes:
```

您可以使用 provisioner 和 suites 區段的預設設定，但必須修改預設的 driver 和 platforms 設定。此範例使用最少的設定清單，接受餘數的預設值。如需完整的 kitchen-ec2 設定清單，請參閱 Kitchen::Ec2：適用於 Amazon EC2 的 Test Kitchen 驅動程式。

此範例設定下列 driver 屬性。假設您已將使用者存取金鑰和秘密金鑰指派給標準環境變數，如前所述。驅動程式預設使用這些金鑰。否則，您必須將 aws_access_key_id 和 aws_secret_access_key 新增至 driver 屬性，設為適當的金鑰值，明確指定金鑰。

name

(必要) 此屬性必須設定為 ec2。

aws_ssh_key_id

（必要）Amazon EC2 SSH 金鑰對名稱，US-East1在此範例中名為 。

transport.ssh_key

(必要) 您為 aws_ssh_key_id 指定之金鑰的私有金鑰 (.pem) 檔案。在本範例中，此檔案名為 US-East1.pem，位在 ../opsworks/ec2_keys 目錄中。

region

(必要) 執行個體的 AWS 區域。此範例使用美國西部（加利佛尼亞北部），由 表示us-west-1)。

availability_zone

(選用) 執行個體的可用區域。如果您省略此設定，Test Kitchen 會使用指定區域的預設可用區域，us-west-1b即美國西部（加利佛尼亞北部）。不過，您的帳戶可能無法使用預設區域。在這種情況下，您必須明確指定可用區域。發生這種情況時，用來準備範例的帳戶不會支援 us-west-1b，所以範例會明確指定 us-west-1c。

require_chef_omnibus

設為 true 時，此設定會確保使用 omnibus 安裝程式在所有平台執行個體上安裝 chef-client。

security_group_ids

(選用) 套用到執行個體的安全群組 ID 清單。此設定會將 default 安全群組套用到執行個體。請確定安全群組傳入規則允許送入 SSH 連線，否則 Test Kitchen 無法與執行個體通訊。如果您使用 default 安全群組，您可能需要因應情況編輯它。如需詳細資訊，請參閱 Amazon EC2 安全群組。

subnet_id

執行個體的目標子網路 ID (如適用)。

associate_public_ip

如果您想要能夠從網際網路存取執行個體，則可以讓 Amazon EC2 將公有 IP 地址與執行個體建立關聯。

interface

您用來存取執行個體的主機名稱組態類型。有效值為 dns、public、private 或 private_dns。如果您不指定此屬性的值，kitchen-ec2 會以下列順序設定主機名稱組態。如果略過此屬性，即不設定組態類型。

1. DNS 名稱
2. 公有 IP 地址
3. 私有 IP 地址
4. 私有 DNS 名稱

Important

您應該建立使用者並將這些登入資料提供給 Test Kitchen，而不是將您的帳戶登入資料用於存取和私密金鑰。如需詳細資訊，請參閱管理 AWS 存取金鑰的最佳實務。

請小心不要將 .kitchen.yml 放在公開存取的位置；例如，上傳到 GitHub 或 Bitbucket 的公有儲存庫。這麼做會公開您的登入資料，而且可能會危害您帳戶的安全性。

kitchen-ec2 驅動程式為下列平台提供預設的支援：

- ubuntu-10.04
- ubuntu-12.04
- ubuntu-12.10
- ubuntu-13.04
- ubuntu-13.10
- ubuntu-14.04
- centos-6.4
- debian-7.1.0
- windows-2012r2
- windows-2008r2

如果您想要使用一或多個這些平台，請將適當的平台名稱新增至 platforms。kitchen-ec2 驅動程式會自動選取適當的 AMI 並產生 SSH 使用者名稱。您可以使用其他平台，此範例使用 Amazon Linux，但您必須明確指定下列platforms屬性。

name

平台名稱。此範例使用 Amazon Linux，所以 name 設為 amazon。

driver

driver 屬性包含下列項目：

- image_id – 平台的 AMI，必須屬於指定的區域。此範例使用 ami-ed8e9284，這是來自美國西部 （加利佛尼亞北部） 區域的 Amazon Linux AMI。
- transport.username – Test Kitchen 用來與執行個體通訊的 SSH 使用者名稱。

  針對 Amazon Linux，請使用 ec2-user。其他 AMI 可能會有不同的使用者名稱。

以範例取代 .kitchen.yml 中的程式碼，將適當的值指派給帳戶專屬屬性，例如 aws_access_key_id。

執行配方

此範例使用重複中的配方。

執行配方

1. 使用下列程式碼建立名為 `default.rb` 的檔案，並將它儲存到技術指南的 `recipes` 資料夾。

```
directory "/srv/www/shared" do
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end
```

2. 執行 `kitchen converge` 以執行配方。請注意，由於啟動和初始化 Amazon EC2 執行個體所需的時間，此命令的完成時間會比先前的範例更長。
3. 前往 Amazon EC2 主控台，選取美國西部（加利佛尼亞北部）區域，然後按一下導覽窗格中的執行個體。您會在清單中看到新建立的執行個體。
4. 執行 `kitchen login` 登入執行個體，就像您對在 VirtualBox 中執行的執行個體做的那樣。您會在 `/srv` 下看到新建立的目錄。您也可以使用您最愛的 SSH 用戶端連線到執行個體。

後續步驟

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本章會引導您了解如何實作 Chef 技術指南的基本概念，但仍有許多等您發掘：

- 此範例向您示範了如何使用一些較常用的資源，但還有很多等您發掘。

  在涵蓋的資源中，範例只使用了一些可用的屬性和動作。如需完整參考，請參閱關於資源和提供者。
- 這些範例只使用了核心技術指南元素：`recipes`、`attributes`、`files` 和 `templates`。

技術指南也可以包含各種其他元素，例如 `libraries`、`definitions` 和 `specs`。如需詳細資訊，請參閱 Chef 文件。

- 這些範例只將 Test Kitchen 用為啟動執行個體、執行配方及登入執行個體的便利方法。

  Test Kitchen 主要是一種測試平台，您可用來對您的配方執行各種測試。如果還未嘗試過，請試試其他的 Test Kitchen 演練，了解其測試功能。
- 實作 Stacks AWS OpsWorks 的技術指南 提供一些更進階的範例，並說明如何實作 Stacks AWS OpsWorks 的技術指南。

實作 Stacks AWS OpsWorks 的技術指南

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

技術指南基本概念 介紹技術指南和配方。該區段中的範例設計簡單，可用於支援 Chef 的任何執行個體，包括 AWS OpsWorks Stacks 執行個體。若要為 AWS OpsWorks Stacks 實作更複雜的技術指南，您通常需要充分利用 AWS OpsWorks Stacks 環境，這與標準 Chef 有許多不同之處。

本主題說明實作 Stacks AWS OpsWorks 執行個體配方的基本概念。

> **Note**
>
> 如果您不熟悉如何實作技術指南，則應該開始使用技術指南基本概念。

主題

- 在 Stacks Linux AWS OpsWorks 執行個體上執行配方
- 在 Windows 執行個體上執行配方
- 執行 Windows PowerShell 指令碼
- 在 Vagrant 上模擬堆疊組態和部署屬性
- 使用堆疊組態和部署屬性值
- 在 Linux 執行個體上使用外部技術指南：Berkshelf

- 使用適用於 Ruby 的 SDK：從 Amazon S3 下載檔案
- 安裝 Windows 軟體
- 覆寫內建屬性
- 覆寫內建範本

在 Stacks Linux AWS OpsWorks 執行個體上執行配方

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Test Kitchen 和 Vagrant 提供簡單且有效率的方式來實作技術指南，但為了驗證技術指南的配方在生產環境中正確執行，您必須在 AWS OpsWorks Stacks 執行個體上執行它們。本主題說明如何在 AWS OpsWorks Stacks Linux 執行個體上安裝自訂技術指南，以及執行簡單配方。本主題也提供一些有效修復配方錯誤的秘訣。

如需如何在 Windows 執行個體上執行配方的描述，請參閱在 Windows 執行個體上執行配方。

主題

- 建立和執行配方
- 自動執行配方
- 故障診斷和修復手冊

建立和執行配方

首先，您需要建立堆疊。以下簡短地彙總如何針對此範例建立堆疊。如需詳細資訊，請參閱建立新的堆疊。

建立 堆疊

1. 開啟 AWS OpsWorks Stacks 主控台，然後按一下 Add Stack (新增堆疊)。
2. 指定下列設定，並接受其他設定的預設值，然後按一下 Add Stack (新增堆疊)。
   - 名稱 – OpsTest

- 預設 SSH 金鑰 – Amazon EC2 金鑰對

如果您需要建立 Amazon EC2 金鑰對，請參閱 Amazon EC2 金鑰對。請注意，金鑰對必須屬於與執行個體相同的 AWS 區域。此範例使用預設的美國西部 （奧勒岡） 區域。

3. 按一下 Add a layer (新增 layer)，並新增自訂 layer 至具有下列設定的堆疊。

- 名稱 – OpsTest
- 簡短名稱：opstest

任何 layer 類型都會實際作用於 Linux 堆疊，但此範例不需要其他 layer 類型所安裝的任何套件，因此自訂 layer 是最簡單的方法。

4. 新增全年無休執行個體 (具有預設設定) 至 layer，以及啟動它。

當執行個體啟動時，通常需要幾分鐘的時間，您可以建立技術指南。此範例將使用條件式邏輯中稍微修改過的配方版本，這會建立其名稱取決於平台的資料目錄。

設定技術指南

1. 在 opsworks_cookbooks 內建立並導覽至名為 opstest 的目錄。
2. 使用下列內容建立 metadata.rb 檔案，並將它儲存至 opstest。

```
name "opstest"
version "0.1.0"
```

3. 在 recipes 內建立 opstest 目錄。
4. 使用下列配方建立 default.rb 檔案，並將它儲存至 recipes 目錄。

```
Chef::Log.info("******Creating a data directory.******")

data_dir = value_for_platform(
  "centos" => { "default" => "/srv/www/shared" },
  "ubuntu" => { "default" => "/srv/www/data" },
  "default" => "/srv/www/config"
)

directory data_dir do
```

```
  mode 0755
  owner 'root'
  group 'root'
  recursive true
  action :create
end
```

請注意，配方會記錄訊息，但做法是呼叫 `Chef::Log.info`。您在此範例中未使用 Test Kitchen，因此 `log`方法不太有用。 會將訊息`Chef::Log.info`放入 Chef 日誌中，您可以在 Chef 執行完成後讀取該日誌。 AWS OpsWorks Stacks 提供檢視這些日誌的簡單方法，如下所述。

> **Note**
>
> Chef 日誌通常包含許多例行作業和相對無趣的資訊。括住訊息文字的 '*' 字元可讓您更輕鬆地找到它們。

5. 建立 `opsworks_cookbooks` 的 `.zip` 存檔。若要在 AWS OpsWorks Stacks 執行個體上安裝技術指南，您必須將其存放在儲存庫中，並提供 Stacks AWS OpsWorks 將技術指南下載至執行個體所需的資訊。您可以將技術指南存放至任何數個支援的儲存庫類型中。此範例會將包含技術指南的封存檔案存放在 Amazon S3 儲存貯體中。如需技術指南儲存庫的詳細資訊，請參閱技術指南儲存庫。

   > **Note**
   >
   > 為求簡化，此範例只會封存整個 `opsworks_cookbooks` 目錄。不過，這表示 Stacks AWS OpsWorks 會將 中的所有技術指南下載`opsworks_cookbooks`到執行個體，即使您只使用其中一個技術指南。若只要安裝範例技術指南，請建立另一個父目錄，並將 `opstest` 移至該目錄。然後，建立父目錄的 `.zip` 封存，並使用它，而非 `opsworks_cookbooks.zip`。
   >
   > 傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

6. 將封存上傳至 Amazon S3 儲存貯體，將封存設為公有，並記錄封存的 URL。

您現在可以安裝技術指南，並執行配方。

執行配方

1. 編輯堆疊以啟用自訂技術指南，然後指定下列設定。

   - 儲存庫類型 – S3 封存
   - 儲存庫 URL – 您先前記錄的技術指南封存 URL

   針對其他設定使用預設值，然後按一下 Save (儲存) 以更新堆疊組態。
2. 執行更新自訂技術指南堆疊命令，以在堆疊執行個體上安裝最新版的自訂技術指南。如果存在舊版的技術指南，則此命令會予以覆寫。
3. 透過使用配方執行執行配方堆疊命令以執行設定為 的配方**opstest::default**。此命令會啟動 Chef 執行，內含包含 `opstest::default` 的回合清單。

成功執行配方之後，您就可以驗證配方。

驗證 opstest

1. 第一步是檢查 Chef 日誌。按一下 opstest1 執行個體日誌欄中的顯示以顯示日誌。向下捲動，您會在接近底端看到您的日誌訊息。

```
...
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
attributes/customize.rb in the cache.
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
metadata.rb in the cache.
[2014-07-31T17:01:46+00:00] INFO: ******Creating a data directory.******
[2014-07-31T17:01:46+00:00] INFO: Processing template[/etc/hosts] action create
 (opsworks_stack_state_sync::hosts line 3)
...
```

2. 使用 SSH 登入執行個體，並列出 `/srv/www/` 的內容。

如果您已遵循所有步驟，則會看到 `/srv/www/config`，而不是您預期的 `/srv/www/shared` 目錄。下節提供一些快速修復這類錯誤的指導方針。

自動執行配方

Execute Recipes (執行配方) 命令是一種輕鬆測試自訂配方的方法，這是在其中大部分範例中使用它的原因。不過，實際上您通常會在執行個體生命週期中的標準點執行配方，例如在執行個體完成開機後或部署應用程式時。 AWS OpsWorks Stacks 透過支援每一層的一組生命週期事件，簡化執行個體上執行的配方：設定、設定、部署、取消部署和關機。您可以將配方指派給適當的生命週期事件，讓 AWS OpsWorks Stacks 在 layer 的執行個體上自動執行配方。

您通常會在執行個體完成開機時立即建立目錄，這對應至安裝事件。以下顯示如何使用您稍早在範例中建立的相同堆疊，在安裝時執行範例配方。您可以針對其他事件使用相同的程序。

在安裝時自動執行配方

1. 選擇導覽窗格中的 Layers (Layer)，然後選擇 OpsTest layer 之 Recipes (配方) 連結旁的鉛筆圖示。
2. 將 **`opstest::default`** 新增至 layer 的 Setup (安裝) 配方，並按一下 + 將它新增至 layer，然後選擇 Save (儲存) 儲存組態。
3. 選擇 Instances (執行個體)，並將另一個執行個體新增至 layer，然後啟動它。

   執行個體應該命名為 `opstest2`。開機完成後， AWS OpsWorks Stacks 會執行 `opstest::default`。
4. 在 `opstest2` 執行個體上線之後，請驗證 `/srv/www/shared` 已存在。

> **Note**
>
> 如果您已將配方指派給安裝、設定或部署事件，則也會手動執行它們，方法是使用堆疊命令 (安裝和設定) 或部署命令 (部署) 來觸發事件。請注意，如果您有多個指派給事件的配方，則這些命令會執行所有配方。

故障診斷和修復手冊

如果您未取得預期的結果，或您的配方甚至未成功執行，則故障診斷通常會從檢查 Chef 日誌開始。它包含執行的詳細描述，而且包括您配方中的任何內嵌日誌訊息。如果您的配方失敗，則日誌特別有用。發生該情況時，Chef 會記錄錯誤 (包括堆疊追蹤)。

如果配方成功 (如此範例所示)，則 Chef 日誌通常不怎麼有幫助。在這種情況下，只要仔細查看配方 (特別是前幾行)，就可以找出問題所在：

```
Chef::Log.info("******Creating a data directory.******")

data_dir = value_for_platform(
  "centos" => { "default" => "/srv/www/shared" },
  "ubuntu" => { "default" => "/srv/www/data" },
  "default" => "/srv/www/config"
)
...
```

當您在 Vagrant 上測試配方時，CentOS 是 Amazon Linux 的適用備用方法，但現在您將在實際 Amazon Linux 執行個體上執行。Amazon Linux 的平台值是 `amazon`，但未包含在 `value_for_platform` 呼叫中，因此配方會建立 `/srv/www/config`。如需故障診斷的詳細資訊，請參閱偵錯和故障診斷指南。

現在您已找到問題，您需要更新配方並驗證修復。您可以返回原始來源檔案、更新 `default.rb`、上傳新的封存至 Amazon S3，以此類推。不過，該程序可能有點繁瑣且耗時。下列更快速的方式特別適用於簡單配方錯誤 (例如範例中的配方錯誤)：編輯執行個體上的配方。

編輯執行個體上的配方

1. 使用 SSH 登入執行個體，然後執行 `sudo su` 以提升您的權限。您需要有 root 權限才能存取技術指南目錄。
2. AWS OpsWorks Stacks 會將您的技術指南存放在 中`/opt/aws/opsworks/current/site-cookbooks`，因此請導覽至 `/opt/aws/opsworks/current/site-cookbooks/opstest/recipes`。

   > **Note**
   >
   > AWS OpsWorks Stacks 也會將技術指南的副本存放在 中`/opt/aws/opsworks/current/merged-cookbooks`。請不要編輯該技術指南。當您執行配方時， AWS OpsWorks Stacks 會將技術指南從 複製到 `.../site-cookbooks` `.../merged-cookbooks`，因此您在 中所做的任何變更`.../merged-cookbooks`都會遭到覆寫。

3. 在執行個體上使用文字編輯器來編輯 `default.rb`，並將 `centos` 取代為 `amazon`。您的配方現在看起來應該與下列類似。

   ```
   Chef::Log.info("******Creating a data directory.******")
   ```

```
data_dir = value_for_platform(
  "amazon" => { "default" => "/srv/www/shared" },
  "ubuntu" => { "default" => "/srv/www/data" },
  "default" => "/srv/www/config"
)
...
```

若要驗證修復，請重新執行 Execute Recipe (執行配方) 堆疊命令來執行配方。執行個體現在應該會有一個 `/srv/www/shared` 目錄。如果您需要進一步變更配方，則可以經常執行 Execute Recipe (執行配方)；每次執行命令時，並不需要停止並重新啟動執行個體。對配方正確運作感到滿意後，請不要忘記更新您來源技術指南中的程式碼。

> **Note**
>
> 如果您已將配方指派給生命週期事件，讓 AWS OpsWorks Stacks 自動執行它，您一律可以使用執行配方來重新執行配方。您也可以使用 Stacks AWS OpsWorks 主控台手動觸發適當的事件，以任意次數重新執行配方，而無需重新啟動執行個體。不過，此方式會執行事件的所有配方。提醒如下：
>
> - 使用堆疊命令來觸發安裝或設定事件。
> - 使用部署命令來觸發部署或解除部署事件。

在 Windows 執行個體上執行配方

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本主題基本上是在 Linux 執行個體上執行配方的縮短版本，顯示如何在 Windows 堆疊上執行配方。建議您先參閱在 Linux 執行個體上執行配方，因為它提供更多詳細討論，其中大部分都是與任一種作業系統有關。

如需如何在 Stacks Linux AWS OpsWorks 執行個體上執行配方的說明，請參閱 在 Linux 執行個體上執行配方。

主題

- 啟用 RDP 存取
- 建立和執行配方
- 自動執行配方

啟用 RDP 存取

開始之前，如果您尚未這麼做，則必須設定安全群組，其具有允許 RDP 存取您執行個體的傳入規則。您在建立堆疊時將會需要該群組。

當您在區域中建立第一個堆疊時， AWS OpsWorks Stacks 會建立一組安全群組。其中包括一個名為的名稱，例如 `AWS-OpsWorks-RDP-Server`， Stacks AWS OpsWorks 會連接到所有 Windows 執行個體以允許 RDP 存取。不過，此安全群組預設沒有任何規則，因此您必須新增傳入規則來允許 RDP 存取您的執行個體。

允許 RDP 存取

1. 開啟 Amazon EC2 主控台，將其設定為堆疊的區域，然後從導覽窗格中選擇安全群組。
2. 選擇 AWS-OpsWorks-RDP-Server，並選擇 Inbound (傳入) 標籤，然後選擇 Edit (編輯)。
3. 新增具有下列設定的規則：
   - 類型 – RDP
   - 來源 – 允許的來源 IP 地址。

     通常，您會允許來自您 IP 地址或指定 IP 地址範圍 (通常是公司的 IP 地址範圍) 的傳入 RDP 請求。

> **Note**
>
> 如稍後所述，您還必須編輯使用者許可來授權 RDP 存取一般使用者。

如需詳細資訊，請參閱使用 RDP 登入。

建立和執行配方

以下簡短地彙總如何針對此範例建立堆疊。如需詳細資訊，請參閱建立新的堆疊。

建立堆疊

1. 開啟 AWS OpsWorks Stacks 主控台，然後選擇 Add Stack (新增堆疊)。指定下列設定，並接受其他設定的預設值，然後選擇 Add Stack (新增堆疊)。

   - 名稱 – WindowsRecipeTest
   - 區域 – 美國西部 （ 奧勒岡 ）

     此範例適用於任何區域，但我們建議您使用美國西部 （ 奧勒岡 ） 進行教學課程。

   - 預設作業系統 – Microsoft Windows Server 2012 R2
2. 選擇 Add a layer (新增 layer)，並新增自訂 layer 至具有下列設定的堆疊。

   - 名稱 – RecipeTest
   - 簡短名稱 – recipetest
3. 新增全年無休執行個體 (具有預設設定) 至 RecipeTest layer，以及啟動它。

   AWS OpsWorks Stacks 會自動指派給`AWS-OpsWorks-RDP-Server`此執行個體，允許授權使用者登入執行個體。
4. 選擇 Permissions (許可)，並選擇 Edit (編輯)，然後選擇 SSH/RDP 和 sudo/admin。除了 `AWS-OpsWorks-RDP-Server` 安全群組之外，一般使用者還需要有此授權，才能登入執行個體。

> **Note**
>
> 您也可以登入為管理員，但需要不同的程序。如需詳細資訊，請參閱使用 RDP 登入。

當執行個體啟動時，通常需要幾分鐘的時間，您可以建立技術指南。此範例的配方會建立資料目錄，而且基本上是範例 3：建立目錄 (針對 Windows 所修改) 中的配方。

> **Note**
>
> 實作 Stacks Windows AWS OpsWorks 執行個體的技術指南時，您使用的目錄結構與實作 Stacks Linux AWS OpsWorks 執行個體的技術指南略有不同。如需詳細資訊，請參閱技術指南儲存庫。

設定技術指南

1. 建立並導覽至名為 windowstest 的目錄。
2. 使用下列內容建立 metadata.rb 檔案，並將它儲存至 windowstest。

```
name "windowstest"
version "0.1.0"
```

3. 在 recipes 內建立 windowstest 目錄。
4. 使用下列配方建立 default.rb 檔案，並將它儲存至 recipes 目錄。

```
Chef::Log.info("******Creating a data directory.******")

directory 'C:\data' do
  rights :full_control, 'instance_name\username'
  inherits false
  action :create
end
```

將 *username* 取代為您的使用者名稱。

5. 將技術指南放在儲存庫中。

若要在 Stacks AWS OpsWorks 執行個體上安裝技術指南，您必須將其存放在儲存庫中，並向 AWS OpsWorks Stacks 提供將技術指南下載至執行個體所需的資訊。您可以將 Windows 技術指南存放為 S3 儲存貯體或 Git 儲存庫中的封存檔。此範例使用 S3 儲存貯體，因此您必須建立 windowstest 目錄的 .zip 封存。如需技術指南儲存庫的詳細資訊，請參閱技術指南儲存庫。

6. 將封存上傳至 S3 儲存貯體，並將封存設為公有，然後記錄封存的 URL。您也可以使用私有封存，但在此範例中公有封存就已足夠，而且使用起來更為簡單。

傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

您現在可以安裝技術指南，並執行配方。

執行配方

1. 編輯堆疊以啟用自訂技術指南，然後指定下列設定。

- 儲存庫類型 – S3 封存
- 儲存庫 URL – 您先前記錄的技術指南封存 URL

接受其他設定的預設值，然後選擇 Save (儲存) 以更新堆疊組態。

2. 執行更新自訂技術指南堆疊命令，以在堆疊執行個體上安裝最新版的自訂技術指南 (包括線上執行個體)。如果存在舊版的技術指南，則此命令會予以覆寫。
3. 更新自訂技術指南完成後，透過使用配方執行執行配方堆疊命令以執行設定為 的配方**windowstest::default**。此命令會啟動 Chef 執行，內含包含您配方的回合清單。

成功執行配方之後，您就可以驗證配方。

驗證 windowstest

1. 檢查 Chef 日誌。選擇 opstest1 執行個體日誌欄中的顯示以顯示日誌。向下捲動，您會在接近底端看到您的日誌訊息。

```
...
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
attributes/customize.rb in the cache.
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
metadata.rb in the cache.
[2014-07-31T17:01:46+00:00] INFO: ******Creating a data directory.******
[2014-07-31T17:01:46+00:00] INFO: Processing template[/etc/hosts] action create
 (opsworks_stack_state_sync::hosts line 3)
...
```

2. 選擇執行個體，在執行個體的動作欄中選擇 rdp，並請求具有適當過期時間的 RDP 密碼。複製 DNS 名稱、使用者名稱和密碼。您接著可以搭配使用該資訊與 RDP 用戶端 (例如 Windows 遠端桌面連線用戶端) 來登入執行個體，然後驗證 `c:\data` 已存在。如需詳細資訊，請參閱使用 RDP 登入。

> **Note**
>
> 如果您的配方未正常運作，請參閱故障診斷和修復手冊中的故障診斷秘訣；其中大部分也適用於 Windows 執行個體。如果您想要在執行個體上編輯配方來測試修正，請在 AWS OpsWorks `C:\chef\cookbooks`目錄中尋找您的技術指南，Stacks 會在其中安裝自訂技術指南。

自動執行配方

Execute Recipes (執行配方) 命令是一種輕鬆測試自訂配方的方法，這是在其中大部分範例中使用它的原因。不過，實際上您通常會在執行個體生命週期中的標準點執行配方，例如在執行個體完成開機後或部署應用程式時。 AWS OpsWorks Stacks 透過支援每一層的一組生命週期事件，簡化執行個體上執行的配方：設定、設定、部署、取消部署和關機。您可以將配方指派給適當的生命週期事件，讓 AWS OpsWorks Stacks 在 layer 的執行個體上自動執行配方。

您通常會在執行個體完成開機時立即建立目錄，這對應至安裝事件。以下顯示如何使用您稍早在範例中建立的相同堆疊，在安裝時執行範例配方。您可以針對其他事件使用相同的程序。

在安裝時自動執行配方

1. 選擇導覽窗格中的 Layers (Layer)，然後選擇 RecipeTest layer 之 Recipes (配方) 連結旁的鉛筆圖示。
2. 將 **`windowstest::default`** 新增至 layer 的 Setup (安裝) 配方，並選擇 + 將它新增至 layer，然後選擇 Save (儲存) 儲存組態。
3. 選擇 Instances (執行個體)，並將另一個執行個體新增至 layer，然後啟動它。

   執行個體應該命名為 `recipetest2`。開機完成後， AWS OpsWorks Stacks 會執行 `windowstest::default`。
4. 在 `recipetest2` 執行個體上線之後，請驗證 `c:\data` 已存在。

> **Note**
>
> 如果您已將配方指派給安裝、設定或部署事件，則也可以手動執行它們，方法是使用堆疊命令 (安裝和設定) 或部署命令 (部署) 來觸發事件。請注意，如果您有多個指派給事件的配方，則這些命令會執行所有配方。

執行 Windows PowerShell 指令碼

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 這些範例假設您已完成在 Windows 執行個體上執行配方範例。否則，您應該先執行該範例。具體而言，它說明如何啟用 RDP 存取您的執行個體。

讓配方在 Windows 執行個體上執行任務的一種方式，特別是沒有對應 Chef 資源的任務，就是讓配方執行 Windows PowerShell 指令碼。本節說明如何使用 Windows PowerShell 指令碼安裝 Windows 功能，以介紹基本概念。

`powershell_script` 資源會在執行個體上執行 Windows PowerShell cmdlet。下列範例使用 Install-WindowsFeature Cmdlet 以在執行個體上安裝 XPS 檢視器。

以下簡短地彙總如何針對此範例建立堆疊。如需詳細資訊，請參閱建立新的堆疊。

建立堆疊

1. 開啟 AWS OpsWorks Stacks 主控台，然後選擇 Add Stack (新增堆疊)。指定下列設定，並接受其他設定的預設值，然後按一下 Add Stack (新增堆疊)。
   - 名稱 – PowerShellTest
   - 區域 – 美國西部 （ 奧勒岡 ）

     此範例適用於任何區域，但我們建議您使用美國西部 （ 奧勒岡 ） 進行教學課程。
   - 預設作業系統 – Microsoft Windows Server 2012 R2
2. 選擇 Add a layer (新增 layer)，並新增自訂 layer 至具有下列設定的堆疊。
   - 名稱 – PowerShell
   - 簡短名稱 – powershell
3. 新增全年無休執行個體 (具有預設設定) 至 PowerShell layer，以及啟動它。

4. 選擇 Permissions (許可)，並選擇 Edit (編輯)，然後選取 SSH/RDP 和 sudo/admin。除了 AWS-OpsWorks-RDP-Server 安全群組之外，您還需要有此授權，才能以一般使用者身分登入執行個體。

當執行個體啟動時，通常需要幾分鐘的時間，您可以建立技術指南。此範例的配方會建立資料目錄，而且基本上是範例 3：建立目錄 (針對 Windows 所修改) 中的配方。

設定技術指南

1. 建立並導覽至名為 powershell 的目錄。
2. 使用下列內容建立 metadata.rb 檔案，並將它儲存至 windowstest。

```
name "powershell"
version "0.1.0"
```

3. 在 recipes 目錄內，建立 powershell 目錄。
4. 使用下列配方建立 default.rb 檔案，並將它儲存至 recipes 目錄。

```
Chef::Log.info("******Installing XPS.******")

powershell_script "Install XPS Viewer" do
  code <<-EOH
    Install-WindowsFeature XPS-Viewer
  EOH
  guard_interpreter :powershell_script
  not_if "(Get-WindowsFeature -Name XPS-Viewer).installed"
end
```

   - powershell_script 資源會執行 Cmdlet 來安裝 XPS 檢視器。

     此範例只會執行一個 Cmdlet，但 code 區塊可以包含任意數目的命令列。
   - guard_interpreter 屬性會指示 Chef 使用 64 位元版本的 Windows PowerShell。
   - not_if 保護屬性可確保 Chef 不會安裝已安裝的功能。
5. 建立 powershell 目錄的 .zip 存檔。
6. 將封存上傳至 Amazon S3 儲存貯體，將封存設為公有，並記錄封存的 URL。您也可以使用私有封存，但在此範例中公有封存就已足夠，而且使用起來更為簡單。

傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

您現在可以安裝技術指南，並執行配方。

執行配方

1. 編輯堆疊以啟用自訂技術指南，然後指定下列設定。

   - 儲存庫類型 – S3 封存
   - 儲存庫 URL – 您先前記錄的技術指南封存 URL

   接受其他設定的預設值，然後選擇 Save (儲存) 以更新堆疊組態。
2. 執行 Update Custom Cookbooks (更新自訂技術指南) 堆疊命令，以在堆疊執行個體上安裝最新版的自訂技術指南。
3. 更新自訂技術指南完成後，透過使用配方執行執行配方堆疊命令以執行設定為 的配方**powershell::default**。

> **Note**
>
> 此範例為了方便起見，使用執行配方，但您通常會讓 AWS OpsWorks Stacks 透過將配方指派給適當的生命週期事件來自動執行配方。您可以手動觸發事件來執行這類配方。您可以使用堆疊命令來觸發安裝和設定事件，以及使用部署命令來觸發部署和解除部署事件。

成功執行配方之後，您就可以驗證配方。

驗證 powershell 配方

1. 檢查 Chef 日誌。按一下 powershell1 執行個體日誌欄中的顯示以顯示日誌。向下捲動，您會在接近底端看到您的日誌訊息。

```
...
[2015-04-27T18:12:09+00:00] INFO: Storing updated cookbooks/powershell/metadata.rb
 in the cache.
[2015-04-27T18:12:09+00:00] INFO: ******Installing XPS.******
```

```
[2015-04-27T18:12:09+00:00] INFO: Processing powershell_script[Install XPS Viewer]
 action run (powershell::default line 3)
[2015-04-27T18:12:09+00:00] INFO: Processing powershell_script[Guard resource]
 action run (dynamically defined)
[2015-04-27T18:12:42+00:00] INFO: powershell_script[Install XPS Viewer] ran
 successfully
...
```

2. 使用 RDP 登入執行個體，並開啟 Start (開始) 選單。XPS 檢視器應該與 Windows Accessories (Windows 附屬應用程式) 一起列出。

在 Vagrant 上模擬堆疊組態和部署屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 本主題僅適用於 Linux 執行個體。Test Kitchen 尚未支援 Windows，因此您將在 Stacks 執行個體上執行所有 Windows AWS OpsWorks 範例。

AWS OpsWorks Stacks 會將堆疊組態和部署屬性新增至堆疊中每個生命週期事件每個執行個體的節點物件。這些屬性提供堆疊組態的快照，包括每 layer 和其線上執行個體的組態、每個已部署應用程式的組態，以此類推。由於這些屬性位於節點物件中，因此任何配方都可以存取它們；大多數 Stacks AWS OpsWorks 執行個體的配方都使用一或多個這些屬性。

在 Vagrant 方塊中執行的執行個體不是由 AWS OpsWorks Stacks 管理，因此其節點物件預設不會包含任何堆疊組態和部署屬性。不過，您可以將適合的屬性集新增至 Test Kitchen 環境。Test Kitchen 接著會將屬性新增至執行個體的節點物件，而您的配方可以像在 Stacks AWS OpsWorks 執行個體上一樣存取屬性。

本主題顯示如何取得適合堆疊組態和部署屬性的複本、在執行個體上安裝屬性，以及存取它們。

> **Note**
>
> 如果您使用 Test Kitchen 在配方上執行測試，則 fauxhai 提供替代方法來模擬堆疊組態和部署 JSON。

設定技術指南

1. 建立並導覽至名為 `printjson` 的 `opsworks_cookbooks` 子目錄。
2. 初始化並設定 Test Kitchen，如範例 1：安裝套件中所述。
3. 將兩個子目錄新增至 `printjson`：`recipes` 和 `environments`。

您可以將具有適當定義的屬性檔案新增至技術指南，來模擬堆疊組態和部署屬性，但更佳的方式是使用 Test Kitchen 環境。有兩種基本方式：

- 將屬性定義新增至 `.kitchen.yml`。

  如果您只有幾個屬性，則此方式最為有用。如需詳細資訊，請參閱 kitchen.yml。
- 在環境檔案中定義屬性，並參考 `.kitchen.yml` 中的檔案。

  此方式通常適用於堆疊組態和部署屬性，因為環境檔案已為 JSON 格式。您可以從適當的 Stacks AWS OpsWorks 執行個體取得 JSON 格式的屬性副本，並直接貼上。所有範例都會使用環境檔案。

建立您技術指南的堆疊組態和部署屬性的最簡單方法是建立已適當設定的堆疊，以及從執行個體複製產生的屬性且其格式為 JSON。為了讓 Test Kitchen 環境易於管理，您可以接著編輯該 JSON 檔案，使其只具有您配方所需的屬性。本章中的範例根據Chef 11 Linux 堆疊入門 中的堆疊，此簡單 PHP 應用程式伺服器堆疊具有一個負載平衡器、多部 PHP 應用程式伺服器和一個 MySQL 資料庫伺服器。

建立堆疊組態和部署 JSON

1. 如Chef 11 Linux 堆疊入門 中所述，建立 MyStack (包括部署 SimplePHPApp)。如果您願意，可以省略在 中呼叫 的第二個 PHP App Server 執行個體步驟 4：橫向擴展 MyStack；範例不會使用這些屬性。
2. 如果您尚未這麼做，則請啟動 `php-app1` 執行個體，然後使用 SSH 登入。
3. 在終端機視窗中，執行下列 agent cli 命令：

```
sudo opsworks-agent-cli get_json
```

此命令會將執行個體的最新堆疊組態和部署屬性以 JSON 格式列印到終端機視窗。

4. 將 JSON 複製至 `.json` 檔案，並將它儲存至工作站上的方便位置。詳細資訊取決於 SSH 用戶端。例如，如果您在 Windows 上使用 PuTTY，則可以執行 `Copy All to Clipboard` 命令，以將終端機視窗中的所有文字都複製至 Windows 剪貼簿。您接著可以將內容貼入 `.json` 檔案，並編輯檔案以移除多餘的文字。
5. 視需要編輯 MyStack JSON。堆疊組態和部署屬性有很多，而且技術指南通常只會使用其中的一小部分。為了讓環境檔案易於管理，您可以編輯 JSON，讓它保留原始結構，但只包含您技術指南實際使用的屬性。

   此範例使用大量編輯過的 MyStack JSON 版本，其中僅包含兩個 `['opsworks']['stack']` 屬性：`['id]` 和 `['name']`。建立編輯過的 MyStack JSON 版本，看起來與下列類似：

```
{
  "opsworks": {
    "stack": {
      "name": "MyStack",
      "id": "42dfd151-6766-4f1c-9940-ba79e5220b58",
    },
  },
}
```

若要讓此 JSON 進入執行個體的節點物件，您需要將它新增至 Test Kitchen 環境。

將堆疊組態和部署屬性新增至 Test Kitchen 環境

1. 使用下列內容建立名為 `test.json` 的環境檔案，並將它儲存至技術指南的 `environments` 資料夾。

```
{
  "default_attributes": {
    "opsworks" : {
      "stack" : {
        "name" : "MyStack",
        "id" : "42dfd151-6766-4f1c-9940-ba79e5220b58"
      }
    }
  },
```

```
  "chef_type" : "environment",
  "json_class" : "Chef::Environment"
}
```

環境檔案具有下列元素：

- default_attributes – JSON 格式的預設屬性。

  這些屬性會新增至具有 default 屬性類型的節點物件，這是所有堆疊組態和部署 JSON 屬性所使用的類型。此範例使用先前顯示且編輯過的堆疊組態和部署 JSON 版本。
- chef_type – 將此元素設定為 environment。
- json_class – 將此元素設定為 Chef::Environment。

2. 編輯 .kitchen.yml 來定義 Test Kitchen 環境，如下所示。

```
---
driver:
  name: vagrant

provisioner:
  name: chef_solo
  environments_path: ./environments

platforms:
  - name: ubuntu-12.04

suites:
  - name: printjson
    provisioner:
      solo_rb:
        environment: test
    run_list:
      - recipe[printjson::default]
    attributes:
```

您可以將下列元素新增至 kitchen init 所建立的預設 .kitchen.yml，來定義環境。

provisioner

新增下列元素。

- name – 將此元素設定為 chef_solo。

若要更緊密地複寫 AWS OpsWorks Stacks 環境，您可以使用 Chef 用戶端本機模式，而不是 Chef 獨佔。本機模式是一個 Chef 用戶端選項，使用在執行個體上本機執行的輕量版 Chef 伺服器 (Chef Zero)，而非遠端伺服器。這可讓您的配方使用 Chef 伺服器功能，例如未連線至遠端伺服器的搜尋或資料包。

- `environments_path` – 技術指南子目錄，其中包含環境檔案，在此範例中`./environments`為 。

suites:provisioner

新增 `solo_rb` 元素，但其 `environment` 元素設定為環境檔案名稱並去除 .json 副檔名。此範例將 `environment` 設定為 `test`。

3. 使用下列內容建立名為 `default.rb` 的配方檔案，並將它儲存至技術指南的 `recipes` 目錄。

```
log "Stack name: #{node['opsworks']['stack']['name']}"
log "Stack id: #{node['opsworks']['stack']['id']}"
```

此配方只會記錄您新增至環境的兩個堆疊組態和部署值。雖然配方是在 Virtual Box 本機執行，但您可以使用與配方在 Stacks AWS OpsWorks 執行個體上執行時相同的節點語法來參考這些屬性。

4. 執行 `kitchen converge`。您應該會看到與下列類似的日誌輸出。

```
...
Converging 2 resources
Recipe: printjson::default
  * log[Stack name: MyStack] action write[2014-07-01T23:14:09+00:00] INFO:
 Processing log[Stack name: MyStack] action write (printjson::default line 1)

[2014-07-01T23:14:09+00:00] INFO: Stack name: MyStack

  * log[Stack id: 42dfd151-6766-4f1c-9940-ba79e5220b58] action
 write[2014-07-01T23:14:09+00:00] INFO: Processing log[Stack id:
 42dfd151-6766-4f1c-9940-ba79e5220b58] action write (printjson::default line 2)

[2014-07-01T23:14:09+00:00] INFO: Stack id: 42dfd151-6766-4f1c-9940-ba79e5220b58

...
```

使用堆疊組態和部署屬性值

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

配方通常需要堆疊組態或已部署應用程式的相關資訊。例如，您可能需要有堆疊 IP 地址清單才能建立組態檔案，或需要有應用程式的部署目錄才能建立日誌目錄。Stack AWS OpsWorks s 不會將此資料儲存在中央伺服器上，而是為每個生命週期事件在每個執行個體的節點物件中安裝一組堆疊組態和部署屬性。這些屬性代表目前堆疊狀態 (包括已部署的應用程式)。配方接著可以從節點物件取得它們所需的資料。

> **Note**
>
> 應用程式有時需要節點物件中的資訊 (例如堆疊組態和部署屬性值)。不過，應用程式無法存取節點物件。若要將節點物件資料提供給應用程式，您可以實作配方，以從節點物件擷取所需的資訊，並將它以方便的格式放入檔案中。應用程式接著可以從檔案讀取資料。如需詳細資訊和範例，請參閱 傳遞資料到應用程式。

配方可以從節點物件取得堆疊組態和部署屬性值，如下所示。

- 直接使用屬性的完整名稱。

  您可以搭配使用此方式與任何 Linux 堆疊，但不能與 Windows 堆疊搭配使用。
- 使用 Chef 搜尋，可用來查詢節點物件的屬性值。

  您可以搭配使用此方式與 Windows 堆疊和 Chef 11.10 Linux 堆疊。

> **Note**
>
> 使用 Linux 堆疊，您可以使用代理程式 CLI 來取得執行個體的堆疊組態和部署屬性複本，且格式為 JSON。如需詳細資訊，請參閱在 Vagrant 上模擬堆疊組態和部署屬性。

主題

- 直接取得屬性值
- 使用 Chef 搜尋取得屬性值

直接取得屬性值

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此方式僅適用於 Linux 堆疊。

在 Vagrant 上模擬堆疊組態和部署屬性顯示如何使用節點語法取得堆疊組態和部署資料，以直接參考特定屬性。這有時是最佳方式。不過，許多屬性都定義在集合或清單中，而集合或清單的內容和名稱可能會因不同的堆疊以及特定堆疊的一段時間而不同。例如，`deploy` 屬性包含應用程式屬性清單，而應用程式屬性以應用程式的短名命名。此清單 (包括應用程式屬性名稱) 通常會因不同的堆疊甚至會因不同的部署而不同。

以清單或集合形式列舉屬性來取得所需的資料，通常更為有用，有時甚至是必要的。例如，假設您想要知道堆疊執行個體的公有 IP 地址。該資訊位於設定為雜湊表的 `['opsworks']['layers']` 屬性中，而雜湊表針對每個堆疊 layer 都包含一個元素，以 layer 的短名命名。每個 layer 元素都會設定為包含 layer 屬性的雜湊表，而其中一個是 `['instances']`。該元素接著會設定為另一個雜湊表，其中針對每個 layer 執行個體都包含一個屬性，以執行個體的短名命名。每個執行個體屬性都會設定為另一個雜湊表，其中包含執行個體屬性 (包括代表公有 IP 地址的 `['ip']`)。如果您無法視覺化此項目，則下列程序包括 JSON 格式的範例。

此範例顯示如何從堆疊 layer 的堆疊組態和部署 JSON 取得資料。

設定技術指南

1. 在 `opsworks_cookbooks` 內建立並導覽至名為 `listip` 的目錄。
2. 初始化並設定 Test Kitchen，如範例 1：安裝套件中所述。

3. 將兩個目錄新增至 `listip`：`recipes` 和 `environments`。
4. 建立編輯過的 MyStack 組態和部署屬性 JSON 版本，其中包含相關的屬性。它看起來應該與下列類似。

```
{
  "opsworks": {
    "layers": {
      "php-app": {
        "name": "PHP App Server",
        "id": "efd36017-ec42-4423-b655-53e4d3710652",
        "instances": {
          "php-app1": {
            "ip": "192.0.2.0"
          }
        }
      },
      "db-master": {
        "name": "MySQL",
        "id": "2d8e0b9a-0d29-43b7-8476-a9b2591a7251",
        "instances": {
          "db-master1": {
            "ip": "192.0.2.5"
          }
        }
      },
      "lb": {
        "name": "HAProxy",
        "id": "d5c4dda9-2888-4b22-b1ea-6d44c7841193",
        "instances": {
          "lb1": {
            "ip": "192.0.2.10"
          }
        }
      }
    }
  }
}
```

5. 建立名為 `test.json` 的環境檔案，並將範例 JSON 貼入 `default_attributes`，然後將檔案儲存至技術指南的 `environments` 資料夾。此檔案應該與下列類似 (為求簡潔，會以省略符號代表大部分的範例 JSON)。

```
{
  "default_attributes" : {
    "opsworks": {
      "layers": {
        ...
      }
    }
  },
  "chef_type" : "environment",
  "json_class" : "Chef::Environment"
}
```

6. 將 .kitchen.yml 中的文字取代為下列內容。

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero
  environments_path: ./environment

platforms:
  - name: ubuntu-12.04

suites:
  - name: listip
    provisioner:
      client_rb:
        environment: test
    run_list:
      - recipe[listip::default]
    attributes:
```

在設定技術指南之後，您可以使用下列配方來記錄 layer ID。

```
node['opsworks']['layers'].each do |layer, layerdata|
  log "#{layerdata['name']} : #{layerdata['id']}"
```

```
end
```

配方會列舉 ['opsworks']['layers'] 中的 layer，並記錄每 layer 的名稱和 ID。

執行 layer ID 記錄配方

1. 使用範例配方建立名為 default.rb 的檔案，並將它儲存至 recipes 目錄。
2. 執行 kitchen converge。

輸出的相關部分應該與下列類似。

```
Recipe: listip::default
  * log[PHP App Server : efd36017-ec42-4423-b655-53e4d3710652] action
 write[2014-07-17T22:56:19+00:00] INFO: Processing log[PHP App Server : efd36017-
ec42-4423-b655-53e4d3710652] action write (listip::default line 4)
[2014-07-17T22:56:19+00:00] INFO: PHP App Server : efd36017-ec42-4423-b655-53e4d3710652



  * log[MySQL : 2d8e0b9a-0d29-43b7-8476-a9b2591a7251] action
 write[2014-07-17T22:56:19+00:00] INFO: Processing log[MySQL : 2d8e0b9a-0d29-43b7-8476-
a9b2591a7251] action write (listip::default line 4)
[2014-07-17T22:56:19+00:00] INFO: MySQL : 2d8e0b9a-0d29-43b7-8476-a9b2591a7251


  * log[HAProxy : d5c4dda9-2888-4b22-b1ea-6d44c7841193] action
 write[2014-07-17T22:56:19+00:00] INFO: Processing log[HAProxy : d5c4dda9-2888-4b22-
b1ea-6d44c7841193] action write (listip::default line 4)
[2014-07-17T22:56:19+00:00] INFO: HAProxy : d5c4dda9-2888-4b22-b1ea-6d44c7841193
```

若要列出執行個體的 IP 地址，您將需要巢狀循環，如下所示。

```
node['opsworks']['layers'].each do |layer, layerdata|
  log "#{layerdata['name']} : #{layerdata['id']}"
  layerdata['instances'].each do |instance, instancedata|
    log "Public IP: #{instancedata['ip']}"
  end
end
```

內部迴圈會逐一查看每 layer 的執行個體，並記錄 IP 地址。

執行執行個體 IP 記錄配方

1. 將 `default.rb` 中的程式碼取代為範例配方。
2. 執行 `kitchen converge` 以執行配方。

輸出的相關部分應該與下列類似。

```
  * log[PHP App Server : efd36017-ec42-4423-b655-53e4d3710652] action
 write[2014-07-17T23:09:34+00:00] INFO: Processing log[PHP App Server : efd36017-
ec42-4423-b655-53e4d3710652] action write (listip::default line 2)
[2014-07-17T23:09:34+00:00] INFO: PHP App Server : efd36017-ec42-4423-b655-53e4d3710652



  * log[Public IP: 192.0.2.0] action write[2014-07-17T23:09:34+00:00] INFO: Processing
 log[Public IP: 192.0.2.0] action write (listip::default line 4)
[2014-07-17T23:09:34+00:00] INFO: Public IP: 192.0.2.0


  * log[MySQL : 2d8e0b9a-0d29-43b7-8476-a9b2591a7251] action
 write[2014-07-17T23:09:34+00:00] INFO: Processing log[MySQL : 2d8e0b9a-0d29-43b7-8476-
a9b2591a7251] action write (listip::default line 2)
[2014-07-17T23:09:34+00:00] INFO: MySQL : 2d8e0b9a-0d29-43b7-8476-a9b2591a7251


  * log[Public IP: 192.0.2.5] action write[2014-07-17T23:09:34+00:00] INFO: Processing
 log[Public IP: 192.0.2.5] action write (listip::default line 4)
[2014-07-17T23:09:34+00:00] INFO: Public IP: 192.0.2.5


  * log[HAProxy : d5c4dda9-2888-4b22-b1ea-6d44c7841193] action
 write[2014-07-17T23:09:34+00:00] INFO: Processing log[HAProxy : d5c4dda9-2888-4b22-
b1ea-6d44c7841193] action write (listip::default line 2)
[2014-07-17T23:09:34+00:00] INFO: HAProxy : d5c4dda9-2888-4b22-b1ea-6d44c7841193


  * log[Public IP: 192.0.2.10] action write[2014-07-17T23:09:34+00:00] INFO: Processing
 log[Public IP: 192.0.2.10] action write (listip::default line 4)
[2014-07-17T23:09:34+00:00] INFO: Public IP: 192.0.2.10
```

當您完成之後，請執行 `kitchen destroy`；下個主題會使用新的技術指南。

> **Note**
>
> 列舉堆疊組態和部署 JSON 集合的其中一個最常見原因，是取得特定已部署應用程式的資料 (例如其部署目錄)。如需範例，請參閱「部署配方」。

使用 Chef 搜尋取得屬性值

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此方式適用於 Windows 堆疊和 Chef 11.10 Linux 堆疊。

直接從節點物件取得堆疊組態和部署屬性值可能十分複雜，而且無法與 Windows 堆疊搭配使用。替代方式是使用 Chef 搜尋查詢感興趣的屬性。如果您熟悉 Chef 伺服器，您會發現 Chef 搜尋與 Stacks AWS OpsWorks 的運作方式略有不同。由於 AWS OpsWorks Stacks 在本機模式中使用 Chef-client，因此 Chef 搜尋取決於稱為 chef-zero 的 Chef 伺服器本機版本，因此搜尋會操作存放在執行個體節點物件本機的資料，而非遠端伺服器。

實際上，限制搜尋本機儲存的資料通常並不重要，因為 Stacks AWS OpsWorks 執行個體上的節點物件包含堆疊組態和部署屬性。如果不是配方通常從 Chef 伺服器取得並使用相同名稱的所有資料，則它們包含大部分的資料，因此您通常可以在 Stacks AWS OpsWorks 執行個體上使用為 Chef 伺服器撰寫的搜尋碼，無需修改。如需詳細資訊，請參閱使用 Chef 搜尋。

以下顯示搜尋查詢的基本結構：

```
result = search(:search_index, "key:pattern")
```

- 搜尋索引指定要套用查詢的屬性，以及決定要傳回的物件類型。

- 索引鍵指定屬性名稱。
- 模式指定您要擷取之屬性的值。

  您可以查詢特定屬性值，或使用萬用字元來查詢某範圍的值。
- 結果是滿足查詢的物件清單，而且各為包含多個相關屬性的雜湊表。

  例如，如果您使用 node 搜尋索引，則查詢會傳回執行個體物件清單，而每個滿足查詢的執行個體都各有一個清單。每個物件都是一個雜湊表，其中包含可定義執行個體組態的一組屬性 (例如主機名稱和 IP 地址)。

例如，下列查詢使用 node 搜尋索引，這是套用至堆疊執行個體 (或 Chef 術語的節點) 的標準 Chef 索引。它會搜尋主機名稱為 myhost 的執行個體。

```
result = search(:node, "hostname:myhost")
```

搜尋會傳回主機名稱為 myhost 的執行個體物件清單。如果您想要第一個執行個體的作業系統 (例如，以 `result[0][:os]` 表示)。如果查詢傳回多個物件，您可以列舉它們以擷取所需資訊。

如何在配方中使用搜尋的詳細資訊取決於您使用 Linux 還是 Windows 堆疊。下列主題提供這兩種堆疊類型的範例。

主題

- 在 Linux 堆疊上使用搜尋
- 在 Windows 堆疊上使用搜尋

在 Linux 堆疊上使用搜尋

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

此範例根據具有單一 PHP 應用程式伺服器的 Linux 堆疊。它使用 Chef 搜尋取得伺服器的公有 IP 地址，並將地址放入 /tmp 目錄的檔案中。它基本上會從節點物件擷取相同的資訊 (如直接取得屬性值 所述)，但程式碼更為簡單，而且不是取決於堆疊組態和部署屬性結構的詳細資訊。

以下簡短地彙總如何針對此範例建立堆疊。如需詳細資訊，請參閱建立新的堆疊。

> **Note**
>
> 如果您之前沒有在 AWS OpsWorks Stacks 執行個體上執行自訂配方，您應該先完成在 Linux 執行個體上執行配方範例。

建立堆疊

1. 開啟 AWS OpsWorks Stacks 主控台，然後按一下 Add Stack (新增堆疊)。
2. 指定下列設定，並接受其他設定的預設值，然後按一下 Add Stack (新增堆疊)。

   - 名稱 – SearchJSON
   - 預設 SSH 金鑰 – Amazon EC2 金鑰對

   如果您需要建立 Amazon EC2 金鑰對，請參閱 Amazon EC2 金鑰對。請注意，金鑰對必須屬於與執行個體相同的 AWS 區域。此範例使用美國西部 （ 奧勒岡 ） 區域。
3. 按一下新增圖層，並使用預設設定將 PHP App Server 圖層新增至堆疊。
4. 新增全年無休執行個體 (具有預設設定) 至 layer，以及啟動它。

設定技術指南

1. 在 `opsworks_cookbooks` 內建立並導覽至名為 `searchjson` 的目錄。
2. 使用下列內容建立 `metadata.rb` 檔案，並將它儲存至 `opstest`。

```
name "searchjson"
version "0.1.0"
```

3. 在 `recipes` 內建立 `searchjson` 目錄。
4. 使用下列配方建立 `default.rb` 檔案，並將它儲存至 `recipes` 目錄。

```
phpserver = search(:node, "layers:php-app").first
Chef::Log.info("**********The public IP address is: '#{phpserver[:ip]}'**********")

file "/tmp/ip_addresses" do
```

```
  content "#{phpserver[:ip]}"
  mode 0644
  action :create
end
```

Linux 堆疊僅支援 node 搜尋索引。配方會使用此索引來取得 php-app layer 中的執行個體清單。因為 layer 已知只有一個執行個體，所以配方只需要將第一個執行個體指派給 phpserver。如果 layer 具有多個執行個體，您可以列舉它們以擷取所需資訊。每個清單項目都是一個雜湊表，其中包含一組執行個體屬性。ip 屬性設定為執行個體的公有 IP 地址，因此您可以將後續配方程式碼中的該地址呈現為 phpserver[:ip]。

將訊息新增至 Chef 日誌之後，配方接著會使用 file 資源來建立名為 ip_addresses 的檔案。content 屬性設定為以字串呈現 phpserver[:ip]。Chef 建立 ip_addresses 時，會將該字串新增至檔案。

5. 建立 的.zip封存opsworks_cookbooks、上傳封存至 Amazon S3 儲存貯體、將封存設為公有，並記錄封存的 URL。如需技術指南儲存庫的詳細資訊，請參閱技術指南儲存庫。

   傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

您現在可以安裝技術指南，並執行配方。

執行配方

1. 編輯堆疊以啟用自訂技術指南，然後指定下列設定。

   - 儲存庫類型 – Http Archive
   - 儲存庫 URL – 您先前記錄的技術指南封存 URL

   針對其他設定使用預設值，然後按一下 Save (儲存) 以更新堆疊組態。

2. 編輯自訂 layer 組態，並指派給 searchjson::default layer 的安裝事件。如果執行個體啟動後或您明確觸發安裝事件， AWS OpsWorks Stacks 將執行配方。

3. 執行更新自訂技術指南堆疊命令，以在堆疊執行個體上安裝最新版的自訂技術指南儲存庫。如果存在舊版的儲存庫，則此命令會予以覆寫。

4. 執行 Setup (安裝) 堆疊命令來執行配方，這會在執行個體上觸發安裝事件，並執行 searchjson::default。保留 Running command setup page (執行命令安裝頁面) 的開啟狀態。

成功執行配方之後，您就可以驗證配方。

驗證 searchjson

1. 第一步是檢查 Chef 日誌中的最新安裝事件。在執行命令設定頁面上，按一下 php-app1 執行個體日誌欄中的顯示以顯示日誌。向下捲動以在接近中間的位置找到您的日誌訊息，這看起來與下列類似。

```
...
[2014-09-05T17:08:41+00:00] WARN: Previous
 bash[logdir_existence_and_restart_apache2]: ...
[2014-09-05T17:08:41+00:00] WARN: Current
 bash[logdir_existence_and_restart_apache2]: ...
[2014-09-05T17:08:41+00:00] INFO: **********The public IP address is:
 '192.0.2.0'**********
[2014-09-05T17:08:41+00:00] INFO: Processing directory[/etc/sysctl.d] action create
 (opsworks_initial_setup::sysctl line 1)
...
```

2. 使用 SSH 登入執行個體，並列出 /tmp 的內容，其中應該包括名為 ip_addresses 且包含 IP 地址的檔案。

在 Windows 堆疊上使用搜尋

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 提供兩個在 Windows 堆疊上使用搜尋的選項。

- node 搜尋索引，可用來查詢一組標準 Chef 屬性。

  如果您有具有使用 之搜尋碼的現有配方node，它們通常會在不修改的情況下在 AWS OpsWorks Stacks 堆疊上運作。
- 可用來查詢 AWS OpsWorks Stacks 特定屬性集的一組額外搜尋索引，以及一些標準屬性。

在 Windows AWS OpsWorks Stacks 上使用 Stacks 特定的搜尋索引中會討論這些索引。

建議使用 node 來擷取標準資訊 (例如主機名稱或 IP 地址)。該方式可讓您的配方與標準 Chef 實務一致。使用 AWS OpsWorks Stacks 搜尋索引來擷取 Stacks AWS OpsWorks 特有的資訊。

主題

- 在 Windows 堆疊上使用搜尋索引節點
- 在 Windows AWS OpsWorks Stacks 上使用 Stacks 特定的搜尋索引

在 Windows 堆疊上使用搜尋索引節點

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此範例假設您已完成在 Windows 執行個體上執行配方範例。否則，您應該先執行該範例。具體而言，它說明如何啟用 RDP 存取您的執行個體。

此範例根據具有單一自訂 layer 和一個執行個體的 Windows 堆疊。它搭配使用 Chef 搜尋與 node 搜尋索引來取得伺服器的公有 IP 地址，並將地址放入 `C:\tmp` 目錄的檔案中。以下簡短地彙總如何針對此範例建立堆疊。如需詳細資訊，請參閱建立新的堆疊。

建立堆疊

1. 開啟 AWS OpsWorks Stacks 主控台，然後選擇 Add Stack (新增堆疊)。
2. 指定下列設定，並接受其他設定的預設值，然後選擇 Add Stack (新增堆疊)。

   - 名稱 – NodeSearch
   - 區域 – 美國西部 （奧勒岡）

   此範例適用於任何區域，但我們建議您使用美國西部 （奧勒岡） 進行教學課程。

- 預設作業系統 – Microsoft Windows Server 2012 R2

3. 選擇 Add a layer (新增 layer)，並新增自訂 layer 至具有下列設定的堆疊。

   - 名稱 – IPTest
   - 簡短名稱 – iptest

4. 新增全年無休 t2.micro 執行個體 (具有預設設定) 至 IPTest layer，以及啟動它。它將會命名為 iptest1。

   AWS OpsWorks Stacks 會自動指派給`AWS-OpsWorks-RDP-Server`此執行個體，允許授權使用者登入執行個體。

5. 選擇 Permissions (許可)，並選擇 Edit (編輯)，然後選取 SSH/RDP 和 sudo/admin。除了 `AWS-OpsWorks-RDP-Server` 安全群組之外，一般使用者還需要有此授權，才能登入執行個體。

> **Note**
>
> 您也可以登入為管理員，但需要不同的程序。如需詳細資訊，請參閱使用 RDP 登入。

設定技術指南

1. 建立並導覽至名為 `nodesearch` 的目錄。
2. 使用下列內容建立 `metadata.rb` 檔案，並將它儲存至 `opstest`。

```
name "nodesearch"
version "0.1.0"
```

3. 在 `recipes` 內建立 `nodesearch` 目錄。
4. 使用下列配方建立 `default.rb` 檔案，並將它儲存至 `recipes` 目錄。

```
directory 'C:\tmp' do
  rights :full_control, 'Everyone'
  recursive true
  action :create
end

windowsserver = search(:node, "hostname:iptest*").first
```

```
Chef::Log.info("**********The public IP address is:
 '#{windowsserver[:ipaddress]}'**********")

file 'C:\tmp\addresses.txt' do
  content "#{windowsserver[:ipaddress]}"
  rights :full_control, 'Everyone'
  action :create
end
```

配方會執行下列動作：

1. 使用目錄資源來建立檔案的 `C:\tmp` 目錄。

   如需此資源的詳細資訊，請參閱範例 3：建立目錄。

2. 搭配使用 Chef 搜尋與 node 搜尋索引，以取得主機名稱開頭為 `iptest` 的節點 (執行個體) 清單。

   如果您使用預設主題 (這會將整數附加至 layer 的短名來建立主機名稱)，則此查詢將會傳回 IPTest layer 中的每個執行個體。在此範例中，layer 已知只有一個執行個體，因此配方只需要將第一個執行個體指派給 `windowsserver`。針對多個執行個體，您可以取得完整清單，然後列舉它們。

3. 將具有 IP 地址的訊息新增至此執行的 Chef 日誌。

   `windowsserver` 物件是 `ipaddress` 屬性設定為執行個體之公有 IP 地址的雜湊表，因此您可以將後續配方程式碼中的該地址呈現為 `windowsserver[:ipaddress]`。配方會將對應的字串插入至訊息，並將其新增至 Chef 日誌。

4. 使用 `file` 資源建立 IP 地址為 `C:\tmp\addresses.txt` 的檔案。

   資源的 `content` 屬性指定要新增至檔案的內容，在此情況下即公有 IP 地址。

5. 建立 `nodesearch` 的 `.zip` 封存、將封存上傳至 S3 儲存貯體、將封存設為公有，並記錄封存的 URL。

   傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

您現在可以安裝技術指南，並執行配方。

安裝技術指南並執行配方

1. 編輯堆疊以啟用自訂技術指南，然後指定下列設定。

   - 儲存庫類型 – S3 封存
   - 儲存庫 URL – 您先前記錄的技術指南封存 URL

   接受其他設定的預設值，然後選擇 Save (儲存) 以更新堆疊組態。
2. 執行更新自訂技術指南堆疊命令，以在堆疊執行個體上安裝最新版的自訂技術指南 (包括線上執行個體)。如果存在舊版的技術指南，則此命令會予以覆寫。
3. 更新自訂技術指南完成後，使用配方執行執行配方堆疊命令以執行設定為 的配方**nodesearch::default**。此命令會啟動 Chef 執行，內含包含您配方的回合清單。保留 execute_recipes 頁面的開啟狀態。

成功執行配方之後，您就可以驗證配方。

驗證 nodesearch

1. 檢查 Chef 日誌中的最新 execute_recipes 事件。在執行命令 execute_recipes 頁面上，選擇在 iptest1 執行個體的日誌欄中顯示 以顯示日誌。向下捲動以在接近底端的位置找到您的日誌訊息，這看起來與下列類似。

```
...
[2015-05-13T18:55:47+00:00] INFO: Storing updated cookbooks/nodesearch/recipes/
default.rb in the cache.
[2015-05-13T18:55:47+00:00] INFO: Storing updated cookbooks/nodesearch/metadata.rb
 in the cache.
[2015-05-13T18:55:47+00:00] INFO: **********The public IP address is:
 '192.0.0.1'**********
[2015-05-13T18:55:47+00:00] INFO: Processing directory[C:\tmp] action create
 (nodesearch::default line 1)
[2015-05-13T18:55:47+00:00] INFO: Processing file[C:\tmp\addresses.txt] action
 create (nodesearch::default line 10)
...
```

2. 使用 RDP 登入執行個體，並檢查 `C:\tmp\addresses.txt` 的內容。

在 Windows AWS OpsWorks Stacks 上使用 Stacks 特定的搜尋索引

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此範例假設您已完成在 Windows 執行個體上執行配方範例。否則，您應該先執行該範例。具體而言，它說明如何啟用 RDP 存取您的執行個體。

AWS OpsWorks 除了 之外， Stacks 還提供下列搜尋索引node：

- `aws_opsworks_stack` – 堆疊組態。
- `aws_opsworks_layer` – 堆疊的 layer 組態。
- `aws_opsworks_instance` – 堆疊的執行個體組態。
- `aws_opsworks_app` – 堆疊的應用程式組態。
- `aws_opsworks_user` – 堆疊的使用者組態。
- `aws_opsworks_rds_db_instance` – 已註冊 RDS 執行個體的連線資訊。

這些索引包含一些標準 Chef 屬性，但主要用於擷取 AWS OpsWorks Stacks 特定的屬性。例如，`aws_opsworks_instance` 包括 `status` 屬性，以提供執行個體的狀態 (例如 `online`)。

> **Note**
>
> 建議的做法是盡可能使用 node，在讓您的配方與標準 Chef 使用量一致。如需範例，請參閱「在 Windows 堆疊上使用搜尋索引節點」。

此範例說明如何使用 AWS OpsWorks Stacks 索引來擷取 AWS OpsWorks Stacks 特定屬性的值。它根據自訂 layer 具有一個執行個體的簡單 Windows 堆疊。它使用 Chef 搜尋來取得執行個體的 AWS OpsWorks Stacks ID，並將結果放入 Chef 日誌中。

以下簡短地彙總如何針對此範例建立堆疊。如需詳細資訊，請參閱建立新的堆疊。

建立堆疊

1. 開啟 AWS OpsWorks Stacks 主控台，然後選擇 + Stack (+ 堆疊)。指定下列設定，並接受其他設定的預設值，然後選擇 Add Stack (新增堆疊)。

   - 名稱 – IDSearch
   - 區域 – 美國西部 （ 奧勒岡 ）

     此範例適用於任何區域，但我們建議您使用美國西部 （ 奧勒岡 ） 進行教學課程。
   - 預設作業系統 – Microsoft Windows Server 2012 R2
2. 選擇 Add a layer (新增 layer)，並新增自訂 layer 至具有下列設定的堆疊。

   - 名稱 – IDCheck
   - 簡短名稱 – idcheck
3. 新增全年無休 t2.micro 執行個體 (具有預設設定) 至 IDCheck layer，以及啟動它。它將會命名為 iptest1。

   AWS OpsWorks Stacks 會自動指派給`AWS-OpsWorks-RDP-Server`此執行個體。 啟用 RDP 存取 說明如何將傳入規則新增至此安全群組，以允許授權使用者登入執行個體。
4. 選擇 Permissions (許可)，並選擇 Edit (編輯)，然後選擇 SSH/RDP 和 sudo/admin。除了 `AWS-OpsWorks-RDP-Server` 安全群組之外，一般使用者還需要有此授權，才能登入執行個體。

   > **Note**
   >
   > 您也可以登入為管理員，但需要不同的程序。如需詳細資訊，請參閱使用 RDP 登入。

設定技術指南

1. 建立並導覽至名為 `idcheck` 的目錄。
2. 使用下列內容建立 `metadata.rb` 檔案，並將它儲存至 `opstest`。

```
name "idcheck"
version "0.1.0"
```

3. 在 `idcheck` 內建立 `recipes` 目錄，並將 `default.rb` 檔案新增至包含下列配方的目錄。

```
windowsserver = search(:aws_opsworks_instance, "hostname:idcheck*").first
Chef::Log.info("**********The public IP address is:
 '#{windowsserver[:instance_id]}'**********")
```

配方使用 Chef 搜尋搭配aws_opsworks_instance搜尋索引來取得堆疊中每個執行個體的執行個體屬性，其主機名稱開頭為 idcheck。如果您使用預設主題 (這會將整數附加至 layer 的短名來建立主機名稱)，則此查詢將會傳回 IDCheck layer 中的每個執行個體。在此範例中，layer 已知只有一個執行個體，因此配方只需要將第一個執行個體指派給 windowsserver。針對多個執行個體，您可以取得完整清單，然後列舉它們。

配方會利用具有此主機名稱之堆疊中只有一個執行個體的事實，因此第一個結果就是正確的結果。如果您的堆疊具有多個執行個體，則搜尋其他屬性可能會傳回多個結果。如需執行個體屬性的清單，請參閱執行個體資料包 (aws_opsworks_instance)。

執行個體屬性基本上是雜湊資料表，且執行個體的 Stacks ID AWS OpsWorks 會指派給 instance_id 屬性，因此您可以將 ID 稱為 windowsserver[:instance_id]。配方會將對應的字串插入至訊息，並將其新增至 Chef 日誌。

4. 建立ipaddress技術指南的.zip封存、上傳封存至 Amazon S3 儲存貯體，並記錄封存的 URL。如需技術指南儲存庫的詳細資訊，請參閱技術指南儲存庫。

   傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

您現在可以安裝技術指南，並執行配方。

安裝技術指南並執行配方

1. 編輯堆疊以啟用自訂技術指南，然後指定下列設定。

   - 儲存庫類型 – S3 封存
   - 儲存庫 URL – 您先前記錄的技術指南封存 URL

   接受其他設定的預設值，然後選擇 Save (儲存) 以更新堆疊組態。

2. 執行更新自訂技術指南堆疊命令，以在堆疊執行個體上安裝最新版的自訂技術指南 (包括線上執行個體)。如果存在舊版的技術指南，則此命令會予以覆寫。

3. 更新自訂技術指南完成後，使用配方執行執行配方堆疊命令以執行設定為 的配方**idcheck::default**。此命令會啟動 Chef 執行，內含包含您配方的回合清單。保留 execute_recipes 頁面的開啟狀態。

成功執行配方之後，您可以檢查 Chef 日誌中的最新 execute_recipes 事件來驗證。在執行命令 execute_recipes 頁面上，選擇在 iptest1 執行個體的日誌欄中顯示 以顯示日誌。向下捲動以在接近底端的位置找到您的日誌訊息，這看起來與下列類似。

```
...
[2015-05-13T20:03:47+00:00] INFO: Storing updated cookbooks/nodesearch/recipes/
default.rb in the cache.
[2015-05-13T20:03:47+00:00] INFO: Storing updated cookbooks/nodesearch/metadata.rb in
 the cache.
[2015-05-13T20:03:47+00:00] INFO: **********The instance ID is: 'i-8703b570'**********
[2015-05-13T20:03:47+00:00] INFO: Chef Run complete in 0.312518 seconds
...
```

在 Linux 執行個體上使用外部技術指南：Berkshelf

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> Berkshelf 僅適用於 Chef 11.10 Linux 堆疊。

在您開始實作技術指南之前，請參閱 Chef 社群技術指南，其中包含 Chef 社群成員基於各種用途所建立的技術指南。其中許多技術指南可以與 AWS OpsWorks Stacks 搭配使用，無需修改，因此您可以針對某些任務利用它們，而不是自行實作所有程式碼。

若要在執行個體上使用外部技術指南，您需要安裝它並管理任何相依性的方法。偏好的方式是實作技術指南，而技術指南支援名為 Berkshelf 的相依性管理員。Berkshelf 適用於 Amazon EC2 執行個體，包括 AWS OpsWorks Stacks 執行個體，但也旨在與 Test Kitchen 和 Vagrant 搭配使用。不

過，Vagrant 上的用量與 Stacks AWS OpsWorks 略有不同，因此本主題包含兩個平台的範例。如需如何使用 Berkshelf 的詳細資訊，請參閱 Berkshelf。

主題

- 搭配使用 Berkshelf 與 Test Kitchen 和 Vagrant
- 搭配 Stacks AWS OpsWorks 使用 Berkshelf

搭配使用 Berkshelf 與 Test Kitchen 和 Vagrant

此範例顯示如何使用 Berkshelf 來安裝 getting-started 社群技術指南並執行其配方，以在執行個體的主目錄中安裝簡短文字檔。

安裝 Berkshelf 並初始化技術指南

1. 在您的工作站上，安裝 Berkshelf Gem，如下所示。

   ```
   gem install berkshelf
   ```

   視您的工作站而定，此命令可能需要 sudo，或者您也可以使用 Ruby 環境管理員，例如 RVM。若要驗證是否已成功安裝 Berkshelf，請執行 `berks --version`。

2. 本主題的技術指南命名為 external_cookbook。您可以使用 Berkshelf 建立初始化的技術指南，而非先前主題所採用的手動方式。若要這麼做，請導覽至 `opsworks_cookbooks` 目錄，並執行下列命令。

   ```
   berks cookbook external_cookbook
   ```

   此命令會建立 `external_cookbook` 目錄以及數種標準 Chef 和 Test Kitchen 子目錄 (包括 `recipes` 和 `test`)。此命令也會建立數種標準檔案的預設版本，包括下列項目：

   - `metadata.rb`
   - Vagrant、Test Kitchen 和 Berkshelf 的組態檔
   - `default.rb` 目錄中的空 `recipes` 配方

   > **Note**
   >
   > 您不需要執行 `kitchen init`；`berks cookbook` 命令會處理這些任務。

3. 執行 `kitchen converge`。新建立的技術指南目前不會執行任何有趣的動作，但會收斂。

> **Note**
>
> 您也可以使用 `berks init`，初始化現有技術指南來使用 Berkshelf。

若要使用 Berkshelf 管理技術指南的外部相依性，技術指南的根目錄必須包含 `Berksfile`，而此組態檔指定 Berkshelf 應該如何管理相依性。當您使用 `berks cookbook` 建立 `external_cookbook` 技術指南時，會建立具有下列內容的 `Berksfile`。

```
source "https://supermarket.chef.io"
metadata
```

此檔案的宣告如下：

- `source` – 技術指南來源的 URL。

  Berksfile 可以有任意數目的 `source` 宣告，而且每個宣告都指定相依技術指南的預設來源。如果您未明確指定技術指南的來源，則 Berkshelf 會查看同名技術指南的預設儲存庫。預設 Berksfile 包括可指定社群技術指南儲存庫的單一 `source` 屬性。該儲存庫包含 getting-started 技術指南，因此您可以將該列保持不變。
- `metadata` – 指示 Berkshelf 包含技術指南檔案中宣告的技術指南相依性`metadata.rb`。

  您也可以包括 `cookbook` 屬性以在 Berksfile 中宣告相依技術指南，稍後將會予以討論。

有兩種方法可以宣告技術指南相依性：

- 在 Berksfile 中包括 `cookbook` 宣告。

  這是 Stacks AWS OpsWorks 使用的方法。例如，若要指定此範例中所使用的 getting-started 技術指南，請在 Berksfile 中包括 `cookbook "getting-started"`。Berkshelf 接著會查看預設儲存庫中是否有該名稱的技術指南。您也可以使用 `cookbook` 明確指定技術指南來源，甚至指定特定版本。如需詳細資訊，請參閱 Berkshelf。
- 在 Berksfile 中包括 `metadata` 宣告，並在 `metadata.rb` 中宣告相依性。

此宣告指示 Berkshelf 包括 metadata.rb 中所宣告的技術指南相依性。例如，若要宣告 getting-started 相依性，請將 depends 'getting-started' 宣告新增至技術指南的 metadata.rb 檔案。

此範例使用第一個方法，以與 Stacks AWS OpsWorks 保持一致。

安裝 getting-started 技術指南

1. 編輯預設 Berksfile，以將 metadata 宣告取代為 cookbook 的 getting-started 宣告。內容應該與下列類似。

```
source "https://supermarket.chef.io"

cookbook 'getting-started'
```

2. 執行 berks install，以將 getting-started 技術指南從社群技術指南儲存庫下載至您工作站的 Berkshelf 目錄 (通常為 ~/.berkshelf)。此目錄通常只會稱為 Berkshelf。查看 Berkshelf 的 cookbooks 目錄，而您應該會看到 getting-started 技術指南的目錄，而其命名類似 getting-started-0.4.0。
3. 將 external_cookbook::default 回合清單中的 .kitchen.yml 取代為 getting-started::default。此範例不會從 external_cookbook 執行任何配方；它基本上就只是使用 getting-started 技術指南的方式。.kitchen.yml 檔案現在應該與下列類似。

```
---
driver:
  name: vagrant

provisioner:
  name: chef_solo

platforms:
  - name: ubuntu-12.04

suites:
  - name: default
    run_list:
      - recipe[getting-started::default]
```

```
    attributes:
```

4. 執行 `kitchen converge`，然後使用 `kitchen login` 登入執行個體。登入目錄應該包含名為 `chef-getting-started.txt` 且具有下列類似內容的檔案：

```
Welcome to Chef!

This is Chef version 11.12.8.
Running on ubuntu.
Version 12.04.
```

   Test Kitchen 會在執行個體的 `/tmp/kitchen/cookbooks` 目錄中安裝技術指南。如果您列出該目錄的內容，則會看到兩個技術指南：external_cookbook 和 getting-started。

5. 執行 `kitchen destroy` 關機執行個體。下一個範例使用 AWS OpsWorks Stacks 執行個體。

搭配 Stacks AWS OpsWorks 使用 Berkshelf

AWS OpsWorks Stacks 選擇性支援 Berkshelf for Chef 11.10 堆疊。若要搭配使用 Berkshelf 與您的堆疊，您必須執行下列動作。

- 啟用堆疊的 Berkshelf。

  AWS OpsWorks Stacks 接著會處理在堆疊執行個體上安裝 Berkshelf 的詳細資訊。

- 將 Berksfile 新增至技術指南儲存庫的根目錄。

  Berksfile 應該包含所有相依技術指南的 `source` 和 `cookbook` 宣告。

當 AWS OpsWorks Stacks 在執行個體上安裝您的自訂技術指南儲存庫時，它會使用 Berkshelf 來安裝儲存庫 Berksfile 中宣告的相依技術指南。如需詳細資訊，請參閱使用 Berkshelf。

此範例示範如何使用 Berkshelf 在 Stacks AWS OpsWorks 執行個體上安裝入門社群技術指南。它也會安裝用來建立 createfile 自訂技術指南的版本，以在指定的目錄中建立檔案。如需 createfile 運作方式的詳細資訊，請參閱從技術指南安裝檔案。

> **Note**
>
> 如果這是您第一次在 Stacks AWS OpsWorks 堆疊上安裝自訂技術指南，您應該先瀏覽在 Linux 執行個體上執行配方範例。

從建立堆疊開始，彙總如下。如需詳細資訊，請參閱建立新的堆疊。

建立堆疊

1. 開啟 AWS OpsWorks Stacks 主控台，然後按一下 Add Stack (新增堆疊)。
2. 指定下列設定，並接受其他設定的預設值，然後按一下 Add Stack (新增堆疊)。

   - 名稱 – BerksTest
   - 預設 SSH 金鑰 – Amazon EC2 金鑰對

   如果您需要建立 Amazon EC2 金鑰對，請參閱 Amazon EC2 金鑰對。請注意，金鑰對必須屬於與執行個體相同的 AWS 區域。此範例使用預設的美國西部 （奧勒岡） 區域。
3. 按一下 Add a layer (新增 layer)，並新增自訂 layer 至具有下列設定的堆疊。

   - 名稱 – BerksTest
   - 簡短名稱 – berkstest

   您可以針對此範例實際使用任何 layer 類型。不過，此範例不需要其他 layer 所安裝的任何套件，因此自訂 layer 是最簡單的方法。
4. 新增全年無休執行個體至 BerksTest layer (具有預設設定)，但尚未啟動它。

使用 AWS OpsWorks Stacks，技術指南必須位於具有標準目錄結構的遠端儲存庫中。然後，您將下載資訊提供給 AWS OpsWorks Stacks，其會在啟動時自動將儲存庫下載到每個堆疊的執行個體。為了簡化，此範例的儲存庫是公有 Amazon S3 封存，但 AWS OpsWorks Stacks 也支援 HTTP 封存、Git 儲存庫和 Subversion 儲存庫。如需詳細資訊，請參閱技術指南儲存庫。

傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

建立技術指南儲存庫

1. 在 opsworks_cookbooks 目錄中，建立名為 berkstest_cookbooks 的目錄。如果您想要，則可以在您方便找到的任何位置建立此目錄，因為您會將它上傳至儲存庫。
2. 使用下列內容，將名為 Berksfile 的檔案新增至 berkstest_cookbooks。

   ```
   source "https://supermarket.chef.io"

   cookbook 'getting-started'
   ```

   此檔案宣告 getting-started 技術指南相依性，並指示 Berkshelf 從社群技術指南網站下載它。
3. 將 createfile 目錄新增至包含下列內容的 berkstest_cookbooks。

   - 具有下列內容的 metadata.rb 檔案。

     ```
     name "createfile"
     version "0.1.0"
     ```

   - 包含具有下列內容之 files/default 檔案的 example_data.json 目錄。

     ```
     {
       "my_name" : "myname",
       "your_name" : "yourname",
       "a_number" : 42,
       "a_boolean" : true
     }
     ```

     檔案的名稱和內容為任意的。配方只需要將檔案複製至指定的位置。

   - 包含具有下列配方程式碼之 recipes 檔案的 default.rb 目錄。

     ```
     directory "/srv/www/shared" do
       mode 0755
       owner 'root'
       group 'root'
       recursive true
       action :create
     ```

```
end

cookbook_file "/srv/www/shared/example_data.json" do
  source "example_data.json"
  mode 0644
  action :create_if_missing
end
```

此配方會建立 `/srv/www/shared`，並將 `example_data.json` 從技術指南的 `files` 目錄複製至該目錄。

4. 建立 的`.zip`封存`berkstest_cookbooks`、上傳封存至 Amazon S3 儲存貯體、將封存設為公有，並記錄封存的 URL。

您現在可以安裝技術指南，並執行配方。

安裝技術指南並執行配方

1. 編輯堆疊以啟用自訂技術指南，然後指定下列設定。

   - 儲存庫類型 – Http Archive
   - 儲存庫 URL – 您先前記錄的技術指南封存 URL
   - 管理 Berkshelf – 是

   前兩個設定提供 AWS OpsWorks Stacks 將技術指南儲存庫下載到您的執行個體所需的資訊。最後一個設定會啟用 Berkshelf 支援，以將 getting-started 技術指南下載至執行個體。接受其他設定的預設值，然後按一下 Save (儲存) 以更新堆疊組態。

2. 編輯 BerksTest layer，以將下列配方新增至 layer 的安裝生命週期事件。

   - `getting-started::default`
   - `createfile::default`

3. 啟動實例。設定事件會在執行個體完成開機後發生。 AWS OpsWorks Stacks 接著會安裝技術指南儲存庫、使用 Berkshelf 下載入門技術指南，以及執行 layer 的設定和部署配方，包括 `getting-started::default`和 `createfile::default`。

4. 執行個體上線之後，請使用 SSH 登入。您應該會看到下列事項：

   - `/srv/www/shared` 應該包含 `example_data.json`。
   - `/root` 應該包含 `chef-getting-started.txt`。

AWS OpsWorks Stacks 會以根目錄的形式執行配方，因此入門會將 檔案安裝在 `/root`目錄中，而不是您的主目錄中。

使用適用於 Ruby 的 SDK：從 Amazon S3 下載檔案

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

使用 Chef 資源無法處理一些任務 (例如與 AWS 服務互動)。例如，有時偏好遠端存放檔案，並讓配方將它們下載至執行個體。您可以使用 remote_file 資源，以從遠端伺服器下載檔案。不過，如果您想要將檔案存放在 Amazon S3 儲存貯體中， 只有在 ACL 允許操作時`remote_file`，才能下載這些檔案。

配方可以使用適用於 Ruby 的 AWS SDK存取大部分的 AWS 服務。本主題說明如何使用適用於 Ruby 的 SDK 從 S3 儲存貯體下載檔案。

> **Note**
>
> 如需如何使用適用於 Ruby 的 AWS SDK處理加密和解密的詳細資訊，請參閱 AWS::S3::S3Object。傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

主題

- 在 Vagrant 執行個體上使用適用於 Ruby 的 SDK
- 在 Stacks Linux 執行個體上使用適用於 Ruby AWS OpsWorks 的 開發套件
- 在 Stacks Windows 執行個體上使用適用於 Ruby AWS OpsWorks 的 開發套件

在 Vagrant 執行個體上使用適用於 Ruby 的 SDK

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本主題說明在 Vagrant 執行個體上執行的配方如何使用 從 Amazon S3 適用於 Ruby 的 AWS SDK下載檔案。開始之前，您必須先擁有一組 AWS 登入資料：存取金鑰和私密存取金鑰，以允許配方存取 Amazon S3。

> **Important**
>
> 強烈建議您不要基於此目的使用 root 帳戶登入資料。反之，請使用適當的政策建立使用者，並將這些登入資料提供給配方。
>
> 請注意不要將登入資料—甚至是 IAM 使用者登入資料—放在可公開存取的位置，例如將包含登入資料的檔案上傳至公有 GitHub 或 Bitbucket 儲存庫。這麼做會公開您的登入資料，而且可能會危害您帳戶的安全性。
>
> 在 EC2Amazon EC2 執行個體上執行的配方可以使用更好的方法 IAM 角色，如中所述在 Stacks Linux 執行個體上使用適用於 Ruby AWS OpsWorks 的 開發套件。
>
> 傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

如果您還沒有適當的 使用者，則可以建立一位使用者，如下所示。如需詳細資訊，請參閱什麼是 IAM。

> **Warning**
>
> IAM 使用者具有長期憑證，這會造成安全風險。為了協助降低此風險，建議您只為這些使用者提供執行任務所需的許可，並在不再需要這些使用者時將其移除。

建立 IAM 使用者

1. 登入 AWS Management Console 並開啟位於 https：//https://console.aws.amazon.com/iam/ 的 IAM 主控台。

2. 在導覽窗格中，選擇使用者，並視需要選擇新增使用者以建立新的管理使用者。
3. 在設定許可頁面上，選擇直接連接政策。
4. **S3** 在許可政策搜尋方塊中輸入 以顯示 Amazon S3 政策。

   選擇 AmazonS3ReadOnlyAccess。如果您想要，則可以指定政策來授予更廣泛的許可 (例如 AmazonS3FullAccess)，但標準實務是僅授予這些所需的許可。在此情況下，配方僅會下載檔案，因此唯讀存取就已足夠。
5. 選擇下一步。
6. 選擇建立使用者
7. 接著為您的使用者建立存取金鑰。如需建立存取金鑰的詳細資訊，請參閱IAM 使用者指南中的管理 IAM 使用者的存取金鑰。

您必須接著提供要下載的檔案。此範例假設您將名為 `myfile.txt` 的檔案放入名為 `cookbook_bucket` 的新建立 S3 儲存貯體中。

提供要下載的檔案

1. 建立名為 `myfile.txt` 且具有下列文字的檔案，並將它儲存至工作站上的方便位置。

   ```
   This is the file that you just downloaded from Amazon S3.
   ```
2. 在 Amazon S3 主控台cookbook_bucket上，在標準區域中建立名為 的儲存貯體，並上傳至myfile.txt儲存貯體。

設定技術指南，如下所示。

設定技術指南

1. 在 `opsworks_cookbooks` 內建立並導覽至名為 `s3bucket` 的目錄。
2. 初始化並設定 Test Kitchen，如範例 1：安裝套件中所述。
3. 將 `.kitchen.yml` 中的文字取代為下列內容。

   ```
   ---
   driver:
     name: vagrant
   ```

```
provisioner:
  name: chef_solo
  environments_path: ./environments

platforms:
  - name: ubuntu-14.04

suites:
  - name: s3bucket
    provisioner:
      solo_rb:
        environment: test
    run_list:
      - recipe[s3bucket::default]
    attributes:
```

4. 將兩個目錄新增至 s3bucket：recipes 和 environments。
5. test.json 使用下列default_attributes區段建立名為 的環境檔案，將 access_key和 secret_key值取代為您的使用者對應的金鑰。將檔案儲存至技術指南的 environments 資料夾。

```
{
  "default_attributes" : {
    "cookbooks_101" : {
      "access_key": "AKIAIOSFODNN7EXAMPLE",
      "secret_key" : "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
    }
  },
  "chef_type" : "environment",
  "json_class" : "Chef::Environment"
}
```

您有各種方法可以將登入資料提供給執行個體上所執行的配方。主要考量是限制意外公開金鑰以及危害您帳戶安全性的機會。因此，不建議在您的程式碼中使用明確金鑰值。此範例改為將金鑰值放入節點物件中，以使用節點語法而非公開常值來允許配方參考它們。您必須擁有 root 權限才能存取節點物件，這會限制可能公開金鑰的可能性。如需詳細資訊，請參閱管理 AWS 存取金鑰的最佳實務。

> **Note**
>
> 請注意，此範例使用巢狀屬性，而且 `cookbooks_101` 為第一個元素。如果節點物件中有其他 `access_key` 或 `secret_key` 屬性，則本實務可限制名稱衝突的機會。

下列配方會從 `myfile.text` 儲存貯體下載 `cookbook_bucket`。

```
gem_package "aws-sdk ~> 3" do
  action :install
end

ruby_block "download-object" do
  block do
    require 'aws-sdk'

    s3 = Aws::S3::Client.new(
          :access_key_id => "#{node['cookbooks_101']['access_key']}",
          :secret_access_key => "#{node['cookbooks_101']['secret_key']}")

    myfile = s3.bucket['cookbook_bucket'].objects['myfile.txt']
    Dir.chdir("/tmp")
    File.open("myfile.txt", "w") do |f|
      f.write(myfile.read)
      f.close
    end
  end
  action :run
end
```

配方的第一部分會安裝適用於 Ruby 的 SDK，這是 Gem 套件。gem_package 資源會安裝配方或其他應用程式所使用的 Gem。

> **Note**
>
> 您的執行個體通常會有兩個 Ruby 執行個體，這一般是不同的版本。其中一個是 Chef 用戶端所使用的專用執行個體。另一個是供執行個體上所執行的應用程式和配方使用。安裝 Gem 套件時，請務必了解這項差異，因為有兩個適用於安裝 Gem (gem_package 和 chef_gem) 的資

源。如果應用程式或配方使用 Gem 套件，則請使用 gem_package 安裝它。chef_gem 僅適用於 Chef 用戶端所使用的 Gem 套件。

配方的其餘部分是 ruby_block 資源，而此資源包含可下載檔案的 Ruby 程式碼。您可能認為配方是 Ruby 應用程式，因此可以直接將程式碼放入配方中。不過，Chef 執行會先編譯整個程式碼，再執行任何資源。如果您將範例程式碼直接放在配方中，則 Ruby 會嘗試先解決 `require 'aws-sdk'` 陳述式，再執行 gem_package 資源。由於尚未安裝適用於 Ruby 的 SDK，編譯將會失敗。

除非執行 ruby_block 資源，否則不會編譯該資源中的程式碼。在此範例中，資源會在ruby_block資源gem_package完成安裝適用於 Ruby 的 SDK 後執行，因此程式碼將成功執行。

ruby_block 中的程式碼會運作，如下所示。

1. 建立新的 Aws::S3 物件，以提供服務界面。

   存取金鑰和秘密金鑰的指定方式是參考節點物件中所存放的值。
2. 呼叫S3物件的bucket.objects關聯，傳回名為 的Aws::S3::Object物件myfile，代表 myfile.txt。
3. 使用 Dir.chdir 將工作目錄設定為 /tmp。
4. 開啟名為 myfile.txt 的檔案，並將 myfile 的內容寫入該檔案，然後關閉該檔案。

執行配方

1. 使用範例配方建立名為 default.rb 的檔案，並將它儲存至 recipes 目錄。
2. 執行 kitchen converge。
3. 執行 kitchen login 登入執行個體，然後執行 ls /tmp。您應該會看到 myfile.txt，以及數種 Test Kitchen 檔案和目錄。

```
vagrant@s3bucket-ubuntu-1204:~$ ls /tmp
install.sh  kitchen  myfile.txt  stderr
```

   您也可以執行 cat /tmp/myfile.txt，確認檔案的內容正確。

完成後，請執行 kitchen destroy 終止執行個體。

在 Stacks Linux 執行個體上使用適用於 Ruby AWS OpsWorks 的 開發套件

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本主題說明如何在 Stacks Linux 執行個體上使用適用於 Ruby AWS OpsWorks 的 SDK，從 Amazon S3 儲存貯體下載檔案。 AWS OpsWorks Stacks 會自動在每個 Linux 執行個體上安裝適用於 Ruby 的 SDK。不過，當您建立服務的用戶端物件時，必須提供一組適合的 AWS 登入資料 `AWS::S3.new` 或其他服務的對等項目。

傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

在 Vagrant 執行個體上使用適用於 Ruby 的 SDK顯示如何降低公開您登入資料的風險，方法是將登入資料存放在節點物件中，並參考配方程式碼中的屬性。當您在 Amazon EC2 執行個體上執行配方時，您有更好的選項：IAM 角色。

IAM 角色的運作方式與 IAM 使用者非常類似。它的已連接政策授予使用各種 AWS 服務的許可。不過，您可以將角色指派給 Amazon EC2 執行個體，而不是個人。該執行個體上執行的應用程式接著可以取得已連接政策所授予的許可。使用角色，登入資料絕不會出現在您的程式碼中，即使是間接也是一樣。本主題說明如何使用 IAM 角色在 Amazon EC2 執行個體在 Vagrant 執行個體上使用適用於 Ruby 的 SDK上從 執行配方。

您可以搭配執行此配方與使用 kitchen-ec2 驅動程式的 Test Kitchen，如範例 9：使用 Amazon EC2 執行個體中所述。不過，在 Amazon EC2 執行個體上安裝適用於 Ruby 的 SDK 有點複雜，而不是您需要為 AWS OpsWorks Stacks 關注的事項。所有 AWS OpsWorks Stacks Linux 執行個體預設都會安裝適用於 Ruby 的 SDK。為了簡化，範例因此使用 AWS OpsWorks Stacks 執行個體。

第一步是設定 IAM 角色。此範例採用最簡單的方法，即使用 AWS OpsWorks Stacks 在您建立第一個堆疊時建立的 Amazon EC2 角色。它命名為 `aws-opsworks-ec2-role`。不過， AWS OpsWorks Stacks 不會將政策連接至該角色，因此預設不會授予任何許可。

您必須將`AmazonS3ReadOnlyAccess`政策連接至`aws-opsworks-ec2-role`角色，才能授予適當的許可。如需如何將政策連接至角色的詳細資訊，請參閱《IAM 使用者指南》中的新增 IAM 身分許可（主控台）。

您可以在建立或更新堆疊時指定角色。設定具有自訂 layer 的堆疊，如在 Linux 執行個體上執行配方中所述，並有一項新增。在新增堆疊頁面上，確認預設 IAM 執行個體描述檔設定為 aws-opsworks-ec2-role。 AWS OpsWorks Stacks 接著會將該角色指派給堆疊的所有執行個體。

設定技術指南的程序與在 Linux 執行個體上執行配方所使用的程序類似。以下是簡短摘要；您應該參閱該範例以了解詳細資訊。

設定技術指南

1. 建立並導覽至名為 s3bucket_ops 的目錄。
2. 使用下列內容建立 metadata.rb 檔案，並將它儲存至 s3bucket_ops。

```
name "s3bucket_ops"
version "0.1.0"
```

3. 在 recipes 內建立 s3bucket_ops 目錄。
4. 使用下列配方建立 default.rb 檔案，並將它儲存至 recipes 目錄。

```
Chef::Log.info("******Downloading a file from Amazon S3.******")

ruby_block "download-object" do
  block do
    require 'aws-sdk'

    s3 = AWS::S3.new

    myfile = s3.buckets['cookbook_bucket'].objects['myfile.txt']
    Dir.chdir("/tmp")
    File.open("myfile.txt", "w") do |f|
      f.syswrite(myfile.read)
      f.close
    end
  end
  action :run
end
```

5. 建立 的.zip封存，s3bucket_ops並將封存上傳到 Amazon S3 儲存貯體。為求簡化，請將封存設為公有，然後記錄封存的 URL 以供日後使用。您也可以將技術指南存放在私有 Amazon S3 封存中，或其他幾種儲存庫類型。如需詳細資訊，請參閱技術指南儲存庫。

此配方與先前範例所使用的配方類似，但例外狀況如下。

- 由於 AWS OpsWorks Stacks 已安裝適用於 Ruby 的 SDK，因此已刪除`chef_gem`資源。
- 配方不會將任何登入資料傳遞給 `AWS::S3.new`。

  會根據執行個體的角色，將登入資料自動指派給應用程式。
- 配方使用 `Chef::Log.info` 將訊息新增至 Chef 日誌。

針對此範例建立堆疊，如下所示。您也可以使用現有 Windows 堆疊。只需要更新技術指南，如後面所述。

建立 堆疊

1. 開啟 AWS OpsWorks Stacks 主控台，然後按一下 Add Stack (新增堆疊)。
2. 指定下列設定，並接受其他設定的預設值，然後按一下 Add Stack (新增堆疊)。

   - 名稱 – RubySDK
   - 預設 SSH 金鑰 – Amazon EC2 金鑰對

   如果您需要建立 Amazon EC2 金鑰對，請參閱 Amazon EC2 金鑰對。請注意，金鑰對必須屬於與執行個體相同的 AWS 區域。此範例使用預設的美國西部 （奧勒岡） 區域。
3. 按一下 Add a layer (新增 layer)，並新增自訂 layer 至具有下列設定的堆疊。

   - 名稱 – S3Download
   - 簡短名稱 – s3download

   任何 layer 類型都會實際作用於 Linux 堆疊，但此範例不需要其他 layer 類型所安裝的任何套件，因此自訂 layer 是最簡單的方法。
4. 新增全年無休執行個體 (具有預設設定) 至 layer，以及啟動它。

您現在可以安裝並執行配方

執行配方

1. 編輯堆疊以啟用自訂技術指南，然後指定下列設定。

   - 儲存庫類型 – Http Archive

- 儲存庫 URL – 您先前記錄的技術指南的封存 URL。

針對其他設定使用預設值，然後按一下 Save (儲存) 以更新堆疊組態。

2. 執行更新自訂技術指南堆疊命令，以在堆疊執行個體上安裝最新版的自訂技術指南。如果存在舊版的技術指南，則此命令會予以覆寫。
3. 透過使用配方執行執行配方堆疊命令以執行設定為 的配方**s3bucket_ops::default**。此命令會啟動 Chef 執行，內含包含 `s3bucket_ops::default` 的回合清單。

> **Note**
>
> 您通常會讓 AWS OpsWorks Stacks 透過將配方指派給適當的生命週期事件來自動執行配方。您可以手動觸發事件來執行這類配方。您可以使用堆疊命令來觸發安裝和設定事件，以及使用部署命令來觸發部署和解除部署事件。

成功執行配方之後，您就可以驗證配方。

驗證 s3bucket_ops

1. 第一步是檢查 Chef 日誌。您的堆疊應該有一個名為 opstest1 的執行個體。在執行個體頁面上，按一下執行個體日誌欄中的顯示以顯示 Chef 日誌。向下捲動，並在接近底端發現您的日誌訊息。

```
...
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
attributes/customize.rb in the cache.
[2014-07-31T17:01:45+00:00] INFO: Storing updated cookbooks/opsworks_cleanup/
metadata.rb in the cache.
[2014-07-31T17:01:46+00:00] INFO: ******Downloading a file from Amazon S3.******
[2014-07-31T17:01:46+00:00] INFO: Processing template[/etc/hosts] action create
 (opsworks_stack_state_sync::hosts line 3)
...
```

2. 使用 SSH 登入執行個體，並列出 `/tmp` 的內容。

在 Stacks Windows 執行個體上使用適用於 Ruby AWS OpsWorks 的 開發套件

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此範例假設您已完成在 Windows 執行個體上執行配方範例。否則，您應該先執行該範例。具體而言，它說明如何啟用 RDP 存取您的執行個體。
> 傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

本主題說明如何在 AWS OpsWorks Stacks Windows 執行個體適用於 Ruby 的 AWS SDK上使用 ，從 S3 儲存貯體下載檔案。

如果 Ruby 應用程式需要存取 AWS 資源，您必須使用具有適當許可的一組 AWS 登入資料來提供它。對於配方，提供 AWS 登入資料的最佳選項是使用 AWS Identity and Access Management (IAM) 角色。IAM 角色的運作方式與具有連接政策的 IAM 使用者非常類似，該政策授予使用各種 AWS 服務的許可。不過，您可以將角色指派給 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體，而不是個人。該執行個體上執行的應用程式接著可以取得已連接政策所授予的許可。使用角色，登入資料絕不會出現在您的程式碼中，即使是間接也是一樣。

第一步是設定 IAM 角色。此範例採用最簡單的方法，即使用 AWS OpsWorks Stacks 在您建立第一個堆疊時建立的 Amazon EC2 角色。它命名為 `aws-opsworks-ec2-role`。不過， AWS OpsWorks Stacks 不會將政策連接至該角色，因此預設不會授予任何許可。

您必須將`AmazonS3ReadOnlyAccess`政策連接至`aws-opsworks-ec2-role`角色，才能授予適當的許可。如需如何將政策連接至角色的詳細資訊，請參閱《IAM 使用者指南》中的新增 IAM 身分許可（主控台）。

您可以在建立或更新堆疊時指定角色。設定具有自訂 layer 的堆疊，如在 Windows 執行個體上執行配方中所述，並有一項新增。在新增堆疊頁面上，確認預設 IAM 執行個體描述檔設定為 aws-opsworks-ec2-role。 AWS OpsWorks Stacks 接著會將該角色指派給堆疊的所有執行個體。

設定技術指南的程序與在 Linux 執行個體上執行配方所使用的程序類似。以下是簡短摘要；如需詳細資訊，請參閱該範例。

設定技術指南

1. 建立並導覽至名為 s3bucket_ops 的目錄。
2. 使用下列內容建立 metadata.rb 檔案，並將它儲存至 s3bucket_ops。

```
name "s3download"
version "0.1.0"
```

3. 在 recipes 內建立 s3download 目錄。
4. 使用下列配方建立 default.rb 檔案，並將它儲存至 recipes 目錄。將 *windows-cookbooks* 取代為您用來存放要下載之檔案的 S3 儲存貯體名稱。

```
Chef::Log.info("******Downloading an object from S3******")

chef_gem "aws-sdk-s3" do
  compile_time false
  action :install
end

ruby_block "download-object" do
  block do
    require 'aws-sdk-s3'

    Aws.use_bundled_cert!

    s3_client = Aws::S3::Client.new(region:'us-west-2')

    s3_client.get_object(bucket: 'windows-cookbooks',
                         key: 'myfile.txt',
                         response_target: '/chef/myfile.txt')
  end
  action :run
end
```

5. 建立 s3download 的 .zip 封存，並將檔案上傳至 S3 儲存貯體。將檔案設為公有，並記錄 URL 以供日後使用。

6. 建立名為 myfile.txt 的文字檔，並將其上傳至 S3 儲存貯體。這是您配方將下載的檔案，因此您可以使用任何方便使用的儲存貯體。

配方會執行下列任務。

1：安裝適用於 Ruby v2 的 SDK。

此範例使用適用於 Ruby 的 SDK 下載 物件。不過， AWS OpsWorks Stacks 不會在 Windows 執行個體上安裝此開發套件，因此配方的第一部分會使用 chef_gem 資源來處理該任務。您可以使用此資源來安裝 Gem 以供 Chef 使用，其中包含配方。

2：下載檔案。

配方的第三個部分使用 ruby_block 資源來執行適用於 Ruby 的 SDK v2 程式碼，myfile.txt從名為 的 S3 儲存貯體下載*windows-cookbooks*到執行個體的/chef目錄。將 *windows-cookbooks* 變更為包含 myfile.txt 的儲存貯體名稱。

> **Note**
>
> 配方是 Ruby 應用程式，因此，您可以將 Ruby 程式碼放入配方主體中；它不需要在 ruby_block 資源中。不過，Chef 會先執行配方主體中的 Ruby 程式碼，接著依序執行每個資源。在此範例中，如果您將下載程式碼放在配方的內文中，它將會失敗，因為它取決於適用於 Ruby 的 SDK，以及安裝 SDK chef_gem的資源尚未執行。ruby_block 資源中的程式碼會在資源執行時執行，並在chef_gem資源安裝適用於 Ruby 的 SDK 後執行。

針對此範例建立堆疊，如下所示。您也可以使用現有 Windows 堆疊。只需要更新技術指南，如後面所述。

建立堆疊

1. 開啟 AWS OpsWorks Stacks 主控台，然後選擇 Add Stack (新增堆疊)。指定下列設定，並接受其他設定的預設值，然後選擇 Add Stack (新增堆疊)。

   - 名稱 – S3Download
   - 區域 – 美國西部 （ 奧勒岡 ）

     此範例適用於任何區域，但我們建議您使用美國西部 （ 奧勒岡 ） 進行教學課程。
   - 預設作業系統 – Microsoft Windows Server 2012 R2

2. 選擇 Add a layer (新增 layer)，並新增自訂 layer 至具有下列設定的堆疊。

   - 名稱 – S3Download
   - 簡短名稱 – s3download

3. 新增全年無休執行個體 (具有預設設定) 至 S3Download layer，以及啟動它。

您現在可以安裝並執行配方

執行配方

1. 編輯堆疊以啟用自訂技術指南，然後指定下列設定。

   - 儲存庫類型 – S3 Archive。
   - 儲存庫 URL – 您先前記錄的技術指南的封存 URL。

   接受其他設定的預設值，然後選擇 Save (儲存) 以更新堆疊組態。

2. 執行更新自訂技術指南堆疊命令，以在堆疊線上執行個體上安裝最新版的自訂技術指南。如果存在舊版的技術指南，則此命令會予以覆寫。
3. 使用配方執行執行執行配方堆疊命令以執行設定為 的配方**`s3download::default`**。此命令會啟動 Chef 執行，內含包含 `s3download::default` 的回合清單。

> **Note**
>
> 您通常會讓 AWS OpsWorks Stacks 透過將配方指派給適當的生命週期事件來自動執行配方。您也可以手動觸發事件來執行這類配方。您可以使用堆疊命令來觸發安裝和設定事件，以及使用部署命令來觸發部署和解除部署事件。

成功執行配方之後，您就可以驗證配方。

驗證 s3download

1. 第一步是檢查 Chef 日誌。您的堆疊應該有一個名為 s3download1 的執行個體。在執行個體頁面上，選擇在執行個體的日誌欄中顯示 ，以顯示 Chef 日誌。向下捲動，以在接近底端發現您的日誌訊息。

```
...
[2015-05-01T21:11:04+00:00] INFO: Loading cookbooks [s3download@0.0.0]
[2015-05-01T21:11:04+00:00] INFO: Storing updated cookbooks/s3download/recipes/
default.rb in the cache.
[2015-05-01T21:11:04+00:00] INFO: ******Downloading an object from S3******
[2015-05-01T21:11:04+00:00] INFO: Processing chef_gem[aws-sdk] action install
 (s3download::default line 3)
[2015-05-01T21:11:05+00:00] INFO: Processing ruby_block[download-object] action run
 (s3download::default line 8)
...
```

2. 使用 RDP 登入執行個體，並檢查 `c:\chef` 的內容。

安裝 Windows 軟體

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 這些範例假設您已完成在 Windows 執行個體上執行配方範例。否則，您應該先執行該範例。具體而言，它說明如何啟用 RDP 存取您的執行個體。

Windows 執行個體是從 Windows Server 2012 R2 Standard 開始，因此，您一般需要安裝一些軟體。詳細資訊取決於軟體類型。

- Windows 功能是選用的系統元件，包括 .NET 架構和網際網路資訊服務 (IIS)，您可以下載到您的執行個體。
- 第三方軟體通常會隨附您必須下載至執行個體後執行的安裝程式套件 (例如 MSI 檔案)。

  一些 Microsoft 軟體也會隨附安裝程式套件。

本節說明如何實作技術指南來安裝 Windows 功能和套件。它也會介紹 Chef Windows 技術指南，其中包含資源和協助程式函數可簡化實作 Windows 執行個體的配方。

主題

- 安裝 Windows 功能：IIS
- 在 Windows 執行個體上安裝套件

安裝 Windows 功能：IIS

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Windows 功能是一組選用的系統元件，包括 .NET 架構和網際網路資訊服務 (IIS)。本主題說明如何實作技術指南來安裝常用的功能：網際網路資訊服務 (IIS)。

> **Note**
>
> 安裝套件顯示如何安裝軟體，其隨附您必須下載至執行個體並執行的安裝程式套件 (例如 MSI 檔案)。IIS 技術指南

在 Windows 執行個體上執行配方顯示如何使用 `powershell_script` 資源來安裝 Windows 功能。此範例顯示替代方式：使用 Chef Windows 技術指南的 `windows_feature` 資源。此技術指南包含一組資源，以使用部署映像服務和管理在 Windows 上執行各種任務，包括功能安裝。

> **Note**
>
> Chef 也具有 IIS 技術指南，以用來管理 IIS。如需詳細資訊，請參閱 IIS 技術指南。

設定技術指南

1. 前往 windows 技術指南 GitHub 儲存庫，並下載 `windows` 技術指南。

   此範例假設您將 `windows` 儲存庫下載為 .zip 檔案，但如果您想要，也可以複製儲存庫。
2. 前往 chef_handler 技術指南 GitHub 儲存庫，並下載 `chef-handler` 技術指南。

windows 技術指南取決於 chef_handler；您將不會直接使用它。此範例假設您將 chef_handler 儲存庫下載為 .zip 檔案，但如果您想要，也可以複製儲存庫。

3. 將 windows 和 chef_handler 技術指南分別解壓縮至 cookbooks 目錄中名為 windows 和 chef_handler 的目錄。
4. 在 cookbooks 目錄中建立並導覽至名為 install-iis 的目錄。
5. 將 metadata.rb 檔案新增至具有以下內容的 install-iis。

```
name "install-iis"
version "0.1.0"

depends "windows"
```

depends 指令可讓您在配方中使用 windows 技術指南資源。

6. 將 recipes 目錄新增至 install-iis，並將名為 default.rb 的檔案新增至包含下列配方程式碼的那個目錄。

```
%w{ IIS-WebServerRole IIS-WebServer }.each do |feature|
  windows_feature feature do
    action :install
  end
end

service 'w3svc' do
  action [:start, :enable]
end
```

配方使用 windows 技術指南的 windows_feature 資源來安裝下列項目：

1. IIS Web 伺服器角色。
2. IIS Web 伺服器。

配方接著會使用 service 資源來啟動和啟用 IIS 服務 (W3SVC)。

> **Note**
>
> 如需完整的可用 Windows 功能清單，請使用 RDP 登入執行個體，並開啟命令提示視窗，然後執行下列命令。請注意，清單相當長。
>
> ```
> dism /online /Get-Features
> ```

7. 建立包含 `install-iis`、`chef_handler` 和 `windows` 技術指南的 `.zip` 封存，並將封存上傳至 S3 儲存貯體。將封存設為公有，並記錄 URL 以供日後使用。此範例假設封存命名為 `install-iis.zip`。如需詳細資訊，請參閱技術指南儲存庫。

   傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

針對此範例建立堆疊，如下所示。您也可以使用現有 Windows 堆疊。只需要更新技術指南，如後面所述。

建立堆疊

1. 開啟 AWS OpsWorks Stacks 主控台，然後選擇 Add Stack (新增堆疊)。指定下列設定，並接受其他設定的預設值，然後選擇 Add Stack (新增堆疊)。

   - 名稱 – InstallIIS
   - 區域 – 美國西部 （ 奧勒岡 ）

     此範例適用於任何區域，但我們建議您使用美國西部 （ 奧勒岡 ） 進行教學課程。
   - 預設作業系統 – Microsoft Windows Server 2012 R2
2. 選擇 Add a layer (新增 layer)，並新增自訂 layer 至具有下列設定的堆疊。

   - 名稱 – IIS
   - 簡短名稱 – iis
3. 新增全年無休執行個體 (具有預設設定) 至 IIS layer，以及啟動它。

您現在可以安裝技術指南，並執行配方

安裝技術指南並執行配方

1. 編輯堆疊以啟用自訂技術指南，然後指定下列設定。

   - 儲存庫類型 – S3 封存
   - 儲存庫 URL – 您先前記錄的技術指南封存的 URL。

   接受其他設定的預設值，然後選擇 Save (儲存) 以更新堆疊組態。
2. 執行 Update Custom Cookbooks (更新自訂技術指南) 堆疊命令，以在堆疊線上執行個體上安裝最新版的自訂技術指南。如果存在舊版的技術指南，則此命令會予以覆寫。
3. 透過使用配方執行執行配方堆疊命令以執行設定為 的配方**`install-iis::default`**。此命令會啟動 Chef 執行，以執行指定的配方。

   > **Note**
   >
   > 為了方便起見，此範例使用執行配方，但您通常會讓 AWS OpsWorks Stacks 透過將配方指派給適當的生命週期事件來自動執行配方。您可以手動觸發事件來執行這類配方。您可以使用堆疊命令來觸發安裝和設定事件，以及使用部署命令來觸發部署和解除部署事件。
4. 若要驗證安裝，請使用 RDP 連線至執行個體，並開啟 Windows 檔案總管。檔案系統現在應該會有一個 `C:\inetpub` 目錄。如果您檢查控制台系統管理工具應用程式中的服務清單，則 IIS 應該接近底端。不過，它將命名為 World Wide Web Publishing Service，而不是 IIS。

在 Windows 執行個體上安裝套件

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此範例假設您已完成在 Windows 執行個體上執行配方範例。否則，您應該先執行該範例。具體而言，它說明如何啟用 RDP 存取您的執行個體。

如果您的軟體隨附安裝程式套件 (例如 MSI)，則您必須將檔案下載至執行個體後執行。此範例顯示如何實作技術指南來安裝 MSI 套件 (Python 執行時間)，包括如何定義相關的環境變數。如需如何安裝 Windows 功能 (例如 IIS) 的詳細資訊，請參閱安裝 Windows 功能：IIS。

設定技術指南

1. 建立並導覽至名為 `installpython` 的目錄。
2. 將 `metadata.rb` 檔案新增至具有以下內容的 `installpython`。

```
name "installpython"
version "0.1.0"
```

3. 將 `recipes` 和 `files` 目錄新增至 `installpython`，並將 `default` 目錄新增至檔案。
4. 將 Python 套件從適用於 Windows 的 Python 版本下載至技術指南的 `files\default` 目錄。此範例會安裝 Windows x86-64 版的 Python 3.5.0a3，以使用名為 `python-3.4.3.amd64.msi` 的 MSI 安裝程式。
5. 將名為 `default.rb` 的檔案新增至包含下列配方程式碼的 `recipes` 目錄。

```
directory 'C:\tmp' do
  rights :full_control, 'Everyone'
  recursive true
  action :create
end

cookbook_file 'C:\tmp\python-3.4.3.amd64.msi' do
  source "python-3.4.3.amd64.msi"
  rights :full_control, 'Everyone'
  action :create
end

windows_package 'python' do
  source 'C:\tmp\python-3.4.3.amd64.msi'
  action :install
end

env "PATH" do
  value 'c:\python34'
  delim ";"
  action :modify
```

```
end
```

配方會執行下列動作：

1. 使用 directory 資源來建立 C:\tmp 目錄。

   如需此資源的詳細資訊，請參閱範例 3：建立目錄。

2. 使用 cookbook_file 資源，將安裝程式從技術指南的 files\default 目錄複製至 C:\tmp。

   如需此資源的詳細資訊，請參閱從技術指南安裝檔案。

3. 使用 windows_package 資源執行 MSI 安裝程式，以將 Python 安裝至 c:\python34。

   安裝程式會建立必要的目錄並安裝檔案，但不會修改系統的 PATH 環境變數。

4. 使用 env 資源，將 c:\python34 新增至系統路徑。

   您可以使用 env 資源定義環境變數。在此情況下，配方透過將 c:\python34 新增至路徑，以讓您輕鬆地從命令列執行 Python 指令碼。

   - 資源名稱指定環境變數的名稱，在此範例中為 PATH。
   - value 屬性指定變數的值，在此範例中為 c:\\python34 (您需要逸出 \ 字元)。
   - :modify 動作會將指定的值附加到變數目前值的前面。
   - delim 屬性指定隔開新值與現有值的分隔符號，在此範例中為 ;。

6. 建立 .zip 的 installpython 封存，並將封存上傳至 S3 儲存貯體，然後將它設為公有。記錄封存的 URL，供日後使用。如需詳細資訊，請參閱技術指南儲存庫。

   傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

針對此範例建立堆疊，如下所示。您也可以使用現有 Windows 堆疊。只需要更新技術指南，如後面所述。

建立堆疊

1. 開啟 AWS OpsWorks Stacks 主控台，然後選擇 Add Stack (新增堆疊)。指定下列設定，並接受其他設定的預設值，然後選擇 Add Stack (新增堆疊)。

   - 名稱 – InstallPython
   - 區域 – 美國西部 （ 奧勒岡 ）

此範例適用於任何區域，但我們建議您使用美國西部（奧勒岡）進行教學課程。

- 預設作業系統 – Microsoft Windows Server 2012 R2

2. 選擇 Add a layer (新增 layer)，並新增自訂 layer 至具有下列設定的堆疊。
   - 名稱 – Python
   - 簡短名稱 – python
3. 新增全年無休執行個體 (具有預設設定) 至 Python layer，以及啟動它。

在執行個體上線之後，您可以安裝技術指南並執行配方

安裝技術指南並執行配方

1. 編輯堆疊以啟用自訂技術指南，然後指定下列設定。
   - 儲存庫類型 – S3 Archive。
   - 儲存庫 URL – 您先前記錄的技術指南的封存 URL。

   接受其他設定的預設值，然後選擇 Save (儲存) 以更新堆疊組態。
2. 執行 Update Custom Cookbooks (更新自訂技術指南) 堆疊命令，以在堆疊線上執行個體上安裝最新版的自訂技術指南。如果存在舊版的技術指南，則此命令會予以覆寫。
3. 透過使用配方執行執行執行配方堆疊命令來執行設定為 的配方**`installpython::default`**。此命令會啟動 Chef 執行，內含包含 `installpython::default` 的回合清單。

   > **Note**
   >
   > 為了方便起見，此範例使用執行配方，但您通常會讓 AWS OpsWorks Stacks 透過將配方指派給適當的生命週期事件來自動執行配方。您可以手動觸發事件來執行這類配方。您可以使用堆疊命令來觸發安裝和設定事件，以及使用部署命令來觸發部署和解除部署事件。

4. 若要驗證安裝，請使用 RDP 連線至執行個體，並開啟 Windows 檔案總管。
   - 檔案系統現在應該會有一個 `C:\Python34` 目錄。
   - 如果您從命令列執行 `path`，則它應該看起來像：`PATH=c:\python34;C:\Windows\system32;...`
   - 如果您從命令列執行 `python --version`，則它應該會傳回 `Python 3.4.3`。

覆寫內建屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 本主題僅適用於 Linux 堆疊。您無法在 Windows 堆疊上覆寫內建屬性。

AWS OpsWorks Stacks 會在每個執行個體上安裝一組內建技術指南。許多內建技術指南都支援內建 layer，而且其屬性檔案定義各種預設系統和應用程式設定 (例如 Apache 伺服器組態設定)。將這些設定放在屬性檔案中，即可使用下列方式覆寫對應內建屬性來自訂許多組態設定：

- 使用自訂 JSON 定義屬性。

  此方式的優點是簡單並具彈性。不過，您必須手動輸入自訂 JSON，因此沒有健全的方法可以管理屬性定義。
- 實作自訂技術指南，並在 `customize.rb` 屬性檔案中定義屬性。

  此方式比使用自訂 JSON 更不具彈性，但是更為健全，因為您可以將自訂技術指南放在來源控制之下。

本主題說明如何使用自訂技術指南屬性檔案覆寫內建屬性，並使用 Apache 伺服器做為範例。如需如何覆寫具有自訂 JSON 之屬性的詳細資訊，請參閱 使用自訂 JSON。如需如何覆寫屬性的一般討論，請參閱覆寫屬性。

> **Note**
>
> 偏好使用覆寫屬性來自訂組態設定，但不會一律透過屬性呈現設定。在該情況下，您通常可以覆寫內建配方用來建立組態檔案的範本，來自訂組態檔案。如需範例，請參閱「覆寫內建範本」。

內建屬性通常代表範本檔案中安裝配方用來建立組態檔案的值。例如，其中一個 apache2 安裝配方 default.rb 使用 apache2.conf.erb 範本來建立 Apache 伺服器的主要組態檔案 httpd.conf (Amazon Linux) 或 apache2.conf (Ubuntu)。以下是範例檔案中的摘錄：

```
...
#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests <%= node[:apache][:keepaliverequests] %>
#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout <%= node[:apache][:keepalivetimeout] %>
##
## Server-Pool Size Regulation (MPM specific)
##

...
```

此範例中的 KeepAliveTimeout 設定是 [:apache][:keepalivetimeout] 屬性的值。此屬性的預設值定義於 apache2 技術指南的 apache.rb 屬性檔案，如下列摘錄所示：

```
...
# General settings
default[:apache][:listen_ports] = [ '80','443' ]
default[:apache][:contact] = 'ops@example.com'
default[:apache][:log_level] = 'info'
default[:apache][:timeout] = 120
default[:apache][:keepalive] = 'Off'
default[:apache][:keepaliverequests] = 100
default[:apache][:keepalivetimeout] = 3
...
```

> **Note**
>
> 如需常用內建屬性的詳細資訊，請參閱內建技術指南屬性。

若要支援覆寫內建屬性，所有內建技術指南都會包含 customize.rb 屬性檔案，而此屬性檔案透過 include_attribute 指令併入所有模組中。內建技術指南的 customize.rb 檔案未包含任何屬性定義，而且不會影響內建屬性。若要覆寫內建屬性，您可以建立與內建技術指南同名的自訂技術指南，並將自訂屬性定義放入也命名為 customize.rb 的屬性檔案中。該檔案的優先順序高於內建版本，並包含在任何相關模組中。如果您在 customize.rb 中定義任何內建屬性，則它們會覆寫對應的內建屬性。

此範例顯示如何覆寫內建 [:apache][:keepalivetimeout] 屬性，以將其值設定為 5，而非 3。您可以將類似的方式用於任何內建屬性。不過，請注意您覆寫哪些屬性。例如，opsworks 命名空間中的覆寫屬性可能會導致一些內建配方的問題。

> **Important**
>
> 請不要修改內建屬性檔案的複本本身來覆寫內建屬性。例如，您「可以」將 apache.rb 的複本放入您自訂技術指南的 apache2/attributes 資料夾，並修改它的一些設定。不過，此檔案的優先順序高於內建版本，而內建配方現在會使用您的 apache.rb 版本。如果 AWS OpsWorks Stacks 稍後修改內建apache.rb檔案，除非您手動更新版本，否則配方將不會取得新值。透過使用 customize.rb，您只會覆寫指定的屬性；內建配方會持續自動取得每個尚未覆寫之屬性的最新值。

若要開始，請建立自訂技術指南。

建立技術指南

1. 在 opsworks_cookbooks 目錄內，建立並導覽至名為 apache2 的技術指南目錄。

   若要覆寫內建屬性，自訂技術指南必須與內建技術指南同名，在此範例中為 apache2。

2. 在 apache2 目錄中，建立 attributes 目錄。
3. 將名為 customize.rb 的檔案新增至 attributes 目錄，並使用它來定義您想要覆寫的內建技術指南屬性。在此範例中，檔案應該包含下列項目：

   ```
   normal[:apache][:keepalivetimeout] = 5
   ```

> **Important**
>
> 若要覆寫內建屬性，自訂屬性必須是 `normal` 類型或以上類型，並且具有與對應內建屬性完全相同的節點名稱。`normal` 類型確保自訂屬性的優先順序高於內建屬性，即所有 `default` 類型。如需詳細資訊，請參閱屬性優先順序。

4. 建立`opsworks_cookbooks`名為 的`.zip`封存，`opsworks_cookbooks.zip`並將封存上傳至 Amazon Simple Storage Service (Amazon S3) 儲存貯體。為求簡化，請將檔案設為公有。記錄 URL，供日後使用。您也可以將技術指南存放在私有 Amazon S3 封存或其他儲存庫類型中。如需詳細資訊，請參閱技術指南儲存庫。

   傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

若要使用自訂屬性，請建立堆疊，並安裝技術指南。

使用自訂屬性

1. 開啟 AWS OpsWorks Stacks 主控台，然後選擇 Add Stack (新增堆疊)。
2. 指定下列標準設定。

   - 名稱 – ApacheConfig
   - 區域 – 美國西部 （ 奧勒岡 ）

     您可以將堆疊放在任何區域，但我們建議美國西部 （ 奧勒岡 ） 進行教學課程。
   - 預設 SSH 金鑰 – EC2 金鑰對

     如果您需要建立 EC2 金鑰對，請參閱 Amazon EC2 金鑰對。請注意，金鑰對必須屬於與堆疊相同的 AWS 區域。

   選擇 Advanced>> (進階>>)，並將 Use custom Chef cookbooks (使用自訂 Chef 技術指南) 設定為 Yes (是)，然後指定下列設定。

   - 儲存庫類型 – Http Archive
   - 儲存庫 URL – 您先前記錄的技術指南封存的 URL

接受其他設定的預設值，然後選擇 Add Stack (新增堆疊) 來建立堆疊。

> **Note**
>
> 此範例使用預設作業系統：Amazon Linux。如果您想要，可以使用 Ubuntu。唯一的差異是，在 Ubuntu 系統上，內建安裝配方會產生具有相同設定且名為 `apache2.conf` 的組態檔案，並將它放入 `/etc/apache2` 目錄中。

3. 選擇新增 layer，然後將具有預設設定的 Java App Server layer 新增至堆疊。
4. 新增全年無休執行個體 (具有預設設定) 至 layer，然後啟動執行個體。

   在此範例中，t2.micro 執行個體就已足夠。
5. 執行個體上線之後，請使用 SSH 連線至它。`httpd.conf` 檔案位在 `/etc/httpd/conf` 目錄中。如果您檢查檔案，則應該會看到您的自訂 `KeepAliveTimeout` 設定。這些設定的其餘部分將會有內建 `apache.rb` 檔案中的預設值。`httpd.conf` 的相關部分應該看起來與下列類似：

```
...
#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout 5
...
```

覆寫內建範本

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 本主題僅適用於 Linux 堆疊。您無法在 Windows 堆疊上覆寫內建範本。

AWS OpsWorks Stacks 內建配方使用 範本在執行個體上建立檔案，主要是伺服器組態檔案，例如 Apache。例如，apache2 配方使用 apache2.conf.erb 範本來建立 Apache 伺服器的主要組態檔案 httpd.conf (Amazon Linux) 或 apache2.conf (Ubuntu)。

這些範本中的大部分組態設定是以屬性呈現，因此自訂組態檔案的偏好方式是覆寫適當的內建屬性。如需範例，請參閱「覆寫內建屬性」。不過，如果您想要自訂的設定不是以內建屬性呈現，或根本不在範本中，則必須覆寫範本本身。本主題說明如何覆寫內建範本，以指定自訂 Apache 組態設定。

您可以將 ErrorDocument 設定新增至 httpd.conf 檔案，以將自訂錯誤回應提供給 Apache。apache2.conf.erb 只會包含一些標示為註解的範例，如下所示：

```
...
#
# Customizable error responses come in three flavors:
# 1) plain text 2) local redirects 3) external redirects
#
# Some examples:
#ErrorDocument 500 "The server made a boo boo."
#ErrorDocument 404 /missing.html
#ErrorDocument 404 "/cgi-bin/missing_handler.pl"
#ErrorDocument 402 http://www.example.com/subscription_info.html
...
```

因為這些設定是硬式編碼註解，所以您無法覆寫屬性來指定自訂值；您必須覆寫範本本身。不過，與屬性不同的是，沒有方法可以覆寫範本檔案的特定部分。您必須建立與內建版本同名的自訂技術指南、將範本檔案複製至相同的子目錄，以及視需要修改檔案。本主題顯示如何覆寫 apache2.conf.erb 以提供錯誤 500 的自訂回應。如需覆寫範本的一般討論，請參閱使用自訂範本 。

> **Important**
>
> 當您覆寫內建範本時，內建配方會使用您自訂的範本版本，而不是內建版本。如果 AWS OpsWorks Stacks 更新內建範本，自訂範本會變得不同步且可能無法正常運作。 AWS OpsWorks Stacks 不會經常進行此類變更，而且當範本確實變更時， AWS OpsWorks Stacks 會列出變更，並讓您選擇升級至新版本。建議您監控 AWS OpsWorks Stacks 儲存庫的變更，

並視需要手動更新自訂範本。請注意，儲存庫的每個支援 Chef 版本都有個別的分支，因此請確定您在正確的分支中。

若要開始，請建立自訂技術指南。

建立技術指南

1. 在 opsworks_cookbooks 目錄中，建立並導覽至名為 apache2 的技術指南目錄。若要覆寫內建範本，自訂技術指南必須與內建技術指南同名，在此範例中為 apache2。

   > **Note**
   >
   > 如果您已經完成 覆寫內建屬性 逐步解說，則可以在此範例中使用相同的 apache2 技術指南，並略過步驟 2。

2. 使用下列內容建立 metadata.rb 檔案，然後將它儲存至 apache2 目錄。

   ```
   name "apache2"
   version "0.1.0"
   ```

3. 在 apache2 目錄中，建立 templates/default 目錄。

   > **Note**
   >
   > templates/default 目錄適用於使用預設apache2.conf.erb範本的 Amazon Linux 執行個體。Ubuntu 14.04 執行個體使用作業系統特定 apache2.conf.erb 範本，其位在 templates/ubuntu-14.04 目錄中。如果您也想要自訂套用至 Ubuntu 14.04 執行個體，則也必須覆寫該範本。

4. 將內建 apache2.conf.erb 範本複製至您的 templates/default 目錄。開啟範本檔案，並將 ErrorDocument 500 行標示為註解，並提供自訂錯誤訊息，如下所示：

   ```
   ...
   ErrorDocument 500 "A custom error message."
   #ErrorDocument 404 /missing.html
   ...
   ```

5. 建立opsworks_cookbooks名為 的.zip封存opsworks_cookbooks.zip，然後將檔案上傳至 Amazon Simple Storage Service (Amazon S3) 儲存貯體。為求簡化，請將封存設為公有。記錄封存的 URL，供日後使用。您也可以將技術指南存放在私有 Amazon S3 封存或其他儲存庫類型中。如需詳細資訊，請參閱技術指南儲存庫。

   傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

> **Note**
>
> 為求簡化，此範例會將硬式編碼錯誤訊息新增至範本。若要變更它，您必須修改範本，並重新安裝技術指南。若要提供更大的彈性，您可以在自訂技術指南的屬性檔案中定義錯誤字串的預設自訂customize.rb屬性，並將該屬性的值指派給 ErrorDocument 500。例如，如果您將屬性命名為 [:apache][:custom][:error500]，則 apache2.conf.erb 中的對應行看起來應該與下列類似：
>
> ```
> ...
> ErrorDocument 500 <%= node[:apache][:custom][:error500] %>
> #ErrorDocument 404 /missing.html
> ...
> ```
>
> 您接著可以覆寫 [:apache][:custom][:error500]，隨時變更自訂錯誤訊息。如果您使用自訂 JSON 覆寫屬性，則甚至不需要接觸技術指南。

若要使用自訂範本，請建立堆疊，並安裝技術指南。

使用自訂範本

1. 開啟 AWS OpsWorks Stacks 主控台，然後選擇 Add Stack (新增堆疊)。
2. 指定下列標準設定：
   - 名稱 – ApacheTemplate
   - 區域 – 美國西部 （ 奧勒岡 ）
   - 預設 SSH 金鑰 – Amazon Elastic Compute Cloud (Amazon EC2) 金鑰對

如果您需要建立 Amazon EC2 金鑰對，請參閱 Amazon EC2 金鑰對。請注意，金鑰對必須屬於與執行個體相同的 AWS 區域。

選擇 Advanced>> (進階>>)，並選擇 Use custom Chef cookbooks (使用自訂 Chef 技術指南) 指定下列設定：

- 儲存庫類型 – Http Archive
- 儲存庫 URL – 您先前記錄的技術指南封存的 URL

接受其他設定的預設值，然後選擇 Add Stack (新增堆疊) 來建立堆疊。

3. 選擇新增 layer，然後使用預設設定將 Java App Server layer 新增至堆疊。
4. 新增全年無休執行個體 (具有預設設定) 至 layer，然後啟動執行個體。

   在此範例中，t2.micro 執行個體就已足夠。
5. 執行個體上線之後，請使用 SSH 連線至它。`httpd.conf` 檔案位在 `/etc/httpd/conf` 目錄中。檔案應該包含自訂 `ErrorDocument` 設定，看起來會與下列類似：

```
...
# Some examples:
ErrorDocument 500 "A custom error message."
#ErrorDocument 404 /missing.html
#ErrorDocument 404 "/cgi-bin/missing_handler.pl"
#ErrorDocument 402 http://www.example.com/subscription_info.html
...
```

平衡 Layer 的負載

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 提供兩種負載平衡選項：Elastic Load Balancing 和 HAProxy，通常用於平衡應用程式伺服器層執行個體的負載。本主題說明其各自的優點和限制，幫助您在 layer 新增負載平衡時決定要選擇的選項。在某些情況下，最好的方法是兩個都使用。

SSL 終止

內建 HAProxy layer 不會處理 SSL 終止；您必須在伺服器上終止 SSL。此方法的優點是會加密流量，直到它到達伺服器。不過，伺服器必須處理解密，這會增加伺服器的負載。此外，您必須將您的 SSL 憑證放在應用程式伺服器上，這讓使用者更容易存取。

使用 Elastic Load Balancing，您可以在負載平衡器終止 SSL。這可減少應用程式伺服器的負載，但負載平衡器與伺服器之間的流量不會加密。Elastic Load Balancing 也可讓您終止伺服器上的 SSL，但設定有點複雜。

擴展

如果傳入流量超過 HAProxy 負載平衡器的容量，您就必須手動增加容量。

Elastic Load Balancing 會自動擴展以處理傳入流量。為了確保 Elastic Load Balancing 負載平衡器在第一次上線時有足夠的容量來處理預期的負載，您可以對其進行預熱。

負載平衡器故障

如果託管您 HAProxy 伺服器的執行個體故障，您可能要讓整個網站離線，直到您可以重新啟動執行個體為止。

Elastic Load Balancing 比 HAProxy 更能防止故障。例如，它在每個已註冊 EC2 執行個體的可用區域中佈建負載平衡節點。如果某個區域的服務中斷，其他節點會繼續處理傳入的流量。如需詳細資訊，請參閱 Elastic Load Balancing 概念。

閒置逾時

如果伺服器閒置超過指定的閒置逾時值，兩種負載平衡器都會終止連線。

- HAProxy – 閒置逾時值沒有上限。
- Elastic Load Balancing – 預設閒置逾時值為 60 秒，上限為 3600 秒 (60 分鐘）。

Elastic Load Balancing 閒置時間限制足以用於大多數用途。如果您需要較長的閒置逾時，我們建議您使用 HAProxy。例如：

- 推送通知所用之長時間執行的 HTTP 連線。
- 您用來執行可能需要超過 60 分鐘任務的管理界面。

以 URL 為基礎的映射

您可能希望使用負載平衡器，根據請求的 URL 將傳入請求轉發到特定的伺服器。例如，假設您有 10 個一組的應用程式伺服器支援線上商務應用程式。八個伺服器處理型錄，兩個處理付款。您想要根據請求 URL，將所有付款相關的 HTTP 請求導向至付款伺服器。在此案例中，您會將包含「付款」或「結帳」的所有 URL 導向至其中一個付款伺服器。

透過 HAProxy，您可以使用以 URL 為基礎的映射將包含指定字串的 URL 導向到特定的伺服器。若要搭配 AWS OpsWorks Stacks 使用 URL 型映射，您必須覆寫`haproxy`內建技術指南中的`haproxy-default.erb`範本，以建立自訂 HAProxy 組態檔案。如需詳細資訊，請參閱 HAProxy 組態手冊和使用自訂範本 。您不能使用以 URL 為基礎的映射處理 HTTPS 請求。HTTPS 請求已加密，所以 HAProxy 無法檢查請求 URL。

Elastic Load Balancing 對 URL 映射的支援有限。如需詳細資訊，請參閱 Elastic Load Balancing 的接聽程式組態。

建議：我們建議您使用 Elastic Load Balancing 進行負載平衡，除非您有只能由 HAProxy 處理的需求。在這種情況下，最佳方法可能是使用 Elastic Load Balancing 作為前端負載平衡器，將傳入流量分配到一組 HAProxy 伺服器。若要執行此作業：

- 設定您堆疊中每個可用區域的 HAProxy 執行個體，將請求分配到區域的應用程式伺服器。
- 將 HAProxy 執行個體指派給 Elastic Load Balancing 負載平衡器，然後將傳入的請求分配至 HAProxy 負載平衡器。

這種方法可讓您使用以 URL 為基礎的 HAProxy 映射，將不同類型的請求分配到適當的應用程式伺服器。不過，如果其中一個 HAProxy 伺服器離線，網站會繼續運作，因為 Elastic Load Balancing 負載平衡器會自動將傳入流量分配到運作狀態良好的 HAProxy 伺服器。請注意，您必須使用 Elastic Load Balancing 做為前端負載平衡器；HAProxy 伺服器無法將請求分發到其他 HAProxy 伺服器。

從 Chef Server 遷移至 AWS OpsWorks Stacks

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

由於 AWS OpsWorks Stacks 是以 Chef 為基礎，因此從 Chef Server 遷移到 AWS OpsWorks Stacks 相當簡單。本主題提供修改 Chef Server 程式碼以使用 AWS OpsWorks Stacks 的準則。

> **Note**
>
> 我們不建議遷移至使用早於 Chef 11.10 版本的堆疊，因為這類版本是以 chef-solo 做為基礎，不支援搜尋或資料包。

主題

- 將角色映射至 Layer
- 使用資料包
- 使用 Chef 搜尋
- 管理技術指南和配方
- 使用 Chef 環境

將角色映射至 Layer

Chef Server 使用角色代表及管理具有相同用途和組態的執行個體，例如一組執行個體，其中每個執行個體都主控一個 Java 應用程式伺服器。基本上，AWS OpsWorks Stacks layer 的用途與 Chef 的角色相同。layer 是建立一組具有相同組態、已安裝套件、應用程式部署程序等 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體的藍圖。

AWS OpsWorks Stacks 包含一組適用於多種應用程式伺服器的內建層、HAProxy 負載平衡器、MySQL 資料庫主伺服器和 Ganglia 監控主伺服器。例如，內建 Java App Server layer 是建立託管 Tomcat 伺服器的執行個體的藍圖。

若要遷移至 AWS OpsWorks Stacks，您需要將每個角色與提供同等功能的 layer 建立關聯。針對某些角色，您可能可以直接使用其中一個內建 layer。其他角色則可能需要不同程度的自訂。從檢查內建 layer 的功能開始 (包含與每個 layer 關聯的配方) 以查看是否有 layer 至少能提供角色的一部分功能。如需內建 layer 的詳細資訊，請參閱 層 及 AWS OpsWorks Stacks Layer 參考。若要檢查內建配方，請參閱 AWS OpsWorks Stacks 公有 GitHub 儲存庫。

接下來您繼續的方式會根據您將 layer 對應到每個角色的程度而有所不同，如下所示。

內建 layer 支援角色所有的功能

您可以直接使用內建 layer，並且僅需要進行微幅的自訂 (若需要的話)。例如，如果角色支援 Tomcat 伺服器，Java App Server layer 的配方可能已經處理角色的所有任務，可能有一些適度的自訂。例如，您可以透過覆寫適當的屬性或範本，讓 layer 的內建配方使用自訂 Tomcat 或 Apache 組態設定。

內建 layer 支援角色的部分 (並非全部) 功能

您可能可以將 layer 延伸來使用內建 layer。這通常會涉及實作自訂配方，以支援沒有的功能，以及將配方指派給 layer 的生命週期事件。例如，假設您的角色在與主控 Tomcat 伺服器相同的執行個體上安裝 Redis 伺服器。您可以擴展 Java App Server layer 以符合角色的功能，方法是實作自訂配方以在 layer 的執行個體上安裝 Redis，並將配方指派給 layer 的設定事件。

沒有任何內建 layer 能適當支援角色的功能

實作自訂 layer。例如，假設您的角色支援 MongoDB 資料庫伺服器，但任何內建的 layer 皆不支援該資料庫伺服器。您可以透過實作配方，安裝必要套件、設定伺服器等，並將配方指派給自訂 layer 的生命週期事件，來提供該支援。通常，針對此目的，您至少可以使用角色一部分的配方。如需如何實作自訂 layer 的詳細資訊，請參閱建立自訂 Tomcat 伺服器 Layer。

使用資料包

Chef Server 允許您使用資料包將使用者定義的資料傳遞給您的配方。

- 您會使用您的技術指南存放您的資料，Chef 會在每個執行個體上安裝它。
- 您可以針對敏感性資料 (例如密碼) 使用加密資料包。

AWS OpsWorks Stacks 支援資料包；配方可以使用與 Chef Server 完全相同的程式碼來擷取資料。但是，支援具有下列限制和差異：

- 僅 Chef 11.10 Linux 及更新版本的堆疊支援資料包。

  Windows 堆疊和執行更早版本的 Linux 堆疊不支援資料包。
- 您不會將資料包存放在您的技術指南儲存庫。

  相反的，您使用自訂 JSON 管理您資料包的資料。
- AWS OpsWorks Stacks 不支援加密的資料包。

若您需要以加密的形式存放敏感性資料 (例如密碼或憑證)，我們建議您將其存放在私有 S3 儲存貯體。然後，您可以建立自訂配方，定義其使用適用於 Ruby 的 Amazon 開發套件擷取資料。如需範例，請參閱「使用適用於 Ruby 的 SDK」。

如需詳細資訊，請參閱使用資料包。

使用 Chef 搜尋

Chef Server 會將堆疊組態資訊 (例如 IP 地址和角色組態) 存放在伺服器上。配方使用 Chef 搜尋來擷取此資料。 AWS OpsWorks Stacks 使用稍微不同的方法。例如，Chef 11.10 Linux 堆疊是基於 Chef 用戶端的本機模式，即一種在執行個體上本機執行輕量版本 Chef Server (通常稱為 Chef Zero) 的 Chef 用戶端選項。Chef Zero 支援針對存放在執行個體節點物件中的資料進行搜尋。

AWS OpsWorks Stacks 不會將堆疊資料儲存在遠端伺服器上，而是為每個生命週期事件將一組堆疊組態和部署屬性新增至每個執行個體的節點物件。這些屬性代表堆疊組態的快照。他們使用與 Chef Server 相同的語法，代表配方需要用來從伺服器擷取的大多數資料。

您通常不需要修改配方的 Stacks AWS OpsWorks 搜尋相依程式碼。由於 Chef 搜尋會在包含堆疊組態和部署屬性的節點物件上運作，因此 Stacks AWS OpsWorks 中的搜尋查詢通常會像使用 Chef Server 一樣運作。

主要例外是因為堆疊組態和部署屬性只包含 AWS OpsWorks Stacks 在執行個體上安裝屬性時所知道的資料。如果您在本機在特定執行個體上建立或修改屬性，這些變更不會傳播回 AWS OpsWorks Stacks，也不會納入安裝在其他執行個體上的堆疊組態和部署屬性。您可以使用搜尋來僅擷取該執行個體上的屬性值。如需詳細資訊，請參閱使用 Chef 搜尋。

為了與 Chef Server 的相容性， AWS OpsWorks Stacks 會將一組`role`屬性新增至節點物件，每個屬性都包含堆疊的其中一個 layer 屬性。若您的配方使用 `roles` 做為搜尋鍵，您不需要變更搜尋程式碼。查詢會自動傳回對應 layer 的資料。例如，以下兩個查詢都會傳回 `php-app` layer 的屬性。

```
phpserver = search(:node, "layers:php-app").first
```

```
phpserver = search(:node, "roles:php-app").first
```

管理技術指南和配方

AWS OpsWorks Stacks 和 Chef Server 處理技術指南和配方的方式略有不同。Chef Server：

- 無論是自行實作，還是使用社群技術指南，您會提供所有的技術指南。
- 您會將技術指南存放在伺服器上。
- 您可以手動執行技術指南，或是根據定期排程執行。

使用 AWS OpsWorks Stacks：

- AWS OpsWorks Stacks 為每個內建 layer 提供一或多個技術指南。這些技術指南會處理標準任務，例如安裝和設定內建 layer 的軟體和部署應用程式。

  若要處理並未由內建技術指南執行的任務，您會將自訂技術指南新增至您的堆疊，或是使用社群技術指南。
- 您可以將 AWS OpsWorks Stacks 技術指南存放在遠端儲存庫中，例如 S3 儲存貯體或 Git 儲存庫。

  如需詳細資訊，請參閱存放技術指南。
- 您可以手動執行配方，但通常讓 AWS OpsWorks Stacks 為您執行配方，以回應在執行個體生命週期期間關鍵點發生的一組生命週期事件。

  如需詳細資訊，請參閱執行配方。
- AWS OpsWorks Stacks 僅支援 Chef 11.10 堆疊上的 Berkshelf。若您使用 Berkshelf 管理您的技術指南依存性，您無法使用執行 Chef 11.4 或更早版本的堆疊。

  如需詳細資訊，請參閱使用 Berkshelf。

主題

- 存放技術指南
- 執行配方

存放技術指南

使用 Chef Server 時，您會將您的技術指南存放在伺服器上，並將他們從伺服器部署到執行個體。使用 AWS OpsWorks Stacks，您可以將技術指南存放在儲存庫中：S3 或 HTTP 封存或 Git 或 Subversion 儲存庫。您可以在安裝技術指南時，指定 AWS OpsWorks Stacks 從儲存庫下載程式碼至堆疊執行個體所需的資訊。

若要從 Chef Server 進行遷移，您必須將您的技術指南放在這些儲存庫中的其中一個。如需如何建構技術指南儲存庫的資訊，請參閱技術指南儲存庫。

執行配方

在 AWS OpsWorks Stacks 中，每個 layer 都有一組生命週期事件：設定、設定、部署、取消部署和關機，這些事件都會在執行個體生命週期期間的關鍵點發生。若要執行自訂配方，您通常會將它指定給適當 layer 上的適當事件。當事件發生時， AWS OpsWorks Stacks 便會執行關聯的配方。例如，安裝事件會在執行個體完成開機之後發生，因此您通常會將執行像是安裝和設定套件，以及啟動服務等任務的配方指派給此事件。

您可以透過使用執行配方堆疊命令手動執行配方。此命令在開發和測試期間非常有用，但您也可以用它來執行不會映射到生命週期事件的配方。您也可以使用執行配方命令手動觸發安裝和設定事件。

除了 AWS OpsWorks Stacks 主控台之外，您還可以使用 AWS CLI 或 SDKs 來執行配方。這些工具支援所有 AWS OpsWorks Stacks API 動作，但通常會比使用 API 更簡單。使用 create-deployment CLI 命令觸發生命週期事件，執行所有關聯配方。您也可以使用此命令來執行一或多個配方，而不觸發事件。對等的軟體開發套件程式碼依存於特定語言，但通常與 CLI 命令相似。

以下範例說明使用 `create-deployment` CLI 命令自動化應用程式部署的兩種方式。

- 透過將具有單一執行個體的自訂 layer 新增至您的堆疊，來根據定期排程部署您的應用程式。

  將自訂安裝配方新增至在執行個體上建立 `cron` 任務的 layer，以根據指定的排程執行命令。如需如何使用配方來建立 `cron` 任務的範例，請參閱在 Linux 執行個體上執行 Cron 任務。
- 將任務新增至您使用 `create-deployment` CLI 命令部署應用程式的連續整合管道。

使用 Chef 環境

AWS OpsWorks Stacks 不支援 Chef 環境；`node.chef_environment` 一律傳回 `_default`。

AWS OpsWorks Stacks Layer 參考

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 部署的每個執行個體都必須是至少一個層的成員，該層定義執行個體在堆疊中的角色，並控制設定執行個體、安裝套件、部署應用程式等的詳細資訊。如需如何使用 AWS OpsWorks Stacks 建立和管理 layer 的詳細資訊，請參閱 層。

每個 layer 描述都包含 Stacks AWS OpsWorks 為每個 layer 生命週期事件執行的內建配方清單。這些配方存放在 https://github.com/aws/opsworks-cookbooks。請注意，清單僅包含直接由 Stacks AWS OpsWorks 執行的配方。這些配方有時會執行相依配方，而相依配方並未列出。若要查看特定事件的完整配方清單 (包括相依和自訂配方)，請檢查適當生命週期事件 Chef 日誌中的執行清單。

主題

- HAProxy Layer 參考
- HAProxy AWS OpsWorks Stacks Layer
- MySQL Layer 參考
- MySQL OpsWorks Layer
- 應用程式伺服器 Layer 參考
- 應用程式伺服器 Layer
- ECS 叢集層參考
- Custom Layer 參考
- 其他 Layer 參考
- 其他 Layer

HAProxy Layer 參考

Important

AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Note

此 layer 僅適用於 Linux 類型堆疊。

HAProxy layer 使用可靠的高效能 TCP/HTTP 負載平衡器 HAProxy，為 TCP 型和 HTTP 型應用程式提供高可用性負載平衡和代理服務。這特別適用於必須在極高負載下網路爬取同時需要持久性或第 7 Layer 處理的網站。

HAProxy 會監控流量，並顯示網頁上相關聯執行個體的統計資料和運作狀態。根據預設，URI 是 http：//*DNSName*/haproxy？stats，其中 *DNSName* 是 HAProxy 執行個體的 DNS 名稱。

Short name: (簡短名稱：) lb

相容性：HAProxy 層與下列層相容：自訂、db-master 和 memcached。

開放連接埠：HAProxy 允許公開存取連接埠 22 (SSH)、80 (HTTP) 和 443 (HTTPS)。

Autoassign Elastic IP addresses: (自動指派彈性 IP 地址：) 預設為開啟

Default EBS volume: (預設 EBS 磁碟區：) 否

Default security group: (預設安全群組：) AWS-OpsWorks-LB-Server

組態：若要設定 HAProxy layer，您必須指定下列項目：

- 運作狀態檢查 URI (預設：http://*DNSName*/)。
- 統計資料 URI (預設：http://*DNSName*/haproxy?stats)。
- 統計資料密碼 (選用)。
- 運作狀態檢查方法 (選用)。HAProxy 預設會使用 HTTP OPTIONS 方法。您也可以指定 GET 或 HEAD。
- 啟用統計資料 (選用)
- 連接埠. 根據預設， AWS OpsWorks Stacks 會將 HAProxy 設定為同時處理 HTTP 和 HTTPS 流量。您可以覆寫 Chef 組態範本 `haproxy.cfg.erb`，以設定 HAProxy 只處理其中任一個。

Setup recipes: (安裝配方：)

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- haproxy

Configure recipes: (設定配方：)

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- haproxy::configure

Deploy recipes: (部署配方：)

- deploy::default
- haproxy::configure

Shutdown recipes: (關機配方：)

- opsworks_shutdown::default
- haproxy::stop

安裝:

- AWS OpsWorks Stacks 會使用執行個體的套件安裝程式，將 HAProxy 安裝到其預設位置。
- 您必須設定 syslog 將日誌檔案導向指定的位置。如需詳細資訊，請參閱 HAProxy。

HAProxy AWS OpsWorks Stacks Layer

> **Note**
>
> 此 layer 僅適用於 Chef 11 或更舊的 Linux 類型堆疊。

AWS OpsWorks Stacks HAProxy layer 是 AWS OpsWorks Stacks layer，可為託管 HAProxy 伺服器的執行個體提供藍圖，這是可靠的高效能 TCP/HTTP 負載平衡。一個小型執行個體通常便足以處理所有應用程式伺服器流量。

> **Note**
>
> 堆疊會限制在單一區域。若要將您的應用程式分散至多個區域，您必須為每個區域分別建立堆疊。

建立 HAProxy layer

1. 在導覽窗格中，按一下 Layers (Layer)。
2. 在 Layers (Layer) 頁面上，按一下 Add a Layer (新增 Layer) 或 + Layer (+Layer)。針對 Layer type (Layer 類型)，選取 HAProxy。

layer 具有下列組態設定，所有設定皆為選擇性。

HAProxy statistics (HAProxy 統計)

layer 是否會收集及顯示統計。預設值為 Yes (是)。

Statistics URL (統計 URL)

統計頁面的 URL 路徑。完整的 URL 為 http://*DNSName*/*StatisticsPath*，其中 *DNSName* 是關聯執行個體的 DNS 名稱。預設 *StatisticsPath* 值為 /haproxy?stats，對應於類似：http://ec2-54-245-151-7.us-west-2.compute.amazonaws.com/haproxy?stats。

Statistics user name (統計使用者名稱)

統計資料頁面的使用者名稱，您必須提供才能檢視統計資料頁面。預設值為「opsworks」。

Statistics password (統計密碼)

統計頁面的密碼，您必須提供此密碼才能檢視統計頁面。預設值為隨機產生的字串。

Health check URL (運作狀態檢查 URL)

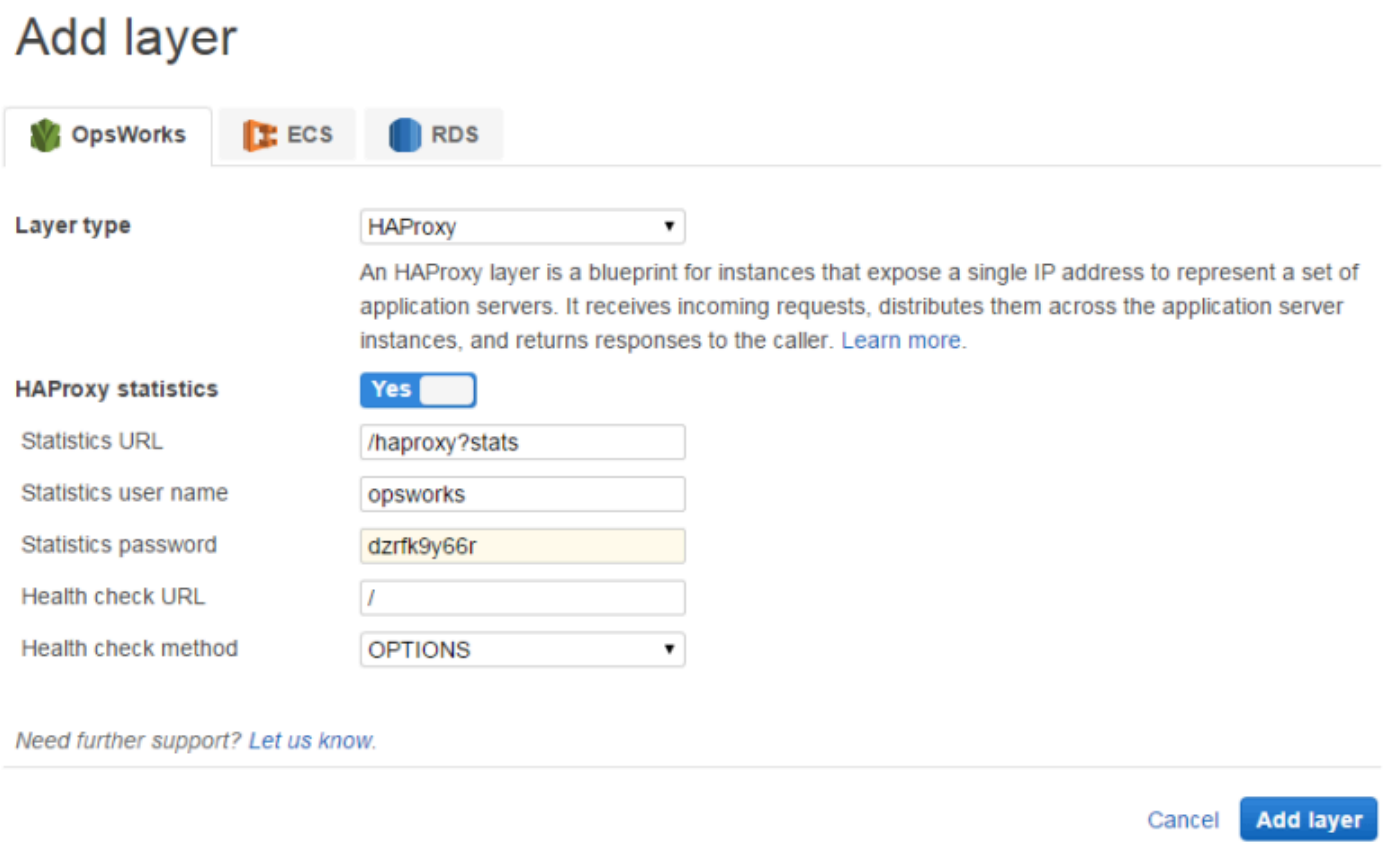
運作狀態檢查 URL 前綴。HAProxy 使用此 URL 在每個應用程式伺服器執行個體上定期呼叫 HTTP 方法，以判斷執行個體是否正常運作。若運作狀態檢查失敗，HAProxy 會停止將流量路由至執行個體，直到其透過手動方式或自動修復重新啟動。URL 前綴的預設值為 "/"，其對應到伺服器執行個體的首頁：http://*DNSName*/。

Health check method (運作狀態檢查方法)

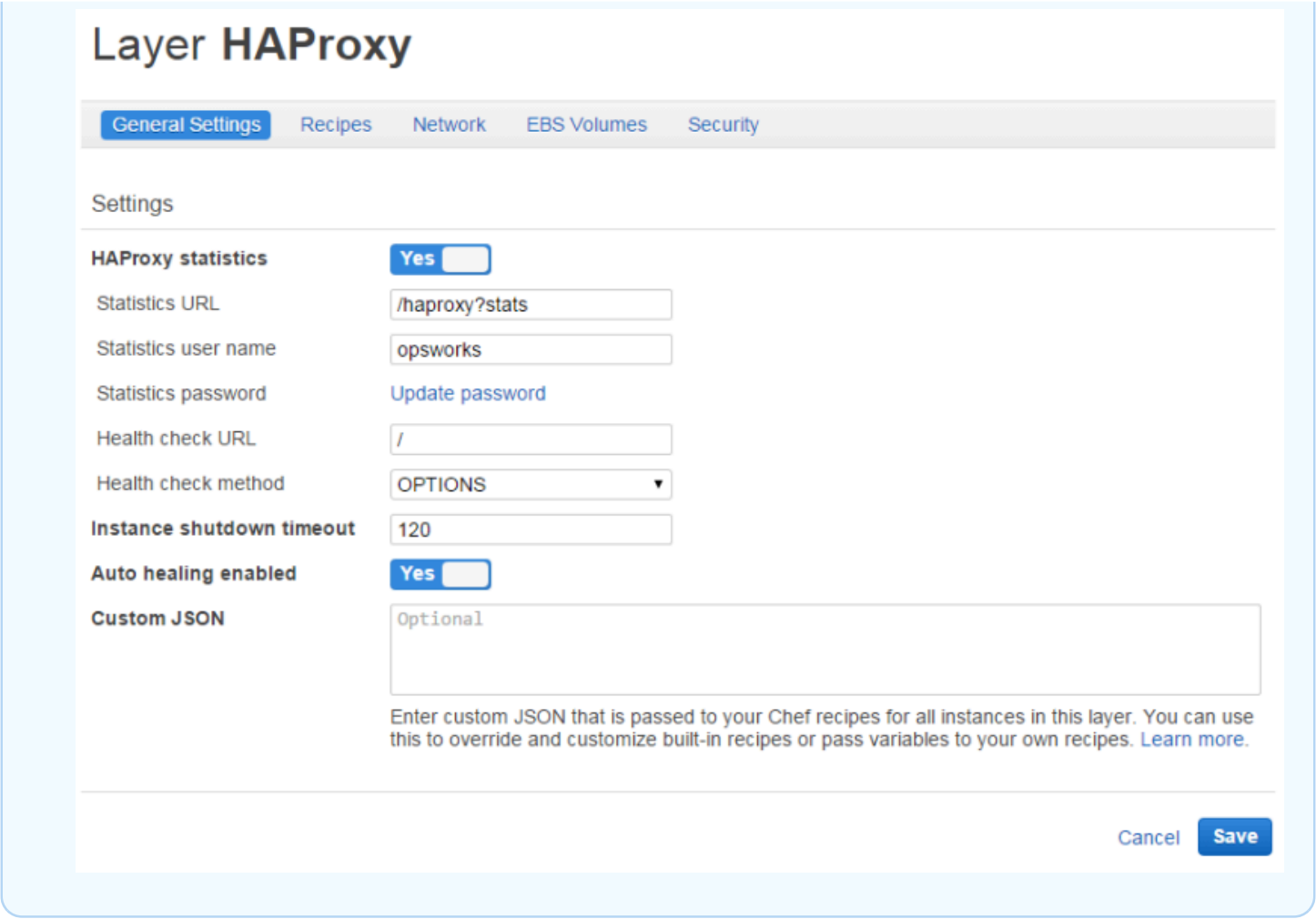
用於檢查執行個體是否正常運作的 HTTP 方法。預設值為 OPTIONS，您也可以指定 GET 或 HEAD。如需詳細資訊，請參閱 httpchk。

自訂安全群組

如果您選擇不自動將內建 AWS OpsWorks 的 Stacks 安全群組與 layer 建立關聯，則會顯示此設定。您必須指定要和 layer 關聯的安全群組有哪些。請確認群組具備正確的設定，可允許 layer 之間的流量。如需詳細資訊，請參閱建立新的堆疊。



> **Note**
>
> 記錄密碼以供日後使用； AWS OpsWorks Stacks 不允許您在建立 layer 之後檢視密碼。但是，您可以透過前往 layer 的 Edit (編輯) 頁面，然後按一下 General Settings (一般設定) 標籤上的 Update password (更新密碼) 來更新密碼。



HAProxy Layer 的運作方式

根據預設，HAProxy 會執行下列作業：

- 接聽 HTTP 和 HTTPS 連接埠的請求。

  您可以透過覆寫 Chef 組態範本 (`haproxy.cfg.erb`) 來設定 HAProxy 僅接聽 HTTP 或 HTTPS 連接埠。
- 將傳入流量路由至任何身為應用程式伺服器 layer 成員的執行個體。

  根據預設， AWS OpsWorks Stacks 會設定 HAProxy，將流量分散至屬於任何應用程式伺服器層成員的執行個體。例如，您可以同時擁有 Rails App Server 和 PHP App Server layer 的堆疊，HAProxy 主伺服器會將流量分配到兩個 layer 中的執行個體。您可以透過使用自訂配方，來設定預設路由。
- 路由流量至多個可用區域

若其中一個可用區域停機，負載平衡器會將傳入流量路由至其他區域內的執行個體，讓您的應用程式可繼續運作，而無須中斷。因此，建議的做法為將您的應用程式伺服器分散至多個可用區域。

- 定期在每個應用程式伺服器執行個體上執行指定的運作狀態檢查方法，以評估其運作狀態。

  如果方法未在指定的逾時期間內傳回，則假設執行個體失敗，HAProxy 會停止將請求路由到執行個體。 AWS OpsWorks Stacks 也提供自動取代失敗執行個體的方法。如需詳細資訊，請參閱使用自動修復。您可以在您建立 layer 時變更運作狀態檢查方法。
- 收集統計並選擇性的將他們顯示在網頁上。

> **Important**
>
> 若要使運作狀態檢查搭配預設的 OPTIONS 方法正常運作，您的應用程式必須傳回 2xx 或 3xx 狀態碼。

根據預設，當您將執行個體新增至 HAProxy layer 時， AWS OpsWorks Stacks 會為其指派一個彈性 IP 地址來代表應用程式，而該應用程式對世界是公有的。由於 HAProxy 執行個體的彈性 IP 地址是應用程式唯一公開的 URL，因此您不需要為基礎應用程式伺服器執行個體建立和管理公有網域名稱。您可以透過前往執行個體頁面，檢查執行個體的公有 IP 地址來取得地址，如下圖所示。後方跟隨 (EIP) 的地址即為彈性 IP 地址。如需彈性 IP 地址的詳細資訊，請參閱彈性 IP 地址 (EIP)。

當您停止 HAProxy 執行個體時， AWS OpsWorks Stacks 會保留彈性 IP 地址，並在重新啟動執行個體時將其重新指派給執行個體。如果您刪除 HAProxy 執行個體，根據預設， AWS OpsWorks Stacks 會刪除執行個體的 IP 地址。若要保留地址，請清除 Delete instance's Elastic IP (刪除執行個體的彈性 IP) 選項，如下圖所示。

此選項會影響您將新的執行個體新增至 layer 以取代遭刪除執行個體時會發生的情況。

- 如果您保留已刪除執行個體的彈性 IP 地址， AWS OpsWorks Stacks 會將該地址指派給新的執行個體。
- 否則， AWS OpsWorks Stacks 會將新的彈性 IP 地址指派給執行個體，而且您必須更新 DNS 註冊商設定，才能映射到新的地址。

當應用程式伺服器執行個體上線或離線時，無論是手動或由於自動擴展或自動修復，負載平衡器組態都必須更新，以將流量路由到目前一組線上執行個體。此任務會由 layer 的內建配方自動處理：

- 當新的執行個體上線時， AWS OpsWorks Stacks 會觸發設定生命週期事件。HAProxy layer 的內建設定配方會更新負載平衡器組態，因此也會將請求分配給任何新的應用程式伺服器執行個體。
- 當執行個體離線或執行個體未通過運作狀態檢查時， AWS OpsWorks Stacks 也會觸發設定生命週期事件。HAProxy 設定配方會更新負載平衡器的組態，只將流量路由至剩餘的線上執行個體。

最後，您也可以搭配 HAProxy layer 使用自訂網域。如需詳細資訊，請參閱使用自訂網域。

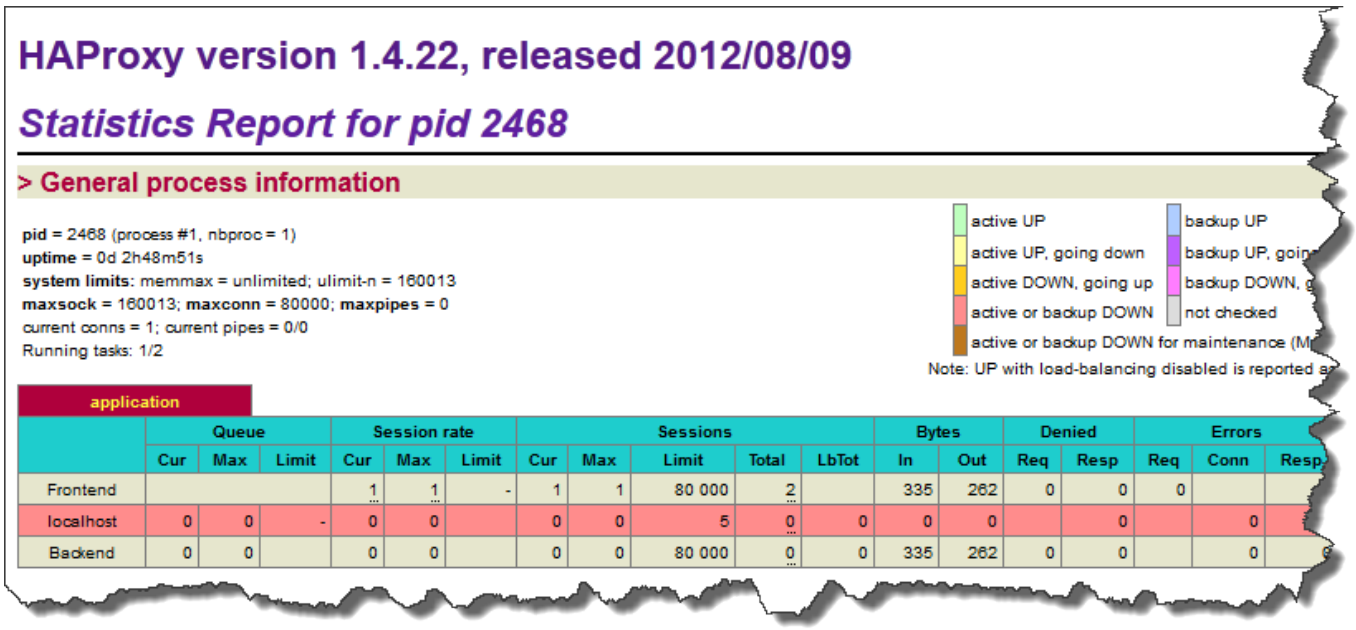
統計頁面

如果您已啟用統計資料頁面，HAProxy 會在指定的 URL 顯示包含各種指標的頁面。

檢視 HAProxy 統計資料

1. 從執行個體的詳細資訊頁面取得 HAProxy 執行個體的公有 DNS 名稱，並將其複製。
2. 在圖層頁面上，按一下 HAProxy 開啟圖層的詳細資訊頁面。

3. 從圖層詳細資訊取得統計資料 URL，並將其附加至公有 DNS 名稱。例如：**`http://ec2-54-245-102-172.us-west-2.compute.amazonaws.com/haproxy?stats`**. 至它。
4. 將先前步驟中的 URL 在您的瀏覽器中貼上，然後使用您在建立 layer 時指定的使用者名稱及密碼以開啟統計頁面。



MySQL Layer 參考

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此 layer 僅適用於 Linux 類型堆疊。

MySQL 層支援 MySQL，這是一種廣泛使用的關聯式資料庫管理系統。 AWS OpsWorks Stacks 會安裝最新的可用版本，這取決於作業系統。如果您新增 MySQL 執行個體，則會將所需的存取資訊提供給應用程式伺服器 layer。您必須撰寫自訂 Chef 配方來設定主要-主要或主要-從屬組態。

Short name: (簡短名稱：) db-master

相容性：MySQL 層與下列層相容：自訂、lb、memcached、 monitoring-master、 nodejs-app、php-app、rails-app 和 Web。

開放連接埠：MySQL layer 允許從堆疊的 Web 伺服器、自訂伺服器和 Rails、PHP 和 Node.js 應用程式伺服器公開存取連接埠 22 (SSH) 和所有連接埠。

Autoassign Elastic IP addresses: (自動指派彈性 IP 地址：) 預設為關閉

Default EBS volume: (預設 EBS 磁碟區：) 是，位於 `/vol/mysql`

Default security group: (預設安全群組：) AWS-OpsWorks-DB-Master-Server

組態：若要設定 MySQL layer，您必須指定下列項目：

- 根使用者密碼
- MySQL 引擎

Setup recipes: (安裝配方：)

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- mysql::server
- dependencies
- deploy::mysql

Configure recipes: (設定配方：)

- opsworks_ganglia::configure-client
- ssh_users
- agent_version

- deploy::mysql

Deploy recipes: (部署配方：)

- deploy::default
- deploy::mysql

Shutdown recipes: (關機配方：)

- opsworks_shutdown::default
- mysql::stop

安裝:

- AWS OpsWorks Stacks 使用執行個體的套件安裝程式，將 MySQL 及其日誌檔案安裝到其預設位置。如需詳細資訊，請參閱 MySQL 文件。

MySQL OpsWorks Layer

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此 layer 僅適用於 Chef 11 或更舊的 Linux 類型堆疊。

MySQL OpsWorks layer 為做為 MySQL 資料庫主節點的 Amazon EC2 執行個體提供藍圖。內建配方會為已部署到應用程式伺服器 layer 的每個應用程式建立資料庫。例如，如果您部署 PHP 應用程式 “myapp”，則配方會建立一個 “myapp” 資料庫。

MySQL layer 具有下列組態設定。

MySQL 根使用者密碼

(必要) 根使用者密碼。

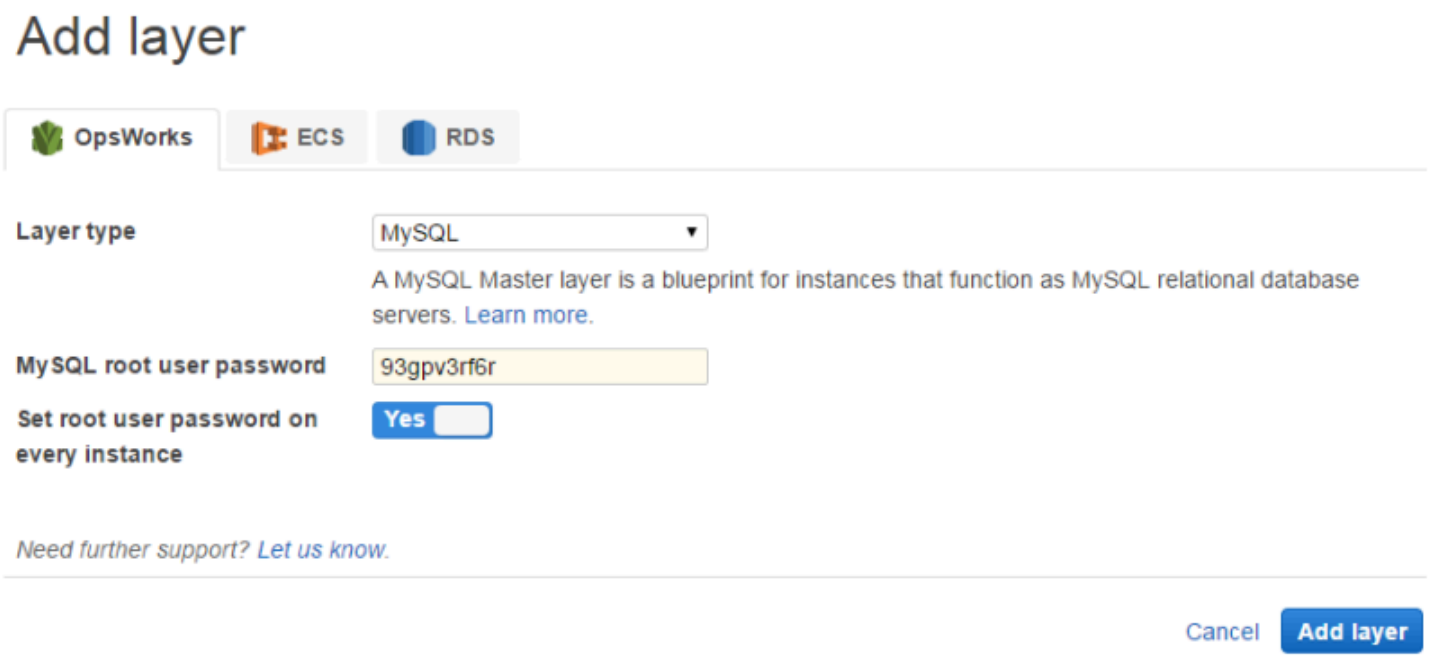
在每個執行個體上設定根使用者密碼

(選用) 無論根使用者密碼是否包含在堆疊中每個執行個體上安裝的堆疊組態和部署屬性中。預設設定為 Yes (是)。

如果您將此值設定為否，AWS OpsWorks Stacks 只會將根密碼傳遞給應用程式伺服器執行個體。

自訂安全群組

(選用) 要與 layer 建立關聯的自訂安全群組。如需詳細資訊，請參閱建立新的堆疊。



您可以將一或多個執行個體新增至 layer，每個執行個體代表一個單獨的 MySQL 資料庫主控。然後，您可以將執行個體連接至應用程式，會在應用程式的應用程式伺服器上安裝必要的連線資訊。然後，應用程式可以使用連線資訊連線到執行個體的資料庫伺服器。

應用程式伺服器 Layer 參考

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 支援數個不同的應用程式和靜態網頁伺服器。

主題

- AWS Flow (Ruby) Layer 參考
- Java App Server Layer 參考
- Node.js 應用程式伺服器層參考
- PHP App Server Layer 參考
- Rails App Server Layer 參考
- 靜態 Web 伺服器層參考

AWS Flow (Ruby) Layer 參考

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此 layer 僅適用於 Linux 類型堆疊。

AWS Flow (Ruby) layer 為託管 Amazon Simple Workflow Service 活動和工作流程工作者的執行個體提供藍圖。

Short name: (簡短名稱：) aws-flow-ruby

相容性：AWS Flow (Ruby) layer 與 PHP App Server、MySQL、Memcached、Ganglia 和自訂 layer 相容。

Open ports: (開放連接埠：) 無。

IAM 角色：aws-opsworks-ec2-role-with-swf 是 AWS OpsWorks Stacks 應要求為您建立的標準 AWS Flow (Ruby) 角色。

Autoassign Elastic IP addresses: (自動指派彈性 IP 地址：) 預設為關閉

Default EBS Volume: (預設 EBS 磁碟區：) 否

Default security group: (預設安全群組：) AWS-OpsWorks-AWS-Flow-Ruby-Server

Setup recipes: (安裝配方：)

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- opsworks_aws_flow_ruby::setup

Configure recipes: (設定配方：)

- opsworks_ganglia::configure-client
- ssh_users
- mysql::client
- agent_version
- opsworks_aws_flow_ruby::configure

Deploy recipes: (部署配方：)

- deploy::default

- deploy::aws-flow-ruby

Undeploy recipes: (解除部署配方：)

- deploy::aws-flow-ruby-undeploy

Shutdown recipes: (關機配方：)

- opsworks_shutdown::default

Java App Server Layer 參考

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此 layer 僅適用於 Linux 類型堆疊。

Java App Server layer 支援 Apache Tomcat 7.0 應用程式伺服器。

Short name: (簡短名稱：) java-app

相容性：Java App Server layer 與下列 layer 相容：Custom、db-master 和 memcached。

開放連接埠：Java App Server layer 允許公開存取連接埠 22 (SSH)、80 (HTTP)、443 (HTTPS) 和來自負載平衡器的所有連接埠。

Autoassign Elastic IP addresses: (自動指派彈性 IP 地址：) 預設為關閉

Default EBS Volume: (預設 EBS 磁碟區：) 否

Default security group: (預設安全群組：) AWS-OpsWorks-Java-App-Server

Setup recipes: (安裝配方：)

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- opsworks_java::setup

Configure recipes: (設定配方：)

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- opsworks_java::configure

Deploy recipes: (部署配方：)

- deploy::default
- deploy::java

Undeploy recipes: (解除部署配方：)

- deploy::java-undeploy

Shutdown recipes: (關機配方：)

- opsworks_shutdown::default
- deploy::java-stop

安裝:

- Tomcat 會安裝至 `/usr/share/tomcat7`。
- 如需如何產生日誌檔案的詳細資訊，請參閱 Logging in Tomcat。

Node.js 應用程式伺服器層參考

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此 layer 僅適用於 Linux 類型堆疊。

Node.js App Server layer 支援 Node.js 應用程式伺服器，這是實作高可擴展性網路應用程式伺服器的平台。程式以 JavaScript 撰寫，並使用事件驅動的非同步 I/O，將額外負荷降到最低，且將可擴展性提到最高。

Short name: (簡短名稱：) nodejs-app

相容性：Node.js App Server layer 與下列層相容：自訂、db-master、memcached 和 monitoring-master。

開放連接埠：Node.js App Server layer 允許從負載平衡器公開存取連接埠 22 (SSH)、80 (HTTP)、443 (HTTPS) 和所有連接埠。

Autoassign Elastic IP addresses: (自動指派彈性 IP 地址：) 預設為關閉

Default EBS volume: (預設 EBS 磁碟區：) 否

Default security group: (預設安全群組：) AWS-OpsWorks-nodejs-App-Server

Setup recipes: (安裝配方：)

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies

- ebs
- opsworks_ganglia::client
- opsworks_nodejs
- opsworks_nodejs::npm

Configure recipes: (設定配方：)

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- opsworks_nodejs::configure

Deploy recipes: (部署配方：)

- deploy::default
- opsworks_nodejs
- opsworks_nodejs::npm
- deploy::nodejs

Undeploy recipes: (解除部署配方：)

- deploy::nodejs-undeploy

Shutdown recipes: (關機配方：)

- opsworks_shutdown::default
- deploy::nodejs-stop

安裝:

- Node.js 會安裝至 `/usr/local/bin/node`。
- 如需如何產生日誌檔案的詳細資訊，請參閱 Nodejitsu 網站上的 How to log in node.js。

Node.js application configuration: (Node.js 應用程式組態：)

- Node.js 所執行的主要檔案必須命名為 `server.js` 並且位在已部署應用程式的根目錄中。
- Node.js 應用程式必須設定為接聽連接埠 80，適用時，也可以接聽連接埠 443。

> **Note**
>
> 執行 Express 的 Node.js 應用程式普遍使用下列程式碼來設定接聽連接埠，其中 `process.env.PORT` 代表預設連接埠並解析為 80：
>
> ```
> app.set('port', process.env.PORT || 3000);
> ```
>
> 使用 AWS OpsWorks Stacks，您必須明確指定連接埠 80，如下所示：
>
> ```
> app.set('port', 80);
> ```

PHP App Server Layer 參考

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此 layer 僅適用於 Linux 類型堆疊。

PHP App Server layer 透過搭配 mod_php 使用 Apache2 來支援 PHP 應用程式伺服器。

Short name: (簡短名稱：) php-app

相容性：PHP App Server layer 與下列 layer 相容：Custom、db-master、memcached、 monitoring-master 和 rails-app。

開放連接埠：PHP App Server layer 允許公開存取連接埠 22 (SSH)、80 (HTTP)、443 (HTTPS) 和來自負載平衡器的所有連接埠。

Autoassign Elastic IP addresses: (自動指派彈性 IP 地址：) 預設為關閉

Default EBS volume: (預設 EBS 磁碟區：) 否

Default security group: (預設安全群組：) AWS-OpsWorks-PHP-App-Server

Setup recipes: (安裝配方：)

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- mysql::client
- dependencies
- mod_php5_apache2

Configure recipes: (設定配方：)

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- mod_php5_apache2::php
- php::configure

Deploy recipes: (部署配方：)

- deploy::default
- deploy::php

Undeploy recipes: (解除部署配方：)

- deploy::php-undeploy

Shutdown recipes: (關機配方：)

- opsworks_shutdown::default
- apache2::stop

Installation: (安裝：)

- AWS OpsWorks Stacks 使用執行個體的套件安裝程式，將 Apache2、mod_php 和相關聯的日誌檔案安裝到其預設位置。如需安裝的詳細資訊，請參閱 Apache。如需記錄的詳細資訊，請參閱 Log Files。

Rails App Server Layer 參考

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此 layer 僅適用於 Linux 類型堆疊。

Rails App Server layer 支援 Ruby on Rails 應用程式伺服器。

Short name: (簡短名稱：) rails-app

相容性：Rails App Server layer 與下列 layer 相容：Custom、db-master、memcached、 monitoring-master、php-app。

連接埠：Rails App Server layer 允許從負載平衡器公開存取連接埠 22 (SSH)、80 (HTTP)、443 (HTTPS) 和所有連接埠。

Autoassign Elastic IP addresses: (自動指派彈性 IP 地址：) 預設為關閉

Default EBS volume: (預設 EBS 磁碟區：) 否

Default security group: (預設安全群組：) AWS-OpsWorks-Rails-App-Server

組態：若要設定 Rails App Server layer，您必須指定下列項目：

- Ruby 版本
- Rails 堆疊
- Rubygems 版本
- 是否安裝和管理 Bundler
- Bundler 版本

Setup recipes: (安裝配方：)

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- apache2 apache2::mod_deflate
- passenger_apache2
- passenger_apache2::mod_rails
- passenger_apache2::rails

Configure recipes: (設定配方：)

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- rails::configure

Deploy recipes: (部署配方：)

- deploy::default
- deploy::rails

Undeploy recipes: (解除部署配方：)

- deploy::rails-undeploy

Shutdown recipes: (關機配方：)

- opsworks_shutdown::default
- apache2::stop

安裝:

- AWS OpsWorks Stacks 使用執行個體的套件安裝程式，將具有 mod_passenger、mod_rails 和相關聯日誌檔案的 Apache2 安裝至其預設位置。如需安裝的詳細資訊，請參閱 Phusion Passenger。如需記錄的詳細資訊，請參閱 Log Files。

靜態 Web 伺服器層參考

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此 layer 僅適用於 Linux 類型堆疊。

靜態 Web Server layer 提供靜態 HTML 頁面，其中可包含用戶端程式碼，例如 JavaScript。它是以 Nginx 為基礎，這是一種開放原始碼 HTTP、反向代理和郵件代理伺服器。

Short name: (簡短名稱：) web

相容性：靜態 Web 伺服器層與下列層相容：自訂、db-master、memcached。

開放連接埠：靜態 Web 伺服器層允許從負載平衡器公開存取連接埠 22 (SSH)、80 (HTTP)、443 (HTTPS) 和所有連接埠。

Autoassign Elastic IP addresses: (自動指派彈性 IP 地址：) 預設為關閉

Default EBS volume: (預設 EBS 磁碟區：) 否

Default security group: (預設安全群組：) AWS-OpsWorks-Web-Server

Setup recipes: (安裝配方：)

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- nginx

Configure recipes: (設定配方：)

- opsworks_ganglia::configure-client
- ssh_users
- agent_version

Deploy recipes: (部署配方：)

- deploy::default
- deploy::web

Undeploy recipes: (解除部署配方：)

- deploy::web-undeploy

Shutdown recipes: (關機配方：)

- opsworks_shutdown::default

- nginx::stop

安裝:

- Nginx 會安裝至 `/usr/sbin/nginx`。
- Nginx 日誌檔案位於 `/var/log/nginx`。

應用程式伺服器 Layer

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 這些 layer 僅供 Chef 11 和更早版本的 Linux 式堆疊使用。

AWS OpsWorks Stacks 支援數個不同的應用程式伺服器，其中「應用程式」包含靜態網頁。每種類型的伺服器都有單獨的 AWS OpsWorks Stacks layer，其內建配方可處理在每個 layer 的執行個體上安裝應用程式伺服器和任何相關套件、部署應用程式等。例如，Java App Server layer 會安裝數個套件，包括 Apache、Tomcat 和 OpenJDK，並將 Java 應用程式部署到 layer 的每個執行個體。

以下是使用應用程式伺服器 layer 的基本程序：

1. 建立一種可用的 App Server (應用程式伺服器) layer 類型。
2. 新增一或多個執行個體至 layer。
3. 建立應用程式並將其部署至執行個體。如需詳細資訊，請參閱應用程式。
4. (選用) 如果 layer 有多個執行個體，您可以新增負載平衡器，以將傳入流量分配到執行個體。如需詳細資訊，請參閱HAProxy AWS OpsWorks Stacks Layer。

主題

- AWS Flow (Ruby) Layer

- Java 應用程式伺服器 AWS OpsWorks 堆疊層
- Node.js 應用程式伺服器 AWS OpsWorks 堆疊層
- PHP 應用程式伺服器 AWS OpsWorks 堆疊層
- Rails 應用程式伺服器 AWS OpsWorks 堆疊層
- 靜態 Web 伺服器 AWS OpsWorks 堆疊層

AWS Flow (Ruby) Layer

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此 layer 僅適用於 Linux 類型堆疊。

AWS Flow (Ruby) layer 是 AWS OpsWorks Stacks layer，可為託管 Amazon SWF 活動和工作流程工作者的執行個體提供藍圖。工作者的實作方式是使用適用於 Ruby 的 AWS Flow Framework，這是一種程式設計架構，可簡化實作分散式非同步應用程式的程序，同時提供 Amazon SWF 的所有優點。這項功能適用於實作應用程式來解決廣泛情境，包括商務程序、媒體編碼、長時間執行的任務和背景處理。

AWS Flow (Ruby) layer 包含下列組態設定。

RubyGems 版本

框架的 Gem 版本。

Bundler 版本

Bundler 版本。

EC2 Instance profile (EC2 執行個體描述檔)

layer 執行個體要使用的使用者定義 Amazon EC2 執行個體描述檔。此設定檔必須授予在 layer 執行個體上執行的應用程式存取 Amazon SWF 的許可。

如果您的帳戶沒有適當的設定檔，您可以選取具有 SWF 存取權的新設定檔，讓 AWS OpsWorks Stacks 更新 的設定檔，或者您可以使用 IAM 主控台自行更新。您接著可以使用針對所有後續 AWS Flow layer 之更新的描述檔。以下是如何使用 IAM 主控台建立設定檔的簡短描述。如需詳細資訊，請參閱 Amazon Simple Workflow Service 中的 Identity and Access Management。

建立 AWS Flow (Ruby) 執行個體的設定檔

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格中選擇政策，然後選擇建立政策以建立新的客戶受管政策。
3. 針對服務，選擇 SWF。
4. 針對動作，選擇所有 SWF 動作 (swf：*)。
5. 針對 Amazon Resource Name (ARN)，輸入指定工作者可存取哪些 Amazon SWF 網域的 ARN。選擇 **`All resources`**以提供所有網域的存取權。
6. 選擇下一步。
7. 或者，輸入標籤以識別政策。
8. 選擇下一步。
9. 完成後，請選擇建立政策。
10. 在導覽窗格中選擇角色，然後選擇建立角色。
11. 指定角色名稱，然後選擇下一步。建立角色之後，就無法變更名稱。
12. 選擇 AWS 服務，然後選擇 EC2。
13. 選擇下一步。
14. 從許可政策清單中，選擇您先前建立的政策。
15. 選擇下一步。
16. 輸入角色名稱，然後選擇 Create role (建立角色)。建立角色之後，就無法變更名稱。
17. 在 Stacks 中建立 AWS Flow (Ruby) AWS OpsWorks layer 時指定此設定檔。

Java 應用程式伺服器 AWS OpsWorks 堆疊層

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此 layer 僅適用於 Linux 類型堆疊。

Java App Server layer 是 AWS OpsWorks Stacks layer，為做為 Java 應用程式伺服器的執行個體提供藍圖。此層以 Apache Tomcat 7.0 和 Open JDK 7. AWS OpsWorks Stacks 為基礎，也會安裝 Java 連接器程式庫，讓 Java 應用程式使用 JDBC `DataSource` 物件連線到後端資料存放區。

Installation (安裝)：Tomcat 安裝在 `/usr/share/tomcat7` 中。

Add Layer (新增 Layer) 頁面提供下列組態選項：

Java VM 選項

您可以使用此設定來指定自訂 Java VM 選項；沒有預設選項。例如，一組常見選項為 `-Djava.awt.headless=true -Xmx128m -XX:+UseConcMarkSweepGC`。如果您使用 Java VM 選項，請確定您已傳遞一組有效的選項； AWS OpsWorks Stacks 不會驗證字串。如果您嘗試傳遞無效的選項，則通常無法啟動 Tomcat 伺服器，進而導致設定失敗。如果發生此情況，您可以檢查執行個體的設定 Chef 日誌以了解詳細資訊。如需如何檢視和解釋 Chef 日誌的詳細資訊，請參閱Chef 日誌。

自訂安全群組

如果您選擇不自動將內建 Stacks AWS OpsWorks 安全群組與 layer 建立關聯，則會顯示此設定。您必須指定要和 layer 關聯的安全群組有哪些。如需詳細資訊，請參閱建立新的堆疊。

Elastic Load Balancer

您可以將 Elastic Load Balancing 負載平衡器連接至 layer 的執行個體。如需詳細資訊，請參閱Elastic Load Balancing Layer。

您可以使用自訂 JSON 或自訂屬性檔案，來指定其他組態設定。如需詳細資訊，請參閱自訂組態。

> **Important**
>
> 如果您的 Java 應用程式使用 SSL，則建議您盡可能停用 SSLv3 來處理 CVE-2014-3566 中所述的漏洞。如需詳細資訊，請參閱停用 Apache 伺服器的 SSLv3。

主題

- 停用 Apache 伺服器的 SSLv3
- 自訂組態
- 部署 Java 應用程式

停用 Apache 伺服器的 SSLv3

若要停用 SSLv3，您必須修改 Apache 伺服器之 `ssl.conf` 檔案的 `SSLProtocol` 設定。若要這樣做，您必須覆寫內建 apache2 技術指南的`ssl.conf.erb`範本檔案，Java App Server 層的設定配方會使用此檔案來建立 `ssl.conf`。詳細資訊取決於您針對 layer 執行個體所指定的作業系統。以下摘要說明 Amazon Linux 和 Ubuntu 系統所需的修改。會自動停用 Red Hat Enterprise Linux (RHEL) 系統的 SSLv3。如需如何覆寫內建範本的詳細資訊，請參閱使用自訂範本 。

Amazon Linux

這些作業系統的 `ssl.conf.erb` 檔案位於 apache2 技術指南的 `apache2/templates/default/mods` 目錄中。以下顯示內建檔案的相關部分。

```
...
#SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

# enable only secure protocols: SSLv3 and TLSv1.2, but not SSLv2
SSLProtocol all -SSLv2
</IfModule>
```

覆寫 `ssl.conf.erb` 和修改 `SSLProtocol` 設定，如下所示。

```
...
#SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

# enable only secure protocols: SSLv3 and TLSv1.2, but not SSLv2
SSLProtocol all -SSLv3 -SSLv2
</IfModule>
```

Ubuntu 14.04 LTS

此作業系統的 `ssl.conf.erb` 檔案位於 apache2 技術指南的 `apache2/templates/ubuntu-14.04/mods` 目錄中。以下顯示內建檔案的相關部分。

```
...
# The protocols to enable.
# Available values: all, SSLv3, TLSv1.2
# SSL v2 is no longer supported
SSLProtocol all
...
```

請將此設定變更為下列內容。

```
...
# The protocols to enable.
# Available values: all, SSLv3, TLSv1.2
# SSL v2 is no longer supported
SSLProtocol all -SSLv3 -SSLv2
...
```

自訂組態

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 會將其他組態設定公開為內建屬性，這些都是在`opsworks_java`命名空間中。您可以使用自訂 JSON 或自訂屬性檔案來覆寫內建屬性，並指定自訂值。例如，JVM 和 Tomcat 版本由內建 `jvm_version` 和 `java_app_server_version` 屬性所代表，而這兩者都設定為 7。您可以使用自訂 JSON 或自訂屬性檔案，將其中一個或兩者設定為 6。下列範例使用自訂 JSON 將兩個屬性都設定為 6：

```
{
  "opsworks_java": {
    "jvm_version": 6,
    "java_app_server_version" : 6
  }
```

```
}
```

如需詳細資訊，請參閱使用自訂 JSON。

另一個自訂組態的範例是覆寫 `use_custom_pkg_location`、`custom_pkg_location_url_debian` 和 `custom_pkg_location_url_rhel` 屬性來安裝自訂 JDK。

> **Note**
>
> 如果您覆寫內建技術指南，則需要自行更新這些元件。

如需屬性和其覆寫方式的詳細資訊，請參閱覆寫屬性。如需內建屬性的清單，請參閱opsworks_java 屬性。

部署 Java 應用程式

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

下列主題說明如何將應用程式部署到 Java App Server layer 的執行個體。這些範例是針對 JSP 應用程式，但您可以使用基本上相同的程序，來安裝其他類型的 Java 應用程式。

您可以從任何支援的儲存庫部署 JSP 頁面。如果您想要部署 WAR 檔案，請注意 Stacks AWS OpsWorks 會自動從 Amazon S3 或 HTTP 封存檔擷取部署的 WAR 檔案，但不會從 Git 或 Subversion 儲存庫擷取。如果您想要將 Git 或 Subversion 用於 WAR 檔案，則可以執行下列其中一項：

- 將擷取的封存檔存放至儲存庫。
- 將 WAR 檔案存放至儲存庫，並使用 Chef 部署勾點來擷取封存檔，如下列範例所述。

您可以使用 Chef 部署勾點，在四個部署階段中的任何一階段，於執行個體上執行使用者提供的 Ruby 應用程式。應用程式名稱可決定階段。下列範例是名為 `before_migrate.rb` 的 Ruby 應用程式，可擷取已從 Git 或 Subversion 儲存庫所部署的 WAR 檔案。該名稱會建立應用程式與 Checkout 部署勾

點的關聯，以在部署操作開始時、檢查程式碼之後但在遷移之前執行。如需如何使用此範例的詳細資訊，請參閱使用 Chef 部署勾點。

```
::Dir.glob(::File.join(release_path, '*.war')) do |archive_file|
  execute "unzip_#{archive_file}" do
    command "unzip #{archive_file}"
    cwd release_path
  end
end
```

> **Note**
>
> 在您將更新部署至 JSP 應用程式時，Tomcat 可能會無法識別更新，而繼續執行現有的應用程式版本。舉例來說，這可能會在您將應用程式做為僅包含 JSP 頁面的 .zip 檔案部署時發生。若要確保 Tomcat 執行的是最近部署的版本，專案的根目錄應包括內含 `web.xml` 檔案的 WEB-INF 目錄。`web.xml` 檔案可包含各種內容，但以下內容便足以確保 Tomcat 識別更新及執行目前部署的應用程式版本。您不需要為每次更新變更版本。若版本沒有變更，Tomcat 將會識別更新。
>
> ```
> <context-param>
>   <param-name>appVersion</param-name>
>   <param-value>0.1</param-value>
> </context-param>
> ```

主題

- 部署 JSP 應用程式
- 使用後端資料庫部署 JSP 應用程式

部署 JSP 應用程式

若要部署 JSP 應用程式，請指定名稱和儲存庫資訊。您也可以選擇指定網域和 SSL 設定。如需如何建立應用程式的詳細資訊，請參閱新增應用程式。下列程序說明如何從公有 Amazon S3 封存建立和部署簡單的 JSP 頁面。如需如何使用其他儲存庫類型的資訊，包括私有 Amazon S3 封存，請參閱 應用程式來源。

下列範例所顯示的 JSP 頁面只會顯示一些系統資訊。

```
<%@ page import="java.net.InetAddress" %>
<html>
<body>
<%
    java.util.Date date = new java.util.Date();
    InetAddress inetAddress = InetAddress.getLocalHost();
%>
The time is
<%
    out.println( date );
    out.println("<br>Your server's hostname is "+inetAddress.getHostName());
%>
<br>
</body>
</html>
```

> **Note**
>
> 下列程序假設您已熟悉建立堆疊、將執行個體新增至 layer 等等的基本知識。如果您是初次使用 AWS OpsWorks Stacks，您應該會先看到 Chef 11 Linux 堆疊入門。

從 Amazon S3 封存檔部署 JSP 頁面

1. 使用 Java App Server layer 建立堆疊，將全年無休的執行個體新增至 layer，然後啟動它。
2. 將程式碼複製至名為 `simplejsp.jsp` 的檔案，並將檔案放入名為 `simplejsp` 的資料夾中，然後建立資料夾的 `.zip` 封存檔。名稱可以是任意名稱；您可以使用您想要的任何檔案或資料夾名稱。您也可以使用其他類型的封存檔，包括 gzip、bzip2、tarball 或 Java WAR 檔案。請注意，AWS OpsWorks Stacks 不支援未壓縮的 tarball。若要部署多個 JSP 頁面，請將它們包括在相同的封存檔中。
3. 將封存上傳到 Amazon S3 儲存貯體並將檔案設為公有。複製檔案的 URL，供日後使用。如需如何建立儲存貯體以及上傳檔案的詳細資訊，請參閱開始使用 Amazon Simple Storage Service。
4. 新增應用程式至堆疊，然後指定下列設定：
   - 名稱 – `SimpleJSP`
   - App type (應用程式類型) – `Java`

- Repository type (儲存庫類型) – `Http Archive`
- 儲存庫 URL – 封存檔案的 Amazon S3 URL。

針對剩餘設定使用預設值，然後按一下 Add App (新增應用程式) 以建立應用程式。

5. 將應用程式部署到 Java App Server 執行個體。

您現在可以前往應用程式的 URL，以及檢視應用程式。如果您尚未指定網域，則可以使用執行個體的公有 IP 地址或其公有 DNS 名稱來建構 URL。若要取得執行個體的公有 IP 地址或公有 DNS 名稱，請前往 AWS OpsWorks Stacks 主控台，然後按一下執行個體頁面上的執行個體名稱以開啟其詳細資訊頁面。

其餘 URL 取決於應用程式的簡短名稱，這是 AWS OpsWorks Stacks 從您在建立應用程式時指定的應用程式名稱產生的小寫名稱。例如，SimpleJSP 的簡短名稱是 simplejsp。您可以從其詳細資訊頁面中取得應用程式的簡短名稱。

- 如果簡短名稱是 `root`，則您可以使用 `http://`*`public_DNS`*`/`*`appname`*`.jsp` 或 `http://`*`public_IP`*`/`*`appname`*`.jsp`。
- 否則，您可以使用 `http://`*`public_DNS`*`/`*`app_shortname`*`/`*`appname`*`.jsp` 或 `http://`*`public_IP`*`/`*`app_shortname`*`/`*`appname`*`.jsp`。

如果您已指定應用程式的網域，則 URL 為 `http://`*`domain`*`/`*`appname`*`.jsp`。

此範例的 URL 會類似 `http://192.0.2.0/simplejsp/simplejsp.jsp`。

如果您想要將多個應用程式部署至相同的執行個體，則不應該使用 `root` 做為簡短名稱。這可能會導致 URL 衝突，進而防止應用程式正常運作。相反地，請將不同的網域名稱指派給每個應用程式。

使用後端資料庫部署 JSP 應用程式

JSP 頁面可以使用 JDBC `DataSource` 物件以連線至後端資料庫。您可以使用上節中的程序來建立和部署這類應用程式，而且只要再一個額外的步驟就可以設定連線。

下列 JSP 頁面顯示如何連線至 `DataSource` 物件。

```
<html>
  <head>
    <title>DB Access</title>
  </head>
```

```
<body>
  <%@ page language="java" import="java.sql.*,javax.naming.*,javax.sql.*" %>
  <%
    StringBuffer output = new StringBuffer();
    DataSource ds = null;
    Connection con = null;
    Statement stmt = null;
    ResultSet rs = null;
    try {
      Context initCtx = new InitialContext();
      ds = (DataSource) initCtx.lookup("java:comp/env/jdbc/mydb");
      con = ds.getConnection();
      output.append("Databases found:<br>");
      stmt = con.createStatement();
      rs = stmt.executeQuery("show databases");
      while (rs.next()) {
        output.append(rs.getString(1));
        output.append("<br>");
      }
    }
    catch (Exception e) {
      output.append("Exception: ");
      output.append(e.getMessage());
      output.append("<br>");
    }
    finally {
      try {
        if (rs != null) {
          rs.close();
        }
        if (stmt != null) {
          stmt.close();
        }
        if (con != null) {
          con.close();
        }
      }
      catch (Exception e) {
        output.append("Exception (during close of connection): ");
        output.append(e.getMessage());
        output.append("<br>");
      }
    }
  %>
```

```
    <%= output.toString() %>
  </body>
</html>
```

AWS OpsWorks Stacks 會建立和初始化DataSource物件，將其繫結至邏輯名稱，並使用 Java 命名和目錄界面 (JNDI) 命名服務註冊名稱。完整的邏輯名稱為 `java:comp/env/`*`user-assigned-name`*。您必須指定名稱中由使用者指派的部分，方法是將自訂 JSON 屬性新增至堆疊組態和部署屬性來定義 `['opsworks_java']['datasources']` 屬性，如下所述。

部署連線至 MySQL 資料庫的 JSP 頁面

1. 使用 Java App Server layer 建立堆疊，將全年無休的執行個體新增至每個 layer，然後啟動它。
2. 將資料庫 layer 新增至堆疊。詳細資訊取決於您使用的資料庫。

   若要使用 MySQL 執行個體做為範例，請將 MySQL layer 新增至堆疊，將全年無休的執行個體新增至 layer，然後啟動它。

   若要使用 Amazon RDS (MySQL) 執行個體做為範例：

   - 指定執行個體的 MySQL 資料庫引擎。
   - 將 AWS-OpsWorks-DB-Master-Server (***`security_group_id`***) 和 AWS-OpsWorks-Java-App-Server (***`security_group_id`***) 安全群組指派給執行個體。當您在區域中建立第一個堆疊時，AWS OpsWorks Stacks 會為您建立這些安全群組。
   - 建立名為 `simplejspdb` 的資料庫。
   - 確定主要使用者名稱和密碼未包含可能會導致 Tomcat 錯誤的 & 或其他字元。

     具體而言，在啟動期間，Tomcat 必須剖析 Web 應用程式內容檔案，而此檔案是一種 XML 檔案，其中包括主要密碼和使用者名稱。如果任一字串包括 & 字元，則 XML 剖析器會將它視為格式異常的 XML 實體，並拋出剖析例外狀況，以防止啟動 Tomcat。如需 Web 應用程式內容檔案的詳細資訊，請參閱tomcat::context。
   - 將 MySQL 驅動程式新增至 Java App Server layer。
   - 向堆疊註冊 RDS 執行個體。

   如需如何搭配 Stacks 使用 Amazon RDS AWS OpsWorks 執行個體的詳細資訊，請參閱 Amazon RDS 服務層。

3. 將範例程式碼複製至名為 simplejspdb.jsp 的檔案，並將檔案放入名為 simplejspdb 的資料夾中，然後建立資料夾的 .zip 封存檔。名稱可以是任意名稱；您可以使用您想要的任何檔案或資料夾名稱。您也可以使用其他類型的封存檔，包括 gzip、bzip2 或 tarball。若要部署多個 JSP 頁面，請將它們包括在相同的封存檔中。如需如何從其他儲存庫類型部署應用程式的資訊，請參閱應用程式來源。
4. 將封存上傳到 Amazon S3 儲存貯體並將檔案設為公有。複製檔案的 URL，供日後使用。如需如何建立儲存貯體以及上傳檔案的詳細資訊，請參閱開始使用 Amazon Simple Storage Service。
5. 新增應用程式至堆疊，然後指定下列設定：

   - 名稱 – SimpleJSPDB
   - App type (應用程式類型) – Java
   - 資料來源類型 – OpsWorks （適用於 MySQL 執行個體） 或 RDS （適用於 Amazon RDS 執行個體）。
   - 資料庫執行個體 – 您先前建立的 MySQL 執行個體，通常名為 db-master1(mysql)，或名為 ***DB_instance_name*** (mysql) 的 Amazon RDS 執行個體。
   - Database name (資料庫名稱) – simplejspdb。
   - Repository type (儲存庫類型) – Http Archive
   - 儲存庫 URL – 封存檔案的 Amazon S3 URL。

   針對剩餘設定使用預設值，然後按一下 Add App (新增應用程式) 以建立應用程式。
6. 將下列自訂 JSON 屬性新增至堆疊組態屬性，其中 simplejspdb 是應用程式的簡短名稱。

   ```
   {
     "opsworks_java": {
       "datasources": {
         "simplejspdb": "jdbc/mydb"
       }
     }
   }
   ```

   AWS OpsWorks Stacks 使用此映射來產生具有必要資料庫資訊的內容檔案。

   如需如何將自訂 JSON 屬性新增至堆疊組態屬性的詳細資訊，請參閱使用自訂 JSON。
7. 將應用程式部署到 Java App Server 執行個體。

您現在可以使用應用程式的 URL 來檢視應用程式。如需如何建構 URL 的描述，請參閱部署 JSP 應用程式。

此範例的 URL 會類似 `http://192.0.2.0/simplejspdb/simplejspdb.jsp`。

> **Note**
>
> `datasources` 屬性可以包含多個屬性。每個屬性的名稱都會有應用程式簡短名稱，並設定為邏輯名稱中由使用者指派的適當部分。如果您有多個應用程式，則可以使用不同的邏輯名稱，這需要與下面類似的自訂 JSON。
>
> ```
> {
>   "opsworks_java": {
>     "datasources": {
>       "myjavaapp": "jdbc/myappdb",
>       "simplejsp": "jdbc/myjspdb",
>       ...
>     }
>   }
> }
> ```

Node.js 應用程式伺服器 AWS OpsWorks 堆疊層

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此 layer 僅適用於 Linux 類型堆疊。

Node.js App Server layer 是 AWS OpsWorks Stacks layer，為做為 Node.js 應用程式伺服器的執行個體提供藍圖。 AWS OpsWorks Stacks 也會安裝 Express，因此 layer 的執行個體同時支援標準和 Express 應用程式。

Installation (安裝)：Node.js 安裝在 `/usr/local/bin/node` 中。

Add Layer (新增 Layer) 頁面提供下列組態選項：

Node.js 版本

如需目前所支援版本的清單，請參閱AWS OpsWorks Stacks 作業系統。

自訂安全群組

如果您選擇不自動將內建 Stacks AWS OpsWorks 安全群組與 layer 建立關聯，則會顯示此設定。您必須指定要和 layer 關聯的安全群組有哪些。如需詳細資訊，請參閱建立新的堆疊。

Elastic Load Balancer

您可以將 Elastic Load Balancing 負載平衡器連接至 layer 的執行個體。

> **Important**
>
> 如果您的 Node.js 應用程式使用 SSL，則建議您盡可能停用 SSLv3 來處理 CVE-2015-8027 中所述的漏洞。若要這麼做，您必須將 Node.js version (Node.js 版本) 設定為 `0.12.9`。

部署 Node.js 應用程式

如需如何實作 AWS OpsWorks Stacks 簡單 Node.js 應用程式並將其部署至堆疊的詳細演練，請參閱建立您的第一個 Node.js 堆疊。一般而言， AWS OpsWorks Stacks 的 Node.js 應用程式應該會符合下列條件：

- 主要檔案必須命名為 `server.js` 並且位在已部署應用程式的根目錄中。
- Express 應用程式必須在應用程式的根目錄中包括 `package.json` 檔案。
- 根據預設，應用程式必須接聽連接埠 80 (HTTP) 或連接埠 443 (HTTPS)。

  您可以監聽其他連接埠，但 Node.js App Server layer 的內建安全群組 AWS-OpsWorks-nodejs-App-Server 僅允許傳入使用者流量至連接埠 80、443 和 22 (SSH)。若要允許傳入使用者流量到其他連接埠，請使用適當的傳入規則建立安全群組，並將其指派給 Node.js App Server layer。請勿透過編輯

內建安全群組來修改傳入規則。每次您建立堆疊時， AWS OpsWorks Stacks 都會以標準設定覆寫內建的安全群組，因此您所做的任何變更都會遺失。

> **Note**
>
> AWS OpsWorks Stacks 會將 PORT 環境變數設定為 80 （預設） 或 443 （如果您啟用 SSL)，因此您可以使用下列程式碼來接聽請求。
>
> ```
> app.listen(process.env.PORT);
> ```

如果您將 Node.js 應用程式設定為支援 SSL，則必須指定金鑰和憑證。 AWS OpsWorks Stacks 會將每個應用程式伺服器執行個體的資料放入 `/srv/www/app_shortname/shared/config`目錄中的個別檔案，如下所示。

- `ssl.crt` – SSL 憑證。
- `ssl.key` – SSL 金鑰。
- `ssl.ca` – 鏈結憑證，如果您已指定。

您的應用程式可以從這些檔案中取得 SSL 金鑰和憑證。

PHP 應用程式伺服器 AWS OpsWorks 堆疊層

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此 layer 僅適用於 Linux 類型堆疊。

PHP App Server layer 是 AWS OpsWorks Stacks layer，可為做為 PHP 應用程式伺服器的執行個體提供藍圖。PHP App Server layer 是以 Apache2 為基礎，mod_php且沒有標準組態選項。PHP 和 Apache 版本取決於您針對 layer 執行個體所指定的作業系統。

作業系統	PHP 版本	Apache 版本
Amazon Linux 2018.03	5.3	2.2
Amazon Linux 2017.09	5.3	2.2
Amazon Linux 2017.03	5.3	2.2
Amazon Linux 2016.09	5.3	2.2
Amazon Linux 2016.03	5.3	2.2
Amazon Linux 2015.09	5.3	2.2
Amazon Linux 2015.03	5.3	2.2
Amazon Linux 2014.09	5.3	2.2
Ubuntu 14.04 LTS	5.5	2.4

Installation： AWS OpsWorks Stacks 使用執行個體的套件安裝程式，在其mod_php預設位置安裝 Apache2 和 。如需安裝的詳細資訊，請參閱 Apache。

Add Layer (新增 Layer) 頁面提供下列組態選項：

自訂安全群組

如果您選擇不自動將內建 Stacks AWS OpsWorks 安全群組與 layer 建立關聯，則會顯示此設定。您必須指定要和 layer 關聯的安全群組有哪些。如需詳細資訊，請參閱建立新的堆疊。

Elastic Load Balancer

您可以將 Elastic Load Balancing 負載平衡器連接至 layer 的執行個體。

您可以使用自訂 JSON 或自訂屬性檔案，來修改一些 Apache 組態設定。如需詳細資訊，請參閱覆寫屬性。如需可覆寫的 Apache 屬性清單，請參閱apache2 屬性。

如需如何部署 PHP 應用程式的範例 (包括如何將應用程式連線至後端資料庫)，請參閱Chef 11 Linux 堆疊入門。

> **Important**
>
> 如果您的 PHP 應用程式使用 SSL，則建議您盡可能停用 SSLv3 來處理 CVE-2014-3566 中所述的漏洞。若要執行此作業，您必須修改 Apache 伺服器 `ssl.conf` 檔案中的 `SSLProtocol` 設定。如需如何修改此設定的詳細資訊，請參閱停用 Apache 伺服器的 SSLv3。

Rails 應用程式伺服器 AWS OpsWorks 堆疊層

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此 layer 僅適用於 Linux 類型堆疊。

Rails App Server layer 是 AWS OpsWorks Stacks layer，可為充當 Rails 應用程式伺服器的執行個體提供藍圖。

Installation： AWS OpsWorks Stacks 使用執行個體的套件安裝程式，在其預設位置安裝伺服器套件。如需 Apache/Passenger 安裝的詳細資訊，請參閱 Phusion Passenger。如需記錄的詳細資訊，請參閱 Log Files。如需 Nginx/Unicorn 安裝的詳細資訊，請參閱 Unicorn。

Add Layer (新增 Layer) 頁面提供下列組態選項，而且全部都是選用的。

Ruby 版本

將由您的應用程式使用的 Ruby 版本。預設值為 2.3.

您也可以覆寫 `[:opsworks][:ruby_version]` 屬性，來指定慣用的 Ruby 版本。

> **Note**
>
> AWS OpsWorks Stacks 會安裝個別的 Ruby 套件，以供配方和執行個體代理程式使用。如需詳細資訊，請參閱Ruby 版本。

Rails 堆疊

預設 Rails 堆疊為具有 Phusion Passenger 的 Apache2。您也可以搭配使用 Nginx 與 Unicorn。

> **Note**
>
> 如果您使用 Nginx 和 Unicorn，則必須將 Unicorn Gem 新增至應用程式的 Gemfile，如下列範例所示：
>
> ```
> source 'https://rubygems.org'
> gem 'rails', '3.2.15'
> ...
> # Use unicorn as the app server
> gem 'unicorn'
> ...
> ```

Passenger 版本

如果您指定 Apache2/Passenger，則必須指定 Passenger 版本。預設值為 5.0.28。

Rubygems 版本

預設 Rubygems 版本是 2.5.1。

安裝和管理 Bundler

可讓您選擇是否安裝和管理 Bundler。預設值為 Yes (是)。

Bundler 版本

預設 Bundler 版本是 1.12.5。

自訂安全群組

如果您選擇不自動將內建 Stacks AWS OpsWorks 安全群組與 layer 建立關聯，則會顯示此設定。您必須指定要和 layer 關聯的安全群組有哪些。如需詳細資訊，請參閱建立新的堆疊。

Elastic Load Balancer

您可以將 Elastic Load Balancing 負載平衡器連接至 layer 的執行個體。

您可以使用自訂 JSON 或自訂屬性檔案，來修改一些組態設定。如需詳細資訊，請參閱覆寫屬性。如需可覆寫的 Apache、Nginx、Phusion Passenger 和 Unicorn 屬性清單，請參閱內建技術指南屬性。

> **Important**
>
> 如果您的 Ruby on Rails 應用程式使用 SSL，則建議您盡可能停用 SSLv3 來處理 CVE-2014-3566 中所述的漏洞。如需詳細資訊，請參閱停用 Rails 伺服器的 SSLv3。

主題

- 停用 Rails 伺服器的 SSLv3
- 連線至資料庫
- 部署 Ruby on Rails 應用程式

停用 Rails 伺服器的 SSLv3

若要停用 Rails 伺服器的 SSLv3，請將 layer 的 Ruby Version (Ruby 版本) 設定更新為 2.1 或更高版本，這會安裝 Ruby 2.1.4 或更高版本做為應用程式所使用的版本。

- 將層級的 Ruby Version (Ruby 版本) 設定更新為 2.1 或更高版本。
- 更新 Rails 堆疊的組態檔案，如下所示。

具有 Phusion Passenger 的 Apache

更新 Apache 伺服器 `SSLProtocol` 檔案中的 `ssl.conf` 設定，如停用 Apache 伺服器的 SSLv3 中所述。

具有 Unicorn 的 Nginx

將明確的 `ssl_protocols` 指示詞新增至 Nginx 伺服器的 `nginx.conf` 檔案。若要停用 SSLv3，請覆寫 Rails App Server layer 用於建立 的內建 nginx 技術指南`nginx.conf.erb`範本檔案`nginx.conf`，並新增下列指令：

```
ssl_protocols TLSv1.2;
```

如需如何設定 nginx.conf 的詳細資訊，請參閱設定 HTTPS 伺服器。如需如何覆寫內建範本的詳細資訊，請參閱使用自訂範本 。

連線至資料庫

當您部署應用程式時， AWS OpsWorks Stacks 會使用應用程式deploy屬性中的資訊建立新的database.yml檔案。如果您將 MySQL 或 Amazon RDS 執行個體連接至應用程式， AWS OpsWorks Stacks 會將連線資訊新增至deploy屬性，讓 database.yml自動包含正確的連線資料。

如果應用程式沒有連接的資料庫，根據預設， AWS OpsWorks Stacks 不會將任何連線資訊新增至deploy屬性，也不會建立 database.yml。如果您想要使用不同的資料庫，則可以使用自訂 JSON 將資料庫屬性與連線資訊新增至應用程式的 deploy 屬性。屬性都在 下["deploy"]["*appshortname*"]["database"]，其中 *appshortname* 是應用程式的簡短名稱，由 AWS OpsWorks Stacks 從應用程式名稱產生。您在自訂 JSON 中指定的值會覆寫任何預設設定。如需詳細資訊，請參閱新增應用程式。

AWS OpsWorks Stacks 會將下列[:...][:database]屬性值納入 database.yml。必要屬性取決於特定資料庫，但您必須具有host屬性，否則 AWS OpsWorks Stacks 不會建立 database.yml。

- [:adapter] (String) – 資料庫轉接器，例如 mysql。
- [:database] (字串) – 資料庫名稱。
- [:encoding] (字串) – 編碼，通常設定為 utf8。
- [:host] (字串) – 主機 URL，例如 railsexample.cdlqlk5uwd0k.us-west-2.rds.amazonaws.com。
- [:reconnect] (布林值) – 如果連線不再存在，應用程式是否應重新連線。
- [:password] (字串) – 資料庫密碼。
- [:port] (數字). – 資料庫的連接埠號碼。使用此屬性可覆寫預設連接埠號碼，這是由轉接器所設定。
- [:username] (字串) – 資料庫使用者名稱。

下列範例顯示簡短名稱為 myapp 之應用程式的自訂 JSON。

```
{
  "deploy" : {
    "myapp" : {
      "database" : {
```

```
        "adapter" : "adapter",
        "database" : "databasename",
        "host" : "host",
        "password" : "password",
        "port" : portnumber
        "reconnect" : true/false,
        "username" : "username"
      }
    }
  }
}
```

如需如何指定自訂 JSON 的資訊，請參閱使用自訂 JSON。若要查看用來建立 `database.yml` (`database.yml.erb`) 的範本，請前往內建技術指南儲存庫。

部署 Ruby on Rails 應用程式

您可以從任何支援的儲存庫部署 Ruby on Rails 應用程式。以下顯示如何將範例 Ruby on Rails 應用程式部署至執行 Apache/Passenger Rails 堆疊的伺服器。範例程式碼會存放在公有 GitHub 儲存庫中，但其他受支援儲存庫的基本程序都相同。如需如何建立和部署應用程式的詳細資訊，請參閱應用程式。若要檢視包含大量註解的範例程式碼，請前往 https://github.com/awslabs/opsworks-demo-rails-photo-share-app。

從 GitHub 儲存庫部署 Ruby on Rails 應用程式

1. 建立具有 Rails App Server layer 的堆疊，並將 Apache/Passenger 作為 Rails 堆疊，將全年無休執行個體新增至 layer，然後啟動它。
2. 執行個體在線上之後，請新增應用程式至堆疊，並指定下列設定：

   - Name (名稱) – 您偏好的任何名稱；範例會使用 `PhotoPoll`。

     AWS OpsWorks Stacks 會將此名稱用於顯示目的，並產生簡短名稱以供內部使用，並在堆疊組態和部署屬性中識別應用程式。例如，PhotoPoll 的簡短名稱是 photopoll。
   - App type (應用程式類型) – Ruby on Rails。
   - Rails environment (Rails 環境) – 可用的環境是由應用程式所決定。

     範例應用程式有三種：**`development`**、**`test`** 及 **`production`**。針對此範例，請將環境設定為 **`development`**。如需每個環境的描述，請參閱範例程式碼。
   - 儲存庫類型 – 任何支援的儲存庫類型。在此範例中，指定 `Git`。
   - Repository URL (儲存庫 URL) – 應該從中部署程式碼的儲存庫。

針對此範例，請將 URL 設為 **`git://github.com/awslabs/opsworks-demo-rails-photo-share-app`**。

針對剩餘設定使用預設值，然後按一下 Add App (新增應用程式) 以建立應用程式。

3. 將應用程式部署至 Rails App Server 執行個體。
4. 部署完成後，請前往執行個體頁面，然後按一下 Rails App Server 執行個體的公有 IP 地址。請查看下列事項：

靜態 Web 伺服器 AWS OpsWorks 堆疊層

Important

AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Note

此 layer 僅適用於 Linux 類型堆疊。

靜態 Web Server layer 是 AWS OpsWorks Stacks layer，為執行個體提供範本來提供靜態 HTML 頁面，其中可能包含用戶端指令碼。此 layer 是以 Nginx 為基礎。

Installation (安裝)：Nginx 安裝在 `/usr/sbin/nginx` 中。

Add Layer (新增 Layer) 頁面提供下列組態選項：

自訂安全群組

如果您選擇不自動將內建 Stacks AWS OpsWorks 安全群組與 layer 建立關聯，則會顯示此設定。您必須指定要和 layer 關聯的安全群組有哪些。如需詳細資訊，請參閱建立新的堆疊。

Elastic Load Balancer

您可以將 Elastic Load Balancing 負載平衡器連接至 layer 的執行個體。

您可以使用自訂 JSON 或自訂屬性檔案，來修改一些 Nginx 組態設定。如需詳細資訊，請參閱覆寫屬性。如需可覆寫的 Apache 屬性清單，請參閱nginx 屬性。

Important

如果您的 Web 應用程式使用 SSL，則建議您盡可能停用 SSLv3 來處理 CVE-2014-3566 中所述的漏洞。
若要停用 SSLv3，您必須修改 Nginx 伺服器的 `nginx.conf` 檔案。若要這麼做，請覆寫 Rails App Server 層用於建立 的內建 nginx 技術指南`nginx.conf.erb`範本檔案`nginx.conf`，並新增下列指令：

```
ssl_protocols TLSv1.2;
```

如需如何設定 `nginx.conf` 的詳細資訊，請參閱設定 HTTPS 伺服器。如需如何覆寫內建範本的詳細資訊，請參閱使用自訂範本 。

ECS 叢集層參考

Note

此 layer 僅適用於 Linux 類型堆疊。

ECS 叢集層代表 Amazon Elastic Container Service (Amazon ECS) 叢集，並簡化叢集管理。

Short name: (簡短名稱：) ecs-cluster

相容性：Amazon ECS 服務層僅與自訂層相容

開放連接埠：ECS 叢集允許公開存取連接埠 22 (SSH)

Autoassign Elastic IP addresses: (自動指派彈性 IP 地址：) 預設為關閉

Default EBS volume: (預設 EBS 磁碟區：) 否

Default security group: (預設安全群組：) AWS-OpsWorks-ECS-Cluster

組態：若要設定 ECS 叢集層，您必須指定下列項目：

- 是否將公有 IP 地址或彈性 IP 地址指派給容器執行個體
- 容器執行個體的執行個體描述檔

Setup recipes: (安裝配方：)

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- opsworks_ecs::setup

Configure recipes: (設定配方：)

- opsworks_ganglia::configure-client
- ssh_users
- mysql::client
- agent_version

- opsworks_ecs::configure

Deploy recipes: (部署配方：)

- deploy::default
- opsworks_ecs::deploy

Undeploy recipes: (解除部署配方：)

- opsworks_ecs::undeploy

Shutdown recipes: (關機配方：)

- opsworks_shutdown::default
- opsworks_ecs::shutdown

安裝:

- AWS OpsWorks Stacks 使用執行個體的套件安裝程式，將 Docker 安裝到其預設位置
- 設定事件的 Chef 日誌會記下 Amazon ECS 代理程式是否已成功安裝。否則，Stacks AWS OpsWorks 提供的日誌不包含 Amazon ECS 錯誤日誌資訊。如需如何處理 handleAmazon ECS 錯誤的詳細資訊，請參閱 Amazon ECS 故障診斷。

Custom Layer 參考

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

如果標準 layer 不符合您的需求，您可以建立自訂 layer。一個堆疊可以有多個自訂 layer。自訂 layer 預設會執行可支援基本功能的一組有限標準配方。您接著可以實作該 layer 的主要功能，方法是針對每個適當的生命週期事件實作一組自訂 Chef 配方，以設定該 layer 的軟體，以此類推。自訂配方會在每個事件的標準 AWS OpsWorks Stacks 配方之後執行。

Short name: (簡短名稱：) 由使用者定義；堆疊中的每個自訂 layer 都必須有不同的簡短名稱

Open ports: (開放連接埠：) 自訂伺服器 layer 預設會開啟下列連接埠的公有存取：連接埠 22 (SSH)、80 (HTTP)、443 (HTTPS)，以及堆疊之 Rails 和 PHP 應用程式伺服器 layer 的所有連接埠

Autoassign Elastic IP Addresses: (自動指派彈性 IP 地址：) 預設為關閉

Default EBS volume: (預設 EBS 磁碟區：) 否

Default Security Group: (預設安全群組：) AWS-OpsWorks-Custom-Server

Compatibility: (相容性：) Custom layer 與下列各 layer 相容：custom、db-master、lb、memcached、monitoring-master、nodejs-app、php-app、rails-app 和 web

Configuration: (組態：) 若要設定自訂 layer，您必須指定下列項目：

- layer 的名稱
- layer 的簡短名稱，可識別 Chef 配方中的 layer，而且只能使用 a-z 和數字

針對 Linux 堆疊，自訂 layer 使用下列配方。

Setup recipes: (安裝配方：)

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client

Configure recipes: (設定配方：)

- opsworks_ganglia::configure-client
- ssh_users
- agent_version

Deploy recipes: (部署配方：)

- deploy::default

Shutdown recipes: (關機配方：)

- opsworks_shutdown::default

其他 Layer 參考

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 也支援下列層。

主題

- Ganglia Layer 參考
- Memcached Layer 參考

Ganglia Layer 參考

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此 layer 僅適用於 Linux 類型堆疊。

Ganglia layer 支援 Ganglia，這是管理執行個體指標儲存和視覺化的分散式監控系統。它的設計是要使用階層執行個體拓撲，這特別適用於執行個體群組。Ganglia 有兩個基本元件：

- 低額外負荷用戶端，安裝於堆疊中的每個執行個體上，並將指標傳送到主要。
- 主伺服器，從用戶端收集指標並將其存放在 Amazon EBS 磁碟區。它也會在網頁上顯示指標。

AWS OpsWorks Stacks 在其管理的每個執行個體上都有 Ganglia 監控代理程式。當您將 Ganglia 層新增至堆疊並啟動它時，每個執行個體上的 Ganglia 代理程式都會向 Ganglia 執行個體報告指標。若要使用 Ganglia，請將具有一個執行個體的 Ganglia 層新增至堆疊。您存取資料的方式是在主要的 IP 地址登入 Ganglia 後端。您可以撰寫 Chef 配方來提供額外的指標定義。

Short name: (簡短名稱：) monitoring-master

相容性：Ganglia 層與下列層相容：Custom、db-master、memcached、php-app、rails-app。

Open ports: (開放連接埠：) 負載平衡器允許連接埠 22 (SSH)、80 (HTTP) 和 443 (HTTPS) 的公有存取。

Autoassign Elastic IP addresses: (自動指派彈性 IP 地址：) 預設為關閉

Default EBS volume: (預設 EBS 磁碟區：) 是，位於 `/vol/ganglia`

Default security group: (預設安全群組：) AWS-OpsWorks-Monitoring-Master-Server

組態：若要設定 Ganglia 層，您必須指定下列項目：

- 提供監控圖表存取權的 URI。預設值為 http：//*DNSName*/ganglia，其中 *DNSName* 是 Ganglia 執行個體的 DNS 名稱。
- 可控制監控統計資料存取權的使用者名稱和密碼。

Setup recipes: (安裝配方：)

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs
- opsworks_ganglia::client
- opsworks_ganglia::server

Configure recipes: (設定配方：)

- opsworks_ganglia::configure-client
- ssh_users
- agent_version
- opsworks_ganglia::configure-server

Deploy recipes: (部署配方：)

- deploy::default
- opsworks_ganglia::configure-server
- opsworks_ganglia::deploy

Shutdown recipes: (關機配方：)

- opsworks_shutdown::default
- apache2::stop

安裝:

- Ganglia 用戶端安裝在 `/etc/ganglia` 下方。
- Ganglia Web 前端安裝在 `/usr/share/ganglia-webfrontend` 下方。
- Ganglia logtailer 安裝在 `/usr/share/ganglia-logtailer` 下方。

Memcached Layer 參考

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此 layer 僅適用於 Linux 類型堆疊。

Memcached 是任意資料的分散式記憶體快取系統。其加速網站的方式是將字串和物件快取為 RAM 中的鍵和值，以降低必須讀取外部資料來源的次數。

若要在堆疊中使用 Memcached，請建立 Memcached layer 並新增一或多個執行個體，其做為 Memcached 伺服器運作。執行個體會自動安裝 Memcached，而堆疊的其他執行個體可以存取和使用 Memcached 伺服器。如果您使用 Rails App Server layer， AWS OpsWorks Stacks 會自動將`memcached.yml`組態檔案放置在 layer 中每個執行個體的組態目錄中。您可以從此檔案取得 Memcached 伺服器和連接埠號碼。

Short name: (簡短名稱：) memcached

相容性：Memcached layer 與下列 layer 相容：Custom、db-master、lb、 monitoring-master、nodejs-app、php-app、rails-app 和 web。

開放連接埠：Memcached layer 允許從堆疊的 Web 伺服器、自訂伺服器和 Rails、PHP 和 Node.js 應用程式伺服器公開存取連接埠 22(SSH) 和所有連接埠。

Autoassign Elastic IP addresses: (自動指派彈性 IP 地址：) 預設為關閉

Default EBS volume: (預設 EBS 磁碟區：) 否

Default security group: (預設安全群組：) AWS-OpsWorks-Memcached-Server

若要設定 Memcached layer，您必須以 MB 為單位指定快取大小。

Setup recipes: (安裝配方：)

- opsworks_initial_setup
- ssh_host_keys
- ssh_users
- mysql::client
- dependencies
- ebs

- opsworks_ganglia::client
- memcached

Configure recipes: (設定配方：)

- opsworks_ganglia::configure-client
- ssh_users
- agent_version

Deploy recipes: (部署配方：)

- deploy::default

Shutdown recipes: (關機配方：)

- opsworks_shutdown::default
- memcached::stop

安裝:

- AWS OpsWorks Stacks 會使用執行個體的套件安裝程式，在其預設位置安裝 Memcached 及其日誌檔案。

其他 Layer

Important

AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Note

這些 layer 僅供 Chef 11 和更早版本的 Linux 式堆疊使用。

AWS OpsWorks Stacks 也支援 Ganglia 和 Memcached 層。

主題

- Ganglia 層
- Memcached

Ganglia 層

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此 layer 僅適用於 Chef 11 或更舊的 Linux 類型堆疊。

AWS OpsWorks Stacks 會將所有執行個體和磁碟區指標傳送至 Amazon CloudWatch，讓您輕鬆檢視圖形和設定警示，以協助您疑難排解，並根據資源的狀態採取自動化動作。您也可以使用 Ganglia AWS OpsWorks Stacks layer 進行其他應用程式監控選項，例如存放您選擇的指標。

Ganglia layer 是執行個體的藍圖，可透過使用 Ganglia 分散式監控來監控您的堆疊。堆疊通常只有一個 Ganglia 執行個體。Ganglia layer 包含下列選用組態設定：

Ganglia URL

統計的 URL 路徑。完整的 URL 為 http://*DNSName*/*URLPath*，其中 *DNSName* 是關聯執行個體的 DNS 名稱。預設 *URLPath* 值為「/ganglia」，對應於類似：http://ec2-54-245-151-7.us-west-2.compute.amazonaws.com/ganglia。

Ganglia user name (Ganglia 使用者名稱)

統計網頁的使用者名稱。當您檢視頁面時，必須提供使用者名稱。預設值為「opsworks」。

Ganglia password (Ganglia 密碼)

控制統計網頁存取的密碼。當您檢視頁面時，您必須提供密碼。預設值為隨機產生的字串。

> **Note**
>
> 記錄密碼以供日後使用； AWS OpsWorks Stacks 不允許您在建立 layer 之後檢視密碼。不過您可以透過前往 layer 的編輯頁面，然後按一下 Update password (更新密碼)，來更新密碼。

自訂安全群組

如果您選擇不自動將內建 Stacks AWS OpsWorks 安全群組與 layer 建立關聯，則會顯示此設定。您必須指定要和 layer 關聯的安全群組有哪些。如需詳細資訊，請參閱建立新的堆疊。

Elastic Load Balancer

您可以將 Elastic Load Balancing 負載平衡器連接至 layer 的執行個體。

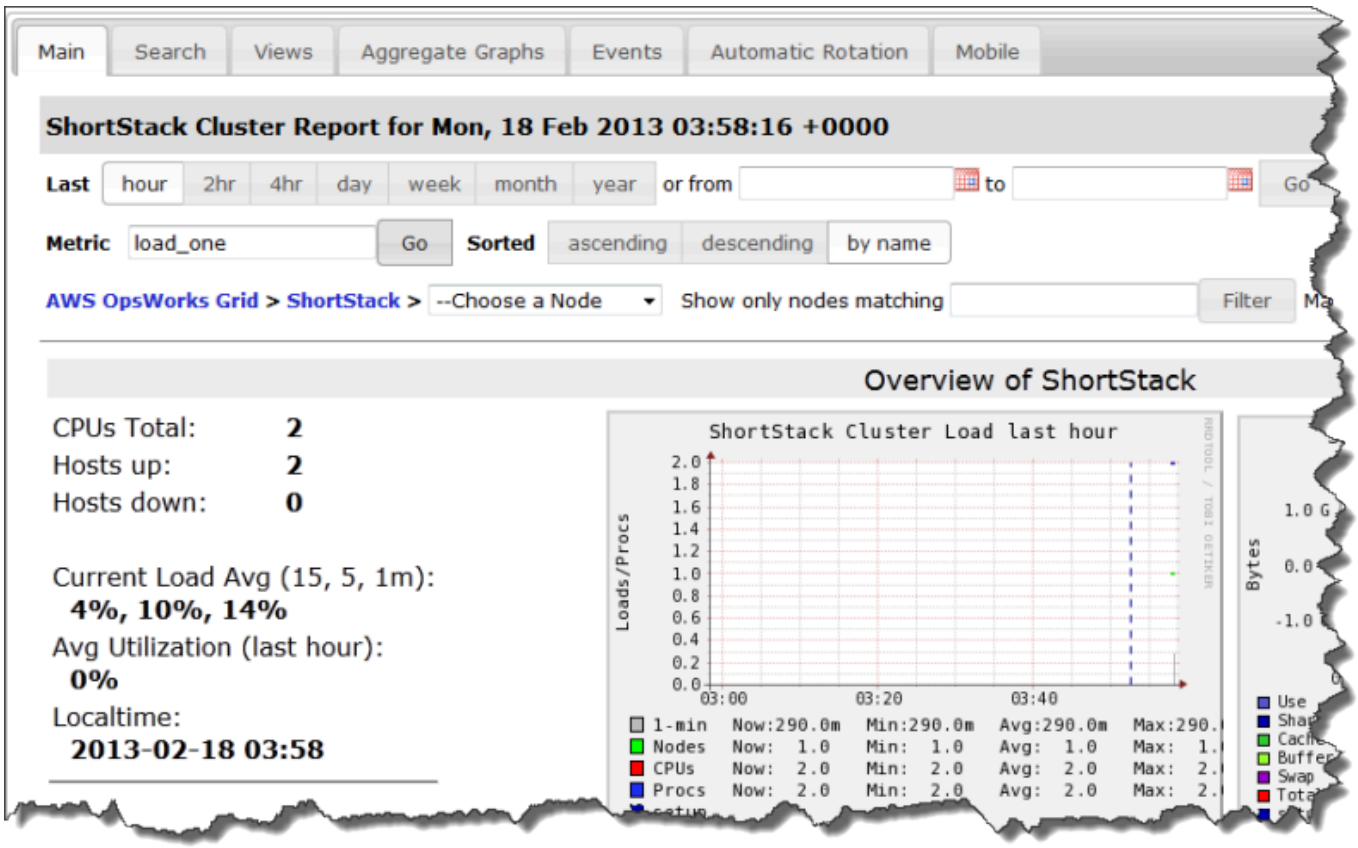
> **Important**
>
> 如果您的堆疊包含 Ganglia 層，我們建議您盡可能停用 SSLv3，以解決 CVE-2014-3566 中所述的漏洞。若要執行此作業，您必須覆寫 Apache 伺服器的 `ssl.conf.erb` 範本來修改 `SSLProtocol` 設定。如需詳細資訊，請參閱停用 Apache 伺服器的 SSLv3。

檢視 Ganglia 統計資料

AWS OpsWorks Stacks 配方會在每個執行個體上安裝低額外負荷 Ganglia 用戶端。如果您的堆疊包含 Ganglia 層，Ganglia 用戶端會在執行個體上線後立即自動開始向 Ganglia 報告。Ganglia 使用用戶端資料來計算各種統計資料，並以圖形方式在其統計資料網頁上顯示結果。

檢視 Ganglia 統計

1. 在圖層頁面上，按一下 Ganglia 開啟圖層的詳細資訊頁面。
2. 在導覽窗格中，按一下 Instances (執行個體)。在 Ganglia 下，按一下執行個體名稱。
3. 複製執行個體的 Public DNS (公有 DNS) 名稱。
4. 使用 DNS 名稱來建構統計資料 URL，看起來像：https：//http://ec2-54-245-151-7.us-west-2.compute.amazonaws.com/ganglia。
5. 將完整的 URL 在您的瀏覽器中貼上，導覽至頁面，然後輸入 Ganglia 使用者名稱和密碼以顯示頁面。範例如下。



Memcached

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此 layer 僅適用於 Chef 11 或更早版本的 Linux 式堆疊。

Memcached layer 是 AWS OpsWorks Stacks layer，可為做為 Memcached 伺服器運作的執行個體提供藍圖，這是用於任意資料的分散式記憶體快取系統。Memcached layer 包含下列組態設定。

Allocated memory (MB) (配置記憶體 (MB))

(選擇性) 每個 layer 執行個體的快取記憶體數量 (單位為 MB)。預設為 512 MB。

自訂安全群組

如果您選擇不自動將內建 Stacks AWS OpsWorks 安全群組與 layer 建立關聯，則會顯示此設定。您必須指定要和 layer 關聯的安全群組有哪些。如需詳細資訊，請參閱建立新的堆疊。

技術指南元件

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

技術指南通常包含下列基本元件：

- 屬性檔案包含一組屬性值，以代表配方和範本所使用的值。
- 範本檔案；為配方用來建立其他檔案 (例如組態檔案) 的範本。

  範本檔案通常可讓您透過覆寫屬性來修改組態檔案，這些屬性可以在不接觸技術指南的情況下完成，而不是重寫組態檔案。標準實務是每當您要變更執行個體上的組態檔案 (即使只要稍微修改) 時，都應該使用範本檔案。
- 配方檔案是 Ruby 應用程式，其定義設定系統 (包括建立和設定資料夾、安裝和設定套件、啟動服務等) 所需的一切項目。

技術指南不一定要包含這三個元件。若要更輕鬆地進行自訂，只需使用屬性或範本檔案。此外，技術指南也可以選擇性加入其他檔案類型，例如定義或規格。

本節說明這三個標準技術指南元件。如需詳細資訊 (尤其是配方的實作方法資訊)，請參閱 Opscode。

主題

- Attributes
- 範本
- 配方

Attributes

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

配方和範本視各種不同的值 (例如組態設定) 而定。您可以建立屬性檔案並以其中的屬性來代表每個值，而不必將這些值硬式編碼在配方或範本中。然後，您可以使用配方或範本中的屬性，而非明確的值。使用屬性的優點是您可以覆寫它們的值，而無需動用到技術指南。因此，您應該一律使用屬性來定義下列類型的值：

- 可能因不同堆疊或時間而異的值，例如使用者名稱。

  如果您硬式編碼這些值，那麼在您每次變更值時都必須變更配方或範本。使用屬性來定義這些值時，您可以為每個堆疊使用相同的技術指南，而只要覆寫適當的屬性即可。
- 機密值，例如密碼或秘密金鑰。

  若您在技術指南中放入明確的機密值，可能會提高曝光的風險。相反的，請使用虛設值定義屬性，再將其覆寫以設定實際的值。覆寫這類屬性的最佳方式是使用自訂 JSON。如需詳細資訊，請參閱使用自訂 JSON。

如需屬性和其覆寫方式的詳細資訊，請參閱覆寫屬性。

下列是範例屬性檔案的部分範例。

```
...
default["apache"]["listen_ports"] = [ '80','443' ]
default["apache"]["contact"] = 'ops@example.com'
default["apache"]["timeout"] = 120
default["apache"]["keepalive"] = 'Off'
default["apache"]["keepaliverequests"] = 100
default["apache"]["keepalivetimeout"] = 3
default["apache"]["prefork"]["startservers"] = 16
default["apache"]["prefork"]["minspareservers"] = 16
default["apache"]["prefork"]["maxspareservers"] = 32
default["apache"]["prefork"]["serverlimit"] = 400
```

```
default["apache"]["prefork"]["maxclients"] = 400
default["apache"]["prefork"]["maxrequestsperchild"] = 10000
...
```

AWS OpsWorks Stacks 使用以下語法定義屬性：

```
node.type["attribute"]["subattribute"]["..."]=value
```

您也可以使用冒號 (:)，如下所示：

```
node.type[:attribute][:subattribute][:...]=value
```

屬性定義具有下列元件：

node.

node. 字首是選用的，且通常會將其省略，如範例所示。

type

類型會管理是否可以覆寫屬性。 AWS OpsWorks Stacks 屬性通常會使用下列其中一種類型：

- default 是最常用的類型，因為它允許屬性覆寫。
- normal 會定義屬性，覆寫其中一個標準 AWS OpsWorks Stacks 屬性值。

> **Note**
>
> Chef 支援其他類型，這些類型對 AWS OpsWorks Stacks 來說並非必要，但可能對您的專案有用。如需詳細資訊，請參閱關於屬性。

attribute name

屬性名稱會使用標準的 Chef 節點語法 [:*attribute*][:*subattribute*][...]。您可以隨喜好使用任何屬性名稱。不過，如覆寫屬性中所述，自訂的技術指南屬性會與來自堆疊組態、部署屬性和 Chef Ohai 工具的屬性，一起合併到執行個體的節點物件中。常用的組態名稱 (如 port 或 user) 可能會出現在各種技術指南中。

為了避免名稱衝突，慣例是建立含有至少兩個元素的完整屬性名稱，如範例所示。第一個元素應該是唯一的，且通常以產品名稱 (如 Apache) 為依據。後面接著一或多個子屬性以識別特定的值，例如 `[:user]` 或 `[:port]`。您可以依據專案使用任何適當數量的子屬性。

`value`

您可以為下列類型的值設定屬性：

- 字串，例如 `default[:apache][:keepalive] = 'Off'`。
- 數字 (不含引號)，例如 `default[:apache][:timeout] = 120`。
- 布林值，其可為 `true` 或 `false` (無引號)。
- 值清單，例如 `default[:apache][:listen_ports] = [ '80','443' ]`。

屬性檔案是一種 Ruby 應用程式，因此您也可以使用節點語法和邏輯運算子，根據其他屬性來指派值。如需如何定義屬性的詳細資訊，請參閱關於屬性。如需運作屬性檔案的範例，請參閱 AWS OpsWorks Stacks 內建技術指南，網址為 https：//https://github.com/aws/opsworks-cookbooks。

範本

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以建立組態檔案並將其放在適當的目錄中，以設定許多套件。您可以在技術指南中加入組態檔案，並將其複製到適當的目錄中，但更靈活的方法是使用配方以透過範本建立組態檔。範本的其中一個優點是，您可以使用屬性來定義範本的值。如此一來，您就可以使用自訂 JSON 覆寫適當的屬性值以修改組態檔案，而無需動用到技術指南。

基本上，範本的內容和結構與其相關聯的檔案相同。下列 `httpd.conf` 即為範例檔。

```
ServerRoot "<%= node[:apache][:dir] %>"
<% if node[:platform] == "debian" || node[:platform] == "ubuntu" -%>
  LockFile /var/lock/apache2/accept.lock
<% else -%>
   LockFile logs/accept.lock
```

```
<% end -%>
PidFile <%= node[:apache][:pid_file] %>
Timeout <%= node[:apache][:timeout] %>
KeepAlive <%= node[:apache][:keepalive] %>
MaxKeepAliveRequests <%= node[:apache][:keepaliverequests] %>
KeepAliveTimeout <%= node[:apache][:keepalivetimeout] %>
<IfModule mpm_prefork_module>
    StartServers          <%= node[:apache][:prefork][:startservers] %>
    MinSpareServers       <%= node[:apache][:prefork][:minspareservers] %>
    MaxSpareServers       <%= node[:apache][:prefork][:maxspareservers] %>
    ServerLimit           <%= node[:apache][:prefork][:serverlimit] %>
    MaxClients            <%= node[:apache][:prefork][:maxclients] %>
    MaxRequestsPerChild   <%= node[:apache][:prefork][:maxrequestsperchild] %>
</IfModule>
...
```

下列範例是針對 Ubuntu 執行個體產生的 `httpd.conf` 檔案：

```
ServerRoot "/etc/httpd"
LockFile logs/accept.lock
PidFile /var/run/httpd/httpd.pid
Timeout 120
KeepAlive Off
MaxKeepAliveRequests 100
KeepAliveTimeout 3
<IfModule mpm_prefork_module>
    StartServers          16
    MinSpareServers       16
    MaxSpareServers       32
    ServerLimit           400
    MaxClients            400
    MaxRequestsPerChild   10000
</IfModule>
...
```

範本的許多文字都是從範本複製到 `httpd.conf` 檔案。不過，`<%= ... %>` 內容的處理方式則如下所示：

- Chef 會將 `<%= node[:attribute][:sub_attribute][:...]%>` 取代為屬性值。

  例如，`StartServers <%= node[:apache][:prefork][:startservers] %>` 會變成 `httpd.conf` 中的 `StartServers 16`。

- 您可以使用 `<%if-%>, <%else-%>, and <%end-%>` 依條件選取值。

  此範例會根據平台設定不同的 `accept.lock` 檔案路徑。

> **Note**
>
> 您並不一定僅限於使用技術指南屬性檔案中的屬性。您可以在執行個體的節點物件中使用任何屬性。例如，由 Chef 工具 (名稱為 Ohai) 產生的屬性也會納入節點物件。如需屬性的詳細資訊，請參閱覆寫屬性。

如需範本的詳細資訊，包括如何採用 Ruby 程式碼，請參閱關於範本。

配方

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

配方是一種 Ruby 應用程式，可定義系統的組態。配方可以安裝套件、透過範本建立組態檔案、執行 shell 命令、建立檔案和目錄等。當執行個體上發生生命週期事件時，您通常會讓 AWS OpsWorks Stacks 自動執行配方，但您也可以使用執行配方堆疊命令隨時明確執行配方。如需詳細資訊，請參閱關於配方。

配方主要是由一系列的「資源」構成，每組資源都代表系統方面的理想狀態。每個資源都包含一組屬性，可定義理想的狀態，並指定要採取的動作。Chef 會將每個資源與適當的「提供者」建立關聯，而提供者可以執行動作。如需詳細資訊，請參閱資源和提供者參考。

`package` 資源可協助您管理 Linux 執行個體上的軟體套件。下列範例會安裝 Apache 套件。

```
...
package 'apache2' do
  case node[:platform]
  when 'centos','redhat','fedora','amazon'
    package_name 'httpd'
  when 'debian','ubuntu'
```

```
    package_name 'apache2'
  end
  action :install
end
...
```

Chef 會針對平台使用適當的套件提供者。資源屬性通常只會指派一個值，但您可以使用 Ruby 邏輯操作來執行條件式指派。此範例使用 `case` 運算子，其會使用 `node[:platform]` 來識別執行個體的作業系統，並依此設定 `package_name` 屬性。您可以使用標準的 Chef 節點語法將屬性插入配方，Chef 即會以相關聯的值進行取代。您可以在節點物件中使用任何屬性，而不只限於技術指南的屬性。

在判斷適當的套件名稱之後，程式碼區段會以 `install` 動做為結尾，由該動作安裝套件。此資源的其他動作包括 `upgrade` 和 `remove`。如需詳細資訊，請參閱套件。

下列方法通常很實用：將複雜的安裝和設定任務拆解為一或多個子任務，再以個別配方形式實作每個子任務，然後讓您的主要配方在適當的時間執行這些任務。下列範例所示的這行程式碼即遵循上述範例：

```
include_recipe 'apache2::service'
```

若要讓配方執行子配方，請使用 `include_recipe` 關鍵字，後接配方名稱。配方是使用標準的 Chef *`CookbookName`*`::`*`RecipeName`* 語法來識別，其中 *`RecipeName`* 會省略 `.rb` 副檔名。

> **Note**
>
> `include_recipe` 陳述式會有效地執行主要配方中該時間點的配方。然而，實際情況是，Chef 會將每個 `include_recipe` 陳述式取代為指定的配方程式碼，之後才執行主要配方。

`directory` 資源代表目錄，例如用來包含套件檔案的目錄。下列 `default.rb` 資源會建立 Linux 日誌目錄。

```
directory node[:apache][:log_dir] do
    mode 0755
    action :create
end
```

日誌目錄定義於其中一個技術指南的屬性檔案中。資源會將目錄的模式指定為 0755，並使用 `create` 動作來建立目錄。如需詳細資訊，請參閱目錄。您也可以搭配使用此資源和 Windows 執行個體。

execute 資源代表命令，如 shell 命令或指令碼。下列範例會產生 module.load 檔案。

```
execute 'generate-module-list' do
  if node[:kernel][:machine] == 'x86_64'
    libdir = 'lib64'
  else
    libdir = 'lib'
  end
  command "/usr/local/bin/apache2_module_conf_generate.pl /usr/#{libdir}/httpd/
modules /etc/httpd/mods-available"
  action :run
end
```

資源首先會先判定 CPU 類型。[:kernel][:machine] 是 Chef 自動產生的另一個屬性，用來代表各種系統屬性，在此案例中為 CPU 類型。接著，它會指定命令 (Perl 指令碼) 並使用 run 動作來執行指令碼，以產生 module.load 檔案。如需詳細資訊，請參閱 execute。

template 資源代表檔案，通常是組態檔案，該檔案會從其中一個技術指南的範本檔案產生。下列範例會透過 apache2.conf.erb 範本 (如範本中所述) 建立一個 httpd.conf 組態檔案。

```
template 'apache2.conf' do
  case node[:platform]
  when 'centos','redhat','fedora','amazon'
    path "#{node[:apache][:dir]}/conf/httpd.conf"
  when 'debian','ubuntu'
    path "#{node[:apache][:dir]}/apache2.conf"
  end
  source 'apache2.conf.erb'
  owner 'root'
  group 'root'
  mode 0644
  notifies :restart, resources(:service => 'apache2')
end
```

資源會根據執行個體的作業系統來判斷產生的檔案名稱和位置。接著，它會指定 apache2.conf.erb 為範本，以用來產生檔案，並設定檔案的擁有者、群組和模式。它會執行 notify 動作來通知 service 資源 (其代表要重新啟動伺服器的 Apache 伺服器)。如需詳細資訊，請參閱範本。

堆疊組態及部署屬性：Linux

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本主題包含最常用的堆疊組態和部署屬性及其相關節點語法。環繞 Linux 堆疊所使用的堆疊組態命名空間結構組織而成。請注意，相同的屬性名稱有時會用於不同的目的，因此出現在不同的命名空間。例如，`id` 可以是堆疊 ID、Layer ID、應用程式 ID 等等，因此您需要完整的名稱才能使用屬性值。將此資料視覺化為 JSON 物件是很方便的方法。如需範例，請參閱「堆疊組態及部署屬性」。

> **Note**
>
> 在 Linux 執行個體上，除了將資料新增至節點物件之外， AWS OpsWorks Stacks 還會在每個執行個體上安裝此 JSON 物件。您可以使用代理程式 CLI 的 `get_json` 命令來擷取它。

主題

- opsworks 屬性
- opsworks_custom_cookbooks 屬性
- dependencies 屬性
- ganglia 屬性
- mysql 屬性
- passenger 屬性
- opsworks_bundler 屬性
- deploy 屬性
- 其他最上層屬性

opsworks 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

opsworks 元素有時稱為opsworks命名空間，包含一組定義基本堆疊組態的屬性。

> **Important**
>
> 不建議覆寫 opsworks 命名空間中的屬性值。這樣會造成內建配方失敗。

主題

- 應用程式
- instance 屬性
- layer 屬性
- rails_stack 屬性
- stack 屬性
- 其他最上層 opsworks 屬性

應用程式

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

包含內嵌物件的清單，堆疊中現有的每個應用程式各一份。每個內嵌物件都包含說明應用程式組態的下列屬性。

> **Note**
>
> 這些屬性的一般節點語法如下，其中 *i* 指定執行個體以零起始的清單索引。
>
> ```
> node["opsworks"]["applications"]["i"]["attribute_name"]
> ```

application_type

應用程式的類型 (字串)。可能的值如下：

- `php`：PHP 應用程式
- `rails`：Ruby on Rails 應用程式
- `java`：Java 應用程式
- `nodejs`：Node.js 應用程式
- `web`：靜態 HTML 頁面
- `other`：所有其他應用程式類型

```
node["opsworks"]["applications"]["i"]["application_type"]
```

name

使用者定義的顯示名稱，例如 `"SimplePHP"` (字串)。

```
node["opsworks"]["applications"]["i"]["name"]
```

slug_name

簡稱，這是全部小寫的名稱，例如 `"simplephp"` 就是 OpsWorks 從應用程式名稱產生的 (字串)。

```
node["opsworks"]["applications"]["i"]["slug_name"]
```

instance 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

`instance` 屬性包含一組指定此執行個體組態的屬性。

架構	availability_zone	backends
aws_instance_id	hostname	id
instance_type	ip	圖層
private_dns_name	private_ip	public_dns_name
region		

架構

該執行個體的架構，例如 "i386" (字串)。

```
node["opsworks"]["instance"]["architecture"]
```

availability_zone

該執行個體的可用區域，例如 "us-west-2a" (字串)。

```
node["opsworks"]["instance"]["availability_zone"]
```

backends

後端 Web 程序的數目 (字串)。例如，它會決定 HAProxy 要轉遞到 Rails 後端的並行連線數目。預設值視執行個體的記憶體和核心數而定。

```
node["opsworks"]["instance"]["backends"]
```

aws_instance_id

EC2 執行個體 ID (字串)。

```
node["opsworks"]["instance"]["aws_instance_id"]
```

hostname

主機名稱，例如 "php-app1" (字串)。

```
node["opsworks"]["instance"]["hostname"]
```

id

執行個體 ID，這是 AWS OpsWorks Stacks 產生的 GUID，可唯一識別執行個體 （ 字串 ）。

```
node["opsworks"]["instance"]["id"]
```

instance_type

執行個體類型，例如 "c1.medium" (字串)。

```
node["opsworks"]["instance"]["instance_type"]
```

ip

公有 IP 地址 (字串)。

```
node["opsworks"]["instance"]["ip"]
```

圖層

該執行個體的 Layer 清單，依簡稱識別，例如 "lb" 或 "db-master" (字串清單)。

```
node["opsworks"]["instance"]["layers"]
```

private_dns_name

私有 DNS 名稱 (字串)。

```
node["opsworks"]["instance"]["private_dns_name"]
```

private_ip

私有 IP 地址 (字串)。

```
node["opsworks"]["instance"]["private_ip"]
```

public_dns_name

公有 DNS 名稱 (字串)。

```
node["opsworks"]["instance"]["public_dns_name"]
```

region

AWS 區域，例如 "us-west-2" (字串)。

```
node["opsworks"]["instance"]["region"]
```

layer 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

`layers` 屬性包含一組 layer 屬性，每個堆疊 layer 一個，以 layer 的簡稱為名，例如 `php-app`。堆疊的每個內建 layer 最多可有一個，其簡稱如下：

- `db-master`：MySQL layer
- `java-app`：Java App Server layer
- `lb`：HAProxy layer
- `monitoring-master`：Ganglia layer
- `memcached`：Memcached layer

- `nodejs-app`：Node.js App Server layer
- `php-app`：PHP App Server layer
- `rails-app`：Rails App Server layer
- `web`：靜態 Web 伺服器層

堆疊可以包含任意數目的自訂 layer，有使用者定義的簡稱。

每個 layer 屬性都包含以下屬性：

- id
- 執行個體
- name

id

layer ID，是 OpsWorks 產生的一種 GUID，且為 layer 的唯一識別 (字串)。

```
node["opsworks"]["layers"]["layershortname"]["id"]
```

執行個體

`instances` 元素包含一組 instance 屬性，每個 layer 的線上執行個體各一個。它們以執行個體的主機名稱為名，例如 `php-app1`。

> **Note**
>
> `instances` 元素只包含那些在建立特定堆疊組態和部署屬性時處於線上狀態的執行個體。

每個 instance 元素都包含以下屬性：

availability_zone	aws_instance_id	backends
booted_at	created_at	elastic_ip
instance_type	ip	private_ip

public_dns_name	private_dns_name	region
status		

availability_zone

可用區域，例如 "us-west-2a" (字串)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["availability_zone"]
```

aws_instance_id

EC2 執行個體 ID (字串)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["aws_instance_id"]
```

backends

後端 Web 程序的數目 (數值)。例如，它會決定 HAProxy 要轉遞到 Rails 後端的並行連線數目。預設值視執行個體的記憶體和核心數而定。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["backends"]
```

booted_at

EC2 執行個體的開機時間，使用 UTC yyyy-mm-dddThh:mm:ss+hh:mm 格式 (字串)。例如，"2013-10-01T08:35:22+00:00" 對應到 2013 年 10 月 10 日 8:35:22，無時區位移。如需詳細資訊，請參閱 ISO 8601。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["booted_at"]
```

created_at

EC2 執行個體建立的時間，使用 UTC yyyy-mm-dddThh:mm:ss+hh:mm 格式 (字串)。例如，"2013-10-01T08:35:22+00:00" 對應到 2013 年 10 月 10 日 8:35:22，無時區位移。如需詳細資訊，請參閱 ISO 8601。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["created_at"]
```

elastic_ip

彈性 IP 地址，如果執行個體沒有此地址則設為 null (字串)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["elastic_ip"]
```

instance_type

執行個體類型，例如 "c1.medium" (字串)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["instance_type"]
```

ip

公有 IP 地址 (字串)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["ip"]
```

private_ip

私有 IP 地址 (字串)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["private_ip"]
```

public_dns_name

公有 DNS 名稱 (字串)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["public_dns_name"]
```

private_dns_name

私有 DNS 名稱 (字串)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["private_dns_name"]
```

region

AWS 區域，例如 "us-west-2" (字串)。

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["region"]
```

status

狀態 (字串)。可能的值如下：

- "requested"
- "booting"
- "running_setup"
- "online"
- "setup_failed"
- "start_failed"
- "terminating"
- "terminated"
- "stopped"
- "connection_lost"

```
node["opsworks"]["layers"]["layershortname"]["instances"]["instancehostname"]
["status"]
```

name

layer 的名稱，用以在主控台中表示 layer (字串)。它可以是使用者定義的，但不一定是唯一的。

```
node["opsworks"]["layers"]["layershortname"]["name"]
```

rails_stack 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

name

指定 Rails 堆疊，並設為 "apache_passenger" 或 "nginx_unicorn" (字串)。

```
node["opsworks"]["rails_stack"]["name"]
```

recipe

相關的配方，取決於您使用的是 Passenger 或 Unicorn (字串)：

- Unicorn："unicorn::rails"
- Passenger："passenger_apache2::rails"

```
node["opsworks"]["rails_stack"]["recipe"]
```

restart_command

restart 命令，取決於您使用的是 Passenger 或 Unicorn (字串)：

- Unicorn："../../shared/scripts/unicorn clean-restart"
- Passenger："touch tmp/restart.txt"

服務

服務名稱，取決於您使用的是 Passenger 或 Unicorn (字串)：

- Unicorn："unicorn"
- Passenger："apache2"

```
node["opsworks"]["rails_stack"]["service"]
```

stack 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

`stack` 屬性指定堆疊組態的某些方面，例如服務 layer 組態。

- elb-load-balancers
- id
- name
- rds_instances
- vpc_id

elb-load-balancers

包含內嵌物件的清單，堆疊中每個 Elastic Load Balancing 負載平衡器各一份。每個內嵌物件都包含說明負載平衡器組態的下列屬性。

> **Note**
>
> 這些屬性的一般節點語法如下，其中 *i* 指定執行個體以零起始的清單索引。
>
> ```
> node["opsworks"]["stack"]["elb-load-balancers"]["i"]["attribute_name"]
> ```

dns_name

負載平衡器的 DNS 名稱 (字串)。

```
node["opsworks"]["stack"]["elb-load-balancers"]["i"]["dns_name"]
```

name

負載平衡器的名稱 (字串)。

```
node["opsworks"]["stack"]["elb-load-balancers"]["i"]["name"]
```

layer_id

負載平衡器連接到的 layer ID (字串)。

```
node["opsworks"]["stack"]["elb-load-balancers"]["i"]["layer_id"]
```

id

堆疊 ID (字串)。

```
node["opsworks"]["stack"]["id"]
```

name

堆疊名稱 (字串)。

```
node["opsworks"]["stack"]["name"]
```

rds_instances

包含內嵌物件的清單，向堆疊註冊的每個 Amazon RDS 執行個體各一份。每個內嵌物件都包含一組屬性，它們會定義執行個體的組態。當您使用 Amazon RDS 主控台或 API 建立執行個體時，可以指定這些值。您也可以在建立執行個體之後，使用 Amazon RDS 主控台或 API 來編輯某些設定。如需詳細資訊，請參閱 Amazon RDS 文件。

> **Note**
>
> 這些屬性的一般節點語法如下，其中 *i* 指定執行個體以零起始的清單索引。
>
> ```
> node["opsworks"]["stack"]["rds_instances"]["i"]["attribute_name"]
> ```

如果您的堆疊有多個 Amazon RDS 執行個體，以下是如何在配方中使用特定執行個體的範例。

```
if my_rds = node["opsworks"]["stack"]["rds_instances"].select{|rds_instance|
 rds_instance["db_instance_identifier"] == 'db_id' }.first
  template "/etc/rds.conf" do
    source "rds.conf.erb"
    variables :address => my_rds["address"]
  end
end
```

address	allocated_storage	arn
auto_minor_version_upgrade	availability_zone	backup_retention_period
db_instance_class	db_instance_identifier	db_instance_status
db_name	db_parameter_groups	db_security_groups
db_user	engine	instance_create_time
license_model	multi_az	option_group_memberships
port	preferred_backup_window	preferred_maintenance_window
publicly_accessible	read_replica_db_instance_identifiers	region
status_infos	vpc_security_groups	

address

執行個體 URL，例如 `opsinstance.ccdvt3hwog1a.us-west-2.rds.amazonaws.com` (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["address"]
```

allocated_storage

配置的儲存體，以 GB 為單位 (數值)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["allocated_storage"]
```

arn

執行個體的 ARN (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["arn"]
```

auto_minor_version_upgrade

是否自動套用次要版本升級 (布林值)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["auto_minor_version_upgrade"]
```

availability_zone

執行個體的可用區域，例如 `us-west-2a` (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["availability_zone"]
```

backup_retention_period

備份保留期，以天為單位 (數值)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["backup_retention_period"]
```

db_instance_class

資料庫執行個體類別，例如 `db.m1.small` (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_instance_class"]
```

db_instance_identifier

使用者定義的資料庫執行個體識別符 (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_instance_identifier"]
```

db_instance_status

執行個體的狀態 (字串)。如需詳細資訊，請參閱資料庫執行個體。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_instance_status"]
```

db_name

使用者定義的資料庫名稱 (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_name"]
```

db_parameter_groups

執行個體的資料庫參數群組，包含內嵌物件的清單，每個參數群組各一份。如需詳細資訊，請參閱使用資料庫參數群組。每個物件包含下列屬性：

db_parameter_group_name

群組名稱 (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_parameter_groups"][j"]
["db_parameter_group_name"]
```

parameter_apply_status

套用狀態 (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_parameter_groups"][j"]
["parameter_apply_status"]
```

db_security_groups

執行個體的資料庫安全群組，包含內嵌物件的清單，每個安全群組各一份。如需詳細資訊，請參閱使用資料庫安全群組。每個物件包含下列屬性

db_security_group_name

安全群組名稱 (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_security_groups"][j"]
["db_security_group_name"]
```

status

狀態 (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_security_groups"][j"]
["status"]
```

db_user

使用者定義的主要使用者名稱 (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["db_user"]
```

engine

資料庫引擎，例如 `mysql(5.6.13)` (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["engine"]
```

instance_create_time

執行個體建立時間，例如 `2014-04-15T16:13:34Z` (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["instance_create_time"]
```

license_model

執行個體的授權模型，例如 `general-public-license` (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["license_model"]
```

multi_az

是否啟用異地同步備份部署 (布林值)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["multi_az"]
```

option_group_memberships

執行個體的選項群組成員資格，包含內嵌物件的清單，每個選項群組各一份。如需詳細資訊，請參閱使用選項群組。每個物件包含下列屬性：

option_group_name

群組名稱 (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["option_group_memberships"]
[j"]["option_group_name"]
```

status

群組狀態 (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["option_group_memberships"]
[j"]["status"]
```

port

資料庫伺服器的連接埠 (數值)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["port"]
```

preferred_backup_window

每天偏好的備份時段，例如 06:26-06:56 (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["preferred_backup_window"]
```

preferred_maintenance_window

每週偏好的維護時段，例如 thu:07:13-thu:07:43 (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["preferred_maintenance_window"]
```

publicly_accessible

資料庫是否可公開存取 (布林值)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["publicly_accessible"]
```

read_replica_db_instance_identifiers

僅供讀取複本執行個體識別符清單 (字串清單)。如需詳細資訊，請參閱使用僅供讀取複本。

```
node["opsworks"]["stack"]["rds_instances"]["i"]
["read_replica_db_instance_identifiers"]
```

region

AWS 區域，例如 us-west-2 (字串)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["region"]
```

status_infos

狀態資訊清單 (字串清單)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["status_infos"]
```

vpc_security_groups

VPC 安全群組清單 (字串清單)。

```
node["opsworks"]["stack"]["rds_instances"]["i"]["vpc_security_groups"]
```

vpc_id

VPC ID (字串)。如果執行個體不是 VPC，則此值為 `null`。

```
node["opsworks"]["stack"]["vpc_id"]
```

其他最上層 opsworks 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節包含沒有子屬性的 `opsworks` 屬性。

activity

與屬性相關聯的活動，例如 `deploy` (字串)。

```
node["opsworks"]["activity"]
```

agent_version

執行個體的 OpsWorks 代理程式版本 (字串)。

```
node["opsworks"]["agent_version"]
```

deploy_chef_provider

Chef 部署供應商，這會影響已部署的應用程式目錄結構 (字串)。您可以將此屬性設定為下列其中一項：

- `Branch`
- `Revision`
- `Timestamped` (預設值)

```
node["opsworks"]["deploy_chef_provider"]
```

ruby_stack

Ruby 堆疊 (字串)。預設設定為企業版本 (`ruby_enterprise`)。若為 MRI 版本，請將此屬性設為 `ruby`。

```
node["opsworks"]["ruby_stack"]
```

ruby_version

應用程式會使用的 Ruby 版本 (字串)。您可以使用此屬性只指定主要和次要版本。您必須使用適當的 ["ruby"] 屬性來指定修補程式版本。如需如何指定版本的詳細資訊，包括範例，請參閱 Ruby 版本。如需 AWS OpsWorks Stacks 如何決定 Ruby 版本的完整詳細資訊，請參閱內建的屬性檔案 ruby.rb。

```
node["opsworks"]["ruby_version"]
```

run_cookbook_tests

是否對您的 Chef 11.4 技術指南執行 minitest-chef-handler 測試 (布林值)。

```
node["opsworks"]["run_cookbook_tests"]
```

sent_at

此命令傳送到執行個體的時間 (數值)。

```
node["opsworks"]["sent_at"]
```

部署

如果這些屬性與部署活動相關聯，deployment 會設為部署 ID，該 ID 是由 AWS OpsWorks Stacks 產生的 GUID (字串)，可唯一識別部署。否則屬性設為 null。

```
node["opsworks"]["deployment"]
```

opsworks_custom_cookbooks 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

包含指定堆疊自訂技術指南的屬性。

已啟用

是否啟用自訂技術指南 (布林值)。

```
node["opsworks_custom_cookbooks"]["enabled"]
```

recipes

要以此命令執行的配方清單，包括自訂配方，使用 *cookbookname*::*recipename* 格式 (字串清單)。

```
node["opsworks_custom_cookbooks"]["recipes"]
```

dependencies 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

包含多個與 update_dependencies 堆疊命令相關的屬性。

gem_binary

Gems 二元碼的位置 (字串)。

upgrade_debs

是否升級 Debs 套件 (布林值)。

update_debs

是否更新 Debs 套件 (布林值)。

ganglia 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

包含 web 屬性，它有多個屬性可指定如何存取 Ganglia 統計資料網頁：

密碼

存取統計資料頁面所需的密碼 (字串)。

```
node["ganglia"]["web"]["password"]
```

url

統計資料頁面的 URL 路徑，例如 "/ganglia" (字串)。完整的 URL 為 http://*DNSNameURLPath*，其中 *DNSName* 是關聯執行個體的 DNS 名稱。

```
node["ganglia"]["web"]["url"]
```

使用者

存取統計資料頁面所需的使用者名稱 (字串)。

```
node["ganglia"]["web"]["user"]
```

mysql 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

包含一組指定 MySQL 資料庫伺服器組態的屬性。

用戶端

用戶端 IP 地址清單 (字串清單)。

```
node["mysql"]["clients"]
```

server_root_password

根密碼 (字串)。

```
node["mysql"]["server_root_password"]
```

passenger 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

包含一組指定 Phusion Passenger 組態的屬性。

gem_bin

RubyGems 二元碼的位置，例如 "/usr/local/bin/gem" (字串)。

```
node["passenger"]["gem_bin"]
```

max_pool_size

集區大小上限 (數值)。

```
node["passenger"]["max_pool_size"]
```

ruby_bin

Ruby 二元碼的位置，例如 "/usr/local/bin/ruby"。

```
node["passenger"]["ruby_bin"]
```

version

Passenger 版本 (字串)。

```
node["passenger"]["version"]
```

opsworks_bundler 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

包含指定 Bundler 支援的元素。

manage_package

是否安裝與管理 Bundler (布林值)。

```
node["opsworks_bundler"]["manage_package"]
```

version

bundler 版本 (字串)。

```
node["opsworks_bundler"]["version"]
```

deploy 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

如果這些屬性與部署事件或執行配方堆疊命令相關，則 deploy 屬性會包含每個以應用程式簡稱為名之已部署應用程式的屬性。每個應用程式屬性都包含下列屬性：

應用程式	application_type	auto_bundle_on_deploy
database	deploy_to	domains
document_root	environment_variables	群組
keep_releases	memcached	migrate
mounted_at	purge_before_symlink	rails_env
restart_command	scm	ssl_certificate
ssl_certificate_ca	ssl_certificate_key	ssl_support
堆疊	symlink_before_migrate	symlinks
使用者		

應用程式

應用程式的動態資料欄位名稱，例如 "simplephp" (字串)。

```
node["deploy"]["appshortname"]["application"]
```

application_type

應用程式類型 (字串)。可能的值如下：

- `java`：Java 應用程式
- `nodejs`：Node.js 應用程式
- `php`：PHP 應用程式
- `rails`：Ruby on Rails 應用程式
- `web`：靜態 HTML 頁面
- `other`：所有其他應用程式類型

```
node["deploy"]["appshortname"]["application_type"]
```

auto_bundle_on_deploy

若為 Rails 應用程式，是否在部署期間執行 bundler (布林值)。

```
node["deploy"]["appshortname"]["auto_bundle_on_deploy"]
```

database

包含連線應用程式資料庫所需的資訊。如果應用程式已連接資料庫層， AWS OpsWorks Stacks 會自動將適當的值指派給這些屬性。

adapter

資料庫轉接器，例如 `mysql` (字串)。

```
node["deploy"]["appshortname"]["database"]["adapter"]
```

database

資料庫名稱，通常是應用程式的動態資料欄位名稱，例如 `"simplephp"` (字串)。

```
node["deploy"]["appshortname"]["database"]["database"]
```

data_source_provider

資料來源：`mysql` 或 `rds` (字串)。

```
node["deploy"]["appshortname"]["database"]["data_source_provider"]
```

託管

資料庫主機的 IP 地址 (字串)。

```
node["deploy"]["appshortname"]["database"]["host"]
```

密碼

資料庫密碼 (字串)。

```
node["deploy"]["appshortname"]["database"]["password"]
```

port

資料庫連接埠 (數值)。

```
node["deploy"]["appshortname"]["database"]["port"]
```

reconnect

若為 Rails 應用程式，當連線不再存在時，是否應該重新連線應用程式 (布林值)。

```
node["deploy"]["appshortname"]["database"]["reconnect"]
```

使用者名稱

使用者名稱 (字串)。

```
node["deploy"]["appshortname"]["database"]["username"]
```

deploy_to

應用程式的部署位置，例如 `"/srv/www/simplephp"` (字串)。

```
node["deploy"]["appshortname"]["deploy_to"]
```

domains

應用程式網域清單 (字串清單)。

```
node["deploy"]["appshortname"]["domains"]
```

document_root

如果指定非預設的根目錄，即為文件根目錄；如果使用預設根目錄，則為 null (字串)。

```
node["deploy"]["appshortname"]["document_root"]
```

environment_variables

最多二十個屬性的集合，代表已為應用程式定義之使用者指定的環境變數。如需如何定義應用程式環境變數的詳細資訊，請參閱新增應用程式。每個屬性名稱都設為環境變數名稱，對應的值則設為變數的值，所以您可以使用下列語法來參考特定的值。

```
node["deploy"]["appshortname"]["environment_variables"]["variable_name"]
```

群組

應用程式的群組 (字串)。

```
node["deploy"]["appshortname"]["group"]
```

keep_releases

Stacks AWS OpsWorks 將存放的應用程式部署數目 （ 數目 ）。此屬性控制您可以轉返應用程式的次數。根據預設，它設定為全球值 deploy_keep_releases ，預設值為 5。您可以覆寫 keep_releases 以指定特定應用程式的已存放部署數目。

```
node["deploy"]["appshortname"]["keep_releases"]
```

memcached

包含定義 memcached 組態的兩個屬性。

託管

Memcached 伺服器執行個體的 IP 地址 （ 字串 ）。

```
node["deploy"]["appshortname"]["memcached"]["host"]
```

port

memcached 伺服器接聽的連接埠 (數值)。

```
node["deploy"]["appshortname"]["memcached"]["port"]
```

migrate

若為 Rails 應用程式，是否執行遷移 (布林值)。

```
node["deploy"]["appshortname"]["migrate"]
```

mounted_at

如果指定非預設的掛載點，即為應用程式的掛載點，如果使用預設的掛載點則為 null (字串)。

```
node["deploy"]["appshortname"]["mounted_at"]
```

purge_before_symlink

Rails 應用程式會先清除路徑陣列，再建立 symlinks (字串清單)。

```
node["deploy"]["appshortname"]["purge_before_symlink"]
```

rails_env

對於 Rails App Server 執行個體，導軌環境，例如 "production"（ 字串 ）。

```
node["deploy"]["appshortname"]["rails_env"]
```

restart_command

重新啟動應用程式時要執行的命令，例如 "echo 'restarting app'"。

```
node["deploy"]["appshortname"]["restart_command"]
```

scm

包含一組屬性，指定 OpsWorks 從其來源控制儲存庫部署應用程式所使用的資訊。屬性隨儲存庫類型而異。

密碼

私有儲存庫為密碼，公有儲存庫則為 null (字串)。對於私有 Amazon S3 儲存貯體， 屬性會設定為私密金鑰。

```
node["deploy"]["appshortname"]["scm"]["password"]
```

repository

儲存庫 URL，例如 "git://github.com/amazonwebservices/opsworks-demo-php-simple-app.git" (字串)。

```
node["deploy"]["appshortname"]["scm"]["repository"]
```

修訂版

如果儲存庫有多個分支，此屬性會指定該應用程式的分支或版本，例如 "version1" (字串)。否則會設為 null。

```
node["deploy"]["appshortname"]["scm"]["revision"]
```

scm_type

儲存庫類型 (字串)。可能的值如下：

- "git"：Git 儲存庫
- "svn"：Subversion 儲存庫
- "s3"：Amazon S3 儲存貯體
- "archive"：HTTP 封存
- "other"：其他儲存庫類型

```
node["deploy"]["appshortname"]["scm"]["scm_type"]
```

ssh_key

用來存取私有 Git 儲存庫的部署 SSH 金鑰，公有儲存庫則為 null (字串)。

```
node["deploy"]["appshortname"]["scm"]["ssh_key"]
```

使用者

私有儲存庫為使用者名稱，公有儲存庫則為 null (字串)。對於私有 Amazon S3 儲存貯體， 屬性會設定為存取金鑰。

```
node["deploy"]["appshortname"]["scm"]["user"]
```

ssl_certificate

如已啟用 SSL 支援，則為應用程式的 SSL 憑證，否則為 null (字串)。

```
node["deploy"]["appshortname"]["ssl_certificate"]
```

ssl_certificate_ca

如已啟用 SSL，則為指定中繼憑證授權單位金鑰或用戶端身分驗證的屬性 (字串)。

```
node["deploy"]["appshortname"]["ssl_certificate_ca"]
```

ssl_certificate_key

如已啟用 SSL 支援，則為應用程式的 SSL 私有金鑰，否則為 null (字串)。

```
node["deploy"]["appshortname"]["ssl_certificate_key"]
```

ssl_support

是否支援 SSL (布林值)。

```
node["deploy"]["appshortname"]["ssl_support"]
```

堆疊

包含一個布林值屬性 `needs_reload`，指定是否在部署期間重新載入應用程式伺服器。

```
node["deploy"]["appshortname"]["stack"]["needs_reload"]
```

symlink_before_migrate

Rails 應用程式包含的 symlink，要先建立再當成 "*link*":"*target*" 配對執行遷移。

```
node["deploy"]["appshortname"]["symlink_before_migrate"]
```

symlinks

包含部署的 symlink 做為 "*link*":"*target*" 配對。

```
node["deploy"]["appshortname"]["symlinks"]
```

使用者

應用程式的使用者 (字串)。

```
node["deploy"]["appshortname"]["user"]
```

其他最上層屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節包含沒有子屬性的最上層堆疊組態屬性。

rails 屬性

包含指定伺服器集區大小上限的 max_pool_size 屬性 (數值)。屬性值由 Stacks AWS OpsWorks 設定，並取決於執行個體類型，但您可以使用自訂 JSON 或自訂屬性檔案來覆寫它。

```
node["rails"]["max_pool_size"]
```

recipes 屬性

此活動過去執行的內建配方清單，使用 "*cookbookname*::*recipename*" 格式 (字串清單)。

```
node["recipes"]
```

opsworks_rubygems 屬性

包含指定 RubyGems 版本的 version 元素 (字串)。

```
node["opsworks_rubygems"]["version"]
```

languages 屬性

包含為每個已安裝語言命名的屬性，例如 ruby。此屬性是一種物件，可包含屬性 (例如 ruby_bin)，也可指定安裝資料夾 (例如 "/usr/bin/ruby") (字串)。

ssh_users 屬性

包含一組屬性，每個屬性都會描述已授予 SSH 許可的其中一個使用者。每個屬性都以使用者的 Unix ID 命名。 AWS OpsWorks Stacks 會為 2000-4000 範圍內的每個使用者產生唯一的 ID，例如 "2001"，並在每個執行個體上建立具有該 ID 的使用者。因為 AWS OpsWorks 保留 2000-4000 範圍，您在 之外建立的使用者 AWS OpsWorks （例如使用技術指南配方，或從 IAM AWS OpsWorks 將使用者匯入）可以擁有由 Stacks AWS OpsWorks 為其他使用者覆寫的 UIDs。最佳實務是在 Stacks AWS OpsWorks 主控台中建立使用者和管理其存取權。如果您在 AWS OpsWorks Stacks 之外建立使用者，請使用大於 4000 的 *UnixID* 值。

每個屬性都包含下列屬性：

email

使用者的電子郵件地址 (字串)。

```
node["ssh_users"]["UnixID"]["email"]
```

public_key

使用者的公有 SSH 金鑰 (字串)。

```
node["ssh_users"]["UnixID"]["public_key"]
```

sudoer

使用者是否有 sudo 許可 (布林值)。

```
node["ssh_users"]["UnixID"]["sudoer"]
```

name

使用者名稱 (字串)。

```
node["ssh_users"]["UnixID"]["name"]
```

內建技術指南屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 其中大部分屬性僅適用於 Linux 堆疊。

大多數內建配方具有一或多個屬性檔案，可定義各種設定。您可以存取自訂配方中的這些設定，並使用自訂 JSON 加以覆寫。您通常需要存取或覆寫屬性，以控制 Stacks AWS OpsWorks 支援的各種伺服器技術的組態。本節摘要說明這些屬性。您可以在 https://github.com/aws/opsworks-cookbooks.git 找到完整的屬性檔案，以及相關聯的配方和範本。

> **Note**
>
> 所有內建配方屬性都是 `default` 類型。

主題

- apache2 屬性
- deploy 屬性
- haproxy 屬性
- memcached 屬性
- mysql 屬性
- nginx 屬性
- opsworks_berkshelf 屬性
- opsworks_java 屬性
- passenger_apache2 屬性
- ruby 屬性
- unicorn 屬性

apache2 屬性

Important

AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Note

這些屬性僅適用於 Linux 堆疊。

apache2 屬性指定 Apache HTTP 伺服器組態。如需詳細資訊，請參閱 Apache Core Features。如需如何覆寫內建屬性以指定自訂值的詳細資訊，請參閱覆寫屬性。

binary	contact	deflate_types
dir	document_root	群組
hide_info_headers	icondir	init_script

keepalive	keepaliverequests	keepalivetimeout
lib_dir	libexecdir	listen_ports
log_dir	logrotate 屬性	pid_file
prefork 屬性	serversignature	servertokens
timeout	traceenable	使用者
version	worker 屬性	

binary

Apache 二進位檔的位置 (字串)。預設值為 '/usr/sbin/httpd'。

```
node[:apache][:binary]
```

contact

電子郵件聯絡人 (字串)。預設值是虛擬地址 'ops@example.com'。

```
node[:apache][:contact]
```

deflate_types

引導 mod_deflate 啟用受瀏覽器支援之指定 MIME 類型的壓縮 (字串清單)。預設值如下：

```
['application/javascript',
 'application/json',
 'application/x-javascript',
 'application/xhtml+xml',
 'application/xml',
 'application/xml+rss',
 'text/css',
 'text/html',
 'text/javascript',
 'text/plain',
 'text/xml']
```

> **Warning**
>
> 壓縮可能會引發安全風險。若要完全停用壓縮，請設定此屬性如下：
>
> ```
> node[:apache][:deflate_types] = []
> ```

```
node[:apache][:deflate_types]
```

dir

伺服器的根目錄 (字串)。預設值如下：

- Amazon Linux 和 Red Hat Enterprise Linux (RHEL)：`'/etc/httpd'`
- Ubuntu：`'/etc/apache2'`

```
node[:apache][:dir]
```

document_root

文件根 (字串)。預設值如下：

- Amazon Linux 和 RHEL：`'/var/www/html'`
- Ubuntu：`'/var/www'`

```
node[:apache][:document_root]
```

群組

群組名稱 (字串)。預設值如下：

- Amazon Linux 和 RHEL：`'apache'`
- Ubuntu：`'www-data'`

```
node[:apache][:group]
```

hide_info_headers

是否省略 HTTP 標頭中的版本和模組資訊 (`'true'`/`'false'`) (字串)。預設值為 `'true'`。

```
node[:apache][:hide_info_headers]
```

icondir

圖示目錄 (字串)。預設值如下：

- Amazon Linux 和 RHEL：`'/var/www/icons/'`
- Ubuntu：`'/usr/share/apache2/icons'`

```
node[:apache][:icondir]
```

init_script

初始化指令碼 (字串)。預設值如下：

- Amazon Linux 和 RHEL：`'/etc/init.d/httpd'`
- Ubuntu：`'/etc/init.d/apache2'`

```
node[:apache][:init_script]
```

keepalive

是否啟用持續連線 (字串)。可能的值為 `'On'` 和 `'Off'` (字串)。預設值為 `'Off'`。

```
node[:apache][:keepalive]
```

keepaliverequests

Apache 會同時處理的持續連線請求數目上限 (數值)。預設值為 `100`。

```
node[:apache][:keepaliverequests]
```

keepalivetimeout

Apache 等待請求的時間，之後會關閉連線 (數值)。預設值為 `3`。

```
node[:apache][:keepalivetimeout]
```

lib_dir

包含物件程式碼程式庫的目錄 (字串)。預設值如下：

- Amazon Linux (x86)：'/usr/lib/httpd'
- Amazon Linux (x64) 和 RHEL：'/usr/lib64/httpd'
- Ubuntu：'/usr/lib/apache2'

```
node[:apache][:lib_dir]
```

libexecdir

包含可執行程式的目錄 (字串)。預設值如下：

- Amazon Linux (x86)：'/usr/lib/httpd/modules'
- Amazon Linux (x64) 和 RHEL：'/usr/lib64/httpd/modules'
- Ubuntu：'/usr/lib/apache2/modules'

```
node[:apache][:libexecdir]
```

listen_ports

伺服器接聽的連接埠清單 (字串清單)。預設值為 ['80','443']。

```
node[:apache][:listen_ports]
```

log_dir

日誌目錄 (字串)。預設值如下：

- Amazon Linux 和 RHEL：'/var/log/httpd'
- Ubuntu：'/var/log/apache2'

```
node[:apache][:log_dir]
```

logrotate 屬性

這些屬性指定如何輪換日誌檔案。

delaycompress

是否將已關閉的日誌檔案壓縮延遲到下一個輪換週期開始 ('true'/'false') (字串)。預設值為 'true'。

```
node[:apache][:logrotate][:delaycompress]
```

群組

日誌檔案的群組 (字串)。預設值為 'adm'。

```
node[:apache][:logrotate][:group]
```

模式

日誌檔案的模式 (字串)。預設值為 '640'。

```
node[:apache][:logrotate][:mode]
```

owner

日誌檔案的擁有者 (字串)。預設值為 'root'。

```
node[:apache][:logrotate][:owner]
```

rotate

輪換週期次數，之後會移除已關閉的日誌檔案 (字串)。預設值為 '30'。

```
node[:apache][:logrotate][:rotate]
```

schedule

輪換排程 (字串)。可能的值如下：

- 'daily'
- 'weekly'
- 'monthly'

預設值為 'daily'。

```
node[:apache][:logrotate][:schedule]
```

pid_file

包含協助程式之處理序 ID 的檔案 (字串)。預設值如下：

- Amazon Linux 和 RHEL：`'/var/run/httpd/httpd.pid'`
- Ubuntu：`'/var/run/apache2.pid'`

```
node[:apache][:pid_file]
```

prefork 屬性

這些屬性指定預先分支處理組態。

maxclients

要儲存的同步請求數目上限 (數值)。預設值為 `400`。

> **Note**
>
> 僅針對執行 Amazon Linux 或 RHEL 的執行個體使用此屬性。如果您的執行個體執行 Ubuntu 14.04 LTS，請使用 maxrequestworkers。

```
node[:apache][:prefork][:maxclients]
```

maxrequestsperchild

子伺服器處理序會處理的請求數目上限 (數值)。預設值為 `10000`。

```
node[:apache][:prefork][:maxrequestsperchild]
```

maxrequestworkers

要儲存的同步請求數目上限 (數值)。預設值為 `400`。

> **Note**
>
> 此屬性只能用於執行 Ubuntu 14.04 LTS 的執行個體。如果您的執行個體正在執行 Amazon Linux 或 RHEL，請使用 maxclients 。

```
node[:apache][:prefork][:maxrequestworkers]
```

maxspareservers

閒置子伺服器處理序數目上限 (數值)。預設值為 32。

```
node[:apache][:prefork][:maxspareservers]
```

minspareservers

閒置子伺服器處理序數目下限 (數值)。預設值為 16。

```
node[:apache][:prefork][:minspareservers]
```

serverlimit

可設定的處理序數目上限 (數值)。預設值為 400。

```
node[:apache][:prefork][:serverlimit]
```

startservers

要在啟動時建立的子伺服器處理序數目 (數值)。預設值為 16。

```
node[:apache][:prefork][:startservers]
```

serversignature

指定是否及如何設定伺服器產生文件尾端的頁尾 (字串)。可能的值為 'On'、'Off' 和 'Email'。預設值為 'Off'。

```
node[:apache][:serversignature]
```

servertokens

指定要包含在回應標頭中的伺服器版本資訊類型 (字串)：

- 'Full'：完整資訊。例如，伺服器：Apache/2.4.2 (Unix) PHP/4.2.2 MyMod/1.2
- 'Prod'：產品名稱。例如，伺服器：Apache

- 'Major'：主要版本。例如，伺服器：Apache/2
- 'Minor'：主要和次要版本。例如，伺服器：Apache/2.4
- 'Min'：最低版本。例如，伺服器：Apache/2.4.2
- 'OS'：作業系統版本。例如，伺服器：Apache/2.4.2 (Unix)

預設值為 'Prod'。

```
node[:apache][:servertokens]
```

timeout

Apache 等待 I/O 的時間 (數值)。預設值為 120。

```
node[:apache][:timeout]
```

traceenable

是否啟用 TRACE 請求 (字串)。可能的值為 'On' 和 'Off'。預設值為 'Off'。

```
node[:apache][:traceenable]
```

使用者

使用者名稱 (字串)。預設值如下：

- Amazon Linux 和 RHEL：'apache'
- Ubuntu：'www-data'

```
node[:apache][:user]
```

version

Apache 版本 (字串)。預設值如下：

- Amazon Linux: 2.2
- Ubuntu 14.04 LTS：2.4
- RHEL: 2.4

```
node[:apache][:version]
```

worker 屬性

這些屬性指定工作者處理序組態。

startservers

要在啟動時建立的子伺服器處理序數目 (數值)。預設值為 4。

```
node[:apache][:worker][:startservers]
```

maxclients

要儲存的同步請求數目上限 (數值)。預設值為 1024。

```
node[:apache][:worker][:maxclients]
```

maxsparethreads

閒置執行緒數目上限 (數值)。預設值為 192。

```
node[:apache][:worker][:maxsparethreads]
```

minsparethreads

閒置執行緒數目下限 (數值)。預設值為 64。

```
node[:apache][:worker][:minsparethreads]
```

threadsperchild

每個子處理序的執行緒數目 (數值)。預設值為 64。

```
node[:apache][:worker][:threadsperchild]
```

maxrequestsperchild

子伺服器處理序會處理的請求數目上限 (數值)。預設值為 10000。

```
node[:apache][:worker][:maxrequestsperchild]
```

deploy 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

內建部署技術指南的 `deploy.rb` 屬性檔案定義 `opsworks` 命名空間中的下列屬性。如需部署目錄的詳細資訊，請參閱部署配方。如需如何覆寫內建屬性以指定自訂值的詳細資訊，請參閱覆寫屬性。

deploy_keep_releases

Stacks AWS OpsWorks 將存放之應用程式部署數量的全域設定 （數字）。預設值為 5。此值控制您可以復原應用程式的次數。

```
node[:opsworks][:deploy_keep_releases]
```

群組

(僅限 Linux) 應用程式部署目錄的 `group` 設定 (字串)。預設值視執行個體的作業系統而定。

- 若是 Ubuntu 執行個體，預設值為 `www-data`。
- 對於使用 Nginx 和 Unicorn 的 Rails App Server 層成員的 Amazon Linux 或 RHEL 執行個體，預設值為 `nginx`。
- 若是其他所有 Amazon Linux 或 RHEL 執行個體，預設值為 `apache`。

```
node[:opsworks][:deploy_user][:group]
```

使用者

(僅限 Linux) 應用程式部署目錄的 `user` 設定 (字串)。預設值為 `deploy`。

```
node[:opsworks][:deploy_user][:user]
```

haproxy 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 這些屬性僅適用於 Linux 堆疊。

haproxy 屬性指定 HAProxy 伺服器組態。如需詳細資訊，請參閱 HAProxy 文件。如需如何覆寫內建屬性以指定自訂值的詳細資訊，請參閱覆寫屬性。

balance	check_interval	client_timeout
connect_timeout	default_max_connections	global_max_connections
health_check_method	health_check_url	queue_timeout
http_request_timeout	maxcon_factor_nodejs_app	maxcon_factor_nodejs_app_ssl
maxcon_factor_php_app	maxcon_factor_php_app_ssl	maxcon_factor_rails_app
maxcon_factor_rails_app_ssl	maxcon_factor_static	maxcon_factor_static_ssl
retries	server_timeout	stats_url
stats_user		

balance

負載平衡器用來選取伺服器的演算法 (字串)。預設值為 `'roundrobin'`。其他選項為：

- 'static-rr'

- 'leastconn'
- 'source'
- 'uri'
- 'url_param'
- 'hdr(name)'
- 'rdp-cookie'
- 'rdp-cookie(name)'

如需這些引數的詳細資訊，請參閱 balance。

```
node[:haproxy][:balance]
```

check_interval

運作狀態檢查時間間隔 (字串)。預設值為 '10s'。

```
node[:haproxy][:check_interval]
```

client_timeout

用戶端可處於非作用中狀態的時間上限 (字串)。預設值為 '60s'。

```
node[:haproxy][:client_timeout]
```

connect_timeout

HAProxy 會等待伺服器連線嘗試成功的時間上限 (字串)。預設值為 '10s'。

```
node[:haproxy][:connect_timeout]
```

default_max_connections

預設的連線數目上限 (字串)。預設值為 '80000'。

```
node[:haproxy][:default_max_connections]
```

global_max_connections

連線數目上限 (字串)。預設值為 '80000'。

```
node[:haproxy][:global_max_connections]
```

health_check_method

運作狀態檢查方法 (字串)。預設值為 'OPTIONS'。

```
node[:haproxy][:health_check_method]
```

health_check_url

用來檢查伺服器運作狀態的 URL 路徑 (字串)。預設值為 '/'。

```
node[:haproxy][:health_check_url ]
```

queue_timeout

可用連線的等待時間上限 (字串)。預設值為 '120s'。

```
node[:haproxy][:queue_timeout]
```

http_request_timeout

HAProxy 會等待 HTTP 請求完成的時間上限 (字串)。預設值為 '30s'。

```
node[:haproxy][:http_request_timeout]
```

retries

伺服器連線失敗之後的重試次數 (字串)。預設值為 '3'。

```
node[:haproxy][:retries]
```

server_timeout

用戶端可處於非作用中狀態的時間上限 (字串)。預設值為 '60s'。

```
node[:haproxy][:server_timeout]
```

stats_url

統計資料頁面的 URL 路徑 (字串)。預設值為 '/haproxy?stats'。

```
node[:haproxy][:stats_url]
```

stats_user

統計資料頁面使用者名稱 (字串)。預設值為 'opsworks'。

```
node[:haproxy][:stats_user]
```

maxcon 屬性代表載入因數乘數，用來運算 HAProxy 允許後端的連線數目上限。例如，假設您在backend值為 4 的小型執行個體上有一個 Rails 應用程式伺服器，這表示 AWS OpsWorks Stacks 將為該執行個體設定四個 Rails 程序。如果您使用預設 maxcon_factor_rails_app 值 7，HAProxy 會處理 Rails 伺服器的 28 (4* 7) 個連線。

maxcon_factor_nodejs_app

Node.js 應用程式伺服器的 maxcon 因數 (數值)。預設值為 10。

```
node[:haproxy][:maxcon_factor_nodejs_app]
```

maxcon_factor_nodejs_app_ssl

使用 SSL 之 Node.js 應用程式伺服器的 maxcon 因數 (數值)。預設值為 10。

```
node[:haproxy][:maxcon_factor_nodejs_app_ssl]
```

maxcon_factor_php_app

PHP 應用程式伺服器的 maxcon 因數 (數值)。預設值為 10。

```
node[:haproxy][:maxcon_factor_php_app]
```

maxcon_factor_php_app_ssl

使用 SSL 之 PHP 應用程式伺服器的 maxcon 因數 (數值)。預設值為 10。

```
node[:haproxy][:maxcon_factor_php_app_ssl]
```

maxcon_factor_rails_app

Rails 應用程式伺服器的 maxcon 因數 (數值)。預設值為 7。

```
node[:haproxy][:maxcon_factor_rails_app]
```

maxcon_factor_rails_app_ssl

使用 SSL 之 Rails 應用程式伺服器的 maxcon 因數 (數值)。預設值為 7。

```
node[:haproxy][:maxcon_factor_rails_app_ssl]
```

maxcon_factor_static

靜態 Web 伺服器的 maxcon 因數 (數值)。預設值為 15。

```
node[:haproxy][:maxcon_factor_static]
```

maxcon_factor_static_ssl

使用 SSL 之靜態 Web 伺服器的 maxcon 因數 (數值)。預設值為 15。

```
node[:haproxy][:maxcon_factor_static_ssl]
```

memcached 屬性

Important

AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Note

這些屬性僅適用於 Linux 堆疊。

memcached 屬性指定 Memcached 伺服器組態。如需如何覆寫內建屬性以指定自訂值的詳細資訊，請參閱覆寫屬性。

memory	max_connections	pid_file

port	start_command	stop_command
使用者		

memory

可使用的記憶體上限，以 MB 為單位 (數值)。預設值為 512。

```
node[:memcached][:memory]
```

max_connections

連線數目上限 (字串)。預設值為 '4096'。

```
node[:memcached][:max_connections]
```

pid_file

包含協助程式之處理序 ID 的檔案 (字串)。預設值為 'var/run/memcached.pid'。

```
node[:memcached][:pid_file]
```

port

要接聽的連接埠 (數值)。預設值為 11211。

```
node[:memcached][:port]
```

start_command

啟動命令 (字串)。預設值為 '/etc/init.d/memcached start'。

```
node[:memcached][:start_command]
```

stop_command

停止命令 (字串)。預設值為 '/etc/init.d/memcached stop'。

```
node[:memcached][:stop_command]
```

使用者

使用者 (字串)。預設值為 'nobody'。

```
node[:memcached][:user]
```

mysql 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 這些屬性僅適用於 Linux 堆疊。

mysql 屬性指定 MySQL 主組態。如需詳細資訊，請參閱 Server System Variables。如需如何覆寫內建屬性以指定自訂值的詳細資訊，請參閱覆寫屬性。

basedir	bind_address	用戶端
conf_dir	confd_dir	datadir
grants_path	mysql_bin	mysqladmin_bin
pid_file	port	root_group
server_root_password	socket	tunable 屬性

basedir

基本目錄 (字串)。預設值為 '/usr'。

```
node[:mysql][:basedir]
```

bind_address

MySQL 接聽的地址 (字串)。預設值為 '0.0.0.0'。

```
node[:mysql][:bind_address]
```

用戶端

用戶端清單 (字串清單)。

```
node[:mysql][:clients]
```

conf_dir

包含組態檔案的目錄 (字串)。預設值如下：

- Amazon Linux 和 RHEL：'/etc'
- Ubuntu：'/etc/mysql'

```
node[:mysql][:conf_dir]
```

confd_dir

包含其他組態檔案的目錄 (字串)。預設值為 '/etc/mysql/conf.d'。

```
node[:mysql][:confd_dir]
```

datadir

資料目錄 (字串)。預設值為 '/var/lib/mysql'。

```
node[:mysql][:datadir]
```

grants_path

授予資料表位置 (字串)。預設值為 '/etc/mysql_grants.sql'。

```
node[:mysql][:grants_path]
```

mysql_bin

mysql 二進位檔位置 (字串)。預設值為 '/usr/bin/mysql'。

```
node[:mysql][:mysql_bin]
```

mysqladmin_bin

mysqladmin 位置 (字串)。預設值為 '/usr/bin/mysqladmin'。

```
node[:mysql][:mysqladmin_bin]
```

pid_file

包含協助程式之處理序 ID 的檔案 (字串)。預設值為 '/var/run/mysqld/mysqld.pid'。

```
node[:mysql][:pid_file]
```

port

伺服器接聽的連接埠 (數值)。預設值為 3306。

```
node[:mysql][:port]
```

root_group

根群組 (字串)。預設值為 'root'。

```
node[:mysql][:root_group]
```

server_root_password

伺服器的根密碼 (字串)。預設值為隨機產生。

```
node[:mysql][:server_root_password]
```

socket

通訊端檔案的位置 (字串)。預設值為 '/var/lib/mysql/mysql.sock'。預設值如下：

- Amazon Linux 和 RHEL：'/var/lib/mysql/mysql.sock'

- Ubuntu：`'/var/run/mysqld/mysqld.sock'`

```
node[:mysql][:socket]
```

tunable 屬性

tunable 屬性用於效能調校。

back_log	innodb_additional_mem_pool_size	innodb_buffer_pool_size
innodb_flush_log_at_trx_commit	innodb_lock_wait_timeout	key_buffer
log_slow_queries	長查詢時間	max_allowed_packet
max_connections	max_heap_table_size	net_read_timeout
net_write_timeout	query_cache_limit	query_cache_size
query_cache_type	thread_cache_size	thread_stack
wait_timeout	table_cache	

back_log

未完成的請求數目上限 (字串)。預設值為 `'128'`。

```
node[:mysql][:tunable][:back_log]
```

innodb_additional_mem_pool_size

Innodb 用來存放內部資料結構的集區大小 (字串)。預設值為 `'20M'`。

```
node[:mysql][:tunable][:innodb_additional_mem_pool_size]
```

innodb_buffer_pool_size

Innodb 緩衝集區大小 (字串)。屬性值由 Stacks AWS OpsWorks 設定，並取決於執行個體類型，但您可以使用自訂 JSON 或自訂屬性檔案來覆寫它。

```
node[:mysql][:tunable][:innodb_buffer_pool_size]
```

innodb_flush_log_at_trx_commit

Innodb 排清日誌緩衝區的頻率 (字串)。預設值為 '2'。如需詳細資訊，請參閱 innodb_flush_log_at_trx_commit。

```
node[:mysql][:tunable][:innodb_flush_log_at_trx_commit]
```

innodb_lock_wait_timeout

Innodb 交易等待資料列鎖定的時間上限，以秒為單位 (字串)。預設值為 '50'。

```
node[:mysql][:tunable][:innodb_lock_wait_timeout]
```

key_buffer

索引緩衝區大小 (字串)。預設值為 '250M'。

```
node[:mysql][:tunable][:key_buffer]
```

log_slow_queries

慢速查詢日誌檔案的位置 (字串)。預設值為 '/var/log/mysql/mysql-slow.log'。

```
node[:mysql][:tunable][:log_slow_queries]
```

長查詢時間

指定查詢為長時間查詢所需的時間，以秒為單位 (字串)。預設值為 '1'。

```
node[:mysql][:tunable][:long_query_time]
```

max_allowed_packet

允許的封包大小上限 (字串)。預設值為 '32M'。

```
node[:mysql][:tunable][:max_allowed_packet]
```

max_connections

同時用戶端連線數目上限 (字串)。預設值為 '2048'。

```
node[:mysql][:tunable][:max_connections]
```

max_heap_table_size

使用者建立的 MEMORY 資料表大小上限 (字串)。預設值為 '32M'。

```
node[:mysql][:tunable][:max_heap_table_size]
```

net_read_timeout

等待更多連線資料的時間，以秒為單位 (字串)。預設值為 '30'。

```
node[:mysql][:tunable][:net_read_timeout]
```

net_write_timeout

等待區塊寫入連線的時間，以秒為單位 (字串)。預設值為 '30'。

```
node[:mysql][:tunable][:net_write_timeout]
```

query_cache_limit

個別快取查詢大小上限 (字串)。預設值為 '2M'。

```
node[:mysql][:tunable][:query_cache_limit]
```

query_cache_size

查詢快取大小 (字串)。預設值為 '128M'。

```
node[:mysql][:tunable][:query_cache_size]
```

query_cache_type

查詢快取類型 (字串)。可能的值如下：

- '0'：不快取或擷取快取的資料。

- '1'：不以 SELECT SQL_NO_CACHE 開頭的快取陳述式。
- '2'：以 SELECT SQL_CACHE 開頭的快取陳述式。

預設值為 '1'。

```
node[:mysql][:tunable][:query_cache_type]
```

thread_cache_size

快取以重複使用的用戶端執行緒數目 (字串)。預設值為 '8'。

```
node[:mysql][:tunable][:thread_cache_size]
```

thread_stack

每個執行緒的堆疊大小 (字串)。預設值為 '192K'。

```
node[:mysql][:tunable][:thread_stack]
```

wait_timeout

等待非互動式連線的時間，以秒為單位。預設值為 '180' (字串)。

```
node[:mysql][:tunable][:wait_timeout]
```

table_cache

開啟的資料表數目 (字串)。預設值為 '2048'。

```
node[:mysql][:tunable][:table_cache]
```

nginx 屬性

Important

AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 這些屬性僅適用於 Linux 堆疊。

nginx 屬性指定 Nginx 組態。如需詳細資訊，請參閱 Directive Index。如需如何覆寫內建屬性以指定自訂值的詳細資訊，請參閱覆寫屬性。

binary	dir	gzip
gzip_comp_level	gzip_disable	gzip_http_version
gzip_proxied	gzip_static	gzip_types
gzip_vary	keepalive	keepalive_timeout
log_dir	使用者	server_names_hash_bucket_size
worker_processes	worker_connections	

binary

Nginx 二進位檔的位置 (字串)。預設值為 `'/usr/sbin/nginx'`。

```
node[:nginx][:binary]
```

dir

檔案 (例如組態檔案) 的位置 (字串)。預設值為 `'/etc/nginx'`。

```
node[:nginx][:dir]
```

gzip

是否啟用 gzip 壓縮 (字串)。可能的值為 `'on'` 和 `'off'`。預設值為 `'on'`。

> **Warning**
>
> 壓縮可能會引發安全風險。若要完全停用壓縮，請設定此屬性如下：

```
node[:nginx][:gzip] = 'off'
```

```
node[:nginx][:gzip]
```

gzip_comp_level

壓縮層級的範圍介於 1–9 之間，其中 1 對應於最小壓縮 （字串）。預設值為 '2'。

```
node[:nginx][:gzip_comp_level]
```

gzip_disable

停用指定使用者代理程式的 gzip 壓縮 (字串)。此值為規則表達式，預設值為 'MSIE [1-6].(?!.*SV1)'。

```
node[:nginx][:gzip_disable]
```

gzip_http_version

啟用指定 HTTP 版本的 gzip 壓縮 (字串)。預設值為 '1.0'。

```
node[:nginx][:gzip_http_version]
```

gzip_proxied

是否及如何壓縮代理請求的回應，可接受下列其中一個值 (字串)：

- 'off'：不壓縮代理請求
- 'expired'：Expire 標頭防止快取時壓縮
- 'no-cache'：Cache-Control 標頭設為 "no-cache" 時壓縮
- 'no-store'：Cache-Control 標頭設為 "no-store" 時壓縮
- 'private'：Cache-Control 標頭設為 "private" 時壓縮
- 'no_last_modified'：未設定 Last-Modified 時壓縮
- 'no_etag'：請求缺少 ETag 標頭時壓縮
- 'auth'：請求包含 Authorization 標頭時壓縮
- 'any'：壓縮所有代理請求

預設值為 'any'。

```
node[:nginx][:gzip_proxied]
```

gzip_static

是否啟用 gzip 靜態模組 (字串)。可能的值為 'on' 和 'off'。預設值為 'on'。

```
node[:nginx][:gzip_static]
```

gzip_types

要壓縮的 MIME 類型清單 (字串清單)。預設值為 ['text/plain', 'text/html', 'text/css', 'application/x-javascript', 'text/xml', 'application/xml', 'application/xml+rss', 'text/javascript']。

```
node[:nginx][:gzip_types]
```

gzip_vary

是否啟用 Vary:Accept-Encoding 回應標頭 (字串)。可能的值為 'on' 和 'off'。預設值為 'on'。

```
node[:nginx][:gzip_vary]
```

keepalive

是否啟用持續連線 (字串)。可能的值為 'on' 和 'off'。預設值為 'on'。

```
node[:nginx][:keepalive]
```

keepalive_timeout

持續連線保持開啟的時間上限，以秒為單位 (數值)。預設值為 65。

```
node[:nginx][:keepalive_timeout]
```

log_dir

日誌檔案的位置 (字串)。預設值為 '/var/log/nginx'。

```
node[:nginx][:log_dir]
```

使用者

使用者 (字串)。預設值如下：

- Amazon Linux 和 RHEL：'www-data'
- Ubuntu：'nginx'

```
node[:nginx][:user]
```

server_names_hash_bucket_size

伺服器名稱雜湊表的儲存貯體大小，可設為 32、64 或 128 (數值)。預設值為 64。

```
node[:nginx][:server_names_hash_bucket_size]
```

worker_processes

工作者處理序數目 (數值)。預設值為 10。

```
node[:nginx][:worker_processes]
```

worker_connections

工作者連線數目上限 (數值)。預設值為 1024。用戶端數目上限會設為 worker_processes * worker_connections。

```
node[:nginx][:worker_connections]
```

opsworks_berkshelf 屬性

Important

AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 這些屬性僅適用於 Linux 堆疊。

opsworks_berkshelf 屬性指定 Berkshelf 組態。如需詳細資訊，請參閱 Berkshelf。如需如何覆寫內建屬性以指定自訂值的詳細資訊，請參閱覆寫屬性。

偵錯

是否在 Chef 日誌中包含 Berkshelf 除錯資訊 (布林值)。預設值為 `false`。

```
node['opsworks_berkshelf]['debug']
```

opsworks_java 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 這些屬性僅適用於 Linux 堆疊。

opsworks_java 屬性指定 Tomcat 伺服器組態。如需詳細資訊，請參閱 Apache Tomcat Configuration Reference。如需如何覆寫內建屬性以指定自訂值的詳細資訊，請參閱覆寫屬性。

datasources	java_app_server_version	java_shared_lib_dir
jvm_pkg 屬性	custom_pkg_location_url_debian	java_home_basedir
custom_pkg_location_url_rhel	use_custom_pkg_location	jvm_options

jvm_version	tomcat 屬性	

datasources

定義 JNDI 資源名稱的一組屬性 (字串)。如需如何使用此屬性的詳細資訊，請參閱使用後端資料庫部署 JSP 應用程式。預設值是空的雜湊，可填入應用程式簡短名稱與 JNDI 名稱之間的自訂映射。如需詳細資訊，請參閱使用後端資料庫部署 JSP 應用程式。

```
node['opsworks_java']['datasources']
```

java_app_server_version

Java 應用程式伺服器版本 (數值)。預設值為 7。您可以覆寫此屬性來指定版本 6。如果您安裝非預設 JDK，則忽略此屬性。

```
node['opsworks_java']['java_app_server_version']
```

java_shared_lib_dir

Java 共享程式庫的目錄 (字串)。預設值為 `/usr/share/java`。

```
node['opsworks_java']['java_shared_lib_dir']
```

jvm_pkg 屬性

您可以覆寫來安裝非預設 JDK 的一組屬性。

use_custom_pkg_location

是否安裝自訂 JDK 而不是 OpenJDK (布林值)。預設值為 `false`。

```
node['opsworks_java']['jvm_pkg']['use_custom_pkg_location']
```

custom_pkg_location_url_debian

要在 Ubuntu 執行個體上安裝 JDK 套件的位置 (字串)。預設值為 `'http://aws.amazon.com/'`，這只是沒有適當意義的初始化值。如果您想要安裝非預設 JDK，您必須覆寫此屬性並將其設為適當的 URL。

```
node['opsworks_java']['jvm_pkg']['custom_pkg_location_url_debian']
```

custom_pkg_location_url_rhel

要在 Amazon Linux 和 RHEL 執行個體上安裝 JDK 套件的位置 (字串)。預設值為 'http://aws.amazon.com/'，這只是沒有適當意義的初始化值。如果您想要安裝非預設 JDK，您必須覆寫此屬性並將其設為適當的 URL。

```
node['opsworks_java']['jvm_pkg']['custom_pkg_location_url_rhel']
```

java_home_basedir

要解壓縮 JDK 套件的目標目錄 (字串)。預設值為 /usr/local。您不需要為 RPM 套件指定此設定；這些套件會包含完整的目錄結構。

```
node['opsworks_java']['jvm_pkg']['java_home_basedir']
```

jvm_options

JVM 命令列選項，可讓您指定堆積大小等設定 (字串)。一組常見選項為 -Djava.awt.headless=true -Xmx128m -XX:+UseConcMarkSweepGC。預設值為沒有選項。

```
node['opsworks_java']['jvm_options']
```

jvm_version

OpenJDK 版本 (數值)。預設值為 7。您可以覆寫此屬性來指定 OpenJDK 6 版。如果您安裝非預設 JDK，則忽略此屬性。

```
node['opsworks_java']['jvm_version']
```

tomcat 屬性

您可以覆寫來安裝預設 Tomcat 組態的一組屬性。

ajp_port	apache_tomcat_bind_mod	apache_tomcat_bind_path

auto_deploy	connection_timeout	mysql_connector_jar
port	secure_port	shutdown_port
threadpool_max_threads	threadpool_min_spare_threads	unpack_wars
uri_encoding	use_ssl_connector	use_threadpool
userdatabase_pathname		

ajp_port

AJP 連接埠 (數值)。預設值為 8009。

```
node['opsworks_java']['tomcat]['ajp_port']
```

apache_tomcat_bind_mod

代理模組 (字串)。預設值為 proxy_http。您可以覆寫此屬性來指定 AJP 代理模組 proxy_ajp。

```
node['opsworks_java']['tomcat]['apache_tomcat_bind_mod']
```

apache_tomcat_bind_path

Apache-Tomcat 繫結路徑 (字串)。預設值為 /。您不應該覆寫此屬性；變更繫結路徑可能會導致應用程式停止運作。

```
node['opsworks_java']['tomcat]['apache_tomcat_bind_path']
```

auto_deploy

是否自動部署 (布林值)。預設值為 true。

```
node['opsworks_java']['tomcat]['auto_deploy']
```

connection_timeout

連線逾時，以毫秒為單位 (數值)。預設值為 20000 (20 秒)。

```
node['opsworks_java']['tomcat]['connection_timeout']
```

mysql_connector_jar

MySQL 連接器程式庫的 JAR 檔案 (字串)。預設值為 `mysql-connector-java.jar`。

```
node['opsworks_java']['tomcat]['mysql_connector_jar']
```

port

標準連接埠 (數值)。預設值為 `8080`。

```
node['opsworks_java']['tomcat]['port']
```

secure_port

安全連接埠 (數值)。預設值為 `8443`。

```
node['opsworks_java']['tomcat]['secure_port']
```

shutdown_port

關機連接埠 (數值)。預設值為 `8005`。

```
node['opsworks_java']['tomcat]['shutdown_port']
```

threadpool_max_threads

執行緒集區中的執行緒數目上限 (數值)。預設值為 `150`。

```
node['opsworks_java']['tomcat]['threadpool_max_threads']
```

threadpool_min_spare_threads

執行緒集區中的備用執行緒數目下限 (數值)。預設值為 `4`。

```
node['opsworks_java']['tomcat]['threadpool_min_spare_threads']
```

unpack_wars

是否解壓縮 WAR 檔案 (布林值)。預設值為 `true`。

```
node['opsworks_java']['tomcat]['unpack_wars']
```

uri_encoding

URI 編碼 (字串)。預設值為 UTF-8。

```
node['opsworks_java']['tomcat]['uri_encoding']
```

use_ssl_connector

是否使用 SSL 連接器 (布林值)。預設值為 `false`。

```
node['opsworks_java']['tomcat]['use_ssl_connector']
```

use_threadpool

是否使用執行緒集區 (布林值)。預設值為 `false`。

```
node['opsworks_java']['tomcat]['use_threadpool']
```

userdatabase_pathname

使用者資料庫路徑名稱 (字串)。預設值為 `conf/tomcat-users.xml`。

```
node['opsworks_java']['tomcat]['userdatabase_pathname']
```

passenger_apache2 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 這些屬性僅適用於 Linux 堆疊。

`passenger_apache2` 屬性指定 Phusion Passenger 組態。如需詳細資訊，請參閱 Phusion Passenger users guide, Apache version。如需如何覆寫內建屬性以指定自訂值的詳細資訊，請參閱覆寫屬性。

friendly_error_pages	gem_bin	gems_path
high_performance_mode	root_path	max_instances_per_app
max_pool_size	max_requests	module_path
pool_idle_time	rails_app_spawner_idle_time	rails_framework_spawner_idle_time
rails_spawn_method	ruby_bin	ruby_wrapper_bin
stat_throttle_rate	version	

friendly_error_pages

是否在應用程式無法啟動時顯示易用的錯誤頁面 (字串)。此屬性可設為 'on' 或 'off'；預設值為 'off'。

```
node[:passenger][:friendly_error_pages]
```

gem_bin

Gem 二進位檔的位置 (字串)。預設值為 `'/usr/local/bin/gem'`。

```
node[:passenger][:gem_bin]
```

gems_path

Gem 路徑 (字串)。預設值視 Ruby 版本而定。例如：

- Ruby 1.8 版：`'/usr/local/lib/ruby/gems/1.8/gems'`

- Ruby 1.9 版：`'/usr/local/lib/ruby/gems/1.9.1/gems'`

```
node[:passenger][:gems_path]
```

high_performance_mode

是否使用 Passenger 的高效能模式 (字串)。可能的值為 `'on'` 和 `'off'`。預設值為 `'off'`。

```
node[:passenger][:high_performance_mode ]
```

root_path

Passenger 根目錄 (字串)。預設值視 Ruby 和 Passenger 版本而定。在 Chef 語法中，此值為 `"#{node[:passenger][:gems_path]}/passenger-#{passenger[:version]}"`。

```
node[:passenger][:root_path]
```

max_instances_per_app

每個應用程式的應用程式處理序數目上限 (數值)。預設值為 `0`。如需詳細資訊，請參閱 PassengerMaxInstancesPerApp。

```
node[:passenger][:max_instances_per_app]
```

max_pool_size

應用程式處理器數目上限 (數值)。預設值為 8。如需詳細資訊，請參閱 PassengerMaxPoolSize。

```
node[:passenger][:max_pool_size]
```

max_requests

請求數目上限 (數值)。預設值為 `0`。

```
node[:passenger][:max_requests]
```

module_path

模組路徑 (字串)。預設值如下：

- Amazon Linux 和 RHEL：`"#{node['apache']['libexecdir']}/mod_passenger.so"`

- Ubuntu：`"#{passenger[:root_path]}/ext/apache2/mod_passenger.so"`

```
node[:passenger][:module_path]
```

pool_idle_time

應用程式處理序可處於閒置狀態的時間上限，以秒為單位 (數值)。預設值為 `14400` (4 小時)。如需詳細資訊，請參閱 PassengerPoolIdleTime。

```
node[:passenger][:pool_idle_time]
```

rails_app_spawner_idle_time

Rails 應用程式 spawner 的閒置時間上限 (數值)。如果此屬性設為零，應用程式 spawner 不會逾時。預設值為 `0`。如需詳細資訊，請參閱 Spawning Methods Explained。

```
node[:passenger][:rails_app_spawner_idle_time]
```

rails_framework_spawner_idle_time

Rails 架構 spawner 的閒置時間上限 (數值)。如果此屬性設為零，架構 spawner 不會逾時。預設值為 `0`。如需詳細資訊，請參閱 Spawning Methods Explained。

```
node[:passenger][:rails_framework_spawner_idle_time]
```

rails_spawn_method

Rails 繁衍方法 (字串)。預設值為 `'smart-lv2'`。如需詳細資訊，請參閱 Spawning Methods Explained。

```
node[:passenger][:rails_spawn_method]
```

ruby_bin

Ruby 二進位檔的位置 (字串)。預設值為 `'/usr/local/bin/ruby'`。

```
node[:passenger][:ruby_bin]
```

ruby_wrapper_bin

Ruby 包裝函式指令碼的位置 (字串)。預設值為 `'/usr/local/bin/ruby_gc_wrapper.sh'`。

```
node[:passenger][:ruby_wrapper_bin]
```

stat_throttle_rate

Passenger 執行檔案系統檢查的速率 (數值)。預設值為 5，這表示最多每 5 秒執行檢查一次。如需詳細資訊，請參閱 PassengerStatThrottleRate。

```
node[:passenger][:stat_throttle_rate]
```

version

版本 (字串)。預設值為 '3.0.9'。

```
node[:passenger][:version]
```

ruby 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 這些屬性僅適用於 Linux 堆疊。

ruby 屬性指定應用程式使用的 Ruby 版本。請注意，屬性用量會隨著引進 Ruby 2.1 中的語意版本控制而產生變更。如需如何指定版本的詳細資訊，包括範例，請參閱 Ruby 版本。如需 AWS OpsWorks Stacks 如何決定 Ruby 版本的完整詳細資訊，請參閱內建屬性檔案 ruby.rb。如需如何覆寫內建屬性以指定自訂值的詳細資訊，請參閱覆寫屬性。

full_version

完整版本號碼 (字串)。您不應該覆寫此屬性。請改用 [:opsworks][:ruby_version] 和適當的修補程式版本屬性來指定版本。

```
[:ruby][:full_version]
```

major_version

主要版本號碼 (字串)。您不應該覆寫此屬性。請改用 [:opsworks][:ruby_version] 來指定主要版本。

```
[:ruby][:major_version]
```

minor_version

次要版本號碼 (字串)。您不應該覆寫此屬性。請改用 [:opsworks][:ruby_version] 來指定次要版本。

```
[:ruby][:minor_version]
```

patch

修補程式等級 (字串)。此屬性適用於 Ruby 2.0.0 版及舊版。對於更新版本的 Ruby，請使用 `patch_version` 屬性。

```
[:ruby][:patch]
```

修補程式號碼前面必須加上 p。例如，您會使用下列自訂 JSON 來指定修補程式等級 484。

```
{
  "ruby":{"patch":"p484"}
}
```

patch_version

修補程式號碼 (字串)。此屬性適用於 Ruby 2.1 版及更新版本。對於舊版的 Ruby，請使用 `patch` 屬性。

```
[:ruby][:patch_version]
```

pkgrelease

套件版次號碼 (字串)。

```
[:ruby][:pkgrelease]
```

unicorn 屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 這些屬性僅適用於 Linux 堆疊。

`unicorn` 屬性指定 Unicorn 組態。如需詳細資訊，請參閱 Unicorn::Configurator。如需如何覆寫內建屬性以指定自訂值的詳細資訊，請參閱覆寫屬性。

accept_filter	backlog	延遲
tcp_nodelay	tcp_nopush	preload_app
timeout	tries	version
worker_processes		

accept_filter

接受篩選條件 `'httpready'` 或 `'dataready'` (字串)。預設值為 `'httpready'`。

```
node[:unicorn][:accept_filter]
```

backlog

佇列可保留的請求數目上限 (數值)。預設值為 `1024`。

```
node[:unicorn][:backlog]
```

延遲

等待重試繫結通訊端的時間，以秒為單位 (數值)。預設值為 `0.5`。

```
node[:unicorn][:delay]
```

preload_app

是否預先載入應用程式，再分支處理工作者處理序 (布林值)。預設值為 `true`。

```
node[:unicorn][:preload_app]
```

tcp_nodelay

是否停用 TCP 通訊端的 Nagle 演算法 (布林值)。預設值為 `true`。

```
node[:unicorn][:tcp_nodelay]
```

tcp_nopush

是否啟用 TCP_CORK (布林值)。預設值為 `false`。

```
node[:unicorn][:tcp_nopush]
```

timeout

工作者可用於每個請求的時間上限，以秒為單位 (數值)。超過逾時值的工作者會終止。預設值為 `60`。

```
node[:unicorn][:timeout]
```

tries

重試繫結至通訊端的次數上限 (數值)。預設值為 `5`。

```
node[:unicorn][:tries]
```

version

Unicorn 版本 (字串)。預設值為 `'4.7.0'`。

```
node[:unicorn][:version]
```

worker_processes

工作者處理序數目 (數值)。如果存在，預設值為 `max_pool_size`；否則為 4。

```
node[:unicorn][:worker_processes]
```

對適用於 Linux 的 Chef 11.10 和舊版故障診斷

Important

AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Note

如需其他故障診斷資訊，請參閱偵錯和故障診斷指南。

適用於 Linux 之 Chef 11.10 和舊版的 Chef 日誌

AWS OpsWorks Stacks 會將每個執行個體的 Chef 日誌存放在其`/var/lib/aws/opsworks/chef`目錄中。您需要 sudo 權限才能存取此目錄。每個回合的日誌位於名為 *YYYY*-*MM*-*DD*-*HH*-*MM*-*SS*-*NN*.log 的檔案中。

如需詳細資訊，請參閱下列內容：

- 使用主控台檢視 Chef 日誌
- 使用 CLI 或 API 檢視 Chef 日誌
- 解讀 Chef 日誌
- 常見的 Chef 日誌錯誤

搭配其他 AWS 服務使用 AWS OpsWorks Stacks

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以讓應用程式伺服器在 Stacks AWS OpsWorks 堆疊中執行，使用各種未與 Stacks 直接整合的 AWS AWS OpsWorks 服務。例如，您可以讓您的應用程式伺服器使用 Amazon RDS 做為後端資料庫。您可以藉由使用下列一般模式，存取這類服務：

1. 透過使用 AWS 主控台、API 或 CLI 建立和設定 AWS 服務，並記錄任何應用程式需要存取服務的必要資訊，例如主機名稱或連接埠。
2. 建立一或多個配方以設定應用程式，使其能夠存取服務。

   配方會從您在執行配方前使用自訂 JSON 定義的堆疊組態及部署 JSON 屬性取得組態資料。
3. 將自訂配方指派給應用程式伺服器 layer 上的部署生命週期事件。
4. 建立將適當的值指派給組態資料屬性的自訂 JSON 物件，並將其新增至您的堆疊組態及部署 JSON。
5. 將應用程式部署到堆疊。

   部署會執行自訂配方，使用您在自訂 JSON 中定義的組態資料值設定應用程式，使其能夠存取服務。

本節說明如何讓 AWS OpsWorks Stacks 應用程式伺服器存取各種 AWS 服務。它假設您已熟悉 Chef 技術指南以及配方使用堆疊和組態 JSON 屬性設定應用程式的方式 (通常是透過建立組態檔案)。若您尚未熟悉，建議您先閱讀技術指南和配方和自訂 AWS OpsWorks Stacks。

主題

- 使用後端資料存放區
- 使用 ElastiCache Redis 做為記憶體內鍵值存放區
- 使用 Amazon S3 儲存貯體
- AWS CodePipeline 搭配 Stacks AWS OpsWorks 使用

使用後端資料存放區

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

應用程式伺服器堆疊通常包含資料庫伺服器，以提供後端資料存放區。 AWS OpsWorks 堆疊透過 MySQL layer 為 MySQL 伺服器提供整合支援，並透過 Amazon Relational Database Service (Amazon RDS) layer 為多種類型的資料庫伺服器提供整合支援。不過，您可以輕鬆自訂堆疊，讓應用程式伺服器使用其他資料庫伺服器，例如 Amazon DynamoDB 或 MongoDB。本主題說明將應用程式伺服器連線至 AWS 資料庫伺服器的基本程序。它使用 Chef 11 Linux 堆疊入門中的堆疊和應用程式，以顯示如何手動將 PHP 應用程式伺服器連線至 RDS 資料庫。雖然範例是根據 Linux 堆疊，但基本原則也適用於 Windows 堆疊。如需如何將 MongoDB 資料庫伺服器併入堆疊的範例，請參閱使用 OpsWorks 部署 MongoDB。

> **Note**
>
> 本主題使用 Amazon RDS 做為方便的範例。不過，如果您想要將 Amazon RDS 資料庫與堆疊搭配使用，則使用 Amazon RDS 層會更簡單。

主題

- 如何設定資料庫連線
- 如何將 Application Server 執行個體連線至 Amazon RDS

如何設定資料庫連線

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以使用自訂配方，來設定應用程式伺服器與其後端資料庫之間的連線。配方會視需要設定應用程式伺服器，一般是透過建立組態檔案。配方會從堆疊組態中的一組屬性，以及 Stacks 在每個執行個體上安裝的部署屬性，取得連線資料，例如主機和資料庫名稱。 AWS OpsWorks

例如， 的步驟 2 Chef 11 Linux 堆疊入門 是以名為 MyStack 的堆疊為基礎，堆疊有兩個層：PHP App Server 和 MySQL，每個堆疊都有一個執行個體。您可以將名為 SimplePHPApp 的應用程式部署到使用 MySQL 執行個體上的資料庫做為後端資料存放區的 PHP App Server 執行個體。當您部署應用程式時， AWS OpsWorks Stacks 會安裝包含資料庫連線資訊的堆疊組態和部署屬性。下列範例顯示資料庫連線屬性 (以 JSON 呈現)：

```
{
  ...
  "deploy": {
    "simplephpapp": {
      ...
      "database": {
        "reconnect": true,
        "password": null,
        "username": "root",
        "host": null,
        "database": "simplephpapp"
        ...
      },
      ...
    }
  }
}
```

屬性值由 Stacks AWS OpsWorks 提供，並且會根據使用者提供的資訊產生或產生。

若要允許 SimplePHPApp 存取資料存放區，您必須將名為 的自訂配方指派給 PHP App Server layer 的部署生命週期事件，以設定 `appsetup.rb` PHP 應用程式伺服器與 MySQL 資料庫之間的連線。當您部署 SimplePHPApp 時， AWS OpsWorks Stacks 會執行 `appsetup.rb`，這會建立名為 的組態檔案`db-connect.php`來設定連線，如下列摘錄所示。

```
node[:deploy].each do |app_name, deploy|
  ...
  template "#{deploy[:deploy_to]}/current/db-connect.php" do
    source "db-connect.php.erb"
```

```
    mode 0660
    group deploy[:group]

    if platform?("ubuntu")
      owner "www-data"
    elsif platform?("amazon")
      owner "apache"
    end

    variables(
      :host =>     (deploy[:database][:host] rescue nil),
      :user =>     (deploy[:database][:username] rescue nil),
      :password => (deploy[:database][:password] rescue nil),
      :db =>       (deploy[:database][:database] rescue nil),
      :table =>    (node[:phpapp][:dbtable] rescue nil)
    )
    ...
  end
end
```

描述連線特性的變數 host-user、 等 - 從部署的 JSON `[:deploy][:app_name][:database]` 屬性設定對應的值。為求簡化，此範例假設您已建立名為 `urler` 的表格，因此表格名稱是由技術指南屬性檔案中的 `[:phpapp][:dbtable]` 所呈現。

此配方實際上可以將 PHP 應用程式伺服器連接到任何 MySQL 資料庫伺服器，而不只是 MySQL layer 的成員。若要使用不同的 MySQL 伺服器，您只需要將`[:database]`屬性設定為適合伺服器的值，您可以使用自訂 JSON 來執行此操作。 AWS OpsWorks 堆疊接著將這些屬性和值整合到堆疊組態和部署屬性中，`appsetup.rb`並使用它們來建立設定連線的範本。如需覆寫堆疊組態和部署 JSON 的詳細資訊，請參閱覆寫屬性。

如何將 Application Server 執行個體連線至 Amazon RDS

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節說明如何從 Chef 11 Linux 堆疊入門自訂 MyStack，讓 PHP 應用程式伺服器連線至 RDS 執行個體。

主題

- 建立 Amazon RDS MySQL 資料庫
- 自訂堆疊以連線至 RDS 資料庫

建立 Amazon RDS MySQL 資料庫

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

現在，您已準備好使用 Amazon RDS 主控台的啟動資料庫執行個體精靈為範例建立 RDS 資料庫。下列程序是基本詳細資訊的簡短摘要。如需如何建立資料庫的詳細描述，請參閱 Amazon RDS 入門。

建立 Amazon RDS 資料庫

1. 如果這是您第一次建立 RDS 資料庫，請按一下 Get Started Now (立即開始)。否則，按一下導覽窗格中的 RDS Dashboard (RDS 儀表板)，然後按一下 Launch a DB Instance (啟動資料庫執行個體)。
2. 選取 MySQL Community Edition 做為資料庫執行個體。
3. 針對 Do you plan to use this database for production purposes? (您打算以生產為目的使用此資料庫嗎？)，選取 No, this instance... (否，此執行個體...)，這對此範例就已足夠。針對生產用途，建議您選取 Yes, use Multi-AZ Deployment... (是，使用異地同步備份部署...)。按一下Next Step (下一步)。
4. 在 Specify DB Details (指定資料庫詳細資訊) 頁面上，指定下列設定：
   - DB Instance Class (資料庫執行個體類別)：db.t2.micro。
   - Multi-AZ Deployment (異地同步備份部署)：No (否)
   - Allocated Storage (配置儲存體)：**5** GB
   - DB Instance Identifier (資料庫執行個體識別符)：**`rdsexample`**
   - Master Username (主要使用者名稱)：**`opsworksuser`**
   - Master Password (主要密碼)：指定並記錄適合的密碼，以供日後使用。

接受其他選項的預設設定，然後按一下 Next Step (下一步)。

5. 在 Configure Advanced Settings (設定進階設定) 頁面上，指定下列設定：

   - 在 Network & Security (網路和安全) 區段中，針對 VPC Security Group(s) (VPC 安全群組)，選取 phpsecgroup (VPC)
   - 在 Database Options (資料庫選項) 區段中，針對 Database Name (資料庫名稱)，輸入 **`rdsexampledb`**。
   - 在 Backup (備份) 區段中，針對本演練的用途，將 Backup Retention Period (備份保留期) 設定為 0。

   接受其他選項的預設設定，然後按一下 Launch DB Instance (啟動資料庫執行個體)。

6. 選擇 View Your DB Instances (檢視資料庫執行個體)，以查看資料庫執行個體清單。
7. 在清單中選取 rdsexample 執行個體，然後按一下箭頭以顯示執行個體端點和其他詳細資訊。記錄端點，供日後使用。它應該類似 `rdsexample.c6c8mntzhgv0.us-west-2.rds.amazonaws.com:3306`。只需要記錄 DNS 名稱；您不需要連接埠號碼。
8. 使用 MySQL Workbench 這類工具，利用下列 SQL 命令在 `urler` 資料庫中建立名為 `rdsexampledb` 的表格：

```
CREATE TABLE urler(id INT UNSIGNED NOT NULL AUTO_INCREMENT,author VARCHAR(63) NOT
 NULL,message TEXT,PRIMARY KEY (id))
```

自訂堆疊以連線至 RDS 資料庫

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

在您建立 RDS 執行個體以當成 PHP 應用程式伺服器的後端資料庫使用之後，即可從 Chef 11 Linux 堆疊入門自訂 MyStack。

將 PHP 應用程式伺服器連線至 RDS 資料庫

1. 開啟 AWS OpsWorks Stacks 主控台，並使用包含一個執行個體的 PHP App Server layer 建立堆疊，並部署 SimplePHPApp，如 中所述Chef 11 Linux 堆疊入門。此堆疊使用 SimplePHPApp 的第 1 版，而此版本不使用資料庫連線。
2. 更新堆疊組態以使用包括 `appsetup.rb` 配方的自訂技術指南，以及相關的範本和屬性檔案。
   1. 將 Use custom Chef cookbooks (使用自訂 Chef 技術指南) 設為 Yes (是)。
   2. 將 Repository type (儲存庫類型) 設定為 Git，並將 Repository URL (儲存庫 URL) 設為 `git://github.com/amazonwebservices/opsworks-example-cookbooks.git`。
3. 將下列內容新增至堆疊的 Custom Chef JSON (自訂 Chef JSON) 方塊，以將 RDS 連線資料指派給 `appsetup.rb` 用來建立組態檔案的 `[:database]` 屬性。

```
{
  "deploy": {
    "simplephpapp": {
      "database": {
        "username": "opsworksuser",
        "password": "your_password",
        "database": "rdsexampledb",
        "host": "rds_endpoint",
        "adapter": "mysql"
      }
    }
  }
}
```

使用下列屬性值：

- username：您在建立 RDS 執行個體時所指定的主要使用者名稱。

  此範例使用 `opsworksuser`。
- password：您在建立 RDS 執行個體時所指定的主要密碼。

  請填入您指定的密碼。
- database：您在建立 RDS 執行個體時所建立的資料庫。

  此範例使用 `rdsexampledb`。

- host：RDS 執行個體的端點，這是您在上節中建立執行個體時從 RDS 主控台取得。請不要包括連接埠號碼。
- adapter：轉接器。

  此範例的 RDS 執行個體使用 MySQL，因此 adapter 設定為 `mysql`。與其他屬性不同，轉接器不會由 使用`appsetup.rb`。PHP App Server layer 的內建設定配方會改用它來建立不同的組態檔案。

4. 編輯 SimplePHPApp 組態，以指定使用後端資料庫的 SimplePHPApp 版本，如下所示：

   - Document root (文件根)：將此選項設為 `web`。
   - Branch/Revision (分支/修訂)：將此選項設為 `version2`。

   將其餘的選項保持不變。

5. 編輯 PHP App Server layer，透過將 `phpapp::appsetup`新增至 layer 的部署配方來設定資料庫連線。
6. 部署新 SimplePHPApp 版本。
7. 部署 SimplePHPApp 時，前往 Instances (執行個體) 頁面並按一下 php-app1 執行個體的公有 IP 地址，來執行應用程式。您應該會在瀏覽器中看到下列頁面，以讓您輸入文字，並將其存放在資料庫中。

> **Note**
>
> 如果您的堆疊有 MySQL layer， AWS OpsWorks Stacks 會自動將對應的連線資料指派給`[:database]`屬性。不過，如果您將自訂 JSON 指派給定義不同 `[:database]` 值的堆疊，則它們會覆寫預設值。由於`[:deploy]`屬性會安裝在每個執行個體上，任何依賴`[:database]`屬性的配方都會使用自訂連線資料，而不是 MySQL layer 的資料。如果您想要特定應用程式伺服器 layer 使用自訂連線資料，請將自訂 JSON 指派給 layer 的部署事件，並將該部署限制為該 layer。如需如何使用部署屬性的詳細資訊，請參閱部署應用程式。如需覆寫 AWS OpsWorks Stacks 內建屬性的詳細資訊，請參閱覆寫屬性。

使用 ElastiCache Redis 做為記憶體內鍵值存放區

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 本主題是以 Linux 堆疊為基礎，但 Windows 堆疊也可以使用 Amazon ElastiCache (ElastiCache)。如需如何搭配 Windows 執行個體使用 ElastiCache 的範例，請參閱 ElastiCache 作為 ASP.NET 工作階段存放區。

您通常可以透過使用快取伺服器為字串等少量資料提供記憶體內金鑰值存放區，來改善應用程式伺服器效能。Amazon ElastiCache 是一項 AWS 服務，可讓您使用 Memcached 或 Redis 快取引擎，輕鬆地為應用程式伺服器提供快取支援。 AWS OpsWorks Stacks 提供 Memcached 的內建支援。不過，如果 Redis 更符合您的需求，您可以自訂您的堆疊，讓您的應用程式伺服器使用 ElastiCache Redis。

本主題會逐步引導您完成基本程序，以 Rails 應用程式伺服器為例，為 Linux 堆疊提供 ElastiCache Redis 快取支援。它假設您已有適當的 Ruby on Rails 應用程式。如需 ElastiCache 的詳細資訊，請參閱什麼是 Amazon ElastiCache？。

主題

- 步驟 1：建立 ElastiCache Redis 叢集
- 步驟 2：設定 Rails 堆疊
- 步驟 3：建立和部署自訂技術指南
- 步驟 4：將配方指派給 LifeCycle (生命週期) 事件
- 步驟 5：將存取資訊新增至堆疊組態 JSON
- 步驟 6：部署和執行應用程式

步驟 1：建立 ElastiCache Redis 叢集

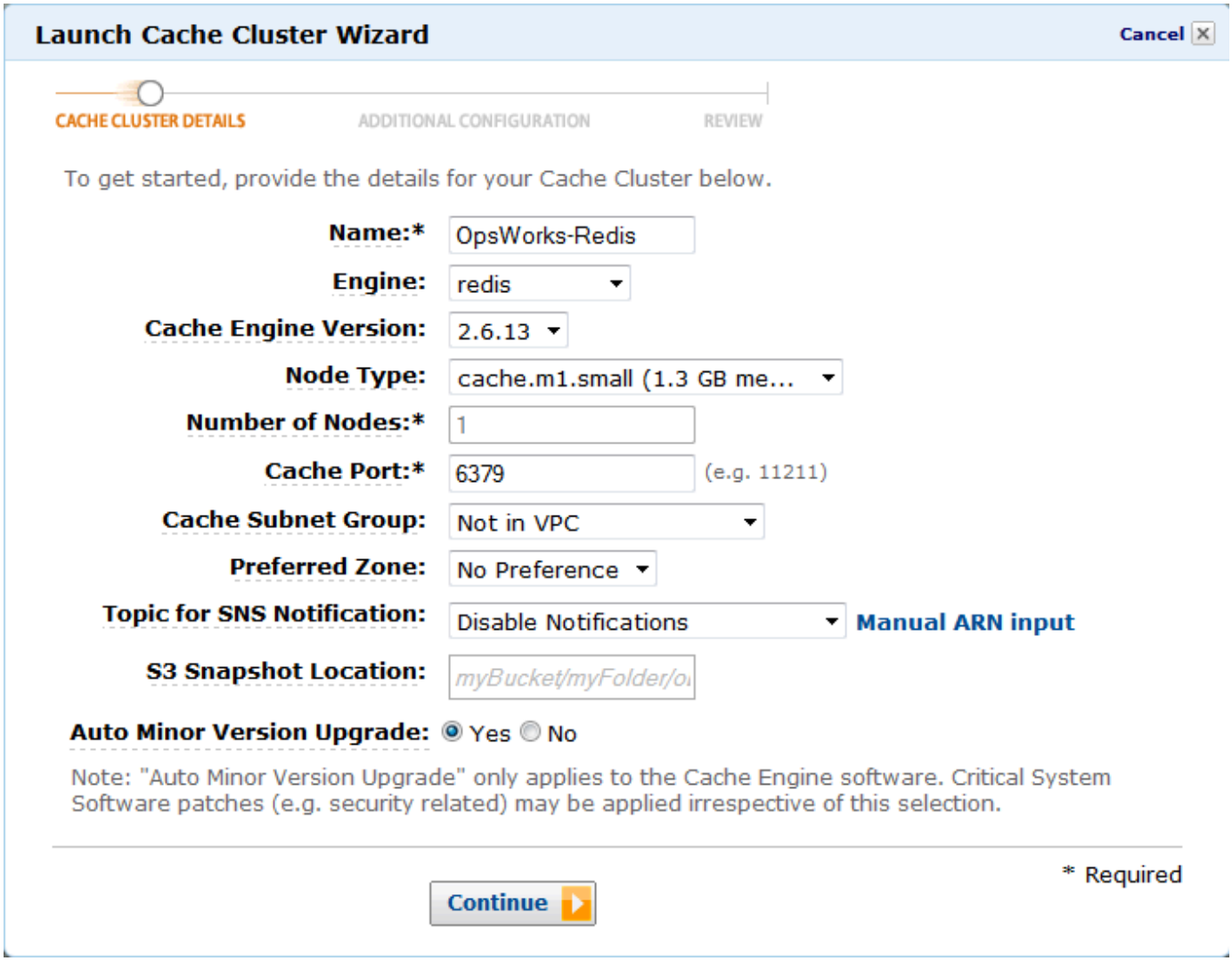
> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您必須先使用 Amazon ElastiCache ElastiCache Redis 叢集。以下說明如何使用主控台來建立叢集。

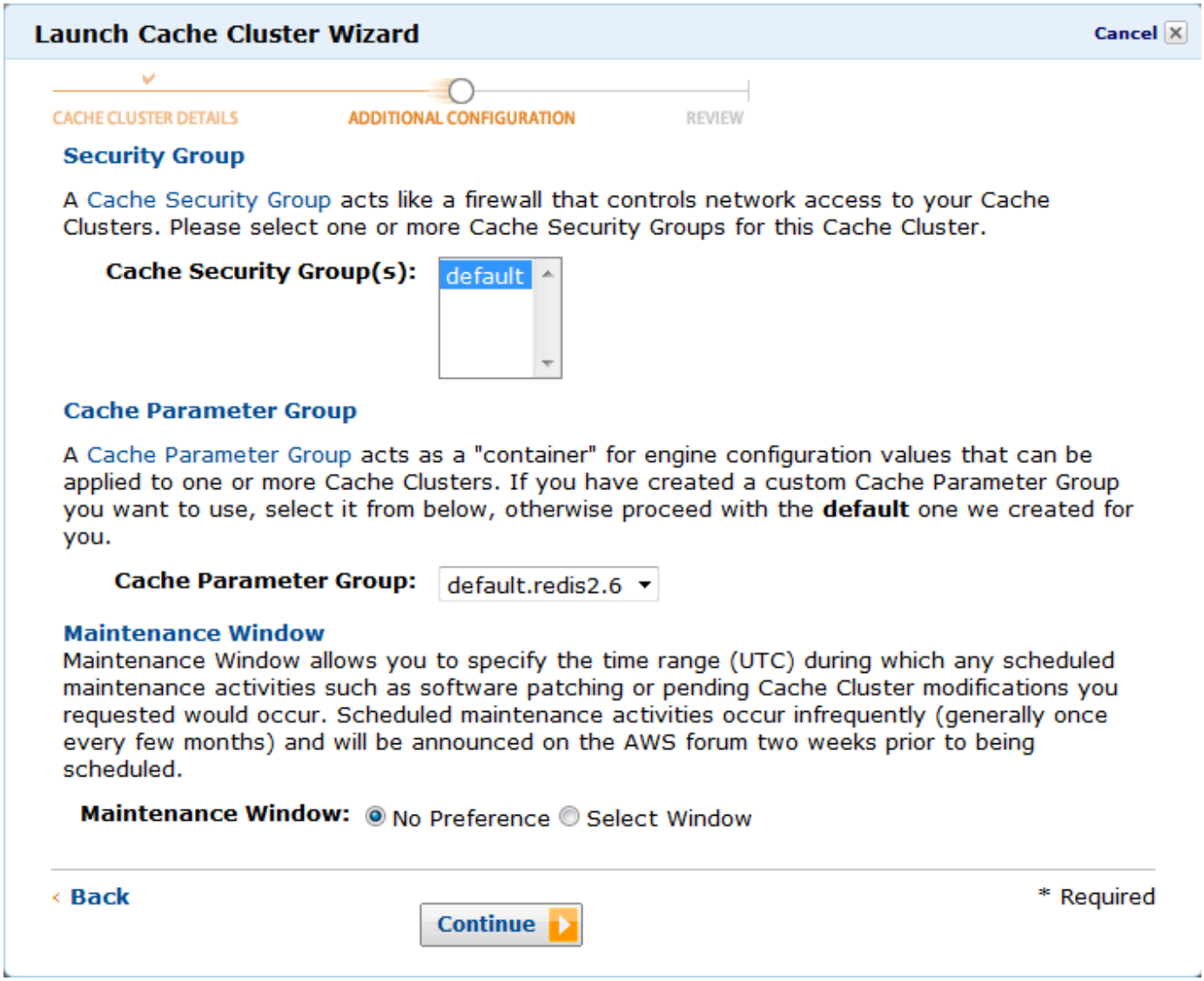
建立 ElastiCache Redis 叢集

1. 前往 ElastiCache 主控台，然後按一下啟動快取叢集以啟動快取叢集精靈。
2. 在 Cache Cluster Details (快取叢集詳細資訊) 頁面上，執行下列作業：
   - 將 Name (名稱) 設為您的快取伺服器名稱。

     此範例使用 OpsWorks-Redis。
   - 將 Engine (引擎) 設為 redis。
   - 將 Topic for SNS Notification (SNS 通知主題) 設為 Disable Notifications (停用通知)。
   - 針對其他設定接受預設值，然後按一下 Continue (繼續)。



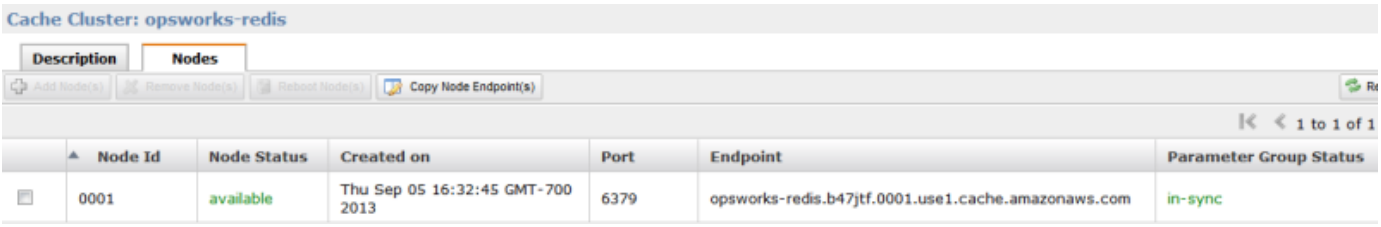
3. 在 Additional Configuration (其他組態) 頁面上，接受預設值，然後按一下 Continue (繼續)。



4. 按一下 Launch Cache Cluster (啟動快取叢集) 以建立叢集。

> **Important**
>
> 針對此範例，預設快取安全群組便已足夠，但若為生產用途，建議您建立適合您環境的快取安全群組。如需詳細資訊，請參閱管理快取安全群組。

5. 在啟動叢集後，按一下名稱以開啟詳細資訊頁面，然後按一下 Nodes (節點) 標籤。記下叢集的 Port (連接埠) 和 Endpoint (端點) 值，以供稍後使用。



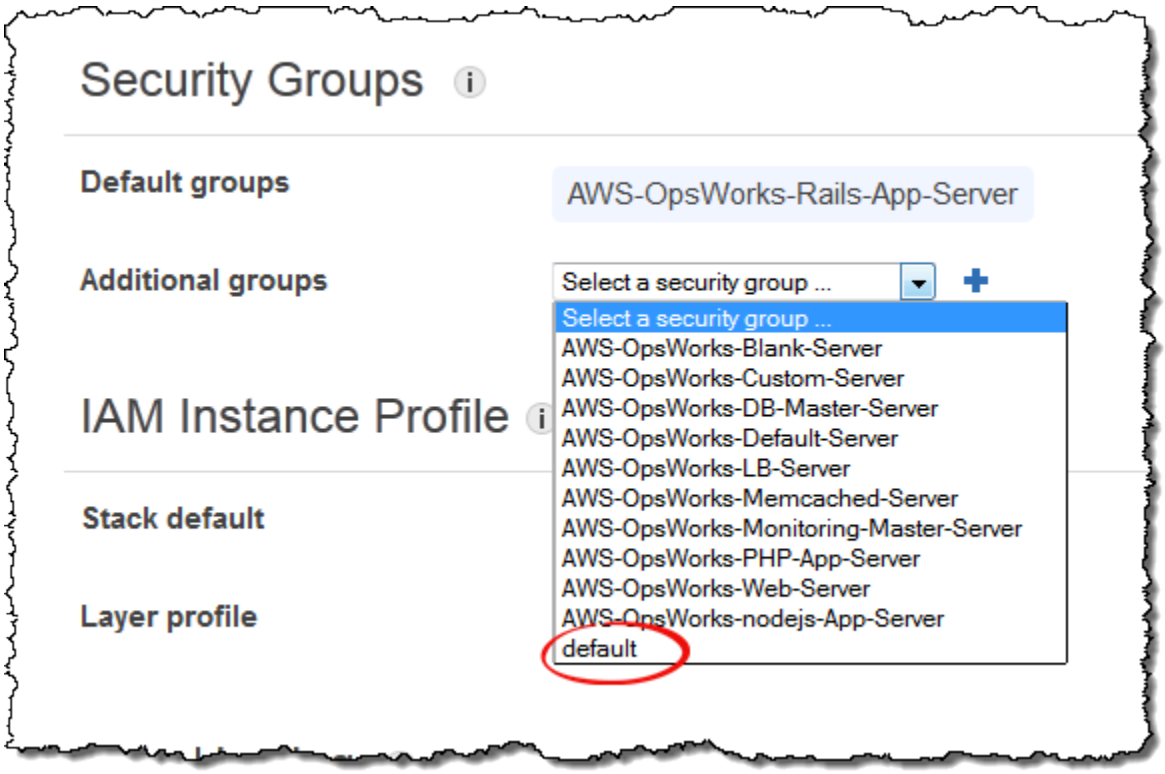
步驟 2：設定 Rails 堆疊

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

除了建立支援 Rails App Server layer 的堆疊之外，您還必須設定 layer 的安全群組，以便 Rails 伺服器可以與 Redis 伺服器正確通訊。

設定堆疊

1. 建立為此**RedisStack**範例命名的新堆疊，並新增 Rails App Server layer。您可以使用兩者的預設設定。如需詳細資訊，請參閱 建立新的堆疊 和 建立 OpsWorks Layer 。
2. 在 Layers 頁面上，針對 Rails App Server，按一下安全性，然後按一下編輯。
3. 前往安全群組區段，並將 ElastiCache 叢集的安全群組新增至其他群組。針對此範例，選取 default (預設) 安全群組，按一下 + 將它新增至 layer，然後按一下 Save (儲存) 以儲存新的組態。



4. 將執行個體新增至 Rails App Server layer 並啟動它。如需如何新增並啟動執行個體的詳細資訊，請參閱將執行個體新增至 Layer。

步驟 3：建立和部署自訂技術指南

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

目前堆疊還沒有什麼功能。您必須啟用您的應用程式，才能存取 Redis 伺服器。最靈活的方式便是在應用程式的 `config` 子資料夾中置放一個帶有存取資訊的 YAML 檔案。應用程式接著便能從檔案取得資訊。使用此方法，您可以變更連線資訊，無需重新撰寫和重新部署應用程式。在此範例中，檔案應命名`redis.yml`並包含 ElastiCache 叢集的主機名稱和連接埠，如下所示：

```
host: cache-cluster-hostname
port: cache-cluster-port
```

您可以手動將此檔案複製到您的伺服器，但更好的方法是實作 Chef 配方來產生檔案，並讓 AWS OpsWorks Stacks 在每個伺服器上執行配方。Chef 配方是專門的 Ruby 應用程式， Stacks AWS OpsWorks 會用來在安裝套件或建立組態檔案等執行個體上執行任務。配方是封裝在「技術指南」中，而技術指南可以包含多個配方和相關檔案 (例如組態檔案的範本)。技術指南會放置於儲存庫 (例如 GitHub) 中，且必須擁有標準目錄結構。如果您還沒有自訂技術指南儲存庫，請參閱技術指南儲存庫，以取得如何設定技術指南的資訊。

針對此範例，請將名為 `redis-config` 的技術指南新增至您的技術指南儲存庫，其內容如下：

```
my_cookbook_repository
  redis-config
    recipes
      generate.rb
    templates
      default
        redis.yml.erb
```

`recipes` 資料夾包含名為 `generate.rb` 的配方，該配方會從 `redis.yml.erb` 產生應用程式的組態檔案，如下所示：

```
node[:deploy].each do |app_name, deploy_config|
  # determine root folder of new app deployment
  app_root = "#{deploy_config[:deploy_to]}/current"

  # use template 'redis.yml.erb' to generate 'config/redis.yml'
  template "#{app_root}/config/redis.yml" do
    source "redis.yml.erb"
    cookbook "redis-config"

    # set mode, group and owner of generated file
    mode "0660"
    group deploy_config[:group]
    owner deploy_config[:user]

    # define variable “@redis” to be used in the ERB template
    variables(
      :redis => deploy_config[:redis] || {}
    )

    # only generate a file if there is Redis configuration
```

```
    not_if do
      deploy_config[:redis].blank?
    end
  end
end
```

配方取決於 Stacks AWS OpsWorks 堆疊組態和部署 JSON 物件的資料，該物件安裝在每個執行個體上，並包含堆疊和任何部署應用程式的詳細資訊。物件的 deploy 節點具有以下結構：

```
{
   ...
  "deploy": {
    "app1": {
      "application" : "short_name",
      ...
    }
    "app2": {
      ...
    }
    ...
  }
}
```

deploy 節點包含每個部署應用程式 (以應用程式的短名命名) 的一組內嵌 JSON 物件。每個應用程式物件都包含一組定義應用程式組態的屬性，例如文件根和應用程式類型。如需部署屬性的清單，請參閱deploy 屬性。配方可使用 Chef 屬性語法來表示堆疊組態及部署 JSON 的。例如：[:deploy][:app1][:application] 表示 app1 應用程式的短名。

針對 [:deploy] 中的每個應用程式，配方會執行關聯的程式碼區塊，其中 deploy_config 表示應用程式屬性。第一個配方會將 app_root 設為應用程式的根目錄，[:deploy][:*app_name*][:deploy_to]/current。它接著會使用 Chef 範本資源從 redis.yml.erb 產生組態檔案，然後將其置放於 app_root/config 中。

組態檔案通常會從範本建立，其中許多設定都由 Chef「屬性」定義。使用屬性，您可以於稍後使用自訂 JSON 變更設定，而無須重新撰寫範本檔案。redis.yml.erb 範本包含下列項目：

```
host: <%= @redis[:host] %>
port: <%= @redis[:port] || 6379 %>
```

<%... %> 元素為代表屬性值的預留位置。

- `<%= @redis[:host] %>` 表示 `redis[:host]` 的值，也就是快取叢集的主機名稱。
- `<%= @redis[:port] || 6379 %>` 表示 `redis[:port]` 的值或預設連接埠值 6379 (若屬性尚未定義的話)。

`template` 資源的運作方式如下：

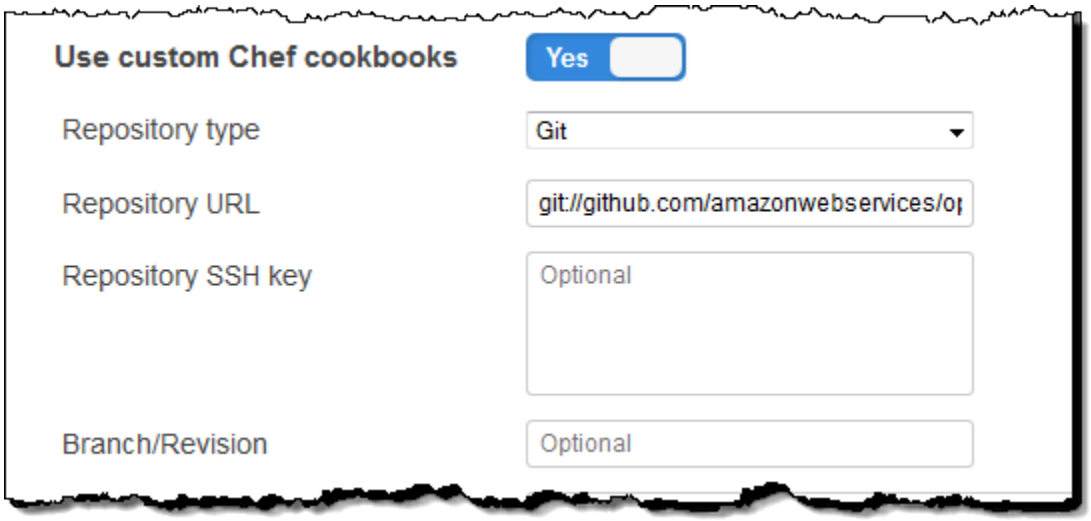
- `source` 和 `cookbook` 分別指定範本和技術指南名稱。
- `mode`、`group` 和 `owner` 會給予組態檔案與應用程式相同的存取權。
- `variables` 區段會將範本中使用的 `@redis` 變數設為應用程式的 `[:redis]` 屬性值。

  `[:redis]` 屬性值是透過使用自訂 JSON 來設定，如稍後所述。它並非標準應用程式屬性之一。
- `not_if` 指示詞會確保配方不會在已有現有檔案時再次產生組態檔案。

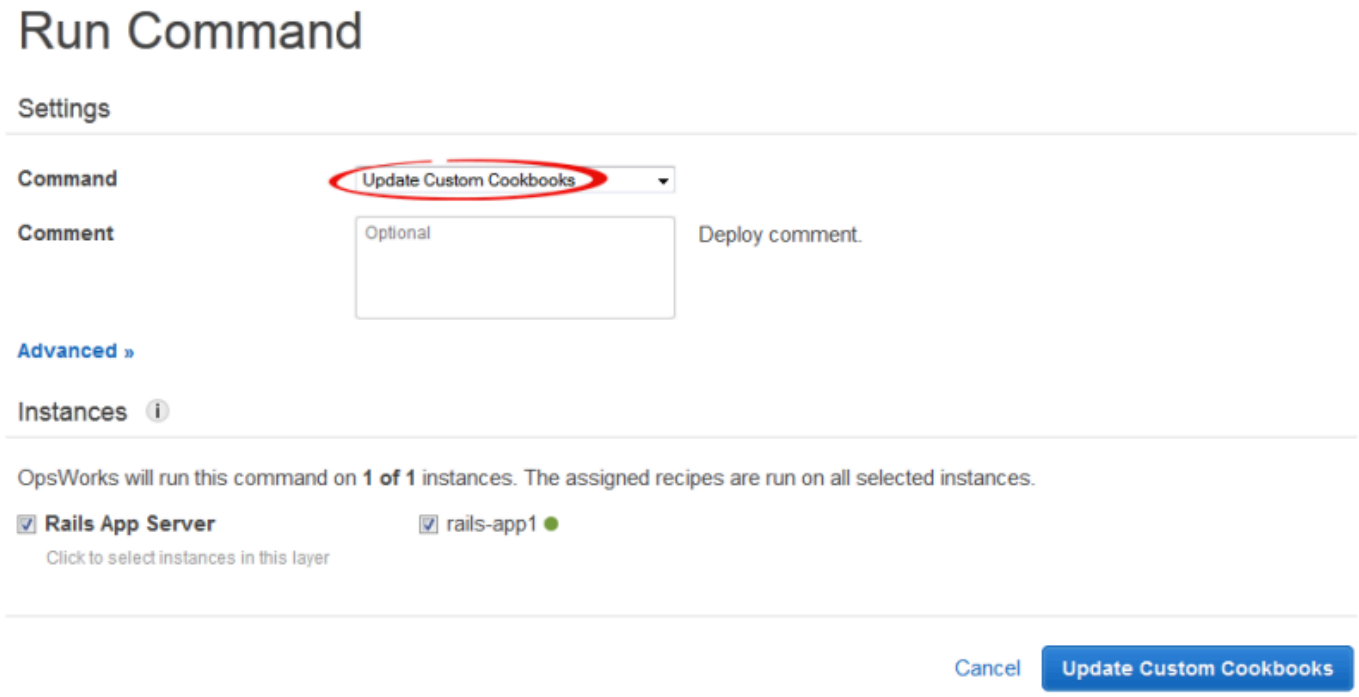
在您撰寫技術指南之後，您必須將其部署至每個執行個體的技術指南快取。此操作不會執行配方。它只會將新的技術指南安裝到堆疊的執行個體上。您通常會藉由將其指派給 layer 的生命週期事件來執行配方，如稍後所述。

部署您的自訂技術指南

1. 在 AWS OpsWorks 堆疊堆疊頁面上，按一下堆疊設定，然後按一下編輯。
2. 在 Configuration Management (組態管理) 區段中，將 Use custom Chef cookbooks (使用自訂 Chef 技術指南) 設為 Yes (是)，輸入技術指南儲存庫資訊，然後按一下 Save (儲存) 以更新堆疊組態。



3. 在 Stack (堆疊) 頁面中，按一下 Run Command (執行命令)，選取 Update Custom Cookbooks (更新自訂技術指南) 堆疊命令，然後按一下 Update Custom Cookbooks (更新自訂技術指南) 以將新的技術指南安裝到執行個體的技術指南快取中。



若您修改您的技術指南，只須再次執行 Update Custom Cookbooks (更新自訂技術指南) 即可安裝更新後的版本。如需此程序的詳細資訊，請參閱安裝自訂技術指南。

步驟 4：將配方指派給 LifeCycle (生命週期) 事件

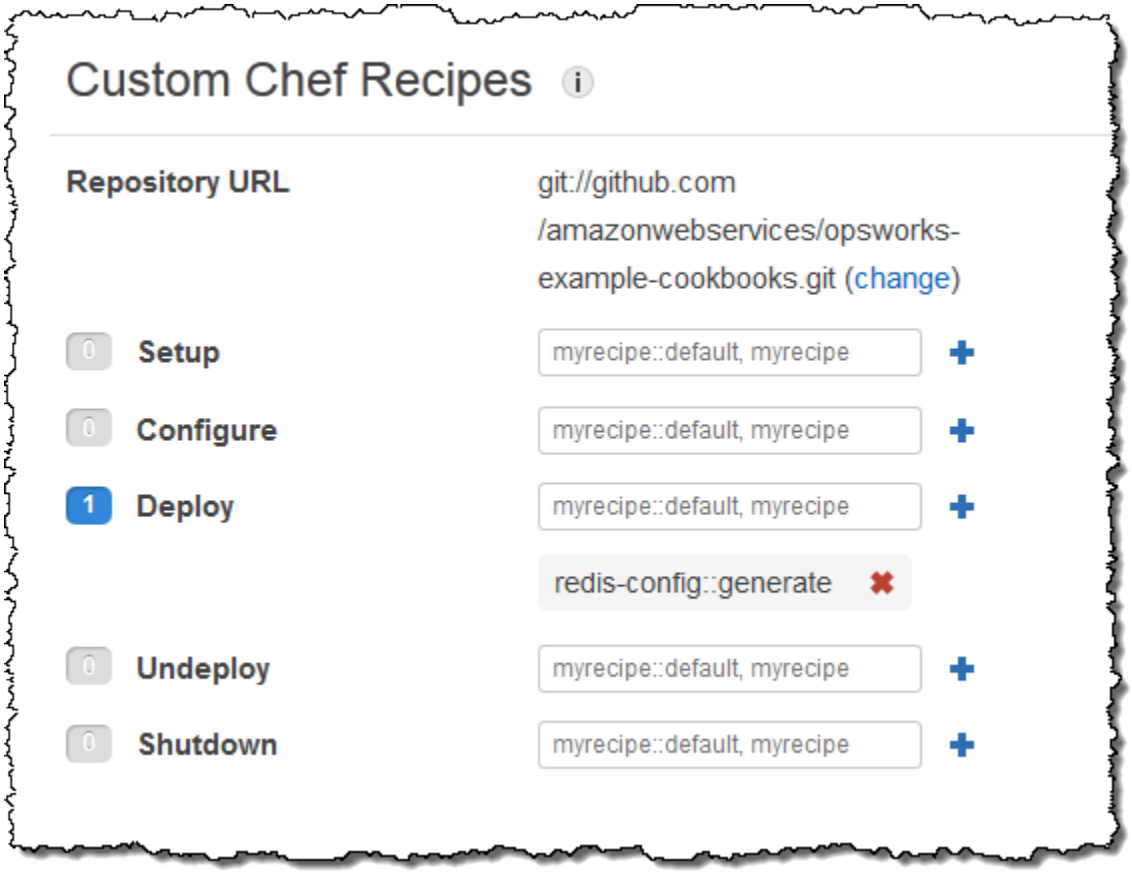
> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以手動執行自訂配方，但最佳方法是讓 Stacks AWS OpsWorks 自動執行。每個 layer 都有一組內建配方，這些配方會指派五個生命週期事件：設定、設定、部署、取消部署和關機。每當執行個體發生事件時， AWS OpsWorks Stacks 就會針對每個執行個體的 layer 執行相關聯配方，以處理對應的任務。例如，當執行個體完成開機時， AWS OpsWorks Stacks 會觸發設定事件。此事件會執行關聯 layer 的安裝配方，通常會處理像是安裝和設定套件等任務。

您可以將配方指派給適當的生命週期事件，讓 AWS OpsWorks Stacks 在 layer 的執行個體上執行自訂配方。在此範例中，您應該將`generate.rb`配方指派給 Rails App Server layer 的部署事件。 AWS OpsWorks 堆疊接著會在啟動期間、安裝配方完成後，以及每次部署應用程式時，在 layer 的執行個體上執行。如需詳細資訊，請參閱自動執行配方。

將配方指派給 Rails App Server layer 的部署事件

1. 在 AWS OpsWorks Stacks Layers 頁面上，針對 Rails App Server，按一下配方，然後按一下編輯。
2. 在 Custom Chef Recipes (自訂 Chef 配方) 下方，將完整配方名稱新增至部署事件，然後按一下 +。完整的配方名稱會使用 *`cookbookname`*`::`*`recipename`* 格式，其中 *`recipename`* 不包含 `.rb` 副檔名。針對此範例，完整名稱為 `redis-config::generate`。然後按一下 Save (儲存)，以更新 layer 組態。



步驟 5：將存取資訊新增至堆疊組態 JSON

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

`generate.rb` 配方依存於一對代表 Redis 伺服器主機名稱和連接埠的堆疊組態及部署 JSON 屬性。雖然這些屬性是標準`[:deploy]`命名空間的一部分，但 Stacks AWS OpsWorks 不會自動定義這些屬性。相反的，您會透過將自訂 JSON 物件新增至堆疊，來定義屬性和他們的值。以下範例顯示此範例的自訂 JSON。

將存取資訊新增至堆疊組態及部署 JSON

1. 在 AWS OpsWorks 堆疊堆疊頁面上，按一下堆疊設定，然後按一下編輯。
2. 在 Configuration Management (組態管理) 區段中，將存取資訊新增至 Custom Chef JSON (自訂 Chef JSON) 方塊。內容看起來應會類似以下範例，而您必須進行這些修改：

   - 將 `elasticache_redis_example` 取代為您應用程式的短名。
   - 將 `host`和 `port`值取代為您在 中建立的 ElastiCache Redis 伺服器執行個體值步驟 1：建立 ElastiCache Redis 叢集。

```
{
  "deploy": {
     "elasticache_redis_example": {
       "redis": {
         "host": "mycluster.XXXXXXXXX.amazonaws.com",
         "port": "6379"
       }
     }
  }
}
```

此方法的優點是，您可以隨時變更連接埠或主機值，而無需碰觸自訂技術指南。 AWS OpsWorks Stacks 會將自訂 JSON 合併到內建 JSON，並針對所有後續生命週期事件安裝在堆疊的執行個體上。應用程式接著便可以透過使用 Chef 節點語法存取屬性值，如步驟 3：建立和部署自訂技術指南中所述。下一次您部署應用程式時， AWS OpsWorks Stacks 會安裝包含新定義的堆疊組態和部署 JSON，並且 `generate.rb` 會建立帶有更新過主機和連接埠值的組態檔案。

> **Note**
>
> 由於 `[:deploy]` 會自動包含每個部署應用程式的屬性，因此 `[:deploy][elasticache_redis_example]` 已位於堆疊和組態 JSON 中。不過，`[:deploy][elasticache_redis_example]` 不包含`[:redis]`屬性，使用自訂 JSON 定義它們會指示 AWS OpsWorks Stacks 將這些屬性新增至 `[:deploy][elasticache_redis_example]`。您也可以使用自訂 JSON 覆寫現有的屬性。如需詳細資訊，請參閱覆寫屬性。

步驟 6：部署和執行應用程式

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

此範例假設您已有使用 Redis 的 Ruby on Rails 應用程式。若要存取組態檔案，您可以將 `redis` gem 新增至您的 Gemfile，並在 `config/initializers/redis.rb` 中建立 Rails 初始設定式，如下所示：

```
REDIS_CONFIG = YAML::load_file(Rails.root.join('config', 'redis.yml'))
$redis = Redis.new(:host => REDIS_CONFIG['host'], :port => REDIS_CONFIG['port'])
```

然後建立應用程式來代表您的應用程式，並將其部署到 Rails App Server layer 的執行個體，以更新應用程式程式碼並執行 `generate.rb` 來產生組態檔案。當您執行應用程式時，它會使用 ElastiCache Redis 執行個體做為記憶體內金鑰值存放區。

使用 Amazon S3 儲存貯體

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

應用程式通常會使用 Amazon Simple Storage Service (Amazon S3) 儲存貯體來存放大型項目，例如映像或其他媒體檔案。雖然 AWS OpsWorks Stacks 不提供 Amazon S3 的整合支援，但您可以輕鬆自訂堆疊，以允許應用程式使用 Amazon S3 儲存。本主題將引導您完成提供 Amazon S3 存取應用程式的基本程序，使用 Linux 堆疊搭配 PHP 應用程式伺服器做為範例。基本原則同樣適用於 Windows 堆疊。

傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

主題

- 步驟 1：建立 Amazon S3 儲存貯體
- 步驟 2：建立 PHP 應用程式伺服器堆疊
- 步驟 3：建立和部署自訂技術指南
- 步驟 4：將配方指派給生命週期事件
- 步驟 5：將存取資訊新增至堆疊組態及部署屬性
- 步驟 6：部署和執行 PhotoApp

步驟 1：建立 Amazon S3 儲存貯體

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您必須先建立 Amazon S3 儲存貯體。您可以使用 Amazon S3 主控台、API 或 CLI 直接執行此操作，但建立資源的簡單方式通常是使用 AWS CloudFormation 範本。下列範本會為此範例建立 Amazon S3 儲存貯體，並使用 IAM 角色設定執行個體描述檔，以授予儲存貯體不受限制的存取權。然後，您可以使用 layer 設定，將執行個體描述檔連接到堆疊的應用程式伺服器執行個體，讓應用程式存取儲存貯體，如稍後所述。執行個體描述檔的實用性不限於 Amazon S3；它們對於整合各種 AWS 服務非常有用。

```
{
   "AWSTemplateFormatVersion" : "2010-09-09",
   "Resources" : {
      "AppServerRootRole": {
         "Type": "AWS::IAM::Role",
         "Properties": {
            "AssumeRolePolicyDocument": {
               "Statement": [ {
                  "Effect": "Allow",
                  "Principal": {
                     "Service": [ "ec2.amazonaws.com" ]
                  },
                  "Action": [ "sts:AssumeRole" ]
               } ]
            },
            "Path": "/"
         }
      },
      "AppServerRolePolicies": {
         "Type": "AWS::IAM::Policy",
         "Properties": {
            "PolicyName": "AppServerS3Perms",
            "PolicyDocument": {
               "Statement": [ {
```

```
                  "Effect": "Allow",
                  "Action": "s3:*",
                  "Resource": { "Fn::Join" : ["", [ "arn:aws:s3:::", { "Ref" :
 "AppBucket" } , "/*" ]
                  ] }
               } ]
            },
            "Roles": [ { "Ref": "AppServerRootRole" } ]
         }
      },
      "AppServerInstanceProfile": {
         "Type": "AWS::IAM::InstanceProfile",
         "Properties": {
            "Path": "/",
            "Roles": [ { "Ref": "AppServerRootRole" } ]
         }
      },
     "AppBucket" : {
      "Type" : "AWS::S3::Bucket"
      }
   },
   "Outputs" : {
       "BucketName" : {
           "Value" : { "Ref" : "AppBucket" }
       },
       "InstanceProfileName" : {
           "Value" : { "Ref" : "AppServerInstanceProfile" }
       }
   }
}
```

當您啟動範本時會發生幾種情況：

- AWS::S3::Bucket 資源會建立 Amazon S3 儲存貯體。
- AWS::IAM::InstanceProfile 資源會建立執行個體描述檔，以指派給應用程式伺服器執行個體。
- AWS::IAM::Role 資源會建立執行個體描述檔的角色。
- AWS::IAM::Policy 資源會設定角色的許可，以允許不受限制地存取 Amazon S3 儲存貯體。
- 在您啟動範本之後， AWS CloudFormation 主控台中的 Outputs 區段會顯示儲存貯體和執行個體描述檔名稱。

您需要這些值來設定堆疊和應用程式。

如需如何建立 AWS CloudFormation 範本的詳細資訊，請參閱了解範本基本概念。

建立 Amazon S3 儲存貯體

1. 將範例範本複製到您系統中的文字檔案。

   此範例假設檔案名稱為 `appserver.template`。
2. 開啟 AWS CloudFormation 主控台，然後選擇 Create Stack (建立堆疊)。
3. 在 Stack Name (堆疊名稱) 方塊中，輸入堆疊名稱。

   此範例假設名稱為 **`AppServer`**。
4. 依序選擇 Upload template file (上傳範本檔案) 和 Browse (瀏覽)，並選取您在步驟 1 中建立的 `appserver.template` 檔案，然後選擇 Next Step (下一步)。
5. 在 Specify Parameters (指定參數) 頁面中，選取 I acknowledge that this template may create IAM resources (我知道此範本可能會建立 IAM 資源)，然後選擇每個精靈頁面的 Next Step (下一步)，直到完成為止。選擇建立。
6. 在 AppServer 堆疊達到 CREATE_COMPLETE 狀態之後，請選取它，然後選擇 Outputs (輸出) 標籤。

   您可能需要重新整理幾次以更新狀態。
7. 在 Outputs (輸出) 標籤中，記錄 BucketName 和 InstanceProfileName 值，供日後使用。

> **Note**
>
> AWS CloudFormation 使用 堆疊一詞來參考從範本建立的資源集合；它與 AWS OpsWorks Stacks 堆疊不同。

步驟 2：建立 PHP 應用程式伺服器堆疊

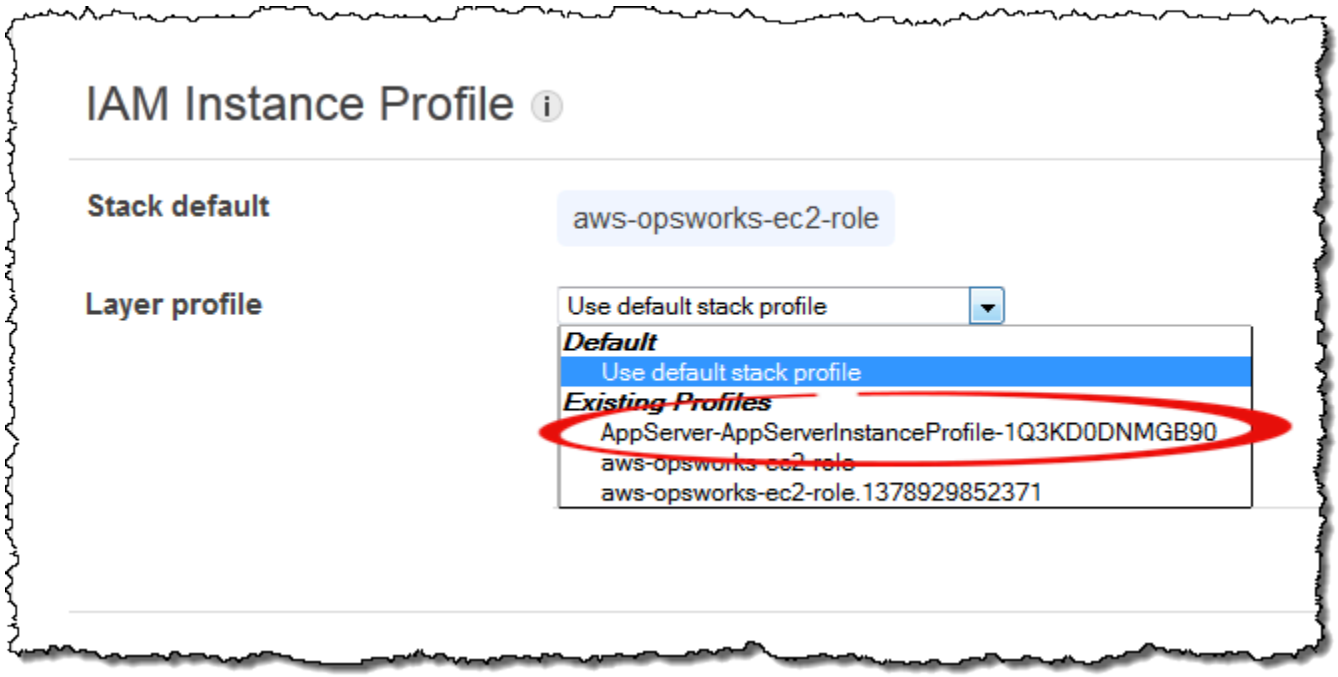
> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

堆疊包含兩個層：PHP App Server 和 MySQL，每個層都有一個執行個體。應用程式會將相片存放在 Amazon S3 儲存貯體上，但會使用 MySQL 執行個體做為後端資料存放區，以保留每張相片的中繼資料。

傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

建立堆疊

1. 建立為此**PhotoSite**範例命名的新堆疊，並新增 PHP App Server layer。您可以使用兩者的預設設定。如需詳細資訊，請參閱建立新的堆疊及建立 OpsWorks Layer 。
2. 在 Layers 頁面上，針對 PHP App Server，選擇安全性，然後選擇編輯。
3. 在 Layer Profile 區段中，選取您在啟動 AppServer AWS CloudFormation 堆疊後先前記錄的執行個體設定檔名稱。它就像 `AppServer-AppServerInstanceProfile-1Q3KD0DNMGB90`. AWS OpsWorks Stacks 會將此設定檔指派給 layer 的所有 Amazon EC2 執行個體，這會授予許可，以存取在 layer 執行個體上執行的應用程式 Amazon S3 儲存貯體。



4. 將執行個體新增至 PHP App Server layer 並啟動它。如需如何新增並啟動執行個體的詳細資訊，請參閱將執行個體新增至 Layer。
5. 將 MySQL layer 新增至堆疊、新增執行個體，然後啟動它。您可以使用 layer 和執行個體的預設設定。特別是，MySQL 執行個體不需要存取 Amazon S3 儲存貯體，因此可以使用預設選取的標準 AWS OpsWorks Stacks 執行個體描述檔。

步驟 3：建立和部署自訂技術指南

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

堆疊尚未就緒：

- 您的應用程式需要一些資訊才能存取 MySQL 資料庫伺服器和 Amazon S3 儲存貯體，例如資料庫主機名稱和 Amazon S3 儲存貯體名稱。
- 您需要在 MySQL 資料庫伺服器中設定資料庫，並建立資料表來保存相片的中繼資料。

您可以手動處理這些任務，但更好的方法是實作 Chef 配方，並讓 AWS OpsWorks Stacks 在適當的執行個體上自動執行配方。Chef 配方是 Stacks AWS OpsWorks 用來在執行個體上執行任務的特殊 Ruby 應用程式，例如安裝套件或建立組態檔案。配方封裝在「技術指南」中，而技術指南可以包含多個配方和相關檔案 (例如組態檔案的範本)。技術指南會放置於儲存庫 (例如 GitHub) 中，且必須擁有標準目錄結構。如果您還沒有自訂技術指南儲存庫，請參閱技術指南儲存庫，以取得如何設定技術指南的資訊。

這個範例已為您實作技術指南，並將其存放在公有 GitHub 儲存庫中。此技術指南包含 `appsetup.rb` 和 `dbsetup.rb` 這兩個配方，以及 `db-connect.php.erb` 範本檔案。

`appsetup.rb` 配方會建立組態檔案，其中包含應用程式存取資料庫和 Amazon S3 儲存貯體所需的資訊。基本上，它是稍經修改的 `appsetup.rb` 配方 (如將應用程式連線到資料庫中所述) 版本 主要差別在於傳遞到範本的變數，其代表存取資訊。

前四個屬性會定義資料庫連線設定，並在您建立 MySQL 執行個體時由 AWS OpsWorks Stacks 自動定義。

這些變數和原始配方中的變數有兩項差異：

- 與原始配方相同，`table` 變數代表由 `dbsetup.rb` 建立的資料庫表格名稱，並設定為技術指南屬性檔案中定義的屬性值。

  不過，屬性具有不同的名稱：`[:photoapp][:dbtable]`。
- `s3bucket` 變數專屬於此範例，並設定為代表 Amazon S3 儲存貯體名稱 的 屬性值`[:photobucket]`。

  `[:photobucket]` 是使用自訂 JSON 來定義，如稍後所述。如需屬性的詳細資訊，請參閱Attributes。

如需屬性的詳細資訊，請參閱Attributes。

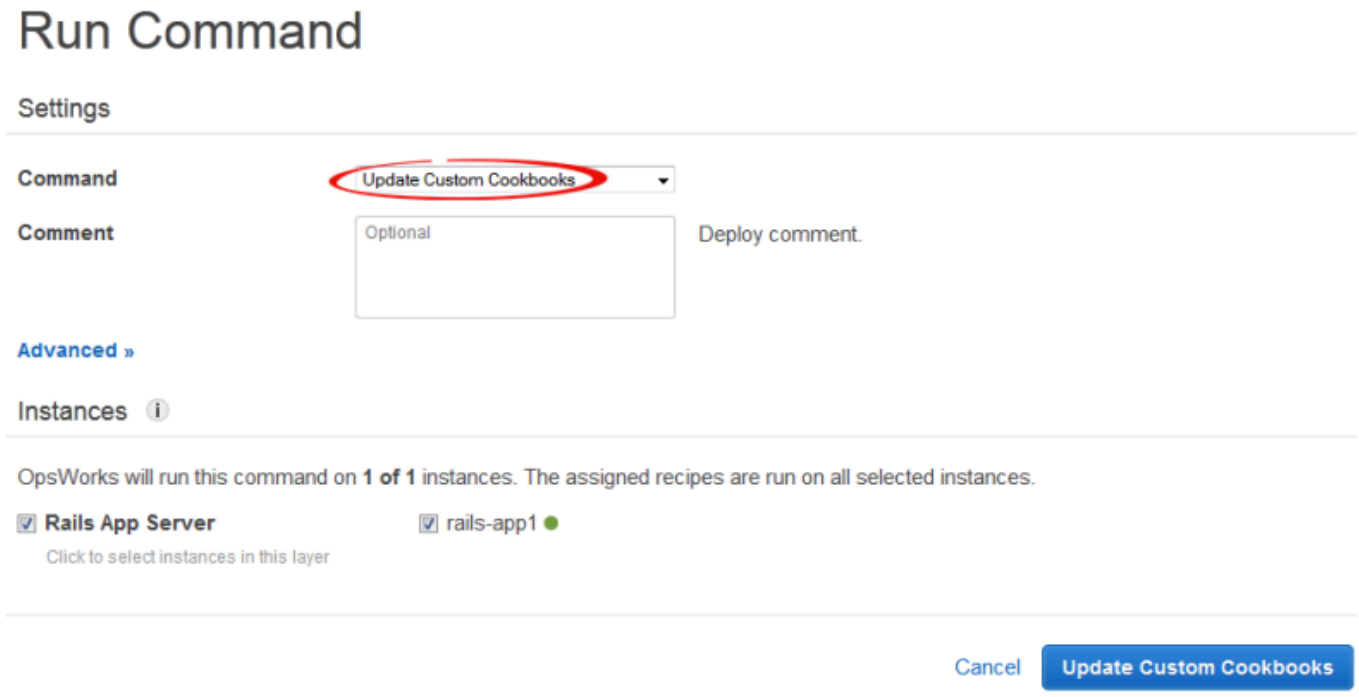
`dbsetup.rb` 配方會設定資料庫表格，以存放每個相片的中繼資料。基本上，它是稍經修改的 `dbsetup.rb` 配方 (如設定資料庫中所述) 版本；請參閱該主題以了解詳細描述。

此範例與原始配方的唯一差異是資料庫結構描述，其中包含三個資料欄，其中包含存放在 Amazon S3 儲存貯體上每張相片的 ID、URL 和字幕。

配方已實作，因此您只需要將相片應用程式技術指南部署到每個執行個體的技術指南快取。 然後，AWS OpsWorks Stacks 會在適當的生命週期事件發生時執行快取的配方，如下所述。

部署 photoapp 技術指南

1. 在 AWS OpsWorks 堆疊堆疊頁面上，選擇堆疊設定，然後選擇編輯。
2. 在 Configuration Management (組態管理) 區段中：
   - 將 Use custom Chef cookbooks (使用自訂 Chef 技術指南) 設為 Yes (是)。
   - 將 Repository type (儲存庫類型) 設定為 Git。
   - 將 Repository URL (儲存庫 URL) 設定為 **`git://github.com/amazonwebservices/opsworks-example-cookbooks.git`**。
3. 在 Stack (堆疊) 頁面上，選擇 Run Command (執行命令)，並選取 Update Custom Cookbooks (更新自訂技術指南) 堆疊命令，然後選擇 Update Custom Cookbooks (更新自訂技術指南)，以將新的技術指南安裝到執行個體的技術指南快取中。



步驟 4：將配方指派給生命週期事件

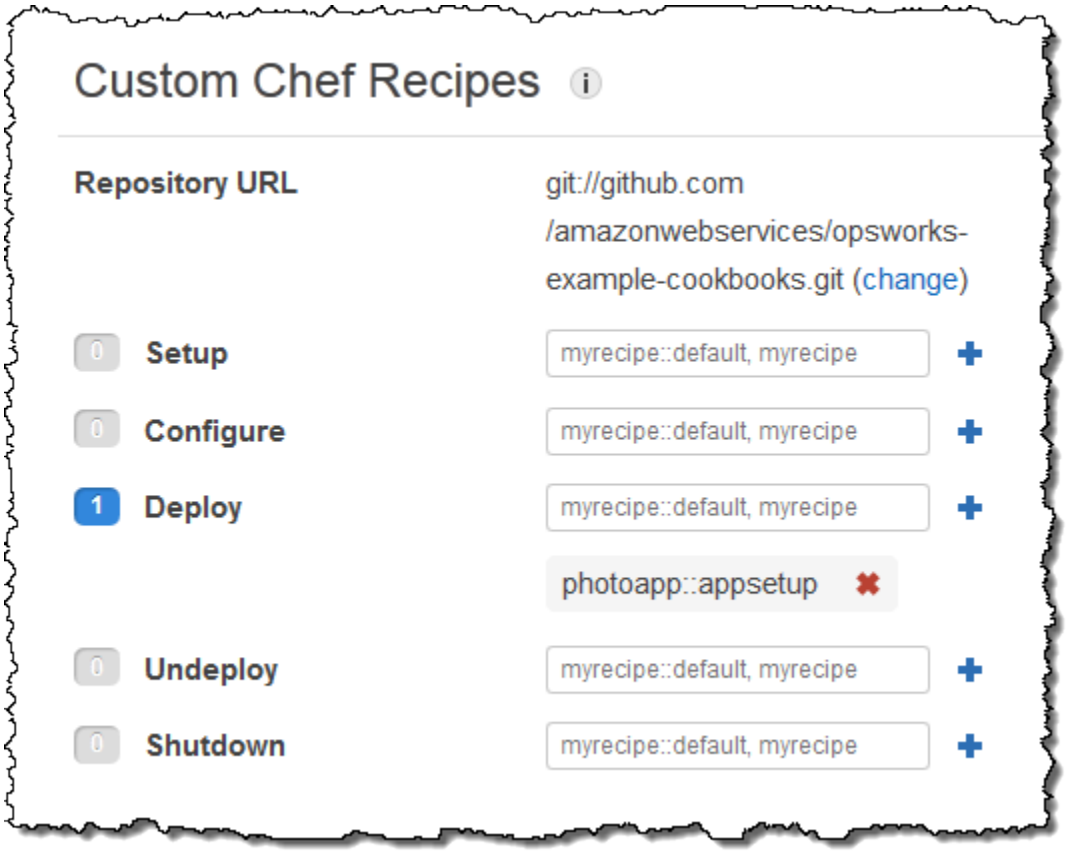
> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

您可以手動執行自訂配方，但最佳方法是讓 Stacks AWS OpsWorks 自動執行。每個 layer 都有一組內建配方，指派給五個生命週期事件的每一個：設定、設定、部署、取消部署和關機。每次在執行個體上發生事件時， AWS OpsWorks Stacks 都會為每個執行個體的 layer 執行相關聯的配方，以處理所需的任務。例如，當執行個體完成開機時， AWS OpsWorks Stacks 會觸發安裝事件來執行安裝配方，這通常會處理安裝和設定套件 等任務。

您可以將每個配方指派給適當的生命週期事件，讓 AWS OpsWorks Stacks 在 layer 的執行個體上執行自訂配方。 AWS OpsWorks Stacks 會在 layer 的內建配方完成後執行任何自訂配方。在此範例中，將 `appsetup.rb`指派給 PHP App Server layer 的部署事件`dbsetup.rb`和 MySQL layer 的部署事件。AWS OpsWorks 堆疊接著會在啟動期間、內建安裝配方完成後，以及每次您部署應用程式時，在內建部署配方完成後，在相關聯的 layer 執行個體上執行配方。如需詳細資訊，請參閱自動執行配方。

將自訂配方指派給 layer 的部署事件

1. 在 AWS OpsWorks Stacks Layers 頁面上，針對 PHP 應用程式伺服器選擇配方，然後選擇編輯。
2. 在 Custom Chef Recipes (自訂 Chef 配方) 下方，將配方名稱新增至部署事件，然後選擇 +。名稱必須為 Chef *`cookbookname`*`::`*`recipename`* 格式，其中 *`recipename`* 不含 `.rb` 副檔名。在此範例中，您可以輸入 `photoapp::appsetup`。然後選擇 Save (儲存)，以更新 layer 組態。



3. 在 Layers 頁面上，選擇 MySQL layer 動作欄中的編輯。

4. 將 photoapp::dbsetup 新增至 layer 的部署事件，並儲存新的組態。

步驟 5：將存取資訊新增至堆疊組態及部署屬性

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

appsetup.rb 配方取決於 Stacks AWS OpsWorks 堆疊組態和部署屬性的資料，這些屬性會安裝在每個執行個體上，並包含堆疊和任何部署應用程式的詳細資訊。物件的 deploy 屬性具有下列結構；為了方便，將其顯示為 JSON：

```
{
   ...
  "deploy": {
    "app1": {
      "application" : "short_name",
      ...
    }
    "app2": {
      ...
    }
    ...
  }
}
```

deploy 節點包含每個部署應用程式 (以應用程式的短名命名) 的屬性。每個應用程式屬性都包含一組定義應用程式組態的屬性，例如文件根和應用程式類型。如需 deploy 屬性的清單，請參閱deploy 屬性。您可以使用 Chef 屬性語法，來代表配方中的堆疊組態和部署屬性值。例如，[:deploy][:app1][:application] 代表 app1 應用程式的短名。

自訂配方取決於代表資料庫和 Amazon S3 存取資訊的數個堆疊組態和部署屬性：

- Stacks 會在建立 MySQL AWS OpsWorks layer 時定義資料庫連線屬性[:deploy][:database][:host]，例如 。

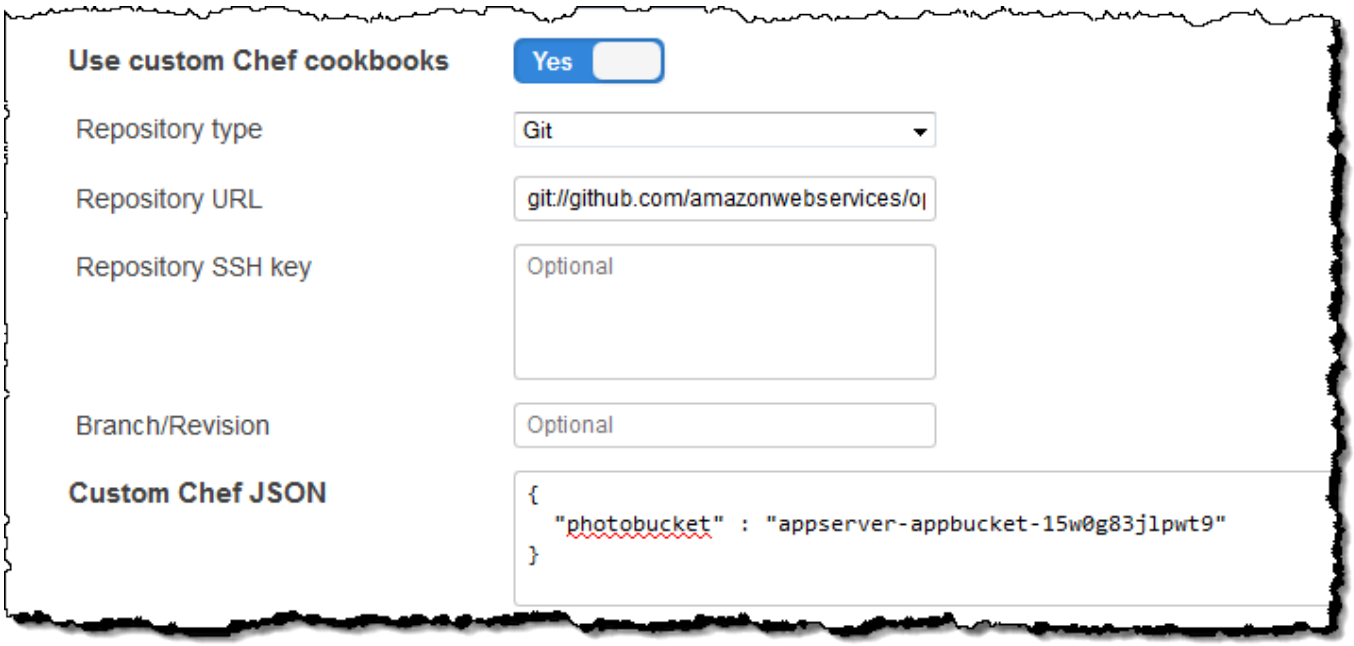
- `[:photoapp][:dbtable]` 資料表名稱屬性定義於自訂技術指南的屬性檔案中，並設為 `foto`。
- 您必須使用自訂 JSON 來定義儲存貯體名稱屬性 `[:photobucket]`，並將該屬性新增至堆疊組態和部署屬性。

定義 Amazon S3 儲存貯體名稱屬性

1. 在 AWS OpsWorks 堆疊堆疊頁面上，選擇堆疊設定，然後選擇編輯。
2. 在 Configuration Management (組態管理) 區段中，將存取資訊新增至 Custom Chef JSON (自訂 Chef JSON) 方塊。它看起來應該與下列類似：

```
{
  "photobucket" : "yourbucketname"
}
```

將 *yourbucketname* 取代為您在步驟 1：建立 Amazon S3 儲存貯體中記錄的儲存貯體名稱。



AWS OpsWorks Stacks 會將自訂 JSON 合併到堆疊組態和部署屬性，再將其安裝在堆疊的執行個體上；然後`appsetup.rb`可以從 `[:photobucket]` 屬性取得儲存貯體名稱。如果您想要變更儲存貯體，您不需要動用配方，而可以直接覆寫屬性以提供新的儲存貯體名稱。

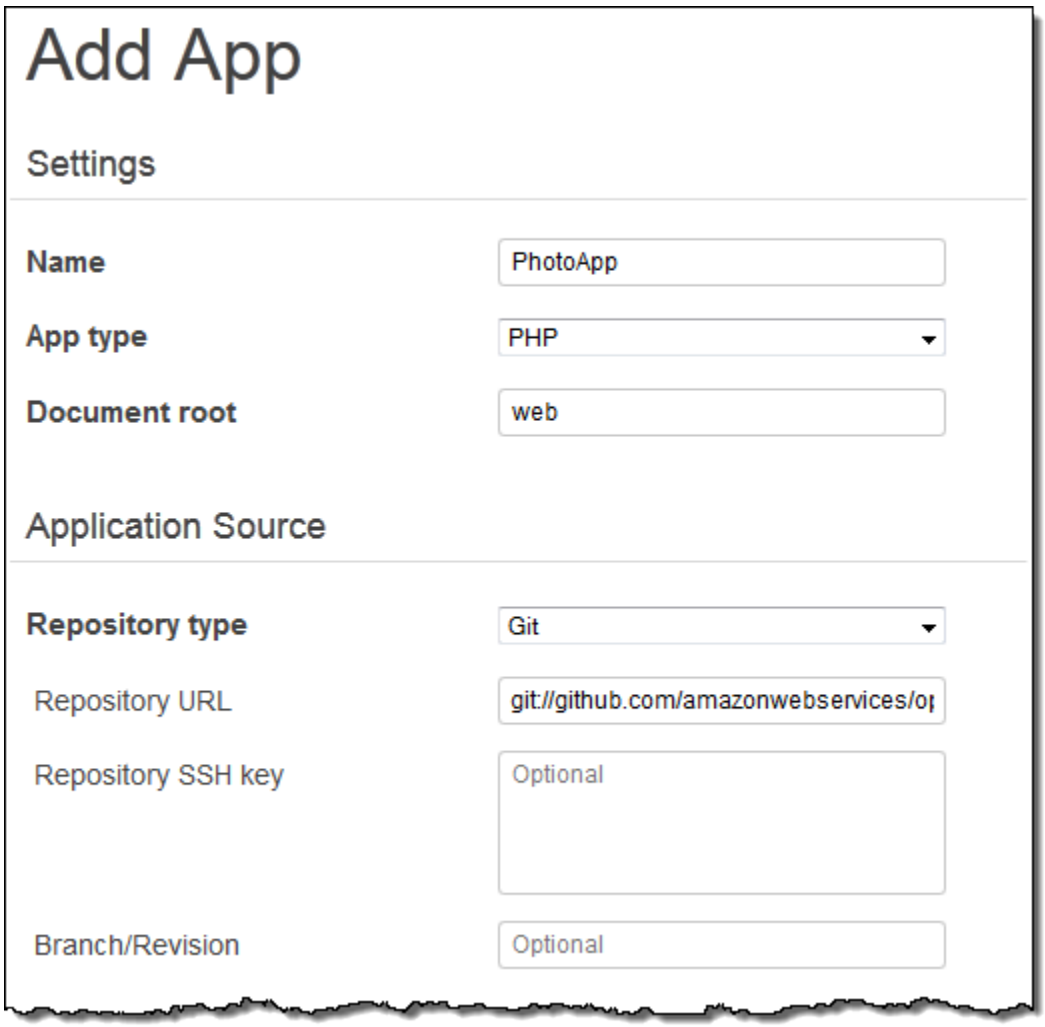
步驟 6：部署和執行 PhotoApp

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

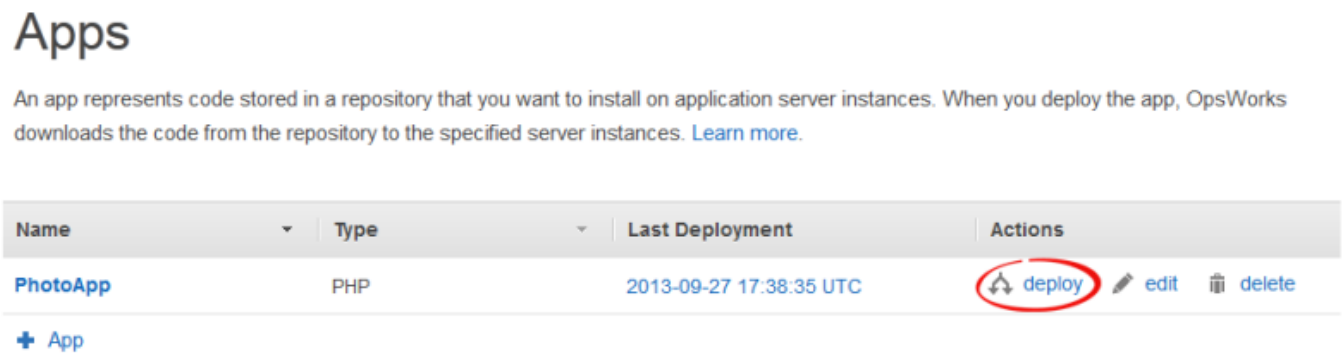
這個範例也已為您實作應用程式，並將其存放在公有 GitHub 儲存庫中。您只需要將應用程式新增至堆疊，並將其部署到應用程式伺服器，然後加以執行。

將應用程式新增至堆疊，並將其部署到應用程式伺服器

1. 開啟 Apps (應用程式) 頁面，然後選擇 Add an app (新增應用程式)。
2. 在 Add App (新增應用程式) 頁面上，執行下列作業：
   - 將 Name (名稱) 設定為 **PhotoApp**。
   - 將 App type (應用程式類型) 設定為 PHP。
   - 將 Document root (文件根) 設定為 **web**。
   - 將 Repository type (儲存庫類型) 設定為 Git。
   - 將 Repository URL (儲存庫 URL) 設定為 **git://github.com/awslabs/opsworks-demo-php-photo-share-app.git**。
   - 選擇 Add App (新增應用程式)，以接受其他設定的預設值。

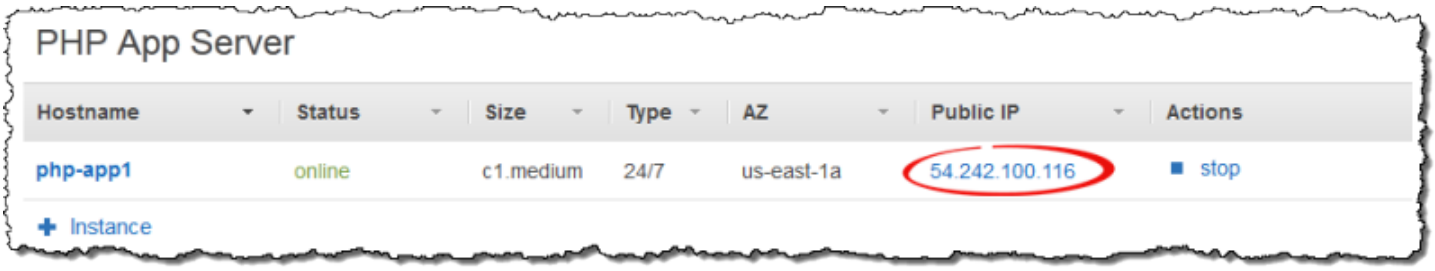


3. 在 Apps (應用程式) 頁面上，選擇 PhotoApp 應用程式 Actions (動作) 欄中的 deploy (部署)。



4. 接受預設值，然後選擇 Deploy (部署) 將應用程式部署至伺服器。

若要執行 PhotoApp，請前往執行個體頁面，然後選擇 PHP App Server 執行個體的公有 IP 地址。



您應該會看到下列使用者界面。選擇新增相片，將相片存放在 Amazon S3 儲存貯體和後端資料存放區中的中繼資料。

AWS CodePipeline 搭配 Stacks AWS OpsWorks 使用

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS CodePipeline 可讓您建立持續交付管道，以追蹤來自 CodeCommit、Amazon Simple Storage Service (Amazon S3) 或 GitHub 等來源的程式碼變更。您可以使用 CodePipeline，在 Chef 11.10、Chef 12 和 Chef 12.2 AWS OpsWorks 堆疊上，自動將 Chef 技術指南和應用程式程式碼發佈至 Stacks。本節中的範例說明如何從 CodePipeline 建立和使用簡單的管道，做為您在 Stacks AWS OpsWorks 層上執行程式碼的部署工具。

> **Note**
>
> CodePipeline 和 AWS OpsWorks Stacks 整合不支援部署至 Chef 11.4 和較舊的堆疊。

主題

- AWS CodePipeline 搭配 AWS OpsWorks Stacks - Chef 12 Stacks
- AWS CodePipeline 搭配 AWS OpsWorks Stacks - Chef 11 Stacks

AWS CodePipeline 搭配 AWS OpsWorks Stacks - Chef 12 Stacks

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS CodePipeline 可讓您建立持續交付管道，以追蹤來自 CodeCommit、Amazon Simple Storage Service (Amazon S3) 或 GitHub 等來源的程式碼變更。本主題中的範例說明如何從 CodePipeline 建立和使用簡單的管道，做為您在 AWS OpsWorks Stacks layer 上執行之程式碼的部署工具。在此範例中，您會為簡單的 Node.js 應用程式建立管道，然後指示 AWS OpsWorks Stacks 在 Chef 12 堆疊內層中的所有執行個體上執行應用程式（在此案例中為單一執行個體）。

> **Note**
>
> 本主題說明如何使用管道在 Chef 12 堆疊上執行與更新應用程式。如需如何使用管道在 Chef 11.10 堆疊上執行與更新應用程式的資訊，請參閱 AWS CodePipeline 搭配 AWS OpsWorks Stacks - Chef 11 Stacks。傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

主題

- 先決條件
- 其他支援的案例
- 步驟 1：在 Stacks 中建立堆疊、 layer AWS OpsWorks 和執行個體
- 步驟 2：設定您的堆疊和 layer 使用自訂的技術指南
- 步驟 3：將應用程式碼上傳至 Amazon S3 儲存貯體
- 步驟 4：將您的應用程式新增至 AWS OpsWorks Stacks
- 步驟 5：在 CodePipeline 中建立管道
- 步驟 6：驗證 AWS OpsWorks Stacks 中的應用程式部署

- 步驟 7 （選用）：更新應用程式程式碼，以查看 CodePipeline 自動重新部署您的應用程式
- 步驟 8 (選用)：清理資源

先決條件

在您開始本演練之前，請確定您有執行下列所有任務的管理員許可。您可以是已套用 AdministratorAccess 政策之群組的成員，也可以是具有下表所示許可和政策之群組的成員。作為安全最佳實務，您應該屬於具有執行下列任務許可的群組，而不是將必要的許可指派給個別使用者。

如需在 IAM 中建立安全群組和指派許可給群組的詳細資訊，請參閱建立 IAM 使用者群組。如需管理 AWS OpsWorks Stacks 許可的詳細資訊，請參閱最佳實務：管理許可。

許可	建議連接至群組的政策
在 Stacks AWS OpsWorks 中建立和編輯堆疊、層和執行個體。	AWSOpsWorks_FullAccess
在 AWS CloudFormation中建立、編輯和執行範本。	AmazonCloudFormationFullAccess
建立、編輯和存取 Amazon S3 儲存貯體。	AmazonS3FullAccess
在 CodePipeline 中建立、編輯和執行管道，尤其是使用 Stacks AWS OpsWorks 做為提供者的管道。	AWSCodePipeline_FullAccess

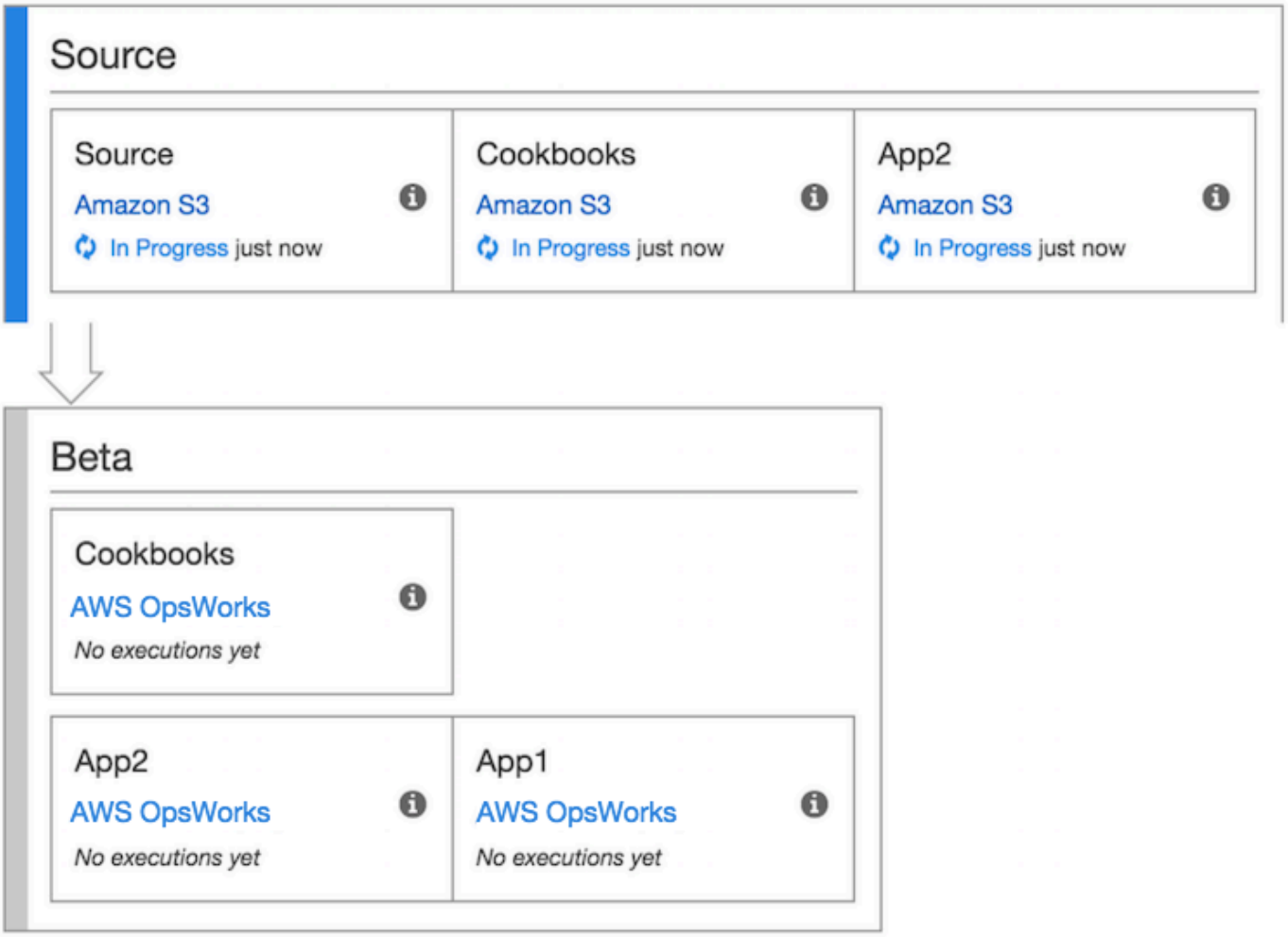
您也必須有 Amazon EC2 金鑰對。當您執行在此演練中建立範例堆疊、 layer 和執行個體的 AWS CloudFormation 範本時，系統會提示您提供此金鑰對的名稱。如需在 Amazon EC2 主控台中取得金鑰對的詳細資訊，請參閱 Amazon EC2 文件中的建立金鑰對。金鑰對必須位於美國東部 （維吉尼亞北部） 區域。如果您在該區域中已有金鑰對，則可以使用該現有金鑰對。

其他支援的案例

本演練會建立包含一個 Source (來源) 和一個 Deploy (部署) 階段的簡易管道。不過，您可以建立使用 Stacks AWS OpsWorks 做為提供者的更複雜管道。下列是支援的管道和案例範例：

- 您可以編輯管道，以將 Chef 技術指南新增至 Source (來源) 階段，以及將已更新技術指南的相關聯目標新增至 Deploy (部署) 階段。在此情況下，您可以新增 Deploy (部署) 動作，以在您變更來源時觸發技術指南更新。已更新的技術指南會比應用程式先部署。

- 您可以使用自訂技術指南和多個應用程式建立複雜的管道，並部署到 Stacks AWS OpsWorks 堆疊。該管道會同時追蹤對應用程式和技術指南來源所做的變更，並在您變更後重新部署。下圖示範類似的複雜管道範例：



如需使用 CodePipeline 的詳細資訊，請參閱 CodePipeline 使用者指南。

步驟 1：在 Stacks 中建立堆疊、 layer AWS OpsWorks 和執行個體

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

若要使用 AWS OpsWorks Stacks 做為管道的部署提供者，您必須先在 layer 中擁有堆疊、 layer 和至少一個執行個體。雖然您可以遵循 Linux AWS OpsWorks Stacks 入門或 Windows Stacks 入門中的指示在 Stacks 中建立堆疊，但為了節省您的時間，此範例會使用 AWS CloudFormation 範本來建立 Linux Chef 12 堆疊、 layer 和執行個體。此範本建立的執行個體會執行 Amazon Linux 2016.03，且執行個體類型為 `c3.large`。雖然範本不會設定您的堆疊使用自訂技術指南，但您稍後仍會在演練中執行此作業。

> **Important**
>
> AWS CloudFormation 範本必須與稍後上傳應用程式的 Amazon S3 儲存貯體位於相同的區域，以及稍後在 CodePipeline 中建立管道的相同區域。目前，CodePipeline 僅支援美國東部（維吉尼亞北部）區域 (us-east-1) 的 AWS OpsWorks Stacks 供應商。此演練中的所有資源都應在美國東部（維吉尼亞北部）區域建立。
>
> 如果堆疊建立失敗，您可能即將達到您帳戶的 IAM 角色允許數目上限。如果您的帳戶無法啟動執行個體類型為 `c3.large` 的執行個體，堆疊建立也可能失敗。例如，如果您使用 AWS 免費方案，您可能會收到錯誤，例如 `Root device type: must be included in EBS`。如果您的帳戶對您可以建立的執行個體類型有限制，例如 AWS 免費方案施加的限制，請嘗試將範本執行個體區塊中的 `InstanceType` 參數值變更為您的帳戶可以使用的執行個體類型。

使用 建立堆疊、 layer 和執行個體 AWS CloudFormation

1. 將下列 AWS CloudFormation 範本複製到新的純文字文件。將檔案儲存到本機電腦中方便使用的位置，並將其命名為 NewOpsWorksStack.template 或您方便使用的其他名稱。

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Mappings": {
    "Region2Principal": {
      "us-east-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "us-west-2": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "us-west-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
```

```
      },
      "eu-west-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-southeast-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-northeast-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-northeast-2": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-southeast-2": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "sa-east-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "cn-north-1": {
        "EC2Principal": "ec2.amazonaws.com.cn",
        "OpsWorksPrincipal": "opsworks.amazonaws.com.cn"
      },
      "eu-central-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      }
    }
  },
  "Parameters": {
    "EC2KeyPairName": {
   "Type": "String",
   "Description": "The name of an existing EC2 key pair that lets you use SSH to
connect to the OpsWorks instance."
  }
  },
  "Resources": {
"CPOpsDeploySecGroup": {
```

```
  "Type": "AWS::EC2::SecurityGroup",
  "Properties": {
    "GroupDescription" : "Lets you manage OpsWorks instances to which you deploy
apps with CodePipeline"
  }
},
"CPOpsDeploySecGroupIngressHTTP": {
  "Type": "AWS::EC2::SecurityGroupIngress",
  "Properties" : {
    "IpProtocol" : "tcp",
       "FromPort" : "80",
       "ToPort" : "80",
       "CidrIp" : "0.0.0.0/0",
 "GroupId": {
   "Fn::GetAtt": [
     "CPOpsDeploySecGroup", "GroupId"
   ]
 }
     }
},
"CPOpsDeploySecGroupIngressSSH": {
  "Type": "AWS::EC2::SecurityGroupIngress",
  "Properties" : {
    "IpProtocol" : "tcp",
       "FromPort" : "22",
       "ToPort" : "22",
       "CidrIp" : "0.0.0.0/0",
 "GroupId": {
   "Fn::GetAtt": [
     "CPOpsDeploySecGroup", "GroupId"
   ]
 }
  }
},
"MyStack": {
     "Type": "AWS::OpsWorks::Stack",
     "Properties": {
       "Name": {
         "Ref": "AWS::StackName"
       },
       "ServiceRoleArn": {
         "Fn::GetAtt": [
           "OpsWorksServiceRole",
           "Arn"
```

```
        ]
      },
"ConfigurationManager" : { "Name": "Chef","Version": "12" },
"DefaultOs": "Amazon Linux 2016.03",
      "DefaultInstanceProfileArn": {
        "Fn::GetAtt": [
          "OpsWorksInstanceProfile",
          "Arn"
        ]
      },
"UseCustomCookbooks": "false"
    }
  },
  "MyLayer": {
    "Type": "AWS::OpsWorks::Layer",
    "Properties": {
      "StackId": {
        "Ref": "MyStack"
      },
      "Name": "Node.js App Server",
"Type": "custom",
      "Shortname": "app1",
"EnableAutoHealing": "true",
      "AutoAssignElasticIps": "false",
      "AutoAssignPublicIps": "true",
"CustomSecurityGroupIds": [
  {
    "Fn::GetAtt": [
            "CPOpsDeploySecGroup", "GroupId"
    ]
  }
 ]
    },
    "DependsOn": [
      "MyStack",
      "CPOpsDeploySecGroup"
    ]
  },
  "OpsWorksServiceRole": {
    "Type": "AWS::IAM::Role",
    "Properties": {
      "AssumeRolePolicyDocument": {
        "Statement": [
          {
```

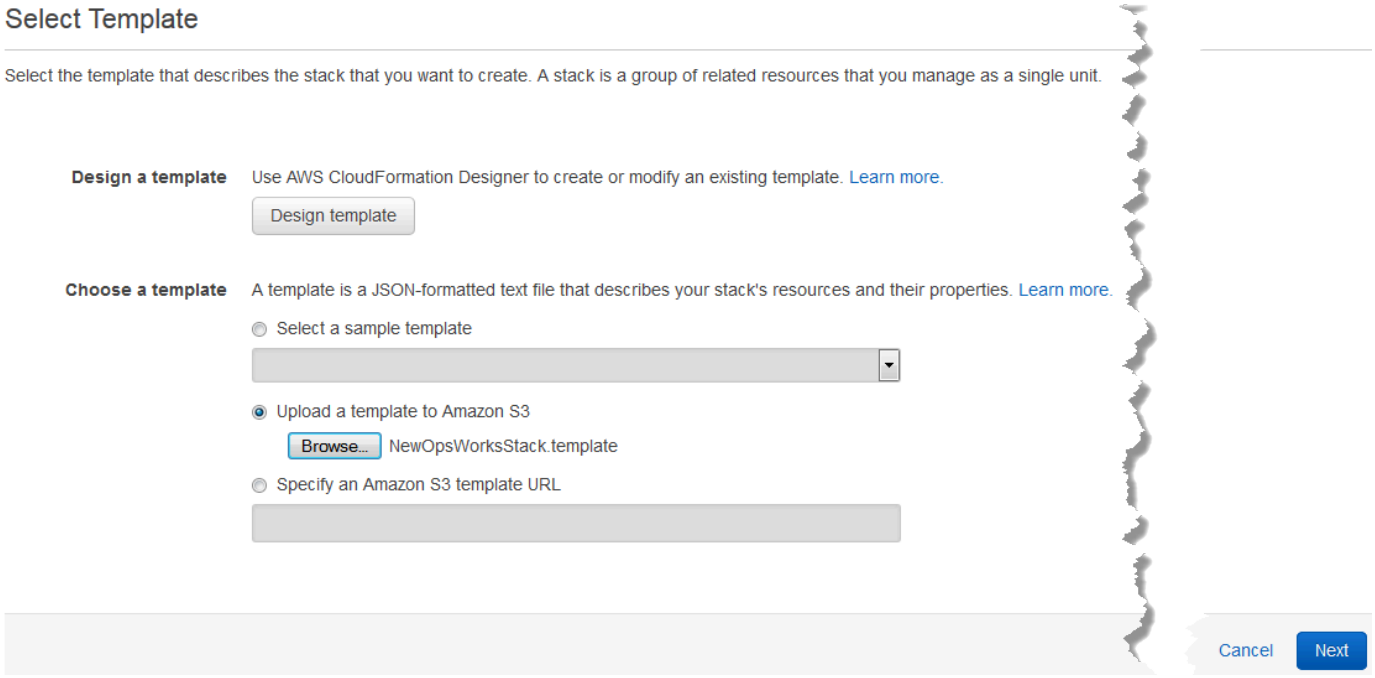
```
            "Effect": "Allow",
            "Principal": {
              "Service": [
                {
                  "Fn::FindInMap": [
                    "Region2Principal",
                    {
                      "Ref": "AWS::Region"
                    },
                    "OpsWorksPrincipal"
                  ]
                }
              ]
            },
            "Action": [
              "sts:AssumeRole"
            ]
          }
        ]
      },
      "Path": "/",
      "Policies": [
        {
          "PolicyName": "opsworks-service",
          "PolicyDocument": {
            "Statement": [
              {
                "Effect": "Allow",
                "Action": [
                  "ec2:*",
                  "iam:PassRole",
                  "cloudwatch:GetMetricStatistics",
                  "elasticloadbalancing:*"
                ],
                "Resource": "*"
              }
            ]
          }
        }
      ]
    }
  },
  "OpsWorksInstanceProfile": {
    "Type": "AWS::IAM::InstanceProfile",
```

```
      "Properties": {
        "Path": "/",
        "Roles": [
          {
            "Ref": "OpsWorksInstanceRole"
          }
        ]
      }
    },
    "OpsWorksInstanceRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  {
                    "Fn::FindInMap": [
                      "Region2Principal",
                      {
                        "Ref": "AWS::Region"
                      },
                      "EC2Principal"
                    ]
                  }
                ]
              },
              "Action": [
                "sts:AssumeRole"
              ]
            }
          ]
        },
        "Path": "/",
  "Policies": [
          {
            "PolicyName": "s3-get",
            "PolicyDocument": {
              "Version": "2012-10-17",
              "Statement": [
                {
                  "Effect": "Allow",
```

```
                  "Action": [
                    "s3:GetObject"
                  ],
                  "Resource": "*"
                }
              ]
            }
          }
        ]
      }
    },
    "myinstance": {
      "Type": "AWS::OpsWorks::Instance",
      "Properties": {
        "LayerIds": [
          {
            "Ref": "MyLayer"
          }
        ],
        "StackId": {
          "Ref": "MyStack"
        },
        "InstanceType": "c3.large",
        "SshKeyName": {
    "Ref": "EC2KeyPairName"
  }
      }
    }
  },
  "Outputs": {
    "StackId": {
      "Description": "Stack ID for the newly created AWS OpsWorks stack",
      "Value": {
        "Ref": "MyStack"
      }
    }
  }
}
```

2. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/cloudformation 開啟 AWS CloudFormation 主控台。
3. 在 AWS CloudFormation 首頁上，選擇建立堆疊。

4. 在 Select Template (選取範本) 頁面上的 Choose a template (選擇範本) 區域中，選擇 Upload a template to Amazon S3 (將範本上傳至 Amazon S3)，然後選擇 Browse (瀏覽)。
5. 瀏覽至您在步驟 1 中儲存的 AWS CloudFormation 範本，然後選擇開啟。在 Select Template (選取範本) 頁面上，請選擇 Next (下一步)。

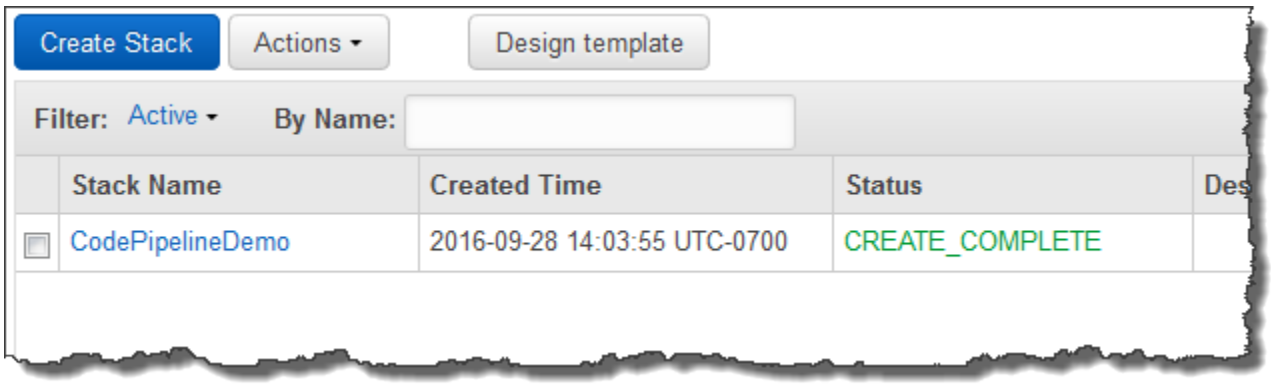


6. 在 Specify Details (指定詳細資訊) 頁面中，將堆疊命名為 CodePipelineDemo 或您帳戶唯一的任何堆疊名稱。如果您選擇不同的堆疊名稱，請變更整個本演練中該堆疊的名稱。
7. 在參數區域中，提供您要用來在建立 Stacks 執行個體之後存取的 EC2 AWS OpsWorks 金鑰對名稱。選擇下一步。
8. 在選項頁面上，選擇下一步。(本演練不需要執行此頁面上的設定。)
9. 您在此演練中使用的 AWS CloudFormation 範本會建立 IAM 角色、執行個體描述檔和執行個體。

> **Important**
>
> 在選擇建立之前，請選擇成本來預估使用此範本 AWS 建立資源時可能衍生的費用。

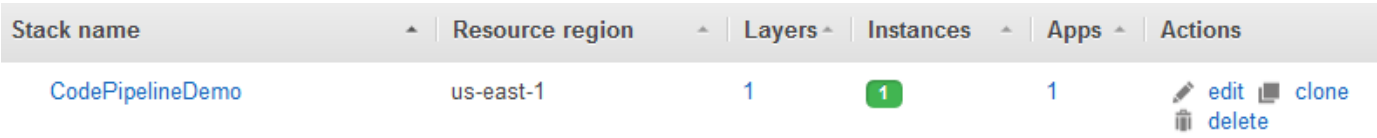
如果可以建立 IAM 資源，請選取我確認此範本可能會導致 AWS CloudFormation 建立 IAM 資源核取方塊，然後選擇建立。如果無法建立 IAM 資源，則無法繼續此程序。

10. 在 AWS CloudFormation 儀表板上，您可以檢視建立堆疊的進度。請先等候 Status (狀態) 欄位顯示 CREATE_COMPLETE 後，再繼續進行下一步。



驗證在 Stacks AWS OpsWorks 中建立堆疊

1. 在 https：//https://console.aws.amazon.com/opsworks/ 開啟 AWS OpsWorks 主控台。
2. 在 AWS OpsWorks Stacks 儀表板上，檢視您建立的堆疊。



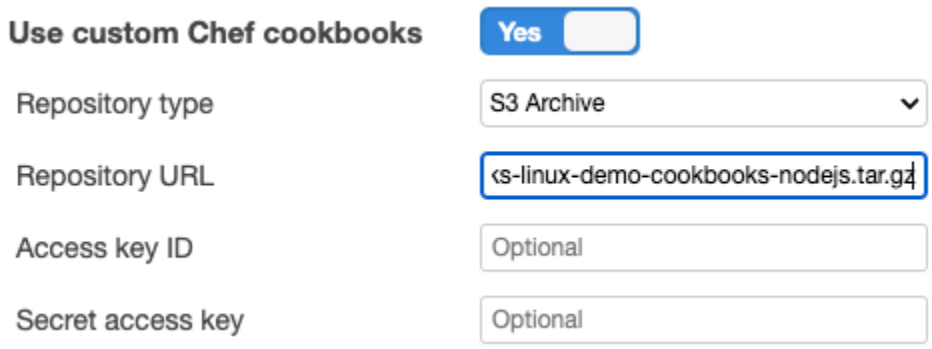
3. 開啟堆疊並檢視 layer 和執行個體。請注意，已使用 AWS CloudFormation 範本中提供的名稱和其他中繼資料建立 layer 和執行個體。您已準備好設定您的堆疊和 layer 使用自訂的 Chef 技術指南和配方。

步驟 2：設定您的堆疊和 layer 使用自訂的技術指南

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Stacks 中的 Chef 12 AWS OpsWorks 堆疊需要您自己或社群建立的技術指南來建置自訂應用程式層。在本演練中，您可以指向包含一組 Chef 技術指南和 Chef 配方的儲存庫。這些配方會在您的執行個體上安裝 Node.js 套件及其相依性。您會使用其他 Chef 配方來部署您即將在步驟 4：將您的應用程式新增至 AWS OpsWorks Stacks中準備的 Node.js 應用程式。您在此步驟中指定的 Chef 配方會在 CodePipeline 部署應用程式的新版本時執行。

1. 在 AWS OpsWorks Stacks 主控台中，開啟您在 中建立的堆疊步驟 1：在 Stacks 中建立堆疊、layer AWS OpsWorks 和執行個體。選擇 Stack Settings (堆疊設定)，然後選擇 Edit (編輯)。
2. 將 Use custom Chef cookbooks (使用自訂 Chef 技術指南) 設為 Yes (是)。這會顯示相關的自訂技術指南設定。
3. 從 Repository type (儲存庫類型) 下拉式清單，選擇 S3 Archive (S3 封存)。若要同時使用 CodePipeline 和 AWS OpsWorks，您的技術指南來源必須是 S3。
4. 針對 Repository URL (儲存庫 URL)，指定 **`https://s3.amazonaws.com/opsworks-demo-assets/opsworks-linux-demo-cookbooks-nodejs.tar.gz`**。您的設定應該類似下列：



5. 選擇儲存。
6. 在導覽視窗中，選擇 圖層。
7. 選擇您在中設定之 layer 的 Settings (設定)步驟 1：在 Stacks 中建立堆疊、 layer AWS OpsWorks 和執行個體。
8. 在 General Settings (一般設定) 標籤中，確定 layer 名稱為 Node.js App Server，layer 簡稱為 app1。選擇 Recipes (配方)。
9. 在 Recipes (配方) 標籤中，將 **`nodejs_demo`** 指定為您要在 Deploy (部署) 生命週期事件中執行的配方。選擇儲存。
10. 在 Security (安全性) 標籤中，從 Security groups (安全群組) 下拉式清單，選擇 AWS-OpsWorks-Webapp 安全群組。
11. 選擇儲存。

步驟 3：將應用程式碼上傳至 Amazon S3 儲存貯體

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

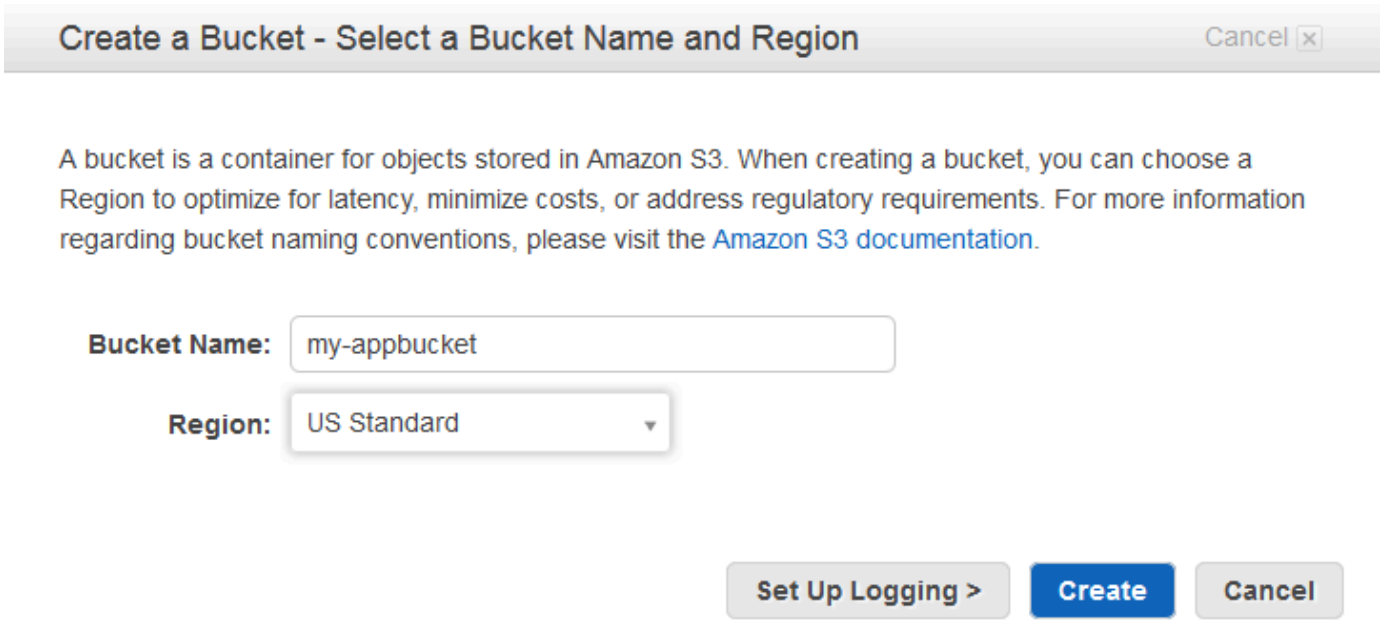
由於您必須提供程式碼儲存庫的連結做為管道設定的一部分，因此在建立管道前，請先準備好程式碼儲存庫。在此逐步解說中，您將 Node.js 應用程式上傳至 Amazon S3 儲存貯體。

雖然 CodePipeline 可以直接從 GitHub 或 CodeCommit 使用程式碼做為來源，但本演練示範如何使用 Amazon S3 儲存貯體。在此演練中，您將範例 Node.js 應用程式上傳到您自己的 Amazon S3 儲存貯體，以便您可以變更應用程式。您在此步驟中建立的 Amazon S3 儲存貯體可讓 CodePipeline 偵測應用程式程式碼的變更，並自動部署變更的應用程式。您也可以視需要使用現有的儲存貯體。請確定儲存貯體符合 CodePipeline 文件中簡易管道演練 (Amazon S3 儲存貯體） 所述的條件。

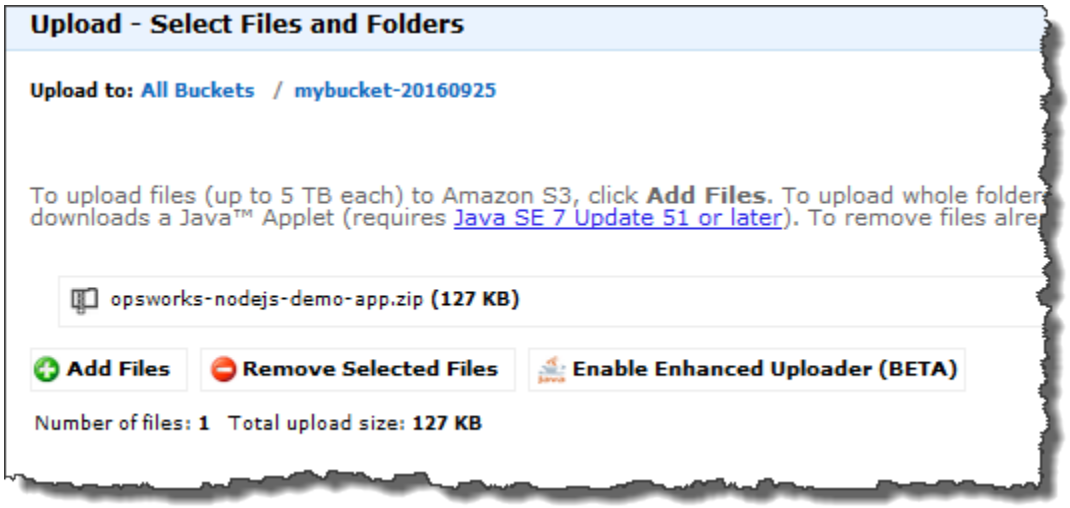
> **Important**
>
> Amazon S3 儲存貯體必須位於您稍後將建立管道的相同區域。目前，CodePipeline 僅支援美國東部（維吉尼亞北部） 區域 (us-east-1) 的 AWS OpsWorks Stacks 供應商。此演練中的所有資源都應在美國東部（維吉尼亞北部） 區域建立。儲存貯體也必須進行版本控制，因為 CodePipeline 需要版本控制來源。如需詳細資訊，請參閱使用版本控制。

將應用程式上傳至 Amazon S3 儲存貯體

1. 下載 AWS OpsWorks Stacks 範例 Node.js 應用程式的 ZIP 檔案，並將其儲存至本機電腦上方便的位置。
2. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
3. 選擇 Create Bucket (建立儲存貯體)。
4. 在 Create a Bucket - Select a Bucket Name and Region (建立儲存貯體 - 選取儲存貯體名稱和區域) 頁面上，針對 Bucket Name (儲存貯體名稱)，輸入儲存貯體的唯一名稱。儲存貯體名稱在所有 AWS 帳戶中必須是唯一的，而不只是在您自己的帳戶中。本演練使用的名稱是 **`my-appbucket`**，但您可使用 `my-appbucket-`*`yearmonthday`*，讓您的儲存貯體名稱成為唯一名稱。從 Region (區域) 下拉式清單，選擇 US Standard (美國標準)，然後選擇 Create (建立)。US Standard (美國標準) 相當於 `us-east-1`。



5. 從 All Buckets (所有儲存貯體) 清單選擇您建立的儲存貯體。
6. 在儲存貯體頁面上，選擇 Upload (上傳)。
7. 在 Upload - Select Files and Folders (上傳 - 選取檔案和資料夾) 頁面上，選擇 Add files (新增檔案)。瀏覽您在步驟 1 中儲存的 ZIP 檔案，選擇 Open (開啟)，再選擇 Start Upload (開始上傳)。



8. 在上傳完成後，從您儲存貯體中的檔案清單選取 ZIP 檔案，然後選擇 Properties (屬性)。
9. 在 Properties (屬性) 窗格中，複製您 ZIP 檔案的連結，並記下此連結。您將需要儲存貯體名稱和此連結的 ZIP 檔案名稱部分來建立管道。

步驟 4：將您的應用程式新增至 AWS OpsWorks Stacks

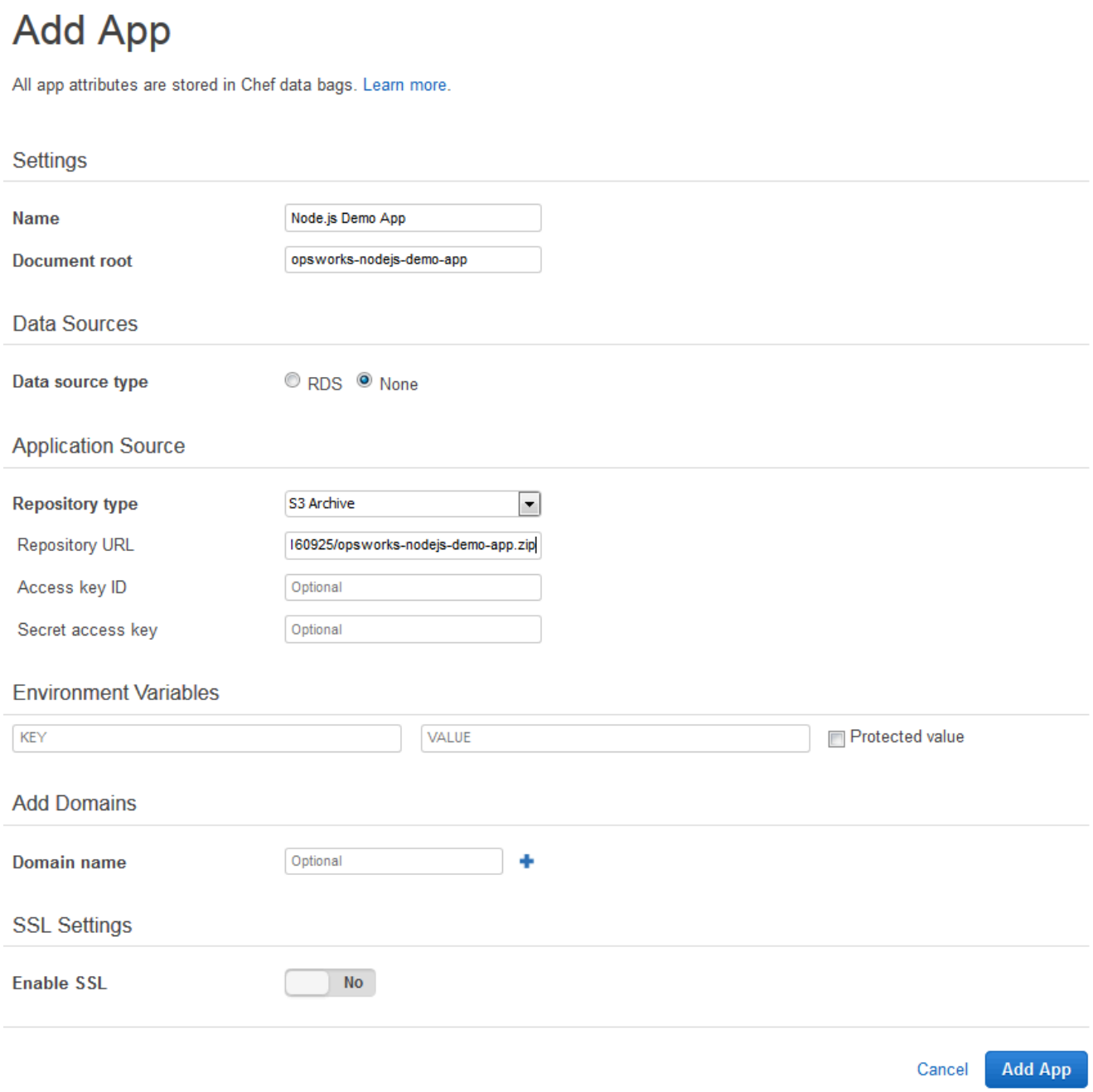
> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

在 CodePipeline 中建立管道之前，請將 Node.js 測試應用程式新增至 AWS OpsWorks Stacks。建立管道時，您需要選取已新增至 Stacks AWS OpsWorks 的應用程式。

準備好先前程序的步驟 9 中的 Amazon S3 儲存貯體連結。您將需要存放測試應用程式所在的儲存貯體連結來完成此程序。

將應用程式新增至 AWS OpsWorks Stacks

1. 在 AWS OpsWorks Stacks 主控台中，開啟 CodePipelineDemo，然後在導覽窗格中選擇應用程式。
2. 選擇 Add app (新增應用程式)。
3. 在 Add App (新增應用程式) 頁面上，提供下列資訊：
   a. 為應用程式指定名稱。本逐步教學使用的名稱是 `Node.js Demo App`。
   b. 針對 Data source type (資料來源類型)，選擇 None (無)。此應用程式不需要外部資料庫或資料來源。
   c. 在 Repository type (儲存庫類型) 下拉式清單中，選擇 S3 Archive (S3 封存)。
   d. 在 Repository URL (儲存庫 URL) 字串方格中，貼上您在步驟 3：將應用程式碼上傳至 Amazon S3 儲存貯體的步驟 9 中複製的 URL。您的表單應類似下列內容：



4. 您不需要變更此表單中的任何其他設定。選擇 Add App (新增應用程式)。
5. 當 Node.js Demo App (Node.js 示範應用程式) 應用程式出現在 Apps (應用程式) 頁面的清單中時，請繼續下一項程序 (步驟 5：在 CodePipeline 中建立管道)。

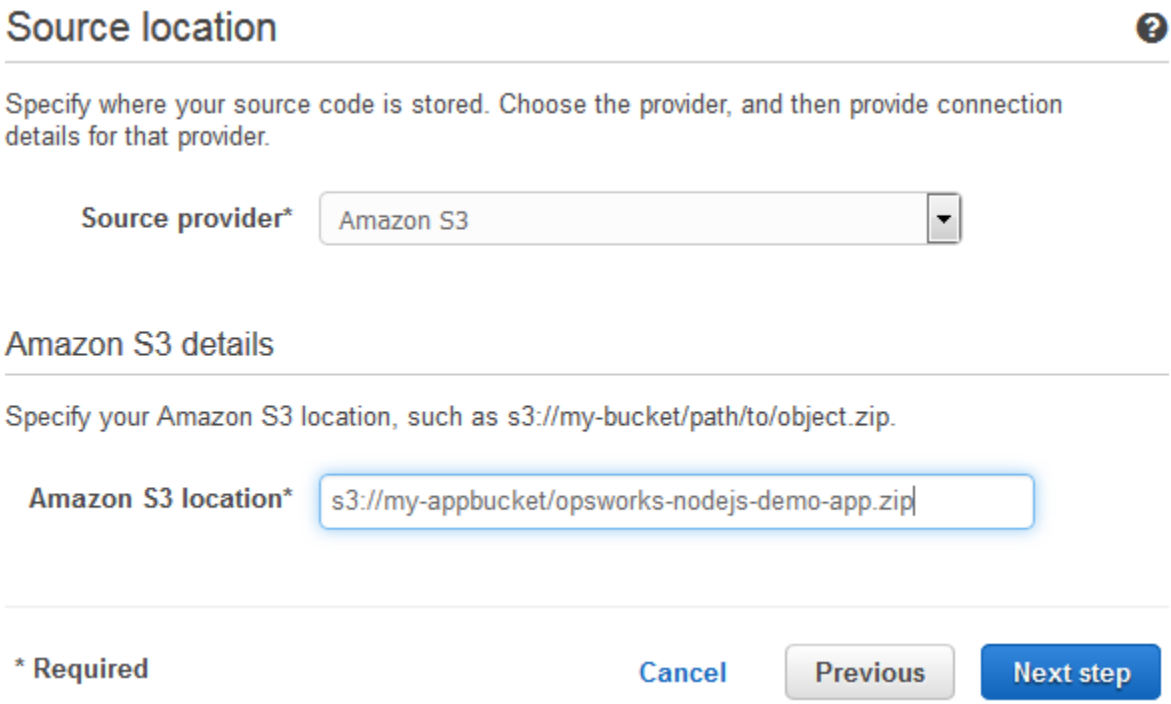
步驟 5：在 CodePipeline 中建立管道

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

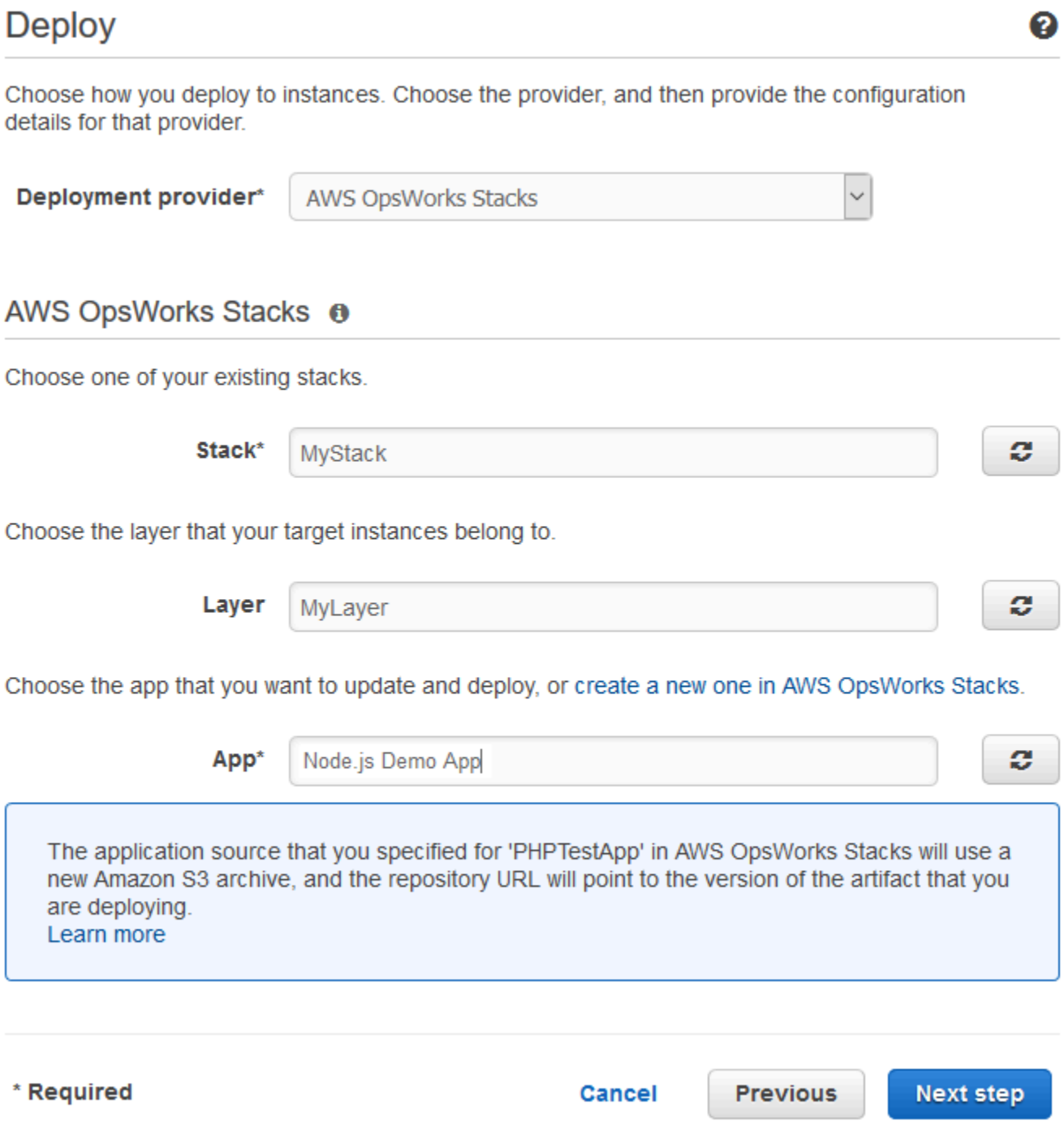
在 Stacks 中設定了具有 layer AWS OpsWorks 和至少一個執行個體的堆疊之後，請使用 AWS OpsWorks Stacks 在 CodePipeline 中建立管道，以將應用程式或 Chef 技術指南部署到您的 AWS OpsWorks Stacks 資源。

建立管道

1. 前往 https://console.aws.amazon.com/codepipeline/ 開啟 CodePipeline 主控台。
2. 選擇 Create pipeline (建立管道)。
3. 在 CodePipeline 入門頁面上，輸入 **MyOpsWorksPipeline**或您的帳戶唯一的任何其他管道名稱，然後選擇下一步。
4. 在 Source Location (來源位置) 頁面上，從 Source provider (來源提供者) 下拉式清單選取 Amazon S3。
5. 在 Amazon S3 詳細資訊區域中，以 格式輸入您的 Amazon S3 儲存貯體路徑**s3://*bucket-name*/*file name***。參考您在步驟 3：將應用程式碼上傳至 Amazon S3 儲存貯體的步驟 9 中記下的連結。在本演練中，路徑為 `s3://my-appbucket/opsworks-nodejs-demo-app.zip`。選擇 下一個步驟。



6. 在 Build (組建) 頁面上，從下拉式清單選擇 No Build (無組建)，然後選擇 Next step (下一步)。
7. 在 Deploy (部署) 頁面上，選擇 AWS OpsWorks Stacks 做為部署提供者。



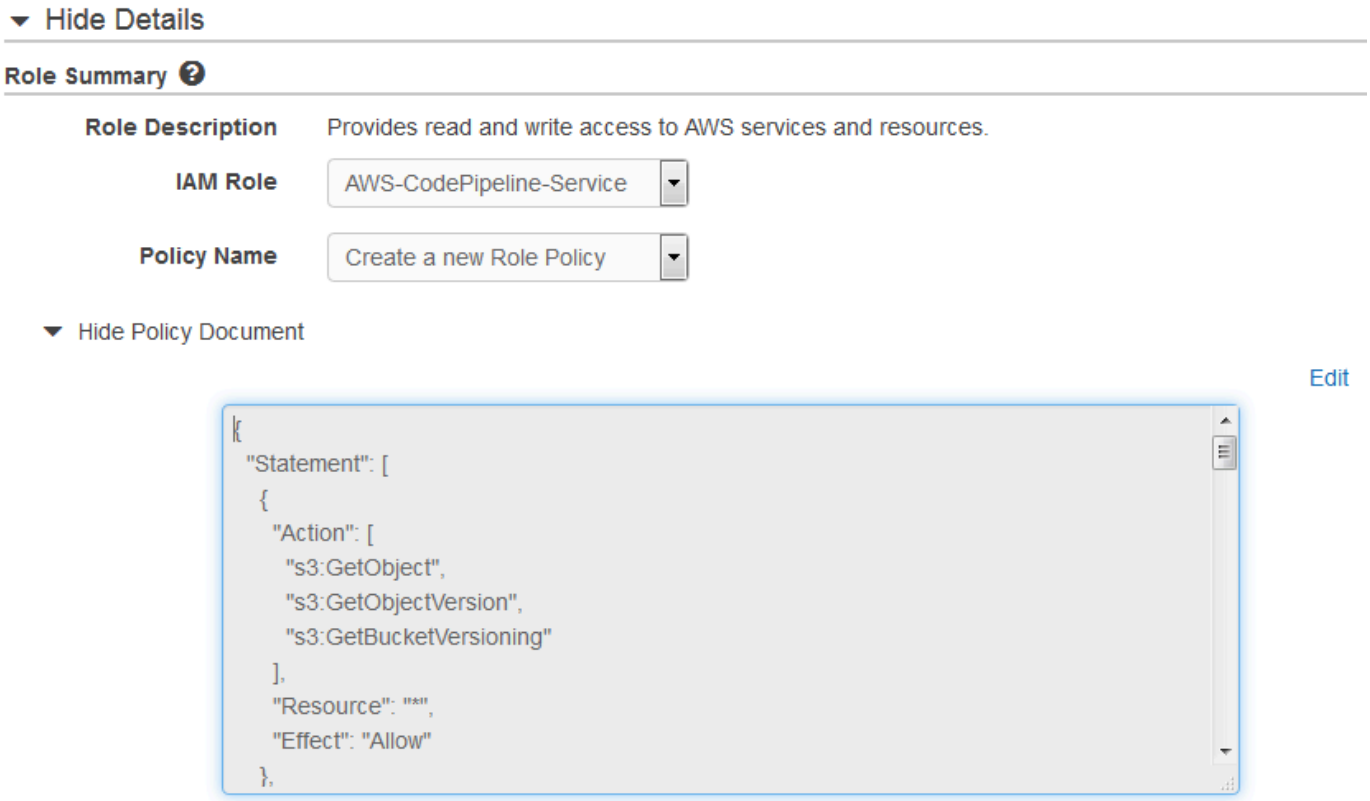
8. 在 Stack (堆疊) 欄位中輸入 `CodePipelineDemo`，或您在步驟 1：在 Stacks 中建立堆疊、 layer AWS OpsWorks 和執行個體中建立的堆疊名稱。
9. 在 Layer 欄位中輸入 `Node.js App Server`，或您在步驟 1：在 Stacks 中建立堆疊、 layer AWS OpsWorks 和執行個體中建立的 layer 名稱。

10. 在應用程式欄位中，選取您在 中上傳到 Amazon S3 的應用程式步驟 3：將應用程式碼上傳至 Amazon S3 儲存貯體，然後選擇下一步。
11. 在AWS 服務角色頁面上，選擇建立角色。

    新的視窗隨即開啟，其中包含 IAM 主控台頁面，說明將為您建立的角色AWS-CodePipeline-Service。從 Policy name (政策名稱) 下拉式清單，選擇 Create new policy (建立新政策)。請確認政策文件包含下列內容。如有需要，請選擇 Edit (編輯) 來變更政策文件。

```
{
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:GetBucketVersioning"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": "opsworks:*",
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

    完成政策文件的變更後，選擇 Allow (允許)。您的變更會顯示在 IAM 主控台中。



> **Note**
>
> 如果角色建立失敗，可能是因為您已經有名為 AWS-CodePipeline-Service 的 IAM 角色。如果您在 2016 年 5 月之前一直使用 AWS-CodePipeline-Service 角色，該角色可能沒有使用 AWS OpsWorks Stacks 做為部署提供者的許可。在本例中，您必須更新此政策陳述式，如本步驟中所示。若您看到錯誤訊息，請回到此步驟的開頭，並選擇 Use existing role (使用現有的角色) 而非 Create role (建立角色)。如果您使用現有的角色，該角色應會連接含有此步驟中所示許可的政策。如需服務角色及其政策陳述式的詳細資訊，請參閱編輯 IAM 服務角色的政策。

12. 如果角色建立程序成功，IAM 頁面會關閉，而且您會返回AWS 服務角色頁面。選擇 下一個步驟。
13. 在 Review your pipeline (檢閱管道) 頁面上，確認頁面顯示的選項，然後選擇 Create pipeline (建立管道)。
14. 當管道就緒時，其應會開始尋找來源碼，並自動將應用程式部署到堆疊。此程序需要幾分鐘的時間。

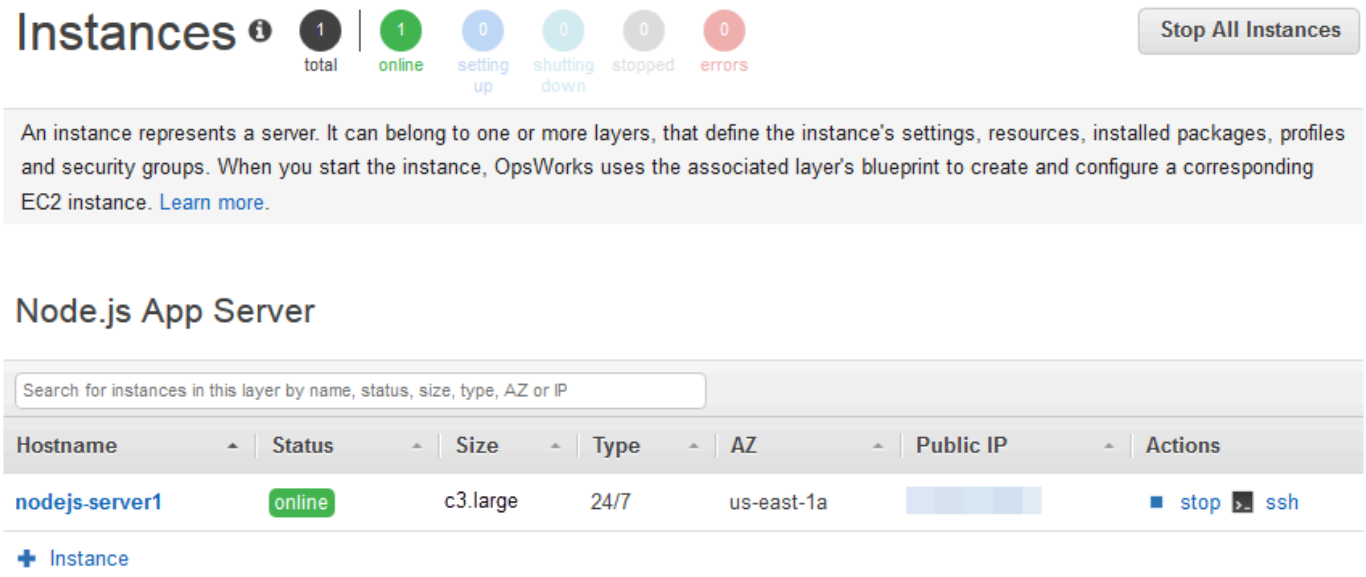
步驟 6：驗證 AWS OpsWorks Stacks 中的應用程式部署

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

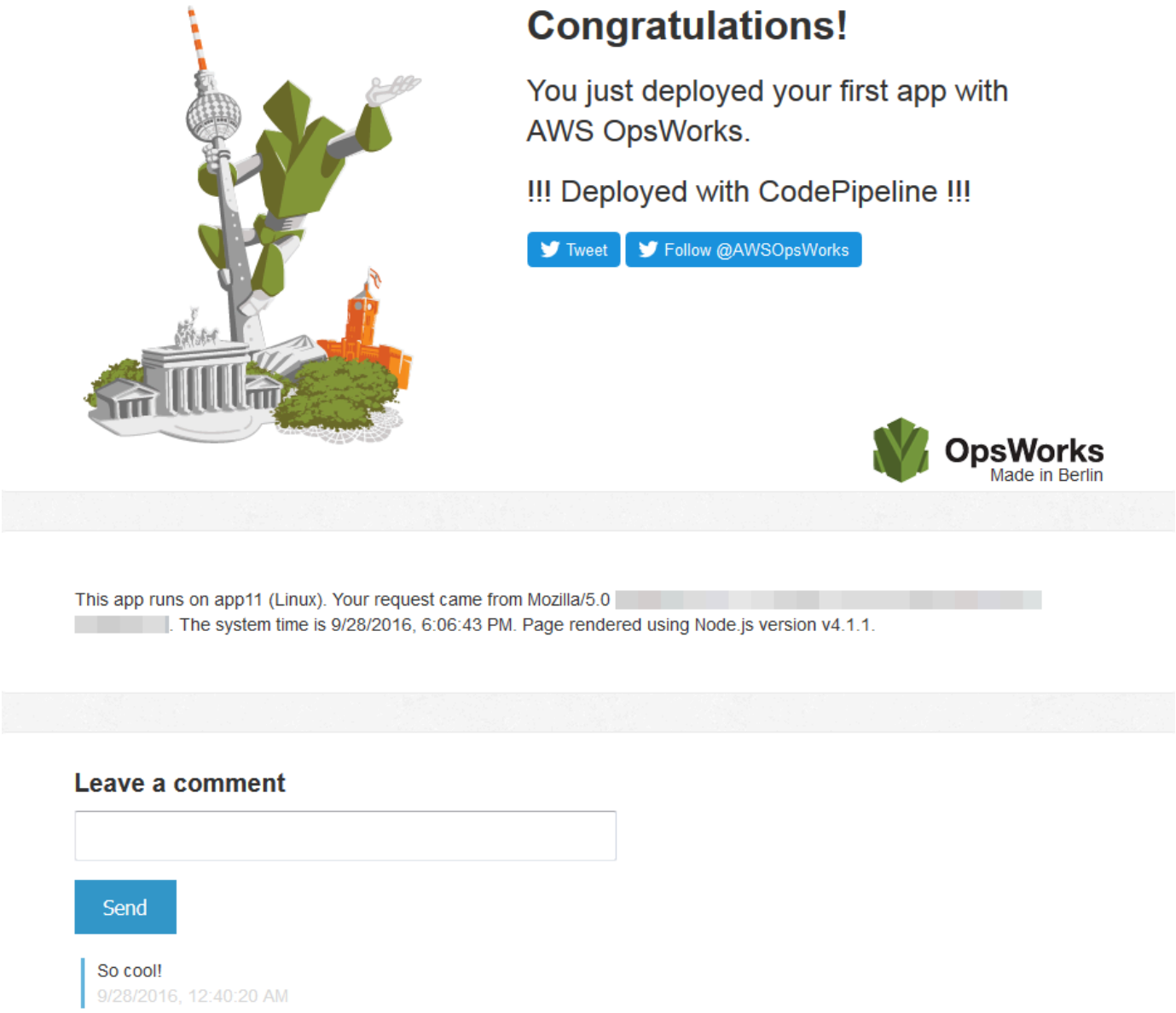
若要驗證 CodePipeline 已將 Node.js 應用程式部署到您的堆疊，請登入您在 中建立的執行個體步驟 1：在 Stacks 中建立堆疊、 layer AWS OpsWorks 和執行個體。您應該可以看到並使用 Node.js Web 應用程式。

驗證 Stacks AWS OpsWorks 執行個體中的應用程式部署

1. 在 https：//https://console.aws.amazon.com/opsworks/ 開啟 AWS OpsWorks 主控台。
2. 在 AWS OpsWorks Stacks 儀表板上，選擇 CodePipelineDemo，然後選擇 Node.js App Server。
3. 在導覽窗格中，選擇 Instances (執行個體)，然後選擇您建立之執行個體的公有 IP 地址，以檢視 Web 應用程式。



該應用程式會顯示在新的瀏覽器標籤中。



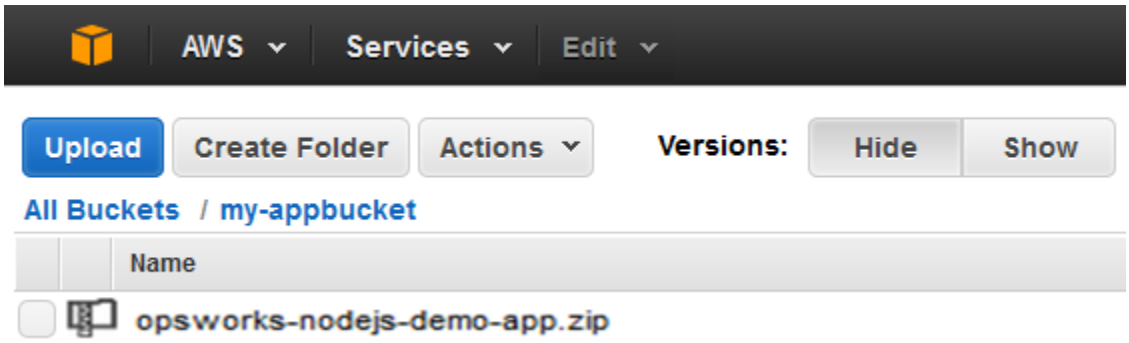
步驟 7 （選用）：更新應用程式程式碼，以查看 CodePipeline 自動重新部署您的應用程式

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

當您使用 CodePipeline 對已部署的應用程式或技術指南中的程式碼進行變更時，已更新的成品將由 CodePipeline 自動部署到您的目標執行個體（在此情況下為目標 Stacks AWS OpsWorks 堆疊）。本節示範在您更新範例 Node.js 應用程式中的程式碼時，自動重新部署。如果您仍將本演練的應用程式程式碼存放在本機，而且在您開始本演練後沒有人變更過程式碼，您可以略過此程序的步驟 1-4。

在範例應用程式中編輯程式碼

1. 登入 AWS Management Console 並開啟位於 https：//https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 開啟您存放範例 Node.js 應用程式的儲存貯體。



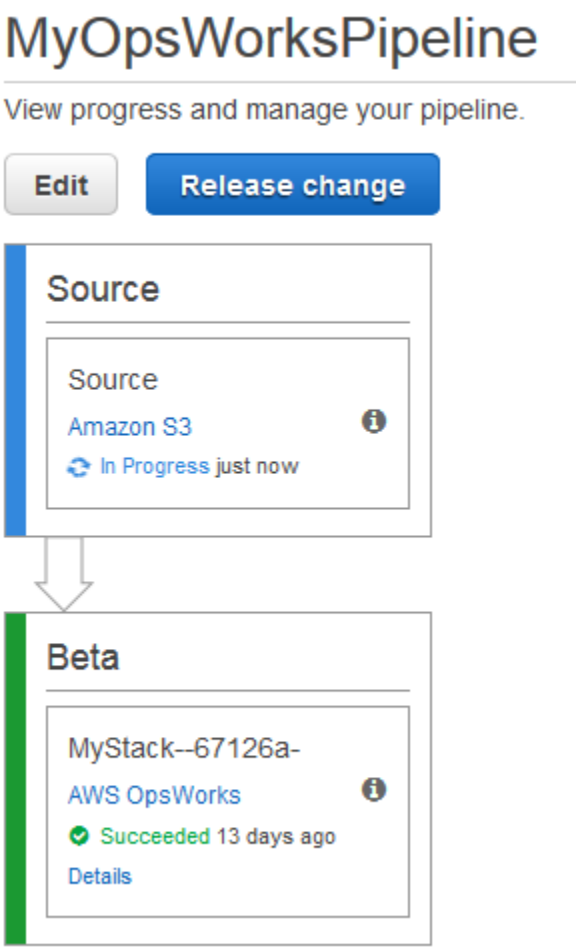
3. 選取內含應用程式的 ZIP 檔案。在 Actions (動作) 選單上，選擇 Download (下載)。
4. 在對話方塊中，(按一下右鍵) 開啟內容功能表，選擇 Download (下載)，然後將 ZIP 檔案儲存到方便使用的位置。選擇確定。
5. 將 ZIP 檔案的內容解壓縮到方便使用的位置。您可能需要變更已解壓縮資料夾及其子資料夾和內容的許可，以允許編輯。在 `opsworks-nodejs-demo-app\views` 資料夾中，開啟 `header.html` 檔案編輯。
6. 搜尋片語 `You just deployed your first app with`。使用 `updated` 取代 `deployed`。在下一行中，將 `AWS OpsWorks.` 變更成 `AWS OpsWorks and AWS CodePipeline.` 請不要編輯文字以外的內容。

```
<div id="main" role="main">LF
  <div class="container">LF
    <div class="hero-unit">LF
      <div class="robot">LF
        <h1>Congratulations!</h1>LF
        <h2>LF
          You just updated your first app with<br/>LF
          AWS OpsWorks and AWS CodePipeline.LF
        </h2>LF
```

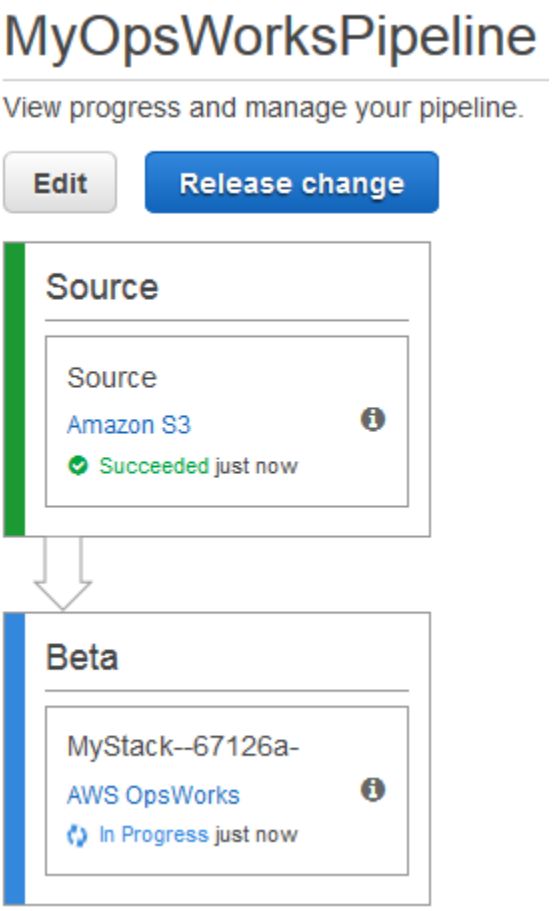
7. 儲存並關閉 `header.html` 檔案。

8. 壓縮 `opsworks-nodejs-demo-app` 資料夾，並將 ZIP 檔案儲存在方便的位置。請勿變更 ZIP 檔案名稱。
9. 將新的 ZIP 檔案上傳至您的 Amazon S3 儲存貯體。在本演練中，儲存貯體的名稱為 `my-appbucket`。
10. 開啟 CodePipeline 主控台，並開啟您的 AWS OpsWorks Stacks 管道 (MyOpsWorksPipeline)。選擇 Release Change (版本變更)。

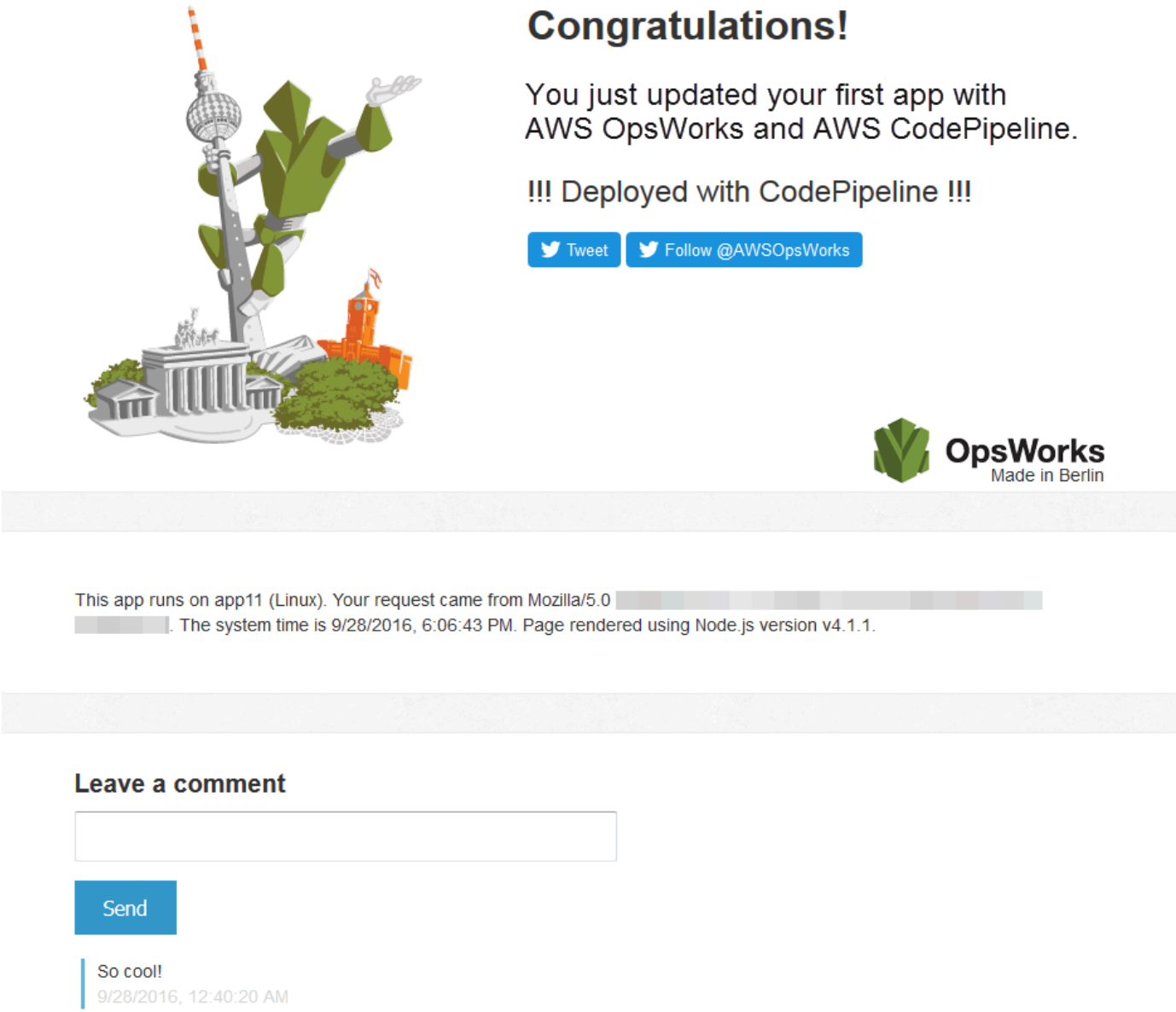
    （您可以等待 CodePipeline 偵測來自 Amazon S3 儲存貯體中應用程式更新版本的程式碼變更。為了節省您的時間，此演練會指示您直接選擇版本變更。）
11. 觀察 CodePipeline 執行管道的階段。首先，CodePipeline 會偵測來源成品的變更。



CodePipeline 會將更新的程式碼推送至 Stacks AWS OpsWorks 中的堆疊。



12. 當管道的兩個階段都成功完成後，開啟 AWS OpsWorks Stacks 中的堆疊。
13. 在堆疊屬性頁面中，選擇 Instances (執行個體)。
14. 在 Public IP (公有 IP) 欄位中，選擇您執行個體的公有 IP 地址，以檢視更新的應用程式文字。



步驟 8 (選用)：清理資源

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

為了協助避免 AWS 您的帳戶產生不必要的費用，您可以刪除您用於本演練 AWS 的資源。這些 AWS 資源包括 AWS OpsWorks Stacks 堆疊、IAM 角色和執行個體描述檔，以及您在 CodePipeline 中建立的管道。不過，當您繼續進一步了解 AWS OpsWorks Stacks 和 CodePipeline 時，建議您繼續使用這些 AWS 資源。如果您要保留這些資源，您已完成本演練。

從堆疊刪除應用程式

由於您未在 AWS CloudFormation 範本中建立或套用應用程式，請先刪除 Node.js 測試應用程式，再刪除堆疊 AWS CloudFormation。

1. 在 AWS OpsWorks Stacks 主控台的服務導覽窗格中，選擇應用程式。
2. 在 Apps (應用程式) 頁面上，選取 Node.js Demo App (Node.js 示範應用程式)，然後在 Actions (動作) 中，選擇 delete (刪除)。當您收到確認提示時，請選擇 Delete. AWS OpsWorks Stacks 將刪除應用程式。

刪除堆疊

由於您透過執行 AWS CloudFormation 範本建立堆疊，因此您可以在 AWS CloudFormation 主控台中刪除堆疊，包括範本建立的 layer、執行個體、執行個體描述檔和安全群組。

1. 開啟 AWS CloudFormation 主控台。
2. 在 AWS CloudFormation 主控台儀表板中，選取您建立的堆疊。在 Actions (動作) 選單上，選擇 Delete Stack (刪除堆疊)。出現確認提示時，選擇 Yes, Delete (是，刪除)。
3. 等待堆疊的 Status (狀態) 欄位中顯示 DELETE_COMPLETE。

刪除管道

1. 開啟 CodePipeline 主控台。
2. 在 CodePipeline 儀表板中，選擇您為此演練建立的管道。
3. 在管道頁面上，選擇 Edit (編輯)。
4. 在 Edit (編輯) 頁面上，選擇 Delete (刪除)。出現確認提示時，選擇 Delete (刪除)。

AWS CodePipeline 搭配 AWS OpsWorks Stacks - Chef 11 Stacks

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS CodePipeline 可讓您建立持續交付管道，以追蹤來自 CodeCommit、Amazon Simple Storage Service (Amazon S3) 或 GitHub 等來源的程式碼變更。本主題中的範例說明如何從 CodePipeline 建立和使用簡單的管道，做為您在 AWS OpsWorks Stacks layer 上執行之程式碼的部署工具。在此範例中，您會為簡單的 PHP 應用程式建立管道，然後指示 AWS OpsWorks Stacks 在 Chef 11.10 堆疊中層中的所有執行個體上執行應用程式（在此案例中為單一執行個體）。

> **Note**
>
> 本主題說明如何使用管道在 Chef 11.10 堆疊上執行與更新應用程式。如需如何使用管道在 Chef 12 堆疊上執行與更新應用程式的資訊，請參閱 AWS CodePipeline 搭配 AWS OpsWorks Stacks - Chef 12 Stacks。傳遞至 Amazon S3 儲存貯體的內容可能包含客戶內容。如需移除敏感資料的詳細資訊，請參閱如何清空 S3 儲存貯體？或如何刪除 S3 儲存貯體？。

主題

- 先決條件
- 其他支援的案例
- 步驟 1：在 AWS OpsWorks Stacks 中建立堆疊、layer 和執行個體
- 步驟 2：將應用程式碼上傳至 Amazon S3 儲存貯體
- 步驟 3：將您的應用程式新增至 AWS OpsWorks Stacks
- 步驟 4：在 CodePipeline 中建立管道
- 步驟 5：在 Stacks AWS OpsWorks 中驗證應用程式部署
- 步驟 6（選用）：更新應用程式程式碼，以查看 CodePipeline 自動重新部署您的應用程式
- 步驟 7 (選用)：清除資源

先決條件

在您開始本演練之前，請確定您具有執行下列所有任務的管理員許可。您可以是已套用 AdministratorAccess 政策之群組的成員，也可以是具有下表所示許可和政策之群組的成員。作為安全最佳實務，您應該屬於具有執行下列任務許可的群組，而不是將必要的許可指派給個別使用者。

如需在 IAM 中建立安全群組和指派許可給群組的詳細資訊，請參閱建立 IAM 使用者群組。如需管理 AWS OpsWorks Stacks 許可的詳細資訊，請參閱最佳實務：管理許可。

許可	建議連接至群組的政策
在 Stacks AWS OpsWorks 中建立和編輯堆疊、層和執行個體。	AWSOpsWorks_FullAccess
在 AWS CloudFormation中建立、編輯和執行範本。	AmazonCloudFormationFullAccess
建立、編輯和存取 Amazon S3 儲存貯體。	AmazonS3FullAccess
在 CodePipeline 中建立、編輯和執行管道，尤其是使用 Stacks AWS OpsWorks 做為提供者的管道。	AWSCodePipeline_FullAccess

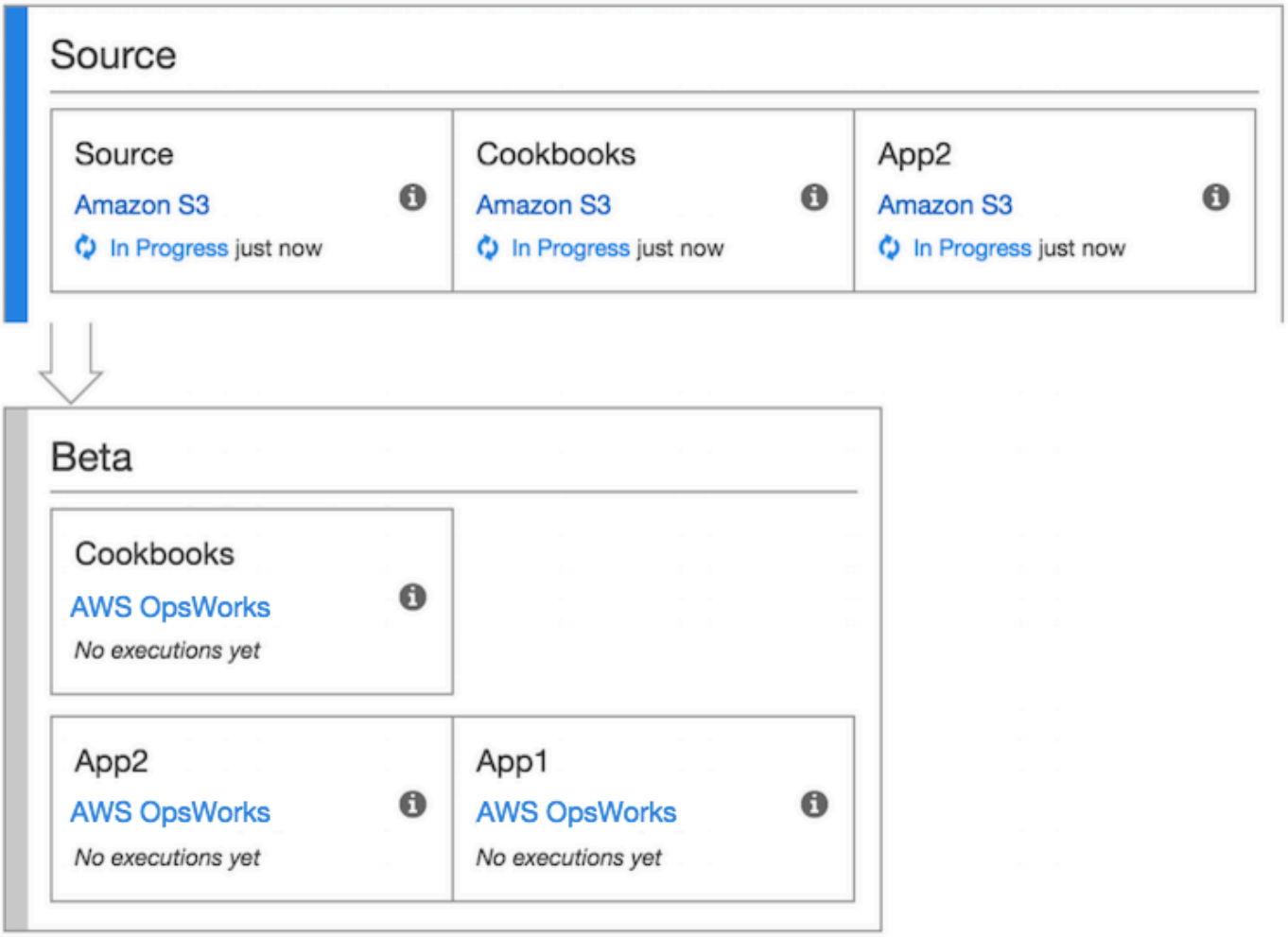
您也必須有 Amazon EC2 金鑰對。當您執行在此演練中建立範例堆疊、 layer 和執行個體的 AWS CloudFormation 範本時，系統會提示您提供此金鑰對的名稱。如需在 Amazon EC2 主控台中取得金鑰對的詳細資訊，請參閱 Amazon EC2 文件中的建立金鑰對。金鑰對應該位於美國東部 （維吉尼亞北部） 區域。如果您在該區域中已有金鑰對，則可以使用該現有金鑰對。

其他支援的案例

本演練會建立包含一個 Source (來源) 和一個 Deploy (部署) 階段的簡易管道。不過，您可以建立使用 Stacks AWS OpsWorks 做為提供者的更複雜管道。下列是支援的管道和案例範例：

- 您可以編輯管道，以將 Chef 技術指南新增至 Source (來源) 階段，以及將已更新技術指南的相關聯目標新增至 Deploy (部署) 階段。在此情況下，您可以新增 Deploy (部署) 動作，以在您變更來源時觸發技術指南更新。已更新的技術指南會比應用程式先部署。

- 您可以使用自訂技術指南和多個應用程式建立複雜的管道，並部署到 Stacks AWS OpsWorks 堆疊。該管道會同時追蹤對應用程式和技術指南來源所做的變更，並在您變更後重新部署。下圖示範類似的複雜管道範例：



如需使用 CodePipeline 的詳細資訊，請參閱 CodePipeline 文件。

步驟 1：在 AWS OpsWorks Stacks 中建立堆疊、layer 和執行個體

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

若要使用 AWS OpsWorks Stacks 做為管道的部署提供者，您必須先在 layer 中擁有堆疊、 layer 和至少一個執行個體。雖然您可以依照 Linux AWS OpsWorks Stacks 入門或 Windows Stacks 入門中的指示，在 Stacks 中建立堆疊，但此範例會使用 AWS CloudFormation 範本來建立 Linux Chef 11.10 堆疊、 layer 和執行個體。此範本建立的執行個體會執行 Amazon Linux 2016.03，且執行個體類型為 `c3.large`。

> **Important**
>
> AWS CloudFormation 範本必須與稍後上傳應用程式的 Amazon S3 儲存貯體位於相同的區域，以及稍後在 CodePipeline 中建立管道的相同區域。目前，CodePipeline 僅支援美國東部（維吉尼亞北部）區域 (us-east-1) 的 AWS OpsWorks Stacks 供應商。此演練中的所有資源都應在美國東部（維吉尼亞北部）區域建立。
>
> 如果堆疊建立失敗，您可能即將達到您帳戶的 IAM 角色允許數目上限。如果您的帳戶無法啟動執行個體類型為 `c3.large` 的執行個體，堆疊建立也可能失敗。例如，如果您使用 AWS 免費方案，您可能會收到錯誤，例如 `Root device type: must be included in EBS`。如果您的帳戶對您可以建立的執行個體類型有限制，例如 AWS 免費方案施加的限制，請嘗試將範本執行個體區塊中的 `InstanceType` 參數值變更為您的帳戶可以使用的執行個體類型。

使用 建立堆疊、 layer 和執行個體 AWS CloudFormation

1. 將下列 AWS CloudFormation 範本複製到新的純文字文件。將檔案儲存到本機電腦中方便使用的位置，並將其命名為 NewOpsWorksStack.template 或您方便使用的其他名稱。

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Mappings": {
    "Region2Principal": {
      "us-east-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "us-west-2": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "us-west-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
```

```
      "eu-west-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-southeast-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-northeast-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-northeast-2": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "ap-southeast-2": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "sa-east-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      },
      "cn-north-1": {
        "EC2Principal": "ec2.amazonaws.com.cn",
        "OpsWorksPrincipal": "opsworks.amazonaws.com.cn"
      },
      "eu-central-1": {
        "EC2Principal": "ec2.amazonaws.com",
        "OpsWorksPrincipal": "opsworks.amazonaws.com"
      }
    }
  },
  "Parameters": {
    "EC2KeyPairName": {
   "Type": "String",
   "Description": "The name of an existing EC2 key pair that allows you to use SSH
to connect to the OpsWorks instance."
  }
  },
  "Resources": {
"CPOpsDeploySecGroup": {
   "Type": "AWS::EC2::SecurityGroup",
```

```
  "Properties": {
    "GroupDescription" : "Lets you manage OpsWorks instances deployed to by
CodePipeline"
  }
},
"CPOpsDeploySecGroupIngressHTTP": {
  "Type": "AWS::EC2::SecurityGroupIngress",
  "Properties" : {
    "IpProtocol" : "tcp",
       "FromPort" : "80",
       "ToPort" : "80",
       "CidrIp" : "0.0.0.0/0",
 "GroupId": {
   "Fn::GetAtt": [
     "CPOpsDeploySecGroup", "GroupId"
   ]
 }
     }
},
"CPOpsDeploySecGroupIngressSSH": {
  "Type": "AWS::EC2::SecurityGroupIngress",
  "Properties" : {
    "IpProtocol" : "tcp",
       "FromPort" : "22",
       "ToPort" : "22",
       "CidrIp" : "0.0.0.0/0",
 "GroupId": {
   "Fn::GetAtt": [
     "CPOpsDeploySecGroup", "GroupId"
   ]
 }
  }
},
"MyStack": {
     "Type": "AWS::OpsWorks::Stack",
     "Properties": {
       "Name": {
         "Ref": "AWS::StackName"
       },
       "ServiceRoleArn": {
         "Fn::GetAtt": [
           "OpsWorksServiceRole",
           "Arn"
         ]
```

```
      },
  "ConfigurationManager" : { "Name": "Chef","Version": "11.10" },
  "DefaultOs": "Amazon Linux 2016.03",
      "DefaultInstanceProfileArn": {
        "Fn::GetAtt": [
          "OpsWorksInstanceProfile",
          "Arn"
        ]
      }
    }
  },
  "MyLayer": {
    "Type": "AWS::OpsWorks::Layer",
    "Properties": {
      "StackId": {
        "Ref": "MyStack"
      },
      "Name": "MyLayer",
      "Type": "php-app",
  "Shortname": "mylayer",
      "EnableAutoHealing": "true",
      "AutoAssignElasticIps": "false",
      "AutoAssignPublicIps": "true",
  "CustomSecurityGroupIds": [
    {
      "Fn::GetAtt": [
              "CPOpsDeploySecGroup", "GroupId"
      ]
          }
        ]
      },
      "DependsOn": [
        "MyStack",
        "CPOpsDeploySecGroup"
      ]
    },
    "OpsWorksServiceRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
```

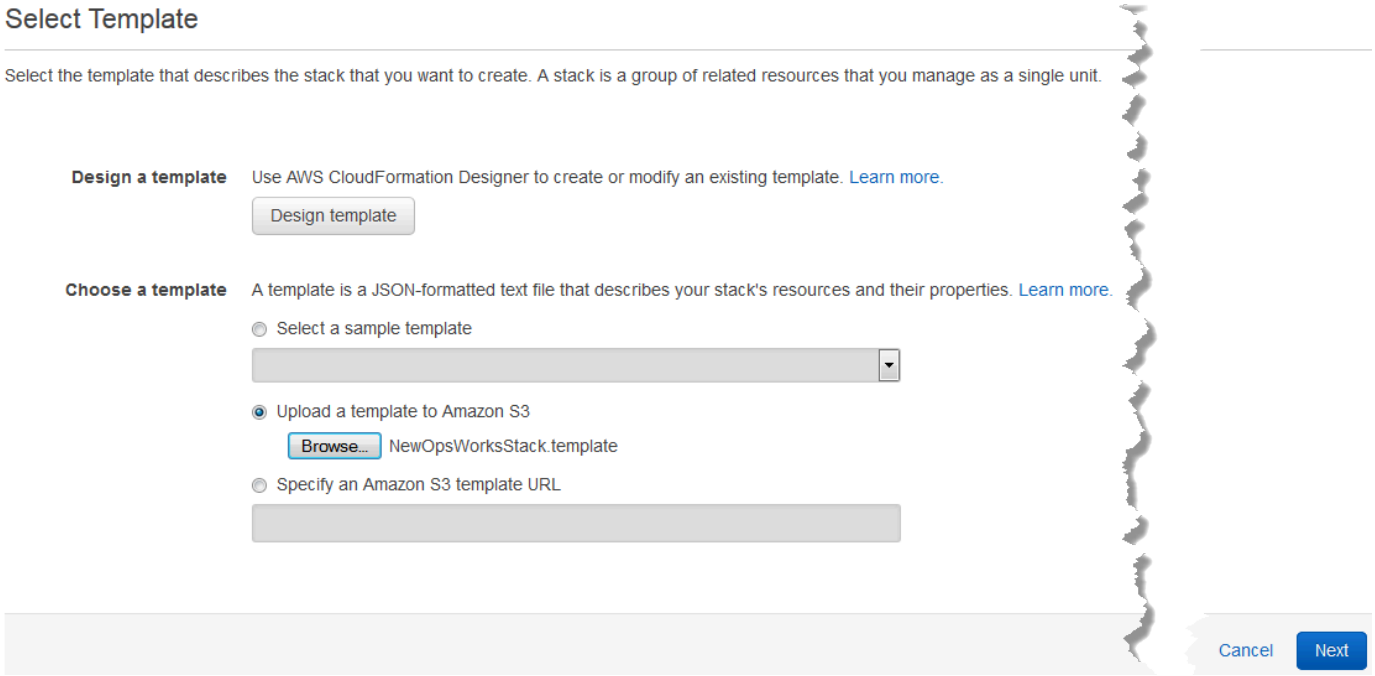
```
            "Service": [
              {
                "Fn::FindInMap": [
                  "Region2Principal",
                  {
                    "Ref": "AWS::Region"
                  },
                  "OpsWorksPrincipal"
                ]
              }
            ]
          },
          "Action": [
            "sts:AssumeRole"
          ]
        }
      ]
    },
    "Path": "/",
    "Policies": [
      {
        "PolicyName": "opsworks-service",
        "PolicyDocument": {
          "Statement": [
            {
              "Effect": "Allow",
              "Action": [
                "ec2:*",
                "iam:PassRole",
                "cloudwatch:GetMetricStatistics",
                "elasticloadbalancing:*"
              ],
              "Resource": "*"
            }
          ]
        }
      }
    ]
  }
},
"OpsWorksInstanceProfile": {
  "Type": "AWS::IAM::InstanceProfile",
  "Properties": {
    "Path": "/",
```

```
        "Roles": [
          {
            "Ref": "OpsWorksInstanceRole"
          }
        ]
      }
    },
    "OpsWorksInstanceRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  {
                    "Fn::FindInMap": [
                      "Region2Principal",
                      {
                        "Ref": "AWS::Region"
                      },
                      "EC2Principal"
                    ]
                  }
                ]
              },
              "Action": [
                "sts:AssumeRole"
              ]
            }
          ]
        },
        "Path": "/",
  "Policies": [
          {
            "PolicyName": "s3-get",
            "PolicyDocument": {
              "Version": "2012-10-17",
              "Statement": [
                {
                  "Effect": "Allow",
                  "Action": [
                    "s3:GetObject"
```

```
                ],
                "Resource": "*"
              }
            ]
          }
        }
      ]
    }
  },
  "myinstance": {
    "Type": "AWS::OpsWorks::Instance",
    "Properties": {
      "LayerIds": [
        {
          "Ref": "MyLayer"
        }
      ],
      "StackId": {
        "Ref": "MyStack"
      },
      "InstanceType": "c3.large",
      "SshKeyName": {
  "Ref": "EC2KeyPairName"
}
    }
  }
},
"Outputs": {
  "StackId": {
    "Description": "Stack ID for the newly created AWS OpsWorks stack",
    "Value": {
      "Ref": "MyStack"
    }
  }
}
}
```

2. 登入 AWS Management Console 並在 https：//https://console.aws.amazon.com/cloudformation 開啟 AWS CloudFormation 主控台。
3. 在 AWS CloudFormation 首頁上，選擇建立堆疊。
4. 在 Select Template (選取範本) 頁面上的 Choose a template (選擇範本) 區域中，選擇 Upload a template to Amazon S3 (將範本上傳至 Amazon S3)，然後選擇 Browse (瀏覽)。

5. 瀏覽至您在步驟 1 中儲存的 AWS CloudFormation 範本，然後選擇開啟。在 Select Template (選取範本) 頁面上，請選擇 Next (下一步)。

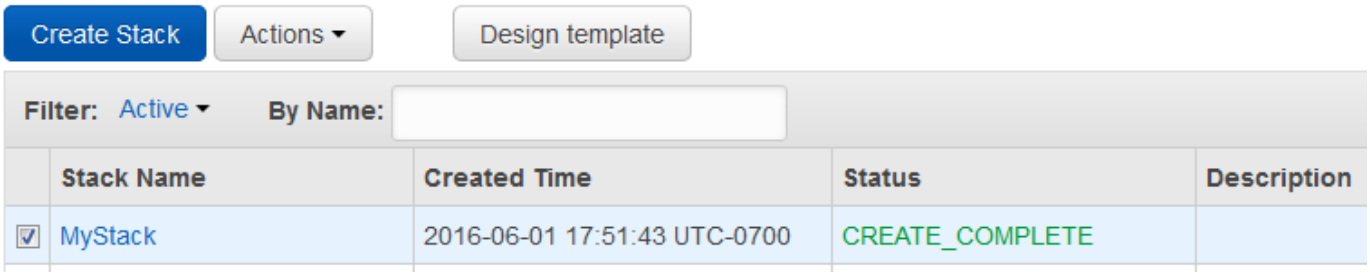


6. 在 Specify Details (指定詳細資訊) 頁面上，將堆疊命名為 MyStack 或對您帳戶而言為唯一的任何堆疊名稱。如果您選擇不同的堆疊名稱，請變更整個本演練中該堆疊的名稱。
7. 在參數區域中，提供您要用來在建立 Stacks 執行個體之後存取的 EC2 AWS OpsWorks 金鑰對名稱。選擇下一步。
8. 在選項頁面上，選擇下一步。(本演練不需要執行此頁面上的設定。)
9. 您在此演練中使用的 AWS CloudFormation 範本會建立 IAM 角色、執行個體描述檔和執行個體。

> **Important**
>
> 在選擇建立之前，請選擇成本來預估使用此範本 AWS 建立資源時可能衍生的費用。

如果可以建立 IAM 資源，請選取我確認此範本可能會導致 AWS CloudFormation 建立 IAM 資源核取方塊，然後選擇建立。如果無法建立 IAM 資源，則無法繼續此程序。

10. 在 AWS CloudFormation 儀表板上，您可以檢視建立堆疊的進度。請先等候 Status (狀態) 欄位顯示 CREATE_COMPLETE 後，再繼續進行下一步。



驗證在 Stacks AWS OpsWorks 中建立堆疊

1. 在 https：//https://console.aws.amazon.com/opsworks/ 開啟 AWS OpsWorks 主控台。
2. 在 AWS OpsWorks Stacks 儀表板上，檢視您建立的堆疊。

3. 開啟堆疊並檢視 layer 和執行個體。請注意，層和執行個體是使用 AWS CloudFormation 範本中提供的名稱和其他中繼資料建立的。您已準備好將應用程式上傳至 Amazon S3 儲存貯體。

步驟 2：將應用程式碼上傳至 Amazon S3 儲存貯體

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

由於您必須提供程式碼儲存庫的連結做為管道設定的一部分，因此在建立管道前，請先準備好程式碼儲存庫。在此演練中，您將 PHP 應用程式上傳到 Amazon S3 儲存貯體。

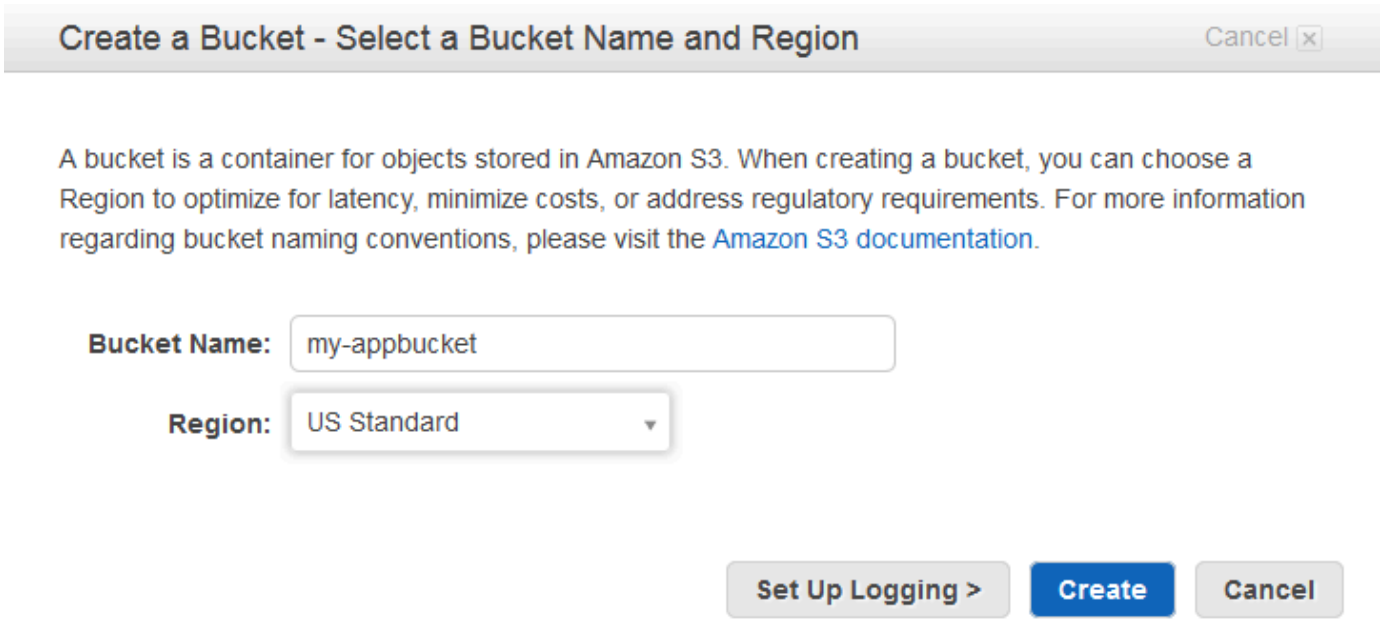
雖然 CodePipeline 可以直接從 GitHub 或 CodeCommit 使用程式碼做為來源，但本演練示範如何使用 Amazon S3 儲存貯體。Amazon S3 儲存貯體可讓 CodePipeline 偵測應用程式程式碼的變更，並自動部署變更的應用程式。您也可以視需要使用現有的儲存貯體。請確定儲存貯體符合 CodePipeline 標準，如 CodePipeline 文件中的簡易管道逐步解說 (Amazon S3 儲存貯體） 所述。

> **Important**
>
> Amazon S3 儲存貯體必須位於您稍後建立管道的相同區域。目前，CodePipeline 僅支援美國東部（維吉尼亞北部） 區域 (us-east-1) 的 AWS OpsWorks Stacks 供應商。此演練中的

所有資源都應在美國東部（維吉尼亞北部）區域建立。儲存貯體也必須進行版本化，因為 CodePipeline 需要版本化來源。如需詳細資訊，請參閱使用版本控制。

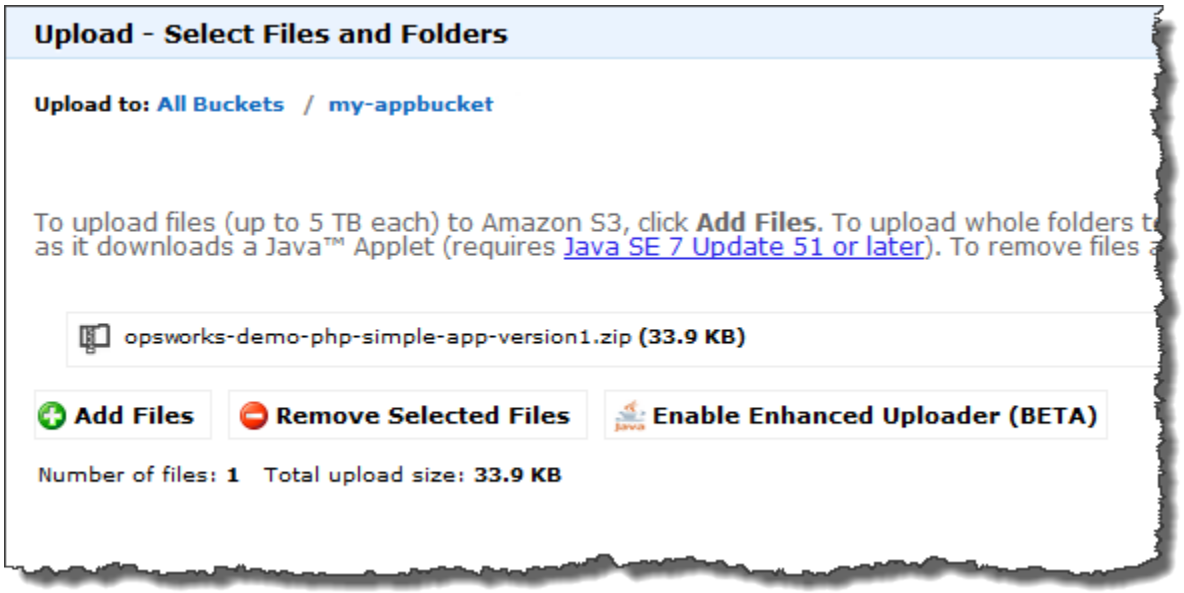
將應用程式上傳至 Amazon S3 儲存貯體

1. 從 GitHub 網站下載 AWS OpsWorks Stacks 範例 PHP 應用程式的 ZIP 檔案，並將其儲存至本機電腦上方便的位置。
2. 請確定 `index.php` 和 `ASSETS` 資料夾在已下載 ZIP 檔案的根層級。若未在根層級，請解壓縮該檔案，並建立讓這些檔案在根層級的新 ZIP 檔案。
3. 開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
4. 選擇 Create Bucket (建立儲存貯體)。
5. 在 Create a Bucket - Select a Bucket Name and Region (建立儲存貯體 - 選取儲存貯體名稱和區域) 頁面上，針對 Bucket Name (儲存貯體名稱)，輸入儲存貯體的唯一名稱。儲存貯體名稱在所有 AWS 帳戶中必須是唯一的，而不只是在您自己的帳戶中。本演練使用的名稱是 **`my-appbucket`**，但您可使用 `my-appbucket-`*`yearmonthday`*，讓您的儲存貯體名稱成為唯一名稱。從 Region (區域) 下拉式清單，選擇 US Standard (美國標準)，然後選擇 Create (建立)。US Standard (美國標準) 相當於 `us-east-1`。



6. 從 All Buckets (所有儲存貯體) 清單選擇您建立的儲存貯體。
7. 在儲存貯體頁面上，選擇 Upload (上傳)。

8. 在 Upload - Select Files and Folders (上傳 - 選取檔案和資料夾) 頁面上，選擇 Add files (新增檔案)。瀏覽您在步驟 1 中儲存的 ZIP 檔案，選擇 Open (開啟)，再選擇 Start Upload (開始上傳)。



9. 在上傳完成後，從您儲存貯體中的檔案清單選取 ZIP 檔案，然後選擇 Properties (屬性)。
10. 在 Properties (屬性) 窗格中，複製您 ZIP 檔案的連結，並記下此連結。您將需要儲存貯體名稱和此連結的 ZIP 檔案名稱部分來建立管道。

步驟 3：將您的應用程式新增至 AWS OpsWorks Stacks

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

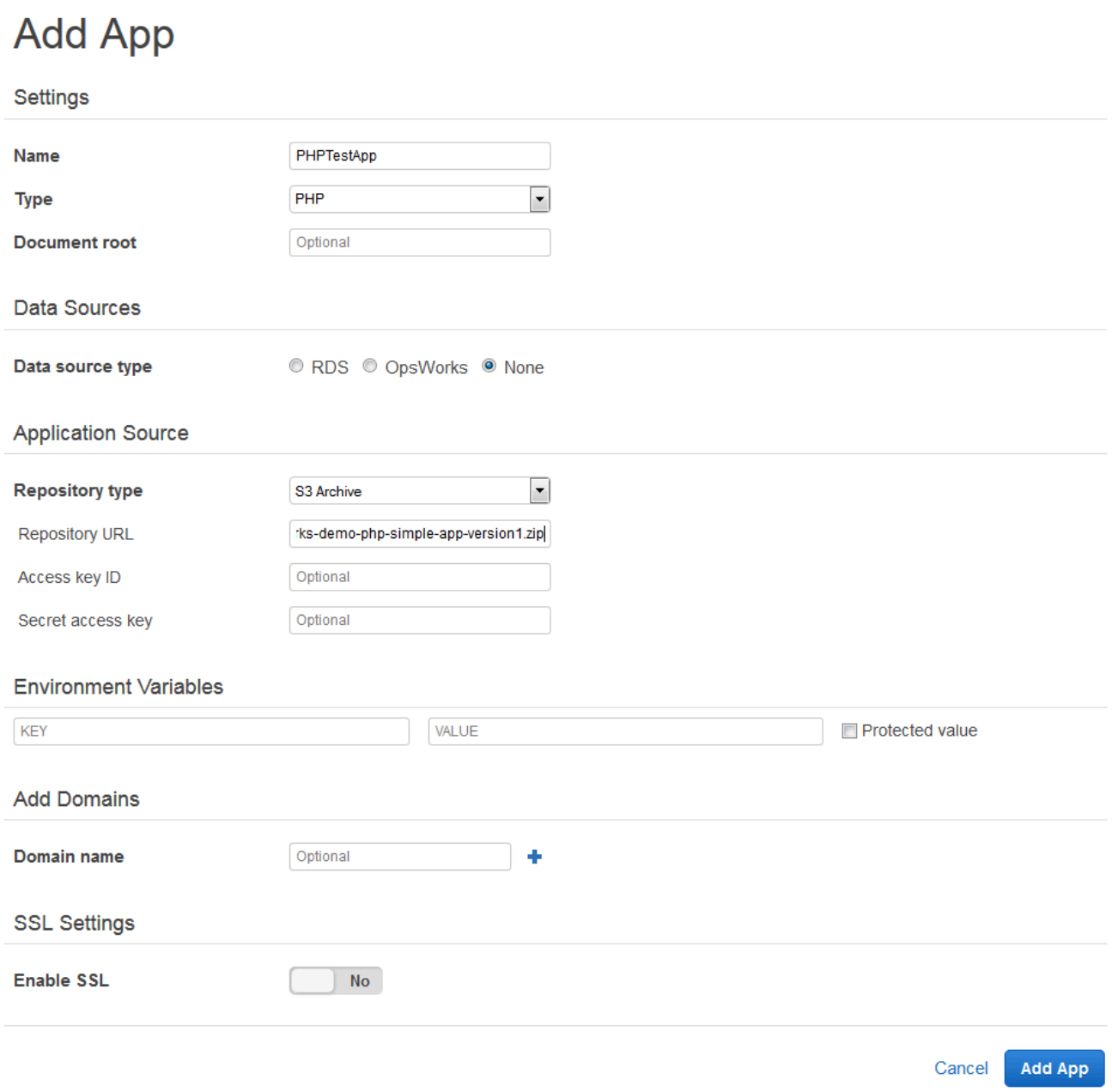
在 CodePipeline 中建立管道之前，請將 PHP 測試應用程式新增至 AWS OpsWorks Stacks。建立管道時，您需要選取已新增至 Stacks AWS OpsWorks 的應用程式。

準備好先前程序的步驟 10 中的 Amazon S3 儲存貯體連結。您將需要存放測試應用程式所在的儲存貯體連結來完成此程序。

將應用程式新增至 AWS OpsWorks Stacks

1. 在 AWS OpsWorks Stacks 主控台中，開啟 MyStack，然後在導覽窗格中選擇應用程式。

2. 選擇 Add app (新增應用程式)。
3. 在 Add App (新增應用程式) 頁面上，提供下列資訊：
   a. 為應用程式指定名稱。本逐步教學使用的名稱是 PHPTestApp。
   b. 在 Type (類型) 下拉式清單中，選擇 PHP。
   c. 針對 Data source type (資料來源類型)，選擇 None (無)。此應用程式不需要外部資料庫或資料來源。
   d. 在 Repository type (儲存庫類型) 下拉式清單中，選擇 S3 Archive (S3 封存)。
   e. 在 Repository URL (儲存庫 URL) 字串方格中，貼上您在步驟 2：將應用程式碼上傳至 Amazon S3 儲存貯體的步驟 10 中複製的 URL。您的表單應類似下列內容：



4. 您不需要變更此表單中的任何其他設定。選擇 Add App (新增應用程式)。
5. 當 PHPTestApp 應用程式出現在 Apps (應用程式) 頁面的清單中時，請繼續下一項程序：[步驟 4：在 CodePipeline 中建立管道](#)。

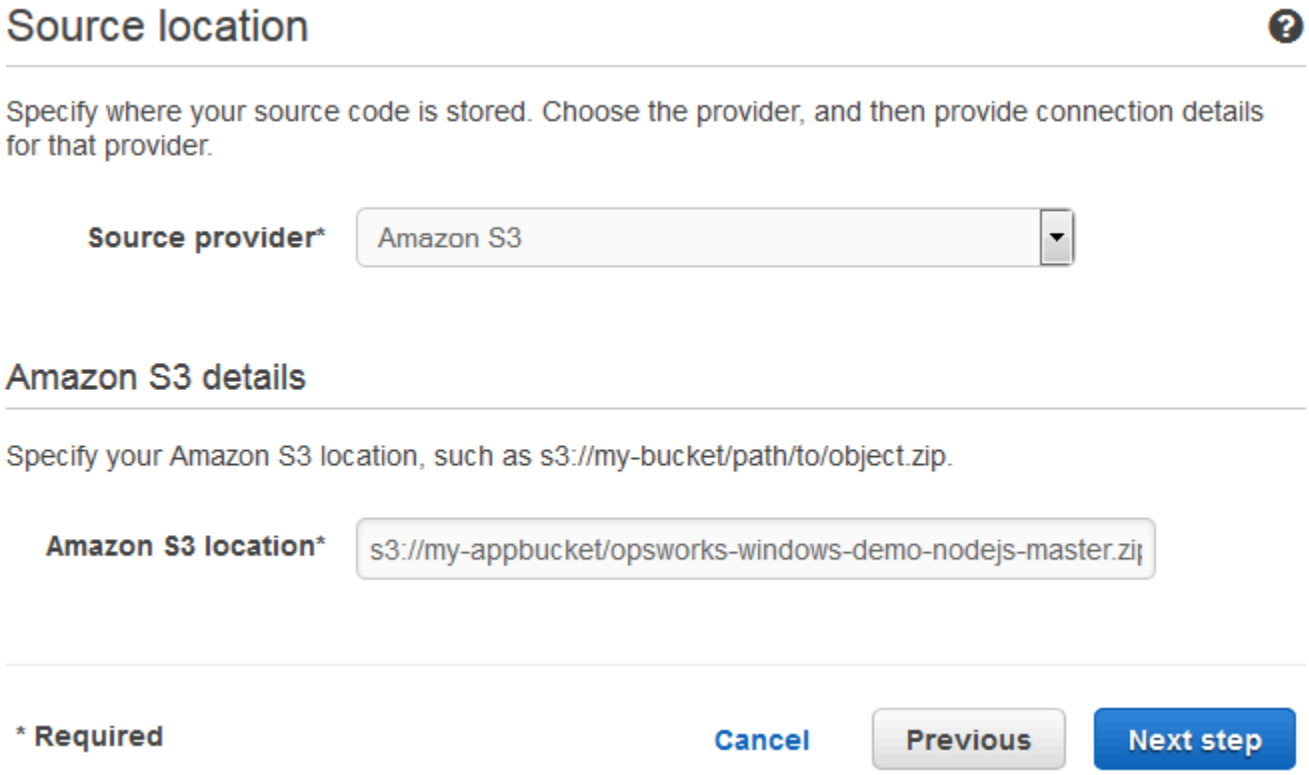
步驟 4：在 CodePipeline 中建立管道

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

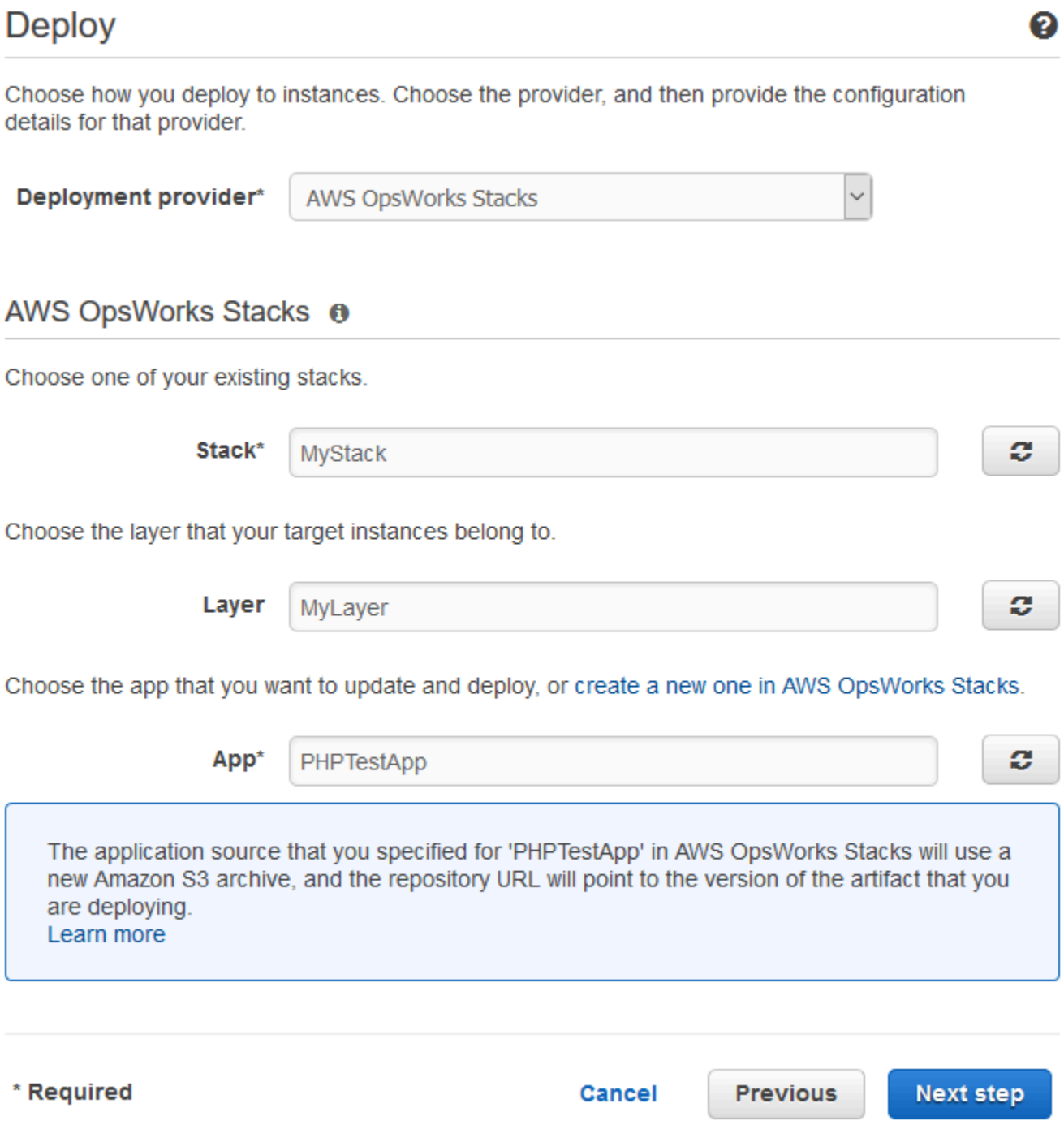
在 Stacks 中設定具有 layer AWS OpsWorks 和至少一個執行個體的堆疊之後，請使用 AWS OpsWorks Stacks 在 CodePipeline 中建立管道，以將應用程式或 Chef 技術指南部署到您的 AWS OpsWorks Stacks 資源。

建立管道

1. 前往 https://console.aws.amazon.com/codepipeline/ 開啟 CodePipeline 主控台。
2. 選擇 Create pipeline (建立管道)。
3. 在 CodePipeline 入門頁面上，輸入 **MyOpsWorksPipeline**或您的帳戶唯一的任何其他管道名稱，然後選擇下一步。
4. 在 Source Location (來源位置) 頁面上，從 Source provider (來源提供者) 下拉式清單選取 Amazon S3。
5. 在 Amazon S3 詳細資訊區域中，以 格式輸入您的 Amazon S3 儲存貯體路徑**s3://*bucket-name*/*file name***。參考您在步驟 2：將應用程式碼上傳至 Amazon S3 儲存貯體的步驟 10 中記下的連結。在本演練中，路徑為 `s3://my-appbucket/opsworks-demo-php-simple-app-version1.zip`。選擇 下一個步驟。



6. 在 Build (組建) 頁面上，從下拉式清單選擇 No Build (無組建)，然後選擇 Next step (下一步)。
7. 在 Deploy (部署) 頁面上，選擇 AWS OpsWorks Stacks 做為部署提供者。



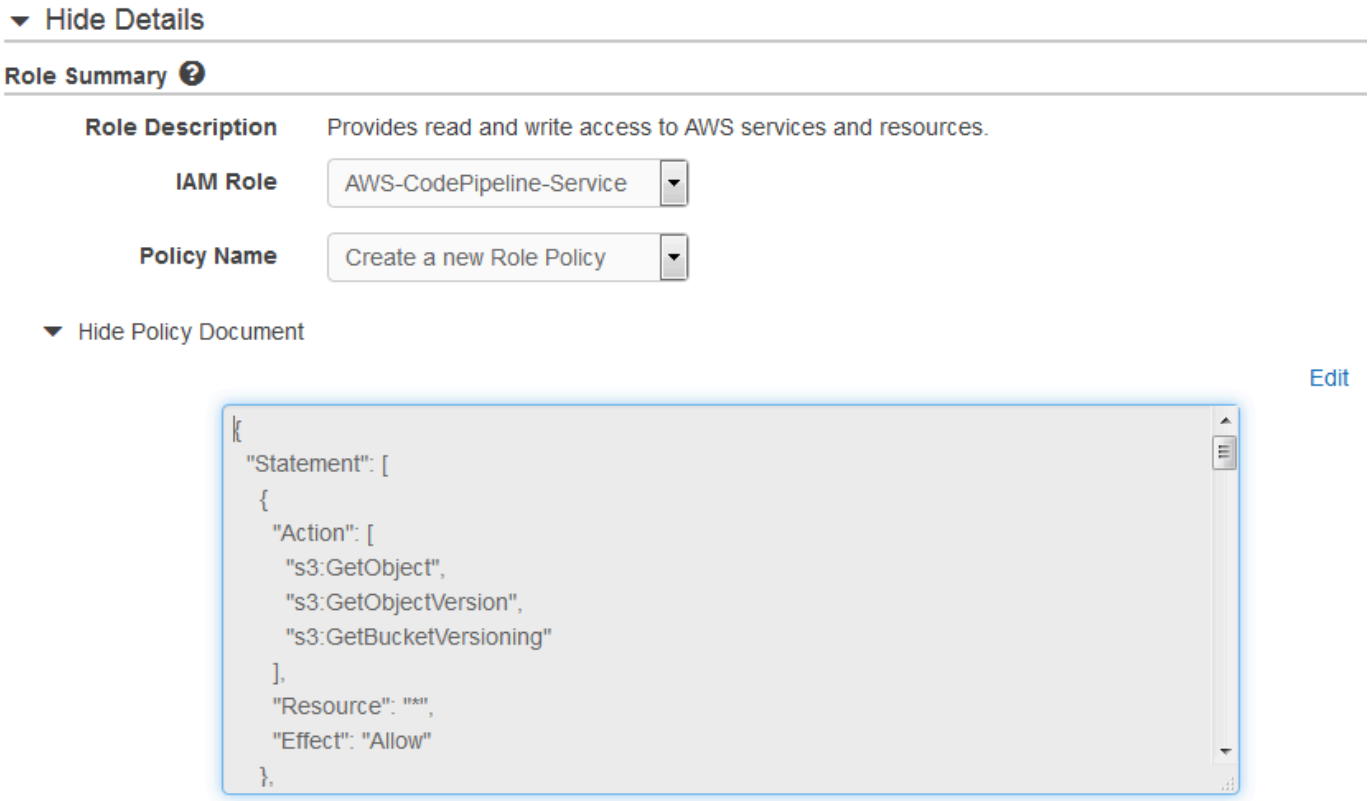
8. 在 Stack (堆疊) 欄位中輸入 `MyStack`，或您在[步驟 1：在 AWS OpsWorks Stacks 中建立堆疊、layer 和執行個體]中建立的堆疊名稱。
9. 在 Layer 欄位中輸入 `MyLayer`，或您在[步驟 1：在 AWS OpsWorks Stacks 中建立堆疊、layer 和執行個體]中建立的 layer 名稱。

10. 在應用程式欄位中，選取您在 中上傳到 Amazon S3 的應用程式步驟 2：將應用程式碼上傳至 Amazon S3 儲存貯體，然後選擇下一步。
11. 在 AWS Service Role (AWS 服務角色) 頁面上，選擇 Create Role (建立角色)。

    新的視窗隨即開啟，其中包含 IAM 主控台頁面，說明將為您建立的角色AWS-CodePipeline-Service。從 Policy name (政策名稱) 下拉式清單，選擇 Create new policy (建立新政策)。請確認政策文件包含下列內容。如有需要，請選擇 Edit (編輯) 來變更政策文件。

```
{
    "Statement": [
        {
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:GetBucketVersioning"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": "opsworks:*",
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

    完成政策文件的變更後，選擇 Allow (允許)。您的變更將在 IAM 主控台中發佈。



> **Note**
>
> 如果角色建立失敗，可能是因為您已經有名為 AWS-CodePipeline-Service 的 IAM 角色。如果您在 2016 年 5 月之前已使用 AWS-CodePipeline-Service 角色，該角色可能沒有使用 AWS OpsWorks Stacks 做為部署提供者的許可；在此情況下，您必須更新政策陳述式，如此步驟所示。若您看到錯誤訊息，請回到此步驟的開頭，並選擇 Use existing role (使用現有的角色) 而非 Create role (建立角色)。如果您使用現有的角色，該角色應會連接含有此步驟中所示許可的政策。如需服務角色及其政策陳述式的詳細資訊，請參閱編輯 IAM 服務角色的政策。

12. 如果角色建立程序成功，IAM 頁面會關閉，而且您會返回 AWS 服務角色頁面。選擇 下一個步驟。
13. 在 Review your pipeline (檢閱管道) 頁面上，確認頁面顯示的選項，然後選擇 Create pipeline (建立管道)。

We will create your pipeline with the following resources.

Source Stage

Source provider Amazon S3

Amazon S3 location s3://my-appbucket0/opsworks-demo-php-simple-app-version1.zip

Build Stage

Build provider No Build

Beta Stage

Deployment provider AWS OpsWorks

Stack MyStack

App PHPTestApp

Layer MyLayer

Pipeline settings

Pipeline name MyOpsWorksPipeline

Artifact location s3://codepipeline-us-east-[redacted]
AWS CodePipeline will use this existing S3 bucket to store artifacts for this pipeline. Depending on the size of your artifacts, you might be charged for storage costs. For more information, see Amazon S3 storage pricing.

Role name AWS-CodePipeline-Service

To save this configuration with these resources, choose Create pipeline.

Would you like to create this pipeline?

Cancel Previous Create pipeline

14. 當管道就緒時，其應會開始尋找來源碼，並自動將應用程式部署到堆疊。此程序需要幾分鐘的時間。

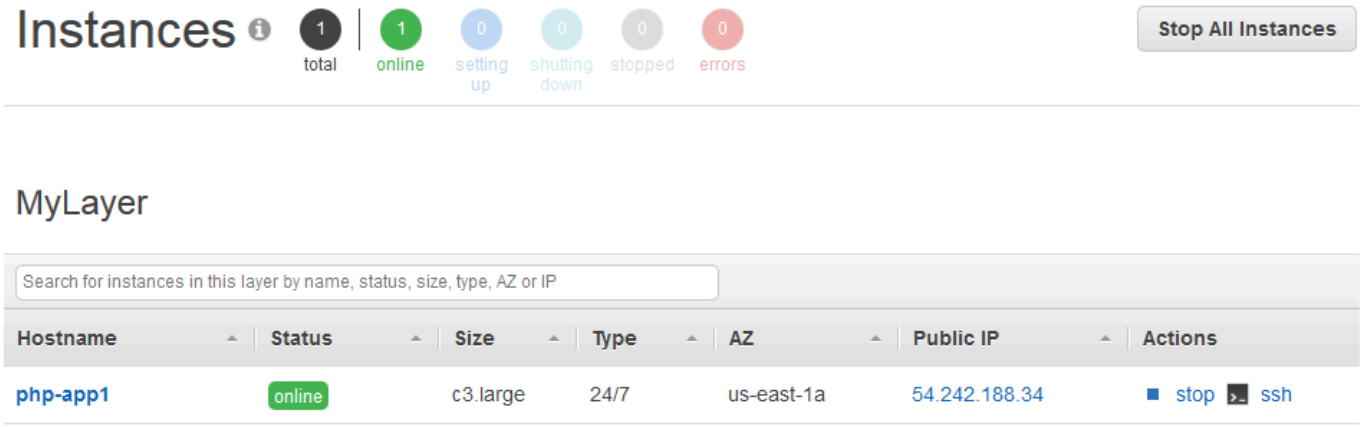
步驟 5：在 Stacks AWS OpsWorks 中驗證應用程式部署

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

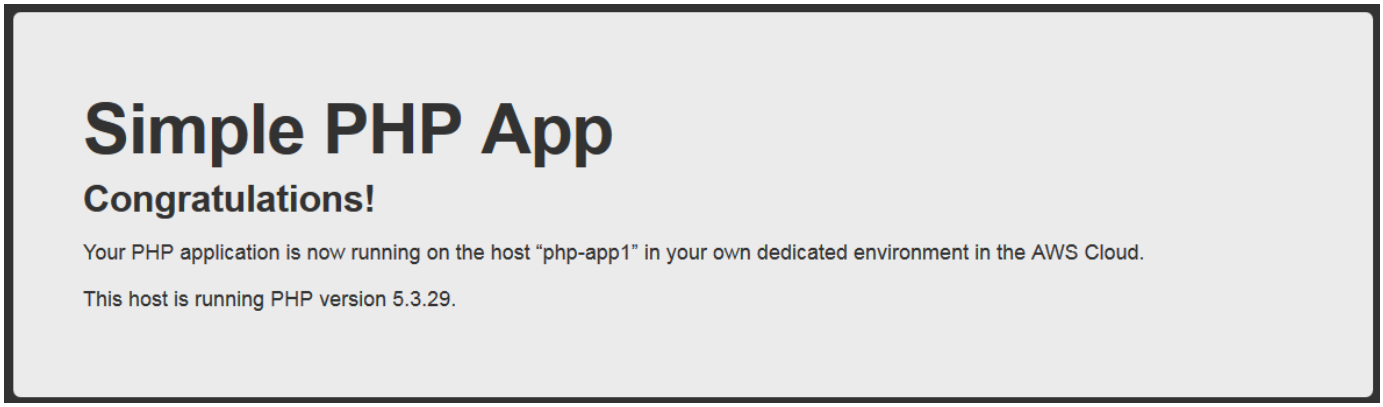
若要驗證 CodePipeline 是否已將 PHP 應用程式部署到您的堆疊，請登入您在 中建立的執行個體步驟 1：在 AWS OpsWorks Stacks 中建立堆疊、layer 和執行個體。您應該可以看到並使用 PHP Web 應用程式。

驗證 Stacks AWS OpsWorks 執行個體中的應用程式部署

1. 在 https：//https://console.aws.amazon.com/opsworks/ 開啟 AWS OpsWorks 主控台。
2. 在 AWS OpsWorks Stacks 儀表板上，選擇 MyStack，然後選擇 MyLayer。
3. 在導覽窗格中，選擇 Instances (執行個體)，然後選擇您建立之執行個體的公有 IP 地址，以檢視 Web 應用程式。



該應用程式會顯示在新的瀏覽器標籤中。



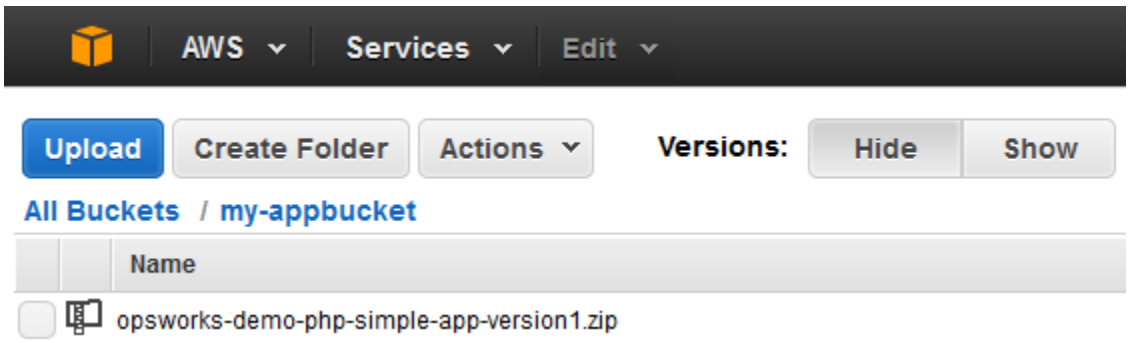
步驟 6 （選用）：更新應用程式程式碼，以查看 CodePipeline 自動重新部署您的應用程式

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

當您使用 CodePipeline 對已部署的應用程式或技術指南中的程式碼進行變更時，已更新的成品將由 CodePipeline 自動部署到您的目標執行個體 （在此情況下為目標 Stacks AWS OpsWorks 堆疊）。本節示範在您更新範例 PHP 應用程式程式碼時的自動重新部署。

在範例應用程式中編輯程式碼

1. 登入 AWS Management Console ，並在 https：//https://console.aws.amazon.com/s3/ 開啟 Amazon S3 主控台。
2. 開啟您存放範例 PHP 應用程式的儲存貯體。



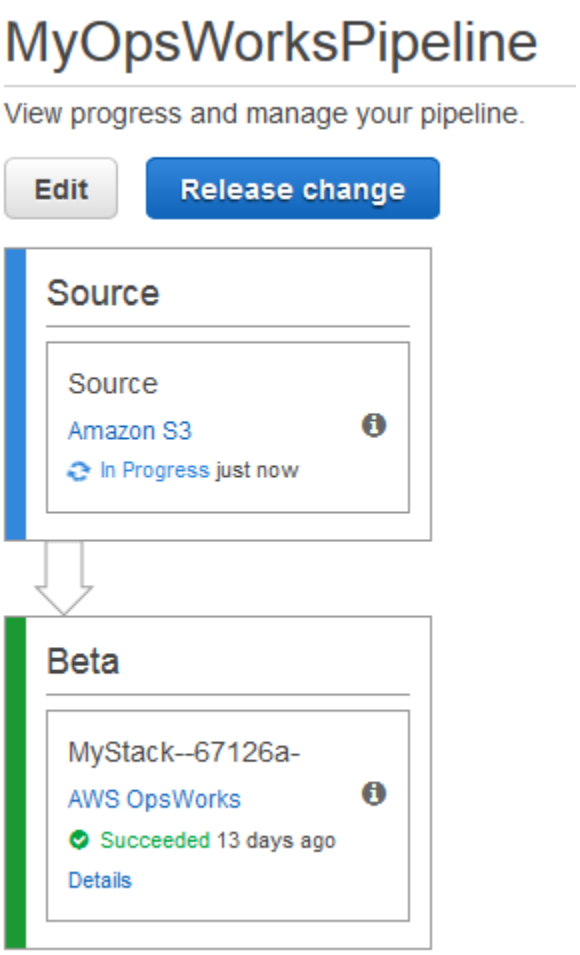
3. 選取內含應用程式的 ZIP 檔案。在 Actions (動作) 選單上，選擇 Download (下載)。

4. 在對話方塊中，(按一下右鍵) 開啟內容功能表，選擇 Download (下載)，然後將 ZIP 檔案儲存到方便使用的位置。選擇確定。
5. 將 ZIP 檔案的內容解壓縮到方便使用的位置。您可能需要變更已解壓縮資料夾及其子資料夾和內容的許可，以允許編輯。在 opsworks-demo-php-simple-app-version1 資料夾中，開啟 index.php 檔案編輯。
6. 搜尋片語 Your PHP application is now running。將 Your PHP application is now running 這段文字取代為 You've just deployed your first app to AWS OpsWorks with AWS CodePipeline,。請勿編輯變數。

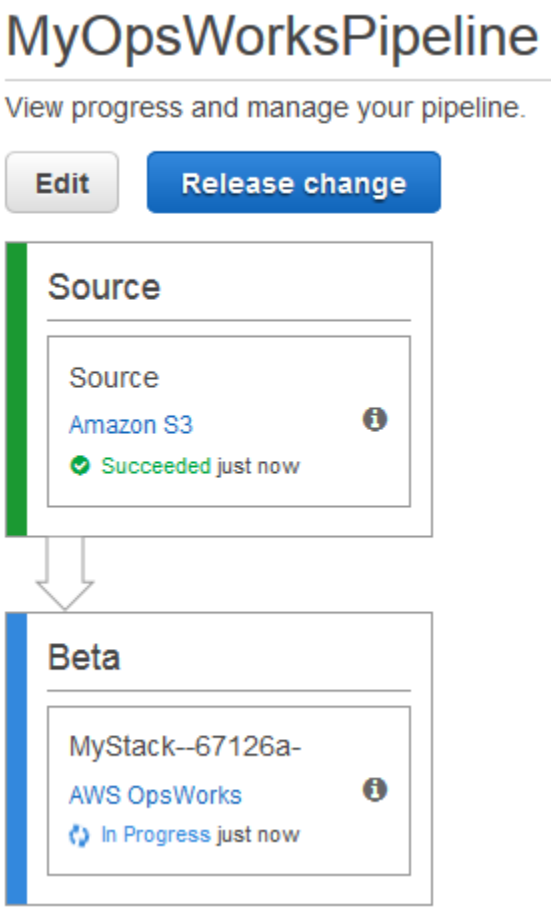
```
<body>
    <div class="container">
        <div class="hero-unit">
            <h1>Simple PHP App</h1>
            <h2>Congratulations!</h2>
            <p>You've just deployed your first app to AWS OpsWorks with AWS CodePipeline,</p>
            <p>on the host &ldquo;<?php echo gethostname(); ?>&rdquo; </p>
            <p>in your own dedicated environment in the AWS Cloud.</p>
            <p>This host is running PHP version <?php echo phpversion(); ?>.</p>
        </div>
    </div>

    <script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
    <script src="assets/js/bootstrap.min.js"></script>
</body>
```

7. 儲存並關閉 index.php 檔案。
8. 壓縮 opsworks-demo-php-simple-app-version1 資料夾，並將 ZIP 檔案儲存在方便的位置。請勿變更 ZIP 檔案名稱。
9. 將新的 ZIP 檔案上傳至您的 Amazon S3 儲存貯體。在本演練中，儲存貯體的名稱為 my-appbucket。
10. 開啟 CodePipeline 主控台，並開啟您的 AWS OpsWorks Stacks 管道 (MyOpsWorksPipeline)。選擇 Release Change (版本變更)。

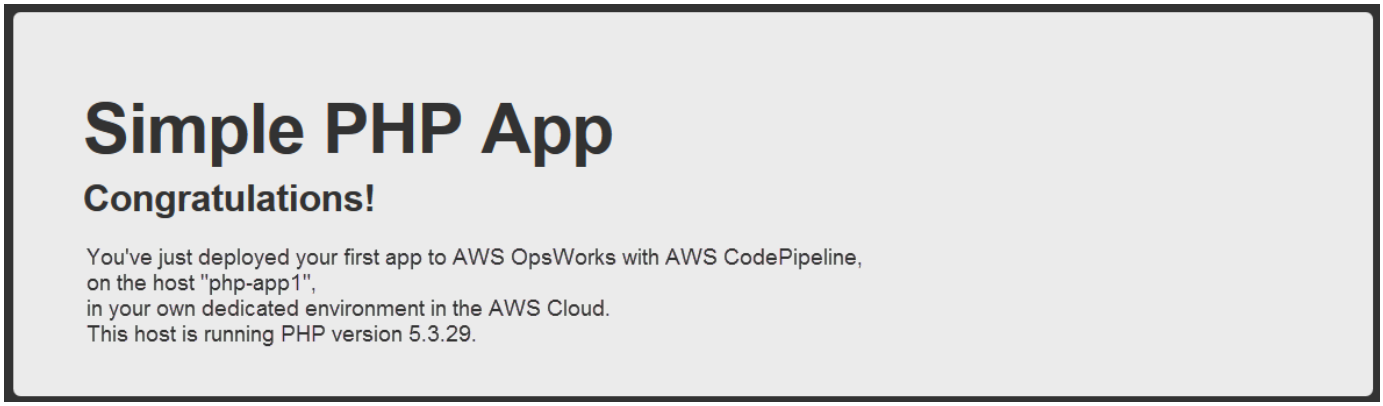
    (您可以等待 CodePipeline 偵測來自 Amazon S3 儲存貯體中應用程式更新版本的程式碼變更。為了節省您的時間，此演練會指示您直接選擇版本變更。)
11. 觀察 CodePipeline 執行管道的階段。首先，CodePipeline 會偵測來源成品的變更。



CodePipeline 會將更新的程式碼推送至 Stacks AWS OpsWorks 中的堆疊。



12. 當管道的兩個階段都成功完成後，開啟 AWS OpsWorks Stacks 中的堆疊 (MyStack)。
13. 在 MyStack 屬性頁面上，選擇 Instances (執行個體)。
14. 在 Public IP (公有 IP) 欄位中，選擇您執行個體的公有 IP 地址，以檢視更新的應用程式文字。



步驟 7 (選用)：清除資源

> **Important**
>
> AWS OpsWorks Stacks 此服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

為了協助避免您的 AWS 帳戶產生不必要的費用，您可以刪除用於本演練 AWS 的資源。這些 AWS 資源包括 AWS OpsWorks Stacks 堆疊、IAM 角色和執行個體描述檔，以及您在 CodePipeline 中建立的管道。不過，當您繼續進一步了解 AWS OpsWorks Stacks 和 CodePipeline 時，建議您繼續使用這些 AWS 資源。如果您要保留這些資源，您已完成本演練。

從堆疊刪除應用程式

由於您未在 AWS CloudFormation 範本中建立或套用應用程式，請先刪除 PHP 測試應用程式，再刪除堆疊 AWS CloudFormation。

1. 在 AWS OpsWorks Stacks 主控台的服務導覽窗格中，選擇應用程式。
2. 在 Apps (應用程式) 頁面上，選取 PHPTestApp，然後在 Actions (動作) 中選擇 delete (刪除)。當您收到確認提示時，請選擇 Delete. AWS OpsWorks Stacks 將刪除應用程式。

刪除堆疊

由於您透過執行 AWS CloudFormation 範本建立堆疊，因此您可以在 AWS CloudFormation 主控台中刪除堆疊，包括範本建立的 layer、執行個體、執行個體描述檔和安全群組。

1. 開啟 AWS CloudFormation 主控台。
2. 在 AWS CloudFormation 主控台儀表板中，選取您建立的堆疊 (MyStack)。在 Actions (動作) 選單上，選擇 Delete Stack (刪除堆疊)。出現確認提示時，選擇 Yes, Delete (是，刪除)。
3. 等待堆疊的 Status (狀態) 欄位中顯示 DELETE_COMPLETE。

刪除管道

1. 開啟 CodePipeline 主控台。
2. 在 CodePipeline 儀表板中，選擇您為此演練建立的管道。
3. 在管道頁面上，選擇 Edit (編輯)。

4. 在 Edit (編輯) 頁面上，選擇 Delete (刪除)。出現確認提示時，選擇 Delete (刪除)。

使用 AWS OpsWorks Stacks CLI

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Stacks AWS OpsWorks 命令列界面 (CLI) 提供與主控台相同的功能，可用於各種任務。Stacks CLI AWS OpsWorks 是 的一部分 AWS CLI。如需詳細資訊，包括如何安裝和設定 AWS CLI，請前往什麼是 AWS Command Line Interface？。如需每個命令的完整說明，請前往 AWS OpsWorks Stacks 參考。

> **Note**
>
> 如果您使用的是 Windows 型工作站，您也可以執行 AWS Tools for Windows PowerShell，從命令列執行 AWS OpsWorks Stacks 操作。如需詳細資訊，請參閱適用於 Windows PowerShell 的 AWS 工具。

AWS OpsWorks Stacks 命令具有下列一般格式：

```
aws opsworks --region us-west-1 opsworks command-name [--argument1 value] [...]
```

若引數值為 JSON 物件，您應逸出 " 字元，否則命令會傳回錯誤，顯示 JSON 無效。例如，若 JSON 物件為 "{"somekey":"somevalue"}"，您應將其格式化為 "{\"somekey\":\"somevalue\"}"。另一種方法為將 JSON 物件放置在檔案中，然後在命令列中使用 file:// 以包含它。以下範例會使用存放在 appsource.json 中的應用程式來源物件建立應用程式。

```
aws opsworks --region us-west-1 create-app --stack-id 8c428b08-a1a1-46ce-a5f8-
feddc43771b8 --name SimpleJSP --type java --app-source file://appsource.json
```

大多數命令會傳回一或多個值，並封裝為 JSON 物件。以下章節包含一些範例。如需每個命令傳回值的詳細說明，請前往 AWS OpsWorks Stacks 參考。

Note

AWS CLI 命令必須指定區域，如範例所示。--region 參數的有效值如下表所示。若要簡化 AWS OpsWorks Stacks 命令字串，請設定 CLI 來指定您的預設區域，讓您可以省略 `--region` 參數。如果您通常在多個區域端點中工作，請勿 AWS CLI 將 設定為使用預設區域端點。加拿大 （中部） 區域端點僅在 API 中提供 AWS CLI ，不適用於您在 中建立的堆疊 AWS Management Console。如需詳細資訊，請參閱設定 AWS 區域。

區域名稱	命令程式碼
美國東部 (俄亥俄) 區域	us-east-2
美國東部 (維吉尼亞北部) 區域	us-east-1
美國西部 (加利佛尼亞北部) 區域	us-west-1
美國西部 (奧勒岡) 區域	us-west-2
加拿大 (中部) 區域	ca-central-1
歐洲 (愛爾蘭) 區域	eu-west-1
歐洲 (倫敦) 區域	eu-west-2
歐洲 (巴黎) 區域	eu-west-3
歐洲 (法蘭克福) 區域	eu-central-1
亞太 (東京) 區域	ap-northeast-1
亞太 (首爾) 區域	ap-northeast-2
亞太區域 (孟買) 區域	ap-south-1
亞太區域 (新加坡) 區域	ap-southeast-1
亞太 (雪梨) 區域	ap-southeast-2
南美洲 (聖保羅) 區域	sa-east-1

若要使用 CLI 命令，您必須擁有適當的許可。如需 AWS OpsWorks Stacks 許可的詳細資訊，請參閱管理使用者許可。若要判斷特定命令需要的許可，請參閱 AWS OpsWorks Stacks 參考中該命令的參考頁面。

下列各節說明如何使用 AWS OpsWorks Stacks CLI 來執行各種常見的任務。

建立執行個體 (create-instance)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

使用 create-instance 命令來在指定堆疊上建立執行個體。

主題

- 使用預設主機名稱建立執行個體
- 使用主題主機名稱建立執行個體
- 使用自訂 AMI 建立執行個體

使用預設主機名稱建立執行個體

```
C:\>aws opsworks --region us-west-1 create-instance --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb
    --layer-ids 5c8c272a-f2d5-42e3-8245-5bf3927cb65b --instance-type m1.large --os
 "Amazon Linux"
```

引數如下：

- `stack-id` – 您可以從主控台上的堆疊設定頁面取得堆疊 ID (尋找 OpsWorks ID) 或呼叫 describe-stacks。
- `layer-ids` – 您可以從 主控台的 layer 詳細資訊頁面取得 layer IDs （ 尋找 OpsWorks ID) 或呼叫 describe-layers。在此範例中，執行個體僅屬於一個 layer。
- `instance-type` – 定義記憶體、CPU、儲存體容量，和執行個體每小時成本的規格。此範例為 `m1.large`。

- os – 執行個體的作業系統。此範例為 Amazon Linux。

命令會傳回包含執行個體 ID 的 JSON 物件，如下所示：

```
{
    "InstanceId": "5f9adeaa-c94c-42c6-aeef-28a5376002cd"
}
```

此範例會使用預設主機名稱 (僅為一個整數) 建立執行個體。下列章節說明如何使用從主題產生的主機名稱建立執行個體。

使用主題主機名稱建立執行個體

您也可以使用主題主機名稱建立執行個體。您會在建立堆疊時指定主題。如需詳細資訊，請參閱建立新的堆疊。若要建立執行個體，首先請呼叫 get-hostname-suggestion 以產生名稱。例如：

```
C:\>aws opsworks get-hostname-suggestion --region us-west-1 --layer-id 5c8c272a-
f2d5-42e3-8245-5bf3927cb65b
```

若您指定預設 Layer Dependent 主題，get-hostname-suggestion 只會將數字附加到 layer 的短名。如需詳細資訊，請參閱建立新的堆疊。

命令會傳回產生的主機名稱。

```
{
    "Hostname": "php-app2",
    "LayerId": "5c8c272a-f2d5-42e3-8245-5bf3927cb65b"
}
```

您接著可以使用 hostname 引數將產生的名稱傳遞給 create-instance，如下所示：

```
c:\>aws --region us-west-1 opsworks create-instance --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb
   --layer-ids 5c8c272a-f2d5-42e3-8245-5bf3927cb65b --instance-type m1.large --os
 "Amazon Linux" --hostname "php-app2"
```

使用自訂 AMI 建立執行個體

以下 create-instance 命令會使用自訂 AMI 建立執行個體，該自訂 AMI 必須來自堆疊的區域。如需如何建立自訂 AMI for Stacks AWS OpsWorks 的詳細資訊，請參閱 使用自訂 AMI。

```
C:\>aws opsworks create-instance --region us-west-1 --stack-id c5ef46ce-3ccd-472c-
a3de-9bec94c6028e
   --layer-ids 6ff8a2ac-c9cc-49cf-9c67-fc852539ade4 --instance-type c3.large --os
 Custom
   --ami-id ami-6c61f104
```

引數如下：

- `stack-id` – 您可以從主控台上的堆疊設定頁面取得堆疊 ID (尋找 OpsWorks ID) 或呼叫 describe-stacks。
- `layer-ids` – 您可以從 主控台的 layer 詳細資訊頁面取得 layer IDs （ 尋找 OpsWorks ID) 或呼叫 describe-layers。在此範例中，執行個體僅屬於一個 layer。
- `instance-type` – 值定義執行個體的記憶體、CPU、儲存體容量和每小時成本，且必須與 AMI 相容 (此範例中為 `c3.large`)。
- `os` – 執行個體的作業系統；自訂 AMI 必須設為 `Custom`。
- `ami-id` – AMI ID，形式看起來應該像 `ami-6c61f104`

> **Note**
>
> 當您使用自訂 AMI 時，不支援區塊型設備映射，因此您為 `--block-device-mappings` 選項指定的值將會遭到忽略。

命令會傳回包含執行個體 ID 的 JSON 物件，如下所示：

```
{
    "InstanceId": "5f9adeaa-c94c-42c6-aeef-28a5376002cd"
}
```

部署應用程式 (create-deployment)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

使用 create-deployment 命令將應用程式部署到指定堆疊。

主題

- 部署應用程式

部署應用程式

```
aws opsworks --region us-west-1 create-deployment --stack-id cfb7e082-ad1d-4599-8e81-
de1c39ab45bf
   --app-id 307be5c8-d55d-47b5-bd6e-7bd417c6c7eb --command "{\"Name\":\"deploy\"}"
```

引數如下：

- `stack-id` – 您可以從主控台上的堆疊設定頁面取得堆疊 ID (尋找 OpsWorks ID) 或呼叫 `describe-stacks`。
- `app-id` – 您可以從應用程式詳細資訊頁面取得應用程式 ID (尋找 OpsWorks ID) 或呼叫 describe-apps。
- `command` – 引數接受一個將命令名稱設為 `deploy` 的 JSON 物件，將指定應用程式部署到堆疊。

  請注意，JSON 物件中的 " 字元全部都已逸出。否則，命令可能傳回錯誤，顯示 JSON 無效。

命令會傳回包含部署 ID 的 JSON 物件，如下所示：

```
{
    "DeploymentId": "5746c781-df7f-4c87-84a7-65a119880560"
}
```

> **Note**
>
> 上述範例會部署到堆疊中的每個執行個體。若要部署到指定的執行個體子集，請新增 `instance-ids` 引數並列出執行個體 ID。

列出堆疊的應用程式 (describe-apps)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

使用 describe-apps 命令來列出堆疊的應用程式或取得特定應用程式的詳細資訊。

```
aws opsworks --region us-west-1 describe-apps --stack-id 38ee91e2-abdc-4208-
a107-0b7168b3cc7a
```

上述範例會傳回一個 JSON 物件，其中包含每個應用程式的相關資訊。此範例只有一個應用程式。如需每個參數的說明，請參閱 describe-apps。

```
{
  "Apps": [
    {
      "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
      "AppSource": {
        "Url": "url",
        "Type": "archive"
      },
      "Name": "SimpleJSP",
      "EnableSsl": false,
      "SslConfiguration": {},
      "AppId": "da1decc1-0dff-43ea-ad7c-bb667cd87c8b",
      "Attributes": {
        "RailsEnv": null,
        "AutoBundleOnDeploy": "true",
        "DocumentRoot": "ROOT"
```

```
      },
      "Shortname": "simplejsp",
      "Type": "other",
      "CreatedAt": "2013-08-01T21:46:54+00:00"
    }
  ]
}
```

列出堆疊的命令 (describe-commands)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

使用 describe-commands 命令來列出堆疊的命令或取得特定命令的詳細資訊。以下範例會取得已於指定之執行個體上執行的命令資訊。

```
aws opsworks --region us-west-1 describe-commands --instance-id
 8c2673b9-3fe5-420d-9cfa-78d875ee7687
```

命令會傳回一個 JSON 物件，其中包含每個命令的詳細資訊。Type 參數會識別命令名稱，此範例中為 deploy 或 undeploy。如需其他參數的說明，請參閱 describe-commands。

```
{
  "Commands": [
    {
      "Status": "successful",
      "CompletedAt": "2013-07-25T18:57:47+00:00",
      "InstanceId": "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
      "DeploymentId": "6ed0df4c-9ef7-4812-8dac-d54a05be1029",
      "AcknowledgedAt": "2013-07-25T18:57:41+00:00",
      "LogUrl": "https://s3.amazonaws.com/prod_stage-log/logs/008c1a91-
ec59-4d51-971d-3adff54b00cc?AWSAccessKeyId=AIDACKCEVSQ6C2EXAMPLE
 &Expires=1375394373&Signature=HkXil6UuNfxTCC37EPQAa462E1E%3D&response-cache-
control=private&response-content-encoding=gzip&response-content- type=text%2Fplain",
      "Type": "undeploy",
      "CommandId": "008c1a91-ec59-4d51-971d-3adff54b00cc",
```

```
      "CreatedAt": "2013-07-25T18:57:34+00:00",
      "ExitCode": 0
    },
    {
      "Status": "successful",
      "CompletedAt": "2013-07-25T18:55:40+00:00",
      "InstanceId": "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
      "DeploymentId": "19d3121e-d949-4ff2-9f9d-94eac087862a",
      "AcknowledgedAt": "2013-07-25T18:55:32+00:00",
      "LogUrl": "https://s3.amazonaws.com/prod_stage-log/
logs/899d3d64-0384-47b6-a586-33433aad117c?AWSAccessKeyId=AIDACKCEVSQ6C2EXAMPLE
 &Expires=1375394373&Signature=xMsJvtLuUqWmsr8s%2FAjVru0BtRs%3D&response-cache-
control=private&response-content-encoding=gzip&response-conten t-type=text%2Fplain",
      "Type": "deploy",
      "CommandId": "899d3d64-0384-47b6-a586-33433aad117c",
      "CreatedAt": "2013-07-25T18:55:29+00:00",
      "ExitCode": 0
    }
  ]
}
```

列出堆疊的部署 (describe-deployments)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

使用 describe-deployments 命令來列出堆疊的部署或取得特定部署的詳細資訊。

```
aws opsworks --region us-west-1 describe-deployments --stack-id 38ee91e2-abdc-4208-
a107-0b7168b3cc7a
```

上述命令會傳回一個 JSON 物件，其中包含指定堆疊每個部署的詳細資訊。如需每個參數的說明，請參閱 describe-deployments。

```
{
  "Deployments": [
      {
```

```
            "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
            "Status": "successful",
            "CompletedAt": "2013-07-25T18:57:49+00:00",
            "DeploymentId": "6ed0df4c-9ef7-4812-8dac-d54a05be1029",
            "Command": {
                "Args": {},
                "Name": "undeploy"
            },
            "CreatedAt": "2013-07-25T18:57:34+00:00",
            "Duration": 15,
            "InstanceIds": [
                "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
                "9e588a25-35b2-4804-bd43-488f85ebe5b7"
            ]
        },
        {
            "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
            "Status": "successful",
            "CompletedAt": "2013-07-25T18:56:41+00:00",
            "IamUserArn": "arn:aws:iam::444455556666:user/example-user",
            "DeploymentId": "19d3121e-d949-4ff2-9f9d-94eac087862a",
            "Command": {
                "Args": {},
                "Name": "deploy"
            },
            "InstanceIds": [
                "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
                "9e588a25-35b2-4804-bd43-488f85ebe5b7"
            ],
            "Duration": 72,
            "CreatedAt": "2013-07-25T18:55:29+00:00"
        }
    ]
}
```


列出堆疊的彈性 IP 地址 (describe-elastic-ips)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

使用 describe-elastic-ips 命令列出已向堆疊註冊的彈性 IP 地址，或取得特定彈性 IP 地址的詳細資訊。

```
aws opsworks --region us-west-2 describe-elastic-ips --instance-id b62f3e04-e9eb-436c-a91f-d9e9a396b7b0
```

上述命令會傳回一個 JSON 物件，其中包含指定之執行個體的每個彈性 IP 地址 (此範例中為一個) 的詳細資訊。如需每個參數的說明，請參閱 describe-elastic-ips。

```
{
  "ElasticIps": [
      {
          "Ip": "192.0.2.0",
          "Domain": "standard",
          "Region": "us-west-2"
      }
  ]
}
```

列出堆疊的執行個體 (describe-instances)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

使用 describe-instances 命令來列出堆疊的執行個體或取得特定執行個體的詳細資訊。

```
C:\>aws opsworks --region us-west-2 describe-instances --stack-id 38ee91e2-abdc-4208-a107-0b7168b3cc7a
```

上述命令會傳回一個 JSON 物件，其中包含指定堆疊中之每個執行個體的詳細資訊。如需每個參數的說明，請參閱 describe-instances。

```
{
  "Instances": [
      {
          "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
```

```
            "SshHostRsaKeyFingerprint":
 "f4:3b:8e:27:1b:73:98:80:5d:d7:33:e2:b8:c8:8f:de",
            "Status": "stopped",
            "AvailabilityZone": "us-west-2a",
            "SshHostDsaKeyFingerprint":
 "e8:9b:c7:02:18:2a:bd:ab:45:89:21:4e:af:0b:07:ac",
            "InstanceId": "8c2673b9-3fe5-420d-9cfa-78d875ee7687",
            "Os": "Amazon Linux",
            "Hostname": "db-master1",
            "SecurityGroupIds": [],
            "Architecture": "x86_64",
            "RootDeviceType": "instance-store",
            "LayerIds": [
                "41a20847-d594-4325-8447-171821916b73"
            ],
            "InstanceType": "c1.medium",
            "CreatedAt": "2013-07-25T18:11:27+00:00"
        },
        {
            "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
            "SshHostRsaKeyFingerprint":
 "ae:3a:85:54:66:f3:ce:98:d9:83:39:1e:10:a9:38:12",
            "Status": "stopped",
            "AvailabilityZone": "us-west-2a",
            "SshHostDsaKeyFingerprint":
 "5b:b9:6f:5b:1c:ec:55:85:f3:45:f1:28:25:1f:de:e4",
            "InstanceId": "9e588a25-35b2-4804-bd43-488f85ebe5b7",
            "Os": "Amazon Linux",
            "Hostname": "tomcustom1",
            "SecurityGroupIds": [],
            "Architecture": "x86_64",
            "RootDeviceType": "instance-store",
            "LayerIds": [
                "e6cbcd29-d223-40fc-8243-2eb213377440"
            ],
            "InstanceType": "c1.medium",
            "CreatedAt": "2013-07-25T18:15:52+00:00"
        }
    ]
}
```

列出帳戶的堆疊 (describe-stacks)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

使用 describe-stacks 命令來列出帳戶的堆疊或取得特定堆疊的詳細資訊。

```
aws opsworks --region us-west-2 describe-stacks
```

上述命令會傳回一個 JSON 物件，其中包含帳戶中每個堆疊的詳細資訊 (此範例中為兩個堆疊)。如需每個參數的說明，請參閱 describe-stacks。

```
{
    "Stacks": [
        {
            "ServiceRoleArn": "arn:aws:iam::444455556666:role/aws-opsworks-service-
role",
            "StackId": "aeb7523e-7c8b-49d4-b866-03aae9d4fbcb",
            "DefaultRootDeviceType": "instance-store",
            "Name": "TomStack-sd",
            "ConfigurationManager": {
                "Version": "11.4",
                "Name": "Chef"
            },
            "UseCustomCookbooks": true,
            "CustomJson": "{\n  \"tomcat\": {\n    \"base_version\": 7,\n
 \"java_opts\": \"-Djava.awt.headless=true -Xmx256m\"\n  },\n  \
"datasources\": {\n    \"ROOT\": \"jdbc/mydb\"\n  }\n}",
            "Region": "us-west-2",
            "DefaultInstanceProfileArn": "arn:aws:iam::444455556666:instance-profile/
aws-opsworks-ec2-role",
            "CustomCookbooksSource": {
                "Url": "git://github.com/example-repo/tomcustom.git",
                "Type": "git"
            },
            "DefaultAvailabilityZone": "us-west-2a",
            "HostnameTheme": "Layer_Dependent",
```

```
            "Attributes": {
                "Color": "rgb(45, 114, 184)"
            },
            "DefaultOs": "Amazon Linux",
            "CreatedAt": "2013-08-01T22:53:42+00:00"
        },
        {
            "ServiceRoleArn": "arn:aws:iam::444455556666:role/aws-opsworks-service-
role",
            "StackId": "40738975-da59-4c5b-9789-3e422f2cf099",
            "DefaultRootDeviceType": "instance-store",
            "Name": "MyStack",
            "ConfigurationManager": {
                "Version": "11.4",
                "Name": "Chef"
            },
            "UseCustomCookbooks": false,
            "Region": "us-west-2",
            "DefaultInstanceProfileArn": "arn:aws:iam::444455556666:instance-profile/
aws-opsworks-ec2-role",
            "CustomCookbooksSource": {},
            "DefaultAvailabilityZone": "us-west-2a",
            "HostnameTheme": "Layer_Dependent",
            "Attributes": {
                "Color": "rgb(45, 114, 184)"
            },
            "DefaultOs": "Amazon Linux",
            "CreatedAt": "2013-10-25T19:24:30+00:00"
        }
    ]
}
```


列出堆疊的 Layer (describe-layers)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

使用 describe-layers 命令來列出堆疊的 layer 或取得特定 layer 的詳細資訊。

```
aws opsworks --region us-west-2 describe-layers --stack-id 38ee91e2-abdc-4208-
a107-0b7168b3cc7a
```

上述命令會傳回 JSON 物件，其中包含指定堆疊中每個 layer 的詳細資訊，在此範例中為 MySQL layer 和自訂 layer。如需每個參數的說明，請參閱 describe-layers。

```
{
  "Layers": [
      {
          "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
          "Type": "db-master",
          "DefaultSecurityGroupNames": [
              "AWS-OpsWorks-DB-Master-Server"
          ],
          "Name": "MySQL",
          "Packages": [],
          "DefaultRecipes": {
              "Undeploy": [],
              "Setup": [
                  "opsworks_initial_setup",
                  "ssh_host_keys",
                  "ssh_users",
                  "mysql::client",
                  "dependencies",
                  "ebs",
                  "opsworks_ganglia::client",
                  "mysql::server",
                  "dependencies",
                  "deploy::mysql"
              ],
              "Configure": [
                  "opsworks_ganglia::configure-client",
                  "ssh_users",
                  "agent_version",
                  "deploy::mysql"
              ],
              "Shutdown": [
                  "opsworks_shutdown::default",
                  "mysql::stop"
              ],
              "Deploy": [
                  "deploy::default",
                  "deploy::mysql"
```

```
            ]
        },
        "CustomRecipes": {
            "Undeploy": [],
            "Setup": [],
            "Configure": [],
            "Shutdown": [],
            "Deploy": []
        },
        "EnableAutoHealing": false,
        "LayerId": "41a20847-d594-4325-8447-171821916b73",
        "Attributes": {
            "MysqlRootPasswordUbiquitous": "true",
            "RubygemsVersion": null,
            "RailsStack": null,
            "HaproxyHealthCheckMethod": null,
            "RubyVersion": null,
            "BundlerVersion": null,
            "HaproxyStatsPassword": null,
            "PassengerVersion": null,
            "MemcachedMemory": null,
            "EnableHaproxyStats": null,
            "ManageBundler": null,
            "NodejsVersion": null,
            "HaproxyHealthCheckUrl": null,
            "MysqlRootPassword": "*****FILTERED*****",
            "GangliaPassword": null,
            "GangliaUser": null,
            "HaproxyStatsUrl": null,
            "GangliaUrl": null,
            "HaproxyStatsUser": null
        },
        "Shortname": "db-master",
        "AutoAssignElasticIps": false,
        "CustomSecurityGroupIds": [],
        "CreatedAt": "2013-07-25T18:11:19+00:00",
        "VolumeConfigurations": [
            {
                "MountPoint": "/vol/mysql",
                "Size": 10,
                "NumberOfDisks": 1
            }
        ]
    },
```

```
    {
        "StackId": "38ee91e2-abdc-4208-a107-0b7168b3cc7a",
        "Type": "custom",
        "DefaultSecurityGroupNames": [
            "AWS-OpsWorks-Custom-Server"
        ],
        "Name": "TomCustom",
        "Packages": [],
        "DefaultRecipes": {
            "Undeploy": [],
            "Setup": [
                "opsworks_initial_setup",
                "ssh_host_keys",
                "ssh_users",
                "mysql::client",
                "dependencies",
                "ebs",
                "opsworks_ganglia::client"
            ],
            "Configure": [
                "opsworks_ganglia::configure-client",
                "ssh_users",
                "agent_version"
            ],
            "Shutdown": [
                "opsworks_shutdown::default"
            ],
            "Deploy": [
                "deploy::default"
            ]
        },
        "CustomRecipes": {
            "Undeploy": [],
            "Setup": [
                "tomcat::setup"
            ],
            "Configure": [
                "tomcat::configure"
            ],
            "Shutdown": [],
            "Deploy": [
                "tomcat::deploy"
            ]
        },
```

```
            "EnableAutoHealing": true,
            "LayerId": "e6cbcd29-d223-40fc-8243-2eb213377440",
            "Attributes": {
                "MysqlRootPasswordUbiquitous": null,
                "RubygemsVersion": null,
                "RailsStack": null,
                "HaproxyHealthCheckMethod": null,
                "RubyVersion": null,
                "BundlerVersion": null,
                "HaproxyStatsPassword": null,
                "PassengerVersion": null,
                "MemcachedMemory": null,
                "EnableHaproxyStats": null,
                "ManageBundler": null,
                "NodejsVersion": null,
                "HaproxyHealthCheckUrl": null,
                "MysqlRootPassword": null,
                "GangliaPassword": null,
                "GangliaUser": null,
                "HaproxyStatsUrl": null,
                "GangliaUrl": null,
                "HaproxyStatsUser": null
            },
            "Shortname": "tomcustom",
            "AutoAssignElasticIps": false,
            "CustomSecurityGroupIds": [],
            "CreatedAt": "2013-07-25T18:12:53+00:00",
            "VolumeConfigurations": []
        }
    ]
}
```

執行配方 (create-deployment)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

使用 create-deployment 命令來執行堆疊命令和部署命令。以下範例會執行堆疊命令來在指定堆疊上執行自訂配方。

```
aws opsworks --region us-west-1 create-deployment --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb
    --command "{\"Name\":\"execute_recipes\", \"Args\":{\"recipes\":[\"phpapp::appsetup
\"]}}"
```

command 引數接受 JSON 物件，其格式如下：

- Name - 指定命令名稱。此範例中使用的 execute_recipes 命令會在堆疊的執行個體上執行指定配方。
- Args - 指定引數清單和其值。此範例有一個引數 (recipes)，已設為要執行的配方 (phpapp::appsetup)。

請注意，JSON 物件中的 " 字元全部都已逸出。否則，命令可能傳回錯誤，顯示 JSON 無效。

命令會傳回部署 ID，您可以用以識別其他 CLI 命令 (例如 describe-commands) 的命令。

```
{
    "DeploymentId": "5cbaa7b9-4e09-4e53-aa1b-314fbd106038"
}
```

安裝相依性 (create-deployment)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

使用 create-deployment 命令來執行堆疊命令和部署命令。以下範例會執行 update_dependencies 堆疊命令來更新堆疊執行個體上的依存項目。

```
aws opsworks --region us-west-1 create-deployment --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb
--command "{\"Name\":\"install_dependencies\"}"
```

command 引數接受帶有 Name 參數的 JSON 物件，該參數會指定命令名稱。此範例中為 install_dependencies。請注意，JSON 物件中的 " 字元全部都已逸出。否則，命令可能傳回錯誤，顯示 JSON 無效。

命令會傳回部署 ID，您可以用以識別其他 CLI 命令 (例如 describe-commands) 的命令。

```
{
    "DeploymentId": "aef5b255-8604-4928-81b3-9b0187f962ff"
}
```

更新堆疊組態 (update-stack)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

使用 update-stack 命令更新指定堆疊的組態。以下範例會更新堆疊，以將自訂 JSON 新增到堆疊組態屬性。

```
aws opsworks --region us-west-1 update-stack --stack-id 935450cc-61e0-4b03-
a3e0-160ac817d2bb
   --custom-json "{\"somekey\":\"somevalue\"}" --service-role-arn
 arn:aws:iam::444455556666:role/aws-opsworks-service-role
```

請注意，JSON 物件中的 " 字元全部都已逸出。否則，命令可能傳回錯誤，顯示 JSON 無效。

> **Note**
>
> 範例也指定了堆疊的服務角色。您必須將 service-role-arn 設為有效的服務角色 ARN，否則動作將會失敗。未有預設值。您可以指定目前堆疊的服務角色 ARN (若您偏好的話)，但您必須清楚明確地指定。

update-stack 命令不會傳回值。

偵錯和故障診斷指南

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

如果您需要除錯配方或故障診斷服務問題，最佳的方法通常是遵循以下步驟依序操作：

1. 檢查常見的除錯和故障診斷問題是否有您的特定問題。
2. 搜尋 AWS OpsWorks Stacks 論壇，查看是否已討論過這些問題。

   論壇包含許多經驗豐富的使用者，並由 AWS OpsWorks Stacks 團隊監控。
3. 對於配方的問題，請參閱除錯配方。
4. 聯絡 AWS OpsWorks Stacks 支援或在 AWS OpsWorks Stacks 論壇上張貼您的問題。

下節提供除錯配方的指導方針。最終節說明常見的除錯和故障診斷問題及其解決方案。

> **Note**
>
> 每個 Chef 執行都會產生日誌，詳細描述回合的狀況，是寶貴的故障診斷資源。若要指定日誌中詳細資訊的數量，請將 `Chef::Log.level` 陳述式新增至指定所需日誌層級的自訂配方。預設值為 `:info`。以下範例說明如何將 Chef 日誌層級設定為 `:debug`，提供最詳細的回合描述。
>
> ```
> Chef::Log.level = :debug
> ```
>
> 如需檢視和解釋 Chef 日誌的詳細資訊，請參閱 Chef 日誌。

主題

- 除錯配方
- 常見的除錯和故障診斷問題

除錯配方

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

當生命週期事件發生，或您執行執行配方堆疊命令時， AWS OpsWorks Stacks 會向代理程式發出命令，在指定的執行個體上初始化 Chef Solo 執行，以執行適當的配方，包括您的自訂配方。本節說明一些您可以為失敗配方除錯的方法。

主題

- 登入故障的執行個體
- Chef 日誌
- 使用 AWS OpsWorks Stacks Agent CLI

登入故障的執行個體

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

如果配方失敗，執行個體將會以 `setup_failed` 狀態結束，而不是線上狀態。即使執行個體在 Stacks AWS OpsWorks 方面並不在線上，EC2 執行個體仍在執行中，登入對問題進行故障診斷通常很有用。例如，您可以檢查應用程式或自訂技術指南是否已正確安裝。Stacks AWS OpsWorks 內建的 SSH 和 RDP 登入支援僅適用於處於線上狀態的執行個體。不過，如果您已為執行個體指派 SSH 金鑰對，您仍可登入，說明如下：

- Linux 執行個體 – 使用 SSH 金鑰對的私有金鑰來登入第三方 SSH 用戶端，例如 OpenSSH 或 PuTTY。

  您可以使用 EC2 金鑰對或您的個人 SSH 金鑰對登入。

- Windows 執行個體 – 使用 EC2 金鑰對的私有金鑰來擷取執行個體的管理員密碼。

  使用該密碼透過您偏好的 RDP 用戶端登入。如需詳細資訊，請參閱以管理員身分登入。您無法使用個人 SSH 金鑰對擷取管理員密碼。

Chef 日誌

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Chef 日誌是您的其中一個關鍵故障診斷資源，特別是用於除錯配方。 AWS OpsWorks Stacks 會擷取每個命令的 Chef 日誌，並保留執行個體最近 30 個命令的日誌。由於執行處於除錯模式，因此日誌內含 Chef 執行的詳細說明，包括傳送到 `stdout` 和 `stderror` 的文字。如果配方失敗，日誌會包含 Chef 堆疊追蹤。

AWS OpsWorks Stacks 提供數種檢視 Chef 日誌的方式。一旦您擁有日誌資訊，就可用其為失敗配方除錯。

> **Note**
>
> 您也可以透過使用 SSH 連線到執行個體並執行代理程式 CLI `show_log` 命令，來檢視指定日誌的結尾。如需詳細資訊，請參閱顯示 Chef 日誌。

主題

- 使用主控台檢視 Chef 日誌
- 使用 CLI 或 API 檢視 Chef 日誌
- 在執行個體上檢視 Chef 日誌
- 解讀 Chef 日誌
- 常見的 Chef 日誌錯誤

使用主控台檢視 Chef 日誌

檢視 Chef 日誌最簡單的方法是前往執行個體的詳細資訊頁面。Logs (日誌) 區段包含每個事件和執行配方命令的進入點。下圖為執行個體的 Logs (日誌) 區段，顯示的內容是 configure (設定) 和 setup (安裝) 命令的日誌，兩者分別對應到設定和安裝生命週期事件。

按一下適當命令日誌欄中的顯示，以檢視對應的 Chef 日誌。如果發生錯誤， AWS OpsWorks Stacks 會自動將日誌開啟為錯誤，通常位於檔案結尾。

使用 CLI 或 API 檢視 Chef 日誌

您可以使用 AWS OpsWorks Stacks CLI `describe-commands`命令或 `DescribeCommands` API 動作來檢視存放在 Amazon S3 儲存貯體上的日誌。以下說明如何使用 CLI 檢視指定執行個體目前任一組日誌檔案。使用 `DescribeCommands` 的程序基本上很類似。

使用 AWS OpsWorks Stacks 來檢視執行個體的 Chef 日誌

1. 開啟執行個體的詳細資訊頁面，並複製其 OpsWorks ID 值。
2. 使用此 ID 值執行 `describe-commands` CLI 命令，如下所示：

```
aws opsworks describe-commands --instance-id 67bf0da2-29ed-4217-990c-d895d51812b9
```

命令會針對 AWS OpsWorks Stacks 在執行個體上執行的每個命令，傳回具有內嵌物件的 JSON 物件，並以最新的第一。`Type` 參數包含每個內嵌物件的命令類型，在此範例中為 `configure` 命令和 `setup` 命令。

```
{
    "Commands": [
        {
            "Status": "successful",
            "CompletedAt": "2013-10-25T19:38:36+00:00",
            "InstanceId": "67bf0da2-29ed-4217-990c-d895d51812b9",
```

```
            "AcknowledgedAt": "2013-10-25T19:38:24+00:00",
            "LogUrl": "https://s3.amazonaws.com/prod_stage-log/logs/
b6c402df-5c23-45b2-a707-ad20b9c5ae40?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&Expires=1382731518&Signature=YkqS5IZN2P4wixjHwoC3aCMbn5s%3D&response-cache-
control=private&response-content-encoding=gzip&response-content-
type=text%2Fplain",
            "Type": "configure",
            "CommandId": "b6c402df-5c23-45b2-a707-ad20b9c5ae40",
            "CreatedAt": "2013-10-25T19:38:11+00:00",
            "ExitCode": 0
        },
        {
            "Status": "successful",
            "CompletedAt": "2013-10-25T19:31:08+00:00",
            "InstanceId": "67bf0da2-29ed-4217-990c-d895d51812b9",
            "AcknowledgedAt": "2013-10-25T19:29:01+00:00",
            "LogUrl": "https://s3.amazonaws.com/prod_stage-log/logs/2a90e862-
f974-42a6-9342-9a4f03468358?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE
&Expires=1382731518&Signature=cxKYHO8mCCd4MvOyFb6ywebeQtA%3D&response-cache-
control=private&response-content-encoding=gzip&response-content-
type=text%2Fplain",
            "Type": "setup",
            "CommandId": "2a90e862-f974-42a6-9342-9a4f03468358",
            "CreatedAt": "2013-10-25T19:26:01+00:00",
            "ExitCode": 0
        }
    ]
}
```


3. 將 LogUrl 值複製到瀏覽器以檢視日誌。

如果執行個體具有多個命令，您可以將參數新增至 describe-commands，篩選出哪些命令應包含在回應物件中。如需詳細資訊，請參閱 describe-commands。

在執行個體上檢視 Chef 日誌

> **Note**
>
> 本節中的主題適用於 Chef 12。如需 Chef 11.10 和較舊版 Chef 日誌位置的資訊，請參閱為 Linux 的 Chef 11.10 和更舊版本進行故障診斷。

Linux 執行個體

AWS OpsWorks Stacks 會將每個執行個體的 Chef 日誌存放在其/var/chef/runs目錄中。(對於 Linux 執行個體，此目錄也包含相關聯的資料包，其以 JSON 格式的檔案存放)。您需要 sudo 權限才能存取此目錄。每個執行的日誌位於個別執行子目錄中名為 chef.log 的檔案。

AWS OpsWorks Stacks 會將其內部日誌存放在執行個體的/var/log/aws/opsworks資料夾中。此資訊通常對故障診斷的幫助不大。不過，這些日誌對於 AWS OpsWorks Stacks 支援很有用，如果您遇到服務問題，可能會請您提供這些日誌。Linux 日誌有時也可提供有用的故障診斷資料。

Windows 執行個體

代理程式日誌

在 Windows 執行個體上，OpsWorks 日誌存放於 ProgramData 路徑，如下所示。數字會包含時間戳記。

```
C:\ProgramData\OpsWorksAgent\var\logs\number
```

> **Note**
>
> 根據預設，ProgramData 是隱藏的資料夾。若要將其取消隱藏，請導覽至 Folder Options (資料夾選項)。在 View (檢視) 下，選擇顯示隱藏檔案的選項。

下列範例示範 Windows 執行個體上的代理程式日誌。

```
Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---         5/24/2015  11:59 PM     127277 command.20150524.txt
-a---         5/25/2015  11:59 PM     546772 command.20150525.txt
-a---         5/26/2015  11:59 PM     551514 command.20150526.txt
-a---         5/27/2015   9:43 PM     495181 command.20150527.txt
-a---         5/24/2015  11:59 PM      24353 keepalive.20150524.txt
-a---         5/25/2015  11:59 PM     106232 keepalive.20150525.txt
-a---         5/26/2015  11:59 PM     106208 keepalive.20150526.txt
-a---         5/27/2015   8:54 PM      92593 keepalive.20150527.txt
-a---         5/24/2015   7:19 PM       3891 service.20150524.txt
-a---         5/27/2015   8:54 PM       1493 service.20150527.txt
-a---         5/24/2015  11:59 PM     112549 wire.20150524.txt
```

```
-a---         5/25/2015  11:59 PM     501501 wire.20150525.txt
-a---         5/26/2015  11:59 PM     499640 wire.20150526.txt
-a---         5/27/2015   8:54 PM     436870 wire.20150527.txt
```

Chef 日誌

在 Windows 執行個體上，Chef 日誌存放於 `ProgramData` 路徑，如下所示。數字會包含時間戳記。

```
C:\ProgramData\OpsWorksAgent\var\commands\number
```

> **Note**
>
> 此目錄僅包含第一個 (OpsWorks 擁有的) Chef 執行的輸出。

下列範例示範 Windows 執行個體上 OpsWorks 擁有的 Chef 日誌。

```
 Mode                 LastWriteTime             Name
 ----                 -------------             ----
 d----         5/24/2015   7:23 PM
 configure-7ecb5f47-7626-439b-877f-5e7cb40ab8be
 d----         5/26/2015   8:30 PM             configure-8e74223b-d15d-4372-aeea-
a87b428ffc2b
 d----         5/24/2015   6:34 PM             configure-
c3980a1c-3d08-46eb-9bae-63514cee194b
 d----         5/26/2015   8:32 PM             grant_remote_access-70dbf834-1bfa-4fce-
b195-e50e85402f4c
 d----         5/26/2015  10:30 PM             revoke_remote_access-1111fce9-843a-4b27-
b93f-ecc7c5e9e05b
 d----         5/24/2015   7:21 PM             setup-754ec063-8b60-4cd4-
b6d7-0e89d7b7aa78
 d----         5/26/2015   8:27 PM             setup-af5bed36-5afd-4115-
af35-5766f88bc039
 d----         5/24/2015   6:32 PM             setup-d8abeffa-24d4-414b-
bfb1-4ad07319f358
 d----         5/24/2015   7:13 PM             shutdown-c7130435-9b5c-4a95-
be17-6b988fc6cf9a
 d----         5/26/2015   8:25 PM
 sync_remote_users-64c79bdc-1f6f-4517-865b-23d2def4180c
 d----         5/26/2015   8:48 PM
 update_custom_cookbooks-2cc59a94-315b-414d-85eb-2bdea6d76c6a
```

使用者 Chef 日誌

您可以在名為 `logfile.txt` 的檔案中找到您 Chef 執行的日誌，該檔案則位於以編號之 Chef 命令命名的資料夾中，如下圖所示。

C:/chef └── runs └── command-12345 ├── attribs.json ├── client.rb └── logfile.txt

解讀 Chef 日誌

日誌的開頭主要包含內部 Chef 記錄。

```
# Logfile created on Thu Oct 17 17:25:12 +0000 2013 by logger.rb/1.2.6
[2013-10-17T17:25:12+00:00] INFO: *** Chef 11.4.4 ***
[2013-10-17T17:25:13+00:00] DEBUG: Building node object for php-app1.localdomain
[2013-10-17T17:25:13+00:00] DEBUG: Extracting run list from JSON attributes provided on
 command line
[2013-10-17T17:25:13+00:00] INFO: Setting the run_list to
 ["opsworks_custom_cookbooks::load", "opsworks_custom_cookbooks::execute"] from JSON
[2013-10-17T17:25:13+00:00] DEBUG: Applying attributes from json file
[2013-10-17T17:25:13+00:00] DEBUG: Platform is amazon version 2013.03
[2013-10-17T17:25:13+00:00] INFO: Run List is [recipe[opsworks_custom_cookbooks::load],
 recipe[opsworks_custom_cookbooks::execute]]
[2013-10-17T17:25:13+00:00] INFO: Run List expands to [opsworks_custom_cookbooks::load,
 opsworks_custom_cookbooks::execute]
[2013-10-17T17:25:13+00:00] INFO: Starting Chef Run for php-app1.localdomain
[2013-10-17T17:25:13+00:00] INFO: Running start handlers
[2013-10-17T17:25:13+00:00] INFO: Start handlers complete.
[2013-10-17T17:25:13+00:00] DEBUG: No chefignore file found at /opt/aws/opsworks/
releases/20131015111601_209/cookbooks/chefignore no files will be ignored
[2013-10-17T17:25:13+00:00] DEBUG: Cookbooks to compile: ["gem_support", "packages",
 "opsworks_bundler", "opsworks_rubygems", "ruby", "ruby_enterprise", "dependencies",
 "opsworks_commons", "scm_helper", :opsworks_custom_cookbooks]
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook gem_support's library file: /
opt/aws/opsworks/releases/20131015111601_209/cookbooks/gem_support/libraries/
current_gem_version.rb
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook packages's library file: /opt/aws/
opsworks/releases/20131015111601_209/cookbooks/packages/libraries/packages.rb
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook dependencies's library file: /
opt/aws/opsworks/releases/20131015111601_209/cookbooks/dependencies/libraries/
current_gem_version.rb
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook opsworks_commons's library file: /
opt/aws/opsworks/releases/20131015111601_209/cookbooks/opsworks_commons/libraries/
activesupport_blank.rb
```

```
[2013-10-17T17:25:13+00:00] DEBUG: Loading cookbook opsworks_commons's library file: /
opt/aws/opsworks/releases/20131015111601_209/cookbooks/opsworks_commons/libraries/
monkey_patch_chefgem_resource.rb
...
```

檔案的這一部分對 Chef 專家來說非常有用。請注意，即使大多數命令包含更多配方，執行清單僅含兩個配方。這兩個配方處理載入和執行其他所有內建和自訂配方的任務。

檔案最有趣的部分通常在結尾。如果執行成功結束，您應該會看到類似以下的內容：

```
...
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: STDERR:
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: ---- End output of /sbin/service mysqld
 restart ----
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: Ran /sbin/service mysqld restart returned 0
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: service[mysql]: restarted successfully
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: Chef Run complete in 84.07096 seconds
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: cleaning the checksum cache
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: removing unused checksum cache file /var/chef/
cache/checksums/chef-file--tmp-chef-rendered-template20130611-4899-8wef7e-0
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: removing unused checksum cache file /var/chef/
cache/checksums/chef-file--tmp-chef-rendered-template20130611-4899-1xpwyb6-0
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: removing unused checksum cache file /var/chef/
cache/checksums/chef-file--etc-monit-conf
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: Running report handlers
[Tue, 11 Jun 2013 16:00:50 +0000] INFO: Report handlers complete
[Tue, 11 Jun 2013 16:00:50 +0000] DEBUG: Exiting
```

> **Note**
>
> 您可以使用代理程式 CLI 在執行期間或結束後顯示日誌結尾。如需詳細資訊，請參閱顯示 Chef 日誌。

如果配方失敗，您應該尋找 ERROR 層級的輸出，該輸出將包含例外狀況，並接續著 Chef 堆疊追蹤，如下所示：

```
...
Please report any problems with the /usr/scripts/mysqlbug script!
```

```
[  OK  ]
MySQL Daemon failed to start.
Starting mysqld:  [FAILED]STDERR: 130611 15:07:55 [Warning] The syntax '--log-slow-
queries' is deprecated and will be removed in a future release. Please use '--slow-
query-log'/'--slow-query-log-file' instead.
130611 15:07:56 [Warning] The syntax '--log-slow-queries' is deprecated and will be
 removed in a future release. Please use '--slow-query-log'/'--slow-query-log-file'
 instead.
---- End output of /sbin/service mysqld start ----

/opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/mixin/command.rb:184:in `handle_command_failures'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/mixin/command.rb:131:in `run_command'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/provider/service/init.rb:37:in `start_service'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/provider/service.rb:60:in `action_start'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource.rb:406:in `send'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource.rb:406:in `run_action'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/runner.rb:53:in `run_action'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/runner.rb:89:in `converge'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/runner.rb:89:in `each'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/runner.rb:89:in `converge'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection.rb:94:in `execute_each_resource'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection/stepable_iterator.rb:116:in `call'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection/stepable_iterator.rb:116:in
 `call_iterator_block'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection/stepable_iterator.rb:85:in `step'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection/stepable_iterator.rb:104:in `iterate'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection/stepable_iterator.rb:55:in
 `each_with_index'
```

```
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/resource_collection.rb:92:in `execute_each_resource'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/runner.rb:84:in `converge'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/client.rb:268:in `converge'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/client.rb:158:in `run'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/application/solo.rb:190:in `run_application'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/application/solo.rb:181:in `loop'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/application/solo.rb:181:in `run_application'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/../lib/chef/application.rb:62:in `run'
  /opt/aws/opsworks/releases/20130605160141_122/vendor/bundle/ruby/1.8/gems/
chef-0.9.15.5/bin/chef-solo:25
  /opt/aws/opsworks/current/bin/chef-solo:16:in `load'
  /opt/aws/opsworks/current/bin/chef-solo:16
```

檔案的結尾是 Chef 堆疊追蹤。您也應該檢查例外狀況之前的輸出，通常包含系統錯誤 (例如 `package not available`)，這在故判斷失敗原因時也很有用。在此範例中，MySQL 協助程式無法啟動。

常見的 Chef 日誌錯誤

下列是一些常見的 Chef 日誌錯誤及其解決方法。

找不到日誌

在 Chef 執行開始時，執行個體會收到預先簽章的 Amazon S3 URL，可讓您在 Chef 執行完成時在網頁上檢視日誌。由於此 URL 會在兩個小時後過期，因此如果 Chef 執行需要超過兩個小時，即使 Chef 執行期間沒有發生任何問題，也不會將日誌上傳到 Amazon S3 網站。建立日誌的命令會成功，但僅可在執行個體上檢視日誌，而不能在預先簽章的 URL 上檢視。

日誌意外結束

如果 Chef 日誌意外結束，並未指出成功或顯示錯誤資訊，您可能發生記憶體低下的狀態，讓 Chef 無法完成日誌。最佳選項就是使用更大的執行個體重試。

缺少技術指南或配方

如果 Chef 執行發生技術指南或配方不在技術指南快取內的情況，您將看到類似以下的內容：

```
DEBUG: Loading Recipe mycookbook::myrecipe via include_recipe
ERROR: Caught exception during execution of custom recipe: mycookbook::myrecipe:
   Cannot find a cookbook named mycookbook; did you forget to add metadata to a
 cookbook?
```

此項目指出 mycookbook 技術指南不在技術指南快取內。使用 Chef 11.4 時，如果您未在 metadata.rb 中正確宣告相依性，也可能發生此錯誤。

AWS OpsWorks Stacks 會從執行個體的技術指南快取執行配方。其會在執行個體啟動時，將儲存庫中的技術指南下載到此快取中。不過，如果您後續修改儲存庫中的技術指南， AWS OpsWorks Stacks 不會自動更新線上執行個體上的快取。如果您在執行個體啟動後，修改技術指南或新增技術指南，請執行以下步驟：

1. 確定您已將變更遞交到儲存庫。
2. 執行更新技術指南堆疊命令，以使用儲存庫中的最新版本來更新技術指南快取。

本機命令失敗

如果 Chef execute 資源無法執行指定命令，您將會看到類似以下的內容：

```
DEBUG: ---- End output of ./configure --with-config-file-path=/ returned 2
ERROR: execute[PHP: ./configure] (/root/opsworks-agent/site-cookbooks/php-fpm/
recipes/install.rb line 48) had an error:
   ./configure --with-config-file-path=/
```

向上捲動日誌，您應該會看到命令的 stderr 和 stdout 輸出，這有助您判斷命令失敗的原因。

套件失敗

如果套件安裝失敗，您將會看到類似以下的內容：

```
ERROR: package[zend-server-ce-php-5.3] (/root/opsworks-agent/site-cookbooks/
zend_server/recipes/install.rb line 20)
   had an error: apt-get -q -y --force-yes install zend-server-ce-php-5.3=5.0.4+b17
 returned 100, expected 0
```

向上捲動日誌，您應該會看到命令的 STDOUT 和 STDERROR 輸出，這有助您判斷套件安裝失敗的原因。

使用 AWS OpsWorks Stacks Agent CLI

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 代理程式 CLI 只適用於 Linux 執行個體。

在每個線上執行個體上， AWS OpsWorks Stacks 都會安裝 代理程式，以與服務通訊。Stacks AWS OpsWorks 服務會依序傳送命令給代理程式，以執行任務，例如在生命週期事件發生時啟動執行個體上的 Chef 執行。在 Linux 執行個體上，代理程式會公開命令列界面 (CLI)，這對故障診斷非常有用。若要執行代理程式 CLI 命令，請使用 SSH 連線到執行個體。然後您可以執行代理程式 CLI 命令來執行各種任務，包括下列任務：

- 執行配方。
- 顯示 Chef 日誌。
- 顯示堆疊組態與部署 JSON。

如需如何設定與執行個體間 SSH 連線的詳細資訊，請參閱使用 SSH 登入。您也必須具有堆疊的 SSH 和 sudo 許可。

本節說明如何使用代理程式 CLI 進行故障診斷。如需詳細資訊和完整命令參考，請參閱 AWS OpsWorks Stacks 代理程式 CLI。

主題

- 執行配方
- 顯示 Chef 日誌
- 顯示堆疊組態與部署 JSON

執行配方

代理程式 CLI `run_command` 命令會指示代理程式重新執行其先前執行過的命令。最實用的故障診斷命令為 `setup`、`configure`、`deploy` 和 `undeploy`，並各自對應到一個生命週期事件。這些命令會指示代理程式起始 Chef 執行，以執行相關聯的配方。

> **Note**
>
> `run_command` 命令僅限執行與指定命令相關聯的配方群組，通常是與生命週期事件相關聯的配方。您無法用它執行特定配方。若要執行一或多個指定配方，請使用執行配方堆疊命令或同等的 CLI 或 API 動作 (`create-deployment` 和 `CreateDeployment`)。

`run_command` 命令在為自訂配方除錯方面相當有用，尤其是指派至安裝和設定生命週期事件的配方，這些事件並無法直接從主控台加以觸發。透過使用 `run_command`，您可以隨時視需要執行特定事件的配方，而無須啟動或停止執行個體。

> **Note**
>
> AWS OpsWorks Stacks 從執行個體的技術指南快取執行配方，而不是技術指南儲存庫。當執行個體啟動時， AWS OpsWorks Stacks 會將技術指南下載到此快取，但如果您後續修改技術指南，則不會自動更新線上執行個體上的快取。如果您在啟動執行個體後修改了技術指南，請務必執行更新技術指南堆疊命令堆疊命令，以使用儲存庫中的最新版本更新技術指南快取。

代理程式僅快取最新的命令。您可以執行 `list_commands` 來列出這些命令，這會傳回已快取命令及其執行時間的清單。

```
sudo opsworks-agent-cli list_commands
2013-02-26T19:08:26        setup
2013-02-26T19:12:01        configure
2013-02-26T19:12:05        configure
2013-02-26T19:22:12        deploy
```

若要重新執行最新的命令，請執行：

```
sudo opsworks-agent-cli run_command
```

若要重新執行最新執行個體的指定命令，請執行：

```
sudo opsworks-agent-cli run_command command
```

例如，若要重新執行安裝配方，您可以執行下列命令：

```
sudo opsworks-agent-cli run_command setup
```

每個命令都有一個相關聯的堆疊組態和部署 JSON，代表該命令執行時堆疊和部署的狀態。由於該資料可能隨著命令不同有所差異，因此較舊執行個體的命令可能使用與最新執行個體不同的資料。若要重新執行特定執行個體的命令，請複製 `list_commands` 輸出中的時間，並執行以下命令：

```
sudo opsworks-agent-cli run_command time
```

上述範例均使用預設 JSON 重新執行命令，也就是為該命令安裝的 JSON。您可以對任意 JSON 檔案重新執行命令，如下所示：

```
sudo opsworks-agent-cli run_command -f /path/to/valid/json.file
```

顯示 Chef 日誌

代理程式 CLI `show_log` 命令會顯示指定的日誌。在命令完成後，您將會看到檔案結尾。因此，`show_log` 命令提供方便查看日誌結尾的方法，您通常會在這裡找到錯誤資訊。您可以向上捲動以查看日誌的先前部分。

若要顯示目前命令的日誌，請執行：

```
sudo opsworks-agent-cli show_log
```

您也可以顯示特定命令的日誌，但請注意，代理程式僅快取最後三十個命令的日誌。您可以執行 `list_commands` 列出執行個體的命令，這會傳回已快取命令及其執行時間的清單。如需範例，請參閱「執行配方」。

若要顯示特定命令最新執行狀況的日誌，請執行下列命令：

```
sudo opsworks-agent-cli show_log command
```

命令參數可以設為 `setup`、`configure`、`deploy`、`undeploy`、`start`、`stop` 或 `restart`。這些命令大部分都對應到生命週期事件，並指示代理程式執行相關聯配方。

若要顯示特定命令執行狀況的日誌，請複製 `list_commands` 輸出中的日期並執行：

```
sudo opsworks-agent-cli show_log date
```

如果命令仍在執行，`show_log` 會顯示日誌的目前狀態。

> **Note**
>
> 使用 `show_log` 來為錯誤和記憶體不足問題進行故障診斷的其中一種方法，是在執行期間顯示日誌的結尾，如下所示：
>
> 1. 使用 `run_command` 來觸發適當的生命週期事件。如需詳細資訊，請參閱執行配方。
> 2. 重複執行 `show_log` 以在寫入日誌時查看日誌的結尾。
>
> 如果 Chef 記憶體不足或意外結束，日誌會突然結束。如果配方失敗，日誌的結尾為例外狀況和堆疊追蹤。

顯示堆疊組態與部署 JSON

配方使用的許多資料來自堆疊組態和部署 JSON，這定義了一組 Chef 屬性，以提供堆疊組態、任何部署和使用者可新增之選用自訂屬性的詳細說明。對於每個命令， AWS OpsWorks Stacks 安裝的 JSON 代表命令 時的堆疊和部署狀態。如需詳細資訊，請參閱堆疊組態及部署屬性。

如果自訂配方從堆疊組態和部署 JSON 取得資料，您可以檢查該 JSON 來驗證資料。顯示堆疊組態和部署 JSON 最簡單的方式，是執行代理程式 CLI `get_json` 命令，這會顯示 JSON 物件的格式化版本。以下提供某些一般輸出的前幾行：

```
{
  "opsworks": {
    "layers": {
      "php-app": {
        "id": "4a2a56c8-f909-4b39-81f8-556536d20648",
        "instances": {
          "php-app2": {
            "elastic_ip": null,
            "region": "us-west-2",
            "booted_at": "2013-02-26T20:41:10+00:00",
            "ip": "10.112.235.192",
```

```
            "aws_instance_id": "i-34037f06",
            "availability_zone": "us-west-2a",
            "instance_type": "c1.medium",
            "private_dns_name": "ip-10-252-0-203.us-west-2.compute.internal",
            "private_ip": "10.252.0.203",
            "created_at": "2013-02-26T20:39:39+00:00",
            "status": "online",
            "backends": 8,
            "public_dns_name": "ec2-10-112-235-192.us-west-2.compute.amazonaws.com"
...
```

您可以顯示最新的堆疊組態和部署 JSON，如下所示：

```
sudo opsworks-agent-cli get_json
```

您可以執行下列命令來顯示指定命令的最新堆疊組態和部署 JSON：

```
sudo opsworks-agent-cli get_json command
```

命令參數可以設為 `setup`、`configure`、`deploy`、`undeploy`、`start`、`stop` 或 `restart`。這些命令大部分都對應到生命週期事件，並指示代理程式執行相關聯配方。

您可以指定命令的日期，來顯示特定命令執行的堆疊組態和部署 JSON，如下所示：

```
sudo opsworks-agent-cli get_json date
```

使用此命令的最簡單方式如下：

1. 執行 `list_commands` 會傳回清單，內含已在執行個體上執行的命令，以及每個命令的執行日期。
2. 複製適當命令的日期，並將其做為 `get_json` *`date`* 引數使用。

常見的除錯和故障診斷問題

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節說明一些常見的除錯和故障診斷問題，以及其解決方案。

主題

- 對 Stacks AWS OpsWorks 進行故障診斷
- 故障診斷執行個體註冊

對 Stacks AWS OpsWorks 進行故障診斷

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

本節包含一些常見的 AWS OpsWorks Stacks 問題及其解決方案。

主題

- 無法管理執行個體
- 在 Chef 執行之後，執行個體無法開機
- Layer 的執行個體皆未通過 Elastic Load Balancing 運作狀態檢查
- 無法與 Elastic Load Balancing Load Balancer 通訊
- 匯入的現場部署執行個體在重新啟動之後無法完成磁碟區設定
- EBS 磁碟區在重新開機後無法重新連接
- 無法刪除 AWS OpsWorks Stacks 安全群組
- 意外刪除 AWS OpsWorks Stacks 安全群組
- Chef 日誌無故終止
- 無法更新技術指南
- 執行個體停滯在開機中狀態
- 執行個體未預期的重新啟動
- opsworks-agent 程序在執行個體上執行
- 未預期的 execute_recipes 命令

無法管理執行個體

問題：您再也無法管理以前能管理的執行個體。在某些情況下，日誌可能會顯示類似下列內容的錯誤。

```
Aws::CharlieInstanceService::Errors::UnrecognizedClientException - The security token
 included in the request is invalid.
```

原因：此問題可能會在執行個體依存之 AWS OpsWorks 以外的資源遭到編輯或刪除時發生。以下是可能會中斷與執行個體通訊之資源變更的範例。

- 與執行個體相關聯的 IAM 使用者或角色在 Stacks AWS OpsWorks 之外被意外刪除。這會導致安裝在執行個體上的 AWS OpsWorks 代理程式與 AWS OpsWorks Stacks 服務之間的通訊失敗。與執行個體關聯的 使用者需要在執行個體的生命存續期間內存在。
- 在執行個體離線時編輯磁碟區或儲存體組態，可能會使執行個體無法管理。
- 手動將 EC2 執行個體新增至 ELB。每次執行個體進入或離開線上狀態時， AWS OpsWorks 請重新設定指派的 Elastic Load Balancing 負載平衡器。 AWS OpsWorks 只會將已知為有效成員的執行個體視為有效成員；在 外部新增的執行個體 AWS OpsWorks，或由其他程序新增的執行個體會遭到移除。每一個其他的執行個體都會遭到移除。

解決方案：請勿刪除執行個體所依賴的 IAM 使用者或角色。若可能的話，請只在依存執行個體執行時編輯磁碟區或儲存體組態。使用 AWS OpsWorks 管理 AWS OpsWorks 執行個體的負載平衡器或 EIP 成員資格。當您註冊執行個體時，為了防止在使用者意外刪除時管理已註冊執行個體的問題，請將 `--use-instance-profile` 參數新增至您的`register`命令，以改用執行個體的內建執行個體描述檔。

在 Chef 執行之後，執行個體無法開機

問題：於設定為使用自訂技術指南的 Chef 11.10 或較舊版本的堆疊上，在使用社群技術指南的 Chef 執行之後，執行個體無法開機。日誌訊息可能會表示配方編譯失敗 ("Recipe Compile Error")，或是因為找不到依存項目而無法載入。

原因：最有可能的原因是自訂或社群技術指南不支援堆疊所使用的 Chef 版本。一些熱門的社群技術指南，例如 `apt` 和 `build-essential` 與 Chef 11.10 有已知的相容性問題。

解決方案：在開啟使用自訂 Chef 技術指南設定的 AWS OpsWorks Stacks 堆疊上，自訂或社群技術指南必須一律支援堆疊使用的 Chef 版本。請將社群技術指南固定 (即將技術指南版本編號設為特定版本) 在與您在堆疊設定中設定之 Chef 版本相容的版本。若要尋找支援的社群技術指南版本，請檢視編譯失敗之技術指南的變更記錄，並只使用您堆疊可支援的最近版本。若要固定技術指南的版本，請在您自訂

技術指南儲存庫的 Berksfile 中指定明確的版本編號。例如：`cookbook 'build-essential', '= 3.2.0'`。

Layer 的執行個體皆未通過 Elastic Load Balancing 運作狀態檢查

問題：您將 Elastic Load Balancing 負載平衡器連接至應用程式伺服器層，但所有執行個體都無法通過運作狀態檢查。

原因：當您建立 Elastic Load Balancing 負載平衡器時，您必須指定負載平衡器呼叫的 ping 路徑，以判斷執行個體是否正常運作。請務必指定適合您應用程式的 ping 路徑。預設值為 /index.html。若您的應用程式不包含 `index.html`，您必須指定適當的路徑，否則運作狀態檢查將會失敗。例如，Chef 11 Linux 堆疊入門中使用的 SimplePHPApp 應用程式並未使用 index.html。那些伺服器的適當 ping 路徑為 /。

解決方案：編輯負載平衡器的 ping 路徑。如需詳細資訊，請參閱 Elastic Load Balancing

無法與 Elastic Load Balancing Load Balancer 通訊

問題：您建立 Elastic Load Balancing 負載平衡器並將其連接到應用程式伺服器層，但當您按一下負載平衡器的 DNS 名稱或 IP 地址來執行應用程式時，您會收到下列錯誤：「遠端伺服器沒有回應」。

原因：如果您的堆疊在預設 VPC 中執行，當您在區域中建立 Elastic Load Balancing 負載平衡器時，您必須指定安全群組。安全群組必須具備輸入規則，允許來自您 IP 地址的傳入流量。若您指定 default VPC security group (預設 VPC 安全群組)，預設的輸入規則不會接受任何傳入流量。

解決方案：編輯安全群組輸入規則，以接受來自適當 IP 地址的傳入流量。

1. 在 Amazon EC2 主控台的導覽窗格中按一下安全群組。
2. 選取負載平衡器的安全群組。
3. 按一下 Inbound (傳入) 標籤上的 Edit (編輯)。
4. 新增一條輸入規則，並將其中的 Source (來源) 設為適當的 CIDR。

   例如，指定 Anywhere (任何位置) 會將 CIDR 設為 0.0.0.0/0，即指示負載平衡器接受來自任何 IP 地址的傳入流量。

匯入的現場部署執行個體在重新啟動之後無法完成磁碟區設定

問題：您重新啟動已匯入 Stacks AWS OpsWorks 的 EC2 執行個體，且 AWS OpsWorks Stacks 主控台顯示失敗為執行個體狀態。這可能會在 Chef 11 或 Chef 12 執行個體上發生。

原因：AWS OpsWorks Stacks 於設定程序中可能無法將磁碟區連接至您的執行個體。其中一個可能的原因是 AWS OpsWorks Stacks 在您執行 `setup` 命令時覆寫了您執行個體上的磁碟區組態。

解決方案：開啟執行個體的 Details (詳細資訊) 頁面，檢查 Volumes (磁碟區) 區域內的磁碟區組態。請注意，您只能在您的執行個體處於 stopped (已停止) 狀態時才能變更您的磁碟區組態。請確認每個磁碟區都有指定的掛載點和名稱。在重新啟動執行個體之前，請確認您在 Stacks AWS OpsWorks 的組態中提供正確的掛載點。

EBS 磁碟區在重新開機後無法重新連接

問題：您使用 Amazon EC2 主控台將 Amazon EBS 磁碟區連接至執行個體，但當您重新啟動執行個體時，磁碟區不再連接。

原因： AWS OpsWorks Stacks 只能重新連接其已知的 Amazon EBS 磁碟區，這些磁碟區僅限於下列項目：

- Stacks AWS OpsWorks 建立的磁碟區。
- 來自您帳戶的磁碟區，且已藉 Resources (資源) 頁面明確向堆疊註冊。

解決方案：僅使用 Stacks 主控台、API 或 CLI AWS OpsWorks 來管理您的 Amazon EBS 磁碟區。如果您想要將其中一個帳戶的 Amazon EBS 磁碟區與堆疊搭配使用，請使用堆疊的資源頁面來註冊磁碟區並將其連接至執行個體。如需詳細資訊，請參閱資源管理。

無法刪除 AWS OpsWorks Stacks 安全群組

問題：刪除堆疊後，會留下許多無法刪除的 Stacks AWS OpsWorks 安全群組。

原因：安全群組必須依照特定順序進行刪除。

解決方案：首先，請確認沒有任何執行個體正在使用安全群組。接著，依照以下順序刪除下列任何安全群組 (若有的話)：

1. AWS-OpsWorks-Blank-Server
2. AWS-OpsWorks-Monitoring-Master-Server
3. AWS-OpsWorks-DB-Master-Server
4. AWS-OpsWorks-Memcached-Server
5. AWS-OpsWorks-Custom-Server
6. AWS-OpsWorks-nodejs-App-Server
7. AWS-OpsWorks-PHP-App-Server

8. AWS-OpsWorks-Rails-App-Server
9. AWS-OpsWorks-Web-Server
10. AWS-OpsWorks-Default-Server
11. AWS-OpsWorks-LB-Server

意外刪除 AWS OpsWorks Stacks 安全群組

問題：您已刪除其中一個 AWS OpsWorks Stacks 安全群組，需要重新建立。

原因：這些安全群組通常都是意外遭到刪除。

解決方案：重新建立的群組必須是和原始群組完全一模一樣的複本，其群組名稱也要有相同的大小寫。與其手動重新建立群組，建議的方法為讓 AWS OpsWorks Stacks 為您執行這項任務。只要在相同的 AWS 區域和 VPC 建立新的堆疊，如果有的話，Stacks AWS OpsWorks 就會自動重新建立所有內建安全群組，包括您刪除的安全群組。您接著可以刪除您不再需要使用的堆疊，安全群組仍會留下。

Chef 日誌無故終止

問題：Chef 日誌無故終止。日誌的最後沒有指出成功執行或顯示異常和堆疊追蹤。

原因：此行為通常是因為記憶體不足。

解決方案：建立更大的執行個體，並使用代理程式 CLI `run_command` 命令重新執行配方。如需詳細資訊，請參閱執行配方。

無法更新技術指南

問題：您更新您的技術指南，但堆疊的執行個體仍然執行舊的配方。

Cause： AWS OpsWorks Stacks 會在每個執行個體上快取技術指南，並從快取執行配方，而非儲存庫。當您啟動新的執行個體時， AWS OpsWorks Stacks 會將您的技術指南從儲存庫下載到執行個體的快取。但是，若您之後修改您的自訂技術指南， AWS OpsWorks Stacks 不會自動更新線上執行個體的快取。

解決方案：執行更新技術指南堆疊命令，明確指示 Stacks AWS OpsWorks 更新線上執行個體的技術指南快取。

執行個體停滯在開機中狀態

問題：當您重新啟動執行個體，或自動修復自動予以重新啟動時，啟動操作停滯在 `booting` 狀態。

原因：此問題其中一個可能的原因是 VPC 組態，包含預設 VPC。執行個體必須始終能夠與 AWS OpsWorks Stacks 服務、Amazon S3 和套件、技術指南和應用程式儲存庫進行通訊。例如，如果您從預設 VPC 移除預設閘道，執行個體會失去與 Stacks AWS OpsWorks 服務的連線。由於 AWS OpsWorks Stacks 無法再與執行個體代理程式通訊，因此會將執行個體視為失敗並自動修復。不過，如果沒有連線， AWS OpsWorks Stacks 就無法在已修復的執行個體上安裝執行個體代理程式。如果沒有代理程式， AWS OpsWorks Stacks 就無法在執行個體上執行設定配方，因此啟動操作無法繼續超過「開機」狀態。

解決方案：修改您的 VPC 組態，讓您的執行個體具備需要的連線能力。

執行個體未預期的重新啟動

問題：已停止的執行個體未預期的重新啟動。

原因 1：如果您已啟用執行個體的自動修復， AWS OpsWorks Stacks 會定期對相關聯的 Amazon EC2 執行個體執行運作狀態檢查，並重新啟動任何運作狀態不佳的執行個體。如果您使用 Amazon EC2 主控台、API 或 AWS OpsWorks CLI 來停止或終止 Stacks 受管執行個體，則不會通知 AWS OpsWorks Stacks。因此，它會將已停止的執行個體視為運作狀態不良，並自動重新啟動它。

解決方案：僅使用 AWS OpsWorks Stacks 主控台、API 或 CLI 管理您的執行個體。如果您使用 AWS OpsWorks Stacks 來停止或刪除執行個體，則不會重新啟動執行個體。如需詳細資訊，請參閱 手動啟動、停止和重新開機全年無休的執行個體 和 刪除 AWS OpsWorks Stacks 執行個體。

原因 2：執行個體可能會因為各種原因而故障。如果您已啟用自動修復， AWS OpsWorks Stacks 會自動重新啟動失敗的執行個體。

解決方案：這是正常操作；除非您不希望 Stacks AWS OpsWorks 重新啟動失敗的執行個體，否則不需要執行任何動作。在這種情況下，建議您停用自動修復。

`opsworks-agent` 程序在執行個體上執行

問題：您的執行個體上有數個 `opsworks-agent` 程序正在執行。例如：

```
aws 24543 0.0 1.3 172360 53332 ? S Feb24 0:29 opsworks-agent: master 24543
aws 24545 0.1 2.0 208932 79224 ? S Feb24 22:02 opsworks-agent: keep_alive of master
 24543
aws 24557 0.0 2.0 209012 79412 ? S Feb24 8:04 opsworks-agent: statistics of master
 24543
aws 24559 0.0 2.2 216604 86992 ? S Feb24 4:14 opsworks-agent: process_command of master
 24
```

原因：這些是維持代理程式正常操作所需的程序。他們會執行像是處理部署，以及將持續連線訊息傳送回服務等任務。

解決方案：這是正常行為。請不要停止這些程序，否則會影響代理程式的操作。

未預期的 execute_recipes 命令

問題：執行個體詳細資訊頁面上的 Logs (日誌) 區段包含未預期的 `execute_recipes` 命令。未預期的 `execute_recipes` 命令也會顯示在 Stack (堆疊) 和 Deployments (部署) 頁面上。

原因：此問題通常是因許可變更而造成。當您變更使用者或群組的 SSH 或 sudo 許可時， AWS OpsWorks Stacks 會執行 `execute_recipes` 來更新堆疊的執行個體，也會觸發設定事件。另一個 `execute_recipes` 命令的來源是 AWS OpsWorks Stacks，其會更新執行個體代理程式。

解決方案：這是正常操作，您不需要採取任何行動。

若要查看 `execute_recipes` 命令執行的動作是什麼，請前往 Deployments (部署) 頁面，然後按一下命令的時間戳記。這會開啟命令的詳細資訊頁面，列出執行的關鍵配方。例如，以下詳細資訊頁面是執行 `ssh_users` 以更新 SSH 許可的 `execute_recipes` 命令。

若要查看所有詳細資訊，請按一下命令的日誌欄中的顯示以顯示相關聯的 Chef 日誌。搜尋 **Run List**. AWS OpsWorks Stacks 維護配方的日誌將位於 下`OpsWorks Custom Run List`。例如，以下是在先前螢幕擷取畫面中顯示之 `execute_recipes` 命令的 run list，顯示每個與命令關聯的配方。

```
[2014-02-21T17:16:30+00:00] INFO: OpsWorks Custom Run List:
 ["opsworks_stack_state_sync",
  "ssh_users", "test_suite", "opsworks_cleanup"]
```

故障診斷執行個體註冊

本節包含一些常見的執行個體註冊問題及其解決方案。

> **Note**
>
> 若您有註冊問題，請搭配 `--debug` 引數執行 `register`，這會提供額外的除錯資訊。

主題

- EC2User 無權執行：...
- 登入資料應設定在有效區域

EC2User 無權執行：...

問題：`register` 命令傳回類似下列內容：

```
A client error (AccessDenied) occurred when calling the CreateGroup operation:
User: arn:aws:iam::123456789012:user/ImportEC2User is not authorized to
perform: iam:CreateGroup on resource:
arn:aws:iam::123456789012:group/AWS/OpsWorks/OpsWorks-b583ce55-1d01-4695-b3e5-
ee19257d1911
```

原因：`register`命令正在執行時，憑證未授予所需的許可。使用者的政策必須允許 `iam:CreateGroup` 動作及其他動作。

解決方案：提供 `register` 具有必要許可的 IAM 使用者登入資料。如需詳細資訊，請參閱安裝並設定 AWS CLI。

登入資料應設定在有效區域

問題：`register` 命令傳回下列內容：

```
A client error (InvalidSignatureException) occurred when calling the
DescribeStacks operation: Credential should be scoped to a valid region, not 'cn-
north-1'.
```

原因：命令的區域必須為有效的 AWS OpsWorks Stacks 區域。如需支援的區域的清單，請參閱區域支援。此錯誤通常會因下列其中一個原因發生：

- 堆疊位於不同的區域中，但您將堆疊的區域指派給命令的 `--region` 引數。

  您不需要指定堆疊區域； AWS OpsWorks Stacks 會自動從堆疊 ID 判斷。
- 您忽略了 `--region` 引數，其隱含指定預設區域，但您的預設區域不受 AWS OpsWorks Stacks 支援。

解決方案：明確`--region`設定為支援的 Stacks AWS OpsWorks 區域，或編輯 `config` 檔案 AWS CLI ，將預設區域變更為支援的 AWS OpsWorks Stacks 區域。如需詳細資訊，請參閱設定 AWS 命令列界面。

AWS OpsWorks Stacks 代理程式 CLI

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

> **Note**
>
> 此功能只適用於 Linux 執行個體。

AWS OpsWorks Stacks 在每個執行個體上安裝的代理程式會公開命令列界面 (CLI)。如果您使用 SSH 登入執行個體，則可以使用 CLI 執行下列作業：

- 存取 Chef 執行的日誌檔案。
- 存取 AWS OpsWorks Stacks 命令。
- 手動執行 Chef 配方。
- 檢視執行個體報告。
- 檢視代理程式報告。
- 檢視一組有限的堆疊組態和部署屬性。

> **Important**
>
> 您只能以 root 身分或透過使用 sudo 來執行代理程式 CLI 命令。

基本命令語法為：

```
sudo opsworks-agent-cli [--help] [command [activity] [date]]
```

四個引數如下：

help

(選用) 單獨使用時會顯示可用命令的概覽。搭配命令使用時，help 會顯示命令的描述。

command

(選用) 代理程式 CLI 命令，必須設為下列其中一項：

- agent_report
- get_json
- instance_report
- list_commands
- run_command
- show_log
- stack_state

activity

(選用) 做為某些命令的引數時可指定特定 AWS OpsWorks Stacks 活動：setup、configure、deploy、undeploy、start、stop 或 restart。

date

(選用) 做為某些命令的引數時可指定特定 AWS OpsWorks Stacks 命令執行。透過將 date 設為執行命令的時間戳記，並使用 *yyyy-mm-ddThh:mm:ss* 格式 (包括單引號)，即可指定命令執行。例如，對於 2013 年 2 月 5 日星期二 10:31:55，請使用：'2013-02-05T10:31:55'。若要判斷何時執行特定 AWS OpsWorks Stacks 命令，請執行 list_commands。

> **Note**
>
> 如果代理程式已多次執行相同的 AWS OpsWorks Stacks 活動，您可以指定活動及其執行時間來挑選特定的執行。如果您指定活動並省略時間，代理程式 CLI 命令會對活動的最近執行採取動作。如果您省略這兩個引數，代理程式 CLI 命令會對最近的活動採取動作。

下列各節說明命令及其相關聯的引數。為簡潔起見，語法部分會省略可搭配任何命令使用的選用 `--help` 選項。

主題

- agent_report
- get_json
- instance_report
- list_commands
- run_command
- show_log
- stack_state

agent_report

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

傳回代理程式報告。

```
sudo opsworks-agent-cli agent_report
```

下列輸出範例來自最近執行設定活動的執行個體。

```
$ sudo opsworks-agent-cli agent_report
```

```
AWS OpsWorks Instance Agent State Report:

  Last activity was a "configure" on 2015-12-01 18:19:23 UTC
  Agent Status: The AWS OpsWorks agent is running as PID 30998
  Agent Version: 4004-20151201152533, up to date
```

get_json

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

傳回 Chef 執行的相關資訊做為 JSON 物件。

```
sudo opsworks-agent-cli get_json [activity] [date] [-i | --internal | --no-i | --no-internal]
```

根據預設，`get_json` 會顯示客戶針對最近 Chef 執行所提供的資訊。使用下列選項來指定一組特定的資訊。

activity

顯示與最近指定活動相關聯之 Chef 執行的資訊。若要取得有效的活動清單，請執行 list_commands。

date

顯示與在指定時間戳記執行的活動相關聯之 Chef 執行的資訊。若要取得有效的時間戳記清單，請執行 list_commands。

-i, --internal

顯示 AWS OpsWorks Stacks 內部用於 Chef 執行的資訊。

--no-i, --no-internal

明確顯示客戶針對 Chef 執行所提供的資訊。如果未指定其他選項，這會是預設值。

> **Note**
>
> 若是 Chef 12 Linux 執行個體，執行此命令將傳回有效的資訊，例如執行個體的堆疊組態和部署屬性。不過，若要取得更完整的資訊，請參閱 AWS OpsWorks Stacks 在執行個體上建立的 Chef 資料包。如需詳細資訊，請參閱 AWS OpsWorks Stacks Data Bag 參考。

下列輸出範例會顯示客戶針對最近設定活動之 Chef 執行所提供的資訊。

```
$ sudo opsworks-agent-cli get_json configure

{
  "run_list": [
    "recipe[opsworks_cookbook_demo::configure]"
  ]
}
```

下列輸出範例顯示 AWS OpsWorks Stacks 在內部用於針對指定時間戳記執行之 Chef 執行的資訊。

```
$ sudo opsworks-agent-cli get_json 2015-12-01T18:20:24 -i

{
  "aws_opsworks_agent": {
    "version": "4004-20151201152533",
    "valid_client_activities": [
      "reboot",
      "stop",
      "deploy",
      "grant_remote_access",
      "revoke_remote_access",
      "update_agent",
      "setup",
      "configure",
      "update_dependencies",
      "install_dependencies",
      "update_custom_cookbooks",
      "execute_recipes",
      "sync_remote_users"
    ],
    "command": {
      "type": "configure",
      "args": {
```

```
        "app_ids": [

        ]
      },
      "sent_at": "2015-12-01T18:19:23+00:00",
      "command_id": "5c2113f3-c6d5-40eb-bcfa-77da2885eeEX",
      "iam_user_arn": null,
      "instance_id": "cfdaa716-42fe-4e3b-9762-fef184ddd8EX"
    },
    "resources": {
      "apps": [

      ],
      "layers": [
        {
          "layer_id": "93f50d83-1e73-45c4-840a-0d4f07cda1EX",
          "name": "MyCookbooksDemoLayer",
          "packages": [

          ],
          "shortname": "cookbooks-demo",
          "type": "custom",
          "volume_configurations": [

          ]
        }
      ],
      "instances": [
        {
          "ami_id": "ami-d93622EX",
          "architecture": "x86_64",
          "auto_scaling_type": null,
          "availability_zone": "us-west-2a",
          "created_at": "2015-11-18T00:21:05+00:00",
          "ebs_optimized": false,
          "ec2_instance_id": "i-a480e960",
          "elastic_ip": null,
          "hostname": "cookbooks-demo1",
          "instance_id": "cfdaa716-42fe-4e3b-9762-fef184ddd8EX",
          "instance_type": "c3.large",
          "layer_ids": [
            "93f50d83-1e73-45c4-840a-0d4f07cda1EX"
          ],
          "os": "Amazon Linux 2015.09",
```

```
          "private_dns": "ip-192-0-2-0.us-west-2.compute.internal",
          "private_ip": "10.122.69.33",
          "public_dns": "ec2-203-0-113-0.us-west-2.compute.amazonaws.com",
          "public_ip": "192.0.2.0",
          "root_device_type": "ebs",
          "root_device_volume_id": "vol-f6f7e8EX",
          "ssh_host_dsa_key_fingerprint": "f2:...:15",
          "ssh_host_dsa_key_public": "ssh-dss AAAAB3Nz...a8vMbqA=",
          "ssh_host_rsa_key_fingerprint": "0a:...:96",
          "ssh_host_rsa_key_public": "ssh-rsa AAAAB3Nz...yhPanvo7",
          "status": "online",
          "subnet_id": null,
          "virtualization_type": "paravirtual",
          "infrastructure_class": "ec2",
          "ssh_host_dsa_key_private": "-----BEGIN DSA PRIVATE KEY-----
\nMIIDVwIB...g5OtgQ==\n-----END DSA PRIVATE KEY-----\n",
          "ssh_host_rsa_key_private": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIB...78kprtIw\n-----END RSA PRIVATE KEY-----\n"
        }
      ],
      "users": [

      ],
      "elastic_load_balancers": [

      ],
      "rds_db_instances": [

      ],
      "stack": {
        "arn": "arn:aws:opsworks:us-west-2:80398EXAMPLE:stack/040c3def-b2b4-4489-bb1b-
e08425886fEX/",
        "custom_cookbooks_source": {
          "type": "s3",
          "url": "https://s3.amazonaws.com/amzn-s3-demo-bucket/opsworks-cookbook-
demo.tar.gz",
          "username": "AKIAJUQN...WG644EXA",
          "password": "O5v+4Zz+...rcKbFTJu",
          "ssh_key": null,
          "revision": null
        },
        "name": "MyCookbooksDemoStack",
        "region": "us-west-2",
        "stack_id": "040c3def-b2b4-4489-bb1b-e08425886fEX",
```

```
        "use_custom_cookbooks": true,
        "vpc_id": null
      },
      "ecs_clusters": [

      ],
      "volumes": [

      ]
    },
    "chef": {
      "customer_recipes": [
        "opsworks_cookbook_demo::configure"
      ],
      "customer_json": "e30=\n",
      "customer_data_bags": "e30=\n"
    }
  }
}
```

instance_report

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

傳回延伸的執行個體報告。

```
sudo opsworks-agent-cli instance_report
```

下列輸出範例來自執行個體。

```
$ sudo opsworks-agent-cli instance_report

AWS OpsWorks Instance Agent State Report:

  Last activity was a "configure" on 2015-12-01 18:19:23 UTC
  Agent Status: The AWS OpsWorks agent is running as PID 30998
```

```
  Agent Version: 4004-20151201152533, up to date
  OpsWorks Stack: MyCookbooksDemoStack
  OpsWorks Layers: MyCookbooksDemoLayer
  OpsWorks Instance: cookbooks-demo1
  EC2 Instance ID: i-a480e9EX
  EC2 Instance Type: c3.large
  Architecture: x86_64
  Total Memory: 3.84 Gb
  CPU: 2x  Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz

Location:

  EC2 Region: us-west-2
  EC2 Availability Zone: us-west-2a

Networking:

  Public IP: 192.0.2.0
  Private IP: 198.51.100.0
```

list_commands

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

列出已在此執行個體上執行之每個活動的時間。您可以在其他代理程式 CLI 命令中使用這些時間來指定特定的執行。

```
sudo opsworks-agent-cli list_commands  [activity] [date]
```

下列輸出範例來自已執行配置、設定及更新自訂技術指南活動的執行個體。

```
$ sudo opsworks-agent-cli list_commands

2015-11-24T21:00:28        update_custom_cookbooks
2015-12-01T18:19:09        setup
2015-12-01T18:20:24        configure
```

run_command

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

執行 AWS OpsWorks Stacks 命令，此 JSON 檔案包含 Chef Run-list，其中包含執行 Stacks AWS OpsWorks 活動（設定、設定、部署等）所需的資訊。run_command 命令會產生您可以透過執行 show_log 檢視的日誌項目。此選項僅用於開發目的，因此 AWS OpsWorks Stacks 不會追蹤變更。

```
sudo opsworks-agent-cli run_command [activity] [date] [/path/to/valid/json.file]
```

根據預設， run_command會執行最新的 AWS OpsWorks Stacks 命令。使用下列選項來指定特定命令。

activity

執行指定的 AWS OpsWorks Stacks 命令：setup、configure、deploy、undeploystart、stop、 或 restart。

date

執行已在指定時間戳記執行的 AWS OpsWorks 命令。若要取得有效的時間戳記清單，請執行 list_commands。

file

執行指定的命令 JSON 檔案。若要取得命令的檔案路徑，請執行 get_json。

下列輸出範例來自執行個體並執行設定命令。

```
$ sudo opsworks-agent-cli run_command configure

[2015-12-02 16:52:53]  INFO [opsworks-agent(21970)]: About to re-run 'configure' from
 2015-12-01T18:20:24
...
[2015-12-02 16:53:02]  INFO [opsworks-agent(21970)]: Finished Chef run with exitcode 0
```

show_log

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

傳回命令的日誌檔案。

```
sudo opsworks-agent-cli show_log [activity] [date]
```

根據預設，show_log 會尾隨最新的日誌檔案。使用下列選項來指定特定命令。

activity

顯示指定活動的日誌檔案。

date

顯示在指定時間戳記執行之活動的日誌檔案。若要取得有效的時間戳記清單，請執行 list_commands。

下列輸出範例會顯示最新的日誌。

```
$ sudo opsworks-agent-cli show_log

[2015-12-02T16:52:59+00:00] INFO: Storing updated cookbooks/opsworks_cookbook_demo/
opsworks-cookbook-demo.tar.gz in the cache.
...
[2015-12-02T16:52:59+00:00] INFO: Report handlers complete
```

stack_state

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

顯示 AWS OpsWorks Stacks 在內部用於最近 Chef 執行的資訊。

```
opsworks-agent-cli stack_state
```

> **Note**
>
> 若是 Chef 12 Linux 執行個體，執行此命令將傳回有效的資訊，例如執行個體的堆疊組態和部署屬性。不過，若要取得更完整的資訊，請參閱 AWS OpsWorks Stacks 在執行個體上建立的 Chef 資料包。如需詳細資訊，請參閱 AWS OpsWorks Stacks Data Bag 參考。

下列輸出範例來自執行個體。

```
$ sudo opsworks-agent-cli stack_state

{
  "last_command": {
    "sent_at": "2015-12-01T18:19:23+00:00",
    "activity": "configure"
  },
  "instance": {
    "ami_id": "ami-d93622EX",
    "architecture": "x86_64",
    "auto_scaling_type": null,
    "availability_zone": "us-west-2a",
    "created_at": "2015-11-18T00:21:05+00:00",
    "ebs_optimized": false,
    "ec2_instance_id": "i-a480e9EX",
    "elastic_ip": null,
    "hostname": "cookbooks-demo1",
    "instance_id": "cfdaa716-42fe-4e3b-9762-fef184ddd8EX",
```

```
    "instance_type": "c3.large",
    "layer_ids": [
      "93f50d83-1e73-45c4-840a-0d4f07cda1EX"
    ],
    "os": "Amazon Linux 2015.09",
    "private_dns": "ip-192-0-2-0.us-west-2.compute.internal",
    "private_ip": "10.122.69.33",
    "public_dns": "ec2-203-0-113-0.us-west-2.compute.amazonaws.com",
    "public_ip": "192.0.2.0",
    "root_device_type": "ebs",
    "root_device_volume_id": "vol-f6f7e8EX",
    "ssh_host_dsa_key_fingerprint": "f2:...:15",
    "ssh_host_dsa_key_public": "ssh-dss AAAAB3Nz...a8vMbqA=",
    "ssh_host_rsa_key_fingerprint": "0a:...:96",
    "ssh_host_rsa_key_public": "ssh-rsa AAAAB3Nz...yhPanvo7",
    "status": "online",
    "subnet_id": null,
    "virtualization_type": "paravirtual",
    "infrastructure_class": "ec2",
    "ssh_host_dsa_key_private": "-----BEGIN DSA PRIVATE KEY-----\nMIIDVwIB...g5OtgQ==
\n-----END DSA PRIVATE KEY-----\n",
    "ssh_host_rsa_key_private": "-----BEGIN RSA PRIVATE KEY-----\nMIIEowIB...78kprtIw
\n-----END RSA PRIVATE KEY-----\n"
  },
  "layers": [
    {
      "layer_id": "93f50d83-1e73-45c4-840a-0d4f07cda1EX",
      "name": "MyCookbooksDemoLayer",
      "packages": [

      ],
      "shortname": "cookbooks-demo",
      "type": "custom",
      "volume_configurations": [

      ]
    }
  ],
  "applications": null,
  "stack": {
    "arn": "arn:aws:opsworks:us-west-2:80398EXAMPLE:stack/040c3def-b2b4-4489-bb1b-
e08425886fEX/",
    "custom_cookbooks_source": {
      "type": "s3",
```

```
      "url": "https://s3.amazonaws.com/amzn-s3-demo-bucket/opsworks-cookbook-
demo.tar.gz",
      "username": "AKIAJUQN...WG644EXA",
      "password": "O5v+4Zz+...rcKbFTJu",
      "ssh_key": null,
      "revision": null
    },
    "name": "MyCookbooksDemoStack",
    "region": "us-west-2",
    "stack_id": "040c3def-b2b4-4489-bb1b-e08425886fEX",
    "use_custom_cookbooks": true,
    "vpc_id": null
  },
  "agent": {
    "valid_activities": [
      "reboot",
      "stop",
      "deploy",
      "grant_remote_access",
      "revoke_remote_access",
      "update_agent",
      "setup",
      "configure",
      "update_dependencies",
      "install_dependencies",
      "update_custom_cookbooks",
      "execute_recipes",
      "sync_remote_users"
    ]
  }
}
```

AWS OpsWorks Stacks Data Bag 參考

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

AWS OpsWorks Stacks 會將各種設定公開到配方，做為 Chef 資料包內容。此參考清單列出資料包的內容。

「資料包」是一種 Chef 概念。資料包為以 JSON 資料存放在執行個體上的全域變數。JSON 資料可從 Chef 進行存取。例如，資料包可以存放全域變數，例如應用程式的來源 URL、執行個體的主機名稱，以及相關聯的堆疊的 VPC 識別符。 AWS OpsWorks Stacks 會將其資料包存放在每個堆疊的執行個體上。在 Linux 執行個體上， AWS OpsWorks Stacks 會將資料包存放在 `/var/chef/runs/run-ID/data_bags`目錄中。在 Windows 執行個體上，它會將資料包存放在 `drive:\chef\runs\run-id\data_bags` 目錄中。在這兩種情況下，*run-ID* 都是 Stacks AWS OpsWorks 指派給執行個體上每個 Chef 的唯一 ID。這些目錄包含一組資料包 (子目錄)。每個資料包都包含零或多個資料包項目，即包含資料包內容組的 JSON 格式檔案。

> **Note**
>
> AWS OpsWorks Stacks 不支援加密的資料包。若要以加密形式存放機密資料 (例如密碼或憑證)，建議您將它存放在私有 S3 儲存貯體中。然後，您可以建立自訂配方，定義其使用適用於 Ruby 的 Amazon 開發套件擷取資料。如需範例，請參閱「使用適用於 Ruby 的 SDK」。

資料包內容可包含下列任何項目：

- 字串內容遵循標準 Ruby 語法，可使用單引號或雙引號，雖然包含特定特殊字元的字串必須使用雙引號。如需詳細資訊，請前往 Ruby 文件網站。
- 布林值內容，為 `true` 或 `false` (無引號)。
- 數字內容，為整數或小數，例如 `4` 或 `2.5` (無引號)。
- 清單內容，其格式為以中括弧圍起的逗點分隔值 (無引號)，例如 `[ '80', '443' ]`
- JSON 物件，包含額外的資料包內容，例如 `"my-app": {"elastic_ip": null,...}`。

Chef 配方可透過 Chef 搜尋或直接存取資料包、資料包項目，以及資料包內容。下列內容說明如何使用兩種存取方式 (雖然 Chef 搜尋為偏好選項)。

若要透過 Chef 搜尋存取資料包，請使用搜尋方法，指定所需的搜尋索引。 AWS OpsWorks Stacks 提供下列搜尋索引：

- aws_opsworks_app，代表堆疊的一組部署應用程式。
- aws_opsworks_command，代表在堆疊上執行的一組命令。

- aws_opsworks_ecs_cluster，代表堆疊的一組 Amazon Elastic Container Service (Amazon ECS) 叢集執行個體。
- aws_opsworks_elastic_load_balancer，代表堆疊的一組 Elastic Load Balancing 負載平衡器。
- aws_opsworks_instance，代表堆疊的一組執行個體。
- aws_opsworks_layer，代表堆疊的一組 layer。
- aws_opsworks_rds_db_instance，代表堆疊的一組 Amazon Relational Database Service (Amazon RDS) 執行個體。
- aws_opsworks_stack，代表堆疊。
- aws_opsworks_user，代表堆疊的一組使用者。

在您了解搜尋索引名稱之後，您便可以存取該搜尋索引的資料包內容。例如，下列配方程式碼使用 aws_opsworks_app 搜尋索引取得 aws_opsworks_app 資料包 (aws_opsworks_app 目錄) 中第一個資料包項目 (第一個 JSON 檔案) 的內容。然後，程式碼將兩個訊息寫入 Chef 日誌，一個有應用程式短名資料包內容 (JSON 檔案中的字串)，另一個有應用程式來源 URL 資料包內容 (JSON 檔案中的另一個字串)：

```
app = search("aws_opsworks_app").first
Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}' **********")
```

其中 ['shortname'] 和 ['app_source']['url'] 指定對應 JSON 檔案中的下列資料包內容：

```
{
  ...
  "shortname": "mylinuxdemoapp",
  ...
  "app_source": {
    ...
    "url": "https://s3.amazonaws.com/opsworks-demo-assets/opsworks-linux-demo-
nodejs.tar.gz",
  },
  ...
}
```

如需您可以搜尋的資料包內容清單，請參閱本節中的參考主題。

您也可以逐一查看資料包中的一組資料包項目。例如，下列配方程式碼 (與先前範例相似) 會在有超過一個資料包項目時，逐一查看資料包中的每個資料包項目。

```
search("aws_opsworks_app").each do |app|
  Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
  Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}'
 **********")
end
```

若您知道特定資料包內容存在，您可以使用下列語法尋找對應的資料包項目：

```
search("search_index", "key:value").first
```

例如，下列配方程式碼使用 aws_opsworks_app 搜尋索引尋找包含應用程式短名 mylinuxdemoapp 的資料包項目。程式碼接著會使用資料包項目的內容將對應應用程式短名和來源 URL 寫入 Chef 日誌：

```
app = search("aws_opsworks_app", "shortname:mylinuxdemoapp").first
Chef::Log.info("********** For the app with the short name '#{app['shortname']}', the
 app's URL is '#{app['app_source']['url']}' **********")
```

僅適用於 aws_opsworks_instance 搜尋索引，您可以指定 self:true 代表正在執行配方的執行個體。下列配方程式碼使用對應的資料包項目內容，以對應執行個體的 Stacks 產生的 ID 和作業系統，將訊息寫入 Chef AWS OpsWorks 日誌：

```
instance = search("aws_opsworks_instance", "self:true").first
Chef::Log.info("********** For instance '#{instance['instance_id']}', the instance's
 operating system is '#{instance['os']}' **********")
```

除了使用 Chef 搜尋存取資料包、資料包項目和資料包內容之外，您也可以直接存取他們。若要執行此作業，請使用 data_bag 和 data_bag_item 方法分別存取資料包和資料包項目。例如，下列配方程式碼會執行與先前範例一模一樣的動作，但是它會直接存取單一資料包項目，並會在超過一個的時候存取多個資料包項目：

```
# Syntax: data_bag_item("the data bag name", "the file name in the data bag without the
 file extension")
app = data_bag_item("aws_opsworks_app", "mylinuxdemoapp")
Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}' **********")

data_bag("aws_opsworks_app").each do |data_bag_item|
  app = data_bag_item("aws_opsworks_app", data_bag_item)
  Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
```

```
  Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}'
 **********")
end
```

在這兩種方法中，我們建議您使用 Chef 搜尋。所有本指南中的相關範例都會示範此方法。

主題

- 應用程式資料包 (aws_opsworks_app)
- 命令資料包 (aws_opsworks_command)
- Amazon ECS 叢集資料包 (aws_opsworks_ecs_cluster)
- Elastic Load Balancing Data Bag (aws_opsworks_elastic_load_balancer)
- 執行個體資料包 (aws_opsworks_instance)
- Layer 資料包 (aws_opsworks_layer)
- Amazon RDS Data Bag (aws_opsworks_rds_db_instance)
- 堆疊資料包 (aws_opsworks_stack)
- 使用者資料包 (aws_opsworks_user)

應用程式資料包 (aws_opsworks_app)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

針對部署事件或執行配方堆疊命令，代表應用程式的設定。

下列範例示範如何使用 Chef 搜尋搜尋單一資料包項目及多個資料包項目，將應用程式的短名和來源 URL 寫入 Chef 日誌：

```
app = search("aws_opsworks_app").first
Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}' **********")

search("aws_opsworks_app").each do |app|
  Chef::Log.info("********** The app's short name is '#{app['shortname']}' **********")
```

```
  Chef::Log.info("********** The app's URL is '#{app['app_source']['url']}'
 **********")
end
```

app_id	app_source	data_sources
部署	屬性	domains
enable_ssl	環境	name
shortname	ssl_configuration	type

app_id

應用程式 ID (字串)。識別應用程式的 GUID。

app_source

一組內容，指定 AWS OpsWorks Stacks 用來從其來源控制儲存庫部署應用程式的資訊。內容會根據儲存庫類型而有所不同。

密碼

私有儲存庫為密碼，公有儲存庫則為 "null" (字串)。若為私有 S3 儲存貯體，則此內容會設為私有金鑰。

修訂版

如果儲存庫有多個分支，此內容會指定該應用程式的分支或版本，例如 "version1" (字串)。否則會設為 "null"。

ssh_key

用來存取私有 Git 儲存庫的部署 SSH 金鑰，公有儲存庫則為 "null" (字串)。

type

應用程式的來源位置 (字串)。有效值包含：

- "archive"
- "git"
- "other"
- "s3"

url

應用程式來源的位置 (字串)。

使用者

私有儲存庫為使用者名稱，公有儲存庫則為 "null" (字串)。若為私有 S3 儲存貯體，此內容會設為存取金鑰。

屬性

一組內容，描述目錄結構和應用程式的內容。

document_root

文件樹狀結構的根目錄。定義指向文件根 (或應用程式首頁位置，例如 home_html) 的路徑，相對於您的部署目錄。除非指定此屬性，否則 document_root 預設為 public。document_root 的值開頭只能為 a-z、A-Z、0-9、_ (底線) 或 - (連字號) 字元。

data_sources

連線到應用程式資料庫所需的資訊。如果應用程式有連接的資料庫層， AWS OpsWorks Stacks 會自動將適當的值指派給此內容。

data_sources 的值為一個陣列，陣列必須使用整數位移存取，而非鍵。例如，若要存取應用程式的第一個資料來源，請使用 app[:data_sources][0][:type]。

database_name

資料庫名稱，通常是應用程式的短名 (字串)。

type

資料庫執行個體的類型，通常為 "RdsDbInstance" (字串)。

arn

資料庫執行個體的 Amazon Resource Name (ARN) (字串)。

部署

是否應部署應用程式 (布林值)。true 表示應用程式應在部署生命週期事件中部署。在安裝生命週期事件中，所有應用程式的此內容將為 true。若要判斷哪些應用程式應在執行個體上部署，請檢查執行個體所屬的 layer。

domains

應用程式的網域清單 (字串清單)。

enable_ssl

是否啟用 SSL 支援 (布林值)。

環境

使用者針對應用程式指定之環境變數的集合。如需如何定義應用程式環境變數的詳細資訊，請參閱新增應用程式。每個內容名稱都會設為環境變數名稱，並且對應的值都會設為變數的值。

name

應用程式的名稱，用於顯示用途 (字串)。

shortname

應用程式的簡短名稱，由 Stacks 從名稱 （ 字串 ） AWS OpsWorks 產生。短名會在內部及由配方使用。短名會做為您應用程式檔案安裝目錄的名稱使用。

ssl_configuration

certificate

若您啟用 SSL 支援，則為應用程式的 SSL 憑證，否則為 "null" (字串)。

chain

若啟用 SSL，則為指定中繼憑證授權單位金鑰或用戶端身分驗證的內容 (字串)。

private_key

若您啟用 SSL 支援，則為應用程式的 SSL 私有金鑰，否則為 "null" (字串)。

type

應用程式的類型，若為 Chef 12 Linux 和 Chef 12.2 Windows 堆疊，則總是會設為 "other" (字串)。

命令資料包 (aws_opsworks_command)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

代表 Stacks AWS OpsWorks 在一或多個執行個體上執行之命令的設定。

下列範例示範如何使用 Chef 搜尋搜尋單一資料包項目及多個資料包項目，將命令的類型和傳送命令的時間寫入 Chef 日誌：

```
command = search("aws_opsworks_command").first
Chef::Log.info("********** The command's type is '#{command['type']}' **********")
Chef::Log.info("********** The command was sent at '#{command['sent_at']}' **********")

search("aws_opsworks_command").each do |command|
  Chef::Log.info("********** The command's type is '#{command['type']}' **********")
  Chef::Log.info("********** The command was sent at '#{command['sent_at']}'
 **********")
end
```

args	command_id	iam_user_arn
instance_id	sent_at	type

args

命令的引數 (字串)。

command_id

命令的隨機唯一識別符，由 Stacks （ 字串 ） AWS OpsWorks 指派。

iam_user_arn

若命令是由客戶建立，則為建立命令之使用者的 Amazon Resource Name (ARN) (字串)。

instance_id

執行命令之執行個體的識別符 (字串)。

sent_at

Stacks 執行命令 （ 字串 ） AWS OpsWorks 時的時間戳記。

type

命令的類型 (字串)。有效值包含：

- `"configure"`
- `"deploy"`

- "deregister"
- "execute_recipes"
- "grant_remote_access"
- "install_dependencies"
- "restart"
- "revoke_remote_access"
- "rollback"
- "setup"
- "shutdown"
- "start"
- "stop"
- "sync_remote_users"
- "undeploy"
- "update_agent"
- "update_custom_cookbooks"
- "update_dependencies"

Amazon ECS 叢集資料包 (aws_opsworks_ecs_cluster)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

代表 Amazon ECS 叢集的設定。

下列範例示範如何使用 Chef 搜尋來搜尋單一資料包項目，然後搜尋多個資料包項目，以使用 Amazon ECS 叢集的名稱和 Amazon Resource Name (ARNs) 將訊息寫入 Chef 日誌：

```
ecs_cluster = search("aws_opsworks_ecs_cluster").first
Chef::Log.info("********** The ECS cluster's name is
 '#{ecs_cluster['ecs_cluster_name']}' **********")
```

```
Chef::Log.info("********** The ECS cluster's ARN is '#{ecs_cluster['ecs_cluster_arn']}'
 **********")

search("aws_opsworks_ecs_cluster").each do |ecs_cluster|
  Chef::Log.info("********** The ECS cluster's name is
 '#{ecs_cluster['ecs_cluster_name']}' **********")
  Chef::Log.info("********** The ECS cluster's ARN is
 '#{ecs_cluster['ecs_cluster_arn']}' **********")
end
```

ecs_cluster_arn	ecs_cluster_name

ecs_cluster_arn

叢集的 Amazon Resource Name (ARN) (字串)。

ecs_cluster_name

叢集的名稱 (字串)。

Elastic Load Balancing Data Bag (aws_opsworks_elastic_load_balancer)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

代表 Elastic Load Balancing 負載平衡器的設定。

下列範例示範如何使用 Chef 搜尋來搜尋單一資料包項目，接著使用多個資料包項目，以 Elastic Load Balancing 負載平衡器的名稱和 DNS 名稱將訊息寫入 Chef 日誌：

```
elastic_load_balancer = search("aws_opsworks_elastic_load_balancer").first
Chef::Log.info("********** The ELB's name is
 '#{elastic_load_balancer['elastic_load_balancer_name']}' **********")
Chef::Log.info("********** The ELB's DNS name is '#{elastic_load_balancer['dns_name']}'
 **********")
```

```
search("aws_opsworks_elastic_load_balancer").each do |elastic_load_balancer|
  Chef::Log.info("********** The ELB's name is
 '#{elastic_load_balancer['elastic_load_balancer_name']}' **********")
  Chef::Log.info("********** The ELB's DNS name is
 '#{elastic_load_balancer['dns_name']}' **********")
end
```

elastic_load_balancer_name	dns_name	layer_id

elastic_load_balancer_name

負載平衡器的名稱 (字串)。

dns_name

負載平衡器的 DNS 名稱 (字串)。

layer_id

負載平衡器指派至之 layer AWS OpsWorks 的 Stacks ID （ 字串 ）。

執行個體資料包 (aws_opsworks_instance)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

代表執行個體的設定。

下列範例示範如何使用 Chef 搜尋搜尋單一資料包項目及多個資料包項目，將執行個體的主機名稱和 ID 寫入 Chef 日誌：

```
instance = search("aws_opsworks_instance").first
Chef::Log.info("********** The instance's hostname is '#{instance['hostname']}'
 **********")
Chef::Log.info("********** The instance's ID is '#{instance['instance_id']}'
 **********")
```

```
search("aws_opsworks_instance").each do |instance|
  Chef::Log.info("********** The instance's hostname is '#{instance['hostname']}'
 **********")
  Chef::Log.info("********** The instance's ID is '#{instance['instance_id']}'
 **********")
end
```

下列範例顯示使用 Chef 搜尋來搜尋多個資料包項目的不同方式，以尋找包含指定 Amazon EC2 執行個體 ID 的資料包項目。範例接著會使用資料包項目的內容將對應執行個體的公有 IP 地址寫入 Chef 日誌：

```
instance = search("aws_opsworks_instance", "ec2_instance_id:i-12345678").first
Chef::Log.info("********** For instance '#{instance['ec2_instance_id']}', the
 instance's public IP address is '#{instance['public_ip']}' **********")

search("aws_opsworks_instance").each do |instance|
  if instance['ec2_instance_id'] == 'i-12345678'
    Chef::Log.info("********** For instance '#{instance['ec2_instance_id']}', the
 instance's public IP address is '#{instance['public_ip']}' **********")
  end
end
```

以下範例示範如何使用 Chef 搜尋以 `self:true` 尋找包含與執行配方之執行個體相關資訊的資料包項目。接著，範例會使用資料包項目的內容，將訊息寫入 Chef 日誌，其中包含對應執行個體的 AWS OpsWorks Stacks 產生的 ID 和執行個體的公有 IP 地址：

```
instance = search("aws_opsworks_instance", "self:true").first
Chef::Log.info("********** For instance '#{instance['instance_id']}', the instance's
 public IP address is '#{instance['public_ip']}' **********")
```

ami_id	架構	auto_scaling_type
availability_zone	created_at	ebs_optimized
ec2_instance_id	elastic_ip	hostname
instance_id	instance_type	layer_ids
os	private_dns	private_ip

public_dns	public_ip	root_device_type
root_device_volume_id	self	ssh_host_dsa_key_fingerprint
ssh_host_dsa_key_private	ssh_host_dsa_key_public	ssh_host_rsa_key_fingerprint
ssh_host_rsa_key_private	ssh_host_rsa_key_public	status
subnet_id	virtualization_type	

ami_id

執行個體的 AMI (Amazon Machine Image) ID (字串)。

架構

執行個體的架構，一律設為 "x86_64" (字串)。

auto_scaling_type

執行個體的擴展類型：null、timer 或 load (字串)。

availability_zone

執行個體的可用區域 (AZ)，例如 "us-west-2a" (字串)。

created_at

執行個體的建立時間，使用 UTC "*yyyy*-*mm*-*ddd*T*hh*:*mm*:*ss*+*hh*:*mm*" 格式 (字串)。例如，"2013-10-01T08:35:22+00:00" 對應到 2013 年 10 月 10 日 8:35:22，無時區位移。如需詳細資訊，請參閱 ISO 8601。

ebs_optimized

執行個體是否為 EBS 最佳化 (布林值)。

ec2_instance_id

EC2 執行個體 ID (字串)。

elastic_ip

彈性 IP 地址，若執行個體沒有彈性 IP 地址，則設為 "null" (字串)。

hostname

主機名稱，例如 "demo1" (字串)。

instance_id

執行個體 ID，這是 AWS OpsWorks Stacks 產生的 GUID，可唯一識別執行個體 （字串）。

instance_type

執行個體類型，例如 "c1.medium" (字串)。

layer_ids

執行個體 layer 的清單，以其唯一 ID 識別。例如：307ut64c-c7e4-40cc-52f0-67d5k1f9992c。

os

執行個體的作業系統 (字串)。有效值包含：

- "Amazon Linux 2"
- "Amazon Linux 2018.03"
- "Amazon Linux 2017.09"
- "Amazon Linux 2017.03"
- "Amazon Linux 2016.09"
- "Custom"
- "Microsoft Windows Server 2022 Base"
- "Microsoft Windows Server 2022 with SQL Server Express"
- "Microsoft Windows Server 2022 with SQL Server Standard"
- "Microsoft Windows Server 2022 with SQL Server Web"
- "Microsoft Windows Server 2019 Base"
- "Microsoft Windows Server 2019 with SQL Server Express"
- "Microsoft Windows Server 2019 with SQL Server Standard"
- "Microsoft Windows Server 2019 with SQL Server Web"
- "CentOS 7"
- "Red Hat Enterprise Linux 7"
- "Ubuntu 20.04 LTS"
- "Ubuntu 18.04 LTS"
- "Ubuntu 16.04 LTS"
- "Ubuntu 14.04 LTS"

private_dns

私有 DNS 名稱 (字串)。

private_ip

私有 IP 地址 (字串)。

public_dns

公有 DNS 名稱 (字串)。

public_ip

公有 IP 地址 (字串)。

root_device_type

根設備類型 (字串)。有效值包含：

- `"ebs`
- `"instance-store"`

root_device_volume_id

根設備的磁碟區 ID (字串)。

self

若此資料包項目包含執行配方之執行個體的相關資訊，則為 `true`。否則為 `false` (布林值)。此值僅適用於配方，而不是透過 AWS OpsWorks Stacks API。

ssh_host_dsa_key_fingerprint

位元組的短序列，用來識別較長的 DSA 公有金鑰 (字串)。

ssh_host_dsa_key_private

由 DSA 為執行個體 SSH 身分驗證產生的私有金鑰 (字串)。

ssh_host_dsa_key_public

由 DSA 為執行個體 SSH 身分驗證產生的公有金鑰 (字串)。

ssh_host_rsa_key_fingerprint

位元組的短序列，用來識別較長的 RSA 公有金鑰 (字串)。

ssh_host_rsa_key_private

由 RSA 為執行個體 SSH 身分驗證產生的私有金鑰 (字串)。

ssh_host_rsa_key_public

由 RSA 為執行個體 SSH 身分驗證產生的公有金鑰 (字串)。

status

執行個體的狀態 (字串)。有效值包含：

- "requested"
- "booting"
- "running_setup"
- "online"
- "setup_failed"
- "start_failed"
- "terminating"
- "terminated"
- "stopped"
- "connection_lost"

subnet_id

執行個體的子網路 ID (字串)。

virtualization_type

執行個體的虛擬化類型 (字串)。

Layer 資料包 (aws_opsworks_layer)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

代表 layer 的設定。

下列範例示範如何使用 Chef 搜尋搜尋單一資料包項目及多個資料包項目，將 layer 的名稱和短名寫入 Chef 日誌：

```
layer = search("aws_opsworks_layer").first
Chef::Log.info("********** The layer's name is '#{layer['name']}' **********")
Chef::Log.info("********** The layer's shortname is '#{layer['shortname']}'
 **********")

search("aws_opsworks_layer").each do |layer|
  Chef::Log.info("********** The layer's name is '#{layer['name']}' **********")
  Chef::Log.info("********** The layer's shortname is '#{layer['shortname']}'
 **********")
end
```

ecs_cluster_arn	layer_id	name
packages	shortname	type
volume_configurations		

ecs_cluster_arn

如果 layer 已指派 Amazon ECS 叢集，Amazon ECS 叢集的 Amazon Resource Name (ARN)（字串）。

encrypted

若 EBS 磁碟區已加密，則為 `true`。否則為 `false` (布林值)。

layer_id

layer ID，這是 Stacks AWS OpsWorks 產生的 GUID，可唯一識別 layer（字串）。

name

layer 的名稱，用以在主控台中表示 layer (字串)。名稱可以是使用者定義的，且不一定要是唯一的。

packages

要安裝的套件清單 (字串清單)。

shortname

layer 的短名，由使用者定義 (字串)。

type

layer 的類型，若為 Chef 12 Linux 和 Chef 12.2 Windows 堆疊，一律會設為 "custom" (字串)。

volume_configurations

Amazon EBS 磁碟區組態的清單。

iops

磁碟區可支援的每秒 I/O 操作數。

mount_point

磁碟區的掛載點目錄。

number_of_disks

磁碟區中的磁碟數。

raid_level

磁碟區的 RAID 組態層級。

size

磁碟區的大小 (GiB)。

volume_type

磁碟區的類型：一般用途、磁性、佈建 IOPS、輸送量最佳化 HDD，或冷 HDD。

Amazon RDS Data Bag (aws_opsworks_rds_db_instance)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

一組資料包內容，指定 Amazon Relational Database Service (Amazon RDS) 執行個體的組態，如下所示：

address	db_instance_identifier	db_password

db_user	engine	rds_db_instance_arn
region		

下列範例示範如何使用 Chef 搜尋來搜尋單一資料包項目，然後搜尋多個資料包項目，以使用 Amazon RDS 執行個體的地址和資料庫引擎類型將訊息寫入 Chef 日誌：

```
rds_db_instance = search("aws_opsworks_rds_db_instance").first
Chef::Log.info("********** The RDS instance's address is
 '#{rds_db_instance['address']}' **********")
Chef::Log.info("********** The RDS instance's database engine type is
 '#{rds_db_instance['engine']}' **********")

search("aws_opsworks_rds_db_instance").each do |rds_db_instance|
  Chef::Log.info("********** The RDS instance's address is
 '#{rds_db_instance['address']}' **********")
  Chef::Log.info("********** The RDS instance's database engine type is
 '#{rds_db_instance['engine']}' **********")
end
```

address

執行個體的 DNS 名稱。

port

執行個體的連接埠。

db_instance_identifier

執行個體的 ID。

db_password

執行個體的主要密碼。

db_user

執行個體的主要使用者名稱。

engine

執行個體的資料庫引擎，例如 `mysql` (字串)。

rds_db_instance_arn

執行個體的 Amazon Resource Name (ARN)。

region

執行個體的 AWS 區域，例如 `us-west-2` (字串)。

堆疊資料包 (aws_opsworks_stack)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

代表堆疊的設定。

下列範例示範如何使用 Chef 搜尋將堆疊的名稱和技術指南的來源 URL 寫入 Chef 日誌：

```
stack = search("aws_opsworks_stack").first
Chef::Log.info("********** The stack's name is '#{stack['name']}' **********")
Chef::Log.info("********** The stack's cookbook URL is
 '#{stack['custom_cookbooks_source']['url']}' **********")
```

arn	custom_cookbooks_source	name
region	stack_id	use_custom_cookbooks
vpc_id		

arn

堆疊的 Amazon Resource Name (ARN) (字串)。

custom_cookbooks_source

一組內容，指定自訂技術指南的來源儲存庫。

type

儲存庫類型 (字串)。有效值包含：

- "archive"
- "git"
- "s3"

url

儲存庫 URL，例如 "git://github.com/amazonwebservices/opsworks-demo-php-simple-app.git" (字串)。

使用者名稱

私有儲存庫為使用者名稱，公有儲存庫則為 null (字串)。對於私有 Amazon Simple Storage Service (Amazon S3) 儲存貯體，內容會設定為存取金鑰。

密碼

私有儲存庫為密碼，公有儲存庫則為 null (字串)。若為私有 S3 儲存貯體，則此內容會設為私有金鑰。

ssh_key

用來存取私有 Git 儲存庫的部署 SSH 金鑰，公有儲存庫則為 null (字串)。

修訂版

如果儲存庫有多個分支，此內容會指定該應用程式的分支或版本，例如 "version1" (字串)。否則會設為 null。

name

堆疊名稱 (字串)。

region

堆疊的 AWS 區域名稱 (字串)。

stack_id

識別堆疊的 GUID (字串)。

use_custom_cookbooks

是否啟用自訂技術指南 (布林值)。

vpc_id

若堆疊正在 VPC 中執行，則為 VPC ID (若堆疊正在 VPC 中執行) (字串)。

使用者資料包 (aws_opsworks_user)

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

代表使用者的設定。

下列範例示範如何使用 Chef 搜尋搜尋單一資料包項目及多個資料包項目，將使用者的使用者名稱和 Amazon Resource Name (ARN) 寫入 Chef 日誌：

```
user = search("aws_opsworks_user").first
Chef::Log.info("********** The user's user name is '#{user['username']}' **********")
Chef::Log.info("********** The user's user ARN is '#{user['iam_user_arn']}'
 **********")

# Or...

search("aws_opsworks_user").each do |user|
  Chef::Log.info("********** The user's user name is '#{user['username']}' **********")
  Chef::Log.info("********** The user's user ARN is '#{user['iam_user_arn']}'
 **********")
end
```

administrator_privileges	iam_user_arn	remote_access
ssh_public_key	unix_user_id	使用者名稱

administrator_privileges

使用者是否具備管理員權限 (布林值)。

iam_user_arn

使用者的 Amazon Resource Name (ARN) (字串)。

remote_access

使用者是否可以使用 RDP 登入執行個體 (布林值)。

ssh_public_key

使用者的公有金鑰，如 Stacks AWS OpsWorks 主控台或 API （ 字串 ） 所提供。

unix_user_id

使用者的 Unix ID (號碼)。

使用者名稱

使用者名稱 (字串)。

OpsWorks 代理程式變更

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

Chef 12 代理版本

下表說明 AWS OpsWorks Stacks 在其管理的執行個體上安裝的 Chef 12 代理程式的重要變更。

代理程式版本	描述	版本日期
4042	• 此代理程式版本僅包含沒有新功能的次要變更	2023 年 2 月 7 日
4041	• 此代理程式版本僅包含沒有新功能的次要變更 • 更新 Amazon CA 憑證	2023 年 1 月 27 日
4040	• 此代理程式版本僅包含沒有新功能的次要變更	2022 年 7 月 22 日

代理程式版本	描述	版本日期
4039	• 修正 Ubuntu AMIs ECS 整合	2020 年 4 月 30 日
4038	• 修正 DST 變更期間傳送執行個體統計資料的錯誤 • 在代理程式下載和安裝期間尊重 `no_proxy` 環境變數	2020 年 3 月 5 日
4037	• 使用 Sigv4 新增對 S3 URL 簽署請求的支援，而不需要區域 • 移除使用 Sigv2 簽署 S3 請求的支援	2019 年 6 月 4 日
4035	• 修復 ECS 設定期間的錯誤 • 修復在執行個體類型變更之後發生重複的 `fstab` 項目	2019 年 5 月 8 日
4033	• 新增 Ubuntu 18.04 的支援 • 修復 Amazon Linux 2 的代理程式安裝錯誤	2018 年 11 月 26 日
4032	• 已新增 Amazon Linux 2 的支援	2018 年 10 月 24 日
4031	• 新增 Amazon Linux 2018.03 支援 • 支援託管於另一個帳戶的公有 S3 存檔	2018 年 8 月 15 日
4030	• c5d 執行個體修復磁碟區處理	2018 年 5 月 31 日
4029	• 安裝 `nvme-cli` 在 Ubuntu 14.04 上 • 修復掛載在 c5、m5 執行個體的磁碟區 • 重新啟動時，一律保留主機名稱	2018 年 5 月 2 日
4028	• 修復 CentOS 的 `monit` 組態	2018 年 3 月 20 日
4027	• 支援在 NVMe 14.04 上掛載 NVMe 磁碟區 (`nvme-cli` 必須手動安裝) • 磁碟區不要求 `name` 屬性	2018 年 2 月 17 日

代理程式版本	描述	版本日期
4026	• 使用 EBS 磁碟區 ID 掛載 NVMe 為基礎的 EBS 磁碟區 • 修復掛載在 i3 執行個體的 EBS 磁碟區 • 修復掛載在 c5 m5 執行個體的 EBS 磁碟區順序	2018 年 1 月 31 日
4025	• 修復 NVMe 裝置的處理	2017 年 12 月 13 日
4024	• 新增 Amazon Linux 2017.09 支援	2017 年 12 月 5 日
4023	• 新增 CloudWatch Logs 整合支援	2017 年 4 月 2 日
4022	• 將 Chef 用戶端版本更新為 12.18.31	2017 年 2 月 1 日
4021	• 改善代理處理	2016 年 12 月 16 日
4020	• 將 Chef 用戶端版本更新為 12.16.42	2016 年 12 月 8 日
4019	• 代理程式安裝期間的來源代理變數 • Red Hat Enterprise Linux 7 現在使用 `systemd`，而非 `monit` • 不要在 Red Hat Enterprise Linux 7 上設定 EPEL • 使用 `flock` 而非 `lockrun.c` 來鎖定程序 • 檢查 `ps -p1` 時避免 `systemd` 的奇數輸出	2016 年 10 月 19 日
4018	• 將 Chef 用戶端版本更新為 12.13.37 • 新增 Amazon Linux 2016.09 支援	2016 年 8 月 25 日
4017	• 將 Chef 用戶端版本更新為 12.12.15	2016 年 8 月 10 日
4016	• 修復未使用 `monit` 之系統上的代理程式解除安裝	2016 年 6 月 23 日
4015	• 修復 Amazon Linux 2016.03 的 ECS 設定	2016 年 6 月 17 日

代理程式版本	描述	版本日期
4011	• 將 Chef 用戶端版本更新為 12.10.24 • 改善日誌上傳處理	2016 年 5 月 19 日
4008	• 新增 Amazon Linux 2016.03 支援 • 新增套件安裝的逾時 • 將 xfs 新增至 /etc/filesystems (若存在)	2016 年 3 月 16 日
4007	• 將 Chef 用戶端版本更新為 12.7.2 • 改善現場部署執行個體 (託管在 AWS 外部的伺服器) 的錯誤處理 • 改善與最新 chef-sugar 的相容性 • 重試要部署的封存下載	2016 年 3 月 4 日
4006	• 將 Chef 用戶端版本更新為 12.6.0 • 不要在代理程式安裝時安裝 libxml2-devel/libxml2-dev 和 libxslt-devel/libxslt-dev 套件	2016 年 1 月 21 日
4005	• 一律在 ohai for ec2 基礎設施中啟用 ec2 資料以修復 ec2 匯入	2015 年 12 月 17 日
4004	• AWS OpsWorks Chef 12 Linux 的 Stacks 支援 - Chef Client 12.5.1	2015 年 12 月 3 日

Chef 11.10 代理版本

下表說明 AWS OpsWorks Stacks 在其管理的執行個體上安裝的 Chef 11.10 代理程式的重要變更。

代理程式版本	描述	版本日期
3456	• 此代理程式版本僅包含沒有新功能的次要變更 • 更新 Amazon CA 憑證	2023 年 1 月 27 日
3455	• 此代理程式版本僅包含沒有新功能的次要變更	2022 年 11 月 1 日

代理程式版本	描述	版本日期
3454	• 修正 Ubuntu AMIs ECS 整合	2020 年 4 月 28 日
3453	• 修正 DST 變更期間傳送執行個體統計資料的錯誤 • 修正 RHEL7 安裝程式中遺失的套件錯誤 • 在代理程式下載和安裝期間尊重 no_proxy 環境變數	2020 年 3 月 5 日
3452	• 不要在 Amazon S3 虛擬路徑 URL 中包含區域 (如果是 us-east-1) • 提取並上傳內部技術指南到階段區域特定的儲存貯體 • 修正 Chef 11.10 的 fstab 項目 • 刪除 S3 的 Sigv2 使用情況並獲取請求中儲存貯體的區域	2019 年 8 月 13 日
3451	• 新增 Ruby Ruby 2.6.1 的支援	2019 年 3 月 20 日
3450	• 修復預設 EBS 屬性 • 修復適用於 Amazon Linux 2 的 CloudWatchLogs 代理程式安裝 • 修復 2.6.14 以上 rubygem 版本的Bundler 安裝 • 修復公有 S3 存檔支援	2018 年 12 月 3 日
3449	• c5d 執行個體修復磁碟區處理 • Fix RAID 陣列支援 NVMe 裝置執行個體	2018 年 6 月 5 日
3448	• 將預設 2.3 版本的 Ruby 升級到 2.3.7 • 修正在 NVMe 14.04 NVMe 執行個體上安裝 EBS 磁碟區 • 支援另一個帳戶託管的公有 Amazon S3 封存 • 修正 Red Hat Enterprise Linux 執行個體的開機問題 opsworks-agent	2018 年 5 月 8 日

代理程式版本	描述	版本日期
3447	• 使用 EBS 磁碟區 ID 掛載 NVMe 為基礎的 EBS 磁碟區 • 修復掛載在 i3 執行個體的 EBS 磁碟區 • 修正掛載在 c5 m5 的 EBS 磁碟區順序 • 將預設 2.3 版本的 Ruby 更新到 2.3.6	2018 年 1 月 31 日
3446	• 修復 NVMe 裝置的處理 • 將預設 2.3 版本的 Ruby 更新到 2.3.5	2017 年 12 月 14 日
3445	• 新增 Amazon Linux 2017.09 支援 • 將預設 2.2 版本的 Ruby 更新到 2.2.8	2017 年 10 月 31 日
3444	• 新增對 CloudWatch Logs 的支援	2017 年 4 月 1 日
3443	• 改善代理處理	2016 年 12 月 15 日
3442	• 將預設 2.3 版本的 Ruby 更新到 2.3.3 • 將預設 2.2 版本的 Ruby 更新到 2.2.6	2016 年 6 月 12 日
3441	• 代理程式安裝期間的來源代理變數	2016 年 10 月 21 日
3440	• 新增 Amazon Linux 2016.09 支援	2016 年 9 月 13 日
3439	• 小幅度修改，沒有新功能	2016 年 7 月 29 日
3438	• 新增 Ruby 2.3.1 的支援 • 用 IAM 執行個體描述檔的登入資料改善執行個體註冊率 • 移除剩餘 `s3curl.pl` • 修復 Amazon Linux 2016.03 的 ECS 設定	2016 年 6 月 17 日
3437	• 將預設 2.2 版本的 Ruby 更新到 2.2.5	2016 年 5 月 4 日

代理程式版本	描述	版本日期
3436	• 更新 Red Hat Enterprise Linux 的 EPEL URL 重要：沒有此變更，Red Hat Enterprise Linux 執行個體無法啟動。	2016 年 4 月 18 日
3435	• 將預設 2.1 版本的 Ruby 更新到 2.1.9 • 改善 Amazon S3 和封存部署的處理	2016 年 4 月 6 日
3434	• 新增 Amazon Linux 2016.03 支援 • 重試套件安裝	2016 年 3 月 16 日
3433	• 內部部署執行個體的一些改進 （託管於 外部的伺服器 AWS) • 改善與最新 的相容性`chef-sugar` • 重試要部署的封存下載 • 修正 Ruby gem 的安裝 URL	2016 年 2 月 27 日
3432	• 提升儲存貯體名稱特殊字元的處理 • 將 `s3_file` 更新到第 2.6.6 版 • 略過無指定掛載點的磁碟區掛載 • 一律重新啟動 `unicorn` 而不是停止再開始，以防止部署期間停機 • 針對 `setup` 命令，一律選擇更新自訂技術指南。 • 在建立 RAID 陣列後，更新 `initramfs` 以防止裝置重新開機時的對應問題	2016 年 1 月 20 日
3431	• 修正 Rails layer 的 `passenger` 和 `unicorn` gem 安裝問題 • 更新預設 Ruby 的 2.0、2.1 和 2.2 版本到 2.0.0p648、2.1.8 和 2.2.4 • 允許設定 `postgres` 套件名稱自訂 JSON • 將 Node.js 預設版本更新到 0.12.9	2015 年 12 月 22 日

代理程式版本	描述	版本日期
3430	• 小幅度修改，沒有新功能	2015 年 11 月 25 日
3429	• 提升 OpsWorks 代理程式 daemonize (關閉 stdout/stderr) • 提升 `s3_file` 資源的健全性 (重試、捕捉到的例外狀況)	2015 年 18 月 11 日
3428	• 根據 Gemfile，新增 `postgres` 轉接器偵測，修正 https://github.com/aws/opsworks-cookbooks/issues/136	2016 年 6 月 17 日
3427	• 修正在代理程式擷取登入資料的問題 • 更新預設 Ruby 的 2.0、2.1 和 2.2 版本到 2.0.0p647、2.1.7 和 2.2.3	2015 年 9 月 11 日
3426	• 更新 `aws-sdk` 到 1.65.0 • 將 取代`s3curl`為 以改善從 Amazon S3 下載 `s3_file cookbook` • 改變預設 Node.js 版本更新到 0.12.7 • 記錄新增至 Node.js 應用程式。`shared/log` 目錄中記錄和輪換的 STDOUT 和 STDERR • 使自訂技術指南子模組切換更新更明確 • 新增解決方法，以確保 https://github.com/aws/opsworks-cookbooks/issues/213 建立 `deploy` 目錄前，已綁定掛載	2015 年 8 月 27 日
3425	• ECS 支援 Amazon Linux 和 Ubuntu	2015 年 7 月 27 日
3424	• 小幅度修改，沒有新功能	2015 年 7 月 9 日
3422	• 完整支援 Red Hat Enterprise Linux 7 • 讓產生 `/etc/hosts` 對錯誤更有彈性	2015 年 6 月 29 日

代理程式版本	描述	版本日期
3421	• 覆寫 Red Hat Enterprise Linux 7 資料庫方案的選項 • 更新 `monit systemd` 組態來防止 `systemd` 傳送 `kill` 信號到 `monit` 監控的程序。	2015 年 6 月 11 日

AWS OpsWorks Stacks 資源

> **Important**
>
> AWS OpsWorks Stacks 服務已於 2024 年 5 月 26 日終止，並已針對新客戶和現有客戶停用。我們強烈建議客戶盡快將其工作負載遷移至其他解決方案。如果您對遷移有任何疑問，請透過 AWS re：Post 或透過 AWS Premium Support 聯絡 AWS 支援 團隊。

以下相關資源可協助您使用此服務。

參考指南、工具和支援資源

AWS OpsWorks Stacks 和 Amazon Web Services 提供數種有用的指南、論壇、聯絡資訊及其他資源。

- AWS OpsWorks Stacks API 參考 – 關於 AWS OpsWorks Stacks 動作和資料類型的描述、語法和使用範例，包括常見的參數和錯誤碼。
- AWS OpsWorks Stacks 技術常見問答集 – 開發人員已針對此產品提出的熱門問題。
- AWS OpsWorks Stacks 版本備註 – 目前版本的高階概觀。此文件特別注意任何新的功能、更正與已知問題。
- AWS Tools for PowerShell – 一組 Windows PowerShell cmdlet，可在 PowerShell 適用於 .NET 的 AWS SDK 環境中公開 的功能。
- AWS 命令列界面 – 存取 AWS 服務的統一命令列語法。AWS CLI 使用單一設定程序，以啟用所有支援服務的存取。
- AWS OpsWorks Stacks Command Line Reference – AWS OpsWorks Stacks 特定的命令，可用於命令列提示字元。

- 課程和研討會 – 連結至以角色為基礎的特殊課程，以及自行安排進度的實驗室，以協助強化您的 AWS 技能並取得實際經驗。
- AWS 開發人員中心 – 探索教學課程、下載工具，並了解 AWS 開發人員事件。
- AWS 開發人員工具 – 開發人員工具、SDKs、IDE 工具組和命令列工具的連結，用於開發和管理 AWS 應用程式。
- 入門資源中心 – 了解如何設定您的 AWS 帳戶、加入 AWS 社群，以及啟動您的第一個應用程式。

- 實用的教學課程 - 按照逐步教學課程在 AWS上啟動第一個應用程式。
- AWS 白皮書 – 技術 AWS 白皮書完整清單的連結，涵蓋架構、安全和經濟等主題，並由 AWS Solutions Architects 或其他技術專家撰寫。
- AWS 支援 中心 – 建立和管理 AWS 支援 案例的中樞。也包含其他實用資源的連結，例如論壇、技術 FAQs、服務運作狀態和 AWS Trusted Advisor。
- 支援 – 有關one-on-one、快速回應支援管道的資訊的主要網頁 支援，可協助您在雲端中建置和執行應用程式。
- 聯絡我們 – 查詢有關 AWS 帳單、帳戶、事件、濫用與其他問題的聯絡中心。
- AWS 網站條款 – 有關我們的著作權和商標、您的帳戶、授權和網站存取，以及其他主題的詳細資訊。

AWS 軟體開發套件

Amazon Web Services 提供軟體開發套件，可讓您從數種不同的程式設計語言存取 AWS OpsWorks Stacks。軟體開發套件程式庫可以自動執行多種常見的任務，包括對服務請求進行加密簽署、重試請求，或處理錯誤回應。

- 適用於 Java 的 AWS SDK – 設定和其他文件
- 適用於 .NET 的 AWS SDK – 設定和其他文件。
- 適用於 PHP 的 AWS 開發套件 – 文件
- 適用於 Ruby 的 AWS SDK – 文件
- 其他文件
- AWS SDK for Python (Boto) – 設定和其他文件

開放原始碼軟體

AWS OpsWorks Stacks 包含各種開放原始碼軟體套件，這些套件由其各自的授權管理。如需詳細資訊，請參閱下列內容：

- 針對 Chef 12 Linux 執行個體，請開啟執行個體上 `/opt/aws/opsworks/current` 目錄中的 `THIRD_PARTY_LICENSES` 檔案。
- 如需適用於 Linux 的 Chef 11.10 和更早版本，請下載 OpsWorks Linux Agent Attributions Document PDF。

AWS OpsWorks 文件歷史記錄

變更	描述	日期
AWS OpsWorks Stacks 的更新	您現在可以使用就地分離工具從 OpsWorks Stacks 服務分離 OpsWorks 執行個體，請參閱本指南中的使用就地 AWS OpsWorks Stacks 分離工具。	2024 年 4 月 11 日
AWS OpsWorks Stacks 的更新	您現在可以使用遷移指令碼將 Stacks AWS OpsWorks 遷移至 AWS Systems Manager Application Manager。如需詳細資訊，請參閱本指南中的將您的 AWS OpsWorks Stacks 應用程式遷移至 AWS Systems Manager Application Manager。	2022 年 12 月 22 日
AWS OpsWorks for Chef Automate 和 的更新 AWS OpsWorks for Puppet Enterprise	故障診斷程序現已推出，說明當 AWS OpsWorks for Chef Automate 或 OpsWorks for Puppet Enterprise 伺服器的系統維護失敗時，您可以執行哪些操作。如需詳細資訊，請參閱本指南中的 Chef Automate 伺服器的系統維護失敗，或 Puppet Enterprise 伺服器的系統維護失敗。	2022 年 9 月 29 日
AWS OpsWorks for Chef Automate 和 的更新 AWS OpsWorks for Puppet Enterprise	如果您的 AWS OpsWorks for Chef Automate 或 OpsWorks for Puppet Enterprise 伺服器進入 `Connection lost` 狀態，現在可以使用疑難排解程序。如需詳細資訊，請參閱本	2022 年 3 月 23 日

	指南中的 Chef Automate 伺服器處於 `Connection lost` 狀態或 Puppet Enterprise 伺服器處於 `Connection lost` 狀態。	
AWS OpsWorks Stacks 的更新	作為安全最佳實務，您現在可以新增 `aws:SourceArn` 或 `aws:SourceAccount` 條件金鑰（或兩者），以信任允許 AWS OpsWorks Stacks 存取在 AWS 其他服務中執行任務的關係政策。如需詳細資訊，請參閱本指南中的 AWS OpsWorks Stacks 中的跨服務混淆代理人預防。	2022 年 3 月 4 日
AWS OpsWorks for Chef Automate 和 的更新 AWS OpsWorks for Puppet Enterprise	作為安全最佳實務，您現在可以新增 `aws:SourceArn` 或 `aws:SourceAccount` 條件金鑰（或兩者），以信任允許 AWS OpsWorks for Chef Automate 和 OpsWorks for Puppet Enterprise 存取以在其他 服務中 AWS 執行任務的關係政策。如需詳細資訊，請參閱本指南中的跨服務混淆代理人預防。	2022 年 1 月 10 日

AWS OpsWorks for Chef Automate 和 的更新 AWS OpsWorks for Puppet Enterprise	AWS OpsWorks for Chef Automate 和 OpsWorks for Puppet Enterprise 已更新受管政策和 `AWSOpsWorksCMServiceRole` `AWSOpsWorksCMInstanceProfileRole` ，現在將秘密存放在 中 AWS Secrets Manager。	2021 年 5 月 3 日
更新至 AWS OpsWorks for Puppet Enterprise	您在主控台中建立的 OpsWorks for Puppet Enterprise 伺服器引擎版本現在是 2019.8.5。使用 API，您可以在建立 Puppet Enterprise 伺服器2017時指定版本2019或 。`DescribeServers` API 現在會傳回在其結果`PUPPET_API_CRL` 中呼叫的屬性。此屬性包含供內部使用的憑證撤銷清單。	2021 年 4 月 28 日
AWS OpsWorks Stacks 使用新的受管政策	AWS OpsWorks Stacks 已變更 受管政策，其中包含在 Stacks AWS OpsWorks 中執行所有動作的許可。新政策是 AWSOpsWorks_FullAccess。如需此政策中許可的詳細資訊，請參閱範例政策。	2021 年 2 月 19 日
將 AWS OpsWorks Stacks 堆疊從 EC2-Classic 遷移至 VPC	已新增文件，說明如何將 AWS OpsWorks Stacks 堆疊從 EC2-Classic 遷移至 VPC。	2020 年 9 月 29 日

為 AWS OpsWorks for Chef Automate 和 重新產生入門套件 AWS OpsWorks for Puppet Enterprise	已新增文件，說明如何為 AWS OpsWorks for Chef Automate 或 AWS OpsWorks for Puppet Enterprise 伺服器重新產生入門套件。	2020 年 7 月 29 日
AWS OpsWorks for Puppet Enterprise 可讓您建立使用自訂網域、憑證和私有金鑰的伺服器	您現在可以建立使用自訂網域、憑證和私有金鑰的 OpsWorks for Puppet Enterprise 伺服器。您可以從現有伺服器的備份建立伺服器，藉此更新現有的 Puppet Enterprise 伺服器以使用自訂網域。	2020 年 4 月 17 日
AWS OpsWorks for Chef Automate 和 AWS OpsWorks for Puppet Enterprise 現在支援在 主控台中標記	您現在可以使用 或 ，將標籤新增至 AWS OpsWorks for Chef Automate 伺服器或 AWS OpsWorks for Puppet Enterprise 主伺服器，或新增至伺服器備份 AWS Management Console AWS CLI。如需詳細資訊，請參閱使用標籤 (Chef) 或使用標籤 (Puppet)。	2020 年 2 月 26 日
AWS OpsWorks for Chef Automate 簡化現有 Chef Automate 1 伺服器的升級到 Chef Automate 2	您可以在 主控台的 AWS OpsWorks for Chef Automate 伺服器詳細資訊頁面上選擇開始升級，或執行 `StartMaintenance` API 動作，將執行 Chef Automate 1 的合格伺服器升級至 Chef Automate 2。如需詳細資訊，請參閱將 AWS OpsWorks for Chef Automate 伺服器升級至 Chef Automate 2。	2020 年 1 月 24 日

AWS OpsWorks for Chef Automate 而且 AWS OpsWorks for Puppet Enterprise	指南已新增有關 CM AWS OpsWorks (AWS OpsWorks for Chef Automate 和 AWS OpsWorks for Puppet Enterprise) 安全性的新章節。	2019 年 12 月 23 日
AWS OpsWorks for Chef Automate 和 AWS OpsWorks for Puppet Enterprise 支援標記	您現在可以使用 將標籤新增至 AWS OpsWorks for Chef Automate 伺服器或 AWS OpsWorks for Puppet Enterprise 主伺服器，或新增至伺服器備份 AWS CLI。AWS OpsWorks CM 現在支援標籤型授權。	2019 年 12 月 18 日
AWS OpsWorks for Chef Automate 可讓您建立使用自訂網域、憑證和私有金鑰的伺服器	您現在可以建立使用自訂網域、憑證和私有金鑰的 AWS OpsWorks for Chef Automate 2.0 伺服器。您可以從現有伺服器的備份建立伺服器，藉此更新現有的 Chef Automate 2.0 伺服器以使用自訂網域。	2019 年 10 月 22 日
AWS OpsWorks Stacks 現在支援 Ruby 2.6.1	AWS OpsWorks Stacks 支援 Chef 11.10 堆疊中的 Ruby 2.6.1 on Rails 應用程式伺服器層。	2019 年 5 月 2 日
AWS OpsWorks for Chef Automate 現在支援 Chef Automate 2.0	新 AWS OpsWorks for Chef Automate 伺服器將執行 Chef Automate 2.0，其中包括 Chef InSpec 的更新、合規掃描和報告的新功能，以及 Chef Infra。	2019 年 4 月 30 日

AWS OpsWorks for Chef Automate 而且 AWS OpsWorks for Puppet Enterprise	您現在可以使用 AWS CloudFormation 來建立 AWS OpsWorks for Chef Automate 伺服器或 AWS OpsWorks for Puppet Enterprise 主伺服器。	2019 年 1 月 24 日
AWS OpsWorks 堆疊	AWS OpsWorks Stacks 現在支援在 Chef 12 堆疊中執行 Ubuntu 18.04 LTS 的執行個體。	2018 年 12 月 18 日
適用於 Puppet Enterprise 的 AWS OpsWorks	新增設定 SSH 型連線至使用 CodeCommit 之控制儲存庫的程序。	2018 年 12 月 3 日
AWS OpsWorks 堆疊	AWS OpsWorks Stacks 現在支援在 Chef 12 堆疊中執行 Amazon Linux 2 的執行個體。	2018 年 11 月 15 日
AWS OpsWorks 堆疊	AWS OpsWorks Stacks 現在支援 Chef 11.10 堆疊中執行 Amazon Linux 2018.03 的執行個體。	2018 年 10 月 23 日
AWS OpsWorks 堆疊	AWS OpsWorks Stacks 現在支援 Chef 12 堆疊中執行 Amazon Linux 2018.03 的執行個體。	2018 年 8 月 23 日
AWS OpsWorks for Chef Automate Puppet Enterprise 的 和 OpsWorks	OpsWorks for Puppet Enterprise 已升級至 PE 2018.1.2。 AWS OpsWorks for Chef Automate 已升級至 Chef Automate 1.8.68。	2018 年 6 月 29 日

- AWS OpsWorks for Chef Automate 和 OpsWorks for Puppet Enterprise API 版本：2016-11-01
- AWS OpsWorks Stacks API 版本：2016-03-08

- 文件最近更新時間：2024-04-11

舊版更新

下表描述 2018 年 6 月前，每個 AWS OpsWorks 使用者指南版本的重要變更。

描述	日期
AWS OpsWorks Stacks Chef 版本適用於升級至 12.22 的 Windows 型堆疊；Ruby 版本現在為 2.3.6。	2018 年 4 月 19 日
使用 建立 AWS OpsWorks for Chef Automate 伺服器或 OpsWorks for Puppet Enterprise 主控端的新程序 AWS CLI。	2018 年 3 月 23 日
Chef Automate 版本已更新至 1.8；新增`opsworks-audit` 技術指南後， Chef Compliance 設定更簡化。	2018 年 3 月 5 日
新增對 AWS OpsWorks Amazon CloudWatch Events 中 Stacks 事件的支援。	2018 年 2 月 20 日
新增對 Stacks 中新 EBS AWS OpsWorks 磁碟區類型的支援，以及新 API DescribeOperatingSystems。	2018 年 1 月 25 日
OpsWorks for Puppet Enterprise AWS OpsWorks for Chef Automate 現在支援在建立伺服器時選取多個安全群組。	2018 年 1 月 18 日
新增對歐洲 AWS OpsWorks （巴黎） 區域中 Stacks 的支援。	2017 年 12 月 19 日
在六個額外區域新增對 AWS OpsWorks for Chef Automate 和 OpsWorks for Puppet Enterprise 的支援，並新增在 中建立 AWS OpsWorks for Chef Automate 和 OpsWorks for Puppet Enterprise 伺服器備份的程序 AWS Management Console。	2017 年 12 月 18 日
新增 OpsWorks for Puppet Enterprise 服務和文件。	2017 年 11 月 16 日
新增對 Amazon Linux 2017.09 的支援至 AWS OpsWorks Stacks。	2017 年 11 月 7 日
新增對 Chef Compliance 的支援 AWS OpsWorks for Chef Automate。	2017 年 10 月 25 日

描述	日期
新增對 Amazon Linux 2017 AWS OpsWorks for Chef Automate.09 的支援。	2017 年 10 月 9 日
已將系統維護主題新增至 AWS OpsWorks for Chef Automate 章節。	2017 年 7 月 28 日
新增對 Stacks AWS OpsWorks 中標籤的支援。	2017 年 6 月 6 日
新增與 CloudWatch Logs 的整合。	2017 年 4 月 10 日
新增了新的 AWS OpsWorks for Chef Automate 服務和文件。	2016 年 12 月 1 日
新增對美國東部（俄亥俄）區域端點的支援。	2016 年 10 月 12 日
新增對執行 Amazon Linux 2016.09 作業系統的堆疊和執行個體的支援。	2016 年 9 月 30 日
新增對亞太區域（首爾）區域和九個額外區域端點的支援。	2016 年 8 月 15 日
新增對內建 layer 中 Node.js 0.12.15 和 Ruby 2.3 的支援。	2016 年 7 月 6 日
新增對亞太區域（孟買）區域的支援。	2016 年 6 月 28 日
新增對執行 CentOS 7 作業系統的堆疊和執行個體的支援。	2016 年 6 月 22 日
新增描述 CodePipeline 和 AWS OpsWorks Stacks 整合的逐步解說。	2016 年 6 月 2 日
新增對執行 Ubuntu 16.04 LTS 作業系統的堆疊和執行個體的支援。	2016 年 6 月 1 日
新增對 Chef 12 Linux 的支援與相關文件。	2015 年 12 月 3 日
新增 Node.js 的入門演練。	2015 年 7 月 14 日
將兩個新的技術指南範例新增至 Cookbooks 101。	2015 年 7 月 14 日
新增對代理程式版本管理的支援。	2015 年 6 月 23 日
新增對代理程式版本的管理支援。	2015 年 6 月 24 日
新增對自訂 Windows AMI 的支援。	2015 年 6 月 22 日

描述	日期
新增三個新的最佳實務主題。	2015 年 6 月 11 日
新增對 Windows 堆疊的支援。	2015 年 5 月 18 日
新增最佳實務章節。	2014 年 12 月 15 日
新增對 Elastic Load Balancing 連線耗盡和自訂關機逾時的支援。	2014 年 12 月 15 日
新增了對註冊在 AWS OpsWorks Stacks 外部建立的執行個體的支援。	2014 年 12 月 9 日
新增對 Amazon SWF 的支援。	2014 年 9 月 4 日
新增將環境變數與應用程式和擴展的 Cookbooks 101 建立關聯的支援。	2014 年 7 月 16 日
新增 Cookbooks 101，其為實作技術指南的教學簡介。	2014 年 7 月 16 日
新增對 CloudTrail 的支援。	2014 年 6 月 4 日
新增對 Amazon RDS 的支援。	2014 年 5 月 14 日
新增對 Chef 11.10 和 Berkshelf 的支援。	2014 年 3 月 27 日
新增對 Amazon EBS PIOPS 磁碟區的支援。	2013 年 12 月 16 日
新增以資源為基礎的許可。	2013 年 12 月 5 日
新增資源管理。	2013 年 10 月 7 日
新增對 VPC 的支援。	2013 年 8 月 29 日
新增對自訂 AMI 和 Chef 11.4 的支援。	2013 年 7 月 24 日
新增主控台對每個執行個體之多個 layer 的支援。	2013 年 7 月 1 日
新增對 Amazon EBS 後端執行個體、Elastic Load Balancing 和 Amazon CloudWatch 監控的支援。	2013 年 5 月 14 日
AWS OpsWorks Stacks 使用者指南的初始版本。	2013 年 2 月 18 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。