使用者指南

AWS Amplify 託管

AWS Amplify 託管: 使用者指南

Table of Contents

什麼是 AWS Amplify 託管？ 1
支援的架構 1
Amplify 託管功能 2
Amplify 託管入門 2
建置後端 2
Amplify 託管定價 3
入門教學課程 4
部署 Next.js 應用程式 4
步驟 1：連接儲存庫 4
步驟 2：確認建置設定 5
步驟 3：部署應用程式 6
步驟 4：（選用） 清除資源 6
將功能新增至您的應用程式 6
部署 Nuxt.js 應用程式 7
部署 Astro.js 應用程式 8
部署 SvelteKit 應用程式 10
部署 SSR 應用程式 13
Next.js 14
Next.js 功能支援 14
將 Next.js SSR 應用程式部署至 Amplify 16
將 Next.js 11 SSR 應用程式遷移至 Amplify 託管運算 19
將 SSR 功能新增至靜態 Next.js 應用程式 21
讓伺服器端執行時間可存取環境變數 22
在單一儲存庫中部署 Next.js 應用程式 25
Nuxt.js 25
Astro.js 26
SvelteKit 27
將 SSR 應用程式部署至 Amplify 28
SSR 支援的功能 29
Next.js 應用程式的 Node.js 版本支援 29
SSR 應用程式的影像最佳化 30
適用於 SSR 應用程式的 Amazon CloudWatch Logs 30
Amplify Next.js 11 SSR 支援 31
對 SSR 部署進行故障診斷 38

進階：開放原始碼轉接器 38
部署規格 38
部署 Express 伺服器 59
架構作者的影像最佳化 65
對任何 SSR 架構使用開放原始碼轉接器 71
從 部署靜態網站 S3 73
從 Amplify 主控台部署 74
使用 SDKs 建立要部署的儲存貯體政策 74
更新從 儲存S3貯體部署的靜態網站 76
更新S3部署以使用儲存貯體和字首，而非 .zip 檔案 77
不使用 Git 部署 78
拖放手動部署 78
Amazon S3 或 URL 手動部署 79
針對手動部署的 Amazon S3 儲存貯體存取進行故障診斷 80
建置設定和組態 81
設定建置設定 81
組建規格參考 82
編輯建置規格 84
Monorepo 建置設定 90
自訂建置映像 96
設定應用程式的自訂建置映像 97
在建置映像中使用特定套件和相依性版本 98
設定建置執行個體 99
了解建置執行個體類型 99
在 Amplify 主控台中設定建置執行個體類型 100
設定應用程式的堆積記憶體以利用大型執行個體類型 102
傳入 Webhook 103
組建通知 104
設定電子郵件通知 104
連接自訂網域 105
了解 DNS 術語和概念 106
DNS 術語 106
DNS 驗證 107
自訂網域啟用程序 107
使用 SSL/TLS 憑證 108
新增由 Amazon Route 53 管理的自訂網域 109

新增由第三方 DNS 供應商管理的自訂網域 110
更新由 GoDaddy 管理之網域的 DNS 記錄 115
更新網域的 SSL/TLS 憑證 118
管理子網域 119
僅新增子網域 119
新增多層級子網域 119
新增或編輯子網域 120
設定萬用字元子網域 120
新增或刪除萬用字元子網域 121
設定 Amazon Route 53 自訂網域的自動子網域 121
具有子網域的 Web 預覽 122
對自訂網域進行故障診斷 122
託管網站的防火牆支援 123
AWS WAF 使用主控台啟用 124
AWS WAF 從應用程式移除 128
AWS WAF 使用 CDK 啟用 128
Amplify 如何與 整合 AWS WAF 130
Amplify Web ACL 資源政策 131
防火牆定價 131
功能分支部署 132
具有完整堆疊 Amplify Gen 2 應用程式的團隊工作流程 132
具有完整堆疊 Amplify Gen 1 應用程式的團隊工作流程 133
功能分支工作流程 133
GitFlow 工作流程 137
每位開發人員的沙盒 138
模式型功能分支部署 140
連接到自訂網域之應用程式的模式型功能分支部署 141
Amplify 組態的自動建置時間產生 （僅限第 1 代應用程式） 141
條件式後端建置 （僅限第 1 代應用程式） 142
跨應用程式使用 Amplify 後端 （僅限第 1 代應用程式） 143
建立新應用程式時重複使用後端 143
將分支連線至現有應用程式時重複使用後端 144
編輯現有的前端以指向不同的後端 144
建置後端 146
為 Gen 2 應用程式建立後端 146
為 Gen 1 應用程式建立後端 146

先決條件 146
步驟 1：部署前端 147
步驟 2：建立後端 148
步驟 3：將後端連接到前端 149
後續步驟 150
進階部署功能 152
密碼保護分支 152
提取請求預覽 153
啟用提取請求的 Web 預覽 154
具有子網域的 Web 預覽存取 155
End-to-end測試 155
將 Cypress 測試新增至現有的 Amplify 應用程式 155
關閉 Amplify 應用程式或分支的測試 157
一鍵式部署按鈕 158
將部署至 Amplify 託管按鈕新增至儲存庫或部落格 158
重新導向和重寫 160
了解 Amplify 支援的重新導向 160
了解重新導向的順序 161
了解 Amplify 如何轉送查詢參數 161
建立和編輯重新導向 162
重新導向和重寫範例 163
簡單重新導向和重寫 164
單一頁面 Web 應用程式 (SPA) 的重新導向 167
反向代理重寫 168
追蹤斜線和乾淨的 URLs 168
預留位置 169
查詢字串和路徑參數 169
區域型重新導向 171
在重新導向和重寫中使用萬用字元表達式 171
環境變數 173
Amplify 環境變數參考 173
前端架構環境變數 178
設定環境變數 178
使用社交登入的身分驗證參數建立新的後端環境 179
管理環境秘密 180
使用 AWS Systems Manager 設定 Amplify Gen 1 應用程式的環境秘密 180

存取 Gen 1 應用程式的環境秘密 180
Amplify 環境秘密參考 181
自訂標頭 182
YAML 參考 182
設定自訂標頭 183
安全性自訂標頭範例 185
設定快取控制自訂標頭 185
遷移自訂標頭 186
Monorepo 自訂標頭 187
管理快取組態 188
Amplify 如何套用快取組態 189
了解 Amplify 的受管快取政策 190
管理快取金鑰 Cookie 193
從快取金鑰中包含或排除 Cookie 193
變更應用程式的快取金鑰 Cookie 組態 195
使用 Cache-Control 標頭來提高應用程式效能 195
偏斜保護 197
設定扭曲保護 198
扭曲保護的運作方式 199
X-Amplify-Dpl 標頭範例 199
監控應用程式 201
CloudWatch 指標和警示 201
支援的 CloudWatch 指標 201
存取 CloudWatch 指標 203
建立 CloudWatch 警示 204
存取 SSR 應用程式的 CloudWatch Logs 205
存取日誌 206
擷取應用程式的存取日誌 207
分析存取日誌 207
使用 記錄 Amplify API 呼叫 AWS CloudTrail 207
CloudTrail 中的 Amplify 資訊 208
了解 Amplify 日誌檔案項目 208
搭配應用程式使用 IAM 角色 212
新增服務角色以部署後端資源 212
在 IAM 主控台中建立 Amplify 服務角色 213
編輯服務角色的信任政策，以防止混淆代理人 213

新增 SSR 運算角色 214
在 IAM 主控台中建立 SSR 運算角色 215
將 IAM SSR 運算角色新增至 Amplify 應用程式 217
管理 IAM SSR 運算角色安全性 217
新增服務角色以存取 CloudWatch Logs 218
Git 儲存庫的統一 Webhook 220
統一 Webhook 入門 220
安全 222
身分和存取權管理 222
目標對象 223
使用身分驗證 223
使用政策管理存取權 224
Amplify 如何與 IAM 搭配使用 225
身分型政策範例 230
AWS 受管政策 232
疑難排解 243
資料保護 244
靜態加密 245
傳輸中加密 245
加密金鑰管理 246
合規驗證 246
基礎設施安全性 246
日誌記錄和監控 246
預防跨服務混淆代理人 247
安全最佳實務 249
搭配 Amplify 預設網域使用 Cookie 249
配額 250
疑難排解 252
一般問題 252
HTTP 429 狀態碼 （太多請求） 252
Amplify 主控台不會顯示我的應用程式的建置狀態和上次更新時間 253
未針對新的提取請求建立 Web 預覽 253
我的手動部署在 Amplify 主控台中停滯待定狀態 254
我需要更新應用程式的 Node.js 版本 255
AL2023 組建映像 256
我想要使用 Python 執行時間執行 Amplify 函數 256

我想要執行需要超級使用者或根權限的命令 257
組建問題 257
我的儲存庫的新遞交不會觸發 Amplify 組建 257
建立新應用程式時，Amplify 主控台中不會列出我的儲存庫名稱 258
我的建置失敗並發生錯誤 Cannot find module aws-exports（僅限第 1 代應用程式） 258
我想要覆寫建置逾時 258
自訂網域 258
我需要驗證我的 CNAME 是否解析 259
由第三方託管的網域卡在待驗證狀態 260
使用 Amazon Route 53 託管的網域停滯在待驗證狀態 260
具有多層級子網域的應用程式卡在待驗證狀態 261
我的 DNS 供應商不支援具有完整網域名稱的記錄 261
我收到 CNAMEAlreadyExistsException 錯誤 262
我收到其他需要驗證的錯誤 263
我在 CloudFront URL 上收到 404 錯誤 263
我在造訪我的網域時收到 SSL 憑證或 HTTPS 錯誤 263
網域重新導向中不支援的路徑元件 264
我收到跨帳戶網域關聯的 400 錯誤 265
伺服器端轉譯 (SSR) 265
我需要使用架構轉接器的協助 265
Edge API 路由導致我的 Next.js 建置失敗 265
隨需增量靜態復原不適用於我的應用程式 266
我應用程式的建置輸出超過允許的大小上限 266
我的建置失敗，並出現記憶體不足錯誤 36
我應用程式的 HTTP 回應大小太大 268
如何在本機測量運算應用程式的啟動時間？ 36
我的建置失敗，並出現已棄用的 Node.js 版本錯誤 269
重新導向和重寫 270
即使使用 SPA 重新導向規則，也會拒絕特定路由的存取。 270
我想要設定 API 的反向代理 270
快取 271
我想要減少應用程式快取的大小 271
我想要停用從應用程式的快取讀取 271
設定 GitHub 存取 271
為新部署安裝和授權 Amplify GitHub 應用程式 272

將現有OAuth應用程式遷移至 Amplify GitHub 應用程式 273
為 CloudFormation、 CLI 和 SDK 部署設定 Amplify GitHub 應用程式 274
使用 Amplify GitHub 應用程式設定 Web 預覽 275
AWS Amplify 託管參考 276
AWS CloudFormation 支援 276
AWS Command Line Interface 支援 276
資源標記支援 276
Amplify 託管 API 276
文件歷史紀錄 277
..... cclxxxvii

歡迎使用 AWS Amplify 託管

Amplify Hosting 提供以 Git 為基礎的工作流程，可透過持續部署來託管全堆疊無伺服器 Web 應用程式。Amplify 會將您的應用程式部署到 AWS 全球內容交付網路 (CDN)。本使用者指南提供開始使用 Amplify 託管所需的資訊。

支援的架構

Amplify Hosting 支援許多常見的 SSR 架構、單頁應用程式 (SPA) 架構和靜態網站產生器，包括下列項目。

SSR 架構

- Next.js
- Nuxt
- Astro 使用社群轉接器
- SvelteKit 使用社群轉接器
- 具有自訂轉接器的任何 SSR 架構

SPA 架構

- React
- Angular
- Vue.js
- Ionic
- Ember

靜態網站產生器

- Eleventy
- Gatsby
- Hugo
- Jekyll

- VuePress

Amplify 託管功能

特徵分支

連接新的分支，為您的前端和後端管理生產和預備環境。

自訂網域

將您的應用程式連接到自訂網域。

提取請求預覽

在程式碼檢閱期間預覽變更。

End-to-end測試

透過end-to-end測試改善您的應用程式品質。

受密碼保護的分支

使用密碼保護您的 Web 應用程式，使得您可以處理新功能，而不需讓它們公開存取。

重新導向和重寫

設定重寫和重新導向，以維護 SEO 排名，並根據用戶端應用程式需求路由流量。

原子部署

原子部署可確保您的 Web 應用程式只會在整個部署完成後更新，以消除維護時段。如此便可減少檔案無法正確上傳的情況。

Amplify 託管入門

若要開始使用 Amplify 託管，請參閱開始將應用程式部署到 Amplify 託管教學課程。完成教學課程後，您將了解如何在 Git 儲存庫 (GitHub、BitBucket、GitLab 或 AWS CodeCommit) 中連接 Web 應用程式，並將其部署到具有連續部署的 Amplify 託管。

建置後端

AWS Amplify Gen 2 推出以 TypeScript 為基礎的程式碼優先開發人員體驗來定義後端。若要了解如何使用 Amplify Gen 2 建置後端並連接至您的應用程式，請參閱 Amplify 文件中的建置和連接後端。

若要進一步了解 Amplify Gen 2的程式碼優先方法，請參閱 AWS Workshop Studio 網站上的 Amplify Gen 2 研討會。在此全面的教學課程中，您會使用 React 和 Next.js 建置無伺服器應用程式，並了解如何使用 Amplify Gen 2 Data and Auth 程式庫和 Amplify UI 程式庫來新增應用程式功能。

如果您要尋找使用 CLI 和 Amplify Studio 建置 Gen 1 應用程式後端的文件，請參閱 Gen 1 Amplify 文件中的建置和連線後端。

Amplify 託管定價

AWS Amplify 只會向您收取使用量的費用。如需詳細資訊，請參閱 AWS Amplify 定價。

開始將應用程式部署到 Amplify 託管

為了協助您了解 Amplify 託管的運作方式，下列教學課程會逐步引導您建置和部署使用 Amplify 支援的常見 SSR 架構建立的應用程式。

教學

- 將 Next.js 應用程式部署至 Amplify 託管
- 將 Nuxt.js 應用程式部署至 Amplify 託管
- 將 Astro.js 應用程式部署至 Amplify 託管
- 將 SvelteKit 應用程式部署至 Amplify 託管

將 Next.js 應用程式部署至 Amplify 託管

本教學課程會逐步引導您從 Git 儲存庫建置和部署 Next.js 應用程式。

開始本教學課程之前，請先完成下列先決條件。

註冊 AWS 帳戶

如果您還不是 AWS 客戶，則需要遵循線上說明來建立 AWS 帳戶。註冊可讓您存取 Amplify 和其他可與應用程式搭配使用 AWS 的服務。

建立應用程式

使用 Next.js 文件中的 create-next-app 說明，建立用於本教學課程的基本 Next.js 應用程式。

建立 Git 儲存庫

Amplify 支援 GitHub、Bitbucket、GitLab 和 AWS CodeCommit。將您的`create-next-app`應用程式推送到您的 Git 儲存庫。

步驟 1：連接 Git 儲存庫

在此步驟中，您將 Git 儲存庫中的 Next.js 應用程式連線至 Amplify Hosting。

連接 Git 儲存庫中的應用程式

1. 開啟 Amplify 主控台。
2. 如果您要在目前區域中部署第一個應用程式，預設會從AWS Amplify服務頁面開始。

選擇頁面頂端的部署應用程式。

3. 在開始使用 Amplify 建置頁面上，選擇您的 Git 儲存庫提供者，然後選擇下一步。

   對於 GitHub 儲存庫，Amplify 會使用 GitHub 應用程式功能來授權 Amplify 存取。如需安裝和授權 GitHub 應用程式的詳細資訊，請參閱設定對 GitHub 儲存庫的 Amplify 存取權。

   > **Note**
   >
   > 使用 Bitbucket、GitLab 或 授權 Amplify 主控台之後 AWS CodeCommit，Amplify 會從儲存庫提供者擷取存取字符，但不會將字符存放在 AWS 伺服器上。Amplify 只會使用安裝在特定儲存庫中的部署金鑰來存取您的儲存庫。

4. 在新增儲存庫分支頁面上，執行下列動作：

   a. 選取要連線的儲存庫名稱。
   b. 選取要連線的儲存庫分支名稱。
   c. 選擇下一步。

步驟 2：確認建置設定

Amplify 會自動偵測要為您部署的分支執行的建置命令序列。在此步驟中，您將檢閱並確認建置設定。

確認應用程式的建置設定

1. 在應用程式設定頁面上，找到建置設定區段。

   驗證前端建置命令和建置輸出目錄是否正確。對於此 Next.js 範例應用程式，建置輸出目錄設定為 `.next`。

2. 新增服務角色的程序會因您要建立新角色或使用現有角色而有所不同。
   - 若要建立新的角色：
     - 選擇建立並使用新的服務角色。
   - 若要使用現有角色：
     a. 選擇使用現有角色。
     b. 在服務角色清單中，選取要使用的角色。
3. 選擇下一步。

步驟 3：部署應用程式

在此步驟中，您將應用程式部署至 AWS 全球內容交付網路 (CDN)。

儲存和部署應用程式

1. 在檢閱頁面上，確認您的儲存庫詳細資訊和應用程式設定正確無誤。
2. 選擇 Save and deploy (儲存並部署)。您的前端建置通常需要 1 到 2 分鐘，但可能會因應用程式的大小而有所不同。
3. 部署完成時，您可以使用`amplifyapp.com`預設網域的連結來檢視您的應用程式。

> **Note**
>
> 為了增強 Amplify 應用程式的安全性，在公有字尾清單 (PSL) 中註冊 amplifyapp.com 網域。為了提高安全性，如果您需要在 Amplify 應用程式的預設網域名稱中設定敏感 Cookie，建議您使用具有`__Host-`字首的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

步驟 4：（選用）清除資源

如果您不再需要為教學課程部署的應用程式，可以將其刪除。此步驟協助確保您不會為了未使用的資源而付費。

刪除應用程式

1. 從導覽窗格中的應用程式設定選單中，選擇一般設定。
2. 在一般設定頁面上，選擇刪除應用程式。
3. 在確認視窗中，輸入 **`delete`**。然後選擇刪除應用程式。

將功能新增至您的應用程式

現在您已將應用程式部署到 Amplify，您可以探索下列一些可供託管應用程式使用的功能。

環境變數

應用程式通常需要在執行時間取得組態資訊。這些組態可以是資料庫連線詳細資訊、API 金鑰或參數。環境變數提供在建置時公開這些組態的方法。如需詳細資訊，請參閱環境變數。

自訂網域

在本教學課程中，Amplify 會將您的應用程式託管在具有 URL 的預設amplifyapp.com網域上，例如 https://branch-name.d1m7bkiki6tdw1.amplifyapp.com。當您將應用程式連線至自訂網域時，使用者會看到您的應用程式託管在自訂 URL 上，例如 https://www.example.com。如需詳細資訊，請參閱設定自訂網域。

提取請求預覽

Web 提取請求預覽可讓團隊預覽提取請求 (PRs) 的變更，然後再將程式碼合併到生產或整合分支。如需詳細資訊，請參閱提取請求的 Web 預覽。

管理多個環境

若要了解 Amplify 如何與特徵分支和 GitFlow 工作流程搭配使用以支援多個部署，請參閱特徵分支部署和團隊工作流程。

將 Nuxt.js 應用程式部署至 Amplify 託管

使用以下指示將 Nuxt.js 應用程式部署到 Amplify Hosting。Nuxt 已使用 Nitro 伺服器實作預設轉接器。這可讓您部署 Nuxt 專案，無需任何其他組態。

將 Nuxt 應用程式部署至 Amplify 託管

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 在所有應用程式頁面上，選擇建立新應用程式。
3. 在開始使用 Amplify 建置頁面上，選擇您的 Git 儲存庫提供者，然後選擇下一步。
4. 在新增儲存庫分支頁面上，執行下列動作：
   a. 選取要連線的儲存庫名稱。
   b. 選取要連線的儲存庫分支名稱。
   c. 選擇下一步。
5. 如果您希望 Amplify 能夠將應用程式日誌交付至 Amazon CloudWatch Logs，您必須在 主控台中明確啟用此功能。開啟進階設定區段，然後在伺服器端轉譯 (SSR) 部署區段中選擇啟用 SSR 應用程式日誌。
6. 選擇下一步。
7. 在檢閱頁面上，選擇儲存並部署。

將 Astro.js 應用程式部署至 Amplify 託管

使用以下指示將 Astro.js 應用程式部署到 Amplify Hosting。您可以使用現有的應用程式，或使用 Astro 提供的其中一個官方範例來建立入門應用程式。若要建立入門應用程式，請參閱 Astro 文件中的使用佈景主題或入門範本。

若要使用 SSR 將 Astro 網站部署至 Amplify Hosting，您必須將轉接器新增至應用程式。我們不會維護 Astro 架構的 Amplify 擁有轉接器。本教學課程使用社群成員建立的`astro-aws-amplify`轉接器。此轉接器可在 GitHub 網站上的 github.com/alexnguyennz/astro-aws-amplify：// 取得。 AWS 不會維護此轉接器。

將 Astro 應用程式部署至 Amplify 託管

1. 在本機電腦上，導覽至要部署的 Astro 應用程式。
2. 若要安裝轉接器，請開啟終端機視窗並執行下列命令。此範例使用社群轉接器，網址為 https：//github.com/alexnguyennz/astro-aws-amplify。您可以將 *`astro-aws-amplify`* 取代為您正在使用的轉接器名稱。

   ```
   npm install astro-aws-amplify
   ```

3. 在 Astro 應用程式的專案資料夾中，開啟 `astro.config.mjs` 檔案。更新 檔案以新增轉接器。檔案看起來應該如下所示。

   ```
   import { defineConfig } from 'astro/config';
   import mdx from '@astrojs/mdx';
   import awsAmplify from 'astro-aws-amplify';

   import sitemap from '@astrojs/sitemap';

   // https://astro.build/config
   export default defineConfig({
     site: 'https://example.com',
     integrations: [mdx(), sitemap()],
     adapter: awsAmplify(),
     output: 'server',
   });
   ```

4. 遞交變更並將專案推送到您的 Git 儲存庫。

   現在您已準備好將 Astro 應用程式部署至 Amplify。

5. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
6. 在所有應用程式頁面上，選擇建立新應用程式。
7. 在開始使用 Amplify 建置頁面上，選擇您的 Git 儲存庫提供者，然後選擇下一步。
8. 在新增儲存庫分支頁面上，執行下列動作：

   a. 選取要連線的儲存庫名稱。

   b. 選取要連線的儲存庫分支名稱。

   c. 選擇下一步。
9. 在應用程式設定頁面上，找到建置設定區段。對於建置輸出目錄，輸入 **.amplify-hosting**。
10. 您也必須在建置規格中更新應用程式的前端建置命令。若要開啟建置規格，請選擇編輯 YML 檔案。
11. 在 amplify.yml檔案中，找到前端建置命令區段。輸入 **mv node_modules ./.amplify-hosting/compute/default**

    您的建置設定檔案看起來應該如下所示。

```
version: 1
frontend:
    phases:
        preBuild:
            commands:
                - 'npm ci --cache .npm --prefer-offline'
        build:
            commands:
                - 'npm run build'
                - 'mv node_modules ./.amplify-hosting/compute/default'
    artifacts:
        baseDirectory: .amplify-hosting
        files:
            - '**/*'
    cache:
        paths:
            - '.npm/**/*'
```

12. 選擇儲存。
13. 如果您希望 Amplify 能夠將應用程式日誌交付至 Amazon CloudWatch Logs，您必須在 主控台中明確啟用此功能。開啟進階設定區段，然後在伺服器端轉譯 (SSR) 部署區段中選擇啟用 SSR 應用程式日誌。

14. 選擇下一步。
15. 在檢閱頁面上，選擇儲存並部署。

將 SvelteKit 應用程式部署至 Amplify 託管

使用下列指示將 SvelteKit 應用程式部署至 Amplify Hosting。您可以使用自己的應用程式，或建立入門應用程式。如需詳細資訊，請參閱 SvelteKit 文件中的建立專案。

若要使用 SSR 將 SvelteKit 應用程式部署至 Amplify Hosting，您必須將轉接器新增至專案。我們不會維護 SvelteKit 架構的 Amplify 擁有轉接器。在此範例中，我們使用社群成員`amplify-adapter`建立的 。轉接器可在 GitHub 網站上的 https：//github.com/gzimbron/amplify-adapter 取得。 AWS 不會維護此轉接器。

將 SvelteKit 應用程式部署至 Amplify 託管

1. 在本機電腦上，導覽至要部署的 SvelteKit 應用程式。
2. 若要安裝轉接器，請開啟終端機視窗並執行下列命令。此範例使用社群轉接器，網址為 https：//github.com/gzimbron/amplify-adapter。如果您使用的是不同的社群轉接器，請以轉接器的名稱取代 *amplify-adapter*。

```
npm install amplify-adapter
```

3. 在 SvelteKit 應用程式的專案資料夾中，開啟 `svelte.config.js` 檔案。編輯 檔案以使用 ，`amplify-adapter`或以轉接器的名稱取代 *'amplify-adapter'*。檔案看起來應該如下所示。

```
import adapter from 'amplify-adapter';
import { vitePreprocess } from '@sveltejs/vite-plugin-svelte';


/** @type {import('@sveltejs/kit').Config} */
const config = {
        // Consult https://kit.svelte.dev/docs/integrations#preprocessors
        // for more information about preprocessors
        preprocess: vitePreprocess(),

        kit: {
                // adapter-auto only supports some environments, see https://
kit.svelte.dev/docs/adapter-auto for a list.
```

```
                // If your environment is not supported, or you settled on a
 specific environment, switch out the adapter.
                // See https://kit.svelte.dev/docs/adapters for more information
 about adapters.
                adapter: adapter()
        }
};

export default config;
```

4. 遞交變更並將應用程式推送到您的 Git 儲存庫。
5. 現在您已準備好將 SvelteKit 應用程式部署至 Amplify。

   登入 AWS 管理主控台 並開啟 Amplify 主控台。
6. 在所有應用程式頁面上，選擇建立新應用程式。
7. 在開始使用 Amplify 建置頁面上，選擇您的 Git 儲存庫提供者，然後選擇下一步。
8. 在新增儲存庫分支頁面上，執行下列動作：

   a. 選取要連線的儲存庫名稱。

   b. 選取要連線的儲存庫分支名稱。

   c. 選擇下一步。
9. 在應用程式設定頁面上，找到建置設定區段。對於建置輸出目錄，輸入 **build**。
10. 您也必須在建置規格中更新應用程式的前端建置命令。若要開啟建置規格，請選擇編輯 YML 檔案。
11. 在 amplify.yml檔案中，找到前端建置命令區段。輸入 **- cd build/compute/default/**和 **- npm i --production**。

    您的建置設定檔案看起來應該如下所示。

```
version: 1
frontend:
    phases:
        preBuild:
            commands:
                - 'npm ci --cache .npm --prefer-offline'
        build:
            commands:
                - 'npm run build'
                - 'cd build/compute/default/'
```

```
            - 'npm i --production'

artifacts:
    baseDirectory: build
    files:
        - '**/*'
cache:
    paths:
        - '.npm/**/*'
```

12. 選擇儲存。
13. 如果您希望 Amplify 能夠將應用程式日誌交付至 Amazon CloudWatch Logs，您必須在 主控台中明確啟用此功能。開啟進階設定區段，然後在伺服器端轉譯 (SSR) 部署區段中選擇啟用 SSR 應用程式日誌。
14. 選擇下一步。
15. 在檢閱頁面上，選擇儲存並部署。

使用 Amplify Hosting 部署伺服器端轉譯應用程式

您可以使用 AWS Amplify 部署和託管使用伺服器端轉譯 (SSR) 的 Web 應用程式。Amplify Hosting 會自動偵測使用 Next.js 架構建立的應用程式，而且您不需要在 中執行任何手動組態。 AWS 管理主控台

Amplify 也支援具有開放原始碼建置轉接器的任何 Javascript 型 SSR 架構，可將應用程式的建置輸出轉換為 Amplify Hosting 預期的目錄結構。例如，您可以透過安裝可用的轉接器，部署使用 Nuxt、Astro 和 SvelteKit 架構建立的應用程式。

進階使用者可以使用部署規格來建立建置轉接器或設定建置後指令碼。

您可以使用最少的組態，將下列架構部署至 Amplify 託管。

Next.js

- Amplify 支援 Next.js 15 應用程式，而不需要轉接器。若要開始使用，請參閱Next.js 的 Amplify 支援。

Nuxt.js

- Amplify 支援具有預設轉接器的 Nuxt.js 應用程式部署。若要開始使用，請參閱Nuxt.js 的 Amplify 支援。

Astro.js

- Amplify 支援使用社群轉接器的 Astro.js 應用程式部署。若要開始使用，請參閱Amplify 支援 Astro.js。

SvelteKit

- Amplify 支援使用社群轉接器的 SvelteKit 應用程式部署。若要開始使用，請參閱對 SvelteKit 的 Amplify 支援。

開放原始碼轉接器

- 使用開放原始碼轉接器 - 如需使用不在上述清單中之任何轉接器的說明，請參閱 對任何 SSR 架構使用開放原始碼轉接器 。
- 建置架構轉接器 - 想要整合架構提供之功能的架構作者，可以使用 Amplify 託管部署規格來設定您的建置輸出，以符合 Amplify 預期的結構。如需詳細資訊，請參閱使用 Amplify 託管部署規格來設定建置輸出。
- 設定建置後指令碼 - 您可以使用 Amplify 託管部署規格，視需要針對特定案例操作建置輸出。如需詳細資訊，請參閱使用 Amplify 主機部署規格來設定建置輸出。如需範例，請參閱 使用部署資訊清單部署 Express 伺服器。

主題

- Next.js 的 Amplify 支援
- Nuxt.js 的 Amplify 支援
- Amplify 支援 Astro.js
- 對 SvelteKit 的 Amplify 支援
- 將 SSR 應用程式部署至 Amplify
- SSR 支援的功能
- 對 SSR 部署進行故障診斷
- 進階：開放原始碼轉接器

Next.js 的 Amplify 支援

Amplify 支援部署和託管使用 Next.js 建立的伺服器端轉譯 (SSR) Web 應用程式。Next.js 是使用 JavaScript 開發 SPAs 的 React 架構。您可以透過 Next.js 15 部署使用 Next.js 版本建置的應用程式，並具有影像最佳化和中介軟體等功能。

開發人員可以使用 Next.js 在單一專案中結合靜態網站產生 (SSG) 和 SSR。SSG 頁面會在建置時渲染，而 SSR 頁面會在請求時渲染。

渲染可以改善效能和搜尋引擎最佳化。由於 Next.js 會呈現伺服器上的所有頁面，因此每個頁面的 HTML 內容會在到達用戶端的瀏覽器時就緒。此內容也可以更快地載入。更快的載入時間可改善最終使用者的網站體驗，並正面影響網站的 SEO 排名。Prerendering 也透過讓搜尋引擎機器人輕鬆尋找和編目網站的 HTML 內容來改善 SEO。

Next.js 提供內建的分析支援，可測量各種效能指標，例如 Time to first byte (TTFB) 和 First contentful paint (FCP)。如需 Next.js 的詳細資訊，請參閱 Next.js 網站上的入門。

Next.js 功能支援

Amplify 託管運算可完整管理使用 Next.js 版本 12 到 15 建置的應用程式的伺服器端轉譯 (SSR)。

如果您在 2022 年 11 月發行 Amplify 託管運算之前，已將 Next.js 應用程式部署至 Amplify，您的應用程式會使用 Amplify 先前的 SSR 供應商 Classic （僅限 Next.js 11)。Amplify 託管運算不支援使用 Next.js 第 11 版或更早版本建立的應用程式。我們強烈建議您將 Next.js 11 應用程式遷移至 Amplify 託管運算受管 SSR 供應商。

下列清單說明 Amplify 託管運算 SSR 供應商支援的特定功能。

支援的功能

- 伺服器端轉譯頁面 (SSR)
- 靜態頁面
- API 路由
- 動態路由
- 擷取所有路由
- SSG （靜態產生）
- 增量靜態再生 (ISR)
- 國際化 (i18n) 子路徑路由
- 國際化 (i18n) 網域路由
- 國際化 (i18n) 自動地區設定偵測
- 中介軟體
- 環境變數
- 映像最佳化
- Next.js 13 應用程式目錄

不支援的功能

- Edge API Routes (不支援邊緣中介軟體)
- 隨需增量靜態再生 (ISR)
- Next.js 串流
- 在靜態資產和最佳化映像上執行中介軟體
- 在回應後使用 執行程式碼 `unstable_after`（使用 Next.js 15 發行實驗功能）

Next.js 影像

影像的最大輸出大小不得超過 4.3 MB。您可以將較大的影像檔案存放在某處，並使用 Next.js Image 元件來調整大小，並將其最佳化為 Webp 或 AVIF 格式，然後將其做為較小的大小。

請注意，Next.js 文件建議您安裝 Sharp 映像處理模組，讓映像最佳化在生產環境中正常運作。不過，Amplify 部署不需要這麼做。Amplify 會自動為您部署 Sharp。

將 Next.js SSR 應用程式部署至 Amplify

根據預設，Amplify 會使用 Amplify Hosting 的運算服務部署新的 SSR 應用程式，並支援 Next.js 版本 12 到 15。Amplify 託管運算可完整管理部署 SSR 應用程式所需的資源。您在 2022 年 11 月 17 日之前部署的 Amplify 帳戶中的 SSR 應用程式正在使用 Classic（僅限 Next.js 11) SSR 供應商。

我們強烈建議您使用 Classic（僅限 Next.js 11) SSR 將應用程式遷移至 Amplify 託管運算 SSR 供應商。Amplify 不會為您執行自動遷移。您必須手動遷移應用程式，然後啟動新建置以完成更新。如需說明，請參閱將 Next.js 11 SSR 應用程式遷移至 Amplify 託管運算。

使用以下指示來部署新的 Next.js SSR 應用程式。

使用 Amplify 託管運算 SSR 供應商將 SSR 應用程式部署至 Amplify

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 在所有應用程式頁面上，選擇建立新應用程式。
3. 在開始使用 Amplify 建置頁面上，選擇您的 Git 儲存庫提供者，然後選擇下一步。
4. 在新增儲存庫分支頁面上，執行下列動作：
   a. 在最近更新的儲存庫清單中，選取要連線的儲存庫名稱。
   b. 在分支清單中，選取要連線的儲存庫分支名稱。
   c. 選擇下一步。
5. 應用程式需要 Amplify 代表您呼叫其他 服務時擔任的 IAM 服務角色。您可以允許 Amplify 託管運算自動為您建立服務角色，也可以指定您已建立的角色。
   - 若要允許 Amplify 自動建立角色並將其連接至您的應用程式：
     - 選擇建立並使用新的服務角色。
   - 若要連接您先前建立的服務角色：
     a. 選擇使用現有的服務角色。
     b. 從清單中選擇要使用的角色。
6. 選擇下一步。
7. 在檢閱頁面上，選擇儲存並部署。

Package.json 檔案設定

當您部署 Next.js 應用程式時，Amplify 會檢查`package.json`檔案中應用程式的建置指令碼，以判斷應用程式類型。

以下是 Next.js 應用程式的建置指令碼範例。建置指令碼`"next build"`表示應用程式同時支援 SSG 和 SSR 頁面。此建置指令碼也用於 Next.js 14 或更新版本的 SSG 專用應用程式。

```
"scripts": {
  "dev": "next dev",
  "build": "next build",
  "start": "next start"
},
```

以下是 Next.js 13 或更早版本的 SSG 應用程式的建置指令碼範例。組建指令碼`"next build && next export"`表示應用程式僅支援 SSG 頁面。

```
"scripts": {
  "dev": "next dev",
  "build": "next build && next export",
  "start": "next start"
},
```

擴增 Next.js SSR 應用程式的建置設定

檢查應用程式`package.json`的檔案後，Amplify 會檢查應用程式的建置設定。您可以在 Amplify 主控台或儲存庫根目錄中的 `amplify.yml` 檔案中儲存建置設定。如需詳細資訊，請參閱設定 Amplify 應用程式的建置設定。

如果 Amplify 偵測到您正在部署 Next.js SSR 應用程式，但沒有`amplify.yml`檔案存在，它會為應用程式產生 buildspec，並將 `baseDirectory`設定為 `.next`。如果您要部署存在 `amplify.yml` 檔案的應用程式， 檔案中的建置設定會覆寫 主控台中的任何建置設定。因此，您必須在 檔案中手動`baseDirectory`將 設定為 `.next` 。

以下是 設定為 之應用程式的建置設定範例`baseDirectory.next`。這表示建置成品適用於支援 SSG 和 SSR 頁面的 Next.js 應用程式。

```
version: 1
frontend:
```

```
  phases:
    preBuild:
      commands:
        - npm ci
    build:
      commands:
        - npm run build
  artifacts:
    baseDirectory: .next
    files:
      - '**/*'
  cache:
    paths:
      - node_modules/**/*
```

擴增 Next.js 13 或更早版本的 SSG 應用程式的建置設定

如果 Amplify 偵測到您正在部署 Next.js 13 或更早版本的 SSG 應用程式，它會為應用程式產生建置規格，並將 baseDirectory設定為 out。如果您要部署存在 amplify.yml 檔案的應用程式，則必須out在 檔案中手動baseDirectory將 設定為 。out 目錄是 Next.js 建立的預設資料夾，用於存放匯出的靜態資產。當您設定應用程式的建置規格設定時，請變更baseDirectory資料夾的名稱以符合應用程式的組態。

以下是應用程式建置設定的範例，其中 baseDirectory 設定為 ，out表示建置成品適用於僅支援 SSG 頁面的 Next.js 13 或更早版本應用程式。

```
version: 1
frontend:
  phases:
    preBuild:
      commands:
        - npm ci
    build:
      commands:
        - npm run build
  artifacts:
    baseDirectory: out
    files:
      - '**/*'
  cache:
    paths:
      - node_modules/**/*
```

Next.js 14 或更新版本 SSG 應用程式的 Amplify 組建設定

在 Next.js 第 14 版`output: 'export'`中，`next export`命令已棄用，並在 `next.config.js` 檔案中取代為 ，以啟用靜態匯出。如果您在主控台中部署僅限 Next.js 14 SSG 的應用程式，Amplify 會為應用程式產生建置規格，並將 `baseDirectory`設定為 `.next`。如果您要部署存在 `amplify.yml` 檔案的應用程式，您必須在 `.next` 檔案中手動`baseDirectory`將 設定為 。這是 Amplify 用於支援 SSG 和 SSR 頁面之 Next.js `WEB_COMPUTE`應用程式的相同`baseDirectory`設定。

以下是將 `baseDirectory`設為 之僅限 Next.js 14 SSG 應用程式的建置設定範例。 `.next`

```
version: 1
frontend:
  phases:
    preBuild:
      commands:
        - npm ci
    build:
      commands:
        - npm run build
  artifacts:
    baseDirectory: .next
    files:
      - '**/*'
  cache:
    paths:
      - node_modules/**/*
```

將 Next.js 11 SSR 應用程式遷移至 Amplify 託管運算

當您部署新的 Next.js 應用程式時，根據預設，Amplify 會使用最新支援的 Next.js 版本。目前，Amplify 託管運算 SSR 提供者支援 Next.js 第 15 版。

Amplify 主控台會偵測帳戶中在 2022 年 11 月發行 Amplify 託管運算服務之前部署的應用程式，並完整支援 Next.js 版本 12 到 15。主控台會顯示資訊橫幅，識別使用 Amplify 先前 SSR 供應商 Classic（僅限 Next.js 11) 部署的分支應用程式。我們強烈建議您將應用程式遷移至 Amplify 託管運算 SSR 供應商。

如果您要將託管的 Next.js 11 應用程式更新為 Next.js 12 或更新版本，您可能會在觸發部署時收到`"target" property is no longer supported`錯誤。在此情況下，您必須遷移至 Amplify 託管運算。

您必須同時手動遷移應用程式及其所有生產分支。應用程式不能同時包含 Classic （僅限 Next.js 11) 和 Next.js 12 或更新版本的分支。

使用下列指示將應用程式遷移至 Amplify 託管運算 SSR 供應商。

將應用程式遷移至 Amplify 託管運算 SSR 供應商

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇您要遷移的 Next.js 應用程式。

> **Note**
>
> 在 Amplify 主控台中遷移應用程式之前，您必須先更新應用程式的 package.json 檔案，以使用 Next.js 第 12 版或更新版本。

3. 在導覽窗格中，選擇應用程式設定，一般。
4. 在應用程式首頁上，如果應用程式已使用 Classic （僅限 Next.js 11) SSR 提供者部署分支，則主控台會顯示橫幅。 在橫幅上，選擇遷移。
5. 在遷移確認視窗中，選取三個陳述式，然後選擇遷移。
6. Amplify 將建置並重新部署您的應用程式以完成遷移。

還原 SSR 遷移

當您部署 Next.js 應用程式時，Amplify Hosting 會偵測應用程式中的設定，並設定應用程式的內部平台值。有三個有效的平台值。SSG 應用程式設定為平台值 `WEB`。使用 Next.js 第 11 版的 SSR 應用程式設定為平台值 `WEB_DYNAMIC`。Next.js 12 或更新版本的 SSR 應用程式設定為平台值 `WEB_COMPUTE`。

當您使用上一節中的指示遷移應用程式時，Amplify 會將應用程式的平台值從 變更為 `WEB_DYNAMIC` `WEB_COMPUTE`。遷移至 Amplify 託管運算完成後，您無法在主控台中還原遷移。若要還原遷移，您必須使用 AWS Command Line Interface 將應用程式的平台變更回 `WEB_DYNAMIC`。開啟終端機視窗並輸入下列命令，以您的唯一資訊更新應用程式 ID 和區域。

```
aws amplify update-app --app-id abcd1234 --platform WEB_DYNAMIC --region us-west-2
```

將 SSR 功能新增至靜態 Next.js 應用程式

您可以將 SSR 功能新增至使用 Amplify 部署的現有靜態 (SSG) Next.js 應用程式。在開始將 SSG 應用程式轉換為 SSR 的程序之前，請更新應用程式以使用 Next.js 第 12 版或更新版本，並新增 SSR 功能。然後，您將需要執行下列步驟。

1. 使用 AWS Command Line Interface 變更應用程式的平台類型。
2. 將服務角色新增至應用程式。
3. 在應用程式的建置設定中更新輸出目錄。
4. 更新應用程式`package.json`的檔案，以指出應用程式使用 SSR。

更新平台

平台類型有三個有效值。SSG 應用程式設定為平台類型 `WEB`。使用 Next.js 第 11 版的 SSR 應用程式設定為平台類型 `WEB_DYNAMIC`。對於使用 Amplify 託管運算管理的 SSR 部署至 Next.js 12 或更新版本的應用程式，平台類型設定為 `WEB_COMPUTE`。

當您將應用程式部署為 SSG 應用程式時，Amplify 會將平台類型設定為 `WEB`。使用 AWS CLI 將應用程式的平台變更為 `WEB_COMPUTE`。開啟終端機視窗並輸入下列命令，使用您唯一的應用程式 ID 和區域更新紅色文字。

```
aws amplify update-app --app-id abcd1234 --platform WEB_COMPUTE --region us-west-2
```

新增服務角色

服務角色是 Amplify 代表您呼叫其他 服務時擔任的 AWS Identity and Access Management (IAM) 角色。請依照下列步驟，將服務角色新增至已使用 Amplify 部署的 SSG 應用程式。

新增服務角色

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 如果您尚未在 Amplify 帳戶中建立服務角色，請參閱新增服務角色以完成此先決條件步驟。
3. 選擇您要新增服務角色的靜態 Next.js 應用程式。
4. 在導覽窗格中，選擇應用程式設定，一般。
5. 在應用程式詳細資訊頁面上，選擇編輯
6. 針對服務角色，選擇現有服務角色的名稱，或您在步驟 2 中建立的服務角色名稱。
7. 選擇儲存。

更新建置設定

使用 SSR 功能重新部署應用程式之前，您必須更新應用程式的建置設定，將輸出目錄設定為 `.next`。您可以在 Amplify 主控台或儲存庫中存放的 `amplify.yml` 檔案中編輯建置設定。如需詳細資訊，請參閱 設定 Amplify 應用程式的建置設定。

以下是 設定為 之應用程式的建置設定範例`baseDirectory.next`。

```
version: 1
frontend:
  phases:
    preBuild:
      commands:
        - npm ci
    build:
      commands:
        - npm run build
  artifacts:
    baseDirectory: .next
    files:
      - '**/*'
  cache:
    paths:
      - node_modules/**/*
```

更新 package.json 檔案

新增服務角色並更新建置設定後，請更新應用程式`package.json`的檔案。如下列範例所示，將建置指令碼設定為 ，`"next build"`表示 Next.js 應用程式同時支援 SSG 和 SSR 頁面。

```
"scripts": {
  "dev": "next dev",
  "build": "next build",
  "start": "next start"
},
```

Amplify 會偵測儲存庫中`package.json`檔案的變更，並使用 SSR 功能重新部署應用程式。

讓伺服器端執行時間可存取環境變數

Amplify 託管支援在 Amplify 主控台的專案組態中設定環境變數，以將環境變數新增至應用程式的建置。

不過，根據預設，Next.js 伺服器元件無法存取這些環境變數。此行為旨在保護您的應用程式在建置階段期間使用的環境變數中存放的任何秘密。

若要讓 Next.js 可存取特定環境變數，您可以修改 Amplify 建置規格檔案，以在 Next.js 辨識的環境檔案中設定它們。這可讓 Amplify 在建置應用程式之前載入這些環境變數。

> **Important**
>
> 強烈建議您不要在環境變數中存放任何登入資料、秘密或敏感資訊，因為任何可存取部署成品的使用者都可以讀取它們。
> 若要讓您的 SSR 運算函數存取 AWS 資源，建議使用 IAM 角色。

下列建置規格範例示範如何在建置命令區段中新增環境變數。

```
version: 1
frontend:
  phases:
    preBuild:
      commands:
        - npm ci
    build:
      commands:
        - env | grep -e API_BASE_URL >> .env.production
        - env | grep -e NEXT_PUBLIC_ >> .env.production
        - npm run build
  artifacts:
    baseDirectory: .next
    files:
      - '**/*'
  cache:
    paths:
      - node_modules/**/*
      - .next/cache/**/*
```

在此範例中，建置命令區段包含兩個命令，在應用程式建置執行之前將環境變數寫入 `.env.production` 檔案。Amplify Hosting 可讓您的應用程式在應用程式接收流量時存取這些變數。

上述範例中建置命令區段的下列行示範如何從建置環境取得特定變數，並將其新增至 `.env.production` 檔案。

```
- env | grep -e API_BASE_URL -e APP_ENV >> .env.production
```

如果您的建置環境中存在變數，.env.production檔案將包含下列環境變數。

```
API_BASE_URL=localhost
APP_ENV=dev
```

上述範例中建置命令區段的下列行示範如何將具有特定字首的環境變數新增至 .env.production 檔案。在此範例中，NEXT_PUBLIC_會新增具有 字首的所有變數。

```
- env | grep -e NEXT_PUBLIC_ >> .env.production
```

如果組建環境中有多個字NEXT_PUBLIC_首為 的變數，則.env.production檔案看起來會類似以下內容。

```
NEXT_PUBLIC_ANALYTICS_ID=abcdefghijk
NEXT_PUBLIC_GRAPHQL_ENDPOINT=uowelalsmlsadf
NEXT_PUBLIC_FEATURE_FLAG=true
```

monorepos 的 SSR 環境變數

如果您要在單一儲存庫中部署 SSR 應用程式，並想要讓 Next.js 可存取特定環境變數，則必須在.env.production檔案前面加上應用程式根目錄。下列範例建置規格適用於 Nx monorepo 中的 Next.js 應用程式，示範如何在建置命令區段中新增環境變數。

```
version: 1
applications:
  - frontend:
      phases:
        preBuild:
          commands:
            - npm ci
        build:
          commands:
            - env | grep -e API_BASE_URL -e APP_ENV >> apps/app/.env.production
            - env | grep -e NEXT_PUBLIC_ >> apps/app/.env.production
            - npx nx build app
      artifacts:
        baseDirectory: dist/apps/app/.next
```

```
        files:
          - '**/*'
      cache:
        paths:
          - node_modules/**/*
      buildPath: /
    appRoot: apps/app
```

上述範例中建置命令區段的下列幾行示範如何從建置環境取得特定變數，並將其新增至具有應用程式根之單一儲存庫中應用程式的 `.env.production` 檔案`apps/app`。

```
- env | grep -e API_BASE_URL -e APP_ENV >> apps/app/.env.production
- env | grep -e NEXT_PUBLIC_ >> apps/app/.env.production
```

在單一儲存庫中部署 Next.js 應用程式

Amplify 支援通用 monorepos 中的應用程式，以及使用 npm 工作區、pnpm 工作區、Yarn 工作區、Nx 和 Turborepo 建立的 monorepo 中的應用程式。部署應用程式時，Amplify 會自動偵測您正在使用的 monorepo 建置架構。Amplify 會自動套用 npm 工作區、Yarn 工作區或 Nx 中應用程式的建置設定。Turborepo 和 pnpm 應用程式需要額外的組態。如需詳細資訊，請參閱設定 monorepo 組建設定。

如需詳細的 Nx 範例，請參閱 Next.js 應用程式與 Nx on AWS Amplify 託管部落格文章。

Nuxt.js 的 Amplify 支援

Nuxt 是使用 Vue.js 建立完整堆疊 Web 應用程式的架構。

轉接器

> 您可以使用具有零組態的預設轉接器，將 Nuxt.js 應用程式部署至 Amplify。如需轉接器的詳細資訊，請參閱 Nuxt 文件。

教學課程

> 若要了解如何將 Nuxt.js 應用程式部署至 Amplify，請參閱 將 Nuxt.js 應用程式部署至 Amplify 託管。

示範

> 如需影片示範，請參閱 YouTube 上的使用 ZERO 組態在幾分鐘內（含 AWS) 進行 Nuxt 託管。

Amplify 支援 Astro.js

Astro 是用於建立內容驅動型 Web 應用程式的 Web 架構。

轉接器

您可以使用社群轉接器將 Astro.js 應用程式部署至 Amplify。我們不會維護 Astro 架構的 Amplify 擁有轉接器。不過，轉接器可在 GitHub 網站上的 github.com/alexnguyennz/astro-aws-amplify：// 取得。此轉接器是由社群成員所建立，並非由 維護 AWS。

教學課程

若要了解如何將 Astro 應用程式部署至 Amplify，請參閱 將 Astro.js 應用程式部署至 Amplify 託管。

示範

如需影片示範，請參閱 Amazon Web Services YouTube 頻道上的如何將 Astro 網站部署至 AWS。

對 SvelteKit 的 Amplify 支援

SvelteKit 是使用 Svelte 建立完整堆疊 Web 應用程式的架構。

轉接器

您可以使用社群轉接器將 SvelteKit 應用程式部署至 Amplify。我們不會維護 SvelteKit 架構的 Amplify 擁有轉接器。不過，轉接器可在 GitHub 網站上的 https：//github.com/gzimbron/amplify-adapter 取得。此轉接器是由社群成員所建立，並非由 維護 AWS。

教學課程

若要了解如何將 SvelteKit 應用程式部署至 Amplify，請參閱 將 SvelteKit 應用程式部署至 Amplify 託管。

示範

如需影片示範，請參閱 Amazon Web Services YouTube 頻道上的如何將 SvelteKit 網站 （使用 API) 部署至 AWS 。

將 SSR 應用程式部署至 Amplify

您可以使用這些指示來部署使用具有符合 Amplify 預期建置輸出之部署套件的任何架構所建立的應用程式。如果您要部署 Next.js 應用程式，則不需要轉接器。

如果您要部署使用架構轉接器的 SSR 應用程式，您必須先安裝和設定轉接器。如需說明，請參閱對任何 SSR 架構使用開放原始碼轉接器。

將 SSR 應用程式部署至 Amplify 託管

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 在所有應用程式頁面上，選擇建立新應用程式。
3. 在開始使用 Amplify 建置頁面上，選擇您的 Git 儲存庫提供者，然後選擇下一步。
4. 在新增儲存庫分支頁面上，執行下列動作：
   a. 選取要連線的儲存庫名稱。
   b. 選取要連線的儲存庫分支名稱。
   c. 選擇下一步。
5. 在應用程式設定頁面上，Amplify 會自動偵測 Next.js SSR 應用程式。

如果您要部署使用另一個架構轉接器的 SSR 應用程式，您必須明確啟用 Amazon CloudWatch Logs。開啟進階設定區段，然後在伺服器端轉譯 (SSR) 部署區段中選擇啟用 SSR 應用程式日誌。

6. 應用程式需要 Amplify 擔任的 IAM 服務角色，才能將日誌交付到您的 AWS 帳戶。

   新增服務角色的程序會因您要建立新角色或使用現有角色而有所不同。

   - 若要建立新的角色：
     - 選擇建立並使用新的服務角色。
   - 若要使用現有角色：
     a. 選擇使用現有角色。
     b. 在服務角色清單中，選取要使用的角色。
7. 選擇下一步。
8. 在檢閱頁面上，選擇儲存並部署。

SSR 支援的功能

本節提供有關 Amplify 支援 SSR 功能的資訊。

Amplify 提供 Node.js 版本支援，以符合用於建置應用程式的 Node.js 版本。

Amplify 提供內建的影像最佳化功能，可支援所有 SSR 應用程式。如果您不想使用預設映像最佳化功能，您可以實作自訂映像最佳化載入器。

主題

- Next.js 應用程式的 Node.js 版本支援
- SSR 應用程式的影像最佳化
- 適用於 SSR 應用程式的 Amazon CloudWatch Logs
- Amplify Next.js 11 SSR 支援

Next.js 應用程式的 Node.js 版本支援

當 Amplify 建置和部署 Next.js 運算應用程式時，它會使用與用來建置應用程式之 的主要版本相符 Node.js的Node.js執行時間版本。

> **Note**
>
> 從 2025 年 9 月 15 日開始，Amplify 託管將不再支援 Node.js 14、Node.js 16 和 Node.js 18 執行時間。支援的執行時間包括 Node.js 20 和 Node.js 22。

您可以在 Amplify 主控台的即時套件覆寫功能中指定要使用的Node.js版本。如需設定即時套件更新的詳細資訊，請參閱 在建置映像中使用特定套件和相依性版本。您也可以使用其他機制指定Node.js版本，例如 nvm 命令。如果您未指定版本，Amplify 會預設為使用 Amplify 組建容器使用的目前版本。

SSR 應用程式的影像最佳化

Amplify 託管提供內建的影像最佳化功能，可支援所有 SSR 應用程式。透過 Amplify 的映像最佳化，您可以為存取它們的裝置提供正確格式、維度和解析度的高品質映像，同時保持最小的檔案大小。

目前，您可以使用 Next.js Image 元件來隨需最佳化映像，也可以實作自訂映像載入器。如果您使用的是 Next.js 13 或更新版本，則不需要採取任何進一步的動作來使用 Amplify 的映像最佳化功能。如果您要實作自訂載入器，請參閱以下使用自訂映像載入器主題。

使用自訂映像載入器

如果您使用自訂映像載入器，Amplify 會偵測應用程式`next.config.js`檔案中的載入器，且不會使用內建映像最佳化功能。如需有關 Next.js 支援的自訂載入器的詳細資訊，請參閱 Next.js 影像文件。

適用於 SSR 應用程式的 Amazon CloudWatch Logs

Amplify 會將 SSR 執行期的相關資訊傳送至您 中的 Amazon CloudWatch Logs AWS 帳戶。當您部署 SSR 應用程式時，應用程式需要 Amplify 代表您呼叫其他 服務時擔任的 IAM 服務角色。您可以允許 Amplify 託管運算自動為您建立服務角色，也可以指定您已建立的角色。

如果您選擇允許 Amplify 為您建立 IAM 角色，該角色將已有建立 CloudWatch Logs 的許可。如果您建立自己的 IAM 角色，則需要將下列許可新增至政策，以允許 Amplify 存取 Amazon CloudWatch Logs。

```
logs:CreateLogStream
logs:CreateLogGroup
logs:DescribeLogGroups
logs:PutLogEvents
```

如需服務角色的詳細資訊，請參閱新增具有部署後端資源許可的服務角色。

Amplify Next.js 11 SSR 支援

如果您在 2022 年 11 月 17 日發行 Amplify 託管運算之前，已將 Next.js 應用程式部署至 Amplify，您的應用程式會使用 Amplify 先前的 SSR 供應商 Classic（僅限 Next.js 11)。本節中的文件僅適用於使用 Classic（僅限 Next.js 11) SSR 提供者部署的應用程式。

> **Note**
>
> 我們強烈建議您將 Next.js 11 應用程式遷移至 Amplify 託管運算受管 SSR 供應商。如需詳細資訊，請參閱將 Next.js 11 SSR 應用程式遷移至 Amplify 託管運算。

下列清單說明 Amplify Classic（僅限 Next.js 11) SSR 提供者支援的特定功能。

支援的功能

- 伺服器端轉譯頁面 (SSR)
- 靜態頁面
- API 路由
- 動態路由
- 擷取所有路由
- SSG（靜態產生）
- 增量靜態再生 (ISR)
- 國際化 (i18n) 子路徑路由
- 環境變數

不支援的功能

- 映像最佳化
- 隨需增量靜態再生 (ISR)
- 國際化 (i18n) 網域路由
- 國際化 (i18n) 自動地區設定偵測
- 中介軟體
- Edge 中介軟體
- Edge API 路由

Next.js 11 SSR 應用程式的定價

部署 Next.js 11 SSR 應用程式時，Amplify 會在 AWS 您的帳戶中建立其他後端資源，包括：

- Amazon Simple Storage Service (Amazon S3) 儲存貯體，可存放應用程式靜態資產的資源。如需 Amazon S3 費用的相關資訊，請參閱 Amazon S3 定價。
- 提供應用程式的 Amazon CloudFront 分佈。如需 CloudFront 費用的相關資訊，請參閱 Amazon CloudFront 定價。
- 四個 Lambda@Edge 函數可自訂 CloudFront 提供的內容。

AWS Identity and Access Management Next.js 11 SSR 應用程式的 許可

Amplify 需要 AWS Identity and Access Management (IAM) 許可才能部署 SSR 應用程式。對於 SSR 應用程式，Amplify 部署資源，例如 Amazon S3 儲存貯體、CloudFront 分佈、 Lambda@Edge 函數、Amazon SQS 佇列 （ 如果使用 ISR) 和 IAM 角色。如果沒有所需的最低許可，當您嘗試部署 SSR 應用程式`Access Denied`時，將會發生錯誤。若要提供 Amplify 所需的許可，您必須指定服務角色。

若要建立 Amplify 在代表您呼叫其他 服務時擔任的 IAM 服務角色，請參閱 新增具有部署後端資源許可的服務角色。這些指示示範如何建立連接 `AdministratorAccess-Amplify`受管政策的角色。

`AdministratorAccess-Amplify` 受管政策可讓您存取多個 AWS 服務，包括 IAM 動作。 和 應視為與`AdministratorAccess`政策一樣強大。此政策提供比部署 SSR 應用程式所需的更多許可。

建議您遵循授予最低權限的最佳實務，並減少授予服務角色的許可。您可以建立自己的客戶受管 IAM 政策，只授予部署 SSR 應用程式所需的許可，而不是授予管理員存取許可給服務角色。如需建立客戶受管政策的說明，請參閱《IAM 使用者指南》中的建立 IAM 政策。

如果您建立自己的政策，請參閱下列部署 SSR 應用程式所需的最低許可清單。

```
acm:DescribeCertificate
acm:DescribeCertificate
acm:ListCertificates
acm:RequestCertificate
cloudfront:CreateCloudFrontOriginAccessIdentity
cloudfront:CreateDistribution
cloudfront:CreateInvalidation
cloudfront:GetDistribution
cloudfront:GetDistributionConfig
cloudfront:ListCloudFrontOriginAccessIdentities
cloudfront:ListDistributions
cloudfront:ListDistributionsByLambdaFunction
```

```
cloudfront:ListDistributionsByWebACLId
cloudfront:ListFieldLevelEncryptionConfigs
cloudfront:ListFieldLevelEncryptionProfiles
cloudfront:ListInvalidations
cloudfront:ListPublicKeys
cloudfront:ListStreamingDistributions
cloudfront:UpdateDistribution
cloudfront:TagResource
cloudfront:UntagResource
cloudfront:ListTagsForResource
iam:AttachRolePolicy
iam:CreateRole
iam:CreateServiceLinkedRole
iam:GetRole
iam:PutRolePolicy
iam:PassRole
lambda:CreateFunction
lambda:EnableReplication
lambda:DeleteFunction
lambda:GetFunction
lambda:GetFunctionConfiguration
lambda:PublishVersion
lambda:UpdateFunctionCode
lambda:UpdateFunctionConfiguration
lambda:ListTags
lambda:TagResource
lambda:UntagResource
route53:ChangeResourceRecordSets
route53:ListHostedZonesByName
route53:ListResourceRecordSets
s3:CreateBucket
s3:GetAccelerateConfiguration
s3:GetObject
s3:ListBucket
s3:PutAccelerateConfiguration
s3:PutBucketPolicy
s3:PutObject
s3:PutBucketTagging
s3:GetBucketTagging
lambda:ListEventSourceMappings
lambda:CreateEventSourceMapping
iam:UpdateAssumeRolePolicy
iam:DeleteRolePolicy
sqs:CreateQueue           // SQS only needed if using ISR feature
```

```
sqs:DeleteQueue
sqs:GetQueueAttributes
sqs:SetQueueAttributes
amplify:GetApp
amplify:GetBranch
amplify:UpdateApp
amplify:UpdateBranch
```

故障診斷 Next.js 11 SSR 部署

如果您在使用 Amplify 部署 Classic （僅限 Next.js 11) SSR 應用程式時遇到意外問題，請檢閱下列疑難排解主題。

主題

- 我的應用程式的輸出目錄遭到覆寫
- 部署我的 SSR 網站後收到 404 錯誤
- 我的應用程式缺少 CloudFront SSR 分佈的重寫規則
- 我的應用程式太大，無法部署
- 我的建置失敗，並出現記憶體不足錯誤
- 我的應用程式同時具有 SSR 和 SSG 分支
- 我的應用程式會將靜態檔案存放在具有預留路徑的資料夾中
- 我的應用程式已達到 CloudFront 限制
- Lambda@Edge 函數是在美國東部（維吉尼亞北部）區域建立
- 我的 Next.js 應用程式使用不支援的功能
- 我的 Next.js 應用程式中的影像未載入
- 不支援的區域

我的應用程式的輸出目錄遭到覆寫

使用 Amplify 部署之 Next.js 應用程式的輸出目錄必須設定為 `.next`。如果您應用程式的輸出目錄遭到覆寫，請檢查 `next.config.js` 檔案。若要將建置輸出目錄預設為 `.next`，請從 檔案移除下列行：

```
distDir: 'build'
```

在您的建置設定`.next`中，確認輸出目錄已設為 。如需有關檢視應用程式建置設定的資訊，請參閱 設定 Amplify 應用程式的建置設定。

以下是 設定為 之應用程式的建置設定範例baseDirectory.next。

```
version: 1
frontend:
  phases:
    preBuild:
      commands:
        - npm ci
    build:
      commands:
        - npm run build
  artifacts:
    baseDirectory: .next
    files:
      - '**/*'
  cache:
    paths:
      - node_modules/**/*
```

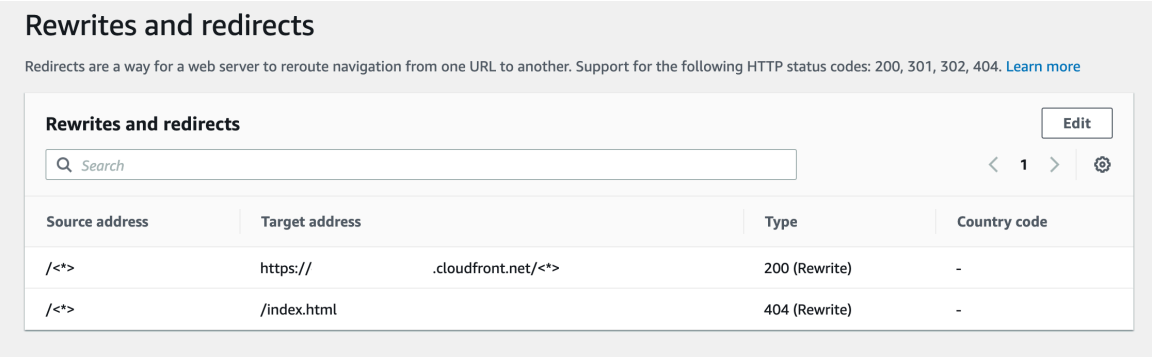
部署我的 SSR 網站後收到 404 錯誤

如果您在部署網站後收到 404 錯誤，問題可能是因為您的輸出目錄遭到覆寫所致。若要檢查next.config.js檔案並驗證應用程式建置規格中的正確建置輸出目錄，請遵循上一個主題中的步驟 我的應用程式的輸出目錄遭到覆寫。

我的應用程式缺少 CloudFront SSR 分佈的重寫規則

當您部署 SSR 應用程式時，Amplify 會為您的 CloudFront SSR 分佈建立重寫規則。如果您無法在 Web 瀏覽器中存取應用程式，請在 Amplify 主控台中確認您的應用程式存在 CloudFront 重寫規則。如果遺失，您可以手動新增或重新部署您的應用程式。

若要在 Amplify 主控台中檢視或編輯應用程式的重寫和重新導向規則，請在導覽窗格中選擇應用程式設定，然後選擇重寫和重新導向。下列螢幕擷取畫面顯示 Amplify 在您部署 SSR 應用程式時為您建立的重寫規則範例。請注意，在此範例中，存在 CloudFront 重寫規則。



我的應用程式太大，無法部署

Amplify 會將 SSR 部署的大小限制為 50 MB。如果您嘗試將 Next.js SSR 應用程式部署到 Amplify 並發生錯誤`RequestEntityTooLargeException`，您的應用程式太大而無法部署。您可以將快取清除程式碼新增至 `next.config.js` 檔案，嘗試解決此問題。

以下是 `next.config.js` 檔案中執行快取清除的程式碼範例。

```
module.exports = {
    webpack: (config, { buildId, dev, isServer, defaultLoaders, webpack }) => {
        config.optimization.splitChunks.cacheGroups = { }
        config.optimization.minimize = true;
        return config
      },
}
```

我的建置失敗，並出現記憶體不足錯誤

Next.js 可讓您快取建置成品，以改善後續建置的效能。此外，Amplify 的 AWS CodeBuild 容器會代您壓縮此快取並上傳至 Amazon S3，以改善後續建置效能。這可能會導致您的建置因記憶體不足錯誤而失敗。

執行下列動作，以防止您的應用程式在建置階段期間超過記憶體限制。首先，`.next/cache/**/*`從建置設定的 cache.paths 區段中移除 。接著，從您的建置設定檔案移除`NODE_OPTIONS`環境變數。反之，請在 Amplify 主控台中設定`NODE_OPTIONS`環境變數，以定義節點最大記憶體限制。如需使用 Amplify 主控台設定環境變數的詳細資訊，請參閱 設定環境變數。

進行這些變更後，請再次嘗試建置。如果成功，請將 `.next/cache/**/*`新增至建置設定檔案的 cache.paths 區段。

如需改善建置效能的 Next.js 快取組態詳細資訊，請參閱 Next.js 網站上的 AWS CodeBuild。

我的應用程式同時具有 SSR 和 SSG 分支

您無法部署同時具有 SSR 和 SSG 分支的應用程式。如果您需要同時部署 SSR 和 SSG 分支，您必須部署一個只使用 SSR 分支的應用程式，以及另一個只使用 SSG 分支的應用程式。

我的應用程式會將靜態檔案存放在具有預留路徑的資料夾中

Next.js 可以從儲存在專案根目錄中`public`名為 的資料夾提供靜態檔案。當您使用 Amplify 部署和託管 Next.js 應用程式時，您的專案無法包含路徑為 的資料夾`public/static`。Amplify 保留分發應用程式時使用的`public/static`路徑。如果您的應用程式包含此路徑，您必須在使用 Amplify 部署之前重新命名`static`資料夾。

我的應用程式已達到 CloudFront 限制

CloudFront 服務配額會將 AWS 您的帳戶限制為 25 個具有連接 Lambda@Edge 函數的分佈。如果您超過此配額，您可以從您的帳戶刪除任何未使用的 CloudFront 分佈，或請求增加配額。如需詳細資訊，請參閱「Service Quotas 使用者指南」中的請求提高配額。

Lambda@Edge 函數是在美國東部（維吉尼亞北部）區域建立

當您部署 Next.js 應用程式時，Amplify 會建立 Lambda@Edge 函數來自訂 CloudFront 提供的內容。Lambda@Edge 函數是在美國東部（維吉尼亞北部）區域中建立的，而不是部署應用程式的 區域。這是 Lambda@Edge 限制。如需 Lambda@Edge 函數的詳細資訊，請參閱《Amazon CloudFront 開發人員指南》中的邊緣函數限制。 Amazon CloudFront

我的 Next.js 應用程式使用不支援的功能

使用 Amplify 部署的應用程式支援直到 11 版的 Next.js 主要版本。如需 Amplify 支援和不支援的 Next.js 功能的詳細清單，請參閱 supported features。

當您部署新的 Next.js 應用程式時，Amplify 預設會使用最新支援的 Next.js 版本。如果您有使用舊版 Next.js 部署至 Amplify 的現有 Next.js 應用程式，您可以將應用程式遷移至 Amplify 託管運算 SSR 供應商。如需說明，請參閱將 Next.js 11 SSR 應用程式遷移至 Amplify 託管運算。

我的 Next.js 應用程式中的影像未載入

當您使用 `next/image`元件將映像新增至 Next.js 應用程式時，映像的大小不得超過 1 MB。當您將應用程式部署到 Amplify 時，大於 1 MB 的影像將傳回 503 錯誤。這是因為 Lambda@Edge 限制會將 Lambda 函數產生的回應大小限制為 1 MB，包括標頭和內文。

1 MB 限制適用於應用程式中的其他成品，例如 PDF 和文件檔案。

不支援的區域

Amplify 在提供 Amplify 的每個 AWS 區域中不支援 Classic （僅限 Next.js 11) SSR 應用程式部署。下列區域不支援 Classic （僅限 Next.js 11) SSR：歐洲 （米蘭） eu-south-1、中東 （巴林） me-south-1 和亞太區域 （香港） ap-east-1。

對 SSR 部署進行故障診斷

如果您在使用 Amplify 託管運算部署 SSR 應用程式時遇到意外問題，請參閱 Amplify 故障診斷章節對伺服器端轉譯應用程式進行故障診斷中的 。

進階：開放原始碼轉接器

架構作者可以使用檔案系統型部署規格來開發為其特定架構自訂的開放原始碼建置轉接器。這些轉接器會將應用程式的建置輸出轉換為符合 Amplify Hosting 預期目錄結構的部署套件。此部署套件將包含託管應用程式所需的所有必要檔案和資產，包括執行期組態，例如路由規則。

如果您不使用架構，您可以開發自己的解決方案，以產生 Amplify 預期的建置輸出。

主題

- 使用 Amplify 託管部署規格來設定建置輸出
- 使用部署資訊清單部署 Express 伺服器
- 架構作者的影像最佳化整合
- 對任何 SSR 架構使用開放原始碼轉接器

使用 Amplify 託管部署規格來設定建置輸出

Amplify Hosting 部署規格是以檔案系統為基礎的規格，可定義目錄結構，以利部署至 Amplify Hosting。架構可以產生此預期的目錄結構做為其建置命令的輸出，讓架構能夠利用 Amplify Hosting 的服務基本概念。Amplify Hosting 了解部署套件的結構，並據此進行部署。

如需說明如何使用部署規格的影片示範，請參閱 Amazon Web Services YouTube 頻道上的如何使用託管任何網站 AWS Amplify。 YouTube

以下是 Amplify 預期部署套件的資料夾結構範例。在高階，它有一個名為 的資料夾`static`、一個名為 的資料夾`compute`和一個名為 的部署資訊清單檔案`deploy-manifest.json`。

```
.amplify-hosting/
### compute/
#   ### default/
#       ### chunks/
#       #   ### app/
#       #       ### _nuxt/
#       #       #   ### index-xxx.mjs
#       #       #   ### index-styles.xxx.js
#       #       ### server.mjs
#       ### node_modules/
#       ### server.js
### static/
#   ### css/
#   #   ### nuxt-google-fonts.css
#   ### fonts/
#   #   ### font.woff2
#   ### _nuxt/
#   #   ### builds/
#   #   #   ### latest.json
#   #   ### entry.xxx.js
```

```
#   ### favicon.ico
#   ### robots.txt
### deploy-manifest.json
```

Amplify SSR 基本支援

Amplify 託管部署規格定義了密切對應至下列基本概念的合約。

靜態資產

為架構提供託管靜態檔案的能力。

運算

提供可在連接埠 3000 上執行 Node.js HTTP 伺服器的架構。

映像最佳化

提供具有 服務的架構，以在執行時間最佳化映像。

路由規則

提供架構，其機制可將傳入請求路徑映射至特定目標。

.amplify-hosting/static 目錄

您必須將所有要從應用程式 URL 提供的可公開存取靜態檔案放入 `.amplify-hosting/static`目錄中。此目錄中的檔案會透過靜態資產基本提供。

靜態檔案可在應用程式 URL 的根 (/) 存取，而不會變更其內容、檔案名稱或副檔名。此外，子目錄會保留在 URL 結構中，並顯示在檔案名稱之前。例如， `.amplify-hosting/static/favicon.ico` 將從 提供`https://myAppId.amplify-hostingapp.com/favicon.ico`，`.amplify-hosting/static/_nuxt/main.js`並從 提供 `https://myAppId.amplify-hostingapp.com/_nuxt/main.js`

如果架構支援修改應用程式基本路徑的功能，則必須在基本路徑前面加上`.amplify-hosting/static`目錄內的靜態資產。例如，如果基本路徑為 `/folder1/folder2`，則名為 之靜態資產的建置輸出`main.css`將為 `.amplify-hosting/static/folder1/folder2/main.css`。

.amplify-hosting/compute 目錄

單一運算資源由`.amplify-hosting/compute`目錄內`default`包含名為 的單一子目錄表示。路徑為 `.amplify-hosting/compute/default`。此運算資源會映射至 Amplify Hosting 的運算基本。

default 子目錄的內容必須符合下列規則。

- 檔案必須存在於 default子目錄的根目錄中，以做為運算資源的進入點。
- 進入點檔案必須是 Node.js 模組，而且必須啟動接聽連接埠 3000 的 HTTP 伺服器。
- 您可以在 default 子目錄中放置其他檔案，並從進入點檔案中的程式碼參考這些檔案。
- 子目錄的內容必須是獨立的。進入點模組中的程式碼無法參考子目錄以外的任何模組。請注意，架構可以任意方式綁定其 HTTP 伺服器。如果可以使用 node server.js命令啟動運算程序，其中 server.js is是子目錄中項目檔案名稱，Amplify 會將目錄結構視為符合部署規格。

Amplify 託管套件，並將default子目錄中的所有檔案部署到佈建的運算資源。每個運算資源配置 512 MB 的暫時性儲存。此儲存體不會在執行執行個體之間共用，而是在相同執行執行個體內的後續調用之間共用。執行執行個體的執行時間上限為 15 分鐘，且執行執行個體內唯一的可寫入路徑是 /tmp目錄。每個運算資源套件的未壓縮大小不能超過 220 MB。例如，.amplify/compute/default子目錄在未壓縮時不得超過 220 MB。

.amplify-hosting/deploy-manifest.json 檔案

使用 deploy-manifest.json 檔案來存放部署的組態詳細資訊和中繼資料。檔案至少deploy-manifest.json必須包含version屬性、具有指定全部截獲路由的routes屬性，以及具有指定架構中繼資料的framework屬性。

下列物件定義示範部署資訊清單的組態。

```
type DeployManifest = {
  version: 1;
  routes: Route[];
  computeResources?: ComputeResource[];
  imageSettings?: ImageSettings;
  framework: FrameworkMetadata;
};
```

下列主題說明部署資訊清單中每個屬性的詳細資訊和用量。

使用版本屬性

version 屬性會定義您正在實作的部署規格版本。目前，Amplify 託管部署規格的唯一版本是版本 1。下列 JSON 範例示範 version 屬性的用量。

```
"version": 1
```

使用路由屬性

routes 屬性可讓架構利用 Amplify 託管路由規則基本。路由規則提供將傳入請求路徑路由到部署套件中特定目標的機制。路由規則只會指定傳入請求的目的地，並在透過重寫和重新導向規則轉換請求之後套用。如需 Amplify 託管如何處理重寫和重新導向的詳細資訊，請參閱 設定 Amplify 應用程式的重新導向和重寫。

路由規則不會重寫或轉換請求。如果傳入請求符合路由的路徑模式，請求會依原狀路由至路由的目標。

routes 陣列中指定的路由規則必須符合下列規則。

- 必須指定全部擷取路由。全部截獲路由的/*模式符合所有傳入請求。
- routes 陣列最多可包含 25 個項目。
- 您必須指定Static路由或Compute路由。
- 如果您指定Static路由，.amplify-hosting/static目錄必須存在。
- 如果您指定Compute路由，.amplify-hosting/compute目錄必須存在。
- 如果您指定ImageOptimization路由，您還必須指定Compute路由。這是必要的，因為映像最佳化尚未支援純靜態應用程式。

下列物件定義示範 Route 物件的組態。

```
type Route = {
  path: string;
  target: Target;
  fallback?: Target;
}
```

下表說明Route物件的屬性。

金鑰	Type	必要	Description
路徑	String	是	定義符合傳入請求路徑的模式（不包括querystring)。 路徑長度上限為 255 個字元。

金鑰	Type	必要	Description
			路徑必須以正斜線 開頭/。 路徑可包含下列任何字元：【A-Z】、【a-z】、【0-9】、【 _-.* $/~'"@： +】。 對於模式比對，僅支援下列萬用字元： • * （符合 0 個以上的字元） • 模式/*稱為全部截獲模式，且將符合所有傳入的請求。
目標	Target	是	定義目標以將相符請求路由至其中的物件。 如果指定Compute路由，則對應的ComputeResource必須存在。 如果指定ImageOptimization 路由，imageSettings 也必須指定 。

金鑰	Type	必要	Description
備用	Target	否	如果原始目標傳回 404 錯誤，則定義要恢復目標的物件。 指定路由的target種類和fallback種類不能相同。例如，Static不允許從 到Static的遞迴。只有沒有內文的 GET 請求才支援備用。如果請求中存在內文，則會在後援期間將其捨棄。

下列物件定義示範 Target 物件的組態。

```
type Target = {
  kind: TargetKind;
  src?: string;
  cacheControl?: string;
}
```

下表說明Target物件的屬性。

金鑰	Type	必要	Description
類型	Targetkind	是	enum 定義目標類型的 。有效值為 Static、Compute 和 ImageOpti mization 。
src	String	是 Compute 否 用於其他基本概念	字串，指定部署套件中包含基本可執行程

金鑰	Type	必要	Description
			式碼的子目錄名稱。有效且僅對運算基本需要。 值必須指向部署套件中存在的其中一個運算資源。目前，此欄位唯一支援的值為 `default`。
cacheControl	String	否	指定要套用至回應之快取控制標頭值的字串。僅適用於靜態和 ImageOptimization 基本概念。 指定的值會由自訂標頭覆寫。如需 Amplify 託管客戶標頭的詳細資訊，請參閱 設定 Amplify 應用程式的自訂標頭。 **Note** 此快取控制標頭只會套用至狀態碼設為 200 (OK) 的成功回應。

下列物件定義示範 `TargetKind` 列舉的用量。

```
enum TargetKind {
  Static = "Static",
```

```
  Compute = "Compute",
  ImageOptimization = "ImageOptimization"
}
```

下列清單指定TargetKind列舉的有效值。

靜態

將請求路由到靜態資產基本。

運算

將請求路由到運算基本。

ImageOptimization

將請求路由至映像最佳化基本。

下列 JSON 範例示範指定了多個路由規則的 routes 屬性用量。

```
"routes": [
    {
      "path": "/_nuxt/image",
      "target": {
        "kind": "ImageOptimization",
        "cacheControl": "public, max-age=3600, immutable"
      }
    },
    {
      "path": "/_nuxt/builds/meta/*",
      "target": {
        "cacheControl": "public, max-age=31536000, immutable",
        "kind": "Static"
      }
    },
    {
      "path": "/_nuxt/builds/*",
      "target": {
        "cacheControl": "public, max-age=1, immutable",
        "kind": "Static"
      }
    },
    {
```

```
      "path": "/_nuxt/*",
      "target": {
        "cacheControl": "public, max-age=31536000, immutable",
        "kind": "Static"
      }
    },
    {
      "path": "/*.*",
      "target": {
        "kind": "Static"
      },
      "fallback": {
        "kind": "Compute",
        "src": "default"
      }
    },
    {
      "path": "/*",
      "target": {
        "kind": "Compute",
        "src": "default"
      }
    }
  ]
```

如需在部署資訊清單中指定路由規則的詳細資訊，請參閱 設定路由規則的最佳實務

使用 computeResources 屬性

computeResources 屬性可讓架構提供有關佈建運算資源的中繼資料。每個運算資源都必須有與其相關聯的對應路由。

下列物件定義示範 ComputeResource 物件的用量。

```
type ComputeResource = {
  name: string;
  runtime: ComputeRuntime;
  entrypoint: string;
};

type ComputeRuntime = 'nodejs20.x' | 'nodejs22.x';
```

下表說明ComputeResource物件的屬性。

金鑰	Type	必要	Description
name	String	是	指定運算資源的名稱。名稱必須符合 內子目錄的名稱.amplify-hosting/compute directory 。 對於部署規格的第 1 版，唯一的有效值為 default。
執行時期	ComputeRuntime	是	定義佈建運算資源的執行時間。 有效值為 nodejs20.x 和 nodejs22.x 。
進入點	String	是	指定程式碼將針對指定的運算資源從中執行的起始檔案名稱。檔案必須存在於代表運算資源的子目錄中。

如果您有如下所示的目錄結構。

```
.amplify-hosting
|---compute
|   |---default
|       |---index.js
```

computeResource 屬性的 JSON 會如下所示。

```
"computeResources": [
```

```
    {
      "name": "default",
      "runtime": "nodejs20.x",
      "entrypoint": "index.js",
    }
  ]
```

使用 imageSettings 屬性

imageSettings 屬性可讓架構自訂映像最佳化基本的行為，以在執行時間提供映像的隨需最佳化。

下列物件定義示範 ImageSettings 物件的用量。

```
type ImageSettings = {
  sizes: number[];
  domains: string[];
  remotePatterns: RemotePattern[];
  formats: ImageFormat[];
  minumumCacheTTL: number;
  dangerouslyAllowSVG: boolean;
};

type ImageFormat = 'image/avif' | 'image/webp' | 'image/png' | 'image/jpeg';
```

下表說明ImageSettings物件的屬性。

金鑰	Type	必要	Description
大小	Number【】	是	支援的映像寬度陣列。
domains	字串 【】	是	可使用映像最佳化的允許外部網域陣列。將陣列保留空白，只允許部署網域使用映像最佳化。
remotePatterns	RemotePattern【】	是	可使用映像最佳化的允許外部模式陣列。與網域類似，但使用

金鑰	Type	必要	Description
			規則表達式 (regex) 提供更多控制。
格式	ImageFormat【】	是	允許輸出影像格式的陣列。
minimumCacheTTL	Number	是	最佳化映像的快取持續時間，以秒為單位。
dangerouslyAllowSVG	Boolean	是	允許 SVG 輸入映像 URLs。基於安全考量，預設會停用此功能。

下列物件定義示範 `RemotePattern` 物件的用量。

```
type RemotePattern = {
  protocol?: 'https';
  hostname: string;
  port?: string;
  pathname?: string;
}
```

下表說明`RemotePattern`物件的屬性。

金鑰	Type	必要	Description
protocol	String	否	允許遠端模式的通訊協定。唯一有效的值為 `https`。
hostname	String	是	允許遠端模式的主機名稱。 您可以指定常值或萬用字元。單一 `*` 符合

金鑰	Type	必要	Description
			單一子網域。雙 `**` 符合任意數量的子網域。Amplify 不允許只指定 `**` 的空白萬用字元。
port	String	否	允許遠端模式的連接埠。
路徑名稱	String	否	允許遠端模式的路徑名稱。

下列範例示範 `imageSettings` 屬性。

```
"imageSettings": {
    "sizes": [
      100,
      200
    ],
    "domains": [
      "example.com"
    ],
    "remotePatterns": [
      {
        "protocol": "https",
        "hostname": "example.com",
        "port": "",
        "pathname": "/**",
      }
    ],
    "formats": [
      "image/webp"
    ],
    "minumumCacheTTL": 60,
    "dangerouslyAllowSVG": false
  }
```

使用架構屬性

使用 framework 屬性指定架構中繼資料。

下列物件定義示範 FrameworkMetadata 物件的組態。

```
type FrameworkMetadata = {
  name: string;
  version: string;
}
```

下表說明FrameworkMetadata物件的屬性。

金鑰	Type	必要	Description
name	String	是	架構的名稱。
version	String	是	架構的版本。 它必須是有效的語意版本控制（轉換器）字串。

設定路由規則的最佳實務

路由規則提供將傳入請求路徑路由到部署套件中特定目標的機制。在部署套件中，架構作者可以將檔案發送到部署到下列任一目標的建置輸出：

- 靜態資產基本 – 檔案包含在 .amplify-hosting/static目錄中。
- 運算基本 – 檔案包含在 .amplify-hosting/compute/default目錄中。

架構作者也會在部署資訊清單檔案中提供一系列路由規則。陣列中的每個規則都會以循序周遊順序比對傳入的請求，直到相符為止。有相符的規則時，請求會路由到相符規則中指定的目標。或者，可以為每個規則指定備用目標。如果原始目標傳回 404 錯誤，請求會路由到備用目標。

部署規格需要周遊順序中的最後一個規則，才能成為全部截獲規則。使用 /* 路徑指定全部擷取規則。如果傳入的請求與路由規則陣列中的任何先前路由不相符，則請求會路由到全部截獲規則目標。

對於 等 SSR 架構Nuxt.js，全部擷取規則目標必須是運算基本目標。這是因為 SSR 應用程式具有伺服器端轉譯頁面，其中包含建置時無法預測的路由。例如，如果Nuxt.js應用程式有一個頁面，/blog/[slug]其中 [slug] 是動態路由參數。全部擷取規則目標是將請求路由到這些頁面的唯一方法。

相反地，特定路徑模式可用於鎖定建置時已知的路由。例如， 會從 /_nuxt 路徑Nuxt.js提供靜態資產。這表示/_nuxt/*路徑可由將請求路由至靜態資產基本的指定路由規則做為目標。

公有資料夾路由

大多數 SSR 架構都能夠從public資料夾提供可變靜態資產。favicon.ico 和 等檔案robots.txt通常會保留在 public 資料夾內，並從應用程式的根 URL 提供。例如，favicon.ico 檔案是從 提供https://example.com/favicon.ico。請注意，這些檔案沒有可預測的路徑模式。它們幾乎完全由檔案名稱決定。將public資料夾內的檔案設為目標的唯一方法是使用全部截獲路由。不過，全部擷取的路由目標必須是運算基本目標。

我們建議您使用下列其中一種方法來管理您的public資料夾。

1. 使用路徑模式來鎖定包含副檔名的請求路徑。例如，您可以使用 /*.*以包含副檔名的所有請求路徑為目標。

   請注意，這種方法可能不可靠。例如，如果public資料夾中有沒有副檔名的檔案，則此規則不會以這些檔案為目標。使用此方法需要注意的另一個問題是，應用程式的名稱中可能有包含句點的頁面。例如，/*.* 規則/blog/2021/01/01/hello.world會將 的頁面設為目標。這並不理想，因為頁面不是靜態資產。不過，您可以將備用目標新增至此規則，以確保當靜態基本命令出現 404 錯誤時，請求會回到運算基本命令。

   ```
   {
       "path": "/*.*",
       "target": {
           "kind": "Static"
       },
       "fallback": {
           "kind": "Compute",
           "src": "default"
       }
   }
   ```

2. 在建置時間識別 public 資料夾中的檔案，並為每個檔案發出路由規則。此方法無法擴展，因為部署規格有 25 個規則的限制。

   ```
   {
   ```

```
    "path": "/favicon.ico",
    "target": {
        "kind": "Static"
    }
},
{
    "path": "/robots.txt",
    "target": {
        "kind": "Static"
    }
}
```

3. 建議架構使用者將所有可變靜態資產存放在 public 資料夾內的子資料夾內。

   在下列範例中，使用者可以將所有可變靜態資產存放在 public/assets 資料夾內。然後，具有路徑模式的路由規則/assets/*可用於將public/assets資料夾中所有可變靜態資產設為目標。

```
{
    "path": "/assets/*",
    "target": {
        "kind": "Static"
    }
}
```

4. 指定全部擷取路由的靜態備用。此方法有缺點，會在下全部擷取備用路由一節中詳細說明。

全部擷取備用路由

對於為運算基本目標指定全部Nuxt.js擷取路由的 SSR 架構，架構作者可能會考慮為全部擷取路由指定靜態備用，以解決public資料夾路由問題。不過，這種類型的路由規則會中斷伺服器端轉譯的 404 頁。例如，如果最終使用者造訪不存在的頁面，應用程式會轉譯狀態碼為 404 的 404 頁面。不過，如果全部擷取路由具有靜態備用，則不會轉譯 404 頁面。反之，請求會回復為靜態原始程式碼，最後仍顯示 404 狀態碼，但不會轉譯 404 頁面。

```
{
    "path": "/*",
    "target": {
        "kind": "Compute",
        "src": "default"
    },
    "fallback": {
        "kind": "Static"
```

```
    }
}
```

基本路徑路由

提供修改應用程式基本路徑能力的架構，預期會在基礎路徑前面加上.amplify-hosting/static目錄內的靜態資產。例如，如果基本路徑為 /folder1/folder2，則名為 main.css 之靜態資產的建置輸出將為 .amplify-hosting/static/folder1/folder2/main.css。

這表示也需要更新路由規則，以反映基本路徑。例如，如果基本路徑為 /folder1/folder2，則public資料夾中靜態資產的路由規則會如下所示。

```
{
    "path": "/folder1/folder2/*.*",
    "target": {
        "kind": "Static"
    }
}
```

同樣地，伺服器端路由也需要先加上基本路徑。例如，如果基本路徑為 /folder1/folder2，則/api路由的路由規則會如下所示。

```
{
    "path": "/folder1/folder2/api/*",
    "target": {
        "kind": "Compute",
        "src": "default"
    }
}
```

不過，基本路徑不應在全部截獲路由的前面。例如，如果基本路徑為 /folder1/folder2，則全部擷取路由將保持如下。

```
{
    "path": "/*",
    "target": {
        "kind": "Compute",
        "src": "default"
    }
}
```

Nuxt.js 路由範例

以下是 Nuxt 應用程式的範例`deploy-manifest.json`檔案，示範如何指定路由規則。

```
{
  "version": 1,
  "routes": [
    {
      "path": "/_nuxt/image",
      "target": {
        "kind": "ImageOptimization",
        "cacheControl": "public, max-age=3600, immutable"
      }
    },
    {
      "path": "/_nuxt/builds/meta/*",
      "target": {
        "cacheControl": "public, max-age=31536000, immutable",
        "kind": "Static"
      }
    },
    {
      "path": "/_nuxt/builds/*",
      "target": {
        "cacheControl": "public, max-age=1, immutable",
        "kind": "Static"
      }
    },
    {
      "path": "/_nuxt/*",
      "target": {
        "cacheControl": "public, max-age=31536000, immutable",
        "kind": "Static"
      }
    },
    {
      "path": "/*.*",
      "target": {
        "kind": "Static"
      },
      "fallback": {
        "kind": "Compute",
        "src": "default"
      }
```

```
    },
    {
      "path": "/*",
      "target": {
        "kind": "Compute",
        "src": "default"
      }
    }
  ],
  "computeResources": [
    {
      "name": "default",
      "entrypoint": "server.js",
      "runtime": "nodejs22.x"
    }
  ],
  "framework": {
    "name": "nuxt",
    "version": "3.8.1"
  }
}
```

以下是 Nuxt 的範例`deploy-manifest.json`檔案，示範如何指定路由規則，包括基本路徑。

```
{
  "version": 1,
  "routes": [
    {
      "path": "/base-path/_nuxt/image",
      "target": {
        "kind": "ImageOptimization",
        "cacheControl": "public, max-age=3600, immutable"
      }
    },
    {
      "path": "/base-path/_nuxt/builds/meta/*",
      "target": {
        "cacheControl": "public, max-age=31536000, immutable",
        "kind": "Static"
      }
    },
    {
      "path": "/base-path/_nuxt/builds/*",
```

```
      "target": {
        "cacheControl": "public, max-age=1, immutable",
        "kind": "Static"
      }
    },
    {
      "path": "/base-path/_nuxt/*",
      "target": {
        "cacheControl": "public, max-age=31536000, immutable",
        "kind": "Static"
      }
    },
    {
      "path": "/base-path/*.*",
      "target": {
        "kind": "Static"
      },
      "fallback": {
        "kind": "Compute",
        "src": "default"
      }
    },
    {
      "path": "/*",
      "target": {
        "kind": "Compute",
        "src": "default"
      }
    }
  ],
  "computeResources": [
    {
      "name": "default",
      "entrypoint": "server.js",
      "runtime": "nodejs22.x"
    }
  ],
  "framework": {
    "name": "nuxt",
    "version": "3.8.1"
  }
}
```

如需使用 routes 屬性的詳細資訊，請參閱 使用路由屬性。

使用部署資訊清單部署 Express 伺服器

此範例說明如何使用 Amplify Hosting 部署規格來部署基本 Express 伺服器。您可以利用提供的部署資訊清單來指定路由、運算資源和其他組態。

在本機設定 Express 伺服器，然後再部署到 Amplify 託管

1. 為您的專案建立新的目錄，並安裝 Express 和 Typescript。

```
mkdir express-app
cd express-app

# The following command will prompt you for information about your project
npm init

# Install express, typescript and types
npm install express --save
npm install typescript ts-node @types/node @types/express --save-dev
```

2. 使用下列內容將tsconfig.json檔案新增至專案的根目錄。

```
{
  "compilerOptions": {
    "target": "es6",
    "module": "commonjs",
    "outDir": "./dist",
    "strict": true,
    "esModuleInterop": true,
    "skipLibCheck": true,
    "forceConsistentCasingInFileNames": true
  },
  "include": ["src/**/*.ts"],
  "exclude": ["node_modules"]
}
```

3. 在專案根目錄中建立名為 src的目錄。
4. 在 src目錄中建立 index.ts 檔案。這將是啟動 Express 伺服器之應用程式的進入點。伺服器應設定為接聽連接埠 3000。

```
// src/index.ts
import express from 'express';

const app: express.Application = express();
const port = 3000;

app.use(express.text());

app.listen(port, () => {
  console.log(`server is listening on ${port}`);
});

// Homepage
app.get('/', (req: express.Request, res: express.Response) => {
  res.status(200).send("Hello World!");
});

// GET
app.get('/get', (req: express.Request, res: express.Response) => {
  res.status(200).header("x-get-header", "get-header-value").send("get-response-
from-compute");
});

//POST
app.post('/post', (req: express.Request, res: express.Response) => {
  res.status(200).header("x-post-header", "post-header-
value").send(req.body.toString());
});

//PUT
app.put('/put', (req: express.Request, res: express.Response) => {
  res.status(200).header("x-put-header", "put-header-
value").send(req.body.toString());
});

//PATCH
app.patch('/patch', (req: express.Request, res: express.Response) => {
  res.status(200).header("x-patch-header", "patch-header-
value").send(req.body.toString());
});

// Delete
app.delete('/delete', (req: express.Request, res: express.Response) => {
```

```
  res.status(200).header("x-delete-header", "delete-header-value").send();
});
```

5. 將下列指令碼新增至您的 package.json 檔案。

```
"scripts": {
  "start": "ts-node src/index.ts",
  "build": "tsc",
  "serve": "node dist/index.js"
}
```

6. 在專案的根public目錄中建立名為 的目錄。然後使用hello-world.txt下列內容建立名為 的檔案。

```
Hello world!
```

7. 使用下列內容將.gitignore檔案新增至您的專案根目錄。

```
.amplify-hosting
dist
node_modules
```

設定 Amplify 部署資訊清單

1. 在deploy-manifest.json專案的根目錄中建立名為 的檔案。
2. 將下列資訊清單複製並貼到您的deploy-manifest.json檔案中。

```
{
  "version": 1,
  "framework": { "name": "express", "version": "4.18.2" },
  "imageSettings": {
    "sizes": [
      100,
      200,
      1920
    ],
    "domains": [],
    "remotePatterns": [],
    "formats": [],
    "minimumCacheTTL": 60,
    "dangerouslyAllowSVG": false
```

```
  },
  "routes": [
    {
      "path": "/_amplify/image",
      "target": {
        "kind": "ImageOptimization",
        "cacheControl": "public, max-age=3600, immutable"
      }
    },
    {
      "path": "/*.*",
      "target": {
        "kind": "Static",
        "cacheControl": "public, max-age=2"
      },
      "fallback": {
        "kind": "Compute",
        "src": "default"
      }
    },
    {
      "path": "/*",
      "target": {
        "kind": "Compute",
        "src": "default"
      }
    }
  ],
  "computeResources": [
    {
      "name": "default",
      "runtime": "nodejs22.x",
      "entrypoint": "index.js"
    }
  ]
}
```

資訊清單說明 Amplify Hosting 應如何處理應用程式的部署。主要設定如下。

- 版本 – 指出您正在使用的部署規格版本。
- 架構 – 調整此值以指定您的Express伺服器設定。
- imageSettings – 除非您正在處理映像最佳化，否則Express伺服器可選用本節。

- 路由 – 這些對於將流量導向應用程式的正確部分至關重要。"kind": "Compute" 路由會將流量導向您的伺服器邏輯。
- computeResources – 使用此區段指定Express伺服器的執行時間和進入點。

接著，設定建置後指令碼，將建置的應用程式成品移至.amplify-hosting部署套件。目錄結構符合 Amplify Hosting 部署規格。

設定建置後指令碼

1. 在專案根目錄中建立名為 bin的目錄。
2. 在 bin目錄中建立名為 postbuild.sh 的檔案。將下列內容新增至 postbuild.sh 檔案：

```
#!/bin/bash

rm -rf ./.amplify-hosting

mkdir -p ./.amplify-hosting/compute

cp -r ./dist ./.amplify-hosting/compute/default
cp -r ./node_modules ./.amplify-hosting/compute/default/node_modules

cp -r public ./.amplify-hosting/static

cp deploy-manifest.json ./.amplify-hosting/deploy-manifest.json
```

3. 將postbuild指令碼新增至您的 package.json 檔案。檔案看起來應該如下所示。

```
"scripts": {
  "start": "ts-node src/index.ts",
  "build": "tsc",
  "serve": "node dist/index.js",
  "postbuild": "chmod +x bin/postbuild.sh && ./bin/postbuild.sh"
}
```

4. 執行下列命令來建置您的應用程式。

```
npm run build
```

5. （選用）調整 Express 的路由。您可以修改部署資訊清單中的路由，以符合您的 Express 伺服器。例如，如果您在 public目錄中沒有任何靜態資產，您可能只需要將全部擷取路由"path": "/*"導向運算。這取決於伺服器的設定。

您的最終目錄結構看起來應該如下所示。

```
express-app/
### .amplify-hosting/
#   ### compute/
#   #   ### default/
#   #       ### node_modules/
#   #       ### index.js
#   ### static/
#   #   ### hello.txt
#   ### deploy-manifest.json
### bin/
#   ### .amplify-hosting/
#   #   ### compute/
#   #   #   ### default/
#   #   ### static/
#   ### postbuild.sh*
### dist/
#   ### index.js
### node_modules/
### public/
#   ### hello.txt
### src/
#   ### index.ts
### deploy-manifest.json
### package.json
### package-lock.json
### tsconfig.json
```

部署您的伺服器

1. 將程式碼推送至 Git 儲存庫，然後將應用程式部署至 Amplify Hosting。
2. 更新您的建置設定以指向 baseDirectory .amplify-hosting ，如下所示。在建置期間，Amplify 會偵測 .amplify-hosting目錄中的資訊清單檔案，並依設定部署您的 Express 伺服器。

```
version: 1
```

```
frontend:
  phases:
    preBuild:
      commands:
        - nvm use 18
        - npm install
    build:
      commands:
        - npm run build
  artifacts:
    baseDirectory: .amplify-hosting
    files:
      - '**/*'
```

3. 若要驗證您的部署是否成功，以及您的伺服器是否正常運作，請使用 Amplify Hosting 提供的預設 URL 造訪您的應用程式。

架構作者的影像最佳化整合

架構作者可以使用 Amplify 託管部署規格來整合 Amplify 的映像最佳化功能。若要啟用映像最佳化，您的部署資訊清單必須包含以映像最佳化服務為目標的路由規則。下列範例示範如何設定路由規則。

```
// .amplify-hosting/deploy-manifest.json

{
  "routes": [
    {
      "path": "/images/*",
      "target": {
        "kind": "ImageOptimization",
        "cacheControl": "public, max-age=31536000, immutable"
      }
    }
  ]
}
```

如需使用部署規格設定映像最佳化設定的詳細資訊，請參閱 使用 Amplify 託管部署規格來設定建置輸出 。

了解映像最佳化 API

您可以在執行時間透過 Amplify 應用程式的網域 URL，在路由規則定義的路徑叫用映像最佳化。

```
GET https://{appDomainName}/{path}?{queryParams}
```

映像最佳化會在映像上套用下列規則。

- Amplify 無法最佳化 GIF、APNG 和 SVG 格式，也無法將其轉換為其他格式。
- 除非啟用 `dangerouslyAllowSVG`設定，否則不會提供 SVG 影像。
- 來源影像的寬度或高度不能超過 11 MB 或 9，000 像素。
- 最佳化映像的大小限制為 4 MB。
- HTTPS 是唯一支援使用遠端 URLs通訊協定。

HTTP 標頭

Accept 請求 HTTP 標頭用於指定用戶端 （通常是 Web 瀏覽器） 允許的影像格式，以 MIME 類型表示。映像最佳化服務會嘗試將映像轉換為指定的格式。為此標頭指定的值具有比格式查詢參數更高的優先順序。例如，接受標頭的有效值為 `image/png, image/webp, */*` 。Amplify 部署資訊清單中指定的格式設定會將格式限制為清單中的格式。即使接受標頭要求特定格式，如果格式不在允許清單中，也會忽略它。

URI 請求參數

下表說明映像最佳化的 URI 請求參數。

查詢參數	Type	必要	Description	範例
url	String	是	來源映像的相對路徑或絕對 URL。對於遠端 URL，僅支援 https 通訊協定。值必須為 URL 編碼。	`?url=https%3A%2F%2Fwww.example.com%2Fbuffalo.png`
width	Number	是	最佳化影像的寬度，以像素為單位。	`?width=800`
height	Number	否	最佳化影像的像素高度。如果未	`?height=600`

查詢參數	Type	必要	Description	範例
			指定，影像將自動縮放以符合寬度。	
適合	列舉 值：`cover`、`cont` `outside`	否	如何調整影像大小以符合指定的寬度和高度。	`?width=800&height=600&fit=cover`
position	列舉 值：`center`、`top` `left`	否	適合時要使用的位置為 `cover`或 `contain`。	`?fit=contain&position=centre`
trim	Number	否	從包含與左上角像素指定背景顏色類似之值的所有邊緣修剪像素。	`?trim=50`
擴充	物件	否	使用從最接近的邊緣像素衍生的顏色，將像素新增至影像邊緣。格式為 `{top}_{right}_{bottom}_{left}` ，其中每個值為要新增的像素數。	`?extend=10_0_5_0`

查詢參數	Type	必要	Description	範例
擷取	物件	否	裁切影像至以頂端、左側、寬度和高度分隔的指定矩形。格式為 {left}_{top}_{width}_{right}，其中每個值都是要裁切的像素數。	`?extract=10_0_5_0`
格式	String	否	最佳化映像所需的輸出格式。	`?format=webp`
quality	Number	否	影像的品質，從 1 到 100。僅在轉換映像的格式時使用。	`?quality=50`
rotate	Number	否	依指定的角度旋轉影像，以度為單位。	`?rotate=45`
翻轉	Boolean	否	在 x 軸上垂直（向上）鏡射影像。這一律會在輪換之前發生，如果有的話。	`?flip`
flop	Boolean	否	在 y 軸上水平鏡射影像（左-右）。這一律會在輪換之前發生，如果有的話。	`?flop`

查詢參數	Type	必要	Description	範例
銳化	Number	否	銳化可增強影像中邊緣的定義。有效值介於 0.000001 和 10 之間。	`?sharpen=1`
median	Number	否	套用中位數篩選條件。這可消除雜訊或平滑影像的邊緣。	`?sharpen=3`
模糊	Number	否	套用指定標準差的高斯模糊。有效值介於 0.3 到 1，000 之間。	`?blur=20`
Gamma	Number	否	套用伽瑪校正，以改善調整大小影像的感知亮度。值必須介於 1.0 和 3.0 之間。	`?gamma=1`
否定	Boolean	否	反轉影像的顏色。	`?negate`
標準化	Boolean	否	透過擴展影像亮度以涵蓋完整的動態範圍來增強影像對比度。	`?normalize`

查詢參數	Type	必要	Description	範例
threshold	Number	否	如果影像中的像素強度小於指定的閾值，則以黑色像素取代影像中的任何像素。或者，如果大於閾值，則使用白色像素。有效值介於 0 到 255 之間。	`?threshold=155`
色調	String	否	使用提供的 RGB 調整影像，同時保留影像亮度。	`?tint=#7743CE`
灰階	Boolean	否	將影像變成灰階（黑色和白色）。	`?grayscale`

回應狀態碼

以下清單說明影像最佳化的回應狀態碼。

成功 - HTTP 狀態碼 200

請求已成功完成。

BadRequest - HTTP 狀態碼 400

- 輸入查詢參數的指定不正確。
- 遠端 URL 不會列為 `remotePatterns`設定中允許的 URL。
- 遠端 URL 不會解析為映像。
- 請求的寬度或高度不會在 `sizes`設定中列為允許。
- 請求的映像是 SVG，但`dangerouslyAllowSvg`設定已停用。

找不到 - HTTP 狀態碼 404

找不到來源映像。

內容太大 - HTTP 狀態碼 413

來源映像或最佳化映像超過允許的位元組大小上限。

了解最佳化映像快取

Amplify 託管會在 CDN 上快取最佳化映像，以便從快取提供對相同映像的後續請求與相同的查詢參數。快取存留時間 (TTL) 由 Cache-Control標頭控制。下列清單說明您指定 Cache-Control標頭的選項。

- 在以影像最佳化為目標的路由規則內使用 Cache-Control金鑰。
- 使用 Amplify 應用程式中定義的自訂標頭。
- 對於遠端映像，系統會遵守遠端映像傳回的Cache-Control標頭。

映像最佳化設定中minimumCacheTTL指定的 會定義Cache-Control max-age指令的下限。例如，如果遠端映像 URL 以 回應Cache-Control s-max-age=10，但 的值minimumCacheTTL為 60，則會使用 60。

對任何 SSR 架構使用開放原始碼轉接器

您可以使用為與 Amplify Hosting 整合而建立的任何 SSR 架構建置轉接器。每個提供轉接器的架構都會決定轉接器的設定方式，並連接到其建置程序。一般而言，您會將轉接器安裝為 npm 開發相依性。

使用架構建立應用程式後，請使用架構的文件，了解如何安裝 Amplify 託管轉接器，並在應用程式的組態檔案中進行設定。

接著，在專案的根目錄中建立 amplify.yml 檔案。在 amplify.yml檔案中，將 baseDirectory 設定為應用程式的建置輸出目錄。框架會在建置程序期間執行轉接器，將輸出轉換為 Amplify Hosting 部署套件。

組建輸出目錄的名稱可以是任何項目，但.amplify-hosting檔案名稱具有重要性。Amplify 首先會尋找定義為 的目錄baseDirectory。如果存在，Amplify 會在該處尋找建置輸出。如果目錄不存在，Amplify 會在 內尋找建置輸出.amplify-hosting，即使尚未由客戶定義。

以下是 應用程式的建置設定範例。baseDirectory 設定為 .amplify-hosting，表示建置輸出位於 .amplify-hosting 資料夾中。只要.amplify-hosting資料夾的內容符合 Amplify 託管部署規格，應用程式就會成功部署。

```
version: 1
```

```
frontend:
  preBuild:
    commands:
      - npm install
  build:
    commands:
      - npm run build
  artifacts:
    baseDirectory: .amplify-hosting
```

將應用程式設定為使用架構轉接器後，您可以將其部署至 Amplify Hosting。如需詳細說明，請參閱將 SSR 應用程式部署至 Amplify。

從 Amazon S3 儲存貯體將靜態網站部署至 Amplify

您可以使用 Amplify Hosting 和 Amazon S3 之間的整合，S3只要按幾下滑鼠，即可託管存放在 上的靜態網站內容。部署至 Amplify 託管可為您提供下列優點和功能。

- 自動部署到採用 CloudFront 技術的全域可用 AWS 內容交付網路 (CDN)
- HTTPS 支援
- 使用 Amplify 主控台輕鬆將網站連線至自訂網域
- 使用您自己的自訂 SSL 憑證
- 使用內建的存取日誌和 CloudWatch 指標來監控您的網站
- 為您的網站設定密碼保護
- 在 Amplify 主控台中建立重新導向和重寫規則

您可以從 Amplify 主控台、 AWS CLI或 AWS SDKs 啟動部署程序。您只能從位於您帳戶中的 Amazon S3 一般用途儲存貯體部署到 Amplify。Amplify 不支援跨帳戶S3儲存貯體存取。

當您將應用程式從 Amazon S3 一般用途儲存貯體部署到 Amplify Hosting 時， AWS 費用會根據 Amplify 的定價模型而定。如需詳細資訊，請參閱AWS Amplify 定價。

> **Important**
>
> Amplify 託管不適用於 Amazon S3 AWS 區域 提供的所有 。若要將靜態網站部署至 Amplify 託管，包含您網站的 Amazon S3 一般用途儲存貯體必須位於提供 Amplify 的區域。如需 Amplify 可用區域的清單，請參閱Amazon Web Services 一般參考中的 Amplify 端點。

請參閱下列主題，了解如何從 Amazon S3 部署和更新靜態網站至 Amplify 託管。

主題

- S3 使用 Amplify 主控台從 部署靜態網站
- 建立儲存貯體政策以S3使用 AWS SDKs 從 部署靜態網站
- 從 S3儲存貯體更新部署至 Amplify 的靜態網站
- 更新S3部署以使用儲存貯體和字首，而非 .zip 檔案

S3 使用 Amplify 主控台從 部署靜態網站

使用以下指示，使用 Amplify 主控台從 Amazon S3 一般用途儲存貯體部署新的靜態網站。

使用 Amplify 主控台從 Amazon S3 一般用途儲存貯體部署靜態網站

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/amplify/ 開啟 Amplify 主控台。
2. 在所有應用程式頁面上，選擇建立新應用程式。
3. 在開始使用 Amplify 建置頁面上，選擇不使用 Git 部署。
4. 選擇下一步。
5. 在啟動手動部署頁面上，執行下列動作。
   a. 在應用程式名稱中，輸入應用程式的名稱。
   b. 針對分支名稱，輸入要部署的分支名稱。
6. 針對方法，選擇 Amazon S3。
7. 針對S3要託管的物件位置，選擇瀏覽。選取要使用的 Amazon S3 一般用途儲存貯體，然後選取選擇字首。
8. 選擇 Save and deploy (儲存並部署)。

建立儲存貯體政策以S3使用 AWS SDKs 從 部署靜態網站

您可以使用 AWS SDKs 將靜態網站從 Amazon S3 部署到 Amplify 託管。如果您使用 SDK 部署網站，則必須建立自己的儲存貯體政策，授予 Amplify Hosting 擷取儲存S3貯體中物件的許可。

若要進一步了解如何建立儲存貯體政策，請參閱《Amazon Simple Storage Service 使用者指南》中的 Amazon S3 儲存貯體政策。

下列範例儲存貯體政策授予 Amplify 託管許可，以列出儲存貯體並擷取指定 AWS 帳戶、Amplify 應用程式 ID 和分支的儲存貯體物件。

若要使用此範例：

- 以您網站的儲存貯體名稱和*#### amzn-s3-demo-website-bucket/prefix*。
- 將 *111122223333* 取代為您的 AWS 帳戶 ID。
- 將 *region-id* 取代 AWS 區域 為您的 Amplify 應用程式所在的 ，例如 **us-east-1**。

- 將 *app_id* 取代為您 Amplify 應用程式 ID。此資訊可在 Amplify 主控台中取得。
- 將 *branch_name* 取代為您的分支名稱。

> **Note**
>
> 在您的儲存貯體政策中，aws:SourceArn 必須是 URL 編碼（百分比編碼）分支 ARN。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAmplifyToListPrefix_appid_branch_prefix_",
            "Effect": "Allow",
            "Principal": {
                "Service": "amplify.amazonaws.com"
            },
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::amzn-s3-demo-website-bucket/prefix/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333",
                    "aws:SourceArn": "arn%3Aaws%3Aamplify%3Aregion-
id%3A111122223333%3Aapps%2Fapp_id%2Fbranches%2Fbranch_name",
                    "s3:prefix": ""
                }
            }
        },
        {
            "Sid": "AllowAmplifyToReadPrefix__appid_branch_prefix_",
            "Effect": "Allow",
            "Principal": {
                "Service": "amplify.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-website-bucket/prefix/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333",
```

```
                    "aws:SourceArn": "arn%3Aaws%3Aamplify%3Aregion-
id%3A111122223333%3Aapps%2Fapp_id%2Fbranches%2Fbranch_name"
                }
            }
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::amzn-s3-demo-website-bucket/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "false"
                }
            }
        }
    ]
}
```

從 S3儲存貯體更新部署至 Amplify 的靜態網站

如果您在 Amplify 上託管的一般用途S3 儲存貯體中更新靜態網站的任何物件，您必須將應用程式重新部署至 Amplify Hosting，以使變更生效。Amplify 託管不會自動偵測儲存S3貯體的變更。我們建議您使用 AWS Command Line Interface (CLI) 來更新您的網站。

同步更新至 S3

變更網站專案檔案後，請使用下列 s3 同步命令，將您對本機來源目錄所做的變更與目標 Amazon S3 一般用途儲存貯體同步。若要使用此範例，請將 *<source>* 取代為本機目錄的名稱，並將 *<target>* 取代為 Amazon S3 儲存貯體的名稱。

```
aws s3 sync <source> <target>
```

將網站重新部署到 Amplify 託管

使用下列 amplify start-deployment 命令，在 Amazon S3 儲存貯體中將更新的應用程式重新部署至 Amplify Hosting。若要使用此範例，請將 *<app_id>* 取代為您的 Amplify 應用程式的 ID、將 *<branch_name>* 取代為您的分支名稱，並將 *s3#//amzn-s3-demo-website-bucket/prefix* 取代為您的S3儲存貯體和字首。

```
aws amplify start-deployment --app-id <app_id> --branch-name <branch_name> --source-
url s3://amzn-s3-demo-website-bucket/prefix --source-url-type BUCKET_PREFIX
```

更新S3部署以使用儲存貯體和字首，而非 .zip 檔案

如果您已從 Amazon S3 一般用途儲存貯體中的 .zip 檔案將現有的靜態網站部署至 Amplify Hosting，您可以更新應用程式部署，以使用包含要託管物件的儲存貯體名稱和字首。這種部署類型不需要將個別檔案上傳至包含建置輸出壓縮內容的儲存貯體。

將靜態網站從 .zip 檔案遷移至儲存貯體內容

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/amplify/ 開啟 Amplify 主控台。
2. 在所有應用程式頁面上，選擇您要從使用 .zip 檔案遷移至直接使用應用程式檔案的手動部署應用程式名稱。
3. 在應用程式的概觀頁面上，選擇部署更新。
4. 在部署更新頁面上，針對方法選擇 Amazon S3。
5. 針對S3要託管的物件位置，選擇瀏覽。選取要使用的儲存貯體，然後選取選擇字首。
6. 選擇 Save and deploy (儲存並部署)。

在沒有 Git 儲存庫的情況下將應用程式部署至 Amplify

手動部署可讓您使用 Amplify Hosting 發佈 Web 應用程式，而無需連接 Git 供應商。您可以從桌面拖放壓縮的資料夾，並在幾秒鐘內託管您的網站。或者，您可以參考 Amazon S3 儲存貯體中的資產，或指定檔案存放位置的公有 URL。

> **Note**
>
> 由於 Amazon S3 複製操作限制，手動部署的 .zip 檔案大小上限為 5GB。如果任何建置成品超過此大小，請考慮將其分成較小的封存檔，或使用替代的部署方法。

對於 Amazon S3，您也可以設定 AWS Lambda 觸發，在每次上傳新資產時更新您的網站。如需設定此案例的詳細資訊，請參閱將 AWS Amplify 存放在 Amazon S3、Dropbox 或桌面上的檔案部署至主控台部落格文章。

Amplify 託管不支援伺服器端轉譯 (SSR) 應用程式的手動部署。如需詳細資訊，請參閱使用 Amplify Hosting 部署伺服器端轉譯應用程式。

拖放手動部署

使用拖放手動部署應用程式

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 在右上角，選擇建立新應用程式。
3. 在使用 Amplify 開始建置頁面上，選擇不使用 Git 部署。然後選擇下一步。
4. 在啟動手動部署頁面上，針對應用程式名稱輸入應用程式的名稱。
5. 針對分支名稱，輸入有意義的名稱，例如 **`development`**或 **`production`**。
6. 針對方法，選擇拖放。
7. 您可以從桌面將資料夾拖放到放置區域，或使用選擇 .zip 資料夾從電腦選取檔案。您拖放或選取的檔案必須是包含建置輸出內容的壓縮資料夾。
8. 選擇 Save and deploy (儲存並部署)。

Amazon S3 或 URL 手動部署

> **Note**
>
> 如果您要從 部署靜態網站S3，下列程序會要求您將包含建置輸出內容的壓縮資料夾上傳至儲存S3貯體。我們建議您S3使用儲存貯體名稱和字首直接從 部署靜態網站。如需此簡化程序的詳細資訊，請參閱 從 Amazon S3 儲存貯體將靜態網站部署至 Amplify。

從 Amazon S3 或公有 URL 手動部署應用程式

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 在右上角，選擇建立新應用程式。
3. 在使用 Amplify 開始建置頁面上，選擇不使用 Git 部署。然後選擇下一步。
4. 在啟動手動部署頁面上，針對應用程式名稱輸入應用程式的名稱。
5. 針對分支名稱，輸入有意義的名稱，例如 **`development`**或 **`production`**。
6. 針對方法，選擇 Amazon S3 或任何 URL。
7. 上傳檔案的程序取決於上傳方法。
   - Amazon S3
     a. 針對 S3 location of objects to host，選擇瀏覽 S3。然後，從清單中選擇 Amazon S3 儲存貯體的名稱。您選取的儲存貯體必須啟用存取控制清單 (ACLs)。如需詳細資訊，請參閱針對手動部署的 Amazon S3 儲存貯體存取進行故障診斷。
     b. 選取要部署的 .zip 檔案名稱。
     c. 選擇選擇字首。
   - 任何 URL
     - 針對資源 URL，輸入要部署之 .zip 檔案的 URL。
8. 選擇 Save and deploy (儲存並部署)。

> **Note**
>
> 當您建立壓縮資料夾時，請務必壓縮建置輸出的內容，而不是頂層資料夾。例如，如果您的建置輸出產生名為「建置」或「公有」的資料夾，請先導覽至該資料夾，選取所有內容，然後從

該處將其壓縮。如果您不這樣做，您會看到「拒絕存取」錯誤，因為網站的根目錄不會正確初始化。

針對手動部署的 Amazon S3 儲存貯體存取進行故障診斷

當您建立 Amazon S3 儲存貯體時，您可以使用其 Amazon S3 物件擁有權設定來控制是否啟用或停用儲存貯體的存取控制清單 (ACLs)。若要從 Amazon S3 儲存貯體手動將應用程式部署至 Amplify，必須在儲存貯體上啟用 ACLs。

如果您從 Amazon S3 儲存貯體部署`AccessControlList`時發生錯誤，儲存貯體會在停用 ACLs 的情況下建立，您必須在 Amazon S3 主控台中啟用。如需說明，請參閱《Amazon Simple Storage Service 使用者指南》中的在現有儲存貯體上設定物件擁有權。

管理 Amplify 應用程式的建置組態

您可以自訂 Amplify 部署的建置設定和組態。部署應用程式時，Amplify 會自動偵測前端架構和相關聯的建置設定。您可以在應用程式的建置規格 (buildspec) 中自訂建置設定，以新增環境變數、執行建置命令，以及指定建置相依性。

Amplify 的預設建置映像隨附預先安裝的數個套件和相依性，但您也可以使用即時套件更新功能來指定特定版本，或確保一律安裝最新版本。如果您有特定的相依性，在使用 Amplify 預設容器的建置期間需要很長時間才能安裝 ，您可以建立自己的自訂建置映像。您也可以自訂建置執行個體大小，以提供應用程式部署所需的 CPU、記憶體和磁碟空間資源。

組建會在每次遞交至 Git 儲存庫以及每次新部署時自動啟動。您可以設定傳入的 Webhook 功能來啟動組建，而無需遞交您的 Git 儲存庫。

建置通知功能可讓您與團隊成員分享建置成功和失敗的相關資訊。

主題

- 設定 Amplify 應用程式的建置設定
- 自訂建置映像
- 設定 Amplify 應用程式的建置執行個體
- 建立傳入 Webhook 以啟動組建
- 設定組建的電子郵件通知

設定 Amplify 應用程式的建置設定

部署應用程式時，Amplify 會透過檢查 Git 儲存庫中的應用程式`package.json`檔案，自動偵測前端架構和相關聯的建置設定。您可以使用下列選項來儲存應用程式的建置設定：

- 在 Amplify 主控台中儲存建置設定 - Amplify 主控台會自動偵測建置設定並儲存，以便 Amplify 主控台存取這些設定。Amplify 會將這些設定套用至所有分支，除非您的儲存庫中存放了 `amplify.yml` 檔案。
- 將建置設定儲存在儲存庫中 - 下載 `amplify.yml` 檔案，並將其新增至儲存庫的根目錄。

> **Note**
>
> 只有當應用程式設定為持續部署並連接到 git 儲存庫時，建置設定才會顯示在 Amplify 主控台的託管選單中。如需此部署類型的說明，請參閱 入門。

組建規格參考

Amplify 應用程式的建置規格 (buildspec) 是一組 YAML 設定和建置命令，供 Amplify 用來執行您的建置。下列清單說明這些設定及其使用方式。

version

Amplify YAML 版本編號。

appRoot

此應用程式所在儲存庫內的路徑。除非定義多個應用程式，否則會忽略。

env

將環境變數新增至本節。您也可以使用主控台新增環境變數。

後端

執行 Amplify CLI 命令來佈建後端、更新 Lambda 函數或 GraphQL 結構描述，作為持續部署的一部分。

前端

執行前端建置命令。

test

在測試階段期間執行命令。了解如何將測試新增至您的應用程式。

建置階段

前端、後端和測試有三個階段，代表建置的每個序列期間執行的命令。

- preBuild - preBuild 指令碼會在實際建置開始之前執行，但在 Amplify 安裝相依性之後執行。
- build - 您的建置命令。
- postBuild - 建置完成後執行建置後指令碼，且 Amplify 已將所有必要成品複製到輸出目錄。

buildpath

用來執行建置的路徑。Amplify 使用此路徑來尋找您的建置成品。如果您未指定路徑，Amplify 會使用 monorepo 應用程式根，例如 `apps/app`。

artifacts>base-directory

您建置成品所在的目錄。

artifacts>files

從您要部署的成品中指定檔案。輸入 `**/*` 以包含所有檔案。

快取

指定建置時間相依性，例如 node_modules 資料夾。在第一次建置期間，會快取此處提供的路徑。在後續建置中，Amplify 會在執行命令之前，將快取還原至相同的路徑。

Amplify 會將所有提供的快取路徑視為與您的專案根目錄相對。不過，Amplify 不允許在專案根目錄之外周遊。例如，如果您指定絕對路徑，組建將會成功而沒有錯誤，但路徑不會快取。

組建規格 YAML 語法參考

下列組建規格範例示範基本 YAML 語法。

```
version: 1
env:
  variables:
    key: value
backend:
  phases:
    preBuild:
      commands:
        - *enter command*
    build:
      commands:
        - *enter command*
    postBuild:
        commands:
        - *enter command*
frontend:
  buildpath:
  phases:
```

```
    preBuild:
      commands:
        - cd react-app
        - npm ci
    build:
      commands:
        - npm run build
  artifacts:
    files:
        - location
        - location
    discard-paths: yes
    baseDirectory: location
  cache:
    paths:
        - path # A cache path relative to the project root
        - path # Traversing outside of the project root is not allowed
test:
  phases:
    preTest:
      commands:
        - *enter command*
    test:
      commands:
        - *enter command*
    postTest:
      commands:
        - *enter command*
  artifacts:
    files:
        - location
        - location
    configFilePath: *location*
    baseDirectory: *location*
```

編輯建置規格

您可以在 Amplify 主控台中編輯建置規格 (buildspec)，以自訂應用程式的建置設定。建置設定會套用至您應用程式中的所有分支，但 Git 儲存庫中儲存了 `amplify.yml` 檔案的分支除外。

在 Amplify 主控台中編輯建置設定

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。

2. 選擇您要編輯建置設定的應用程式。
3. 在導覽窗格中，選擇託管，然後選擇建置設定。
4. 在建置設定頁面的應用程式建置規格區段中，選擇編輯。
5. 在編輯建置規格視窗中，輸入您的更新。
6. 選擇儲存。

您可以使用下列主題中所述的範例來更新特定案例的建置設定。

主題

- 使用指令碼設定分支特定的建置設定
- 設定命令以導覽至子資料夾
- 使用 Gen 1 應用程式的前端部署後端
- 設定輸出資料夾
- 在組建中安裝套件
- 使用私有 npm 登錄檔
- 安裝 OS 套件
- 為每個組建設定鍵值儲存
- 略過遞交的建置
- 關閉每個遞交上的自動建置
- 設定以差異為基礎的前端建置和部署
- 為 Gen 1 應用程式設定差異型後端建置

使用指令碼設定分支特定的建置設定

您可以使用 bash shell 指令碼來配置分支特定的建置設定。例如，下列指令碼使用系統環境變數 $AWS_BRANCH，在分支名稱為主要時執行一組命令，而在分支名稱為開發時執行另一組命令。

```
frontend:
  phases:
    build:
      commands:
        - if [ "${AWS_BRANCH}" = "main" ]; then echo "main branch"; fi
        - if [ "${AWS_BRANCH}" = "dev" ]; then echo "dev branch"; fi
```

設定命令以導覽至子資料夾

對於 monorepos，使用者希望能夠在cd資料夾中執行建置。執行 cd命令後，它會套用至建置的所有階段，因此您不需要分階段重複執行命令。

```
version: 1
env:
  variables:
    key: value
frontend:
  phases:
    preBuild:
      commands:
        - cd react-app
        - npm ci
    build:
      commands:
        - npm run build
```

使用 Gen 1 應用程式的前端部署後端

> **Note**
>
> 本節僅適用於 Amplify Gen 1 應用程式。第 1 代後端是使用 Amplify Studio 和 Amplify 命令列界面 (CLI) 建立。

amplifyPush 命令是一種協助程式指令碼，可協助您進行後端部署。下面的建置設定會自動判斷要為目前分支部署的正確後端環境。

```
version: 1
env:
  variables:
    key: value
backend:
  phases:
    build:
      commands:
        - amplifyPush --simple
```

設定輸出資料夾

以下建置設定會將輸出目錄設定為公有資料夾。

```
frontend:
  phases:
    commands:
      build:
        - yarn run build
  artifacts:
    baseDirectory: public
```

在組建中安裝套件

您可以使用 npm或 yarn命令在建置期間安裝套件。

```
frontend:
  phases:
    build:
      commands:
        - npm install -g <package>
        - <package> deploy
        - yarn run build
  artifacts:
    baseDirectory: public
```

使用私有 npm 登錄檔

您可以新增對您的建置設定中私有登錄的參考，或將它新增為環境變數。

```
build:
  phases:
    preBuild:
      commands:
        - npm config set <key> <value>
        - npm config set registry https://registry.npmjs.org
        - npm config set always-auth true
        - npm config set email hello@amplifyapp.com
        - yarn install
```

安裝 OS 套件

Amplify 的 AL2023 映像會使用名為 的非特殊權限使用者來執行您的程式碼amplify。Amplify 授予此使用者使用 Linux 命令執行作業系統sudo命令的權限。如果您想要為缺少的相依性安裝作業系統套件，您可以使用 和 等命令yumrpm搭配 sudo。

下列範例建置區段示範使用 sudo命令安裝作業系統套件的語法。

```
build:
  phases:
    preBuild:
      commands:
        - sudo yum install -y <package>
```

為每個組建設定鍵值儲存

會在建置時envCache提供鍵值儲存。儲存在 中的值envCache只能在建置期間修改，並可在下一次建置時重複使用。使用 envCache，我們可以將資訊存放在已部署的環境上，並連續提供建置容器使用。與存放在 中的值不同envCache，在建置期間的環境變數變更不會保留到未來的建置。

使用範例：

```
envCache --set <key> <value>
envCache --get <key>
```

略過遞交的建置

若要略過特定遞交的自動建置，請在遞交訊息結尾加入文字 【skip-cd】。

關閉每個遞交上的自動建置

您可以設定 Amplify 來關閉每個程式碼遞交的自動建置。若要設定，請選擇應用程式設定、分支設定，然後尋找列出已連接分支的分支區段。選取分支，然後選擇動作、停用自動建置。對該分支的新遞交將不再啟動新的組建。

設定以差異為基礎的前端建置和部署

您可以設定 Amplify 使用 diff 型前端建置。如果啟用，在每個組建開始時，Amplify 會嘗試在您的 appRoot或 /src/ 資料夾上執行 diff。如果 Amplify 找不到任何差異，它會略過前端建置、測試（如果已設定）和部署步驟，而且不會更新您的託管應用程式。

設定以 diff 為基礎的前端建置和部署

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇要設定 diff 型前端建置和部署的應用程式。
3. 在導覽窗格中，選擇託管、環境變數。
4. 在環境變數區段中，選擇管理變數。
5. 設定環境變數的程序會根據您是否啟用或停用以差異為基礎的前端建置和部署而有所不同。
   - 啟用 diff 型前端建置和部署
     a. 在管理變數區段的變數下，輸入 AMPLIFY_DIFF_DEPLOY。
     b. 針對數值，輸入 true。
   - 停用以 diff 為基礎的前端建置和部署
     - 執行以下任意一項：
       - 在管理變數區段中，找到 AMPLIFY_DIFF_DEPLOY。針對數值，輸入 false。
       - 移除AMPLIFY_DIFF_DEPLOY環境變數。
6. 選擇儲存。

或者，您可以設定AMPLIFY_DIFF_DEPLOY_ROOT環境變數，以相對於儲存庫根目錄的路徑覆寫預設路徑，例如 dist。

為 Gen 1 應用程式設定差異型後端建置

> **Note**
> 本節僅適用於 Amplify Gen 1 應用程式。第 1 代後端是使用 Amplify Studio 和 Amplify 命令列界面 (CLI) 建立。

您可以使用 AMPLIFY_DIFF_BACKEND環境變數，將 Amplify 託管設定為使用 diff 型後端建置。當您啟用 diff 型後端建置時，在每個建置開始時，Amplify 會嘗試在儲存庫中的amplify資料夾上執行 diff。如果 Amplify 找不到任何差異，它會略過後端建置步驟，而不會更新您的後端資源。如果您的專案在儲存庫中沒有 amplify 資料夾，Amplify 會忽略AMPLIFY_DIFF_BACKEND環境變數的值。

如果您目前在後端階段的建置設定中指定了自訂命令，則條件式後端建置將無法運作。如果您希望這些自訂命令執行，則必須將其移至您應用程式amplify.yml檔案中建置設定的前端階段。

設定 diff 型後端建置

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇要設定 diff 型後端建置的應用程式。
3. 在導覽窗格中，選擇託管、環境變數。
4. 在環境變數區段中，選擇管理變數。
5. 環境變數的設定程序會因您是否啟用或停用 diff 型後端建置而有所不同。
   - 啟用 diff 型後端建置
     a. 在管理變數區段的變數下，輸入 AMPLIFY_DIFF_BACKEND。
     b. 針對數值，輸入 `true`。
   - 停用 diff 型後端建置
     - 執行以下任意一項：
       - 在管理變數區段中，找到 AMPLIFY_DIFF_BACKEND。針對數值，輸入 `false`。
       - 移除AMPLIFY_DIFF_BACKEND環境變數。
6. 選擇儲存。

設定 monorepo 組建設定

當您在單一儲存庫中存放多個專案或微服務時，稱為單一儲存庫。您可以使用 Amplify Hosting 在單一儲存庫中部署應用程式，而無需建立多個建置組態或分支組態。

Amplify 支援通用 monorepos 中的應用程式，以及使用 npm 工作區、pnpm 工作區、Yarn 工作區、Nx 和 Turborepo 建立的 monorepo 中的應用程式。部署應用程式時，Amplify 會自動偵測您正在使用的 monorepo 建置工具。Amplify 會自動套用 npm 工作區、Yarn 工作區或 Nx 中應用程式的建置設定。Turborepo 和 pnpm 應用程式需要額外的組態。如需詳細資訊，請參閱設定 Turborepo 和 pnpm monorepo 應用程式。

您可以在 Amplify 主控台中儲存單一儲存庫的建置設定，也可以下載 `amplify.yml` 檔案並將其新增至儲存庫的根目錄。Amplify 會將主控台中儲存的設定套用至所有分支，除非其在您的儲存庫中找到 `amplify.yml` 檔案。當`amplify.yml`檔案存在時，其設定會覆寫 Amplify 主控台中儲存的任何建置設定。

Monorepo 建置規格 YAML 語法參考

單一儲存庫建置規格的 YAML 語法與包含單一應用程式的儲存庫的 YAML 語法不同。對於單一儲存庫，您可以在應用程式清單中宣告每個專案。您必須為在 monorepo 建置規格中宣告的每個應用程式提供下列額外`appRoot`金鑰：

appRoot

應用程式啟動所在的儲存庫根目錄。此金鑰必須存在，且具有與`AMPLIFY_MONOREPO_APP_ROOT`環境變數相同的值。如需設定此環境變數的說明，請參閱 設定 AMPLIFY_MONOREPO_APP_ROOT 環境變數。

下列單儲存庫建置規格範例示範如何在同一個儲存庫中宣告多個 Amplify 應用程式。這兩個應用程式`react-app`和 `angular-app`會在`applications`清單中宣告。每個應用程式的`appRoot`金鑰表示應用程式位於儲存庫的`apps`根資料夾中。

`buildpath` 屬性設定為 `/`，以從 monorepo 專案根目錄執行和建置應用程式。`baseDirectory` 屬性是 的相對路徑`buildpath`。

Monorepo 建置規格 YAML 語法

```
version: 1
applications:
  - appRoot: apps/react-app
    env:
      variables:
        key: value
    backend:
      phases:
        preBuild:
          commands:
            - *enter command*
        build:
          commands:
            - *enter command*
        postBuild:
            commands:
            - *enter command*
    frontend:
      buildPath: / # Run install and build from the monorepo project root
      phases:
        preBuild:
```

```
        commands:
          - *enter command*
          - *enter command*
      build:
        commands:
          - *enter command*
    artifacts:
      files:
          - location
          - location
      discard-paths: yes
      baseDirectory: location
    cache:
      paths:
          - path
          - path
  test:
    phases:
      preTest:
        commands:
          - *enter command*
      test:
        commands:
          - *enter command*
      postTest:
        commands:
          - *enter command*
    artifacts:
      files:
          - location
          - location
      configFilePath: *location*
      baseDirectory: *location*
- appRoot: apps/angular-app
  env:
    variables:
      key: value
  backend:
    phases:
      preBuild:
        commands:
          - *enter command*
      build:
        commands:
```

```
            - *enter command*
        postBuild:
            commands:
            - *enter command*
    frontend:
      phases:
        preBuild:
          commands:
            - *enter command*
            - *enter command*
        build:
          commands:
            - *enter command*
      artifacts:
        files:
            - location
            - location
        discard-paths: yes
        baseDirectory: location
      cache:
        paths:
            - path
            - path
    test:
      phases:
        preTest:
          commands:
            - *enter command*
        test:
          commands:
            - *enter command*
        postTest:
          commands:
            - *enter command*
      artifacts:
        files:
            - location
            - location
        configFilePath: *location*
        baseDirectory: *location*
```

使用下列範例建置規格的應用程式，將建置在專案根目錄下，而建置成品將位於 /packages/nextjs-app/.next。

```
applications:
  - frontend:
      buildPath: '/'  # run install and build from monorepo project root
      phases:
        preBuild:
          commands:
            - npm install
        build:
          commands:
            - npm run build --workspace=nextjs-app
      artifacts:
        baseDirectory: packages/nextjs-app/.next
        files:
          - '**/*'
      cache:
        paths:
          - node_modules/**/*
    appRoot: packages/nextjs-app
```

設定 AMPLIFY_MONOREPO_APP_ROOT 環境變數

當您部署存放在單一儲存庫中的應用程式時，應用程式AMPLIFY_MONOREPO_APP_ROOT的環境變數必須具有與應用程式根目錄路徑相同的值，相對於儲存庫根目錄。例如，名為 ExampleMonorepo且根資料夾名為 的 monorepoapps，其中包含 、 app1 app2和 app3的目錄結構如下：

```
ExampleMonorepo
  apps
    app1
    app2
    app3
```

在此範例中， AMPLIFY_MONOREPO_APP_ROOT的環境變數值app1為 apps/app1。

當您使用 Amplify 主控台部署 monorepo 應用程式時，主控台會自動使用您為應用程式根目錄路徑指定的值來設定AMPLIFY_MONOREPO_APP_ROOT環境變數。不過，如果您的 monorepo 應用程式已存在於 Amplify 中或使用 部署 AWS CloudFormation，您必須在 Amplify 主控台AMPLIFY_MONOREPO_APP_ROOT的環境變數區段中手動設定環境變數。

在部署期間自動設定 AMPLIFY_MONOREPO_APP_ROOT 環境變數

下列指示示範如何使用 Amplify 主控台部署 monorepo 應用程式。Amplify 會使用您在主控台中指定的應用程式根資料夾自動設定`AMPLIFY_MONOREPO_APP_ROOT`環境變數。

使用 Amplify 主控台部署 monorepo 應用程式

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇右上角的新建應用程式。
3. 在開始使用 Amplify 建置頁面上，選擇您的 Git 供應商，然後選擇下一步。
4. 在新增儲存庫分支頁面上，執行下列動作：
   a. 從清單中選擇儲存庫的名稱。
   b. 選擇要使用的分支名稱。
   c. 選取 我的應用程式是單一儲存庫
   d. 在 monorepo 中輸入應用程式的路徑，例如 **apps/app1**。
   e. 選擇下一步。
5. 在應用程式設定頁面上，您可以使用預設設定或自訂應用程式的建置設定。在環境變數區段中，Amplify `AMPLIFY_MONOREPO_APP_ROOT`會將 設定為您在步驟 4d 中指定的路徑。
6. 選擇下一步。
7. 在檢閱頁面上，選擇儲存並部署。

設定現有應用程式的 AMPLIFY_MONOREPO_APP_ROOT 環境變數

使用下列指示，為已部署至 Amplify 或使用 CloudFormation 建立的應用程式手動設定`AMPLIFY_MONOREPO_APP_ROOT`環境變數。

設定現有應用程式的 AMPLIFY_MONOREPO_APP_ROOT 環境變數

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇要為其設定環境變數的應用程式名稱。
3. 在導覽窗格中，選擇託管，然後選擇環境變數。
4. 在環境變數頁面上，選擇管理變數。
5. 在管理變數區段中，執行下列動作：
   a. 選擇 Add new (新增)。

b. 針對變數，輸入金鑰 AMPLIFY_MONOREPO_APP_ROOT。

c. 針對值，輸入應用程式的路徑，例如 **apps/app1**。

d. 對於分支，根據預設，Amplify 會將環境變數套用至所有分支。

6. 選擇儲存。

設定 Turborepo 和 pnpm monorepo 應用程式

Turborepo 和 pnpm 工作區 monorepo 建置工具會從 .npmrc 檔案取得組態資訊。當您部署使用其中一個工具建立的 monorepo 應用程式時，您的專案根目錄中必須有 .npmrc 檔案。

在 .npmrc檔案中，將安裝 Node 套件的連結器設定為 hoisted。您可以將以下行複製到您的 檔案。

```
node-linker=hoisted
```

如需.npmrc檔案和設定的詳細資訊，請參閱 pnpm 文件中的 pnpm .npmrc。

Pnpm 不包含在 Amplify 預設建置容器中。對於 pnpm 工作區和 Turborepo 應用程式，您必須新增命令，才能在應用程式的建置設定preBuild階段中安裝 pnpm。

下列範例摘錄自建置規格，顯示具有安裝 pnpm 命令的preBuild階段。

```
version: 1
applications:
  - frontend:
      phases:
        preBuild:
          commands:
            - npm install -g pnpm
```

自訂建置映像

您可以使用自訂建置映像，為 Amplify 應用程式提供自訂建置環境。如果您有特定的相依性，在使用 Amplify 預設容器的建置期間需要很長時間才能安裝 ，您可以建立自己的 Docker 映像，並在建置期間加以參考。映像可以在 Amazon Elastic Container Registry Public 上託管。

若要讓自訂建置映像做為 Amplify 建置映像，它必須符合下列要求。

自訂建置映像需求

1. 支援 GNU C Library (glibc) 的 Linux 發行版本，例如針對 x86-64 架構編譯的 Amazon Linux。
2. cURL：在啟動您的自訂影像時，我們會下載建置執行器至容器，因此需要有 cURL。如果缺少此相依性，則建置會立即失敗而沒有任何輸出，因為我們的建置執行器無法產生任何輸出。
3. Git：該影像上需安裝 Git，才能複製 Git 儲存庫。如果缺少此相依性，複製儲存庫步驟將會失敗。
4. OpenSSH：為了安全地複製您的儲存庫，我們需要 OpenSSH 在建置期間暫時設定 SSH 金鑰。OpenSSH 套件提供建置執行器執行此作業所需的命令。
5. Bash 和 Bourne Shell：這兩個公用程式用於在建置時執行命令。如果未安裝，您的組建可能會在開始之前失敗。
6. Node.JS+NPM：我們的建置執行器不會安裝 Node。相反地，它依賴於在映像中安裝的節點和 NPM。只有需要 NPM 套件或 Node 特定命令的建置，才要進行此操作。不過，我們強烈建議安裝它們，因為當它們存在時，Amplify 建置執行器可以使用這些工具來改善建置執行。當您為 Hugo 設定覆寫時，Amplify 的套件覆寫功能會使用 NPM 安裝 Hugo 延伸套件。

下列套件並非必要項目，但強烈建議您安裝。

1. NVM (Node Version Manager)：如果您需要處理不同版本的 ，建議您安裝此版本管理員Node。當您設定覆寫時，Amplify 的套件覆寫功能會在每次建置之前NVM使用 來變更 Node.js 版本。
2. Wget：Amplify 可以使用 Wget公用程式在建置程序期間下載檔案。建議您將其安裝在自訂映像中。
3. Tar：Amplify 可以使用 Tar公用程式，在建置過程中解壓縮下載的檔案。建議您將其安裝在自訂映像中。

設定應用程式的自訂建置映像

使用下列程序，在 Amplify 主控台中設定應用程式的自訂建置映像。

設定 Amazon ECR 中託管的自訂建置映像

1. 請參閱 Amazon ECR 公有使用者指南中的入門，以使用 Docker 映像設定 Amazon ECR 公有儲存庫。
2. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
3. 選擇您要為其設定自訂建置映像的應用程式。
4. 在導覽窗格中，選擇託管、建置設定。
5. 在建置設定頁面的建置映像設定區段中，選擇編輯。
6. 在編輯建置映像設定頁面上，展開建置映像功能表，然後選擇自訂建置映像。

7. 輸入您在步驟一中建立的 Amazon ECR Public 儲存庫名稱。這是您的建置映像託管的位置。例如，如果您的儲存庫名稱是 ecr-examplerepo，您會輸入 **`public.ecr.aws/xxxxxxxx/ecr-examplerepo`**。
8. 選擇儲存。

在建置映像中使用特定套件和相依性版本

即時套件更新可讓您指定要在 Amplify 預設建置映像中使用的套件和相依性版本。預設建置映像隨附預先安裝的數個套件和相依性（例如 Hugo、Amplify CLI、Yarn 等）。透過即時套件更新，您可以覆寫這些相依性的版本，並指定特定版本，或確保一律安裝最新版本。

如果啟用即時套件更新，則在建置執行之前，建置執行器會先更新（或降級）指定的相依性。這會根據更新相依性所需的時間來增加建置時間，但優點是您可以確保使用相同版本的相依性來建置應用程式。

> **Warning**
>
> 將 Node.js 版本設定為最新會導致建置失敗。反之，您必須指定確切的 Node.js 版本，例如 `18`、 `21.5`或 `v0.1.2`。

設定即時套件更新

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇您要為其設定即時套件更新的應用程式。
3. 在導覽窗格中，選擇託管、建置設定。
4. 在建置設定頁面的建置映像設定區段中，選擇編輯。
5. 在編輯建置映像設定頁面上，即時套件更新清單，選擇新增。
6. 針對套件，選取要覆寫的相依性。
7. 對於版本，請保持預設最新或輸入特定版本的相依性。如果您使用最新版本，相依性將一律升級至可用的最新版本。
8. 選擇儲存。

設定 Amplify 應用程式的建置執行個體

Amplify Hosting 提供可設定的建置執行個體大小，可讓您提供應用程式的建置執行個體所需的 CPU、記憶體和磁碟空間資源。在發行此功能之前，Amplify 提供了 8 GiB 記憶體和 4 個 vCPUs的固定大小建置執行個體組態。

Amplify 支援三種建置執行個體類型：Standard、 Large和 XLarge。如果您未指定執行個體類型，Amplify 會使用預設Standard執行個體。您可以使用 Amplify 主控台 AWS CLI、 或 SDKs 來設定應用程式的建置執行個體類型。

每個組建執行個體類型的成本會按組建分鐘計算。如需定價詳細資訊，請參閱 AWS Amplify 定價。

下表說明每個建置執行個體類型的運算規格：

組建執行個體類型	vCPUs	記憶體	磁碟空間
Standard	4 vCPUs	8 GiB	128 GB
Large	8 vCPUs	16 GiB	128 GB
XLarge	36 vCPUs	72 GiB	256 GB

主題

- 了解建置執行個體類型
- 在 Amplify 主控台中設定建置執行個體類型
- 設定應用程式的堆積記憶體以利用大型執行個體類型

了解建置執行個體類型

建置執行個體類型設定是在應用程式層級設定，並延伸至應用程式的所有分支。下列金鑰詳細資訊適用於建置執行個體類型：

- 您為應用程式設定的建置執行個體類型會自動套用至自動建立的分支和提取請求預覽。
- 並行任務服務配額適用於您 中的所有建置執行個體類型 AWS 帳戶。例如，如果您的並行任務限制為 5，您最多可以在 中的所有執行個體類型中執行 5 個組建 AWS 帳戶。

- 每個組建執行個體類型的成本會按組建分鐘計算。建置執行個體配置程序可能需要額外的額外額外負荷時間，才能開始建置。對於較大的執行個體，特別是 XLarge，由於此額外負荷時間，您的建置可能會在建置開始之前遇到延遲。不過，您只需支付實際建置時間的費用，而非額外負荷時間。

您可以在建立新應用程式時設定建置執行個體類型，也可以更新現有應用程式的執行個體類型。如需在 Amplify 主控台中設定此設定的說明，請參閱 在 Amplify 主控台中設定建置執行個體類型。您也可以使用 SDKs更新此設定。如需詳細資訊，請參閱 Amplify APIs 參考中的 CreateApp 和 UpdateApp API。

如果您的帳戶中有在可自訂建置執行個體類型功能發行之前建立的現有應用程式，則會使用預設 Standard執行個體類型。當您更新現有應用程式的建置執行個體類型時，任何在更新之前排入佇列或進行中的建置都會使用先前設定的建置執行個體類型。例如，如果您有部署到 Amplify 的`main`分支的現有應用程式，並且將其建置執行個體類型從標準更新為大型，則從`main`分支啟動的所有新建置將使用大型建置執行個體類型。不過，在您更新組建執行個體類型時正在進行的任何組建都會繼續在標準執行個體上執行。

在 Amplify 主控台中設定建置執行個體類型

當您建立新的 Amplify 應用程式時，請使用下列程序來設定建置執行個體類型。

設定新應用程式的建置執行個體類型

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 在所有應用程式頁面上，選擇建立新應用程式。
3. 在開始使用 Amplify 建置頁面上，選擇您的 Git 儲存庫提供者，然後選擇下一步。
4. 在新增儲存庫分支頁面上，執行下列動作：
   a. 在最近更新的儲存庫清單中，選取要連線的儲存庫名稱。
   b. 在分支清單中，選取要連線的儲存庫分支名稱。
   c. 選擇下一步。
5. 在應用程式設定頁面上，開啟進階設定區段。
6. 針對建置執行個體類型，從清單中選擇所需的執行個體類型。
7. 如果您要部署以 Node.js 執行期為基礎的應用程式，請設定堆積記憶體大小，以有效利用大型執行個體類型。您可以透過設定環境變數或更新建置設定，在應用程式設定頁面上執行此操作。如需詳細資訊，請參閱設定應用程式的堆積記憶體以利用大型執行個體類型。
   - 設定環境變數
     a. 在進階設定、環境變數區段中，選擇新增。

b. 針對金鑰輸入 **NODE_OPTIONS**。

c. 針對數值，輸入 `--max-old-space-size=`*`memory_size_in_mb`*。將 *`memory_size_in_mb`* 取代為所需的堆積記憶體大小，以 MB 為單位。

- 更新建置設定

a. 在建置設定區段中，選擇編輯 YML 檔案。

b. 將下列命令新增至 `preBuild`階段。將 *`memory_size_in_mb`* 取代為所需的堆積記憶體大小，以 MB 為單位。

```
export NODE_OPTIONS='--max-old-space-size=memory_size_in_mb'
```

c. 選擇儲存。

8. 選擇下一步。
9. 在檢閱頁面上，選擇儲存並部署。

使用下列程序來設定現有 Amplify 應用程式的建置執行個體類型。

設定現有應用程式的建置執行個體類型

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇您要為其設定建置執行個體類型的應用程式。
3. 在導覽窗格中，選擇託管，選擇建置設定。
4. 在建置設定頁面的進階設定區段中，選擇編輯。
5. 在編輯設定頁面上，針對建置執行個體類型，從清單中選擇所需的執行個體類型。
6. 選擇儲存。此變更將在您下次部署應用程式時生效。
7. （選用）若要立即部署更新的應用程式，請執行下列動作：

a. 在導覽窗格中，選擇概觀。

b. 在應用程式的概觀頁面上，選擇要重新部署的分支。

c. 在部署頁面上，選擇部署，例如最新的部署。然後，選擇重新部署此版本。新的部署將會開始。

d. 部署完成時，應用程式的建置設定會顯示分支正在使用更新的建置執行個體類型。

設定應用程式的堆積記憶體以利用大型執行個體類型

如果您要建置記憶體密集型應用程式，請使用本節了解如何設定應用程式以利用大型執行個體類型。在執行時間期間，程式設計語言和架構通常依賴配置動態記憶體，也稱為堆積記憶體，以管理應用程式記憶體需求。堆積記憶體由執行時間環境請求，並由主機作業系統配置。根據預設，執行時間環境會強制執行應用程式可用的堆積大小上限。這表示即使主機作業系統或容器有較多的記憶體可用，應用程式也無法使用超過堆積大小的額外記憶體。

例如，JavaScript Node.js v8 執行時間環境會強制執行預設堆積大小限制，這取決於多種因素，包括主機記憶體大小。因此， Standard和 Large 組建執行個體的預設 Node.js 堆積大小為 2096 MB，而 XLarge執行個體的預設堆積大小為 4144 MB。因此，在任何 Amplify 建置執行個體類型上使用預設 Node.js 堆積大小，以 6000 MB 記憶體需求建置應用程式，將導致因out-of-memory錯誤而導致建置失敗。

若要解決 Node.js 預設堆積大小記憶體限制，您可以執行下列其中一項作業：

- 將 Amplify 應用程式中NODE_OPTIONS的環境變數設定為值 `--max-old-space-size=`*`memory_size_in_mb`*。針對 memory_size_in_mb，以 MB 為單位指定您想要的堆積記憶體大小。

  如需說明，請參閱設定環境變數。

- 將下列命令新增至 Amplify 應用程式建置規格中的 preBuild階段。

  ```
  export NODE_OPTIONS='--max-old-space-size=memory_size_in_mb'
  ```

  您可以在 Amplify 主控台或專案儲存庫的應用程式amplify.yml檔案中更新建置規格。如需說明，請參閱設定 Amplify 應用程式的建置設定。

  下列範例 Amplify 建置規格會將 Node.js 堆積記憶體大小設定為 7000 MB，以建置 React 前端應用程式：

  ```
  version: 1
  frontend:
    phases:
      preBuild:
        commands:
          # Set the heap size to 7000 MB
          - export NODE_OPTIONS='--max-old-space-size=7000'
          # To check the heap size memory limit in MB
  ```

```
        - node -e "console.log('Total available heap size (MB):',
v8.getHeapStatistics().heap_size_limit / 1024 / 1024)"
        - npm ci --cache .npm --prefer-offline
    build:
      commands:
        - npm run build
  artifacts:
    baseDirectory: build
    files:
      - '**/*'
  cache:
    paths:
      - .npm/**/*
```

若要有效利用大型執行個體類型，請務必設定足夠的堆積記憶體大小。為記憶體密集型應用程式設定小型堆積大小可能會導致建置失敗。應用程式建置日誌可能不會直接指出out-of-memory錯誤，因為應用程式執行時間可能會意外當機。將堆積大小設定為主機記憶體的大小，可能會導致主機作業系統交換或終止其他程序，並可能中斷您的建置程序。作為參考，Node.js 建議在具有大約 2000 MB 記憶體的機器上設定最大堆積大小為 1536 MB，以保留一些記憶體供其他用途使用。

最佳堆積大小取決於應用程式的需求和資源用量。如果您遇到out-of-memory錯誤，請從中等堆積大小開始，然後視需要逐漸增加。作為準則，我們建議從Standard執行個體類型的 6000 MB、Large執行個體類型的 12000 MB 和XLarge執行個體類型的 60000 MB 開始。

建立傳入 Webhook 以啟動組建

在 Amplify 主控台中設定傳入 Webhook 以啟動組建，而無需將程式碼遞交至 Git 儲存庫。您可以使用 Webhook 搭配無周邊 CMS 工具（例如 Contentful 或 GraphCMS)，在內容變更時啟動組建，或使用 Zapier 等服務執行每日組建。

建立傳入 Webhook

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇您要為其建立 Webhook 的應用程式。
3. 在導覽窗格中，選擇託管，然後選擇建置設定。
4. 在建置設定頁面上，向下捲動至傳入 Webhook 區段，然後選擇建立 Webhook。
5. 在建立 Webhook 對話方塊中，執行下列動作：

    a. 針對 Webhook 名稱，輸入 Webhook 的名稱。

b. 針對要建置的分支，選取要在傳入 Webhook 請求上建置的分支。

c. 選擇建立 Webhook。

6. 在傳入 Webhook 區段中，執行下列其中一項操作：

   - 複製 Webhook URL，並將其提供給無周邊 CMS 工具或其他 服務以啟動組建。
   - 在終端機視窗中執行 curl 命令，以啟動新的組建。

設定組建的電子郵件通知

您可以為 AWS Amplify 應用程式設定電子郵件通知，以在建置成功或失敗時提醒利益相關者或團隊成員。Amplify 託管會在您的帳戶中建立 Amazon Simple Notification Service (SNS) 主題，並使用它來設定電子郵件通知。通知可設定為套用至 Amplify 應用程式的所有分支或特定分支。

設定電子郵件通知

使用下列程序來設定 Amplify 應用程式的所有分支或特定分支的電子郵件通知。

設定 Amplify 應用程式的電子郵件通知

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇您要為其設定電子郵件通知的應用程式。
3. 在導覽窗格中，選擇託管、建置通知。在建置通知頁面上，選擇管理通知。
4. 在管理通知頁面上，選擇新增。
5. 執行以下任意一項：
   - 若要傳送單一分支的通知，請在電子郵件中輸入要傳送通知的電子郵件地址。針對分支，選取要傳送通知的分支名稱。
   - 若要傳送所有連線分支的通知，請在電子郵件中輸入要傳送通知的電子郵件地址。針對分支，選擇所有分支。
6. 選擇儲存。

連接自訂網域

您可以將已使用 Amplify Hosting 部署的應用程式連線至自訂網域。當您使用 Amplify 部署 Web 應用程式時，Amplify 會使用 URL 為您託管在預設amplifyapp.com網域上，例如 `https://branch-name.d1m7bkiki6tdw1.amplifyapp.com`。當您將應用程式連線至自訂網域時，使用者會看到您的應用程式託管在自訂 URL 上，例如 `https://www.example.com`。

您可以透過 Amazon Route 53 或 GoDaddy 等認可的網域註冊商購買自訂網域。Route 53 是 Amazon 的網域名稱系統 (DNS) Web 服務。如需使用 Route 53 的詳細資訊，請參閱什麼是 Amazon Route 53。如需第三方認可網域註冊商的清單，請參閱 ICANN 網站上的認可註冊商目錄。

設定自訂網域時，您可以使用 Amplify 為您佈建的預設受管憑證，也可以使用自己的自訂憑證。您可以隨時變更網域正在使用的憑證。如需管理憑證的詳細資訊，請參閱 使用 SSL/TLS 憑證。

在您繼續設定自訂網域之前，請確認您符合下列先決條件。

- 您擁有已註冊的網域名稱。
- 您有由 發行或匯入 的憑證 AWS Certificate Manager。
- 您已將應用程式部署到 Amplify Hosting。

  如需完成此步驟的詳細資訊，請參閱 開始將應用程式部署到 Amplify 託管。
- 您對網域和 DNS 術語有基本的了解。

  如需網域和 DNS 的詳細資訊，請參閱 了解 DNS 術語和概念。

> **Warning**
>
> 為 Amplify 應用程式啟動 DomainAssociation (Amplify 應用程式） 請求時，如果網域已經或先前已與相同區域中其他 AWS 帳戶中的不同 Amplify 應用程式相關聯，則這會被視為跨帳戶網域關聯。跨帳戶網域關聯請求需要手動驗證。如果您想要繼續進行跨帳戶網域關聯，請聯絡 AWS Support 尋求協助。

主題

- 了解 DNS 術語和概念
- 使用 SSL/TLS 憑證

- 新增由 Amazon Route 53 管理的自訂網域
- 新增由第三方 DNS 供應商管理的自訂網域
- 更新由 GoDaddy 管理之網域的 DNS 記錄
- 更新網域的 SSL/TLS 憑證
- 管理子網域
- 設定萬用字元子網域
- 設定 Amazon Route 53 自訂網域的自動子網域
- 對自訂網域進行故障診斷

了解 DNS 術語和概念

如果您不熟悉與網域名稱系統 (DNS) 相關聯的術語和概念，下列主題可協助您了解新增自訂網域的程序。

DNS 術語

以下是 DNS 常用的詞彙清單。他們可協助您了解新增自訂網域的程序。

CNAME

正式記錄名稱 (CNAME) 是一種 DNS 記錄，可遮罩一組網頁的網域，使其看起來像位於其他地方。CNAME 會將子網域指向完整網域名稱 (FQDN)。例如，您可以建立新的 CNAME 記錄，將子網域 www.example.com，其中 www 是子網域，對應至 Amplify 主控台中指派給您應用程式的 FQDN 網域 branch-name.d1m7bkiki6tdw1.cloudfront.net。

ANAME

ANAME 記錄就像 CNAME 記錄，但位於根層級。ANAME 會將網域的根指向 FQDN。該 FQDN 指向 IP 地址。

名稱伺服器

名稱伺服器是網際網路上的伺服器，專門處理有關網域名稱各種服務位置的查詢。如果您在 Amazon Route 53 中設定網域，名稱伺服器清單已指派給您的網域。

NS 記錄

NS 記錄會指向查詢網域詳細資訊的名稱伺服器。

DNS 驗證

網域名稱系統 (DNS) 就像一本電話簿，將人類可讀取的網域名稱轉譯為適合電腦的 IP 地址。當您**https://google.com**在瀏覽器中輸入 時，會在 DNS 供應商中執行查詢操作，以尋找託管網站的伺服器的 IP 地址。

DNS 提供者包含網域記錄及其對應的 IP 地址。最常用的 DNS 記錄是 CNAME、ANAME 和 NS 記錄。

Amplify 使用 CNAME 記錄來驗證您擁有自訂網域。如果您使用 Route 53 託管網域，則會代表您自動完成驗證。不過，如果您使用 GoDaddy 等第三方供應商託管網域，則必須手動更新網域的 DNS 設定，並新增 Amplify 提供的新 CNAME 記錄。

自訂網域啟用程序

> **Warning**
>
> 為 Amplify 應用程式啟動 DomainAssociation (Amplify 應用程式） 請求時，如果網域已經或先前已與相同區域中其他 AWS 帳戶中的不同 Amplify 應用程式相關聯，則這會被視為跨帳戶網域關聯。跨帳戶網域關聯請求需要手動驗證。如果您想要繼續進行跨帳戶網域關聯，請聯絡 AWS Support 尋求協助。

當您將 Amplify 應用程式連線到 Amplify 主控台中的自訂網域時，Amplify 必須先完成幾個步驟，才能使用自訂網域檢視您的應用程式。下列清單說明網域設定和啟用程序中的每個步驟。

SSL/TLS 建立

如果您使用受管憑證， 會 AWS Amplify 發出 SSL/TLS 憑證來設定安全的自訂網域。

SSL/TLS 組態和驗證

在發出受管憑證之前，Amplify 會驗證您是網域的擁有者。對於 Amazon Route 53 管理的網域，Amplify 會自動更新 DNS 驗證記錄。對於在 Route 53 外部管理的網域，您必須使用第三方 DNS 供應商，手動將 Amplify 主控台中提供的 DNS 驗證記錄新增至您的網域。

如果您使用的是自訂憑證，您必須負責驗證網域擁有權。

網域啟用

網域已成功驗證。對於 Route 53 外部管理的網域，您需要使用第三方 DNS 供應商，手動將 Amplify 主控台中提供的 CNAME 記錄新增至您的網域。

使用 SSL/TLS 憑證

SSL/TLS 憑證是一種數位文件，可讓 Web 瀏覽器使用安全 SSL/TLS 通訊協定來識別和建立與網站的加密網路連線。設定自訂網域時，您可以使用 Amplify 為您佈建的預設受管憑證，也可以使用自己的自訂憑證。

使用受管憑證，Amplify 會為連接到應用程式的所有網域發出 SSL/TLS 憑證，以便透過 HTTPS/2 保護所有流量。 AWS Certificate Manager (ACM) 產生的預設憑證有效期為 13 個月，只要您的應用程式使用 Amplify 託管，則會自動續約。

> **Warning**
>
> 如果在網域提供者的 DNS 設定中修改或刪除 CNAME 驗證記錄，Amplify 無法續約憑證。您必須在 Amplify 主控台中刪除並再次新增網域。

若要使用自訂憑證，您必須先從您選擇的第三方憑證授權單位取得憑證。Amplify 託管支援兩種類型的憑證：RSA (Rivest-Shamir-Adleman) 和 ECDSA (Elliptic Curve 數位簽章演算法）。每個憑證類型都必須符合下列要求。

RSA 憑證

- Amplify 託管支援 1024 位元、2048 位元、3072 位元和 4096 位元 RSA 金鑰。
- AWS Certificate Manager (ACM) 發行最多 2048 位元金鑰的 RSA 憑證。
- 若要使用 3072 位元或 4096 位元 RSA 憑證，請在外部取得憑證並將其匯入 ACM。然後，它將可用於 Amplify 託管。

ECDSA 憑證

- Amplify 託管支援 256 位元金鑰。
- 使用 prime256v1 橢圓曲線來取得 Amplify 託管的 ECDSA 憑證。

取得憑證後，將其匯入 AWS Certificate Manager。ACM 是一項服務，可讓您輕鬆佈建、管理和部署公有和私有 SSL/TLS 憑證，以搭配 AWS 服務 和您的內部連線資源使用。請務必在美國東部（維吉尼亞北部）(us-east-1) 區域中請求或匯入憑證。

請確定您的自訂憑證涵蓋您計劃新增的所有子網域。您可以在網域名稱開頭使用萬用字元來涵蓋多個子網域。例如，如果您的網域是 `example.com`，您可以包含萬用字元網域 `*.example.com`。這將涵蓋子網域，例如 `product.example.com`和 `api.example.com`。

在 ACM 中提供自訂憑證之後，您就可以在網域設定程序期間選取它。如需將憑證匯入 的指示 AWS Certificate Manager，請參閱AWS Certificate Manager 《 使用者指南》中的將憑證匯入 AWS Certificate Manager。

如果您在 ACM 中續約或重新匯入自訂憑證，Amplify 會重新整理與自訂網域相關聯的憑證資料。如果是匯入的憑證，ACM 不會自動管理續約。您有責任續約自訂憑證，並重新匯入。

您可以隨時變更網域正在使用的憑證。例如，您可以從預設受管憑證切換至自訂憑證，或從自訂憑證變更為受管憑證。此外，您可以將正在使用的自訂憑證變更為不同的自訂憑證。如需更新憑證的指示，請參閱更新網域的 SSL/TLS 憑證。

新增由 Amazon Route 53 管理的自訂網域

Amazon Route 53 是高度可用且可擴展的 DNS 服務。如需詳細資訊，請參閱《Amazon Route 53 開發人員指南》中的 Amazon Route 53。如果您已有 Route 53 網域，請使用下列指示將自訂網域連線至 Amplify 應用程式。

新增由 Route 53 管理的自訂網域

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇您要連接到自訂網域的應用程式。
3. 在導覽窗格中，選擇託管、自訂網域。
4. 在自訂網域頁面上，選擇新增網域。
5. 輸入根網域的名稱。例如，如果您的網域名稱是 https://example.com，請輸入 **`example.com`**。

   當您開始輸入時，您在 Route 53 中管理的任何根網域都會出現在清單中。您可以從清單中選擇要使用的網域。如果您尚未擁有網域，而且該網域可供使用，您可以在 Amazon Route 53 中購買該網域。
6. 輸入網域名稱後，請選擇設定網域。
7. 根據預設，Amplify 會自動為您的網域建立兩個子網域項目。例如，如果您的網域名稱是 example.com，您會看到子網域 https://www.example.com 和 https://example.com，其中包含從根網域到 www 子網域的重新導向設定。

（選用）如果您只想要新增子網域，您可以修改預設組態。若要變更預設組態，請從導覽窗格選擇重寫和重新導向，然後設定您的網域。

8. 選擇要使用的 SSL/TLS 憑證。您可以使用 Amplify 為您佈建的預設受管憑證，或您匯入的自訂第三方憑證 AWS Certificate Manager。
   - 使用預設 Amplify 受管憑證。
     - 選擇 Amplify 受管憑證。
   - 使用自訂第三方憑證。
     a. 選擇自訂 SSL 憑證。
     b. 從清單中選擇要使用的憑證。
9. 選擇新增網域。

> **Note**
>
> DNS 傳播和發出憑證最多可能需要 24 小時。如需解決發生錯誤的說明，請參閱 對自訂網域進行故障診斷。

新增由第三方 DNS 供應商管理的自訂網域

如果您不是使用 Amazon Route 53 來管理您的網域，您可以將由第三方 DNS 提供者管理的自訂網域新增至使用 Amplify 部署的應用程式。

如果您使用的是 GoDaddy，請參閱 the section called “更新由 GoDaddy 管理之網域的 DNS 記錄” 以取得此提供者的特定指示。

新增由第三方 DNS 供應商管理的自訂網域

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇您要新增自訂網域的應用程式。
3. 在導覽窗格中，選擇託管、自訂網域。
4. 在自訂網域頁面上，選擇新增網域。
5. 輸入根網域的名稱。例如，如果您的網域名稱是 https://example.com，請輸入 **`example.com`**。
6. Amplify 偵測到您未使用 Route 53 網域，並提供您在 Route 53 中建立託管區域的選項。

- 在 Route 53 中建立託管區域

  a. 選擇在 Route 53 上建立託管區域。

  b. 選擇設定網域。

  c. 託管區域名稱伺服器會顯示在主控台中。前往 DNS 提供者的網站，並將名稱伺服器新增至您的 DNS 設定。

  d. 選取我已將上述名稱伺服器新增至我的網域登錄檔。

  e. 繼續進行步驟 7。

- 繼續手動設定

  a. 選擇手動組態

  b. 選擇設定網域。

  c. 繼續進行步驟 7。

7. 根據預設，Amplify 會自動為您的網域建立兩個子網域項目。例如，如果您的網域名稱是 example.com，您會看到子網域 https://www.example.com 和 https://example.com，其中包含從根網域到 www 子網域的重新導向設定。

   （選用）如果您只想要新增子網域，您可以修改預設組態。若要變更預設組態，請從導覽窗格選擇重寫和重新導向，並設定您的網域。

8. 選擇要使用的 SSL/TLS 憑證。您可以使用 Amplify 為您佈建的預設受管憑證，或您匯入的自訂第三方憑證 AWS Certificate Manager。

   - 使用預設 Amplify 受管憑證。

     - 選擇 Amplify 受管憑證。

   - 使用自訂第三方憑證。

     a. 選擇自訂 SSL 憑證。

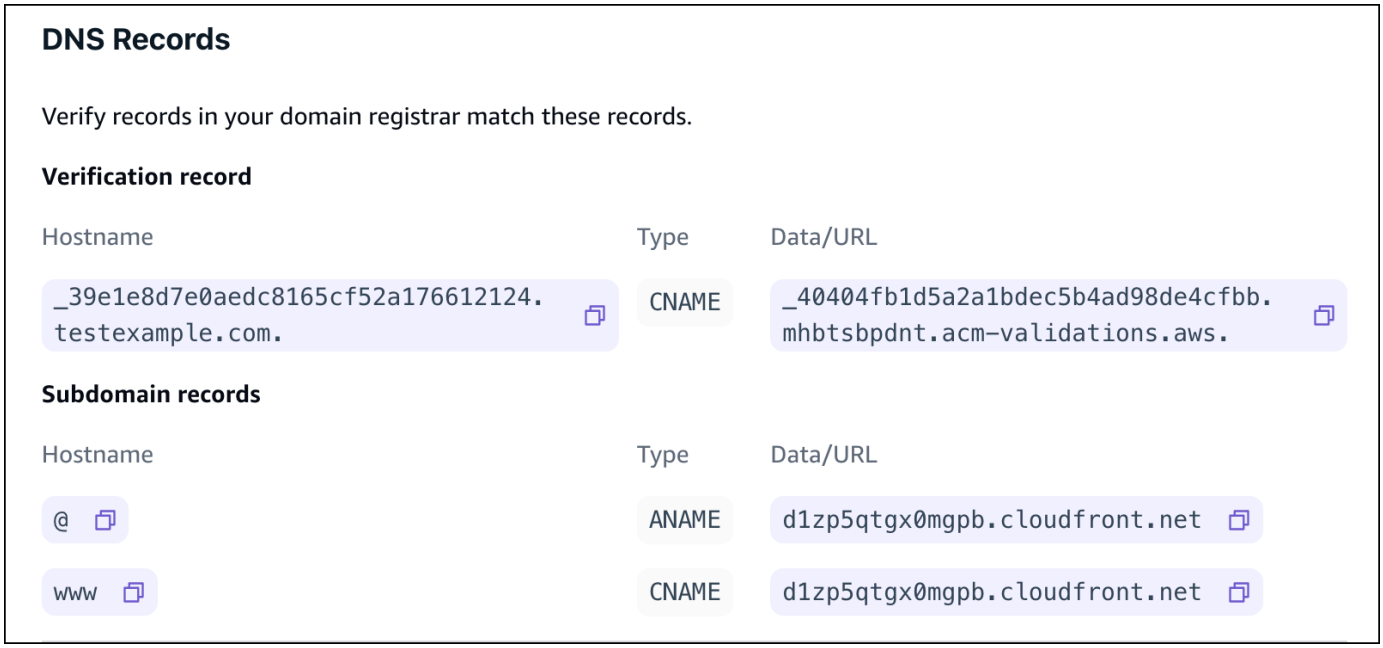
     b. 從清單中選擇要使用的憑證。

9. 選擇新增網域。

10. 如果您在步驟六中選擇在 Route 53 上建立託管區域，請繼續步驟 15。

    如果您選擇手動組態，在步驟六中，您必須使用第三方網域提供者更新您的 DNS 記錄。

    在動作功能表中，選擇檢視 DNS 記錄。下列螢幕擷取畫面顯示主控台中顯示的 DNS 記錄。



11. 執行以下任意一項：

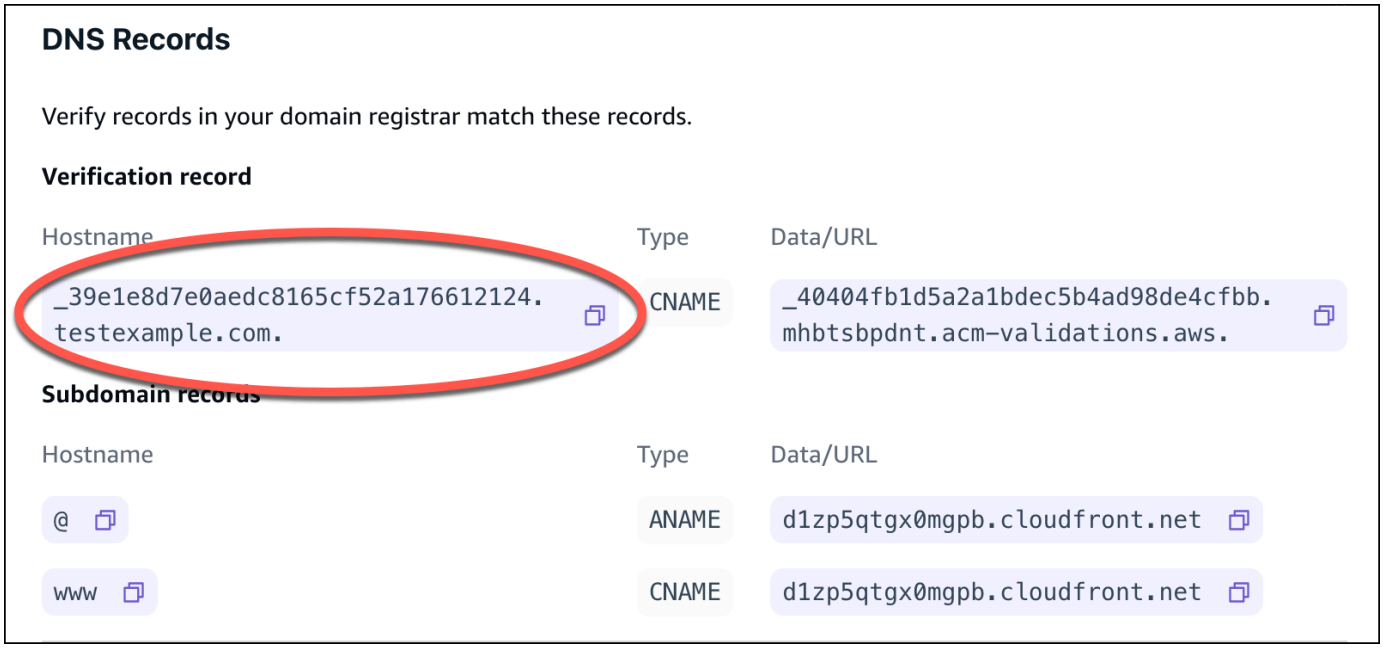
    - 如果您使用的是 GoDaddy，請前往 更新由 GoDaddy 管理之網域的 DNS 記錄。
    - 如果您使用的是不同的第三方 DNS 供應商，請前往此程序的下一個步驟。

12. 前往 DNS 提供者的網站，登入您的帳戶，並尋找網域的 DNS 管理設定。您將設定兩個 CNAME 記錄。

13. 設定第一個 CNAME 記錄，將您的子網域指向 AWS 驗證伺服器。

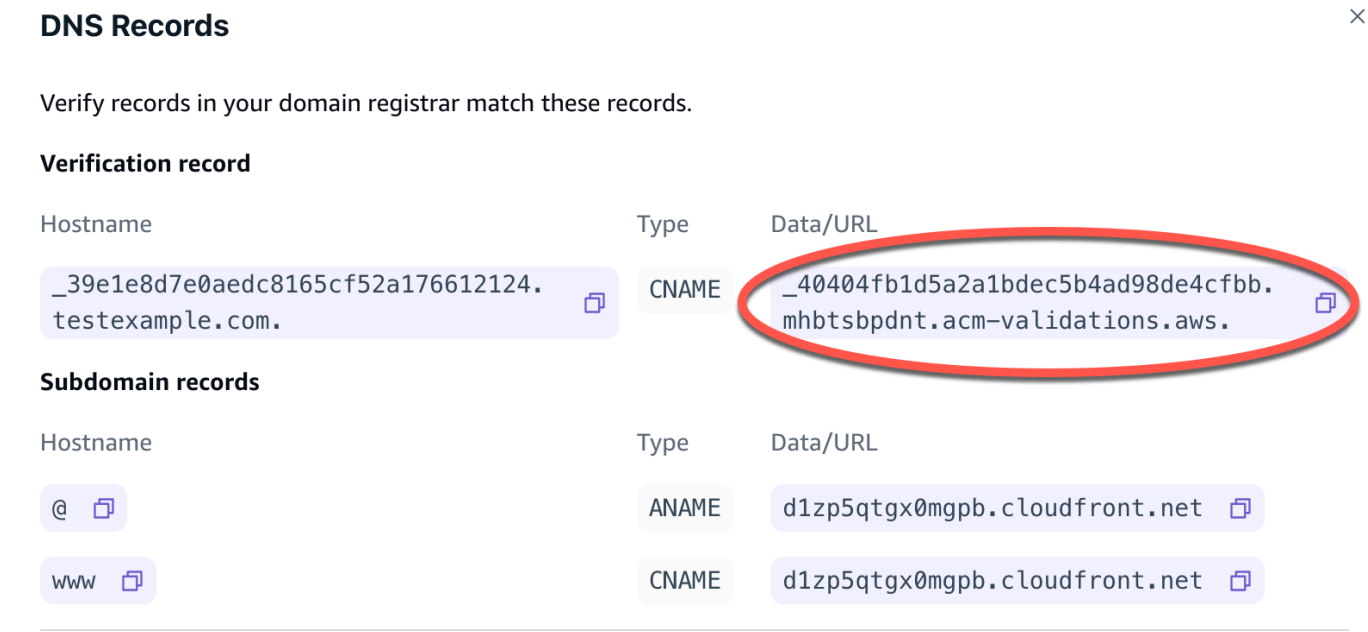
    如果 Amplify 主控台顯示 DNS 記錄來驗證子網域的擁有權，例如 _c3e2d7eaf1e656b73f46cd6980fdc0e.example.com，則僅針對 CNAME 記錄子網域名稱輸入 。**`_c3e2d7eaf1e656b73f46cd6980fdc0e`**

    下列螢幕擷取畫面顯示要使用的驗證記錄位置。



如果 Amplify 主控台顯示 ACM 驗證伺服器記錄，例如 _cjhwou20vhu2exampleuw20vuyb2ovb9.j9s73ucn9vy.acm-validations.aws，**`_cjhwou20vhu2exampleuw20vuyb2ovb9.j9s73ucn9vy.acm-validations.aws`請**在 CNAME 記錄值中輸入 。

下列螢幕擷取畫面顯示要使用的 ACM 驗證記錄位置。



Amplify 會使用此資訊來驗證網域的擁有權，並為您的網域產生 SSL/TLS 憑證。一旦 Amplify 驗證網域的擁有權，所有流量都將使用 HTTPS/2 提供。

> **Note**
>
> AWS Certificate Manager (ACM) 產生的預設 Amplify 憑證有效期為 13 個月，只要您的應用程式是使用 Amplify 託管，則會自動續約。如果修改或刪除 CNAME 驗證記錄，Amplify 無法續約憑證。您必須在 Amplify 主控台中刪除並再次新增網域。

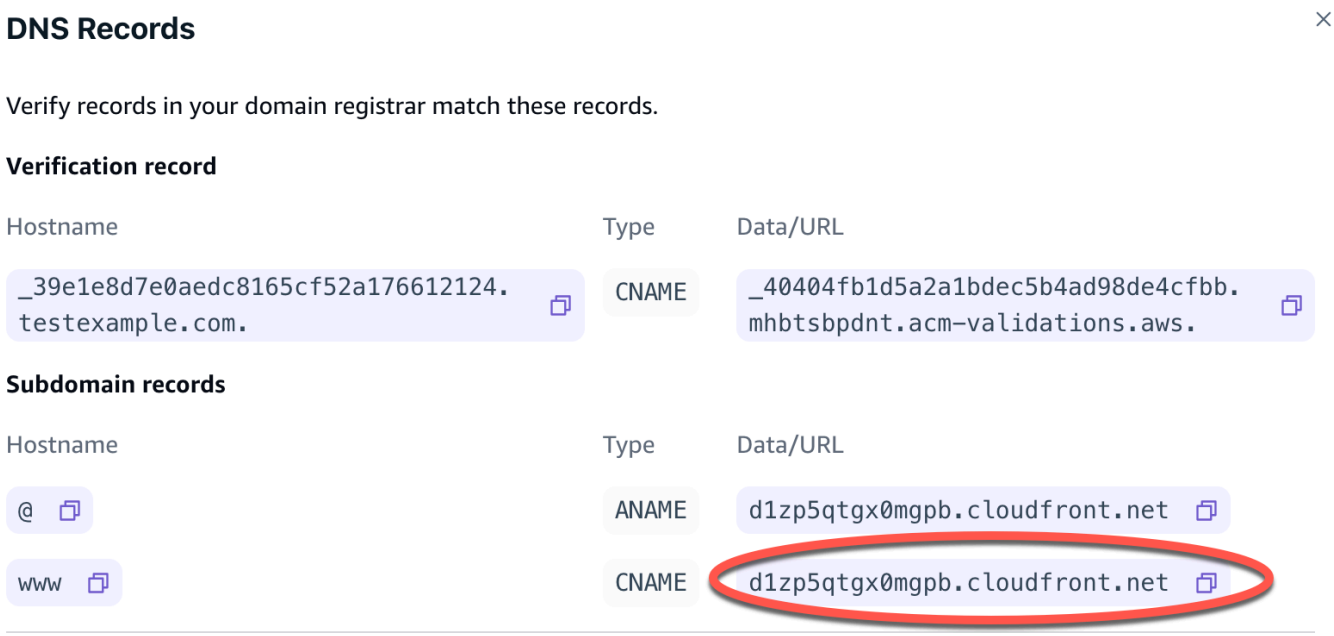
> **Important**
>
> 在 Amplify 主控台中新增自訂網域之後，請務必立即執行此步驟。 AWS Certificate Manager (ACM) 立即開始嘗試驗證擁有權。隨著時間的推移，檢查頻率會降低。如果您在建立應用程式數小時後新增或更新 CNAME 記錄，這可能會導致應用程式卡在等待驗證狀態。

14. 設定第二個 CNAME 記錄，將您的子網域指向 Amplify 網域。例如，如果您的子網域是 www.example.com，請輸入子網域名稱的 www。

    如果 Amplify 主控台將應用程式的網域顯示為 d111111abcdef8.cloudfront.net，**`d111111abcdef8.cloudfront.net`**請在 Amplify 網域中輸入 。

    如果您有生產流量，我們建議您在網域狀態在 Amplify 主控台中顯示為可用之後更新此 CNAME 記錄。

    下列螢幕擷取畫面顯示要使用的網域名稱記錄位置。



15. 設定 ANAME/ALIAS 記錄以指向應用程式的根網域（例如 https://example.com)。ANAME 記錄會將網域的根指向主機名稱。如果您有生產流量，我們建議您在網域狀態在主控台中顯示為可用之後更新您的 ANAME 記錄。對於沒有 ANAME/ALIAS 支援的 DNS 供應商，強烈建議將您的 DNS 遷移至 Route 53。如需詳細資訊，請參閱將 Amazon Route 53 設定為 DNS 服務。

> **Note**
>
> 驗證第三方網域的網域擁有權和 DNS 傳播最多可能需要 48 小時。如需解決錯誤的說明，請參閱對自訂網域進行故障診斷。

更新由 GoDaddy 管理之網域的 DNS 記錄

如果 GoDaddy 是您的 DNS 供應商，請使用下列指示來更新 GoDaddy UI 中的 DNS 記錄，以完成將您的 Amplify 應用程式連線至 GoDaddy 網域。

新增由 GoDaddy 管理的自訂網域

1. 在使用 GoDaddy 更新 DNS 記錄之前，請先完成程序 的步驟一到九the section called “新增由第三方 DNS 供應商管理的自訂網域”。
2. 登入您的 GoDaddy 帳戶。
3. 在網域清單中，尋找要新增的網域，然後選擇管理 DNS。

4. 在 DNS 頁面上，GoDaddy 會在 DNS 記錄區段中顯示網域的記錄清單。您需要新增兩個新的 CNAME 記錄。
5. 建立第一個 CNAME 記錄，將您的子網域指向 Amplify 網域。

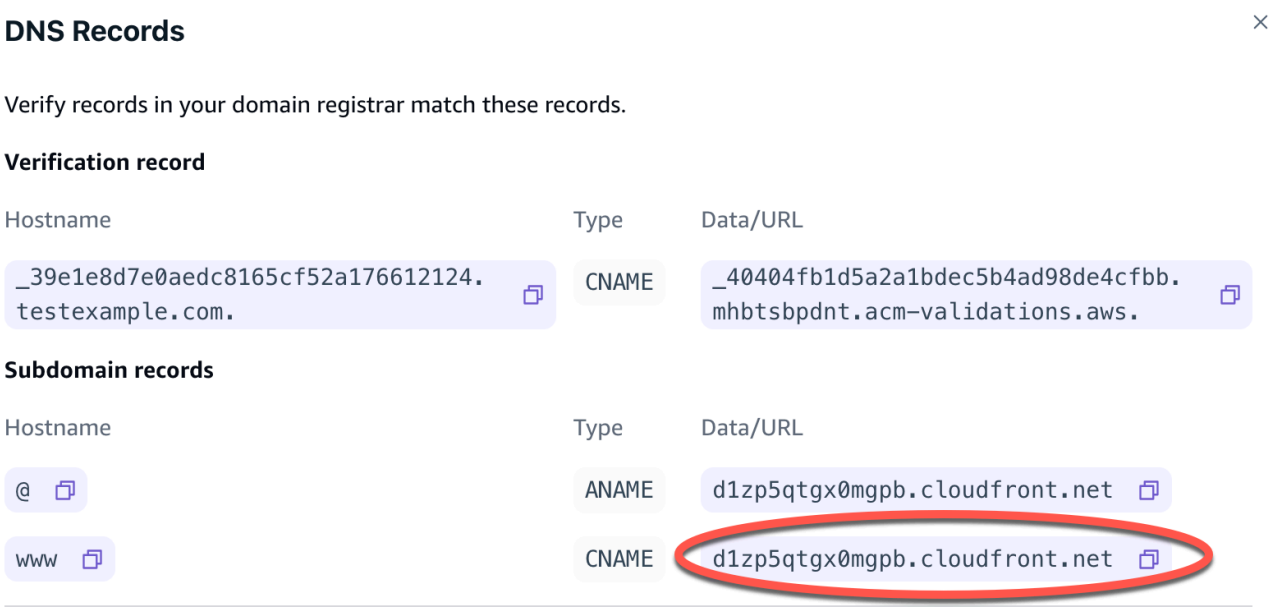
   a. 在 DNS 記錄區段中，選擇新增記錄。

   b. 針對類型，選擇 CNAME。

   c. 在名稱中，僅輸入子網域。例如，如果您的子網域是 www.example.com，請輸入 www for Name。

   d. 對於值，請查看 Amplify 主控台中的 DNS 記錄，然後輸入值。如果 Amplify 主控台將應用程式的網域顯示為 d111111abcdef8.cloudfront.net**d111111abcdef8.cloudfront.net**：//。

   下列螢幕擷取畫面顯示要使用的網域名稱記錄位置。



   e. 選擇儲存。

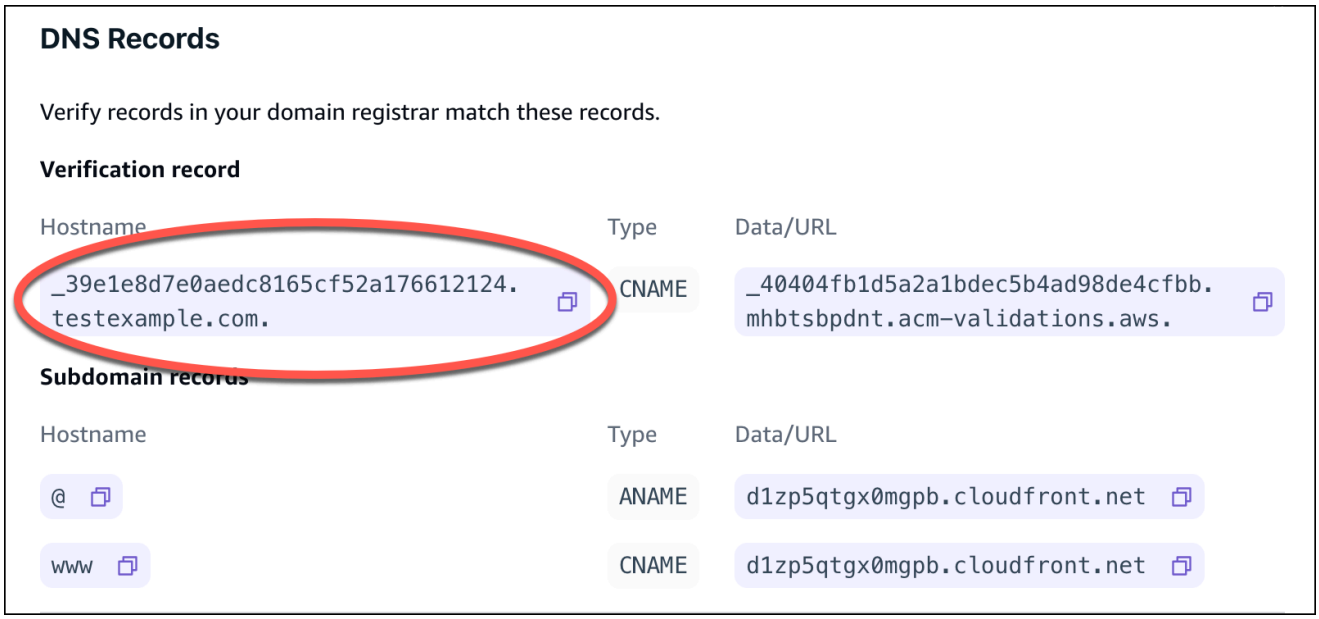
6. 建立第二個 CNAME 記錄以指向 AWS Certificate Manager (ACM) 驗證伺服器。單一經過驗證的 ACM 會為您的網域產生 SSL/TLS 憑證。

   a. 針對類型，選擇 CNAME。

   b. 在名稱中，輸入子網域。

   例如，如果 Amplify 主控台中用於驗證子網域擁有權的 DNS 記錄為 _c3e2d7eaf1e656b73f46cd6980fdc0e.example.com，則僅針對名稱輸入 。**_c3e2d7eaf1e656b73f46cd6980fdc0e**

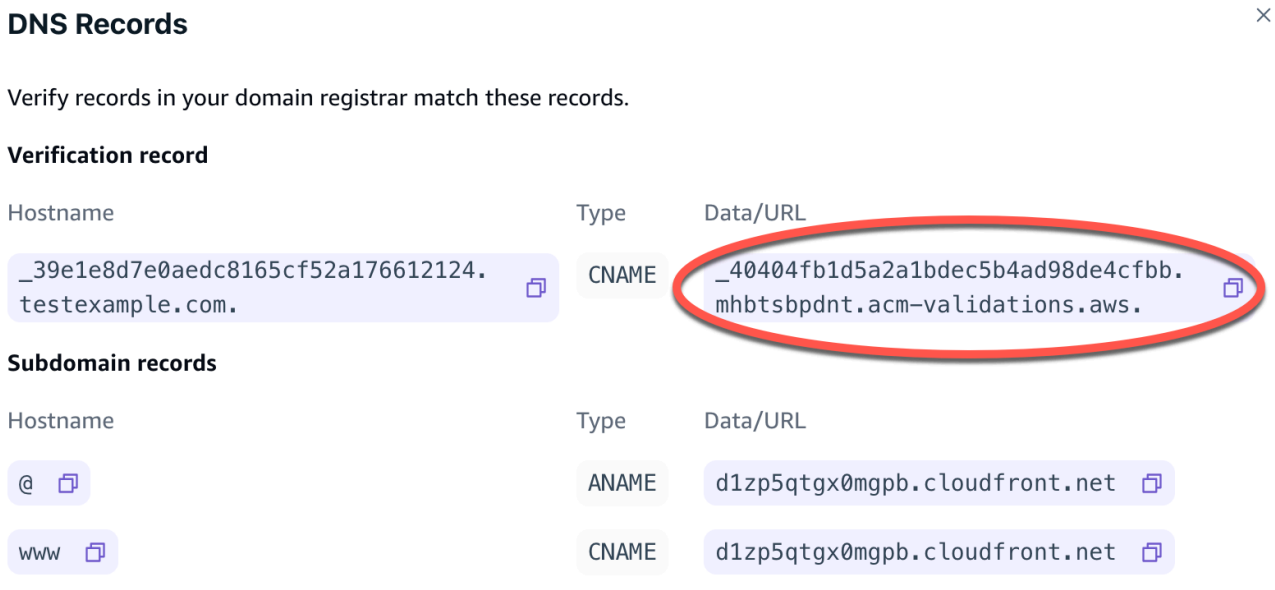
下列螢幕擷取畫面顯示要使用的驗證記錄位置。



c. 針對值，輸入 ACM 驗證憑證。

例如，如果驗證伺服器是 _cjhwou20vhu2exampleuw20vuyb2ovb9.j9s73ucn9vy.acm-validations.aws，請輸入 _cjhwou20vhu2exampleuw20vuyb2ovb9.j9s73ucn9vy.acm-validations.aws 作為值。

下列螢幕擷取畫面顯示要使用的 ACM 驗證記錄位置。



d. 選擇儲存。

> **Note**
>
> AWS Certificate Manager (ACM) 產生的預設 Amplify 憑證有效期為 13 個月，只要您的應用程式是使用 Amplify 託管，則會自動續約。如果修改或刪除 CNAME 驗證記錄，Amplify 無法續約憑證。您必須在 Amplify 主控台中刪除並再次新增網域。

7. 子網域不需要此步驟。GoDaddy 不支援 ANAME/ALIAS 記錄。對於不具有 ANAME/ALIAS 支援的 DNS 提供者，我們強烈建議將您的 DNS 遷移到 Amazon Route 53。如需詳細資訊，請參閱將 Amazon Route 53 設定為 DNS 服務。

   如果您想要將 GoDaddy 保留為提供者並更新根網域，請新增轉送並設定網域：

   a. 在 DNS 頁面上，找到頁面頂端的選單，然後選擇轉送。
   b. 在網域區段中，選擇新增轉送。
   c. 選擇 http：//，然後輸入要轉送至目的地 URL 的子網域名稱（例如 www.example.com)。
   d. 針對轉送類型，選擇暫時 (302)。
   e. 選擇、儲存。

更新網域的 SSL/TLS 憑證

您可以隨時變更網域正在使用的 SSL/TLS 憑證。例如，您可以從使用受管憑證變更為使用自訂憑證。如果您想要管理憑證及其過期通知，這會很有幫助。您也可以變更網域正在使用的自訂憑證。變更 SSL 憑證不會對作用中的網域造成任何停機時間。如需憑證的詳細資訊，請參閱使用 SSL/TLS 憑證。

使用下列程序來更新憑證類型或網域正在使用的自訂憑證。

更新網域的憑證

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇您要更新的應用程式。
3. 在導覽窗格中，選擇託管、自訂網域。
4. 在自訂網域頁面上，選擇網域組態。
5. 在網域的詳細資訊頁面上，找到自訂 SSL 憑證區段。更新憑證的程序會根據您要進行的變更類型而有所不同。
   - 從自訂憑證變更為預設 Amplify 受管憑證

    - 選擇 Amplify 受管憑證。
- 從受管憑證變更為自訂憑證

    a. 選擇自訂 SSL 憑證。
    b. 從清單中選擇要使用的憑證。
- 將自訂憑證變更為不同的自訂憑證

    - 對於自訂 SSL 憑證，請從清單中選擇要使用的新憑證。

6. 選擇儲存。網域的狀態詳細資訊會指出 Amplify 已啟動受管憑證的 SSL 建立程序，或自訂憑證的組態程序。

管理子網域

子網域是 URL 的一部分，會出現在您的網域名稱之前。例如，www 是 www.amazon.com 的子網域，aws 是 aws.amazon.com 的子網域。如果您已經有生產網站，您可能只想要連接子網域。子網域也可以是多層級，例如 beta.alpha.example.com 具有多層級子網域 beta.alpha。

僅新增子網域

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇您要新增子網域的應用程式。
3. 在導覽窗格中，選擇託管，然後選擇自訂網域。
4. 在自訂網域頁面上，選擇新增網域。
5. 輸入根網域的名稱，然後選擇設定網域。例如，如果您的網域名稱是 https://example.com，請輸入 example.com。
6. 選擇排除根目錄並修改子網域的名稱。例如，如果網域是 example.com：//。
7. 選擇新增網域。

新增多層級子網域

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇您要新增多層級子網域的應用程式。
3. 在導覽窗格中，選擇託管，然後選擇自訂網域。
4. 在自訂網域頁面上，選擇新增網域。

5. 輸入具有子網域的網域名稱，選擇排除根，然後修改子網域以新增關卡。

   例如，如果您有一個名為 alpha.example.com 的網域，並且想要建立多層級子網域 beta.alpha.example.com，則輸入 beta 做為子網域值。

6. 選擇新增網域。

新增或編輯子網域

將自訂網域新增至應用程式後，您可以編輯現有的子網域或新增新的子網域。

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇您要管理子網域的應用程式。
3. 在導覽窗格中，選擇託管，然後選擇自訂網域。
4. 在自訂網域頁面上，選擇網域組態。
5. 在子網域區段中，您可以視需要編輯現有的子網域。
6. （選用）若要新增新的子網域，請選擇新增。
7. 選擇儲存。

設定萬用字元子網域

Amplify 託管現在支援萬用字元子網域。萬用字元子網域是全擷取子網域，可讓您將現有和不存在的子網域指向應用程式的特定分支。當您使用萬用字元將應用程式中的所有子網域與特定分支建立關聯時，您可以將相同的內容提供給任何子網域中的應用程式使用者，並避免個別設定每個子網域。

若要建立萬用字元子網域，請指定星號 (*) 做為子網域名稱。例如，如果您`*.example.com`為應用程式的特定分支指定萬用字元子網域，則以 example.com 結尾的任何 URL 都會路由到該分支。在這種情況下， `dev.example.com`和 的請求`prod.example.com`將路由到`*.example.com`子網域。

請注意，Amplify 僅支援自訂網域的萬用字元子網域。您無法將此功能與預設`amplifyapp.com`網域搭配使用。

下列需求適用於萬用字元子網域：

- 子網域名稱只能以星號 (*) 指定。
- 您無法使用萬用字元取代子網域名稱的一部分，如下所示：*domain.example.com。
- 您無法取代網域名稱中間的子網域，如下所示：subdomain.*.example.com。

- 根據預設，所有 Amplify 佈建的憑證都會涵蓋自訂網域的所有子網域。

新增或刪除萬用字元子網域

將自訂網域新增至應用程式之後，您可以為應用程式分支新增萬用字元子網域。

1. 登入 AWS 管理主控台 並開啟 Amplify 託管主控台。
2. 選擇您要管理萬用字元子網域的應用程式。
3. 在導覽窗格中，選擇託管，然後選擇自訂網域。
4. 在自訂網域頁面上，選擇網域組態。
5. 在子網域區段中，您可以新增或刪除萬用字元子網域。
   - 新增萬用字元子網域
     a. 選擇 Add new (新增)。
     b. 針對子網域，輸入 *。
     c. 針對您的應用程式分支，從清單中選擇分支名稱。
     d. 選擇儲存。
   - 刪除萬用字元子網域
     a. 選擇子網域名稱旁的移除。未明確設定之子網域的流量會停止，Amplify Hosting 會將 404 狀態碼傳回這些請求。
     b. 選擇儲存。

設定 Amazon Route 53 自訂網域的自動子網域

應用程式連線至 Route 53 中的自訂網域後，Amplify 可讓您自動為新連線的分支建立子網域。例如，如果您連接開發分支，Amplify 可以自動建立 dev.exampledomain.com。當您刪除分支時，任何相關聯的子網域都會自動刪除。

設定新連線分支的自動子網域建立

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇連線至 Route 53 中受管自訂網域的應用程式。
3. 在導覽窗格中，選擇託管，然後選擇自訂網域。
4. 在自訂網域頁面上，選擇網域組態。

5. 在自動子網域建立區段中，開啟 功能。

> **Note**
>
> 此功能僅適用於根網域，例如 exampledomain.com。如果您的網域已經是子網域，例如 dev.exampledomain.com，Amplify 主控台不會顯示此核取方塊。

具有子網域的 Web 預覽

使用上述指示啟用自動建立子網域後，您應用程式的提取請求 Web 預覽也會透過自動建立的子網域存取。關閉提取請求時，會自動刪除相關聯的分支和子網域。如需設定提取請求的 Web 預覽的詳細資訊，請參閱 提取請求的 Web 預覽。

對自訂網域進行故障診斷

如果您在將自訂網域新增至 主控台中的 AWS Amplify 應用程式時遇到問題，請參閱 Amplify 故障診斷章節對自訂網域進行故障診斷中的 。如果您在該處找不到問題的解決方案，請聯絡 支援。如需詳細資訊，請參閱 AWS 支援 使用者指南中的建立支援案例。

Amplify 託管網站的防火牆支援

Amplify 託管網站的防火牆支援可讓您透過直接整合來保護 Web 應用程式 AWS WAF。 AWS WAF 可讓您設定一組規則，稱為 Web 存取控制清單 (Web ACL)，可根據您定義的可自訂 Web 安全規則和條件來允許、封鎖或監控（計數）Web 請求。當您將 Amplify 應用程式與 整合時 AWS WAF，您可以更好地控制和查看應用程式接受的 HTTP 流量。若要進一步了解 AWS WAF，請參閱《AWS WAF 開發人員指南》中的 AWS WAF 運作方式。

Amplify Hosting 運作的所有 AWS 區域 都提供防火牆支援。此整合屬於 AWS WAF 全域資源，類似於 CloudFront。Web ACLs可以連接到多個 Amplify 託管應用程式，但必須位於相同的區域。

您可以使用 AWS WAF 來保護 Amplify 應用程式免受常見的 Web 入侵，例如 SQL Injection 和跨網站指令碼。這些可能會影響應用程式的可用性和效能、危及安全性或耗用過多的資源。例如，您可以建立規則，以允許或封鎖來自指定 IP 地址範圍的請求、來自 CIDR 區塊的請求、來自特定國家或地區的請求，或包含非預期 SQL 程式碼或指令碼的請求。

您也可以建立規則，以符合在 HTTP 標頭、方法、查詢字串、URI 和請求本文 (限於前 8 KB) 的指定字串或常規表達式模式。此外，您可以建立規則來封鎖來自特定使用者代理程式、機器人和內容抓取器的事件。例如，您可以使用以速率為基礎的規則，以指定每個用戶端 IP 在尾隨、持續更新的 5 分鐘期間，允許的 Web 請求數。

若要進一步了解支援的規則類型和其他 AWS WAF 功能，請參閱 AWS WAF 開發人員指南和 AWS WAF API 參考。

> **Important**
>
> 安全是 AWS 與您之間的共同責任。 AWS WAF 不是所有網際網路安全問題的解決方案，您必須將其設定為滿足您的安全和合規目標。為了協助您了解如何在使用 時套用共同責任模型 AWS WAF，請參閱您使用 AWS WAF 服務時的安全性。

主題

- 在 中 AWS WAF 為 Amplify 應用程式啟用 AWS 管理主控台
- 取消 Web ACL 與 Amplify 應用程式的關聯
- 使用 AWS WAF 啟用 Amplify 應用程式 AWS CDK
- Amplify 如何與 整合 AWS WAF

- Amplify 應用程式的防火牆定價

在 中 AWS WAF 為 Amplify 應用程式啟用 AWS 管理主控台

您可以在 Amplify 主控台或 AWS WAF 主控台中啟用 Amplify 應用程式的 AWS WAF 保護。

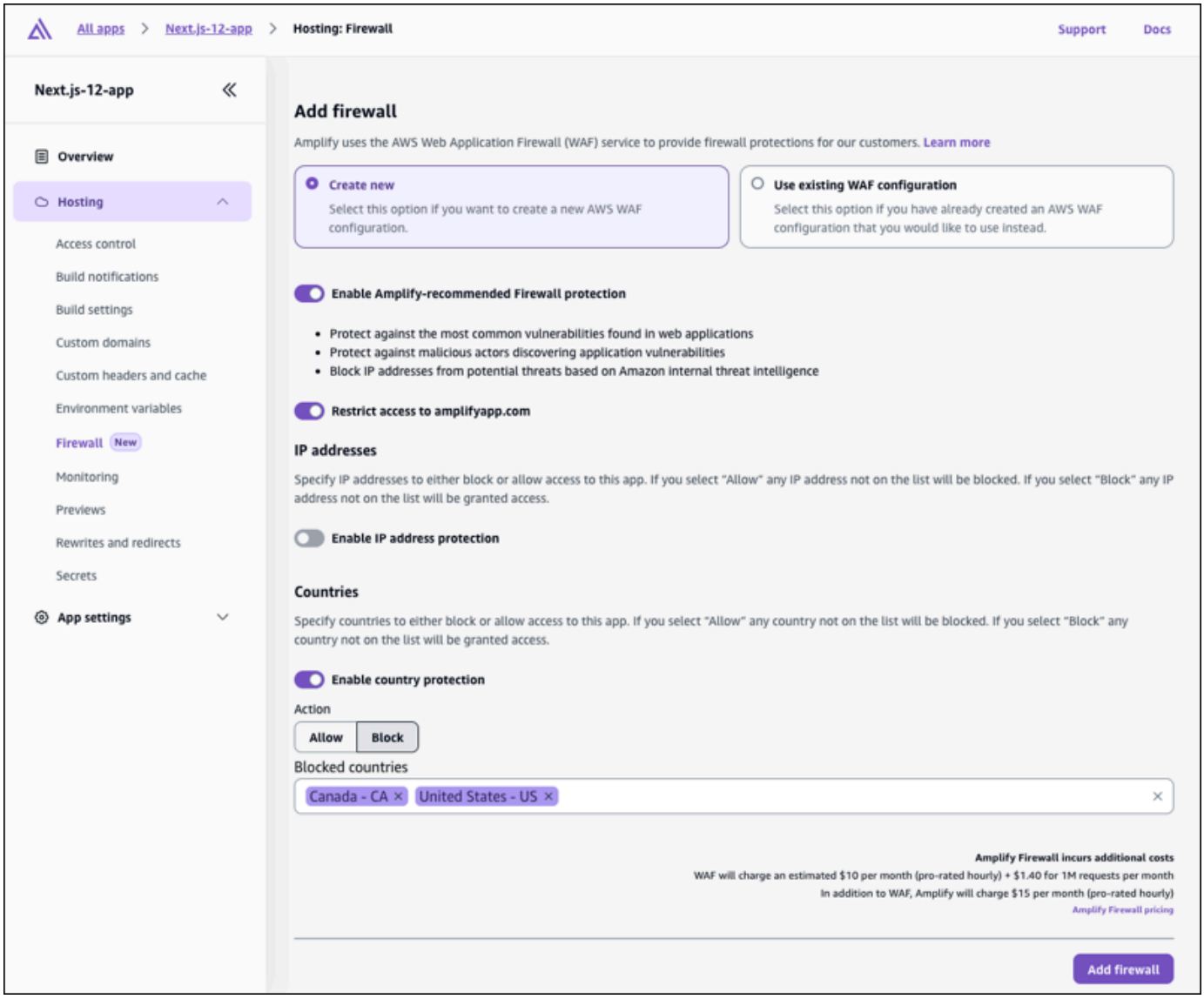
- Amplify 主控台 — 您可以透過將 AWS WAF Web ACL 與 Amplify 主控台中的應用程式建立關聯，為現有 Amplify 應用程式啟用防火牆功能。使用一鍵式保護來建立 Web ACL，其中包含我們視為大多數應用程式最佳實務的預先設定規則。您可以選擇依 IP 地址和國家/地區自訂存取權。本節中的指示說明設定一鍵式保護。
- AWS WAF 主控台 — 使用您在 AWS WAF 主控台中或使用 AWS WAF APIs 建立的預先設定 Web ACL。您必須建立要與全球 (CloudFront) 區域中 Amplify 應用程式建立關聯的 Web ACLs。區域性 Web ACLs 可能已存在於您的 中 AWS 帳戶，但與 Amplify 不相容。如需入門說明，請參閱《 AWS WAF 開發人員指南》中的設定 AWS WAF 及其元件。

使用下列程序，在 Amplify 主控台中 AWS WAF 為現有應用程式啟用 。

AWS WAF 為現有的 Amplify 應用程式啟用

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/amplify/ 開啟 Amplify 主控台。
2. 在所有應用程式頁面上，選擇要啟用防火牆功能之已部署應用程式的名稱。
3. 在導覽窗格中，選擇託管，然後選擇防火牆。

   下列螢幕擷取畫面顯示如何導覽至 Amplify 主控台中的新增防火牆頁面。



4. 在新增防火牆頁面上，您的動作將取決於您是否要建立新的組態或使用現有的 AWS WAF 組態。

   - 建立新的 AWS WAF 組態。

     a. 選擇建立新項目。

     b. 或者，啟用下列任何組態：

        i. 開啟啟用 Amplify 建議的防火牆保護。

        ii. 開啟限制對 amplifyapp.com 的存取，以防止存取預設 Amplify 網域上的應用程式。

        iii. 對於 IP 地址，開啟啟用 IP 地址保護。

A. 對於動作，如果您想要指定具有存取權的 IP 地址，且所有其他地址都會遭到封鎖，請選擇允許。如果您想要指定要封鎖的 IP 地址，且所有其他地址都可以存取，請選擇封鎖。

B. 針對 IP 版本，選取 IPV4 或 IPV6。

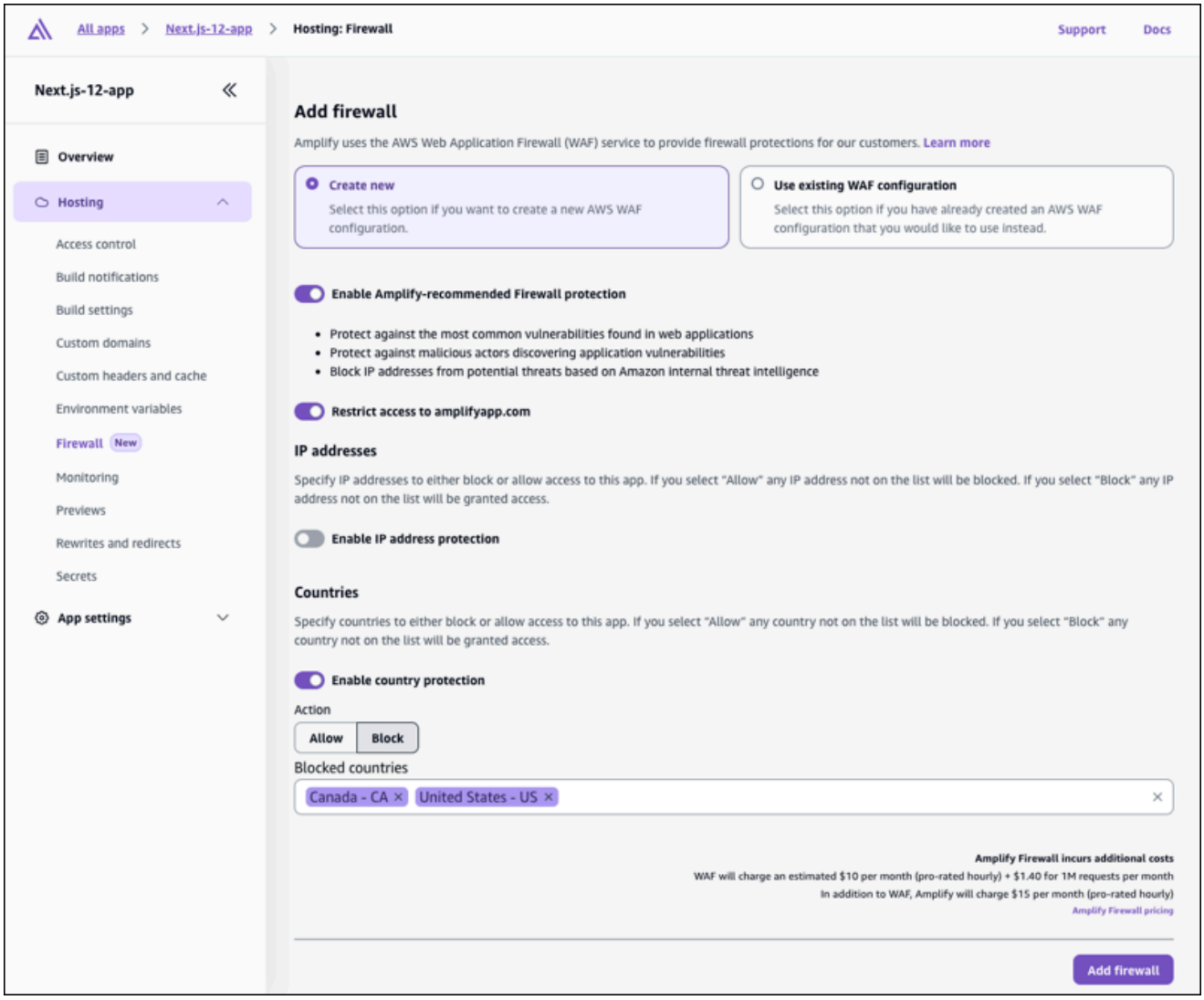
C. 在 IP 地址文字方塊中，以 CIDR 格式輸入允許或封鎖的 IP 地址，每行一個。

iv. 對於國家/地區，開啟啟用國家/地區保護。

A. 針對動作，如果您想要指定將可存取的國家/地區，且所有其他國家/地區將遭到封鎖，請選擇允許。如果您想要指定將被封鎖的國家/地區，且所有其他國家/地區都有存取權，請選擇封鎖。

B. 對於國家/地區，請從清單中選取允許或封鎖的國家/地區。

下列螢幕擷取畫面示範如何啟用應用程式的新 AWS WAF 組態。



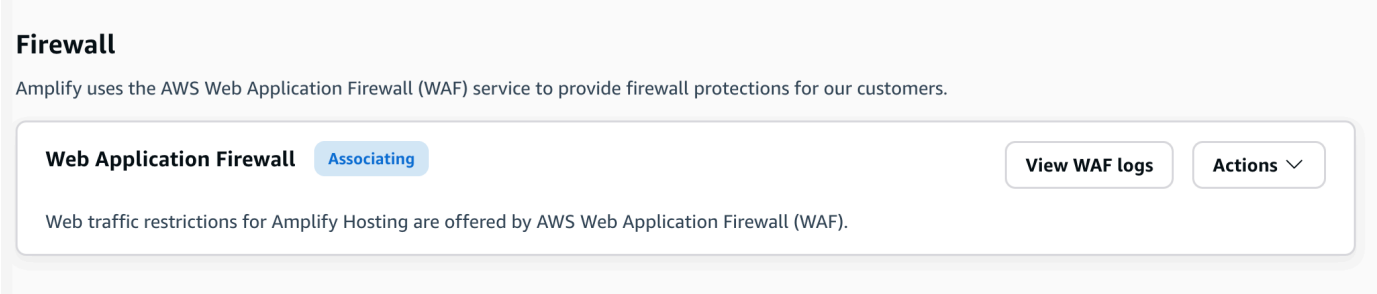
- 使用現有的 AWS WAF 組態。

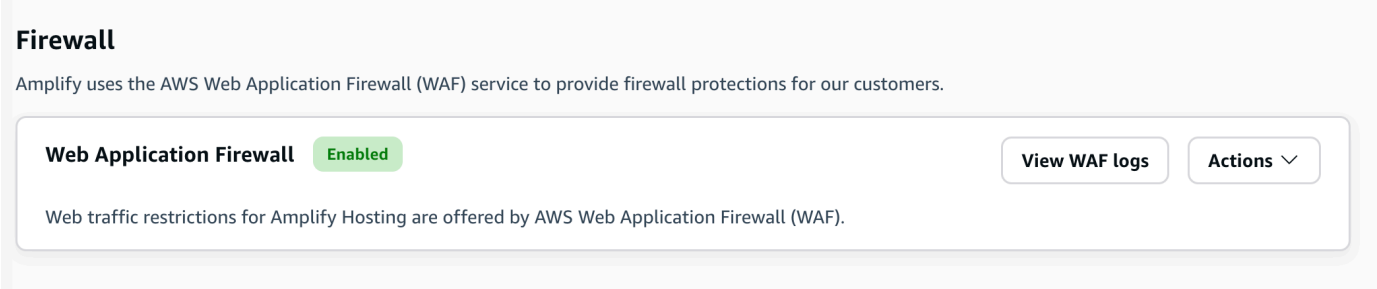
  a. 選擇使用現有 AWS WAF 組態。

  b. 從 AWS WAF 中的 Web ACLs 清單中選取已儲存的組態 AWS 帳戶。與 Amplify 應用程式建立關聯的 Web ACL 必須在全域 (CloudFront) 區域中建立。區域性 Web ACLs 可能已存在於您的 中 AWS 帳戶，但與 Amplify 不相容。

5. 選擇新增防火牆。
6. 在防火牆頁面上，會顯示關聯狀態，指出正在傳播 AWS WAF 設定。程序完成時，狀態會變更為已啟用。

   下列螢幕擷取畫面顯示 Amplify 主控台中的防火牆進度狀態，指出 AWS WAF 組態何時關聯且已啟用。





取消 Web ACL 與 Amplify 應用程式的關聯

您無法刪除與 Amplify 應用程式相關聯的 Web ACL。您必須先在 Amplify 主控台中取消 Web ACL 與應用程式的關聯。然後，您可以在 AWS WAF 主控台中刪除它。

取消 Web ACL 與 Amplify 應用程式的關聯

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/amplify/ 開啟 Amplify 主控台。
2. 在所有應用程式頁面上，選擇要與 Web ACL 取消關聯的應用程式名稱。
3. 在導覽窗格中，選擇託管，然後選擇防火牆。
4. 在防火牆頁面上，選擇動作，然後選擇取消防火牆的關聯。
5. 在確認模式中，輸入 **`disassociate`**，然後選擇取消關聯防火牆。
6. 在防火牆頁面上，會顯示取消關聯狀態，以指出正在傳播 AWS WAF 設定。

   程序完成時，您可以在 AWS WAF 主控台中刪除 Web ACL。

使用 AWS WAF 啟用 Amplify 應用程式 AWS CDK

您可以使用 AWS Cloud Development Kit (AWS CDK) 為 Amplify 應用程式啟用 AWS WAF 。若要進一步了解如何使用 CDK，請參閱《 AWS Cloud Development Kit (AWS CDK) 開發人員指南》中的什麼是 CDK？。

下列 TypeScript 程式碼範例示範如何使用兩個 CDK 堆疊建立 AWS CDK 應用程式：一個用於 Amplify，另一個用於 Amplify AWS WAF。請注意， AWS WAF 堆疊必須部署到美國東部（維吉尼亞北部）(us-east-1) 區域。Amplify 應用程式堆疊可以部署到不同的區域。您必須建立要與全球 (CloudFront) 區域中 Amplify 應用程式建立關聯的 Web ACL。區域性 Web ACLs 可能已存在於您的 中 AWS 帳戶，但與 Amplify 不相容。

```
import * as cdk from "aws-cdk-lib";
import { Construct } from "constructs";
import * as wafv2 from "aws-cdk-lib/aws-wafv2";
import * as amplify from "aws-cdk-lib/aws-amplify";

interface WafStackProps extends cdk.StackProps {
  appArn: string;
}

export class AmplifyStack extends cdk.Stack {
  public readonly appArn: string;
  constructor(scope: Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);
    const amplifyApp = new amplify.CfnApp(this, "AmplifyApp", {
      name: "MyApp",
    });
    this.appArn = amplifyApp.attrArn;
  }
}

export class WAFStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props: WafStackProps) {
    super(scope, id, props);
    const webAcl = new wafv2.CfnWebACL(this, "WebACL", {
      defaultAction: { allow: {} },
      scope: "CLOUDFRONT",
      rules: [
        // Add your own rules here.
      ],
      visibilityConfig: {
        cloudWatchMetricsEnabled: true,
        metricName: "my-metric-name",
        sampledRequestsEnabled: true,
      },
    });

    new wafv2.CfnWebACLAssociation(this, "WebACLAssociation", {
```

```
      resourceArn: props.appArn,
      webAclArn: webAcl.attrArn,
    });
  }
}

const app = new cdk.App();

// Create AmplifyStack in your desired Region.
const amplifyStack = new AmplifyStack(app, 'AmplifyStack', {
  env: { region: 'us-west-2' },
});

// Create WAFStack in IAD region, passing appArn from AmplifyStack.
new WAFStack(app, 'WAFStack', {
  env: { region: 'us-east-1' },
  crossRegionReferences: true,

  appArn: amplifyStack.appArn,  // Pass appArn from AmplifyStack.
});
```

Amplify 如何與 整合 AWS WAF

以下清單提供防火牆支援如何與 整合的特定詳細資訊， AWS WAF 以及建立 Web ACLs 並將其與 Amplify 應用程式建立關聯時應考慮的限制。

- 您可以 AWS WAF 為任何類型的 Amplify 應用程式啟用 。這包括任何支援的架構、伺服器端轉譯 (SSR) 應用程式，以及完全靜態的網站。Amplify Gen 1 和 Gen 2 應用程式 AWS WAF 支援 。
- 您必須建立要與全球 (CloudFront) 區域中的 Amplify 應用程式建立關聯的 Web ACLs。區域性 Web ACLs 可能已存在於您的 中 AWS 帳戶，但與 Amplify 不相容。
- Web ACL 和 Amplify 應用程式必須在相同的 中建立 AWS 帳戶。您可以使用 AWS Firewall Manager 跨 複寫 AWS WAF 規則 AWS 帳戶，以簡化將組織規則集中並跨多個規則分佈 AWS 帳戶。如需詳細資訊，請參閱《 AWS WAF 開發人員指南》中的 AWS Firewall Manager。
- 您可以在相同 中的多個 Amplify 應用程式之間共用相同的 Web ACL AWS 帳戶。所有應用程式都必須位於相同的 區域。
- 當您將 Web ACL 與 Amplify 應用程式建立關聯時，Web ACL 預設會連接到應用程式中的每個分支。當您建立新的分支時，它們將擁有 Web ACL。
- 當您將 Web ACL 與 Amplify 應用程式建立關聯時，它會自動與應用程式的所有網域建立關聯。不過，您可以使用 Host-header 比對規則來設定套用至單一網域名稱的規則。

- 您無法刪除與 Amplify 應用程式相關聯的 Web ACL。在 AWS WAF 主控台中刪除 Web ACL 之前，您需要將其與應用程式取消關聯。

Amplify Web ACL 資源政策

若要允許 Amplify 存取您的 Web ACL，資源政策會在關聯期間連接到 Web ACL。Amplify 會自動建構此資源政策，但您可以使用 AWS WAFV2 GetPermissionPolicy API 檢視它。將 Web ACL 與 Amplify 應用程式建立關聯時，需要下列 IAM 許可。

- amplify：AssociateWebACL
- wafv2:AssociateWebACL
- wafv2:PutPermissionPolicy
- wafv2:GetPermissionPolicy

Amplify 應用程式的防火牆定價

在 Amplify 應用程式 AWS WAF 實作的成本是根據下列兩個元件計算：

- AWS WAF 用量 – 您將需要支付 AWS WAF 與 AWS WAF 定價模型相關的用量費用。 AWS WAF 費用是根據您建立的 Web 存取控制清單 (Web ACLs)、每個 Web ACL 新增的規則數目，以及您收到的 Web 請求數目。如需定價詳細資訊，請參閱 AWS WAF 定價。
- Amplify 託管整合成本 – 當您將 Web ACL 連接到 Amplify 應用程式時，每個應用程式每月需支付 15.00 USD 的費用。這是每小時按比例分配。

功能分支部署和團隊工作流程

Amplify 託管旨在使用功能分支和 GitFlow 工作流程。每次您連接儲存庫中的新分支時，Amplify 都會使用 Git 分支建立新的部署。連接第一個分支之後，您可以建立其他功能分支。

將分支新增至應用程式

1. 選擇您要新增分支的應用程式。
2. 選擇應用程式設定，然後選擇分支設定。
3. 在分支設定頁面上，選擇新增分支。
4. 從儲存庫中選取分支。
5. 選擇新增分支。
6. 重新部署您的應用程式。

新增分支之後，您的應用程式有兩個可在 Amplify 預設網域使用的部署，例如 https://main.appid.amplifyapp.com 和 https://dev.appid.amplifyapp.com。這可能因team-to-team，但通常主要分支會追蹤發行程式碼，並且是您的生產分支。開發分支則可做為測試新功能的整合分支來使用。這可讓 Beta 版測試人員在開發分支部署上測試未發行的功能，而不會影響主分支部署上的任何生產最終使用者。

主題

- 具有完整堆疊 Amplify Gen 2 應用程式的團隊工作流程
- 具有完整堆疊 Amplify Gen 1 應用程式的團隊工作流程
- 模式型功能分支部署
- Amplify 組態的自動建置時間產生（僅限第 1 代應用程式）
- 條件式後端建置（僅限第 1 代應用程式）
- 跨應用程式使用 Amplify 後端（僅限第 1 代應用程式）

具有完整堆疊 Amplify Gen 2 應用程式的團隊工作流程

AWS Amplify Gen 2 推出以 TypeScript 為基礎的程式碼優先開發人員體驗來定義後端。若要了解 Amplify Gen 2 應用程式的完整堆疊工作流程，請參閱 Amplify 文件中的完整堆疊工作流程。

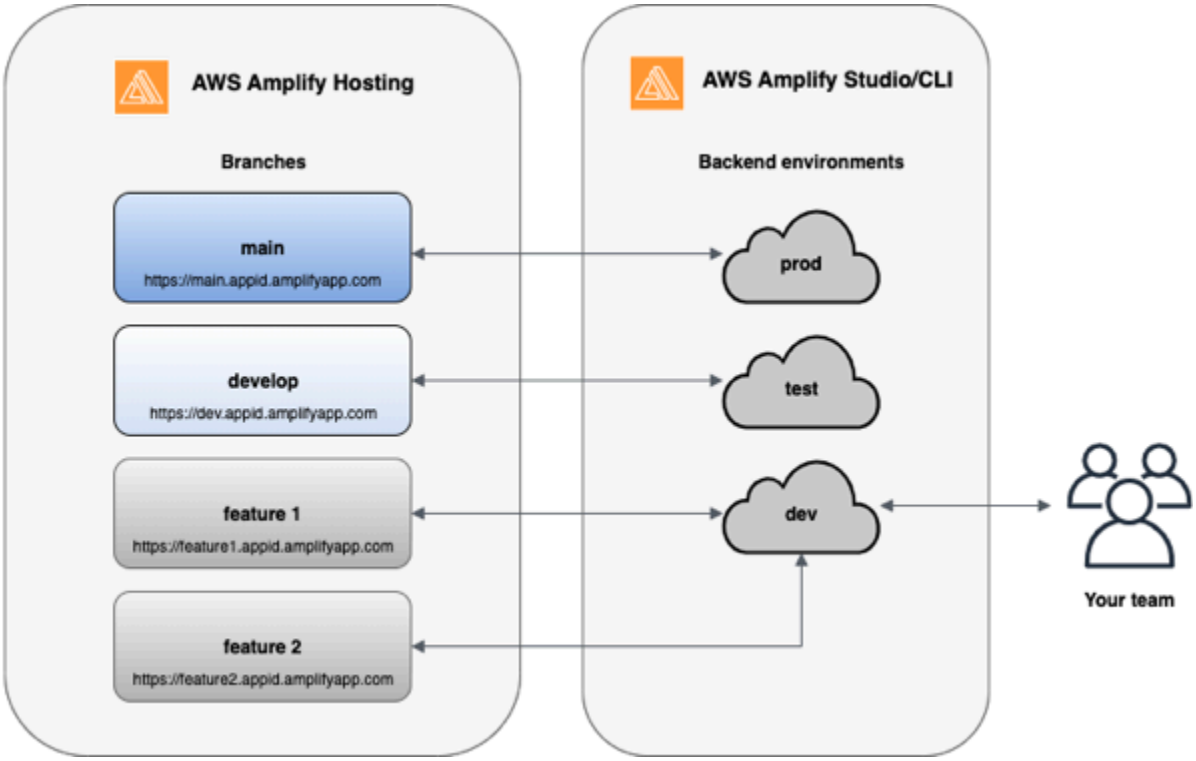
具有完整堆疊 Amplify Gen 1 應用程式的團隊工作流程

功能分支部署包含前端和選用的後端環境。前端建置並部署至全球內容交付網路 (CDN)，而後端則由 Amplify Studio 或 Amplify CLI 部署至 AWS。若要了解如何設定此部署案例，請參閱 建置應用程式的後端。

Amplify Hosting 會使用功能分支部署持續部署後端資源，例如 GraphQL APIs 和 Lambda 函數。您可以使用下列分支模型，透過 Amplify Hosting 部署後端和前端。

功能分支工作流程

- 使用 Amplify Studio 或 Amplify CLI 建立生產、測試和開發後端環境。
- 將 prod 後端映射至主分支。
- 將測試後端映射至開發分支。
- 團隊成員可以使用開發後端環境來測試個別功能分支。



1. 安裝 Amplify CLI 以初始化新的 Amplify 專案。

```
npm install -g @aws-amplify/cli
```

2. 為專案初始化「prod」後端環境。如果您沒有專案，請使用 create-react-app 或 Gatsby 等引導工具建立專案。

```
create-react-app next-unicorn
cd next-unicorn
amplify init
 ? Do you want to use an existing environment? (Y/n): n
 ? Enter a name for the environment: prod
...
amplify push
```

3. 新增「test」和「dev」後端環境。

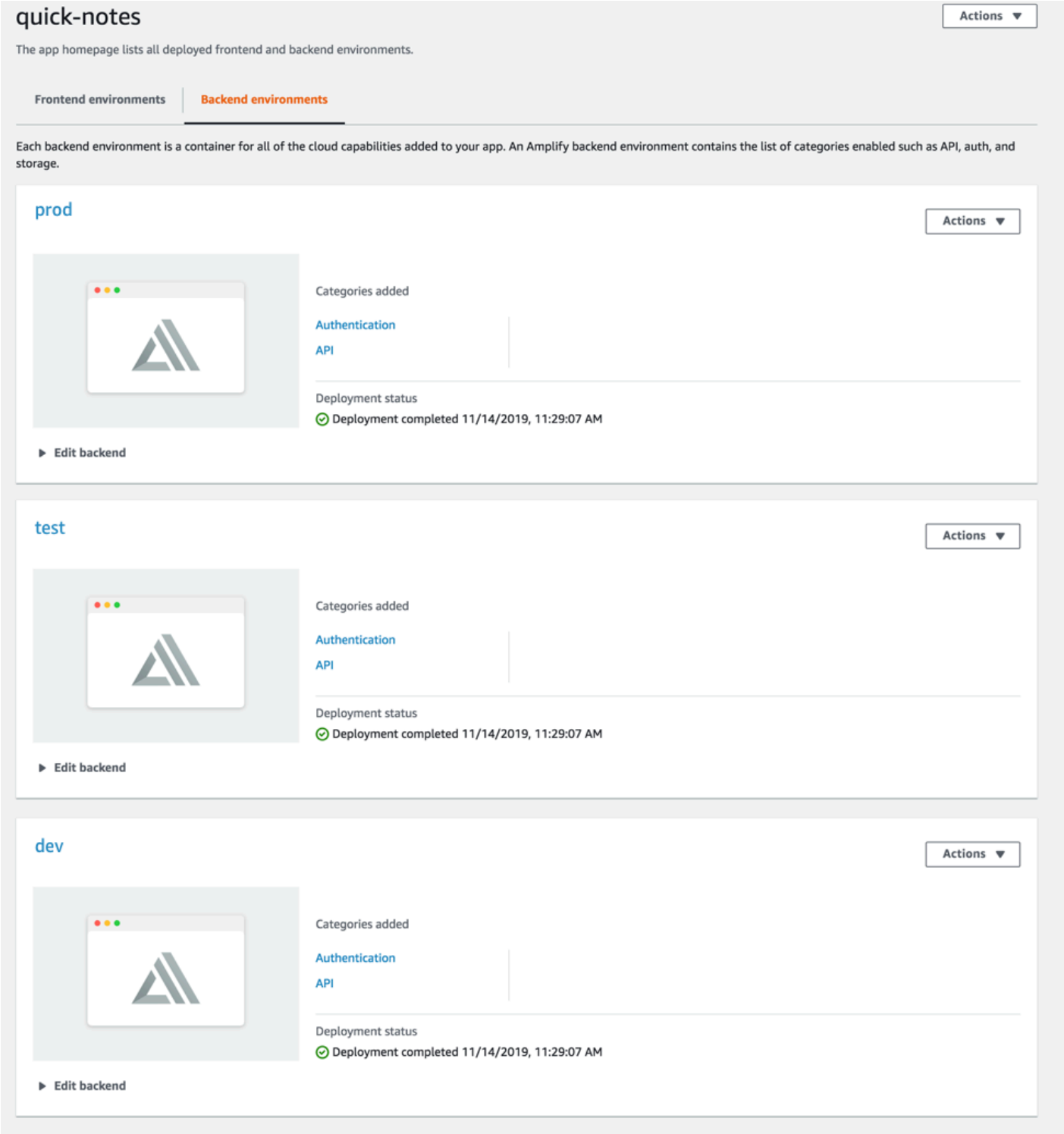
```
amplify env add
 ? Do you want to use an existing environment? (Y/n): n
 ? Enter a name for the environment: test
...
amplify push

amplify env add
 ? Do you want to use an existing environment? (Y/n): n
 ? Enter a name for the environment: dev
...
amplify push
```

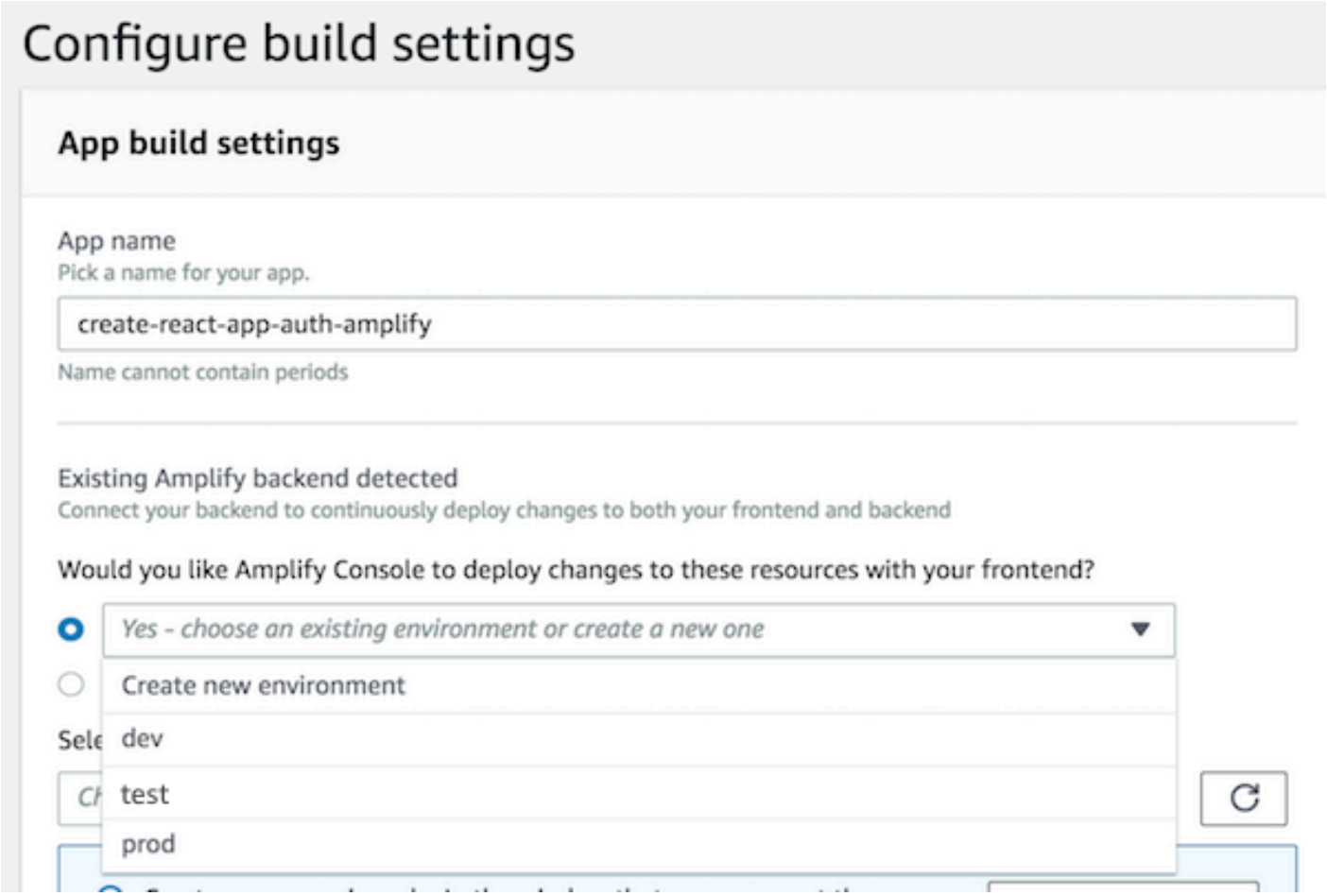
4. 將程式碼推送至您選擇的 Git 儲存庫 （在此範例中，我們將假設您已推送至主要儲存庫）。

```
git commit -am 'Added dev, test, and prod environments'
git push origin main
```

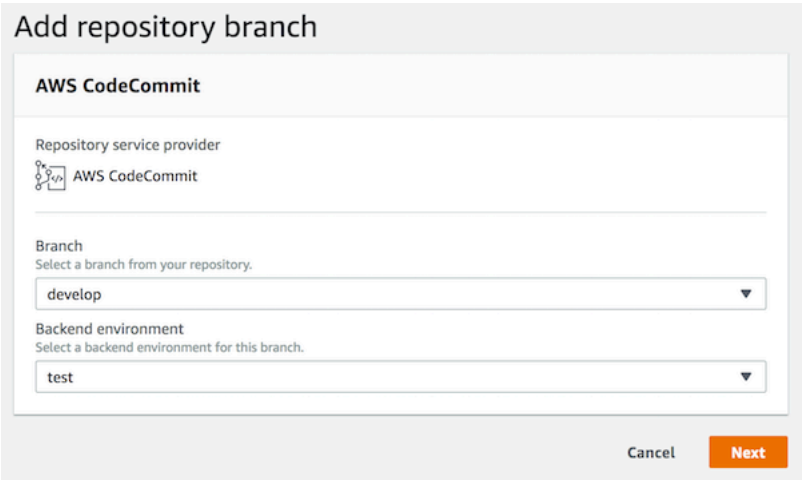
5. 請造訪 中的 Amplify AWS 管理主控台 以查看您目前的後端環境。從導覽列向上導覽關卡，以檢視在後端環境索引標籤中建立的所有後端環境清單。



6. 切換到前端環境索引標籤，並連接您的儲存庫提供者和主分支。
7. 在建置設定頁面上，選取現有的後端環境，以使用主分支設定持續部署。從清單中選擇 prod，並將服務角色授予 Amplify。選擇 Save and deploy (儲存並部署)。建置完成後，您會在 https：//https://main.appid.amplifyapp.com 取得主要分支部署。



8. 在 Amplify 中連接開發分支（假設開發和主要分支目前相同）。選擇「test」後端環境。



9. Amplify 現在已設定。您可以開始在功能分支中處理新功能。請從本機工作站使用「dev」後端環境來新增後端功能。

```
git checkout -b newinternet
amplify env checkout dev
```

```
amplify add api
...
amplify push
```

10.功能處理完畢後，請遞交程式碼並建立提取請求，以在內部進行檢閱。

```
git commit -am 'Decentralized internet v0.1'
git push origin newinternet
```

11.若要預覽變更的外觀，請前往 Amplify 主控台並連接您的功能分支。注意：如果系統 AWS CLI 已安裝（而非 Amplify CLI)，您可以直接從終端機連接分支。依序前往 App settings (應用程式設定) > General (一般) > AppARN: arn:aws:amplify:<region>:<region>:apps/<appid>，即可找到 appid。

```
aws amplify create-branch --app-id <appid> --branch-name <branchname>
aws amplify start-job --app-id <appid> --branch-name <branchname> --job-type RELEASE
```

12.您可前往 https://newinternet.appid.amplifyapp.com 存取該功能，以與團隊夥伴共享。如果看起來都沒問題，請將 PR 合併至開發分支。

```
git checkout develop
git merge newinternet
git push
```

13.這將啟動建置，以使用位於 https：//https://dev.appid.amplifyapp.com 的分支部署更新 Amplify 中的後端和前端。您可與內部合作夥伴分享此連結，讓他們檢閱新功能。

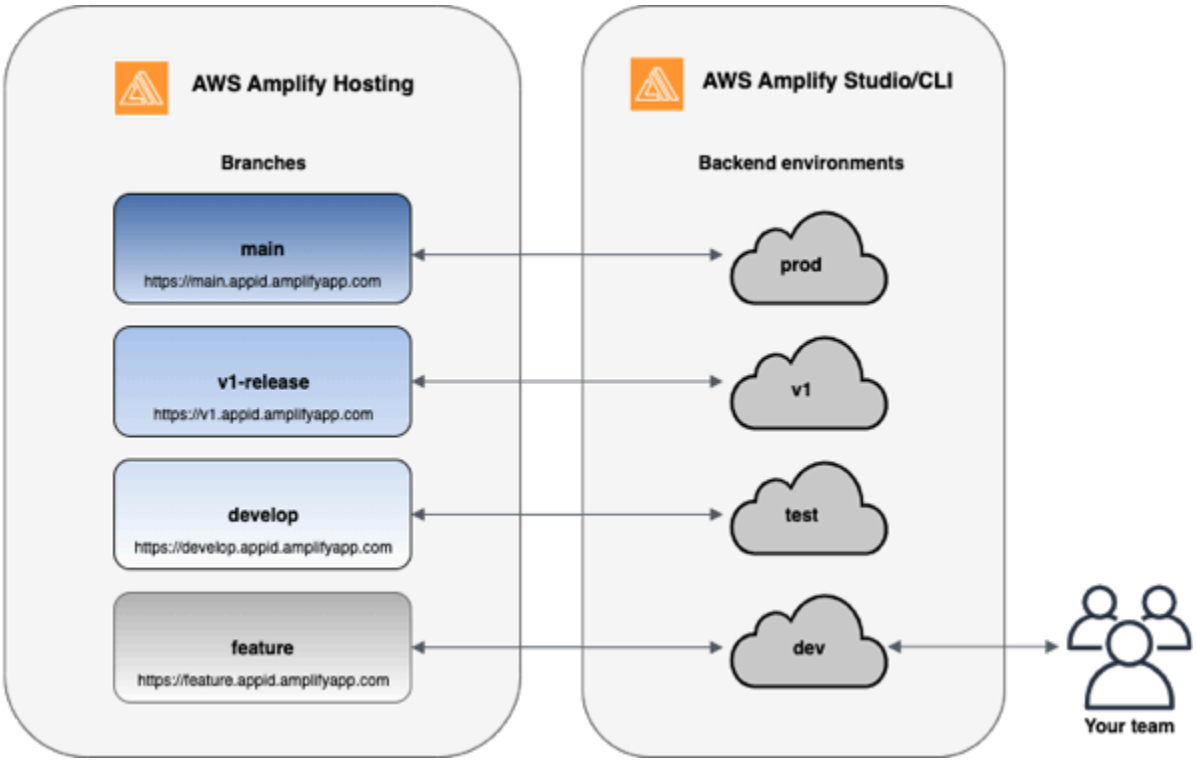
14.從 Git、Amplify 刪除您的功能分支，並從雲端移除後端環境（您可以隨時執行「amplify env checkout prod」和執行「amplify env add」來啟動新的分支）。

```
git push origin --delete newinternet
aws amplify delete-branch --app-id <appid> --branch-name <branchname>
amplify env remove dev
```

GitFlow 工作流程

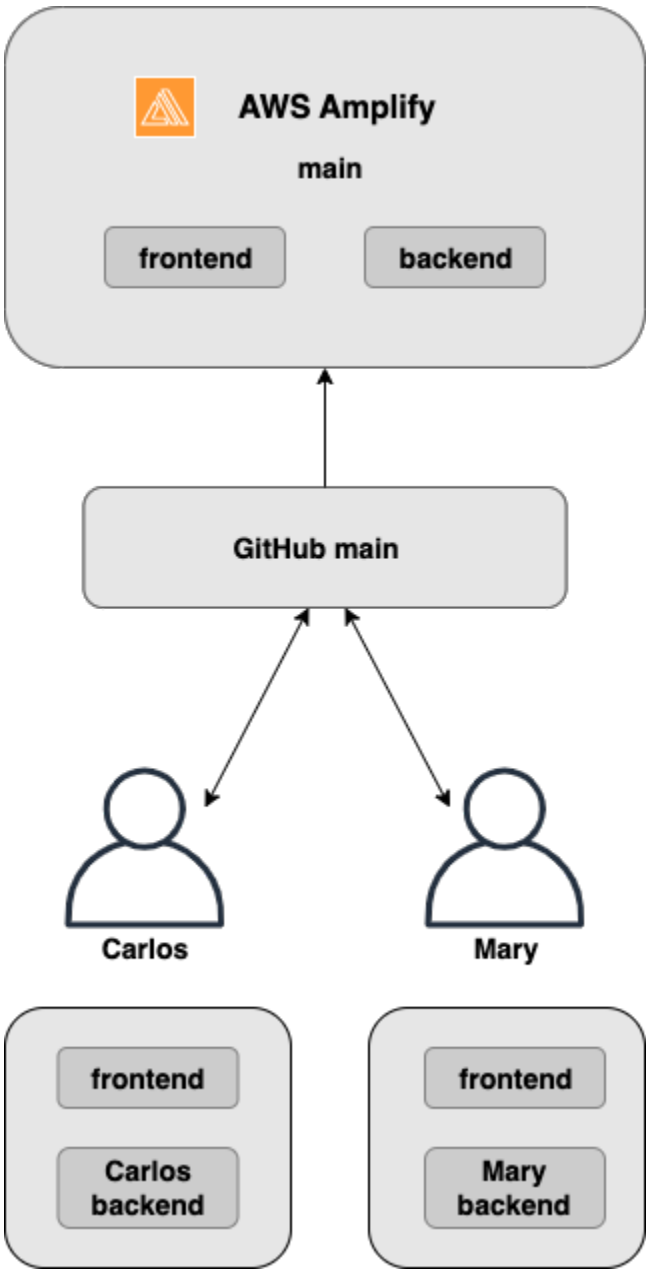
GitFlow 會使用兩個分支來記錄專案的歷史記錄。主要分支只會追蹤發行程式碼，而開發分支會用作新功能的整合分支。GitFlow 會將新開發與完成的工作分開，以簡化平行開發作業。系統會在「功能」分支中完成新的開發 (例如功能和非緊急錯誤修正)。當開發人員認為程式碼隨時可發佈時，系統就會將該「功能」分支合併回整合「開發」分支。對主分支的唯一遞交是從發行分支和 Hotfix 分支合併（以修正緊急錯誤）。

下圖顯示建議的 GitFlow 設定。您可以遵循上述功能分支工作流程一節的相同程序來操作。



每位開發人員的沙盒

- 團隊中的每位開發人員都應該在與其本機電腦分開的雲端中建立沙盒環境。這可讓開發人員彼此獨立運作，而不會覆寫其他團隊成員的變更。
- Amplify 中的每個分支都有自己的後端。這可確保 Amplify 使用 Git 儲存庫作為部署變更的單一事實來源，而不是依賴團隊的開發人員從本機電腦手動將其後端或前端推送至生產環境。



1. 安裝 Amplify CLI 以初始化新的 Amplify 專案。

```
npm install -g @aws-amplify/cli
```

2. 為您的專案初始化 mary 後端環境。如果您沒有專案，請使用 create-react-app 或 Gatsby 等引導工具建立專案。

```
cd next-unicorn
amplify init
 ? Do you want to use an existing environment? (Y/n): n
```

```
 ? Enter a name for the environment: mary
...
amplify push
```

3. 將程式碼推送至您選擇的 Git 儲存庫 （在此範例中，我們將假設您已推送至主要儲存庫。

```
git commit -am 'Added mary sandbox'
git push origin main
```

4. 將您的儲存庫 > 主要連線至 Amplify。
5. Amplify 主控台會偵測 Amplify CLI 建立的後端環境。從下拉式清單中選擇建立新環境，並將服務角色授予 Amplify。選擇 Save and deploy (儲存並部署)。建置完成後，您會在 https：//https://main.appid.amplifyapp.com 取得主要分支部署，其中包含連結至分支的新後端環境。
6. 在 Amplify 中連接開發分支 （假設開發和主要分支目前相同），然後選擇建立

模式型功能分支部署

模式型分支部署可讓您自動部署符合特定模式的分支至 Amplify。使用功能分支或 GitFlow 工作流程進行發行的產品團隊現在可以定義模式，例如**`release**`**將開頭為「發行」的 Git 分支自動部署到可共用的 URL。

1. 選擇應用程式設定，然後選擇分支設定。
2. 在分支設定頁面上，選擇編輯。
3. 選取分支自動偵測，以自動將分支連線至符合模式集的 Amplify。
4. 在分支自動偵測 - 模式方塊中，輸入自動部署分支的模式。
   - **`*`** – 部署儲存庫中的所有分支。
   - **`release*`** – 部署以「發行」一詞開頭的所有分支。
   - **`release*/`** – 部署符合「發行 /」模式的所有分支。
   - 在逗號分隔清單中指定多個模式。例如 **`release*, feature*`**。
5. 選取分支自動偵測存取控制 ，為自動建立的所有分支設定自動密碼保護。
6. 對於使用 Amplify 後端建置的第 1 代應用程式，您可以選擇為每個連線的分支建立新的環境，或將所有分支指向現有的後端。
7. 選擇儲存。

連接到自訂網域之應用程式的模式型功能分支部署

您可以針對連線至 Amazon Route 53 自訂網域的應用程式使用模式型功能分支部署。

- 如需設定模式型功能分支部署的指示，請參閱 設定 Amazon Route 53 自訂網域的自動子網域
- 如需將 Amplify 應用程式連線至 Route 53 中管理的自訂網域的說明，請參閱 新增由 Amazon Route 53 管理的自訂網域
- 如需使用 Route 53 的詳細資訊，請參閱什麼是 Amazon Route 53。

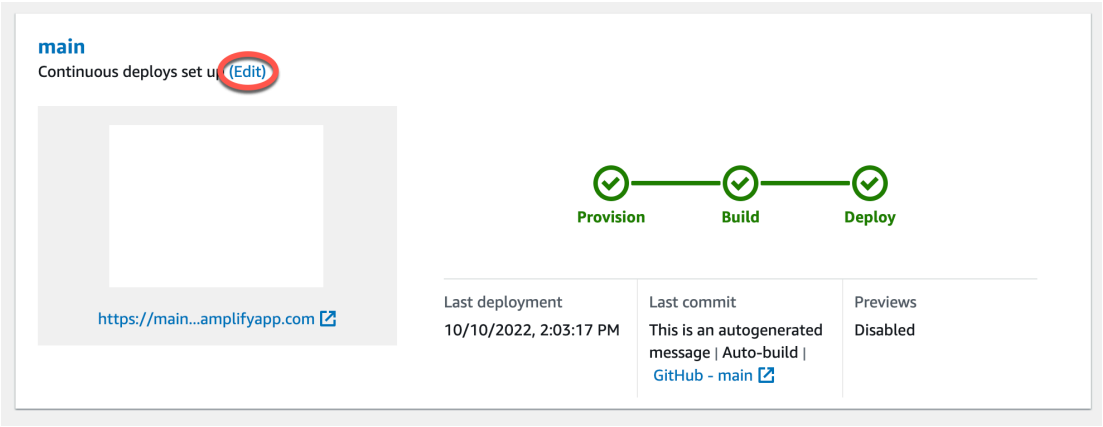
Amplify 組態的自動建置時間產生 （僅限第 1 代應用程式）

> **Note**
>
> 本節中的資訊僅適用於 Gen 1 應用程式。如果您想要從 Gen 2 應用程式的功能分支自動部署基礎設施和應用程式程式碼變更，請參閱 Amplify 文件中的 Fullstack 分支部署

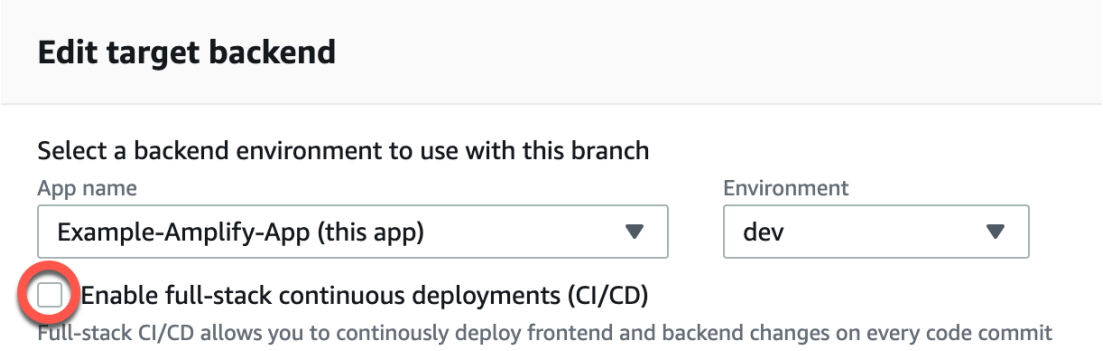
Amplify 支援產生適用於 Gen 1 應用程式的 Amplify 組態`aws-exports.js`檔案的自動建置時間。透過關閉完整堆疊 CI/CD 部署，您可以讓應用程式自動產生`aws-exports.js`檔案，並確保在建置時不會對後端進行更新。

在`aws-exports.js`建置時間自動產生

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇要編輯的應用程式。
3. 選擇託管環境索引標籤。
4. 找到要編輯的分支，然後選擇編輯。



5. 在編輯目標後端頁面上，取消勾選啟用完整堆疊連續部署 (CI/CD)，關閉此後端的完整堆疊 CI/CD。



6. 選取現有的服務角色，為 Amplify 提供變更應用程式後端所需的許可。如果您需要建立服務角色，請選擇建立新角色。如需建立服務角色的詳細資訊，請參閱新增具有部署後端資源許可的服務角色。
7. 選擇儲存。Amplify 會在您下次建置應用程式時套用這些變更。

條件式後端建置 （僅限第 1 代應用程式）

> **Note**
>
> 本節中的資訊僅適用於 Gen 1 應用程式。Amplify Gen 2 推出以 TypeScript 為基礎的程式碼優先開發人員體驗。因此，第 2 代後端不需要此功能。

Amplify 支援在 Gen 1 應用程式的所有分支上建置條件式後端。若要設定條件式後端建置，請將AMPLIFY_DIFF_BACKEND環境變數設定為 `true`。啟用條件式後端建置有助於加速僅對前端進行變更的建置。

當您啟用 diff 型後端建置時，Amplify 會在每次建置開始時嘗試在儲存庫中的`amplify`資料夾上執行 diff。如果 Amplify 找不到任何差異，它會略過後端建置步驟，而不會更新您的後端資源。如果您的專案在儲存庫中沒有 `amplify` 資料夾，Amplify 會忽略AMPLIFY_DIFF_BACKEND環境變數的值。如需設定AMPLIFY_DIFF_BACKEND環境變數的指示，請參閱 為 Gen 1 應用程式設定差異型後端建置。

如果您目前在後端階段的建置設定中指定了自訂命令，則條件式後端建置將無法運作。如果您希望這些自訂命令執行，則必須將其移至您應用程式`amplify.yml`檔案中建置設定的前端階段。如需更新`amplify.yml`檔案的詳細資訊，請參閱 組建規格參考。

跨應用程式使用 Amplify 後端 （僅限第 1 代應用程式）

> **Note**
>
> 本節中的資訊僅適用於 Gen 1 應用程式。如果您想要共用第 2 代應用程式的後端資源，請參閱 Amplify 文件中的跨分支共用資源

Amplify 可讓您在指定區域中的所有 Gen 1 應用程式之間重複使用現有的後端環境。您可以在建立新應用程式、將新分支連接至現有應用程式，或更新現有前端以指向不同的後端環境時執行此操作。

建立新應用程式時重複使用後端

在建立新的 Amplify 應用程式時重複使用後端

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 若要建立新的後端以用於此範例，請執行下列動作：
   a. 在導覽窗格中，選擇所有應用程式。
   b. 選擇新增應用程式、建置應用程式。
   c. 輸入應用程式的名稱，例如 **`Example-Amplify-App`**。
   d. 選擇確認部署。
3. 若要將前端連接到新的後端，請選擇託管環境索引標籤。
4. 選擇您的 Git 提供者，然後選擇 Connect 分支。
5. 在新增儲存庫分支頁面上，針對最近更新的儲存庫，選擇您的儲存庫名稱。針對分支，從您的儲存庫選取要連線的分支。
6. 在建置設定中，執行下列動作：
   a. 針對應用程式名稱，選取要用來新增後端環境的應用程式。您可以選擇目前的應用程式或目前區域中的任何其他應用程式。
   b. 針對環境，選取要新增的後端環境名稱。您可以使用現有環境或建立新的環境。
   c. 根據預設，全堆疊 CI/CD 會關閉。關閉完整堆疊 CI/CD 會導致應用程式以僅提取模式執行。在建置時，Amplify 只會自動產生`aws-exports.js`檔案，而不會修改您的後端環境。
   d. 選取現有的服務角色，為 Amplify 提供變更應用程式後端所需的許可。如果您需要建立服務角色，請選擇建立新角色。如需建立服務角色的詳細資訊，請參閱新增具有部署後端資源許可的服務角色。

e. 選擇下一步。

7. 選擇 Save and deploy (儲存並部署)。

將分支連線至現有應用程式時重複使用後端

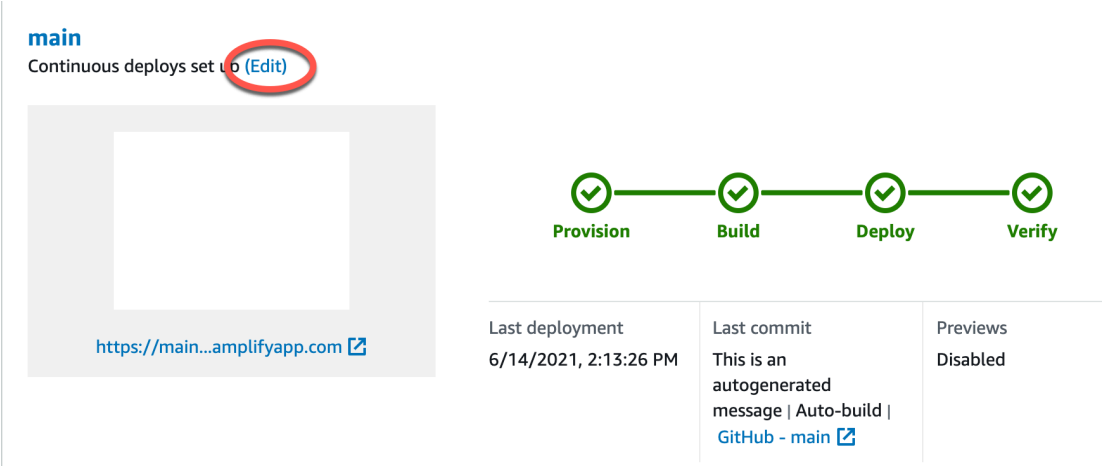
在將分支連線至現有的 Amplify 應用程式時重複使用後端

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇要連接新分支的應用程式。
3. 在導覽窗格中，選擇應用程式設定，一般。
4. 在分支區段中，選擇連接分支。
5. 在新增儲存庫分支頁面上，針對分支，從要連線的儲存庫中選取分支。
6. 針對應用程式名稱，選取要用於新增後端環境的應用程式。您可以選擇目前的應用程式或目前區域中的任何其他應用程式。
7. 針對環境，選取要新增的後端環境名稱。您可以使用現有環境或建立新的環境。
8. 如果您需要設定服務角色來授予 Amplify 對應用程式後端進行變更所需的許可，主控台會提示您執行此任務。如需建立服務角色的詳細資訊，請參閱新增具有部署後端資源許可的服務角色。
9. 根據預設，全堆疊 CI/CD 會關閉。關閉全堆疊 CI/CD 會導致應用程式以僅提取模式執行。在建置時，Amplify 只會自動產生`aws-exports.js`檔案，而不會修改您的後端環境。
10. 選擇下一步。
11. 選擇 Save and deploy (儲存並部署)。

編輯現有的前端以指向不同的後端

編輯前端 Amplify 應用程式以指向不同的後端

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇要編輯後端的應用程式。
3. 選擇託管環境索引標籤。
4. 找到要編輯的分支，然後選擇編輯。



5. 在選取要與此分支搭配使用的後端環境頁面上，針對應用程式名稱，選取您要編輯後端環境的前端應用程式。您可以選擇目前的應用程式或目前區域中的任何其他應用程式。
6. 針對後端環境，選取要新增的後端環境名稱。
7. 預設會啟用全堆疊 CI/CD。取消勾選此選項可關閉此後端的完整堆疊 CI/CD。關閉完整堆疊 CI/CD 會導致應用程式以僅提取模式執行。在建置時，Amplify 只會自動產生`aws-exports.js`檔案，而不會修改後端環境。
8. 選擇儲存。Amplify 會在您下次建置應用程式時套用這些變更。

建置應用程式的後端

透過 AWS Amplify ，您可以使用部署至 的資料、身分驗證、儲存和前端託管來建置完整堆疊應用程式 AWS。

AWS Amplify Gen 2 推出以 TypeScript 為基礎的程式碼優先開發人員體驗來定義後端。若要了解如何使用 Amplify Gen 2 建置後端並連接至您的應用程式，請參閱 Amplify 文件中的建置和連線後端。

如果您要尋找使用 CLI 和 Amplify Studio 為 Gen 1 應用程式建置後端的文件，請參閱 Gen 1 Amplify 文件中的建置和連接後端。

主題

- 為 Gen 2 應用程式建立後端
- 為 Gen 1 應用程式建立後端

為 Gen 2 應用程式建立後端

如需引導您使用 TypeScript 型後端建立 Amplify Gen 2 完整堆疊應用程式的教學課程，請參閱 Amplify 文件中的入門。

為 Gen 1 應用程式建立後端

在本教學課程中，您將使用 Amplify 設定完整堆疊 CI/CD 工作流程。您將部署前端應用程式到 Amplify 託管。然後，您將使用 Amplify Studio 建立後端。最後，您將將雲端後端連線至前端應用程式。

先決條件

開始本教學課程之前，請先完成下列先決條件。

註冊 AWS 帳戶

如果您還不是 AWS 客戶，則需要遵循線上說明來建立 AWS 帳戶。註冊可讓您存取 Amplify 和其他可與應用程式搭配使用 AWS 的服務。

建立 Git 儲存庫

Amplify 支援 GitHub、Bitbucket、GitLab 和 AWS CodeCommit。將您的應用程式推送到您的 Git 儲存庫。

安裝 Amplify 命令列界面 (CLI)

如需說明，請參閱 Amplify Framework 文件中的安裝 Amplify CLI。

步驟 1：部署前端

如果您在您想要用於此範例的 git 儲存庫中有現有的前端應用程式，您可以繼續執行部署前端應用程式的指示。

如果您需要建立新的前端應用程式以用於此範例，您可以遵循建立 React 應用程式文件中的建立 React 應用程式說明。

部署前端應用程式

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 在所有應用程式頁面上，選擇新增應用程式，然後選擇右上角的託管 Web 應用程式。
3. 選取您的 GitHub、Bitbucket、GitLab 或 AWS CodeCommit 儲存庫提供者，然後選擇繼續。
4. Amplify 授權存取您的 git 儲存庫。對於 GitHub 儲存庫，Amplify 現在使用 GitHub 應用程式功能來授權 Amplify 存取。

   如需安裝和授權 GitHub 應用程式的詳細資訊，請參閱 設定對 GitHub 儲存庫的 Amplify 存取權。
5. 在新增儲存庫分支頁面上，執行下列動作：

   a. 在最近更新的儲存庫清單中，選取要連線的儲存庫名稱。

   b. 在分支清單中，選取要連線的儲存庫分支名稱。

   c. 選擇下一步。
6. 在設定建置設定頁面上，選擇下一步。
7. 在檢閱頁面上，選擇儲存並部署。部署完成時，您可以在`amplifyapp.com`預設網域上檢視您的應用程式。

> **Note**
>
> 為了增強 Amplify 應用程式的安全性，在公有字尾清單 (PSL) 中註冊 amplifyapp.com 網域。為了進一步提高安全性，如果您需要在 Amplify 應用程式的預設網域名稱中設定敏感 Cookie，建議您使用字__`Host-`首為 的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

步驟 2：建立後端

現在您已將前端應用程式部署至 Amplify Hosting，您可以建立後端。使用下列指示來建立具有簡單資料庫和 GraphQL API 端點的後端。

建立後端

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 在所有應用程式頁面上，選取您在步驟 1 中建立的應用程式。
3. 在應用程式首頁上，選擇後端環境索引標籤，然後選擇開始使用。這會啟動預設預備環境的設定程序。
4. 設定完成後，選擇啟動 Studio 以存取 Amplify Studio 中的預備後端環境。

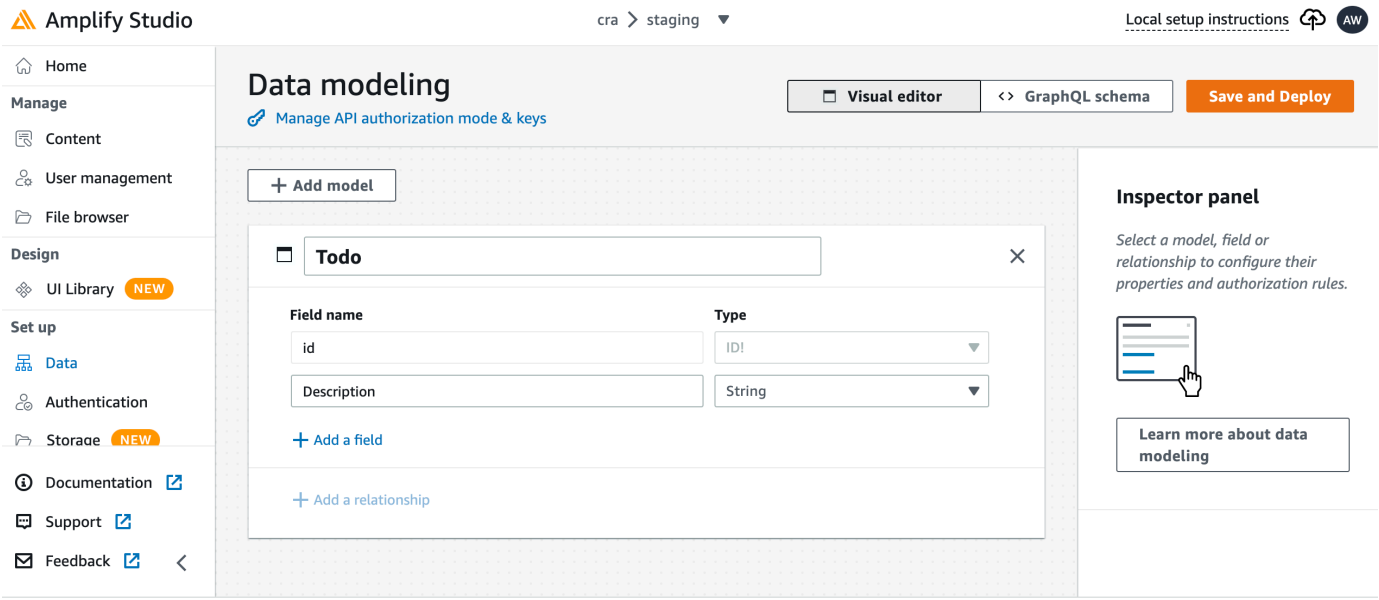
Amplify Studio 是一種視覺化界面，可用來建立和管理後端，並加速前端 UI 開發。如需 Amplify Studio 的詳細資訊，請參閱 Amplify Studio 文件。

使用以下指示，使用 Amplify Studio 視覺化後端建置器界面建立簡單的資料庫。

建立資料模型

1. 在應用程式的預備環境首頁上，選擇建立資料模型。這會開啟資料模型設計工具。
2. 在資料建模頁面上，選擇新增模型。
3. 針對標題，輸入 **Todo**。
4. 選擇新增欄位。
5. 對於欄位名稱，輸入 **Description**。

   下列螢幕擷取畫面是資料模型在設計工具中的外觀範例。



6. 選擇儲存並部署。
7. 返回 Amplify 託管主控台，預備環境部署將正在進行中。

在部署期間，Amplify Studio 會在後端建立所有必要 AWS 的資源，包括用於存取資料的 AWS AppSync GraphQL API 和用於託管 Todo 項目的 Amazon DynamoDB 資料表。Amplify 使用 CloudFormation 來部署後端，這可讓您將後端定義儲存為infrastructure-as-code。

步驟 3：將後端連接到前端

現在您已部署前端並建立包含資料模型的雲端後端，您需要連接它們。使用下列指示，透過 Amplify CLI 將後端定義向下提取至本機應用程式專案。

將雲端後端連線至本機前端

1. 開啟終端機視窗並導覽至本機專案的根目錄。
2. 在終端機視窗中執行下列命令，將紅色文字取代為專案的唯一應用程式 ID 和後端環境名稱。

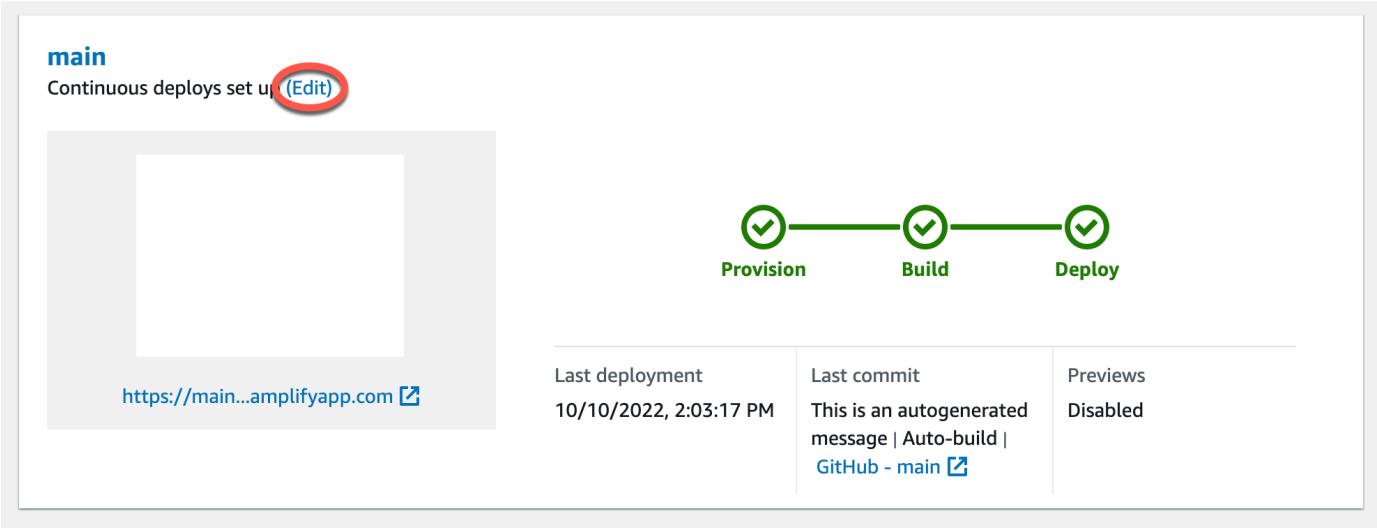
```
amplify pull --appId abcd1234 --envName staging
```

3. 依照終端機視窗中的指示完成專案設定。

現在您可以設定建置程序，將後端新增至連續部署工作流程。使用下列指示，在 Amplify 託管主控台中將前端分支與後端連接。

連接前端應用程式分支和雲端後端

1. 在應用程式首頁上，選擇託管環境索引標籤。
2. 找到主要分支，然後選擇編輯。



3. 在編輯目標後端視窗中，針對環境，選取要連線的後端名稱。在此範例中，選擇您在步驟 2 中建立的預備後端。

   預設會啟用全堆疊 CI/CD。取消勾選此選項可關閉此後端的完整堆疊 CI/CD。關閉完全堆疊 CI/CD 會導致應用程式以僅提取模式執行。在建置時，Amplify 只會自動產生`aws-exports.js`檔案，而不會修改您的後端環境。
4. 接著，您必須設定服務角色，為 Amplify 提供變更應用程式後端所需的許可。您可以使用現有的服務角色或建立新的服務角色。如需說明，請參閱新增具有部署後端資源許可的服務角色。
5. 新增服務角色後，返回編輯目標後端視窗，然後選擇儲存。
6. 若要完成將預備後端連接到前端應用程式的主要分支，請執行專案的新建置。

   執行以下任意一項：

   - 從您的 git 儲存庫中，推送一些程式碼以在 Amplify 主控台中啟動組建。
   - 在 Amplify 主控台中，導覽至應用程式的建置詳細資訊頁面，然後選擇重新部署此版本。

後續步驟

設定功能分支部署

依照我們建議的工作流程，設定具有多個後端環境的功能分支部署。

在 Amplify Studio 中建立前端 UI

使用 Studio 以一組ready-to-use UI 元件建置您的前端 UI，然後將其連接到您的應用程式後端。如需詳細資訊和教學課程，請參閱 Amplify Framework 文件中的 Amplify Studio 使用者指南。

進階部署功能

本章涵蓋增強 Amplify 託管工作流程的進階部署功能。這些功能提供額外的控制和功能，可協助團隊更有效地管理部署、確保程式碼品質，並在整個開發生命週期中維護安全性。

了解如何使用密碼身分驗證保護您的功能分支，以限制對未發行功能的存取。啟用提取請求的 Web 預覽，以在將程式碼合併至生產分支之前檢閱唯一預覽 URLs的變更。使用 Cypress 架構設定end-to-end測試，在將程式碼推送至生產環境之前擷取迴歸。雖然部署至 Amplify 按鈕功能不再可用，您仍然可以使用 Amplify 託管直接從儲存庫輕鬆部署應用程式。

主題

- 限制對 Amplify 應用程式分支的存取
- 提取請求的 Web 預覽
- 為您的 Amplify 應用程式設定end-to-end Cypress 測試
- 使用部署至 Amplify 按鈕來共用 GitHub 專案

限制對 Amplify 應用程式分支的存取

如果您正在使用未發行的功能，則可以使用密碼保護功能分支，以限制特定使用者的存取。在分支上設定存取控制時，當使用者嘗試存取分支的 URL 時，系統會提示他們輸入使用者名稱和密碼。

您可以設定套用至個別分支或全域套用至所有連線分支的密碼。在分支和全域層級啟用存取控制時，分支層級密碼優先於全域（應用程式）層級密碼。

Amplify 會調節嘗試存取受密碼保護資源的失敗請求。此行為可保護應用程式免於字典攻擊或其他嘗試讀取存取控制背後的資料。

使用下列程序來設定密碼，以限制對 Amplify 應用程式分支的存取。

在特徵分支上設定密碼

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇您要設定功能分支密碼的應用程式。
3. 在導覽窗格中，選擇託管，然後選擇存取控制。
4. 在存取控制設定區段中，選擇管理存取權。
5. 在管理存取控制頁面上，執行下列其中一項操作。
    - 設定套用至所有連線分支的使用者名稱和密碼

- 開啟管理所有分支的存取。例如，如果您已連接主要分支、開發分支和功能分支，您可以為所有分支套用相同的使用者名稱和密碼。
- 設定套用至個別分支的使用者名稱和密碼

a. 關閉管理所有分支的存取。

b. 找到您要管理的分支。針對存取設定，選擇所需的受限制密碼。

c. 針對使用者名稱，輸入使用者名稱。

d. 對於密碼，輸入密碼。

- 選擇儲存。

6. 如果您要管理伺服器端轉譯 (SSR) 應用程式的存取控制，請從 Git 儲存庫執行新建置，以重新部署應用程式。此步驟是啟用 Amplify 以套用存取控制設定的必要步驟。

提取請求的 Web 預覽

Web 預覽提供開發和品質保證 (QA) 團隊在將程式碼合併到生產或整合分支之前，預覽提取請求 (PRs) 的變更。提取請求可讓您告訴其他人您已推送到儲存庫中分支的變更。開啟提取請求後，您可以與協作者討論和檢閱潛在變更，並在變更合併到基本分支之前新增後續遞交。

Web 預覽會將對儲存庫提出的每個提取請求部署到唯一的預覽 URL，這與主要網站使用的 URL 完全不同。對於使用 Amplify CLI 或 Amplify Studio 佈建後端環境的應用程式，每個提取請求 (僅限私有 Git 儲存庫) 都會建立臨時後端，並在 PR 關閉時刪除。

為您的應用程式開啟 Web 預覽時，每個 PR 都會計入每個應用程式的 50 個分支的 Amplify 配額。若要避免超過此配額，請務必關閉您的 PRs。如需配額的詳細資訊，請參閱 Amplify 託管服務配額。

> **Note**
>
> 目前，使用 AWS CodeCommit 做為儲存庫提供者時，無法使用 `AWS_PULL_REQUEST_ID`環境變數。

Web 預覽安全性

基於安全考量，您可以在所有具有私有儲存庫的應用程式上啟用 Web 預覽，但並非所有具有公有儲存庫的應用程式上都啟用。如果您的 Git 儲存庫是公有的，您只能為不需要 IAM 服務角色的應用程式設定預覽。例如，具有後端的應用程式和部署到`WEB_COMPUTE`託管平台的應用程式需要 IAM 服務角色。

因此，如果這些類型的應用程式儲存庫是公有的，則您無法啟用這些類型的 Web 預覽。Amplify 會強制執行此限制，以防止第三方提交使用您應用程式的 IAM 角色許可執行的任意程式碼。

當公有儲存庫中的應用程式啟用 Web 預覽時，使用 SSR 運算角色時，您需要仔細管理哪些分支可以存取該角色。建議您不要使用應用程式層級角色。反之，您應該在分支層級連接運算角色。這可讓您僅將許可授予需要存取特定資源的分支。如需詳細資訊，請參閱 新增 SSR 運算角色以允許存取 AWS 資源。

啟用提取請求的 Web 預覽

對於存放在 GitHub 儲存庫中的應用程式，Web 預覽使用 Amplify GitHub 應用程式進行儲存庫存取。如果您在先前使用 OAuth 從 GitHub 儲存庫部署的現有 Amplify 應用程式上啟用 Web 預覽，您必須先遷移應用程式以使用 Amplify GitHub 應用程式。如需遷移說明，請參閱 將現有OAuth應用程式遷移至 Amplify GitHub 應用程式。

啟用提取請求的 Web 預覽

1. 選擇託管，然後選擇預覽。

   > **Note**
   >
   > 只有當應用程式設定為持續部署並連接到 git 儲存庫時，預覽才會出現在應用程式設定選單中。如需此部署類型的指示，請參閱開始使用現有程式碼。

2. 僅適用於 GitHub 儲存庫，請執行下列動作，在您的帳戶中安裝並授權 Amplify GitHub 應用程式：

   a. 在安裝 GitHub 應用程式以啟用預覽視窗中，選擇安裝 GitHub 應用程式。

   b. 選取您要設定 Amplify GitHub 應用程式的 GitHub 帳戶。

   c. 在 https：//Github.com 上開啟一個頁面，以設定您帳戶的儲存庫許可。

   d. 執行以下任意一項：

      - 若要將安裝套用至所有儲存庫，請選擇所有儲存庫。
      - 若要將安裝限制為您選取的特定儲存庫，請選擇僅選取儲存庫。請務必在您選取的儲存庫中包含您要為其啟用 Web 預覽的應用程式儲存庫。

   e. 選擇儲存

3. 啟用儲存庫的預覽後，請返回 Amplify 主控台以啟用特定分支的預覽。在預覽頁面上，從清單中選擇分支，然後選擇編輯設定。

4. 在管理預覽設定頁面上，開啟提取請求預覽。然後選擇 Confirm (確認)。

5. 對於 Fullstack 應用程式，請執行下列其中一項操作：
   - 選擇，為每個提取請求建立新的後端環境。此選項可讓您測試變更，而不會影響生產。
   - 選擇將此分支的所有提取請求指向現有環境。
6. 選擇確認。

下次您提交分支的提取請求時，Amplify 會建置您的 PR 並將其部署至預覽 URL。關閉提取請求後，會刪除預覽 URL，並刪除連結至提取請求的任何暫時後端環境。僅適用於 GitHub 儲存庫，您可以直接從 GitHub 帳戶中的提取請求存取 URL 預覽。

具有子網域的 Web 預覽存取

提取請求的 Web 預覽可透過連線至由 Amazon Route 53 管理之自訂網域的 Amplify 應用程式子網域存取。關閉提取請求時，會自動刪除與提取請求相關聯的分支和子網域。這是您為應用程式設定模式型功能分支部署後 Web 預覽的預設行為。如需設定自動子網域的指示，請參閱 設定 Amazon Route 53 自訂網域的自動子網域。

為您的 Amplify 應用程式設定end-to-end Cypress 測試

您可以在 Amplify 應用程式的測試階段執行end-to-end (E2E) 測試，以在將程式碼推送至生產環境之前擷取迴歸。您可以在建置規格 YAML 中設定測試階段。目前，您只能在建置期間執行 Cypress 測試架構。

Cypress 是以 JavaScript 為基礎的測試架構，可讓您在瀏覽器上執行 E2E 測試。如需示範如何設定 E2E 測試的教學課程，請參閱部落格文章使用 Amplify 執行完整堆疊 CI/CD 部署的end-to-end Cypress 測試。

將 Cypress 測試新增至現有的 Amplify 應用程式

您可以在 Amplify 主控台中更新應用程式的建置設定，將 Cypress 測試新增至現有應用程式。建置規格 YAML 包含組建命令的集合，以及 Amplify 用來執行組建的相關設定。使用 `test` 步驟在建置時執行任何測試命令。對於 E2E 測試，Amplify Hosting 提供與 Cypress 的深度整合，可讓您為測試產生 UI 報告。

下列清單說明測試設定及其使用方式。

preTest

安裝執行 Cypress 測試所需的相依性。Amplify Hosting 使用 mochawesome 來產生報告，以檢視您的測試結果，並等待在建置期間設定 localhost 伺服器。

test

執行 cypress 命令，以使用 mochawesome 執行測試。

postTest

mochawesome 報告是從輸出 JSON 產生。請注意，如果您使用的是 Yarn，您必須以無提示模式執行此命令，以產生mochawesome 報告。對於 Yarn，您可以使用下列命令。

```
yarn run --silent mochawesome-merge cypress/report/mochawesome-report/
mochawesome*.json > cypress/report/mochawesome.json
```

artifacts>baseDirectory

執行測試的目錄。

artifacts>configFilePath

產生的測試報告資料。

artifacts>files

產生的成品（螢幕擷取畫面和影片）可供下載。

下列範例摘錄自建置規格amplify.yml檔案，說明如何將 Cypress 測試新增至您的應用程式。

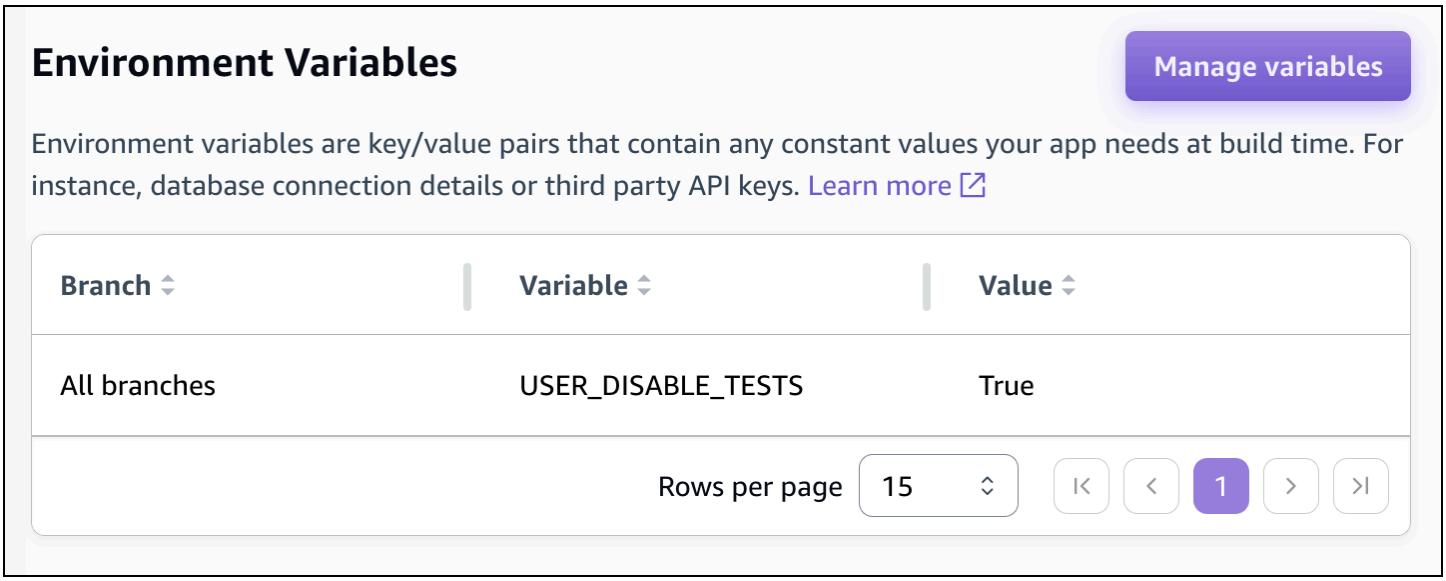
```
test:
  phases:
    preTest:
      commands:
        - npm ci
        - npm install -g pm2
        - npm install -g wait-on
        - npm install mocha mochawesome mochawesome-merge mochawesome-report-generator
        - pm2 start npm -- start
        - wait-on http://localhost:3000
    test:
      commands:
        - 'npx cypress run --reporter mochawesome --reporter-options
 "reportDir=cypress/report/mochawesome-
report,overwrite=false,html=false,json=true,timestamp=mmddyyyy_HHMMss"'
    postTest:
```

```
      commands:
        - npx mochawesome-merge cypress/report/mochawesome-report/mochawesome*.json >
cypress/report/mochawesome.json
        - pm2 kill
  artifacts:
    baseDirectory: cypress
    configFilePath: '**/mochawesome.json'
    files:
      - '**/*.png'
      - '**/*.mp4'
```

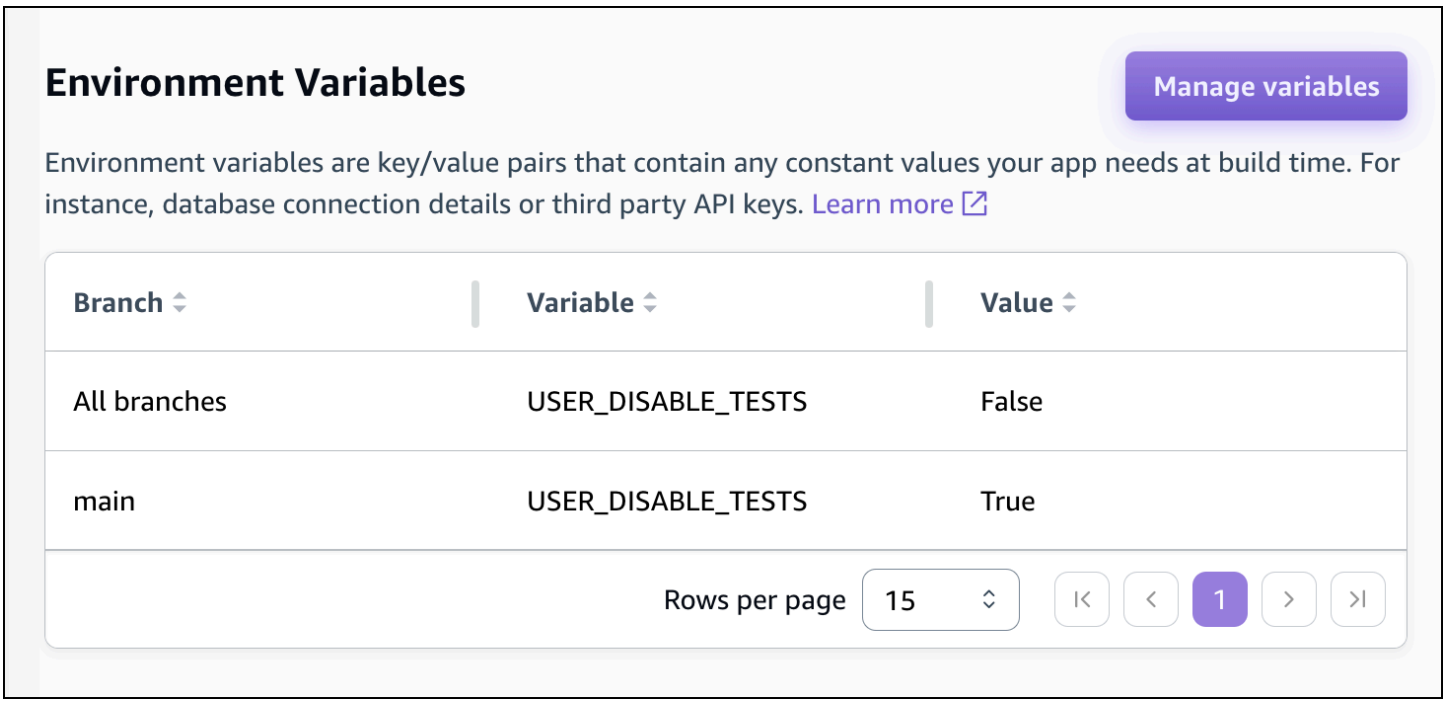
關閉 Amplify 應用程式或分支的測試

將測試組態新增至`amplify.yml`建置設定後，每個建置、每個分支都會執行 `test` 步驟。如果您想要全域停用執行中的測試，或只對特定分支執行測試，您可以使用 USER_DISABLE_TESTS環境變數而不修改建置設定。

若要全域停用所有分支的測試，請`true`為所有分支新增值為 USER_DISABLE_TESTS的環境變數。下列螢幕擷取畫面顯示 Amplify 主控台中的環境變數區段，並停用所有分支的測試。



若要停用特定分支的測試，請`false`為所有分支新增值為 USER_DISABLE_TESTS的環境變數，然後為您要停用的每個分支新增值為 的覆寫`true`。在下列螢幕擷取畫面中，會停用主分支上的測試，並針對其他每個分支啟用測試。



使用此變數停用測試會導致在建置期間完全略過測試步驟。若要重新啟用測試，請將此值設定為 `false`，或刪除環境變數。

使用部署至 Amplify 按鈕來共用 GitHub 專案

> **Important**
>
> 使用部署至 Amplify 託管按鈕的一鍵式部署不再可用。若要從儲存庫部署，請在 Amplify Hosting 中建立新的應用程式。如需說明，請參閱開始將應用程式部署到 Amplify 託管。

部署至 Amplify 託管按鈕可讓您公開或在團隊內共用 GitHub 專案。以下是 按鈕的影像：

將部署至 Amplify 託管按鈕新增至儲存庫或部落格

將按鈕新增至您的 GitHub README.md 檔案、部落格文章或任何其他轉譯 HTML 的標記頁面。按鈕具有下列兩個元件：

1. 位於 URL 的 SVG 映像 `https://oneclick.amplifyapp.com/button.svg`

2. 包含 GitHub 儲存庫連結的 Amplify 主控台 URL。您的 可以複製儲存庫的 URL，例如 `https://github.com/username/repository`，也可以提供特定資料夾的深層連結，例如 `https://github.com/username/repository/tree/branchname/folder`。Amplify 託管會在您的儲存庫中部署預設分支。應用程式連線後，即可連接其他分支。

使用下列範例將按鈕新增至 Markdown 檔案，例如您的 GitHub README.md：//。`https://github.com/username/repository` 將 取代為儲存庫的 URL。

```
[![amplifybutton](https://oneclick.amplifyapp.com/button.svg)](https://
console.aws.amazon.com/amplify/home#/deploy?repo=https://github.com/username/
repository)
```

使用下列範例將 按鈕新增至任何 HTML 文件。`https://github.com/username/repository` 將取代為儲存庫的 URL。

```
<a href="https://console.aws.amazon.com/amplify/home#/deploy?repo=https://github.com/
username/repository">
    <img src="https://oneclick.amplifyapp.com/button.svg" alt="Deploy to Amplify
 Hosting">
</a>
```

設定 Amplify 應用程式的重新導向和重寫

重新導向可讓 Web 伺服器將導覽從一個 URL 重新路由到另一個 URL。使用重新導向的常見原因包括自訂 URL 的外觀、避免連結中斷、在不變更其地址的情況下移動應用程式或網站的託管位置，以及將請求的 URL 變更為 Web 應用程式所需的格式。

了解 Amplify 支援的重新導向

Amplify 支援主控台中的下列重新導向類型。

永久重新導向 (301)

301 重新導向預期用於網址目的地的持續變更。原始地址的搜尋引擎排名歷史記錄會套用到新目的地地址。重新導向會在用戶端上進行，因此瀏覽器導覽列會在重新導向後顯示目的地地址。

使用 301 重新導向的常見原因包括：

- 為了避免頁面的地址變更時造成連結中斷。
- 為了避免當使用者在地址中提供了可預測的錯字時造成連結中斷。

暫時重新導向 (302)

302 重新導向預期用於網址目的地的暫時變更。原始地址的搜尋引擎排名歷史記錄不適用於新的目的地地址。重新導向會在用戶端上進行，因此瀏覽器導覽列會在重新導向後顯示目的地地址。

使用 302 重新導向的常見原因包括：

- 為了在對原始地址進行修復時提供繞道目的地。
- 提供使用者介面 A/B 比較的測試頁面。

> **Note**
>
> 如果您的應用程式傳回非預期的 302 回應，則錯誤可能是因為您對應用程式的重新導向和自訂標頭組態所做的變更所造成。若要解決此問題，請確認您的自訂標頭有效，然後為您的應用程式重新啟用預設的 404 重寫規則。

重寫 (200)

200 重新導向 (重寫) 是為了就好像是從原始地址提供一般，顯示來自目的地地址的內容。搜尋引擎排名歷史記錄會繼續套用到原始地址。重新導向會在伺服器端上進行，因此瀏覽器導覽列會在重新導向後顯示原始地址。使用 200 重新導向的常見原因包括：

- 為了將整個網站重新導向到新的託管位置，而不變更網站的地址。
- 為了將對單一頁面 Web 應用程式 (SPA) 的所有流量重新導向其 index.html 頁面，以由用戶端路由器功能處理。

找不到 (404)

當請求指向不存在的地址時，會發生 404 重新導向。會顯示 404 目的地頁面，而不是請求的頁面。404 重新導向發生的常見原因包括：

- 為了避免使用者輸入錯誤的 URL 時的中斷連結訊息。
- 為了將 Web 應用程式不存在頁面的請求指向其 index.html 頁面，以由用戶端路由器功能處理。

了解重新導向的順序

重新導向會從清單頂端向下套用。請確定您的排序具有預期的效果。例如，下列重新導向順序會造成將 /docs/ 下指定路徑的所有請求重新導向到 /documents/ 下的相同路徑，/docs/specific-filename.html 除外，它會重新導向至 /documents/different-filename.html：

```
/docs/specific-filename.html /documents/different-filename.html 301
/docs/<*> /documents/<*>
```

以下重新導向順序會忽略 specific-filename.html 到 different-filename.html 的重新導向：

```
/docs/<*> /documents/<*>
/docs/specific-filename.html /documents/different-filename.html 301
```

了解 Amplify 如何轉送查詢參數

您可以使用查詢參數來進一步控制您的 URL 比對。Amplify 會將所有查詢參數轉送到 301 和 302 重新導向的目的地路徑，但以下情況除外：

- 如果原始地址包含設為特定值的查詢字串，Amplify 不會轉送查詢參數。在此情況下，重新導向僅適用於具有指定查詢值的目的地 URL 請求。

- 如果相符規則的目的地地址具有查詢參數，則不會轉送查詢參數。例如，如果重新導向的目的地地址為 `https://example-target.com?q=someParam`，則不會傳遞查詢參數。

在 Amplify 主控台中建立和編輯重新導向

您可以在 Amplify 主控台中建立和編輯應用程式的重新導向。開始之前，您將需要以下重新導向部分的相關資訊。

原始地址

使用者請求的地址。

目的地地址

實際提供使用者所看到內容的地址。

重新導向類型

類型包括永久重新導向 (301)、暫時重新導向 (302)、重寫 (200) 或找不到 (404)。

兩個字母的國家/地區代碼（選用）

您可以包含的值，以依地理區域分割應用程式的使用者體驗。

在 Amplify 主控台中建立重新導向

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇您要為其建立重新導向的應用程式。
3. 在導覽窗格中，選擇託管，然後選擇重寫和重新導向。
4. 在重寫和重新導向頁面上，選擇管理重新導向。
5. 在重寫和重新導向 JSON 編輯器中手動新增或更新重新導向。
   a. 針對 `source`，指定使用者請求的原始地址。
   b. 針對 `status`，指定重新導向的類型。
   c. 針對 `target`，指定將內容轉譯給使用者的目的地地址。
   d. （選用）針對 `condition`，輸入兩個字母的國家/地區代碼條件。
6. 選擇儲存。

重新導向和重寫範例參考

本節提供各種常見重新導向案例的範例。

Important

網域特定的重新導向不支援來源欄位中的路徑元件。
支援：

- `"source": "https://example.com"` （路徑會自動附加）

不支援：

- `"source": "https://example.com/specific-path"`
- 目前不支援具有domain+path組合的規則。

替代模式
對於網域特定的路徑重新導向，請使用：

1. 單獨的僅限網域規則（路徑會自動附加）
2. 具有條件式邏輯的僅限路徑規則
3. 多個規則組合

您可以使用這些範例來了解在 Amplify 主控台 JSON 編輯器中建立您自己的重新導向和重寫的 JSON 語法。

Note

原始地址網域比對不區分大小寫。

主題

- 簡單重新導向和重寫
- 單一頁面 Web 應用程式 (SPA) 的重新導向
- 反向代理重寫
- 追蹤斜線和乾淨的 URLs

- 預留位置
- 查詢字串和路徑參數
- 區域型重新導向
- 在重新導向和重寫中使用萬用字元表達式

簡單重新導向和重寫

您可以使用下列範例，將特定頁面永久重新導向至新地址。

原始地址	目的地地址	重新導向類型	國家代碼
/original.html	/destination.html	permanent redirect (301)	

JSON format (JSON 格式)

```
[
  {
    "source": "/original.html",
    "status": "301",
    "target": "/destination.html",
    "condition": null
  }
]
```

您可以使用下列範例，將資料夾下的任何路徑重新導向至不同資料夾下的相同路徑。

原始地址	目的地地址	重新導向類型	國家代碼
/docs/<*>	/documents/<*>	permanent redirect (301)	

JSON format (JSON 格式)

```
[
  {
    "source": "/docs/<*>",
```

```
    "status": "301",
    "target": "/documents/<*>",
    "condition": null
  }
]
```

您可以使用下列範例，將所有流量重新導向至 index.html 作為重寫。在此情況下，重寫會讓使用者以為他們已抵達原始地址。

原始地址	目的地地址	重新導向類型	國家代碼
/<*>	/index.html	rewrite (200)	

JSON format (JSON 格式)

```
[
  {
    "source": "/<*>",
    "status": "200",
    "target": "/index.html",
    "condition": null
  }
]
```

您可以使用下列範例來使用重寫來變更顯示給使用者的子網域。

原始地址	目的地地址	重新導向類型	國家代碼
https://mydomain.com	https://www.mydomain.com	rewrite (200)	

JSON format (JSON 格式)

```
[
  {
    "source": "https://mydomain.com",
    "status": "200", "target": "https://www.mydomain.com",
    "condition": null
```

```
  }
]
```

您可以使用下列範例來重新導向至具有路徑字首的不同網域。

原始地址	目的地地址	重新導向類型	國家代碼
https://mydomain.com	https://www.mydomain.com/documents	temporary redirect (302)	

JSON format (JSON 格式)

```
[
  {
    "source": "https://mydomain.com",
    "status": "302",
    "target": "https://www.mydomain.com/documents/",
    "condition": null
  }
]
```

您可以使用下列範例，將找不到的資料夾下的路徑重新導向至自訂 404 頁面。

原始地址	目的地地址	重新導向類型	國家代碼
/<*>	/404.html	not found (404)	

JSON format (JSON 格式)

```
[
  {
    "source": "/<*>",
    "status": "404",
    "target": "/404.html",
    "condition": null
  }
```

```
]
```

> **Important**
>
> 不支援以網域為基礎的來源規則（例如 "https://domain.com/path") 中的路徑元件，這會導致規則被忽略而不會發生錯誤。

單一頁面 Web 應用程式 (SPA) 的重新導向

大多數 SPA 架構都支援 HTML5 history.pushState() 來變更瀏覽器位置，而無需啟動伺服器請求。這對於從根目錄 (或 /index.html) 開始導覽的使用者有效，但對於直接導覽至任何其他頁面的使用者則無效。

下列範例使用規則表達式，將所有檔案設定為 index.html 的 200 重寫，但規則表達式中指定的副檔名除外。

原始地址	目的地地址	重新導向類型	國家代碼
</^[^.]+$\|\.(?!(css\|gif\|ico\|jpg\|js\|png\|txt\|svg\|woff\|woff2\|ttf\|map\|json\|webp)$)([^.]+$)/>	/index.html	200	

JSON format (JSON 格式)

```
[
  {
    "source": "</^[^.]+$|\.(?!(css|gif|ico|jpg|js|png|txt|svg|woff|woff2|ttf|map|json|
webp)$)([^.]+$)/>",
    "status": "200",
    "target": "/index.html",
    "condition": null
  }
]
```

反向代理重寫

下列範例使用重寫從另一個位置代理內容，讓使用者看到網域未變更。HTTPS 是唯一支援反向代理的通訊協定。

原始地址	目的地地址	重新導向類型	國家代碼
/images/<*>	https://images.otherdomain.com/<*>	rewrite (200)	

JSON format (JSON 格式)

```
[
  {
    "source": "/images/<*>",
    "status": "200",
    "target": "https://images.otherdomain.com/<*>",
    "condition": null
  }
]
```

追蹤斜線和乾淨的 URLs

若要建立乾淨的 URL 結構，像是 about 而不是 about.html，Hugo 之類的靜態網站產生器會為具有 index.html 的頁面產生目錄 (/about/index.html)。Amplify 會視需要新增結尾斜線，以自動建立乾淨的 URLs。下表重點說明不同的案例：

瀏覽器中的使用者輸入	地址列中的 URL	提供的文件
/about	/about	/about.html
/about (when about.html returns 404)	/about/	/about/index.html
/about/	/about/	/about/index.html

預留位置

您可以使用下列範例，將資料夾結構中的路徑重新導向至另一個資料夾中的相符結構。

原始地址	目的地地址	重新導向類型	國家代碼
/docs/<year>/<month>/<date>/<itemid>	/documents/<year>/<month>/<date>/<itemid>	permanent redirect (301)	

JSON format (JSON 格式)

```
[
  {
    "source":  "/docs/<year>/<month>/<date>/<itemid>",
    "status": "301",
    "target": "/documents/<year>/<month>/<date>/<itemid>",
               "condition": null
   }
]
```

查詢字串和路徑參數

> **Warning**
>
> 請勿在 URLs 中包含秘密、登入資料或敏感資料做為路徑或查詢參數。這些值可在 Amplify 應用程式的存取日誌中以純文字顯示。

您可以使用下列範例，將路徑重新導向至名稱符合原始地址中查詢字串元素值的資料夾：

原始地址	目的地地址	重新導向類型	國家代碼
/docs?id=<my-blog-id-value	/documents/<my-blog-post-id-value>	permanent redirect (301)	

JSON format (JSON 格式)

```
[
  {
    "source": "/docs?id=<my-blog-id-value>",
    "status": "301",
    "target": "/documents/<my-blog-id-value>",
    "condition": null
  }
]
```

> **Note**
>
> Amplify 會將所有查詢字串參數轉送到 301 和 302 重新導向的目的地路徑。不過，如果原始地址包含設為特定值的查詢字串，如本範例所示，Amplify 不會轉送查詢參數。在此情況下，重新導向僅適用於具有指定查詢值 的目的地地址請求id。

您可以使用下列範例，將所有在資料夾結構指定層級找不到的路徑重新導向至指定資料夾中的 index.html。

原始地址	目的地地址	重新導向類型	國家代碼
/documents/<folder>/<child-folder>/<grand-child-folder>	/documents/index.html	not found (404)	

JSON format (JSON 格式)

```
[
  {
    "source": "/documents/<x>/<y>/<z>",
    "status": "404",
    "target": "/documents/index.html",
    "condition": null
  }
```

```
]
```

區域型重新導向

您可以使用下列範例，根據區域重新導向請求。

原始地址	目的地地址	重新導向類型	國家代碼
/documents	/documents/us/	temporary redirect (302)	<US>

JSON format (JSON 格式)

```
[
  {
    "source": "/documents",
    "status": "302",
    "target": "/documents/us/",
    "condition": "<US>"
  }
]
```

在重新導向和重寫中使用萬用字元表達式

您可以在原始地址<*>中使用萬用字元表達式 進行重新導向或重寫。您必須將表達式放在原始地址的結尾，而且必須是唯一的。Amplify 會忽略包含多個萬用字元表達式的原始地址，或在不同的配置中使用它。

以下是具有萬用字元表達式的有效重新導向範例。

原始地址	目的地地址	重新導向類型	國家代碼
/docs/<*>	/documents/<*>	permanent redirect (301)	

下列兩個範例示範使用萬用字元表達式的無效重新導向。

原始地址	目的地地址	重新導向類型	國家代碼
/docs/<*>/content	/documents/<*>/content	permanent redirect (301)	
/docs/<*>/content/<*>	/documents/<*>/content/<*>	permanent redirect (301)	

在 Amplify 應用程式中使用環境變數

環境變數是索引鍵/值對，您可以將其新增至應用程式的設定，以便 Amplify Hosting 使用。最佳實務是，您可以使用環境變數來公開應用程式組態資料。您新增的所有環境變數都會加密，以防止惡意存取。

Amplify 對您建立的環境變數強制執行下列限制。

- Amplify 不允許您建立具有 AWS字首的環境變數名稱。此字首僅供 Amplify 內部使用。
- 環境變數的值不能超過 5500 個字元。

> **Important**
>
> 請勿使用環境變數來存放秘密。對於 Gen 2 應用程式，請使用 Amplify 主控台中的秘密管理功能。如需詳細資訊，請參閱 Amplify 文件中的秘密和環境變數。對於 Gen 1 應用程式，將秘密存放在使用 AWS Systems Manager 參數存放區建立的環境秘密中。如需詳細資訊，請參閱管理環境秘密。

Amplify 環境變數參考

依預設，您可以在 Amplify 主控台中存取下列環境變數。

變數名稱	Description	範例值
_BUILD_TIMEOUT	建置逾時持續時間，以分鐘為單位。 最小值為 5。 最大值為 120。	`30`
_LIVE_UPDATES	工具將升級至最新版本。	`[{"name":"Amplify CLI","pkg":"@aws-amplify/cli","type":"npm","version":"latest"}]`

變數名稱	Description	範例值
USER_DISABLE_TESTS	在建置期間略過測試步驟。您可以停用應用程式中所有分支或特定分支的測試。 此環境變數用於在建置階段執行測試的應用程式。如需設定此變數的詳細資訊，請參閱 關閉 Amplify 應用程式或分支的測試。	`true`
AWS_APP_ID	目前建置的應用程式 ID	`abcd1234`
AWS_BRANCH	目前建置的分支名稱	`main`, `develop`, `beta`, `v2.0`
AWS_BRANCH_ARN	目前建置的分支 Amazon Resource Name (ARN)	`aws:arn:amplify:us-west-2:123456789012:appname/branch/...`
AWS_CLONE_URL	用來擷取 Git 儲存庫內容的複製 URL	`git@github.com:<user-name>/<repo-name>.git`
AWS_COMMIT_ID	目前建置的遞交 ID 重建的「HEAD」	`abcd1234`
AWS_JOB_ID	目前建置的任務 ID。 這包括一些「0」的填補，因此一律具有相同的長度。	`0000000001`

變數名稱	Description	範例值
AWS_PULL_REQUEST_ID	提取請求 Web 預覽組建的提取請求 ID。 使用 AWS CodeCommit 做為儲存庫提供者時，無法使用此環境變數。	1
AWS_PULL_REQUEST_SOURCE_BRANCH	Amplify 主控台中要提交至應用程式分支之提取請求預覽的功能分支名稱。	featureA
AWS_PULL_REQUEST_DESTINATION_BRANCH	Amplify 主控台中要提交功能分支提取請求的應用程式分支名稱。	main
AMPLIFY_AMAZON_CLIENT_ID	Amazon 用戶端 ID	123456
AMPLIFY_AMAZON_CLIENT_SECRET	Amazon 用戶端秘密	example123456
AMPLIFY_FACEBOOK_CLIENT_ID	Facebook 用戶端 ID	123456
AMPLIFY_FACEBOOK_CLIENT_SECRET	Facebook 用戶端秘密	example123456
AMPLIFY_GOOGLE_CLIENT_ID	Google 用戶端 ID	123456
AMPLIFY_GOOGLE_CLIENT_SECRET	Google 用戶端秘密	example123456
AMPLIFY_DIFF_DEPLOY	啟用或停用 diff 型前端部署。如需詳細資訊，請參閱設定以差異為基礎的前端建置和部署。	true

變數名稱	Description	範例值
AMPLIFY_DIFF_DEPLOY_ROOT	相對於儲存庫根目錄，用於 diff 型前端部署比較的路徑。	`dist`
AMPLIFY_DIFF_BACKEND	啟用或停用以 diff 為基礎的後端建置。這僅適用於 Gen 1 應用程式。如需詳細資訊，請參閱為 Gen 1 應用程式設定差異型後端建置	`true`
AMPLIFY_BACKEND_PULL_ONLY	Amplify 會管理此環境變數。這僅適用於 Gen 1 應用程式。如需詳細資訊，請參閱編輯現有的前端以指向不同的後端	`true`
AMPLIFY_BACKEND_APP_ID	Amplify 會管理此環境變數。這僅適用於 Gen 1 應用程式。如需詳細資訊，請參閱編輯現有的前端以指向不同的後端	`abcd1234`
AMPLIFY_SKIP_BACKEND_BUILD	如果您的建置規格中沒有後端區段，且想要停用後端建置，請將此環境變數設定為 `true`。這僅適用於 Gen 1 應用程式。	`true`
AMPLIFY_ENABLE_DEBUG_OUTPUT	將此變數設定為 ，`true`以在日誌中列印堆疊追蹤。這有助於偵錯後端建置錯誤。	`true`
AMPLIFY_MONOREPO_APP_ROOT	用來指定 monorepo 應用程式根目錄的路徑，相對於儲存庫根目錄。	`apps/react-app`
AMPLIFY_USERPOOL_ID	為驗證匯入的 Amazon Cognito 使用者集區的 ID	`us-west-2_example`

變數名稱	Description	範例值
AMPLIFY_WEBCLIENT_ID	Web 應用程式要使用的應用程式用戶端 ID 應用程式用戶端必須設定為可存取 AMPLIFY_USERPOOL_ID 環境變數指定的 Amazon Cognito 使用者集區。	123456
AMPLIFY_NATIVECLIENT_ID	原生應用程式要使用的應用程式用戶端 ID 應用程式用戶端必須設定為可存取 AMPLIFY_USERPOOL_ID 環境變數指定的 Amazon Cognito 使用者集區。	123456
AMPLIFY_IDENTITYPOOL_ID	Amazon Cognito 身分集區的 ID	example-identitypool-id
AMPLIFY_PERMISSIONS_BOUNDARY_ARN	IAM 政策的 ARN，可用作套用至 Amplify 建立之所有 IAM 角色的許可界限。	arn:aws:iam::123456789012:policy/example-policy
AMPLIFY_DESTRUCTIVE_UPDATES	將此環境變數設定為 true，以允許使用可能導致資料遺失的結構描述操作來更新 GraphQL API。	true

> **Note**
>
> AMPLIFY_AMAZON_CLIENT_ID 和 AMPLIFY_AMAZON_CLIENT_SECRET環境變數是 OAuth 權杖，而不是 AWS 存取金鑰和私密金鑰。

前端架構環境變數

如果您使用支援自己的環境變數的前端架構來開發應用程式，請務必了解這些變數與您在 Amplify 主控台中設定的環境變數不同。例如，React （ 字首 REACT_APP) 和 Gatsby （ 字首 GATSBY) 可讓您建立執行期環境變數，這些架構會自動綁定到您的前端生產建置中。若要了解使用這些環境變數來存放值的效果，請參閱您正在使用的前端架構文件。

在這些前端架構字首的環境變數中存放敏感值，例如 API 金鑰，並非最佳實務，而且強烈建議不要這麼做。

設定環境變數

使用下列指示，在 Amplify 主控台中設定應用程式的環境變數。

> **Note**
>
> 只有當應用程式設定為持續部署並連接到 git 儲存庫時，環境變數才會顯示在 Amplify 主控台的應用程式設定選單中。如需此部署類型的說明，請參閱開始使用現有程式碼。

設定環境變數

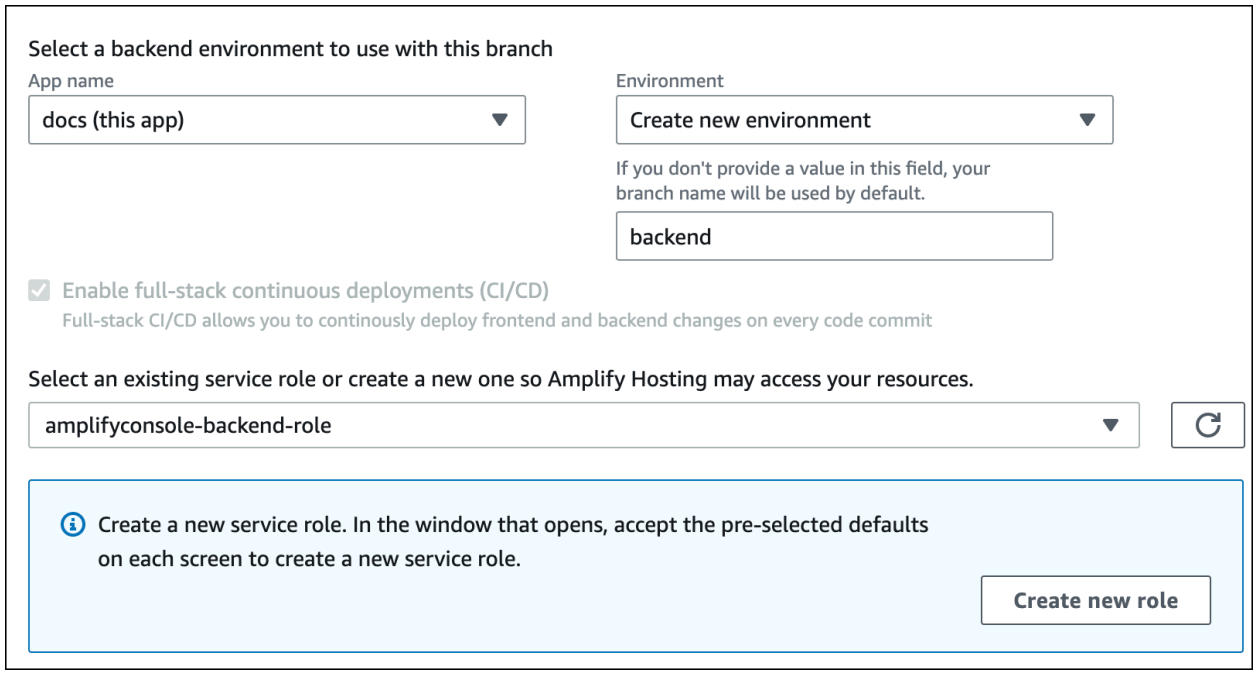
1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 在 Amplify 主控台中，選擇託管，然後選擇環境變數。
3. 在環境變數頁面上，選擇管理變數。
4. 針對變數，輸入您的金鑰。在值中，輸入您的值。根據預設，Amplify 會將環境變數套用至所有分支，因此您不必在連接新分支時重新輸入變數。
5. （選用） 若要自訂特定於分支的環境變數，請新增分支覆寫，如下所示：
   a. 選擇動作，然後選擇新增變數覆寫。
   b. 您現在有您分支專屬的一組環境變數。
6. 選擇儲存。

使用社交登入的身分驗證參數建立新的後端環境

將分支連線至應用程式

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 將分支連線至應用程式的程序會因您要將分支連線至新應用程式或現有應用程式而有所不同。
   - 將分支連線至新的應用程式

     a. 在建置設定頁面上，找到選取要與此分支搭配使用的後端環境區段。針對環境，選擇建立新環境，然後輸入後端環境的名稱。下列螢幕擷取畫面顯示選取後端環境，以搭配建置設定頁面的此分支使用，並為後端環境名稱**backend**輸入 。



     b. 展開建置設定頁面上的進階設定區段，並新增社交登入金鑰的環境變數。例如，**AMPLIFY_FACEBOOK_CLIENT_SECRET** 是有效的環境變數。如需預設可用的 Amplify 系統環境變數清單，請參閱 中的 資料表Amplify 環境變數參考。

   - 將分支連接至現有應用程式

     a. 如果您要將新的分支連接到現有的應用程式，請在連接分支之前設定社交登入環境變數。在導覽窗格中，選擇應用程式設定、環境變數。

     b. 在環境變數區段中，選擇管理變數。

     c. 在管理變數區段中，選擇新增變數。

     d. 針對變數（索引鍵），輸入您的用戶端 ID。在值中，輸入您的用戶端秘密。

     e. 選擇、儲存。

管理環境秘密

隨著 Amplify Gen 2 的推出，環境秘密的工作流程經過簡化，以集中管理 Amplify 主控台中的秘密和環境變數。如需設定和存取 Amplify Gen 2 應用程式秘密的說明，請參閱 Amplify 文件中的秘密和環境差異。

Gen 1 應用程式的環境秘密類似於環境變數，但它們是可以加密的 AWS Systems Manager 參數存放區金鑰值對。某些值必須加密，例如 Amplify 的 Sign in with Apple 私有金鑰。

使用 AWS Systems Manager 設定 Amplify Gen 1 應用程式的環境秘密

使用以下指示，使用 AWS Systems Manager 主控台設定 Gen 1 Amplify 應用程式的環境秘密。

設定環境秘密

1. 登入 AWS 管理主控台 並開啟 AWS Systems Manager 主控台。
2. 在導覽窗格中，選擇應用程式管理，然後選擇參數存放區。
3. 在 AWS Systems Manager 參數存放區頁面上，選擇建立參數。
4. 在建立參數頁面上的參數詳細資訊區段中，執行下列動作：
   a. 針對名稱，輸入格式為 的參數**`/amplify/{your_app_id}/{your_backend_environment_name}/{your_parameter_name}`**。
   b. 針對類型，選擇 SecureString。
   c. 針對 KMS 金鑰來源，選擇我的目前帳戶以使用您帳戶的預設金鑰。
   d. 在值中，輸入要加密的秘密值。
5. 選擇建立參數。

> **Note**
>
> Amplify 只能存取`/amplify/{your_app_id}/{your_backend_environment_name}`特定環境建置之 下的金鑰。您必須指定預設值 AWS KMS key ，以允許 Amplify 解密該值。

存取 Gen 1 應用程式的環境秘密

Gen 1 應用程式的環境秘密會以 JSON 字串`process.env.secrets`形式存放在 中。

Amplify 環境秘密參考

以 格式指定 Systems Manager 參數`/amplify/{your_app_id}/{your_backend_environment_name}/AMPLIFY_SIWA_CLIENT_ID`。

您可以使用 Amplify 主控台預設可存取的下列環境秘密。

變數名稱	Description	範例值
AMPLIFY_SIWA_CLIENT_ID	使用 Apple 用戶端 ID 登入	`com.yourapp.auth`
AMPLIFY_SIWA_TEAM_ID	使用 Apple 團隊 ID 登入	`ABCD123`
AMPLIFY_SIWA_KEY_ID	使用 Apple 金鑰 ID 登入	`ABCD123`
AMPLIFY_SIWA_PRIVATE_KEY	使用 Apple 登入私有金鑰	-----從私有金鑰開始----- ****..... -----END 私有金鑰-----

設定 Amplify 應用程式的自訂標頭

自訂 HTTP 標頭可讓您為每個 HTTP 回應指定標頭。回應標頭可用在偵錯、安全和資訊方面。您可以在 Amplify 主控台中指定標頭，或下載並編輯應用程式`customHttp.yml`的檔案，並將其儲存在專案的根目錄中。如需詳細程序，請參閱設定自訂標頭。

先前，自訂 HTTP 標頭是透過在 Amplify 主控台中編輯建置規格 (buildspec)，或下載並更新`amplify.yml`檔案，並將其儲存在專案的根目錄中，為應用程式指定。我們強烈建議將以此方式指定的自訂標頭從 buildspec 和 `amplify.yml` 檔案遷移。如需說明，請參閱從建置規格遷移自訂標頭和 amplify.yml。

主題

- 自訂標頭 YAML 參考
- 設定自訂標頭
- 從建置規格遷移自訂標頭和 amplify.yml
- Monorepo 自訂標頭需求

自訂標頭 YAML 參考

使用下列 YAML 格式指定自訂標頭：

```
customHeaders:
  - pattern: '*.json'
    headers:
    - key: 'custom-header-name-1'
      value: 'custom-header-value-1'
    - key: 'custom-header-name-2'
      value: 'custom-header-value-2'
  - pattern: '/path/*'
    headers:
    - key: 'custom-header-name-1'
      value: 'custom-header-value-2'
```

對於單儲存庫，請使用下列 YAML 格式：

```
applications:
```

```
  - appRoot: app1
    customHeaders:
    - pattern: '**/*'
      headers:
      - key: 'custom-header-name-1'
        value: 'custom-header-value-1'
  - appRoot: app2
    customHeaders:
    - pattern: '/path/*.json'
      headers:
      - key: 'custom-header-name-2'
        value: 'custom-header-value-2'
```

當您將自訂標頭新增至應用程式時，您將為下列項目指定自己的值：

pattern

自訂標頭會套用至符合模式的所有 URL 檔案路徑。

標頭

定義符合檔案模式的標頭。

金鑰

自訂標頭的名稱。

value

自訂標頭的值。

若要進一步了解 HTTP 標頭，請參閱 Mozilla 的 HTTP 標頭清單。

設定自訂標頭

有兩種方式可以指定 Amplify 應用程式的自訂 HTTP 標頭。您可以在 Amplify 主控台中指定標頭，也可以透過下載和編輯應用程式`customHttp.yml`檔案並將其儲存在專案的根目錄中來指定標頭。

設定應用程式的自訂標頭，並將其儲存在主控台中

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。

2. 選擇要為其設定自訂標頭的應用程式。
3. 在導覽窗格中，選擇託管，然後選擇自訂標頭。
4. 在自訂標頭頁面上，選擇編輯。
5. 在編輯自訂標頭視窗中，使用自訂標頭 YAML 格式輸入自訂標頭的資訊。
   a. 針對 `pattern`，輸入要比對的模式。
   b. 針對 `key`，輸入自訂標頭的名稱。
   c. 針對 `value`，輸入自訂標頭的值。
6. 選擇儲存。
7. 重新部署應用程式以套用新的自訂標頭。
   - 對於 CI/CD 應用程式，導覽至要部署的分支，然後選擇重新部署此版本。您也可以從 Git 儲存庫執行新的組建。
   - 對於手動部署應用程式，請在 Amplify 主控台中再次部署應用程式。

設定應用程式的自訂標頭，並將其儲存在儲存庫的根目錄中

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇要為其設定自訂標頭的應用程式。
3. 在導覽窗格中，選擇託管，然後選擇自訂標頭。
4. 在自訂標頭頁面上，選擇下載 YML。
5. 在您選擇的程式碼編輯器中開啟下載`customHttp.yml`的檔案，並使用自訂標頭 YAML 格式輸入自訂標頭的資訊。
   a. 針對 `pattern`，輸入要比對的模式。
   b. 針對 `key`，輸入自訂標頭的名稱。
   c. 針對 `value`，輸入自訂標頭的值。
6. 將編輯`customHttp.yml`的檔案儲存在專案的根目錄中。如果您使用的是單儲存庫，請將`customHttp.yml`檔案儲存在儲存庫的根目錄中。
7. 重新部署應用程式以套用新的自訂標頭。
   - 針對 CI/CD 應用程式，請從包含新`customHttp.yml`檔案的 Git 儲存庫執行新組建。
   - 對於手動部署應用程式，請在 Amplify 主控台中再次部署應用程式，並將新`customHttp.yml`檔案包含您上傳的成品。

> **Note**
>
> 在customHttp.yml檔案中設定並在應用程式的根目錄中部署的自訂標頭會覆寫 Amplify 主控台自訂標頭區段中定義的自訂標頭。

安全性自訂標頭範例

自訂安全標頭可強制執行 HTTPS、防止 XSS 攻擊，以及保護您的瀏覽器免於點擊劫持。使用下列 YAML 語法，將自訂安全標頭套用至您的應用程式。

```
customHeaders:
  - pattern: '**'
    headers:
      - key: 'Strict-Transport-Security'
        value: 'max-age=31536000; includeSubDomains'
      - key: 'X-Frame-Options'
        value: 'SAMEORIGIN'
      - key: 'X-XSS-Protection'
        value: '1; mode=block'
      - key: 'X-Content-Type-Options'
        value: 'nosniff'
      - key: 'Content-Security-Policy'
        value: "default-src 'self'"
```

設定快取控制自訂標頭

使用 Amplify 託管的應用程式會遵守原始伺服器傳送的Cache-Control標頭，除非您使用您定義的自訂標頭來覆寫它們。Amplify 只會為具有200 OK狀態碼的成功回應套用快取控制自訂標頭。這可防止錯誤回應被快取，並提供給提出相同請求的其他使用者。

您可以手動調整 s-maxage 指令，以進一步控制應用程式的效能和部署可用性。例如，若要增加內容在邊緣保持快取的時間長度，您可以透過將值更新s-maxage為超過預設 600 秒 (10 分鐘） 來手動增加存留時間 (TTL)。

若要指定 的自訂值s-maxage，請使用下列 YAML 格式。此範例會將相關聯的內容在邊緣快取 3600 秒 （ 一小時 ）。

```
customHeaders:
  - pattern: '/img/*'
```

```
    headers:
      - key: 'Cache-Control'
        value: 's-maxage=3600'
```

如需使用 標頭控制應用程式效能的詳細資訊，請參閱 使用 Cache-Control 標頭來提高應用程式效能。

從建置規格遷移自訂標頭和 amplify.yml

先前，自訂 HTTP 標頭是透過在 Amplify 主控台中編輯建置規格，或下載並更新`amplify.yml`檔案，並將其儲存在專案的根目錄中，為應用程式指定。強烈建議您從建置規格和 `amplify.yml` 檔案遷移自訂標頭。

在 Amplify 主控台的自訂標頭區段中，或透過下載和編輯`customHttp.yml`檔案來指定您的自訂標頭。

遷移 Amplify 主控台中存放的自訂標頭

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇要在其中執行自訂標頭遷移的應用程式。
3. 在導覽窗格中，選擇託管、建置設定。在應用程式建置規格區段中，您可以檢閱應用程式的 buildspec。
4. 選擇下載以儲存目前 buildspec 的副本。如果您需要復原任何設定，您可以稍後參考此副本。
5. 下載完成時，請選擇編輯。
6. 請記下 檔案中的自訂標頭資訊，因為您稍後會在步驟 9 中使用它。在編輯視窗中，從 檔案刪除任何自訂標頭，然後選擇儲存。
7. 在導覽窗格中，選擇託管、自訂標頭。
8. 在自訂標頭頁面上，選擇編輯。
9. 在編輯自訂標頭視窗中，輸入您在步驟 6 中刪除的自訂標頭資訊。
10. 選擇儲存。
11. 重新部署您希望套用新自訂標頭的任何分支。

將自訂標頭從 amplify.yml 遷移至 customHttp.yml

1. 導覽至目前部署在應用程式根目錄中`amplify.yml`的檔案。
2. 在您選擇的程式碼編輯器`amplify.yml`中開啟 。

3. 請記下 檔案中的自訂標頭資訊，因為您稍後會在步驟 8 中使用它。刪除 檔案中的自訂標頭。儲存並關閉檔案。
4. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
5. 選擇要為其設定自訂標頭的應用程式。
6. 在導覽窗格中，選擇託管、自訂標頭。
7. 在自訂標頭頁面上，選擇下載。
8. 在您選擇的程式碼編輯器中開啟下載`customHttp.yml`的檔案，並輸入您在步驟 3 `amplify.yml`中刪除的自訂標頭資訊。
9. 將編輯`customHttp.yml`的檔案儲存在專案的根目錄中。如果您使用的是單儲存庫，請將檔案儲存在儲存庫的根目錄中。
10. 重新部署應用程式以套用新的自訂標頭。
    - 針對 CI/CD 應用程式，請從包含新`customHttp.yml`檔案的 Git 儲存庫執行新組建。
    - 對於手動部署應用程式，請在 Amplify 主控台中再次部署應用程式，並包含含有您上傳成品的新`customHttp.yml`檔案。

> **Note**
>
> 在 `customHttp.yml` 檔案中設定並在應用程式的根目錄中部署的自訂標頭會覆寫 Amplify 主控台自訂標頭區段中定義的自訂標頭。

Monorepo 自訂標頭需求

當您在 monorepo 中為應用程式指定自訂標頭時，請注意下列設定要求：

- 單一儲存庫有特定的 YAML 格式。如需正確的語法，請參閱 自訂標頭 YAML 參考。
- 您可以使用 Amplify 主控台的自訂標頭區段，在單一儲存庫中指定應用程式的自訂標頭。您必須重新部署應用程式，才能套用新的自訂標頭。
- 除了使用 主控台之外，您也可以在 `customHttp.yml` 檔案的 monorepo 中指定應用程式的自訂標頭。您必須在儲存庫的根目錄中儲存`customHttp.yml`檔案，然後重新部署應用程式以套用新的自訂標頭。`customHttp.yml` 檔案中指定的自訂標頭會覆寫使用 Amplify 主控台的自訂標頭區段指定的任何自訂標頭。

管理應用程式的快取組態

Amplify 使用 Amazon CloudFront 來管理託管應用程式的快取組態。快取組態會套用至每個應用程式，以最佳化最佳效能。

2024 年 8 月 13 日，Amplify 發佈了應用程式快取效率的改進。如需詳細資訊，請參閱 CDN 快取改進，以透過 AWS Amplify 託管提升應用程式效能。

下表摘要說明 Amplify 支援快取改進發行前後的特定快取行為。

快取行為	先前的支援	使用快取改進
您可以在 Amplify 主控台或`customHeaders.yaml` 檔案中新增應用程式的自訂標頭。您可以覆寫的其中一個標頭是 `Cache-Control` 。如需詳細資訊，請參閱設定 Amplify 應用程式的自訂標頭。	是	是
Amplify 遵守您在`customHeaders.yaml` 檔案中定義的`Cache-Control` 標頭，它們優先於 Amplify 的預設快取設定。	是	是
Amplify 會遵守應用程式架構中為動態路由設定的`Cache-Control` 標頭（例如 Next.js SSR 路由）。如果在應用程式的 `customHeaders.yaml` 檔案中設定`Cache-Control` 標頭，這會優先於 `next.config.js` 檔案中的設定。	是	是

快取行為	先前的支援	使用快取改進
每個新的 CI/CD 應用程式部署都會清除快取。	是	是
您可以開啟應用程式的效能模式。	是	否 Amplify 主控台不再提供效能模式設定。不過，您可以建立設定 `s-maxage`指令的`Cache-Control` 標頭。如需說明，請參閱使用 Cache-Control 標頭來提高應用程式效能。

下表列出特定快取設定的預設值變更。

快取設定	先前的預設值	具有快取改進的預設值
靜態資產的快取持續時間	兩秒	一年
反向代理回應的快取持續時間	兩秒	零秒 （無快取）
最長存留時間 (TTL)	十分鐘	一年

如需有關 Amplify 如何決定要套用至應用程式的快取組態的詳細資訊，以及管理快取金鑰組態的指示，請參閱下列主題。

主題

- Amplify 如何將快取組態套用至應用程式
- 管理快取金鑰 Cookie
- 使用 Cache-Control 標頭來提高應用程式效能

Amplify 如何將快取組態套用至應用程式

若要管理應用程式的快取，Amplify 會透過檢查應用程式的平台類型和重寫規則，來判斷服務的內容類型。對於`Compute`應用程式，Amplify 也會檢查部署資訊清單中的路由規則。

> **Note**
>
> 應用程式平台類型由 Amplify Hosting 在部署期間設定。SSG （靜態） 應用程式設定為平台類型 `WEB`。SSR 應用程式 (Next.js 12 或更新版本） 設定為平台類型 。 `WEB_COMPUTE`

Amplify 會識別以下四種類型的內容，並套用指定的受管快取政策。

靜態

從具有 `WEB`平台的應用程式提供的內容，或`WEB_COMPUTE`應用程式中的靜態路由。

此內容使用Amplify-StaticContent快取政策。

映像最佳化

應用程式中`ImageOptimization`路由提供的影像`WEB_COMPUTE`。

此內容使用Amplify-ImageOptimization快取政策。

運算

`WEB_COMPUTE` 應用程式中`Compute`路由提供的內容。這包括所有伺服器端轉譯 (SSR) 內容。

此內容會根據 Amplify 上設定的 值`cacheConfig.type`，使用 Amplify-Default或 Amplify-DefaultNoCookies快取政策App。

反向 Proxy

符合反向代理重寫自訂規則之路徑提供的內容。如需建立此自訂規則的詳細資訊，請參閱使用重新導向章節反向代理重寫中的 。

此內容會根據 Amplify 上設定的 值`cacheConfig.type`，使用 Amplify-Default或 Amplify-DefaultNoCookies快取政策App。

了解 Amplify 的受管快取政策

Amplify 使用以下預先定義的受管快取政策來最佳化託管應用程式的預設快取組態。

- Amplify-Default
- Amplify-DefaultNoCookies
- Amplify-ImageOptimization

- Amplify-StaticContent

Amplify-Default 受管快取政策設定

在 CloudFront 主控台中檢視此政策

此政策是專為與 AWS Amplify Web 應用程式的原始伺服器搭配使用而設計。

此政策包括下列設定：

- 最小 TTL：0 秒
- 最大 TTL：31536000 秒（一年）
- 預設 TTL：0 秒
- 包含在快取金鑰中的標頭：
  - `Authorization`
  - `Accept`
  - `CloudFront-Viewer-Country`
  - `Host`
- 快取金鑰中包含的 Cookie：所有 Cookie 都包含在內。
- 快取金鑰中包含的查詢字串：包含所有查詢字串。
- 快取壓縮物件設定：Gzip 和 Brotli 已啟用。

Amplify-DefaultNoCookies 受管快取政策設定

在 CloudFront 主控台中檢視此政策

此政策是專為與 AWS Amplify Web 應用程式的原始伺服器搭配使用而設計。

此政策包括下列設定：

- 最小 TTL：0 秒
- 最大 TTL：31536000 秒（一年）
- 預設 TTL：0 秒
- 包含在快取金鑰中的標頭：
  - `Authorization`

  - `Accept`
  - `CloudFront-Viewer-Country`
  - `Host`
- 快取金鑰中包含的 Cookie：不包含 Cookie。
- 快取金鑰中包含的查詢字串：包含所有查詢字串。
- 快取壓縮物件設定：Gzip 和 Brotli 已啟用。

Amplify-ImageOptimization 受管快取政策設定

在 CloudFront 主控台中檢視此政策

此政策是專為與 AWS Amplify Web 應用程式的原始伺服器搭配使用而設計。

此政策包括下列設定：

- 最小 TTL：0 秒
- TTL 上限：31536000 秒（一年）
- 預設 TTL：0 秒
- 包含在快取金鑰中的標頭：
  - `Authorization`
  - `Accept`
  - `Host`
- 快取金鑰中包含的 Cookie：不包含 Cookie。
- 快取金鑰中包含的查詢字串：包含所有查詢字串。
- 快取壓縮物件設定：Gzip 和 Brotli 已啟用。

Amplify-StaticContent 受管快取政策設定

在 CloudFront 主控台中檢視此政策

此政策是專為與 AWS Amplify Web 應用程式的原始伺服器搭配使用而設計。

此政策包括下列設定：

- 最小 TTL：0 秒

- 最大 TTL：31536000 秒（一年）
- 預設 TTL：0 秒
- 包含在快取金鑰中的標頭：
  - `Authorization`
  - `Host`
- 快取金鑰中包含的 Cookie：不包含 Cookie。
- 快取金鑰中包含的查詢字串：不包含任何查詢字串。
- 快取壓縮物件設定：Gzip 和 Brotli 已啟用。

管理快取金鑰 Cookie

當您將應用程式部署到 Amplify 時，您可以選擇是否要在快取金鑰中包含或排除 Cookie。在 Amplify 主控台中，使用快取金鑰設定切換，在自訂標頭和快取頁面上指定此設定。 如需說明，請參閱從快取金鑰中包含或排除 Cookie。

在快取金鑰中包含 Cookie

透過此設定，Amplify 會根據服務的內容類型，自動為您的應用程式選擇最佳快取組態。您必須明確選擇此快取組態類型。

如果您使用 SDKs或 AWS CLI，此設定會對應至`AMPLIFY_MANAGED`使用 `CreateApp`或 UpdateApp API `cacheConfig.type`將 設定為 。 APIs

從快取金鑰排除 Cookie

這是預設快取組態。此快取組態類似於`AMPLIFY_MANAGED`組態，但從快取金鑰排除所有 Cookie。

選擇從快取金鑰中排除 Cookie 可能會導致更好的快取效能。不過，在您選擇此快取組態之前，請務必考慮您的應用程式是否使用 Cookie 來提供動態內容。

如果您使用 SDKs或 AWS CLI，此設定會對應至`AMPLIFY_MANAGED_NO_COOKIES`使用 `CreateApp`或 UpdateApp API `cacheConfig.type`將 設定為 。 APIs

如需快取金鑰的詳細資訊，請參閱《Amazon CloudFront 開發人員指南》中的了解快取金鑰。

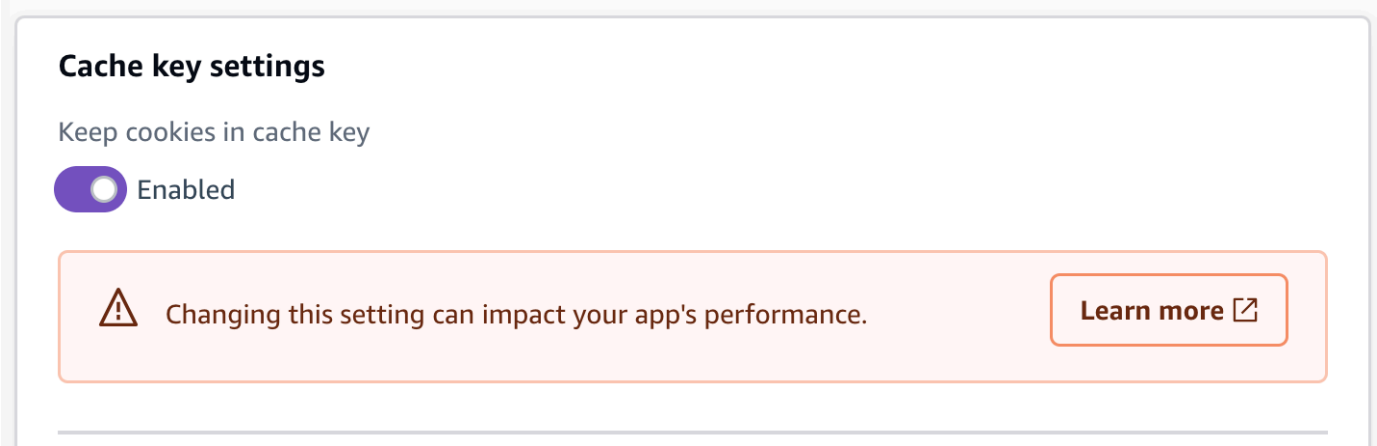
從快取金鑰中包含或排除 Cookie

您可以在 Amplify 主控台、 SDKs 或 中設定應用程式的快取金鑰 Cookie 組態 AWS CLI。

當您使用 Amplify 主控台部署新應用程式時，請使用下列程序指定是否要從快取金鑰包含或排除 Cookie。

在將應用程式部署至 Amplify 時設定快取金鑰 Cookie 組態

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 在所有應用程式頁面上，選擇建立新應用程式。
3. 在開始使用 Amplify 建置頁面上，選擇您的 Git 儲存庫提供者，然後選擇下一步。
4. 在新增儲存庫分支頁面上，執行下列動作：
   a. 選取要連線的儲存庫名稱。
   b. 選取要連線的儲存庫分支名稱。
   c. 選擇下一步。
5. 如果應用程式需要 IAM 服務角色，您可以允許 Amplify 託管運算自動為您建立服務角色，也可以指定您已建立的角色。
   - 若要允許 Amplify 自動建立角色並將其連接至您的應用程式：
     - 選擇建立並使用新的服務角色。
   - 若要連接您先前建立的服務角色：
     a. 選擇使用現有的服務角色。
     b. 從清單中選擇要使用的角色。
6. 選擇進階設定，然後尋找快取金鑰設定區段。
7. 選擇將 Cookie 保留在快取金鑰中或從快取金鑰中移除 Cookie。下列螢幕擷取畫面顯示 主控台中的快取金鑰設定切換。



8. 選擇下一步。

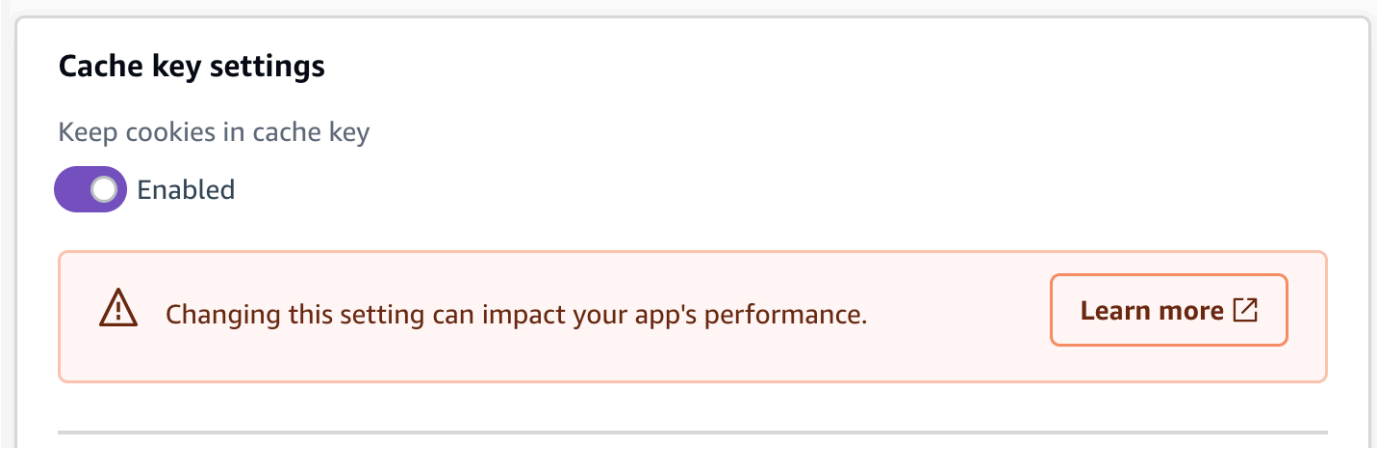
9. 在檢閱頁面上，選擇儲存並部署。

變更應用程式的快取金鑰 Cookie 組態

您可以變更已部署至 Amplify 之應用程式的快取金鑰 Cookie 組態。使用下列程序來變更是否要使用 Amplify 主控台從應用程式的快取金鑰中包含或排除 Cookie。

變更已部署應用程式的快取金鑰 Cookie 組態

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 在所有應用程式頁面上，選擇您要更新的應用程式。
3. 在導覽窗格中，選擇託管，然後選擇自訂標頭和快取。
4. 在自訂標頭和快取頁面上，找到快取金鑰設定區段，然後選擇編輯。
5. 選擇將 Cookie 保留在快取金鑰中或從快取金鑰中移除 Cookie。下列螢幕擷取畫面顯示 主控台中的快取金鑰設定切換。



6. 選擇儲存。

使用 Cache-Control 標頭來提高應用程式效能

Amplify 的預設託管架構可最佳化託管效能和部署可用性之間的平衡。對於大多數客戶，我們建議您使用預設架構。

如果您需要更精確地控制應用程式的效能，您可以手動設定 HTTP `Cache-Control`標頭，讓內容在內容交付網路 (CDN) 邊緣快取更長的間隔，以最佳化託管效能。

HTTP `Cache-Control`標頭`max-age`和 `s-maxage` 指令會影響應用程式的內容快取持續時間。`max-age` 指令會告知瀏覽器，您希望內容在快取中保留多久（以秒為單位），再從原始伺服器重新整理內

容。s-maxage 指令會覆寫max-age並可讓您指定要內容保留在 CDN 邊緣多久（以秒為單位），再從原始伺服器重新整理內容。

使用 Amplify 託管的應用程式會遵守原始伺服器傳送的Cache-Control標頭，除非您使用您定義的自訂標頭來覆寫它們。Amplify 只會為具有200 OK狀態碼的成功回應套用Cache-Control自訂標頭。這可防止錯誤回應被快取，並提供給提出相同請求的其他使用者。

您可以手動調整 s-maxage 指令，以進一步控制應用程式的效能和部署可用性。例如，若要變更內容在邊緣保持快取的時間長度，您可以透過將 更新s-maxage為預設 31536000 秒（一年）以外的值，手動設定存留時間 (TTL)。

您可以在 Amplify 主控台的自訂標頭區段中定義應用程式的自訂標頭。如需 YAML 格式的範例，請參閱 設定快取控制自訂標頭。

使用下列程序來設定 s-maxage指令，讓內容在 CDN 邊緣快取 24 小時。

設定自訂Cache-Control標頭

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇要為其設定自訂標頭的應用程式。
3. 在導覽窗格中，選擇託管、自訂標頭。
4. 在自訂標頭頁面上，選擇編輯。
5. 在編輯自訂標頭視窗中，輸入自訂標頭的資訊，如下所示：
   a. 對於 pattern，輸入 **`**/*`** 表示所有路徑。
   b. 針對 key，請輸入 **`Cache-Control`**。
   c. 針對 value，請輸入 **`s-maxage=86400`**。
6. 選擇儲存。
7. 重新部署應用程式以套用新的自訂標頭。

Amplify 部署的偏斜保護

Amplify 應用程式可以使用部署扭曲保護，以消除 Web 應用程式中用戶端和伺服器之間的版本扭曲問題。當您將偏斜保護套用至 Amplify 應用程式時，無論部署何時發生，都可以確保您的用戶端一律與正確的伺服器端資產版本互動。

版本扭曲是 Web 開發人員的常見挑戰。當 Web 瀏覽器執行過時版本的應用程式，且伺服器執行新的應用程式時，就會發生這種情況。此差異可能會導致無法預測的行為、錯誤，以及應用程式使用者的降級體驗。Amplify 部署扭曲保護功能會將在 Web 瀏覽器上執行的用戶端釘選至特定部署。這可確保 Amplify 始終為該特定部署提供資產，保持用戶端和伺服器同步。

Amplify 的扭曲保護功能可在您發佈新部署時減少應用程式使用者的錯誤。它也可以透過減少管理回溯和轉送相容性問題所花費的時間來改善開發人員體驗。

偏斜保護功能詳細資訊：

支援的應用程式類型

> 您可以將偏斜保護新增至使用 Amplify 支援的任何架構建立的靜態和 SSR 應用程式。您可以從 Git 儲存庫或手動部署部署應用程式。
>
> 您無法將偏斜保護新增至部署到`WEB_DYNAMIC`平台的應用程式 (Next.js 11 版或更早版本）。

持續時間

> 對於靜態應用程式，Amplify 提供一週的部署。對於 SSR 應用程式，我們保證最多可提供八個先前部署的偏斜保護。

Cost

> 將偏斜保護新增至應用程式無需額外費用。

效能考量

> 為應用程式啟用偏斜保護時，Amplify 必須更新其 CDN 快取組態。因此，您應該預期在啟用扭曲保護後，第一次部署最多需要十分鐘。

主題

- 設定 Amplify 應用程式的部署扭曲保護
- 扭曲保護的運作方式

設定 Amplify 應用程式的部署扭曲保護

您可以使用 Amplify 主控台 AWS Command Line Interface、 或 SDKs 來新增或移除應用程式的部署扭曲保護。此功能會套用至分支層級。只有在分支啟用偏斜保護之後進行的新部署才會受到偏斜保護。

若要使用 或 SDKs 新增 AWS CLI 或移除部署扭曲保護，請使用 `CreateBranch.enableSkewProtection`和 `UpdateBranch.enableSkewProtection` 欄位。如需詳細資訊，請參閱 Amplify API 參考文件中的 CreateBranch 和 UpdateBranch。

如果您想要移除特定部署，使其不再提供服務，請使用 `DeleteJob` API。如需詳細資訊，請參閱 Amplify API 參考文件中的 DeleteJob。

目前，您只能在已部署至 Amplify Hosting 的應用程式上啟用偏斜保護。使用以下指示，使用 Amplify 主控台將偏斜保護新增至分支。

啟用 Amplify 應用程式分支的偏斜保護

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/amplify/ 開啟 Amplify 主控台。
2. 在所有應用程式頁面上，選擇要啟用扭曲保護的應用程式名稱。
3. 在導覽窗格中，選擇應用程式設定，然後選擇分支設定。
4. 在分支區段中，選擇要更新的分支名稱。
5. 在動作功能表中，選擇啟用偏斜保護。
6. 在確認視窗中，選擇確認。分支現在已啟用偏斜保護。
7. 重新部署您的應用程式分支。只有啟用偏斜保護之後進行的部署才會受到偏斜保護。

使用以下指示，使用 Amplify 主控台從應用程式分支移除偏斜保護。

從 Amplify 應用程式分支移除偏斜保護

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/amplify/ 開啟 Amplify 主控台。
2. 在所有應用程式頁面上，選擇要移除扭曲保護的已部署應用程式名稱。
3. 在導覽窗格中，選擇應用程式設定，然後選擇分支設定。
4. 在分支區段中，選擇要更新的分支名稱。
5. 在動作功能表中，選擇停用偏斜保護。分支的偏斜保護現在已停用，而且只會提供最新的內容。

扭曲保護的運作方式

在大多數情況下，_dpl Cookie 的預設行為將滿足您的扭曲保護需求。不過，在下列進階案例中，使用 X-Amplify-Dpl標頭和dpl查詢參數更能啟用偏斜保護。

- 同時在多個瀏覽器索引標籤中載入您的網站
- 使用 服務工作者

Amplify 在決定要提供給用戶端的內容時，會依下列順序評估傳入請求：

1. **X-Amplify-Dpl** 標頭 – 應用程式可以使用此標頭將請求導向特定 Amplify 部署。您可以使用 的值來設定此請求標頭process.env.AWS_AMPLIFY_DEPLOYMENT_ID。
2. **dpl** 查詢參數 – Next.js 應用程式會自動為對指紋資產 (.js 和 .css 檔案） 的請求設定 _dpl 查詢參數。
3. _dpl Cookie – 這是所有受扭曲保護的應用程式的預設值。對於特定瀏覽器，每個與網域互動的瀏覽器索引標籤或執行個體都會傳送相同的 Cookie。

   請注意，如果不同的瀏覽器標籤載入了不同版本的網站，則所有標籤都會共用 _dpl Cookie。在這種情況下，無法使用 _dpl Cookie 達到完全扭曲保護，您應該考慮使用 X-Amplify-Dpl標頭進行扭曲保護。

X-Amplify-Dpl 標頭範例

下列範例示範透過 X-Amplify-Dpl標頭存取扭曲保護的 Next.js SSR 頁面程式碼。頁面會根據其中一個 API 路由轉譯其內容。要提供給 api 路由的部署是使用 X-Amplify-Dpl標頭指定，該標頭設定為 的值process.env.AWS_AMPLIFY_DEPLOYMENT_ID。

```
import { useEffect, useState } from 'react';

export default function MyPage({deploymentId}) {
    const [data, setData] = useState(null);

    useEffect(() => {
        fetch('/api/hello', {
            headers: {
                'X-Amplify-Dpl': process.env.AWS_AMPLIFY_DEPLOYMENT_ID
            },
        })
```

```
        .then(res => res.json())
        .then(data => setData(data))
        .catch(error => console.error("error", error))
    }, []);

    return <div>
        {data ? JSON.stringify(data) : "Loading ... " }
    </div>
}
```

監控 Amplify 應用程式

AWS Amplify 提供下列功能來監控您的託管應用程式：

- CloudWatch 指標 – Amplify 透過 Amazon CloudWatch 發出指標，您可以用來監控應用程式的流量、錯誤、資料傳輸和延遲。
- 存取日誌 – Amplify 提供存取日誌，其中包含對應用程式提出之請求的詳細資訊。
- CloudTrail 記錄 – Amplify 與 整合 AWS CloudTrail ，可提供使用者、角色或 Amplify 中 AWS 服務所採取之動作的記錄。您可以在 CloudTrail 主控台中檢視這些事件。

主題

- 使用 Amazon CloudWatch 監控 Amplify 應用程式
- 擷取和分析 Amplify 應用程式的存取日誌
- 使用 記錄 Amplify API 呼叫 AWS CloudTrail

使用 Amazon CloudWatch 監控 Amplify 應用程式

AWS Amplify 與 Amazon CloudWatch 整合，可讓您近乎即時地監控 Amplify 應用程式的指標，並建立警示，在指標超過您設定的閾值時傳送通知。如需 CloudWatch 服務運作方式的詳細資訊，請參閱 Amazon CloudWatch 使用者指南。

支援的 CloudWatch 指標

Amplify 在`AWS/AmplifyHosting`命名空間中支援七個 CloudWatch 指標，用於監控應用程式的流量、錯誤、資料傳輸、延遲和請求權杖。這些指標會每隔一分鐘彙總一次。CloudWatch 監控指標是免費的，不會計入 CloudWatch 服務配額。

下表說明每個支援的指標，並列出最相關的統計資料。並非所有可用的統計資料都適用於每個指標。

指標	Description
要求	您的應用程式收到的檢視器請求總數。 最相關的統計資料是 Sum。使用 Sum 統計資料來取得請求總數。

指標	Description
BytesDownloaded	檢視器針對 GET、 和 OPTIONS請求從應用程式（下載）傳輸的資料總量HEAD，以位元組為單位。 最相關的統計資料是 Sum。
BytesUploaded	任何請求傳輸到您的應用程式（上傳）的資料總量，以位元組為單位，包括 標頭。 Amplify 不會向您收取上傳到應用程式中的資料費用。 最相關的統計資料是 Sum。
4xxErrors	傳回 HTTP 狀態碼 400-499 範圍內錯誤的請求數目。 最相關的統計資料是 Sum。使用 Sum 統計資料來取得這些錯誤的總發生次數。
5xxErrors	傳回 HTTP 狀態碼 500-599 範圍內錯誤的請求數目。 最相關的統計資料是 Sum。使用 Sum 統計資料來取得這些錯誤的總發生次數。
延遲	第一個位元組的時間，以秒為單位。這是 Amplify Hosting 收到請求到將回應傳回至網路之間的總時間。這不包括回應到達檢視器裝置時所遇到的網路延遲。 最相關的統計資料為 Average、Maximum、Minimum、p10、p50、p95、p90和 p100。 使用 Average 統計資料來評估預期的延遲。

指標	Description
TokensConsumed	您的應用程式使用的請求字符。 Sum 統計資料代表總請求字符消耗量。您可以將此統計資料與目前的Request tokens per second服務配額進行比較，以判斷是否需要請求提高配額，以避免在未來的高流量事件期間發生潛在的限流。 Average 統計資料代表正常和尖峰時間的請求字符消耗。較高的字符使用量通常會導致更長的第一位元組時間 (TTFB)。因此，您可以在評估應用程式的延遲時使用此統計資料。如果您的延遲不佳，您可以改善下游 APIs以減少字符消耗，並避免字符消耗超過應用程式Request tokens per second的服務配額時可能發生的限流。 如需Request tokens per second服務配額的詳細資訊，請參閱 Amplify 託管服務配額。

Amplify 提供下列 CloudWatch 指標維度。

維度	Description
應用程式	指標資料由應用程式提供。
AWS 帳戶	指標資料會在 中的所有應用程式中提供 AWS 帳戶。

存取 CloudWatch 指標

您可以使用下列程序，直接從 Amplify 主控台存取 CloudWatch 指標。

> **Note**
>
> 您也可以存取 中的 CloudWatch 指標， AWS 管理主控台 網址為 https：//https://console.aws.amazon.com/cloudwatch/。

在 Amplify 主控台中存取指標

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇您要檢視指標的應用程式。
3. 在導覽窗格中，選擇監控，然後選擇指標。

建立 CloudWatch 警示

您可以在 Amplify 主控台中建立 CloudWatch 警示，在符合特定條件時傳送通知。警示會監看單一 CloudWatch 指標，並在指標超過指定數量的評估期間閾值時傳送 Amazon Simple Notification Service 通知。

您可以在 CloudWatch 主控台或使用 CloudWatch APIs 建立使用指標數學表達式的更進階警示。例如，您可以建立警示，在連續三個期間 的百分比4xxErrors超過 15% 時通知您。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的根據指標數學表達式建立 CloudWatch 警示。 Amazon CloudWatch

標準 CloudWatch 定價適用於警示。如需詳細資訊，請參閱 Amazon CloudWatch 定價。

使用下列程序在 Amplify 主控台中建立警示。

為 Amplify 指標建立 CloudWatch 警示

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇您要設定警示的應用程式。
3. 在導覽窗格中，選擇監控，然後選擇警示。
4. 在警示頁面上，選擇建立警示。
5. 在建立警示視窗中，設定您的警示，如下所示：

    a. 針對指標，從清單中選擇要監控的指標名稱。

    b. 在警示名稱中，輸入警示的有意義的名稱。例如，如果您正在監控請求，您可以命名警示 **`HighTraffic`**。名稱只能包含 ASCII 字元。

c. 對於設定通知，請執行下列其中一項操作：

- i. 選擇新增以設定新的 Amazon SNS 主題。
  ii. 針對電子郵件地址，輸入通知收件人的電子郵件地址。
  iii. 選擇新增電子郵件地址以新增其他收件人。
- i. 選擇現有以重複使用 Amazon SNS 主題。
  ii. 針對 SNS 主題，從清單中選擇現有 Amazon SNS 主題的名稱。

d. 對於每當指標的統計資料，請設定警示的條件，如下所示：

i. 指定指標是否必須大於或等於閾值。
ii. 指定閾值。
iii. 指定必須處於警示狀態才能叫用警示的連續評估期間數目。
iv. 指定評估期間的時間長度。

e. 選擇確認。

> **Note**
>
> 您指定的每個 Amazon SNS 收件人都會收到來自 AWS Notifications 的確認電子郵件。電子郵件包含收件人必須遵循的連結，以確認其訂閱並接收通知。

存取 SSR 應用程式的 CloudWatch Logs

Amplify 會將 SSR 執行時間的相關資訊傳送至 中的 Amazon CloudWatch Logs AWS 帳戶。當您將 SSR 應用程式部署到 Amplify 託管運算時，應用程式需要 Amplify 代表您呼叫其他 服務時擔任的 IAM 服務角色。您可以允許 Amplify 託管運算自動為您建立服務角色，也可以指定您已建立的角色。

如果您選擇允許 Amplify 為您建立 IAM 角色，該角色將已有建立 CloudWatch Logs 的許可。如果您建立自己的 IAM 角色，則需要將下列許可新增至政策，以允許 Amplify 存取 Amazon CloudWatch Logs。

```
logs:CreateLogStream
logs:CreateLogGroup
logs:DescribeLogGroups
logs:PutLogEvents
```

如需新增服務角色的詳細資訊，請參閱 新增具有部署後端資源許可的服務角色。如需部署伺服器端轉譯應用程式的詳細資訊，請參閱 使用 Amplify Hosting 部署伺服器端轉譯應用程式。

您可以在 CloudWatch 主控台或 Amplify 主控台中檢視 SSR 應用程式的 Amplify 託管運算日誌。使用下列指示在 Amplify 主控台中檢視日誌。

在 Amplify 主控台中檢視 SSR 應用程式的 CloudWatch 日誌

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇要檢視 CloudWatch 日誌的 SSR 應用程式。
3. 在導覽窗格中，選擇監控，然後選擇託管運算日誌。
4. 在託管運算日誌頁面上，搜尋並選取特定分支的 CloudWatch 日誌群組。

擷取和分析 Amplify 應用程式的存取日誌

Amplify 會儲存您在 Amplify 中託管的所有應用程式的存取日誌。存取日誌包含對託管應用程式提出之請求的相關資訊。Amplify 會保留應用程式的所有存取日誌，直到您刪除應用程式為止。應用程式的所有存取日誌都可以在 Amplify 主控台中使用。不過，存取日誌的每個個別請求僅限於您指定的兩週期間。

> **Warning**
>
> 請勿在 URLs 中包含秘密、登入資料或敏感資料做為路徑或查詢參數。這些值可在 Amplify 應用程式的存取日誌中以純文字顯示。

Amplify 絕不會在客戶之間重複使用 CloudFront 分佈。Amplify 會事先建立 CloudFront 分佈，這樣您就不必在部署新應用程式時等待 CloudFront 分佈建立。在這些分發指派給 Amplify 應用程式之前，他們可能會從機器人接收流量。不過，它們已設定為一律在指派之前回應為找不到。如果您應用程式的存取日誌包含建立應用程式前一段時間內的項目，這些項目會與此活動相關。

> **Important**
>
> 我們建議您使用日誌，了解內容請求的性質，而不是像完全考量所有請求。Amplify 會盡力提供存取日誌。在實際處理請求之後，才可能長時間交付特定請求的日誌項目，在極少數的情況下，有可能完全不會交付日誌項目。從存取日誌省略日誌項目時，存取日誌中的項目數量將與 AWS 帳單和用量報告中顯示的用量不符。

擷取應用程式的存取日誌

使用下列程序擷取 Amplify 應用程式的存取日誌。

檢視存取日誌

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇您要檢視其存取日誌的應用程式。
3. 在導覽窗格中，選擇監控，然後選擇存取日誌。
4. 選擇編輯時間範圍。
5. 在編輯時間範圍視窗中，執行下列動作。
   a. 針對開始日期，指定要擷取日誌之兩週間隔的第一天。
   b. 針對開始時間，選擇第一天開始日誌擷取的時間。
   c. 選擇確認。
6. Amplify 主控台會在存取日誌區段中顯示您指定時間範圍的日誌。選擇下載以 CSV 格式儲存日誌。

分析存取日誌

若要分析存取日誌，您可以將 CSV 檔案存放在 Amazon S3 儲存貯體中。分析存取日誌的一種方法是使用 Athena。Athena 是一種互動式查詢服務，可協助您分析 AWS 服務的資料。您可以遵循此處step-by-step說明來建立資料表。建立資料表後，您可以查詢資料，如下所示。

```
SELECT SUM(bytes) AS total_bytes
FROM logs
WHERE "date" BETWEEN DATE '2018-06-09' AND DATE '2018-06-11'
LIMIT 100;
```

使用 記錄 Amplify API 呼叫 AWS CloudTrail

AWS Amplify 已與 服務整合 AWS CloudTrail，此服務提供由使用者、角色或 Amplify 中的 AWS 服務所採取之動作的記錄。CloudTrail 會將 Amplify 的所有 API 呼叫擷取為事件。擷取的呼叫包括來自 Amplify 主控台的呼叫，以及對 Amplify API 操作的程式碼呼叫。如果您建立線索，則可以將 CloudTrail 事件持續交付至 Amazon S3 儲存貯體，包括 Amplify 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

斷向 Amplify 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 中的 Amplify 資訊

AWS 您的帳戶預設會啟用 CloudTrail。在 Amplify 中發生活動時，該活動會與事件歷史記錄中的其他 AWS 服務事件一起記錄在 CloudTrail 事件中。您可以檢視、搜尋和下載 AWS 帳戶的最新事件。如需詳細資訊，請參閱 AWS CloudTrail 使用者指南中的 使用 CloudTrail 事件歷史記錄檢視事件。

若要持續記錄您 AWS 帳戶中的事件，包括 Amplify 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，當您在主控台建立線索時，線索會套用到所有 AWS 區域。線索會記錄 AWS 分割區中所有區域的事件，並將日誌檔案傳送到您指定的 Amazon S3 儲存貯體。此外，您可以設定其他 AWS 服務，以進一步分析和處理 CloudTrail 日誌中所收集的事件資料。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》 中的下列主題：

- 為 AWS 您的帳戶建立追蹤
- CloudTrail 支援的服務和整合
- 設定 CloudTrail 的 Amazon SNS 通知
- 接收多個區域的 CloudTrail 日誌檔案和接收多個帳戶的 CloudTrail 日誌檔案

CloudTrail 會記錄所有 Amplify 操作，並記錄在AWS Amplify 主控台 API 參考、AWS Amplify Admin UI API 參考和 Amplify UI Builder API 參考中。例如，對 `CreateApp`、`DeleteApp` 和 `DeleteBackendEnvironment` 作業的呼叫都會在 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：

- 是否使用根或 AWS Identity and Access Management (IAM) 使用者登入資料提出請求。
- 是否使用角色或聯合身分使用者的臨時安全登入資料提出請求。
- 該請求是否由其他服務提出 AWS 。

如需詳細資訊，請參閱AWS CloudTrail 《 使用者指南》中的 CloudTrail userIdentity 元素。

了解 Amplify 日誌檔案項目

追蹤是一種組態，能讓事件以日誌檔案的形式交付到您指定的 Amazon S3 儲存貯體。CloudTrail 日誌檔案包含一或多個日誌專案。一個事件為任何來源提出的單一請求，並包含請求動作、請求的日期和時

間、請求參數等資訊。CloudTrail 日誌檔並非依公有 API 呼叫的堆疊追蹤排序，因此不會以任何特定順序出現。

下列範例顯示示範 AWS Amplify 主控台 API 參考ListApps操作的 CloudTrail 日誌項目。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::444455556666:user/Mary_Major",
        "accountId": "444455556666",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Mary_Major",
        "sessionContext": {
            "sessionIssuer": {},
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-01-12T05:48:10Z"
            }
        }
    },
    "eventTime": "2021-01-12T06:47:29Z",
    "eventSource": "amplify.amazonaws.com",
    "eventName": "ListApps",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.255",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.898
 Linux/4.9.230-0.1.ac.223.84.332.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.275-b01
 java/1.8.0_275 vendor/Oracle_Corporation",
    "requestParameters": {
        "maxResults": "100"
    },
    "responseElements": null,
    "requestID": "1c026d0b-3397-405a-95aa-aa43aexample",
    "eventID": "c5fca3fb-d148-4fa1-ba22-5fa63example",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "444455556666"
}
```

下列範例顯示示範 AWS Amplify Admin UI API 參考ListBackendJobs操作的 CloudTrail 日誌項目。


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::444455556666:user/Mary_Major",
        "accountId": "444455556666",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Mary_Major",
        "sessionContext": {
            "sessionIssuer": {},
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-01-13T00:47:25Z"
            }
        }
    },
    "eventTime": "2021-01-13T01:15:43Z",
    "eventSource": "amplifybackend.amazonaws.com",
    "eventName": "ListBackendJobs",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.255",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.898
Linux/4.9.230-0.1.ac.223.84.332.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.275-b01
java/1.8.0_275 vendor/Oracle_Corporation",
    "requestParameters": {
        "appId": "d23mv2oexample",
        "backendEnvironmentName": "staging"
    },
    "responseElements": {
        "jobs": [
            {
                "appId": "d23mv2oexample",
                "backendEnvironmentName": "staging",
                "jobId": "ed63e9b2-dd1b-4bf2-895b-3d5dcexample",
                "operation": "CreateBackendAuth",
                "status": "COMPLETED",
                "createTime": "1610499932490",
                "updateTime": "1610500140053"
            },
            {
```

```
                "appId": "d23mv2oexample",
                "backendEnvironmentName": "staging",
                "jobId": "06904b10-a795-49c1-92b7-185dfexample",
                "operation": "CreateBackend",
                "status": "COMPLETED",
                "createTime": "1610499657938",
                "updateTime": "1610499704458"
            }
        ],
        "appId": "d23mv2oexample",
        "backendEnvironmentName": "staging"
    },
    "requestID": "7adfabd6-98d5-4b11-bd39-c7deaexample",
    "eventID": "68769310-c96c-4789-a6bb-68b52example",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "444455556666"
}
```

搭配 Amplify 應用程式使用 IAM 角色

IAM 角色是具有特定許可的 IAM 身分。角色的許可決定身分可以和不可以執行的動作 AWS。您可以在中建立 IAM 角色 AWS 帳戶 ，並使用它們將許可委派給 Amplify 託管。若要進一步了解 角色，請參閱《IAM 使用者指南》中的 IAM 角色。

您可以使用下列類型的 IAM 角色，授予 Amplify Hosting 代表您執行動作所需的許可，或執行存取其他 AWS 資源的運算程式碼。

IAM 服務角色

Amplify 會擔任此角色來代表您執行動作。具有後端資源的應用程式需要此角色。

IAM SSR 運算角色

允許伺服器端轉譯 (SSR) 應用程式安全地存取特定 AWS 資源。

IAM SSR CloudWatch Logs 角色

當您部署 SSR 應用程式時，應用程式需要 Amplify 擔任的 IAM 服務角色，以允許 Amplify 存取 Amazon CloudWatch Logs。

主題

- 新增具有部署後端資源許可的服務角色
- 新增 SSR 運算角色以允許存取 AWS 資源
- 新增具有存取 CloudWatch Logs 許可的服務角色

新增具有部署後端資源許可的服務角色

Amplify 需要許可，才能使用您的前端部署後端資源。您會使用服務角色來達成此目標。服務角色是 AWS Identity and Access Management (IAM) 角色，提供 Amplify Hosting 代表您部署、建立和管理後端的許可。

當您建立新的應用程式需要 IAM 服務角色時，您可以允許 Amplify Hosting 為您自動建立服務角色，也可以選取您已建立的 IAM 角色。在本節中，您將了解如何建立具有帳戶管理許可的 Amplify 服務角色，並明確允許直接存取 Amplify 應用程式部署、建立和管理後端所需的資源。

在 IAM 主控台中建立 Amplify 服務角色

若要建立服務角色

1. 開啟 IAM 主控台，然後從左側導覽列中選擇角色，然後選擇建立角色。
2. 在選取信任的實體頁面中，選擇 AWS 服務。針對使用案例，選取 Amplify - 後端部署，然後選擇下一步。
3. 在 Add permissions (新增許可) 頁面上，選擇 Next (下一步)。
4. 在名稱、檢視和建立頁面上，針對角色名稱輸入有意義的名稱，例如 **AmplifyConsoleServiceRole-AmplifyRole**。
5. 接受所有預設值，然後選擇建立角色。
6. 返回 Amplify 主控台，將角色連接至您的應用程式。
   - 如果您正在部署新的應用程式，請執行下列動作：
     a. 重新整理服務角色的清單。
     b. 選取您剛建立的角色。在此範例中，它應該看起來像 AmplifyConsoleServiceRole-AmplifyRole。
     c. 選擇下一步，然後依照步驟完成應用程式部署。
   - 如果您有現有的應用程式，請執行下列動作：
     a. 在導覽窗格中，選擇應用程式設定，然後選擇 IAM 角色。
     b. 在 IAM 角色頁面的服務角色區段中，選擇編輯。
     c. 在服務角色頁面上，從服務角色清單中選取您剛建立的角色。
     d. 選擇儲存。
7. Amplify 現在具有為您的應用程式部署後端資源的許可。

編輯服務角色的信任政策，以防止混淆代理人

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。如需詳細資訊，請參閱預防跨服務混淆代理人。

目前，`Amplify-Backend Deployment`服務角色的預設信任政策會強制執行 `aws:SourceArn`和 `aws:SourceAccount`全域內容條件索引鍵，以防止混淆代理人。不過，如果您之前在帳戶中建立 `Amplify-Backend Deployment`角色，您可以更新角色的信任政策，以新增這些條件，以防止混淆代理人。

使用下列範例來限制存取您帳戶中的應用程式。將範例中的區域和應用程式 ID 取代為您自己的資訊。

```
"Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:amplify:us-east-1:123456789012:apps/*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
```

如需使用 編輯角色信任政策的說明 AWS 管理主控台，請參閱《IAM 使用者指南》中的修改角色（主控台）。

新增 SSR 運算角色以允許存取 AWS 資源

此整合可讓您將 IAM 角色指派給 Amplify SSR Compute 服務，以允許伺服器端轉譯 (SSR) 應用程式根據角色的許可安全地存取特定 AWS 資源。例如，您可以允許應用程式的 SSR 運算函數根據指派的 IAM 角色中定義的許可，安全地存取其他 AWS 服務或資源，例如 Amazon Bedrock 或 Amazon S3 儲存貯體。

IAM SSR 運算角色提供臨時登入資料，無需在環境變數中硬式編碼長期安全登入資料。使用 IAM SSR 運算角色符合授予最低權限許可以及盡可能使用短期憑證 AWS 的安全性最佳實務。

本節稍後的說明說明如何建立具有自訂許可的政策，並將政策連接至角色。建立角色時，您必須連接自訂信任政策，授予 Amplify 擔任角色的許可。如果未正確定義信任關係，您會在嘗試新增角色時收到錯誤。下列自訂信任政策會授予 Amplify 擔任角色的許可。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "amplify.amazonaws.com"
                ]
            },
```

```
            "Action": "sts:AssumeRole"
        }
    ]
}
```

您可以使用 Amplify 主控台、 AWS SDKs 或 ，將 中的 IAM 角色 AWS 帳戶 與現有的 SSR 應用程式建立關聯 AWS CLI。您連接的角色會自動與 Amplify SSR 運算服務建立關聯，授予它您指定存取其他 AWS 資源的許可。由於應用程式的需求會隨著時間而變更，因此您可以修改連接的 IAM 角色，而無需重新部署應用程式。這可提供彈性並減少應用程式停機時間。

> **Important**
>
> 您有責任設定您的應用程式以符合您的安全與合規目標。這包括管理您的 SSR 運算角色，該角色應設定為具有支援您的使用案例所需的最低許可集。如需詳細資訊，請參閱管理 IAM SSR 運算角色安全性。

在 IAM 主控台中建立 SSR 運算角色

在您將 IAM SSR 運算角色連接至 Amplify 應用程式之前，該角色必須已存在於您的 中 AWS 帳戶。在本節中，您將了解如何建立 IAM 政策，並將其連接至 Amplify 可以擔任的角色，以存取特定 AWS 資源。

我們建議您在建立 IAM 角色時，遵循授予最低權限許可的 AWS 最佳實務。IAM SSR 運算角色只能從 SSR 運算函數呼叫，因此應該只授予執行程式碼所需的許可。

您可以使用 AWS 管理主控台 AWS CLI或 SDKs 在 IAM 中建立政策。如需詳細資訊，請參閱《IAM 使用者指南》中的使用客戶受管政策定義自訂 IAM 許可。

下列指示示範如何使用 IAM 主控台建立 IAM 政策，以定義授予 Amplify Compute 服務的許可。

使用 IAM 主控台 JSON 政策編輯器建立政策

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在左側的導覽窗格中，選擇 Policies (政策)。
3. 選擇建立政策。
4. 在政策編輯器中，選擇 JSON 選項。
5. 輸入或貼上 JSON 政策文件。

6. 將許可新增至政策後，請選擇下一步。
7. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。檢視此政策中定義的許可，來查看您的政策所授予的許可。
8. 選擇 Create policy (建立政策) 儲存您的新政策。

建立政策後，請使用下列指示將政策連接至 IAM 角色。

建立將 Amplify 許可授予特定 AWS 資源的角色

1. 登入 AWS 管理主控台 並開啟位於 https：//https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在主控台的導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 選擇 Custom trust policy (自訂信任政策) 角色類型。
4. 在自訂信任政策區段中，輸入角色的自訂信任政策。角色信任政策是必要的，並定義您信任擔任該角色的委託人。

   複製並貼上下列信任政策，以授予 Amplify 服務擔任此角色的許可。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "amplify.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

5. 解決政策驗證期間產生的任何安全性警告、錯誤或一般性警告，然後選擇下一步。
6. 在新增許可頁面上，搜尋您在上一個程序中建立的政策名稱，並加以選取。然後選擇下一步。

7. 在 Role name (角色名稱) 中，輸入角色名稱。角色名稱在您的 中必須是唯一的 AWS 帳戶。它們不區分大小寫。例如，您無法建立名為 **PRODROLE** 和 **prodrole** 的角色。由於其他 AWS 資源可能會參考角色，因此您無法在建立角色之後編輯角色的名稱。
8. (選用) 在 Description (說明) 中，輸入新角色的說明。
9. (選用) 在步驟 1：選取信任的實體或步驟 2：新增許可區段中選擇編輯，以編輯角色的自訂政策和許可。
10. 檢閱角色，然後選擇建立角色。

將 IAM SSR 運算角色新增至 Amplify 應用程式

在 中建立 IAM 角色後 AWS 帳戶，您可以在 Amplify 主控台中將其與應用程式建立關聯。

在 Amplify 主控台中將 SSR 運算角色新增至應用程式

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/amplify/ 開啟 Amplify 主控台。
2. 在所有應用程式頁面上，選擇要新增運算角色的應用程式名稱。
3. 在導覽窗格中，選擇應用程式設定，然後選擇 IAM 角色。
4. 在運算角色區段中，選擇編輯。
5. 在預設角色清單中，搜尋您要連接的角色名稱，然後選取該角色。在此範例中，您可以選擇您在先前程序中建立的角色名稱。根據預設，您選擇的角色將與您應用程式的所有分支相關聯。

   如果角色的信任關係未正確定義，您會收到錯誤，而且無法新增角色。
6. （選用） 如果您的應用程式位於公有儲存庫中，並使用自動分支建立或已啟用提取請求的 Web 預覽，我們不建議使用應用程式層級角色。反之，僅將運算角色連接到需要存取特定資源的分支。若要覆寫預設應用程式層級行為並將角色連接至特定分支，請執行下列動作：

   a. 對於分支，選取要使用的分支名稱。

   b. 針對運算角色，選取要與分支建立關聯的角色名稱。
7. 選擇、儲存。

管理 IAM SSR 運算角色安全性

安全性是 AWS 與您之間的共同責任。您有責任設定您的應用程式以符合您的安全與合規目標。這包括管理您的 SSR 運算角色，該角色應設定為具有支援您的使用案例所需的最低許可集。您指定的 SSR

運算角色登入資料可在 SSR 函數的執行時間立即使用。如果您的 SSR 程式碼因錯誤或允許遠端程式碼執行 (RCE) 而刻意公開這些登入資料，未經授權的使用者可以存取 SSR 角色及其許可。

當公有儲存庫中的應用程式針對提取請求使用 SSR 運算角色和自動分支建立或 Web 預覽時，您需要謹慎管理哪些分支可以存取該角色。建議您不要使用應用程式層級角色。反之，您應該在分支層級連接運算角色。這可讓您僅將許可授予需要存取特定資源的分支。

如果您角色的登入資料公開，請採取下列動作來移除角色登入資料的所有存取權。

1. 撤銷所有工作階段

   如需立即撤銷角色登入資料所有許可的指示，請參閱撤銷 IAM 角色臨時安全登入資料。
2. 從 Amplify 主控台刪除角色

   此動作會立即生效。您不需要重新部署應用程式。

在 Amplify 主控台中刪除運算角色

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/amplify/ 開啟 Amplify 主控台。
2. 在所有應用程式頁面上，選擇要從中移除運算角色的應用程式名稱。
3. 在導覽窗格中，選擇應用程式設定，然後選擇 IAM 角色。
4. 在運算角色區段中，選擇編輯。
5. 若要刪除預設角色，請選擇角色名稱右側的 X。
6. 選擇儲存。

新增具有存取 CloudWatch Logs 許可的服務角色

Amplify 會將 SSR 執行時間的相關資訊傳送至 中的 Amazon CloudWatch Logs AWS 帳戶。當您部署 SSR 應用程式時，應用程式需要 Amplify 代表您呼叫其他 服務時擔任的 IAM 服務角色。您可以允許 Amplify 託管運算自動為您建立服務角色，也可以指定您已建立的角色。

如果您選擇允許 Amplify 為您建立 IAM 角色，該角色將已有建立 CloudWatch Logs 的許可。如果您建立自己的 IAM 角色，則需要將下列許可新增至政策，以允許 Amplify 存取 Amazon CloudWatch Logs。

```
logs:CreateLogStream
```

```
logs:CreateLogGroup
logs:DescribeLogGroups
logs:PutLogEvents
```

Git 儲存庫的統一 Webhook

Amplify 託管使用 Webhook，在對 Git 儲存庫的新遞交之後自動啟動組建。統一 Webhook 功能可改善 Amplify 與 Git 提供者的整合，並可讓您將更多 Amplify 應用程式連線至單一儲存庫。使用統一 Webhook，Amplify 現在會針對儲存庫中的所有關聯應用程式，在每個區域使用單一 Webhook。例如，如果您的儲存庫同時連線到美國東部（維吉尼亞北部）和美國西部（奧勒岡）區域中的應用程式，您將有兩個統一的 Webhook。

在此版本之前，Amplify 為與儲存庫相關聯的每個應用程式建立新的 Webhook。如果您在單一儲存庫中有多個應用程式，您可以達到個別 Git 供應商強制執行的 Webhook 限制，並且無法新增更多應用程式。對於在單一儲存庫中工作，其中有多個專案存在於單一儲存庫中的團隊來說，這特別具有挑戰性。

統一 Webhook 提供下列優點：

- 克服 Git 提供者 Webhook 限制：您可以視需要將任意數量的 Amplify 應用程式連接到單一儲存庫。
- 增強型單儲存庫支援：您在使用單儲存庫時擁有更高的靈活性和效率，其中多個專案共用單一儲存庫。
- 簡化管理：使用單一儲存庫 Webhook 管理多個 Amplify 應用程式，可降低複雜性和潛在的故障點。
- 改善工作流程整合：您可以將 Git 供應商配置的 Webhook 用於開發過程中的其他基本工作流程。

統一 Webhook 入門

建立新的應用程式

當您從 Git 儲存庫將新應用程式部署至 Amplify 託管時，會自動為您的儲存庫實作統一 Webhooks 功能。如需建立新應用程式的指示，請參閱 開始將應用程式部署到 Amplify 託管。

更新現有的應用程式

對於現有的 Amplify 應用程式，您必須將 Git 儲存庫重新連線至應用程式，以統一的 Webhook 取代現有的 Webhook。如果您已經達到 Git 提供者允許的 Webhook 數量上限，遷移至統一 Webhook 可能不會成功。在此情況下，請在重新連線之前手動移除至少一個現有的 Webhook。

您可以在部署到不同 AWS 區域的儲存庫中擁有多個應用程式。由於 Amplify 操作是以區域為基礎，因此只有您重新連接 Amplify 應用程式的區域中的 Webhook 才會遷移至統一的 Webhook。因此，您可能會在儲存庫中同時看到應用程式 ID 型 Webhook 和區域型統一 Webhook。

使用下列指示將現有的 Amplify 應用程式遷移至統一的 Webhook。

將現有的 Amplify 應用程式遷移至統一的 Webhook

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇您要遷移至統一 Webhook 的應用程式。
3. 在導覽窗格中，選擇應用程式設定，然後選擇分支設定。
4. 在分支設定頁面上，選擇重新連線儲存庫。
5. 若要確認成功遷移至統一的 Webhook，請導覽至 Git 儲存庫中的 Webhook 設定。您應該會看到格式為 的單一 Webhook URL`https://amplify-webhooks.Region.amazonaws.com/git-provider`。

Amplify 的安全性

的雲端安全 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，這些架構專為滿足最安全敏感組織的需求而建置。

安全性是 AWS 與您之間的共同責任。共同責任模式將其描述為雲端的安全性，和雲端中的安全性：

- 雲端的安全性 – AWS 負責保護在 AWS Cloud 中執行 AWS 服務的基礎設施。 AWS 也為您提供可安全使用的服務。在AWS 合規計畫中，第三方稽核人員會定期測試和驗證我們安全的有效性。若要了解適用的合規計劃 AWS Amplify，請參閱AWS 合規計劃的 服務範圍。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件可協助您了解如何在使用 Amplify 時套用共同責任模型。下列主題說明如何設定 Amplify 以符合您的安全與合規目標。您也會了解如何使用其他 AWS 服務來協助您監控和保護 Amplify 資源。

主題

- Amplify 的 Identity and Access Management
- Amplify 中的資料保護
- 的合規驗證 AWS Amplify
- 中的基礎設施安全 AWS Amplify
- Amplify 中的安全事件記錄和監控
- 預防跨服務混淆代理人
- Amplify 的安全最佳實務

Amplify 的 Identity and Access Management

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） 來使用 Amplify 資源。IAM 是您可以免費使用 AWS 服務 的 。

主題

- 目標對象

- 使用身分驗證
- 使用政策管理存取權
- Amplify 如何與 IAM 搭配使用
- Amplify 的身分型政策範例
- AWS 的 受管政策 AWS Amplify
- 針對 Amplify 身分和存取進行故障診斷

目標對象

使用方式 AWS Identity and Access Management (IAM) 會根據您的角色而有所不同：

- 服務使用者 — 若無法存取某些功能，請向管理員申請所需許可 (請參閱 針對 Amplify 身分和存取進行故障診斷)
- 服務管理員 — 負責設定使用者存取權並提交相關許可請求 (請參閱 Amplify 如何與 IAM 搭配使用)
- IAM 管理員 — 撰寫政策以管理存取控制 (請參閱 Amplify 的身分型政策範例)

使用身分驗證

身分驗證是您 AWS 使用身分憑證登入 的方式。您必須以 AWS 帳戶根使用者、IAM 使用者或擔任 IAM 角色身分進行身分驗證。

您可以使用身分來源的登入資料，例如 AWS IAM Identity Center (IAM Identity Center)、單一登入身分驗證或 Google/Facebook 登入資料，以聯合身分的形式登入。如需有關登入的詳細資訊，請參閱《AWS 登入 使用者指南》中的如何登入您的 AWS 帳戶。

對於程式設計存取， AWS 提供 SDK 和 CLI 以密碼編譯方式簽署請求。如需詳細資訊，請參閱《IAM 使用者指南》中的 API 請求的AWS 第 4 版簽署程序。

AWS 帳戶 根使用者

當您建立 時 AWS 帳戶，您會從一個名為 AWS 帳戶 theroot 使用者的登入身分開始，該身分具有對所有 AWS 服務 和 資源的完整存取權。強烈建議不要使用根使用者來執行日常任務。有關需要根使用者憑證的任務，請參閱《IAM 使用者指南》中的需要根使用者憑證的任務。

聯合身分

最佳實務是要求人類使用者使用聯合身分提供者，以 AWS 服務 使用臨時憑證存取 。

聯合身分是您企業目錄、Web 身分提供者的使用者，或使用來自身分來源的 AWS 服務 憑證存取 Directory Service 。聯合身分會擔任角色，而該角色會提供臨時憑證。

若需集中化管理存取權限，建議使用 AWS IAM Identity Center。如需詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html是一種身分具備單人或應用程式的特定許可權。建議以臨時憑證取代具備長期憑證的 IAM 使用者。如需詳細資訊，請參閱《IAM 使用者指南》中的要求人類使用者使用聯合身分提供者來 AWS 使用臨時憑證存取 。

IAM 群組會指定 IAM 使用者集合，使管理大量使用者的許可權更加輕鬆。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 使用者的使用案例。

IAM 角色

IAM 角色https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html的身分具有特定許可權，其可以提供臨時憑證。您可以透過從使用者切換到 IAM 角色 （主控台） 或呼叫 AWS CLI 或 AWS API 操作來擔任角色。如需詳細資訊，請參閱《IAM 使用者指南》中的擔任角色的方法。

IAM 角色適用於聯合身分使用者存取、臨時 IAM 使用者許可、跨帳戶存取權與跨服務存取，以及在 Amazon EC2 執行的應用程式。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 中的快帳戶資源存取。

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。政策定義與身分或資源相關聯的許可。當委託人提出請求時 AWS ， 會評估這些政策。大多數政策會以 JSON 文件 AWS 的形式存放在 中。如需進一步了解 JSON 政策文件，請參閱《IAM 使用者指南》中的 JSON 政策概觀。

管理員會使用政策，透過定義哪些主體可在哪些條件下對哪些資源執行動作，以指定可存取的範圍。

預設情況下，使用者和角色沒有許可。IAM 管理員會建立 IAM 政策並將其新增至角色，供使用者後續擔任。IAM 政策定義動作的許可，無論採用何種方式執行。

身分型政策

身分型政策是附加至身分 (使用者、使用者群組或角色) 的 JSON 許可政策文件。這類政策控制身分可對哪些資源執行哪些動作，以及適用的條件。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。

身分型政策可分為內嵌政策 (直接內嵌於單一身分) 與受管政策 (可附加至多個身分的獨立政策)。如需了解如何在受管政策及內嵌政策之間做選擇，請參閱《IAM 使用者指南》中的在受管政策與內嵌政策之間選擇。

資源型政策

資源型政策是附加到資源的 JSON 政策文件。範例包括 IAM 角色信任政策與 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。您必須在資源型政策中指定主體。

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。

其他政策類型

AWS 支援其他政策類型，可設定更多常見政策類型授予的最大許可：

- 許可界限 — 設定身分型政策可授與 IAM 實體的最大許可。如需詳細資訊，請參閱《 IAM 使用者指南》中的 IAM 實體許可界限。
- 服務控制政策 (SCP) — 為 AWS Organizations中的組織或組織單位指定最大許可。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的服務控制政策。
- 資源控制政策 (RCP) — 設定您帳戶中資源可用許可的上限。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的資源控制政策 (RCP)。
- 工作階段政策 — 在以程式設計方式為角色或聯合身分使用者建立臨時工作階段時，以參數形式傳遞的進階政策。如需詳細資訊，請參《IAM 使用者指南》中的工作階段政策。

多種政策類型

當多種類型的政策適用於請求時，產生的許可會更複雜而無法理解。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。

Amplify 如何與 IAM 搭配使用

在您使用 IAM 管理 Amplify 的存取權之前，請先了解哪些 IAM 功能可與 Amplify 搭配使用。

您可以搭配 Amplify 使用的 IAM 功能

IAM 功能	Amplify 支援
身分型政策	是

IAM 功能	Amplify 支援
資源型政策	否
政策動作	是
政策資源	是
政策條件索引鍵	是
ACL	否
ABAC(政策中的標籤)	部分
臨時憑證	是
轉送存取工作階段 (FAS)	是
服務角色	是
服務連結角色	否

若要全面了解 Amplify 和其他 AWS 服務如何與大多數 IAM 功能搭配使用，請參閱《AWS IAM 使用者指南》中的與 IAM 搭配使用的 服務。

Amplify 的身分型政策

支援身分型政策：是

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的透過客戶管理政策定義自訂 IAM 許可。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

Amplify 的身分型政策範例

若要檢視 Amplify 身分型政策的範例，請參閱 Amplify 的身分型政策範例。

Amplify 中的資源型政策

支援資源型政策：否

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。委託人可以包含帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

如需啟用跨帳戶存取權，您可以在其他帳戶內指定所有帳戶或 IAM 實體作為資源型政策的主體。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 中的快帳戶資源存取。

Amplify 的政策動作

支援政策動作：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。政策會使用動作來授予執行相關聯動作的許可。

如需 Amplify 動作的清單，請參閱《服務授權參考》中的 定義的動作 AWS Amplify。

Amplify 中的政策動作在動作之前使用以下字首：

```
amplify
```

若要在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "amplify:action1",
      "amplify:action2"
         ]
```

若要檢視 Amplify 身分型政策的範例，請參閱 Amplify 的身分型政策範例。

Amplify 的政策資源

支援政策資源：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

`Resource` JSON 政策元素可指定要套用動作的物件。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。若動作不支援資源層級許可，使用萬用字元 (*) 表示該陳述式適用於所有資源。

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

如需 Amplify 資源類型及其 ARNs，請參閱《服務授權參考》中的 定義的資源類型 AWS Amplify。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱 AWS Amplify定義的動作。

若要檢視 Amplify 身分型政策的範例，請參閱 Amplify 的身分型政策範例。

Amplify 的政策條件索引鍵

支援服務特定政策條件金鑰：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

`Condition` 元素會根據定義的條件，指定陳述式的執行時機。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。若要查看所有 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。

如需 Amplify 條件索引鍵的清單，請參閱《服務授權參考》中的 的條件索引鍵 AWS Amplify。若要了解您可以使用條件索引鍵的動作和資源，請參閱 定義的動作 AWS Amplify。

若要檢視 Amplify 身分型政策的範例，請參閱 Amplify 的身分型政策範例。

Amplify 中的存取控制清單 (ACLs)

支援 ACL：否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

使用 Amplify 的屬性型存取控制 (ABAC)

支援 ABAC (政策中的標籤)：部分

屬性型存取控制 (ABAC) 是一種授權策略，根據稱為標籤的屬性定義許可權。您可以將標籤連接至 IAM 實體 AWS 和資源，然後設計 ABAC 政策，以便在委託人的標籤符合資源上的標籤時允許操作。

如需根據標籤控制存取，請使用 aws:ResourceTag/*key-name*、aws:RequestTag/*key-name* 或 aws:TagKeys 條件索引鍵，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的使用 ABAC 授權定義許可。如要查看含有設定 ABAC 步驟的教學課程，請參閱《IAM 使用者指南》中的使用屬性型存取控制 (ABAC)。

搭配 Amplify 使用臨時登入資料

支援臨時憑證：是

臨時登入資料提供 AWS 資源的短期存取權，當您使用聯合或切換角色時，會自動建立。 AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 中的臨時安全憑證與可與 IAM 搭配運作的AWS 服務。

轉送 Amplify 的存取工作階段

支援轉寄存取工作階段 (FAS)：是

轉送存取工作階段 (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。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。

Amplify 的服務角色

支援服務角色：是

服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的建立角色以委派許可給 AWS 服務。

> **Warning**
> 變更服務角色的許可可能會中斷 Amplify 功能。只有在 Amplify 提供指引時，才能編輯服務角色。

Amplify 的服務連結角色

支援服務連結角色：否

服務連結角色是連結至 的一種服務角色 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 中 AWS 帳戶 ，並由服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需建立或管理服務連結角色的詳細資訊，請參閱《AWS IAM 使用者指南》中的使用 IAM 的 服務。在資料表中尋找服務，其中包含服務連結角色欄中的 Yes。選擇是連結，以檢視該服務的服務連結角色文件。

Amplify 的身分型政策範例

根據預設，使用者和角色沒有建立或修改 Amplify 資源的許可。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。

如需了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策 (主控台)。

如需有關 Amplify 定義的動作和資源類型的詳細資訊，包括每種資源類型的 ARNs 格式，請參閱《服務授權參考》中的適用於 的動作、資源和條件索引鍵 AWS Amplify。

主題

- 政策最佳實務
- 使用 Amplify 主控台
- 允許使用者檢視他們自己的許可

政策最佳實務

身分型政策會判斷您帳戶中的某個人員是否可以建立、存取或刪除 Amplify 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，請使用將許可授予許多常見使用案例的 AWS 受管政策。它們可在您的 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，以進一步減少許可。如需更多資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱《IAM 使用者指南》中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。如果透過特定 例如 使用服務動作

AWS 服務，您也可以使用條件來授予其存取權 CloudFormation。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素：條件。

- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM Access Analyzer 驗證政策。
- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或 中的根使用者 AWS 帳戶，請開啟 MFA 以提高安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的透過 MFA 的安全 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱《IAM 使用者指南》中的 IAM 安全最佳實務。

使用 Amplify 主控台

若要存取 AWS Amplify 主控台，您必須擁有一組最低許可。這些許可必須允許您列出和檢視 中 Amplify 資源的詳細資訊 AWS 帳戶。如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

對於僅呼叫 AWS CLI 或 AWS API 的使用者，您不需要允許最低主控台許可。反之，只需允許存取符合他們嘗試執行之 API 操作的動作就可以了。

隨著 Amplify Studio 的發行，刪除應用程式或後端需要 `amplify`和 `amplifybackend`許可。如果 IAM 政策僅提供`amplify`許可，使用者在嘗試刪除應用程式時會收到許可錯誤。如果您是管理員撰寫政策，請判斷正確的許可，以授予需要執行刪除動作的使用者。

為了確保使用者和角色仍然可以使用 Amplify 主控台，請將 Amplify `ConsoleAccess`或 `ReadOnly` AWS 受管政策連接到實體。如需詳細資訊，請參閱《IAM 使用者指南》中的新增許可到使用者。

允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此政策包含在主控台或使用 或 AWS CLI AWS API 以程式設計方式完成此動作的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
```

```
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS 的 受管政策 AWS Amplify

AWS 受管政策是由 AWS AWS 受管政策建立和管理的獨立政策旨在為許多常用案例提供許可，以便您可以開始將許可指派給使用者、群組和角色。

請記住， AWS 受管政策可能不會授予特定使用案例的最低權限許可，因為這些許可可供所有 AWS 客戶使用。我們建議您定義特定於使用案例的客戶管理政策，以便進一步減少許可。

您無法變更 AWS 受管政策中定義的許可。如果 AWS 更新受 AWS 管政策中定義的許可，則更新會影響政策連接的所有主體身分（使用者、群組和角色）。當新的 AWS 服務 啟動或新的 API 操作可供現有服務使用時， AWS 最有可能更新 AWS 受管政策。

如需詳細資訊，請參閱 IAM 使用者指南中的 AWS 受管政策。

AWS 受管政策： AdministratorAccess-Amplify

您可將 AdministratorAccess-Amplify 政策連接到 IAM 身分。Amplify 也會將此政策連接至允許 Amplify 代表您執行動作的服務角色。

在 Amplify 主控台中部署後端時，您必須建立 Amplify 用來建立和管理 AWS 資源Amplify-Backend Deployment的服務角色。IAM 會將 AdministratorAccess-Amplify受管政策連接至Amplify-Backend Deployment服務角色。

此政策授予帳戶管理許可，同時明確允許直接存取 Amplify 應用程式建立和管理後端所需的資源。

許可詳細資訊

此政策可讓您存取多個 AWS 服務，包括 IAM 動作。這些動作可讓具有此政策的身分使用 AWS Identity and Access Management 來建立具有任何許可的其他身分。這允許許可提升，且此政策應視為與AdministratorAccess政策一樣強大。

此政策會授予所有 資源iam:PassRole的動作許可。這是支援 Amazon Cognito 使用者集區組態的必要項目。

若要檢視此政策的許可，請參閱《 AWS 受管政策參考》中的 AdministratorAccess-Amplify。

AWS 受管政策：AmplifyBackendDeployFullAccess

您可將 AmplifyBackendDeployFullAccess 政策連接到 IAM 身分。

此政策授予 Amplify 使用 部署 Amplify 後端資源的完整存取許可 AWS Cloud Development Kit (AWS CDK)。許可會延遲至具有必要AdministratorAccess政策許可 AWS CDK 的角色。

許可詳細資訊

此政策包含執行下列 的許可。

- Amplify– 擷取已部署應用程式的中繼資料。
- CloudFormation– 建立、更新和刪除 Amplify 受管堆疊。
- SSM– 建立、更新和刪除 Amplify 受管 SSM 參數存放區String和SecureString參數。
- AWS AppSync– AWS AppSync 更新和擷取結構描述、解析程式和函數資源。其目的是支援 Gen 2 沙盒熱插拔功能。
- Lambda– 更新和擷取 Amplify 受管函數的組態。其目的是支援 Gen 2 沙盒熱插拔功能。

  擷取 Lambda 函數的標籤。目的是支援客戶定義的 Lambda 函數。

- `Amazon S3`– 擷取 Amplify 部署資產。
- `AWS Security Token Service`– 讓 AWS Cloud Development Kit (AWS CDK) CLI 擔任部署角色。
- `Amazon RDS`– 讀取資料庫執行個體、叢集和代理的中繼資料。
- `Amazon EC2`– 讀取子網路的可用區域資訊。
- `CloudWatch Logs`– 擷取客戶 Lambda 函數的日誌。目的是允許 Amplify 雲端開發沙盒環境將 Lambda 函數的日誌串流到客戶的終端機。

若要檢視此政策的許可，請參閱《 AWS 受管政策參考》中的 AmplifyBackendDeployFullAccess。

Amplify AWS 受管政策的更新

檢視自此服務開始追蹤 Amplify AWS 受管政策更新以來的詳細資訊。如需有關此頁面變更的自動提醒，請訂閱 的文件歷史記錄 AWS Amplify 頁面的 RSS 摘要。

變更	描述	Date
AmplifyBackendDeployFullAccess – 更新現有政策	新增 `logs:FilterLogEvents` 資源的讀取存取權，以允許 Amplify 從建立自訂日誌群組的函數串流日誌。這是現有串流 Lambda 函數日誌功能的延伸。	2024 年 11 月 14 日
AmplifyBackendDeployFullAccess – 更新現有政策	新增 `lambda:ListTags` 和 `logs:FilterLogEvents` 資源的讀取存取權，以支援客戶定義的 Lambda 函數。這些許可允許 Amplify 雲端開發沙盒環境將 Lambda 函數的日誌串流到客戶的終端機。	2024 年 7 月 18 日
AmplifyBackendDeployFullAccess – 更新現有政策	新增`arn:aws:ssm:*:*:parameter/cdk-bootstrap/*` 資源的讀取存取權，以	2024 年 5 月 31 日

變更	描述	Date
	允許 Amplify 偵測客戶帳戶中的 CDK 引導版本。	
AmplifyBackendDeployFullAccess – 更新現有政策	使用資源和帳戶條件範圍的 Amazon RDS 和 Amazon EC2 唯讀許可來新增`AmplifyDiscoverRDSVpcConfig` 政策陳述式。這些許可支援 Amplify Gen 2 `npx amplify generate schema-from-database` 命令，可讓客戶從現有的 SQL 資料庫產生 Typescript 資料結構描述。 新增 `rds:DescribeDBProxies` 、`rds:DescribeDBInstances` 、`rds:DescribeDBSubnetGroups` 、`rds:DescribeDBClusters` 和 `ec2:DescribeSubnets` 許可。`npx amplify generate schema-from-database` 命令需要這些許可來檢查指定的資料庫主機是否託管在 Amazon RDS 中，並自動產生佈建由 SQL 資料庫支援之 AWS AppSync API 所需的其他資源所需的 Amazon VPC 組態。	2024 年 4 月 17 日

變更	描述	Date
AmplifyBackendDeployFullAccess – 更新現有政策	新增`cloudformation:DeleteStack` 政策動作，以支援在呼叫 `DeleteBranch` API 時刪除堆疊。 新增`lambda:GetFunction` 政策動作以支援熱插拔函數。 新增`lambda:UpdateFunctionConfiguration` 政策動作以支援 Lambda 函數的更新。	2024 年 4 月 5 日
AdministratorAccess-Amplify – 更新至現有政策	新增 `cloudformation:TagResource` 和 `cloudformation:UnTagResource` 許可，以支援對 CloudFormation APIs呼叫。	2024 年 4 月 4 日
AmplifyBackendDeployFullAccess – 更新現有政策	新增`lambda:InvokeFunction` 政策動作以支援 AWS Cloud Development Kit (AWS CDK) 熱插拔。會 AWS CDK 直接呼叫 Lambda 函數來執行 Amazon S3 資產熱插拔。 新增`lambda:UpdateFunctionCode` 政策動作以支援熱插拔函數。	2024 年 1 月 2 日
AmplifyBackendDeployFullAccess – 更新現有政策	新增政策動作以支援 `UpdateApiKey` 操作。這是在退出並重新啟動沙盒後啟用成功應用程式部署的必要條件，而不會刪除資源。	2023 年 11 月 17 日

變更	描述	Date
AmplifyBackendDeployFullAccess – 更新現有政策	新增支援 Amplify 應用程式部署的`amplify:GetBackendEnvironment` 許可。	2023 年 11 月 6 日
AmplifyBackendDeployFullAccess – 新政策	Amplify 新增了具有部署 Amplify 後端資源所需最低許可的新政策。	2023 年 10 月 8 日
AdministratorAccess-Amplify – 更新至現有政策	新增 Amplify 命令列界面 (CLI) 所需的`ecr:DescribeRepositories` 許可。	2023 年 6 月 1 日

變更	描述	Date
AdministratorAccess-Amplify – 更新至現有政策	新增政策動作，以支援從 AWS AppSync 資源移除標籤。 新增政策動作以支援 Amazon Polly 資源。 新增政策動作，以支援更新 OpenSearch 網域組態。 新增政策動作，以支援從角色 AWS Identity and Access Management 移除標籤。 新增政策動作，以支援從 Amazon DynamoDB 資源移除標籤。 將 `cloudfront:GetCloudFrontOriginAccessIdentity` 和 `cloudfront:GetCloudFrontOriginAccessIdentityConfig` 許可新增至`CLISDKCalls` 陳述式區塊，以支援 Amplify 發佈和託管工作流程。 將 `s3:PutBucketPublicAccessBlock` 許可新增至`CLIManageviaCFNPolicy` 陳述式區塊， AWS CLI 以允許 支援在內部儲存貯體上啟用 Amazon S3 封鎖公開存取功能的 Amazon S3 安全最佳實務。	2023 年 2 月 24 日

變更	描述	Date
	將 `cloudformation:DescribeStacks` 許可新增至`CLISDKCalls` 陳述式區塊，以支援擷取客戶在 Amplify 後端處理器中重試時的 CloudFormation 堆疊，以避免在堆疊更新時重複執行。 將 `cloudformation:ListStacks` 許可新增至`CLICloudformationPolicy` 陳述式區塊。需要此許可才能完全支援 CloudFormation DescribeStacks 動作。	
AdministratorAccess-Amplify – 更新至現有政策	新增政策動作，以允許 Amplify 伺服器端轉譯功能將應用程式指標推送至客戶 中的 CloudWatch AWS 帳戶。	2022 年 8 月 30 日
AdministratorAccess-Amplify – 更新至現有政策	新增政策動作以封鎖對 Amplify 部署 Amazon S3 儲存貯體的公開存取。	2022 年 4 月 27 日
AdministratorAccess-Amplify – 更新至現有政策	新增動作以允許客戶刪除其伺服器端轉譯 (SSR) 應用程式。這也允許成功刪除對應的 CloudFront 分佈。 新增動作以允許客戶指定不同的 Lambda 函數，以使用 Amplify CLI 處理現有事件來源的事件。透過這些變更，AWS Lambda 將能夠執行 UpdateEventSourceMapping 動作。	2022 年 4 月 17 日

變更	描述	Date
AdministratorAccess-Amplify – 更新至現有政策	新增政策動作，以啟用所有資源上的 Amplify UI Builder 動作。	2021 年 12 月 2 日
AdministratorAccess-Amplify – 更新至現有政策	新增政策動作以支援使用社交身分提供者的 Amazon Cognito 身分驗證功能。 新增政策動作以支援 Lambda 層。 新增政策動作以支援 Amplify Storage 類別。	2021 年 11 月 8 日

變更	描述	Date
AdministratorAccess-Amplify – 更新至現有政策	新增 Amazon Lex 動作以支援 Amplify Interactions 類別。 新增 Amazon Rekognition 動作以支援 Amplify 預測類別。 新增 Amazon Cognito 動作，以支援 Amazon Cognito 使用者集區上的 MFA 組態。 新增 CloudFormation 動作以支援 CloudFormation StackSets。 新增 Amazon Location Service 動作以支援 Amplify Geo 類別。 新增 Lambda 動作以支援 Amplify 中的 Lambda 層。 新增 CloudWatch Logs 動作以支援 CloudWatch Events。 新增 Amazon S3 動作以支援 Amplify Storage 類別。 新增政策動作以支援伺服器端轉譯 (SSR) 應用程式。	2021 年 9 月 27 日

變更	描述	Date
AdministratorAccess-Amplify – 更新至現有政策	將所有 Amplify 動作合併為單一`amplify:*` 動作。 新增 Amazon S3 動作以支援加密客戶 Amazon S3 儲存貯體。 新增 IAM 許可界限動作，以支援已啟用許可界限的 Amplify 應用程式。 新增 Amazon SNS 動作以支援檢視起始電話號碼，以及檢視、建立、驗證和刪除目的地電話號碼。 Amplify Studio：新增 Amazon Cognito AWS Lambda、、IAM 和 CloudFormation 政策動作，以在 Amplify 主控台和 Amplify Studio 中啟用管理後端。 新增 AWS Systems Manager (SSM) 政策陳述式來管理 Amplify 環境秘密。 新增動作 CloudFormation `ListResources` 以支援 Amplify 應用程式的 Lambda 層。	2021 年 7 月 28 日
Amplify 已開始追蹤變更	Amplify 開始追蹤其 AWS 受管政策的變更。	2021 年 7 月 28 日

針對 Amplify 身分和存取進行故障診斷

使用以下資訊來協助您診斷和修正使用 Amplify 和 IAM 時可能遇到的常見問題。

主題

- 我無權在 Amplify 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 Amplify 資源

我無權在 Amplify 中執行動作

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

下列範例錯誤會在mateojackson IAM 使用者嘗試使用主控台檢視一個虛構 *my-example-widget* 資源的詳細資訊，但卻無虛構 amplify:*GetWidget* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 amplify:GetWidget on resource: my-example-widget
```

在此情況下，必須更新 mateojackson 使用者的政策，允許使用 amplify:*GetWidget* 動作存取 *my-example-widget* 資源。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

隨著 Amplify Studio 的發行，刪除應用程式或後端需要 amplify和 amplifybackend許可。如果管理員已撰寫僅提供amplify許可的 IAM 政策，您在嘗試刪除應用程式時會收到許可錯誤。

當 IAM mateojackson 使用者嘗試使用主控台刪除虛構*example-amplify-app*資源，但沒有 amplifybackend:*RemoveAllBackends*許可時，會發生下列範例錯誤。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 amplifybackend;:RemoveAllBackends on resource: example-amplify-app
```

在此情況下，Mateo 會請求管理員更新他的政策，允許他使用 *example-amplify-app* 動作存取 amplifybackend:*RemoveAllBackends* 資源。

我未獲得執行 iam:PassRole 的授權

如果您收到錯誤，告知您無權執行 iam:PassRole動作，您的政策必須更新，以允許您將角色傳遞給 Amplify。

有些 AWS 服務 可讓您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為 的 IAM `marymajor` 使用者嘗試使用主控台在 Amplify 中執行動作時，會發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞給服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 `iam:PassRole` 動作。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我想要允許 AWS 帳戶外的人員存取我的 Amplify 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了解 Amplify 是否支援這些功能，請參閱 Amplify 如何與 IAM 搭配使用。
- 若要了解如何 AWS 帳戶 在您擁有的 資源之間提供存取權，請參閱《IAM 使用者指南》中的在您擁有 AWS 帳戶 的另一個 IAM 使用者中提供存取權。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《IAM 使用者指南》中的將存取權提供給第三方 AWS 帳戶 擁有。
- 如需了解如何透過聯合身分提供存取權，請參閱《IAM 使用者指南》中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 中的跨帳戶資源存取。

Amplify 中的資料保護

AWS Amplify 符合 AWS 共同責任模型、 其中包括資料保護的法規和指導方針。 AWS 負責保護執行所有 AWS 服務的全球基礎設施。 AWS 會維持對此基礎設施上託管資料的控制。 包括處理客戶內容和個人資料的安全組態控制。 AWS 客戶和 APN 合作夥伴， 做為資料控制者或資料處理者， 負責他們放入 AWS 雲端的任何個人資料。

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予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。
- 使用 設定 API 和使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案，以及 服務中的所有 AWS 預設安全控制。
- 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 Simple Storage Service (Amazon Simple Storage Service (Amazon S3)) 的個人資料。

我們強烈建議您絕對不要將客戶帳戶號碼等敏感的識別資訊，放在自由格式的欄位中，例如Name (名稱) 欄位。這包括當您使用 Amplify 或使用主控台、API AWS CLI或 AWS SDKs的其他 AWS 服務時。您在 Amplify 或其他 服務中輸入的任何資料都可能被挑選納入診斷日誌中。當您提供外部伺服器的 URL 時，請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

如需關於資料保護的詳細資訊，請參閱 AWS 安全部落格上的 AWS 共同責任模型和歐盟《一般資料保護規範》(GDPR) 部落格文章。

靜態加密

靜態加密是指在存放時對資料進行加密，以保護您的資料免受未經授權的存取。Amplify 預設會使用由管理的 AWS KMS keys for Amazon S3 來加密應用程式的建置成品 AWS Key Management Service。

Amplify 使用 Amazon CloudFront 將應用程式提供給客戶。CloudFront 使用了適用於節點連接點 (POP) 加密的 SSD 和區域邊緣快取 (REC) 加密的 EBS 磁碟區。CloudFront Functions 中的函數程式碼和組態始終以加密格式儲存在節點 POP 上的加密 SSD 中，以及 CloudFront 使用的其他儲存位置中。

傳輸中加密

傳輸中的加密指的是保護您的資料免於在通訊端點間移動時遭到攔截。Amplify 託管預設會為傳輸中的資料提供加密。客戶與 Amplify 之間的所有通訊，以及 Amplify 與其下游相依性之間的通訊，都會使用使用 Signature 第 4 版簽署程序簽署的 TLS 連線進行保護。所有 Amplify 託管端點都使用由 管理的 SHA-256 憑證 AWS 私有憑證授權單位。如需詳細資訊，請參閱 Signature 第 4 版簽署程序和什麼是 AWS 私有憑證授權單位 。

加密金鑰管理

AWS Key Management Service (KMS) 是一種用於建立和控制的受管服務 AWS KMS keys，用於加密客戶資料的加密金鑰。 會代表客戶 AWS Amplify 產生和管理用於加密資料的密碼編譯金鑰。沒有您要管理的加密金鑰。

的合規驗證 AWS Amplify

在多個合規計畫 AWS Amplify 中，第三方稽核人員會評估 的安全與 AWS 合規。其中包括 SOC、PCI、ISO、HIPAA、MTCS、C5、K-ISMS、ENS High、OSPAR、HITRUST CSF 和 FINMA。

若要了解 AWS 服務 是否在特定合規計劃範圍內，請參閱AWS 服務 合規計劃範圍內然後選擇您感興趣的合規計劃。如需一般資訊，請參閱AWS 合規計劃。

您可以使用 下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載報告 in AWS Artifact

您使用 時的合規責任 AWS 服務 取決於資料的機密性、您公司的合規目標，以及適用的法律和法規。如需使用 時合規責任的詳細資訊 AWS 服務，請參閱 AWS 安全文件。

中的基礎設施安全 AWS Amplify

作為受管服務， AWS Amplify 受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及如何 AWS 保護基礎設施的資訊，請參閱AWS 雲端安全。若要使用基礎設施安全的最佳實務來設計您的 AWS 環境，請參閱安全支柱 AWS Well-Architected Framework 中的基礎設施保護。

您可以使用 AWS 發佈的 API 呼叫，透過網路存取 Amplify。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。

Amplify 中的安全事件記錄和監控

監控是維護 Amplify 和其他 AWS 解決方案的可靠性、可用性和效能的重要部分。 AWS 提供下列監控工具來監看 Amplify、在發生錯誤時回報，並適時採取自動動作：

- Amazon CloudWatch 會即時監控您的 AWS 資源和您在 上執行的應用程式 AWS。您可以收集和追蹤指標、建立自訂儀表板，以及設定警示，在特定指標達到您指定的閾值時通知您或採取動作。例如，您可以讓 CloudWatch 追蹤 CPU 使用量或其他 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體指標，並在需要時自動啟動新的執行個體。如需搭配 Amplify 使用 CloudWatch 指標和警示的詳細資訊，請參閱 監控 Amplify 應用程式。
- Amazon CloudWatch Logs 可讓您監控、存放及存取來自 Amazon EC2 執行個體、 AWS CloudTrail 或其他來源的日誌檔案。CloudWatch Logs 可監控日誌檔內的資訊，並在滿足特定閾值時向您發出通知。您也可以將日誌資料存檔在高耐用性的儲存空間。如需詳細資訊，請參閱 Amazon CloudWatch Logs 使用者指南。
- AWS CloudTrail 會擷取由 AWS 您的帳戶發出或代表您的帳戶發出的 API 呼叫和相關事件，並將日誌檔案交付至您指定的 Amazon Simple Storage Service (Amazon S3) 儲存貯體。您可以識別呼叫的使用者和帳戶 AWS、進行呼叫的來源 IP 地址，以及呼叫的時間。如需詳細資訊，請參閱使用 記錄 Amplify API 呼叫 AWS CloudTrail。
- Amazon EventBridge 為無伺服器事件匯流排服務，可讓您輕鬆將應用程式與來自各種來源的資料互相連線。EventBridge 可從您自己的應用程式、Software-as-a-Service(SaaS) 應用程式和服務 AWS 提供即時資料串流，並將該資料路由至 等目標 AWS Lambda。這可讓您監控 服務中發生的事件，並建置事件驅動型架構。如需詳細資訊，請參閱「Amazon EventBridge 使用者指南」。

預防跨服務混淆代理人

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 Amplify 提供給資源的許可。如果同時使用全域條件內容金鑰，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。

的值`aws:SourceArn`必須是 Amplify 應用程式的分支 ARN。以 格式指定此值`arn:`*`Partition`*`:amplify:`*`Region`*`:`*`Account`*`:apps/`*`AppId`*`/branches/`*`BranchName`*。

防範混淆代理人問題最有效的方法，是使用 `aws:SourceArn` 全域條件內容金鑰，以及資源的完整 ARN。如果不知道資源的完整 ARN，或者如果您指定了多個資源，請使用 `aws:SourceArn` 全域條件內容金鑰，同時使用萬用字元 (*) 表示 ARN 的未知部分。例如 `arn:aws:`*`servicename`*`::`*`123456789012`*`:*`。

下列範例顯示您可以套用的角色信任政策，以限制對帳戶中任何 Amplify 應用程式的存取，並防止混淆代理人問題。若要使用此政策，請以您自己的資訊取代範例政策中的紅色斜體文字。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": [
          "amplify.me-south-1.amazonaws.com",
          "amplify.eu-south-1.amazonaws.com",
          "amplify.ap-east-1.amazonaws.com",
          "amplifybackend.amazonaws.com",
          "amplify.amazonaws.com"
        ]
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:amplify:us-east-1:123456789012:apps/*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

下列範例顯示您可以套用的角色信任政策，以限制對帳戶中指定 Amplify 應用程式的存取，並防止混淆代理人問題。若要使用此政策，請以您自己的資訊取代範例政策中的紅色斜體文字。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
```

```
    "Effect": "Allow",
    "Principal": {
      "Service": [
          "amplify.me-south-1.amazonaws.com",
          "amplify.eu-south-1.amazonaws.com",
          "amplify.ap-east-1.amazonaws.com",
          "amplifybackend.amazonaws.com",
          "amplify.amazonaws.com"
        ]
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:amplify:us-east-1:123456789012:apps/d123456789/
branches/*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

Amplify 的安全最佳實務

Amplify 提供許多安全功能，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。

搭配 Amplify 預設網域使用 Cookie

當您使用 Amplify 部署 Web 應用程式時，Amplify 會在預設`amplifyapp.com`網域上為您託管。您可以在格式化為 的 URL 上檢視您的應用程式`https://branch-name.d1m7bkiki6tdw1.amplifyapp.com`。

為了增強 Amplify 應用程式的安全性，在公有字尾清單 (PSL) 中註冊 amplifyapp.com 網域。為了提高安全性，如果您需要在 Amplify 應用程式的預設網域名稱中設定敏感 Cookie，建議您使用具有`__Host-`字首的 Cookie。此做法將有助於保護您的網域免受跨站請求偽造 (CSRF) 攻擊。如需更多資訊，請參閱 Mozilla 開發人員網路中的設定 Cookie 頁面。

Amplify 託管服務配額

以下是 AWS Amplify 託管的服務配額。服務配額 （ 先前稱為限制) 是 的服務資源或操作數量上限 AWS 帳戶。

新的 AWS 帳戶 減少了應用程式和並行任務配額。 會根據您的用量自動 AWS 提高這些配額。您還可以請求增加配額。

Service Quotas 主控台可提供您的帳戶配額的相關資訊。您可以使用 Service Quotas 主控台來檢視預設配額，並對可調整的配額請求提高配額。如需詳細資訊，請參閱《Service Quotas 使用者指南》中的請求提高配額。

名稱	預設	可調整	Description
應用程式	每個受支援的區域：25	是	您可以在目前區域中此帳戶中的 AWS Amplify Console 中建立的應用程式數量上限。
每個應用程式的分支	每個受支援的區域：50	否	目前區域中，您可以在此帳戶中為每個應用程式建立的分支數量上限。
建置成品大小	所有受支援的區域：5 GB	否	應用程式建置成品的大小上限 (以 GB 為單位)。Amplify AWS 主控台會在建置之後部署建置成品。
快取成品大小	所有受支援的區域：5 GB	否	快取成品的大小上限 （ 以 GB 為單位 ）。
並行任務數	每個受支援的區域：5	是	您可以在目前區域中在此帳戶中建立的並行任務數目上限。

名稱	預設	可調整	Description
每個應用程式的網域	每個受支援的區域：5	是	目前區域中，您可以在此帳戶中為每個應用程式建立的網域數量上限。
環境快取成品大小	所有受支援的區域：5 GB	否	環境快取成品的大小上限（以 GB 為單位）。
手動部署 ZIP 檔案大小	所有受支援的區域：5 GB	否	手動部署 ZIP 檔案的大小上限 (以 GB 為單位)。
每小時應用程式建立次數上限	每個受支援的區域：25	否	在目前區域中，您可以在 AWS Amplify 主控台中在此帳戶中每小時建立的應用程式數量上限。
每秒請求權杖數	每個受支援的區域：20,000	是	應用程式每秒請求字符的數量上限。Amplify Hosting 會根據權杖使用的資源量（處理時間和資料傳輸），將權杖配置給請求。
每個網域的子網域數目	每個受支援的區域：50	否	目前區域中，您可以在此帳戶中為每個網域建立的子網域數量上限。
每個應用程式的 Webhook	每個受支援的區域：50	是	目前區域中，您可以在此帳戶中為每個應用程式建立的 Webhook 數量上限。

如需 Amplify 服務配額的詳細資訊，請參閱 中的AWS Amplify 端點和配額AWS 一般參考。

Amplify 託管疑難排解

如果您在使用 Amplify Hosting 時遇到錯誤或部署問題，請參閱本節中的主題。

主題

- 疑難排解一般 Amplify 問題
- 對 Amazon Linux 2023 建置映像問題進行故障診斷
- 對建置問題進行故障診斷
- 對自訂網域進行故障診斷
- 對伺服器端轉譯應用程式進行故障診斷
- 對重新導向和重寫進行故障診斷
- 對快取進行故障診斷
- 設定對 GitHub 儲存庫的 Amplify 存取權

疑難排解一般 Amplify 問題

下列資訊可協助您疑難排解 Amplify 託管的一般問題。

主題

- HTTP 429 狀態碼 （ 太多請求 ）
- Amplify 主控台不會顯示我的應用程式的建置狀態和上次更新時間
- 未針對新的提取請求建立 Web 預覽
- 我的手動部署在 Amplify 主控台中停滯待定狀態
- 我需要更新應用程式的 Node.js 版本

HTTP 429 狀態碼 （ 太多請求 ）

Amplify 會根據傳入請求使用的處理時間和資料傳輸，控制網站的每秒請求數 (RPS)。如果您的應用程式傳回 HTTP 429 狀態碼，傳入的請求會超過分配給應用程式的處理時間和資料傳輸量。此應用程式限制由 Amplify REQUEST_TOKENS_PER_SECOND的服務配額管理。如需配額的詳細資訊，請參閱 Amplify 託管服務配額。

若要修正此問題，我們建議您最佳化應用程式，以減少請求持續時間和資料傳輸，以增加應用程式的 RPS。例如，使用相同的 20，000 個字符，與延遲高於 200 毫秒的頁面相比，在 100 毫秒內回應的高度最佳化 SSR 頁面可以支援更高的 RPS。

同樣地，相較於傳回 250 KB 回應大小的應用程式，傳回 1 MB 回應大小的應用程式會耗用更多字符。

我們也建議您透過設定可將指定回應保留在快取中的時間最大化的Cache-Control標頭，來利用 Amazon CloudFront 快取。從 CloudFront 快取提供的請求不會計入速率限制。每個 CloudFront 分佈每秒最多可以處理 250，000 個請求，讓您可以使用快取來擴展應用程式。如需 CloudFront 快取的詳細資訊，請參閱《Amazon CloudFront 開發人員指南》中的最佳化快取和可用性。

Amplify 主控台不會顯示我的應用程式的建置狀態和上次更新時間

當您導覽至 Amplify 主控台中的所有應用程式頁面時，會顯示目前區域中每個應用程式的圖磚。如果您沒有看到建置狀態，例如已部署，以及應用程式的上次更新時間，則應用程式沒有與其相關聯的`Production`階段分支。

若要列出 主控台中的應用程式，Amplify 會使用 `ListApps` API。Amplify `ProductionBranch.status` 使用 屬性來顯示建置狀態，並使用 `ProductionBranch.lastDeployTime` 屬性來顯示上次更新時間。如需此 API 的詳細資訊，請參閱 Amplify Hosting API 文件中的 ProductionBranch。

使用下列指示，將`Production`階段與應用程式的分支建立關聯。

1. 登入 Amplify 主控台。
2. 在所有應用程式頁面上，選擇您要更新的應用程式。
3. 在導覽窗格中，選擇應用程式設定，然後選擇分支設定。
4. 在分支設定區段中，選擇編輯。
5. 針對生產分支，選擇您要使用的分支名稱。
6. 選擇儲存。
7. 返回所有應用程式頁面。現在應該為您的應用程式顯示建置狀態和上次更新時間。

未針對新的提取請求建立 Web 預覽

Web 預覽功能可讓您在將請求合併到整合分支之前，從提取請求預覽變更。Web 預覽會將對儲存庫提出的每個提取請求部署到唯一的預覽 URL，這與主要網站使用的 URL 不同。

如果您已開啟應用程式的 Web 預覽，但並未為新 PRs建立，請調查下列其中一項是否為問題的原因。

1. 檢查您的應用程式是否已達到Branches per app服務配額上限。如需配額的詳細資訊，請參閱 Amplify 託管服務配額。

   若要保持在每個應用程式 50 個分支的預設配額內，請考慮在您的應用程式中啟用自動分支刪除。這將防止您在帳戶中累積不再存在於儲存庫中的分支。

2. 如果您使用公有 GitHub 儲存庫，且您的 Amplify 應用程式已連接 IAM 服務角色，則 Amplify 不會基於安全理由建立預覽。例如，具有後端的應用程式和部署到WEB_COMPUTE託管平台的應用程式需要 IAM 服務角色。因此，如果這些類型的應用程式儲存庫是公有的，則您無法啟用這些類型的 Web 預覽。

   若要讓 Web 預覽適用於您的應用程式，您可以取消與服務角色的關聯（如果應用程式沒有後端或不是WEB_COMPUTE應用程式），或者您可以將 GitHub 儲存庫設為私有。

我的手動部署在 Amplify 主控台中停滯待定狀態

手動部署可讓您使用 Amplify Hosting 發佈 Web 應用程式，而無需連接 Git 供應商。您可以使用下列四個部署選項之一。

1. 在 Amplify 主控台中拖放您的應用程式資料夾。
2. 在 Amplify 主控台中拖放 .zip 檔案（包含您網站的建置成品）。
3. 將 .zip 檔案（包含您網站的建置成品）上傳至 Amazon S3 儲存貯體，並將儲存貯體連接至 Amplify 主控台中的應用程式。
4. 在 Amplify 主控台中使用指向 .zip 檔案的公有 URL（其中包含您網站的建置成品）。

在 Amplify 主控台中使用應用程式資料夾進行手動部署時，我們注意到拖曳功能的問題。這些部署可能會失敗，原因如下。

- 發生暫時性網路問題。
- 在上傳期間，檔案會有本機變更。
- 瀏覽器工作階段會嘗試同時上傳大量靜態資產。

當我們努力改善拖放上傳的可靠性時，建議您使用 .zip 檔案，而不是拖放應用程式資料夾。

我們強烈建議將 .zip 檔案上傳至 Amazon S3 儲存貯體，因為這可避免從 Amplify 主控台上傳檔案，並為手動部署提供更高的可靠性。Amplify 與 Amazon S3 的整合可簡化此程序。如需詳細資訊，請參閱從 Amazon S3 儲存貯體將靜態網站部署至 Amplify。

我需要更新應用程式的 Node.js 版本

使用 Node.js 版本 14、16 和 18 的應用程式 Amplify 支援將於 2025 年 9 月 15 日結束。此日期之後的行為取決於您的應用程式類型：

- SSR 應用程式：使用已棄用 Node.js 版本時發生建置失敗。您必須先升級至 Node.js 20 或更新版本，才能部署更新。
- 非 SSR 應用程式：如果您透過 buildspec 或即時套件更新手動安裝已取代的 Node.js 版本，則可以繼續使用。

無論 Node.js 版本為何，已部署的應用程式都會繼續執行。

如果您使用的是 Amazon Linux 2023 建置映像，預設支援 Node.js 20 版。從 2025 年 9 月 15 日開始，AL2023 映像將自動支援 Node.js 22，並將其預設 Node.js 版本從 18 變更為 22。

Amazon Linux 2 (AL2) 不會自動支援 Node.js 第 20 版或更新版本。如果您目前正在使用 AL2，我們建議您切換到 AL2023。您可以在 Amplify 主控台中變更建置映像。您也可以使用支援您指定之 Node.js 版本的自訂建置映像。

升級之前，建議您在新的分支上測試應用程式，以確認其正常運作。

升級選項

Amplify 主控台

您可以使用 Amplify 主控台中的即時套件更新功能來指定要使用的 Node.js 版本。如需說明，請參閱在建置映像中使用特定套件和相依性版本。

自訂建置映像

如果您使用的是自訂建置映像，且映像上安裝 NVM，則可以將 `nvm install 20`新增至 Dockerfile。若要進一步了解自訂建置映像的需求和組態指示，請參閱 自訂建置映像。

組建設定

您可以將 `nvm use`命令新增至 preBuild 命令區段，以指定要在應用程式的`amplify.yml`建置設定中使用的 Node.js 版本。如需更新應用程式建置設定的說明，請參閱 設定 Amplify 應用程式的建置設定。

下列範例示範如何自訂建置設定，將名為 的測試分支上的預設 Node.js 版本設定為 Node.js 18，並升級至 Node.js 版本 20`node-20`。

```
frontend:
  phases:
    preBuild:
      commands:
        - nvm use 18
        - if [ "${AWS_BRANCH}" = "node-20" ]; then nvm use 20; fi
```

> **Warning**
>
> 請注意，`preBuild`命令會在即時套件更新後執行。`nvm use` 命令指定的 Node.js 版本會覆寫即時套件更新設定的 Node.js 版本。

對 Amazon Linux 2023 建置映像問題進行故障診斷

以下資訊可協助您疑難排解 Amazon Linux 2023 (AL2023) 建置映像的問題。

主題

- 我想要使用 Python 執行時間執行 Amplify 函數
- 我想要執行需要超級使用者或根權限的命令

我想要使用 Python 執行時間執行 Amplify 函數

當您部署新的應用程式時，Amplify Hosting 現在預設會使用 Amazon Linux 2023 建置映像。AL2023 已預先安裝 Python 3.8、3.9、3.10 和 3.11 版。

為了回溯相容於 Amazon Linux 2 映像，AL2023 建置映像具有預先安裝舊版 Python 的符號連結。

根據預設，Python 3.10 版會全域使用。若要使用特定 Python 版本建置函數，請在應用程式的建置規格檔案中執行下列命令。

```
version: 1
backend:
  phases:
    build:
      commands:
        # use a python version globally
        - pyenv global 3.11
```

```
        # verify python version
        - python --version
        # install pipenv
        - pip install --user pipenv
        # add to path
        - export PATH=$PATH:/root/.local/bin
        # verify pipenv version
        - pipenv --version
        - amplifyPush --simple
```

我想要執行需要超級使用者或根權限的命令

如果您使用 Amazon Linux 2023 建置映像，並在執行需要超級使用者或根權限的系統命令時收到錯誤，則必須使用 Linux 命令執行這些sudo命令。例如，如果您收到執行 的錯誤`yum install -y gcc`，請使用 `sudo yum install -y gcc`。

Amazon Linux 2 建置映像使用根使用者，但 Amplify 的 AL2023 映像會使用自訂`amplify`使用者執行您的程式碼。Amplify 授予此使用者使用 Linux 命令執行sudo命令的權限。最佳實務是sudo針對需要超級使用者權限的命令使用 。

對建置問題進行故障診斷

如果您在建立或建置 Amplify 應用程式時遇到問題，請參閱本節中的主題以取得協助。

主題

- 我的儲存庫的新遞交不會觸發 Amplify 組建
- 建立新應用程式時，Amplify 主控台中不會列出我的儲存庫名稱
- 我的建置失敗並發生錯誤 Cannot find module aws-exports（僅限第 1 代應用程式）
- 我想要覆寫建置逾時

我的儲存庫的新遞交不會觸發 Amplify 組建

如果對 Git 儲存庫的新遞交未觸發 Amplify 組建，請確認您的 Webhook 仍存在於儲存庫中。如果存在，請檢查 Webhook 請求的歷史記錄，以查看是否有任何失敗。Amplify 的傳入 Webhook 承載大小限制為 256 KB。如果您將遞交推送到具有大量變更檔案的儲存庫，則可能會超過此限制，並導致組建不觸發。

建立新應用程式時，Amplify 主控台中不會列出我的儲存庫名稱

當您在 Amplify 主控台中建立新的應用程式時，您可以在新增儲存庫和分支頁面上從組織的可用儲存庫中進行選擇。如果目標儲存庫最近尚未更新，則可能不會顯示在清單中。如果您的組織有大量儲存庫，可能會發生這種情況。若要解決此問題，請將遞交推送至儲存庫，然後在主控台中重新整理儲存庫清單。這應該會導致儲存庫顯示。

我的建置失敗並發生錯誤 **Cannot find module aws-exports**（僅限第 1 代應用程式）

如果您的應用程式在建置期間找不到`aws-exports.js`檔案，則會傳回下列錯誤。

```
TS2307: Cannot find module 'aws-exports'
```

Amplify 命令列界面 (CLI) 會在您的後端建置期間產生 `aws-exports.js` 檔案。若要解決此錯誤，您必須建立 `aws-exports.js` 檔案以用於建置。將下列程式碼新增至您的建置規格，以建立 檔案：

```
backend:
  phases:
    build:
      commands:
        - "# Execute Amplify CLI with the helper script"
        - amplifyPush --simple
```

如需 Amplify 應用程式建置規格設定的完整範例，請參閱 組建規格 YAML 語法參考。

我想要覆寫建置逾時

預設建置逾時為 30 分鐘。您可以使用 `_BUILD_TIMEOUT`環境變數覆寫預設建置逾時。最短建置逾時為 5 分鐘。建置逾時上限為 120 分鐘。

如需在 Amplify 主控台中設定應用程式環境變數的說明，請參閱 設定環境變數。

對自訂網域進行故障診斷

如果您在將自訂網域連線至 Amplify 應用程式時遇到問題，請參閱本節中的主題以取得協助。

如果您在這裡找不到問題的解決方案，請聯絡 支援。如需詳細資訊，請參閱 AWS 支援 使用者指南中的建立支援案例。

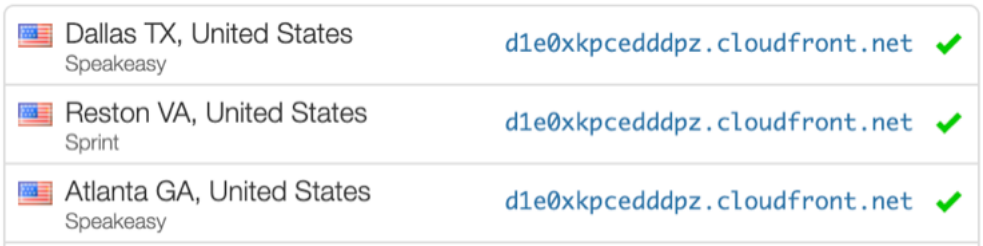
主題

- 我需要驗證我的 CNAME 是否解析
- 由第三方託管的網域卡在待驗證狀態
- 使用 Amazon Route 53 託管的網域停滯在待驗證狀態
- 具有多層級子網域的應用程式卡在待驗證狀態
- 我的 DNS 供應商不支援具有完整網域名稱的記錄
- 我收到 CNAMEAlreadyExistsException 錯誤
- 我收到其他需要驗證的錯誤
- 我在 CloudFront URL 上收到 404 錯誤
- 我在造訪我的網域時收到 SSL 憑證或 HTTPS 錯誤
- 網域重新導向中不支援的路徑元件
- 我收到跨帳戶網域關聯的 400 錯誤

我需要驗證我的 CNAME 是否解析

1. 使用第三方網域提供者更新 DNS 記錄後，您可以使用 dig 等工具或 https://www.whatsmydns.net/ 等免費網站，以確認您的 CNAME 記錄已正確解析。下列螢幕擷取畫面示範如何使用 whatsmydns.net 來檢查網域 www.example.com 的 CNAME 記錄。

2. 選擇搜尋， whatsmydns.net會顯示 CNAME 的結果。下列螢幕擷取畫面是結果清單的範例，可驗證 CNAME 是否正確解析為 cloudfront.net URL。



由第三方託管的網域卡在待驗證狀態

1. 如果您的自訂網域卡在待驗證狀態，請確認您的CNAME記錄正在解析。如需執行此任務的說明，請參閱先前的疑難排解主題：如何驗證我的 已CNAME解決。
2. 如果您的CNAME記錄未解析，請向網域提供者確認該CNAME項目存在於您的 DNS 設定中。

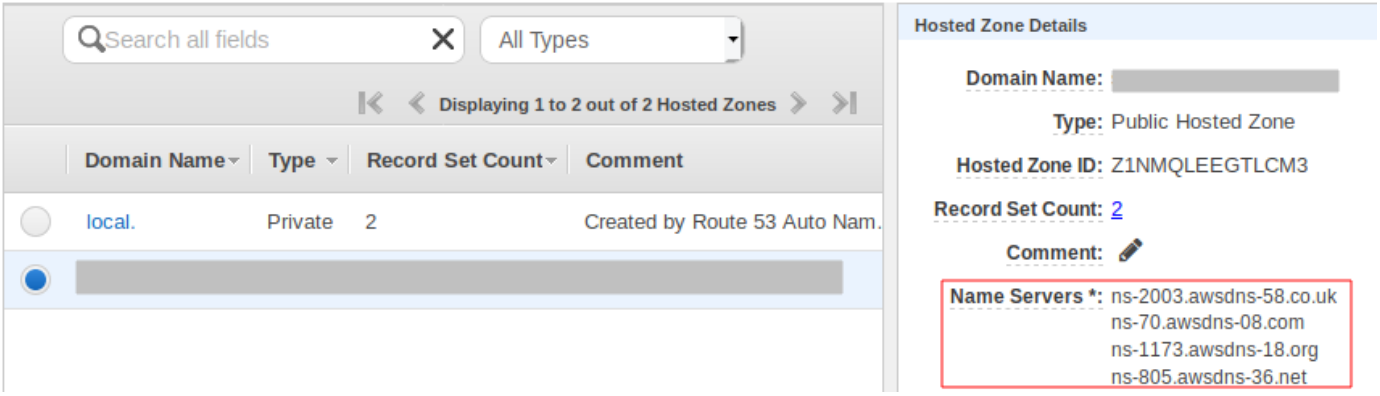
> **Important**
>
> 建立自訂網域後，請務必立即更新您的CNAME記錄。在 Amplify 主控台中建立應用程式後，每隔幾分鐘會檢查您的CNAME記錄，以判斷是否解析。如果一小時後仍未解決，則會每隔幾個小時進行檢查，這可能會導致您的網域延遲可供使用。如果您在建立應用程式數小時後新增或更新CNAME記錄，這最有可能讓您的應用程式卡在待驗證狀態。

3. 如果您已驗證CNAME記錄是否存在，則 DNS 供應商可能有問題。您可以聯絡 DNS 供應商來診斷 DNS 驗證CNAME未解決的原因，也可以將 DNS 遷移至 Route 53。如需詳細資訊，請參閱將 Amazon Route 53 設為現有網域的 DNS 服務。

使用 Amazon Route 53 託管的網域停滯在待驗證狀態

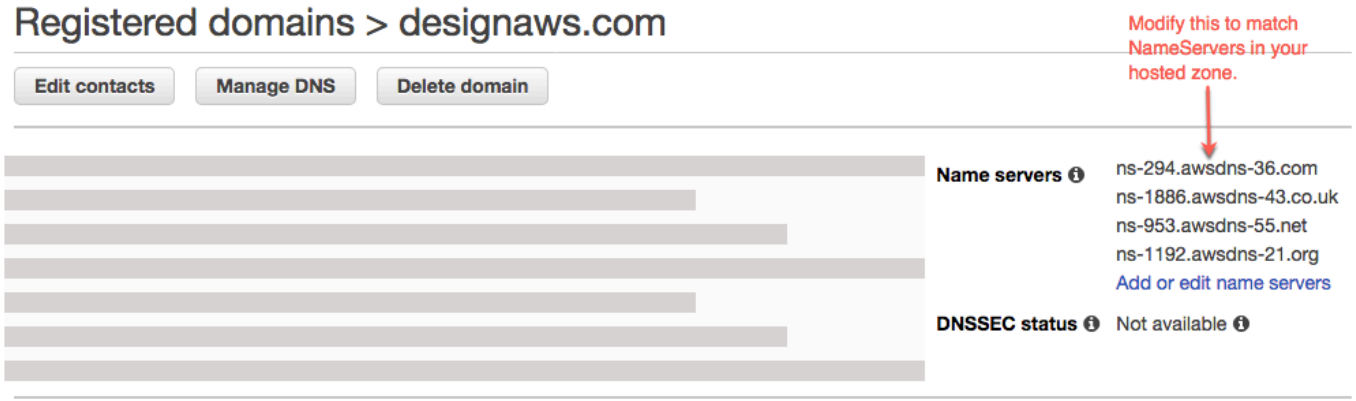
如果您將網域轉移到 Amazon Route 53，您的網域可能會有與建立應用程式時由 Amplify 發行的不同名稱伺服器。執行下列步驟來診斷錯誤的原因。

1. 登入 Amazon Route 53 主控台
2. 在導覽窗格中，選擇託管區域，然後選擇您要連線的網域名稱。
3. 從託管區域詳細資訊區段記錄名稱伺服器值。您需要這些值才能完成下一個步驟。下列 Route 53 主控台螢幕擷取畫面會在右下角顯示名稱伺服器值的位置。



4. 在導覽窗格中，選擇 Registered domains (已註冊的網域)。確認已註冊網域區段上顯示的名稱伺服器符合您在上一個步驟的託管區域詳細資訊區段中記錄的名稱伺服器值。如果不相符，請編輯名

稱伺服器值，以符合託管區域中的值。下列 Route 53 主控台螢幕擷取畫面會在右側顯示名稱伺服器值的位置。



5. 如果這無法解決問題，請聯絡 支援。如需詳細資訊，請參閱 AWS 支援 使用者指南中的建立支援案例。

具有多層級子網域的應用程式卡在待驗證狀態

如果具有多層級子網域的應用程式在連線至第三方 DNS 供應商時卡在待驗證狀態，則 DNS 記錄的格式可能有問題。有些 DNS 供應商會自動將第二層網域 (SLD) 和最上層網域 (TLD) 網域尾碼新增至您的記錄。如果您也以包含 SLD 和 TLD 的格式指定網域，這可能會導致網域驗證問題。

當您連接網域時，請先嘗試使用 Amplify 提供的完整格式指定網域名稱，例如 `_hash.docs.backend.example.com`。如果 SSL 組態卡在待驗證狀態，請嘗試從記錄中移除 TLD 和 SLD。例如，如果完整格式為 `_hash.docs.backend.example.com`，請指定 `_hash.docs.backend`。等待 15 到 30 分鐘，讓記錄傳播。然後使用 MX Toolbox 等工具來檢查驗證程序是否正常運作。

我的 DNS 供應商不支援具有完整網域名稱的記錄

有些 DNS 供應商不支援具有完整網域名稱 (FQDN) 的記錄，例如 `example.cloudfront.net`。例如，Cloudflare A records 只能寫入IPv4地址，不支援 FQDNs。若要解決此限制，建議您在DNS組態A records中使用CNAME記錄，而非 。

例如，下列DNS組態使用 A record。

```
A     | @ | ***.cloudfront.net
CNAME | www | ***.cloudfront.net
```

將其變更為下列DNS組態，以僅使用CNAME記錄。

```
CNAME | @ | ***.cloudfront.net
CNAME | www | ***.cloudfront.net
```

此解決方法可讓您將頂點網域 (@ 記錄） 正確指向 CloudFront 等服務，同時避免 Cloudflare A records 系統中IPv4-only 的限制。

我收到 CNAMEAlreadyExistsException 錯誤

如果您收到 CNAMEAlreadyExistsException 錯誤，這表示您嘗試連線的其中一個主機名稱（子網域或頂點網域） 已部署到另一個 Amazon CloudFront 分佈。錯誤來源取決於您目前的託管和 DNS 供應商。

別名，例如 CNAME `example.com`或 `sub.example.com` 一次只能與單一 CloudFront 分佈相關聯。CNAMEAlreadyExistsException 表示您的網域已與另一個 CloudFront 分佈建立關聯，無論是在相同或 AWS 帳戶可能在不同帳戶中。網域必須與先前的 CloudFront 分佈取消關聯，Amplify Hosting 建立的新分佈才能運作。如果您或組織擁有多個 AWS 帳戶，您可能需要檢查多個帳戶。

執行下列步驟來診斷 CNAMEAlreadyExistsException 錯誤的原因。

1. 登入 Amazon CloudFront 主控台，並確認沒有將此網域部署到另一個分佈。單一CNAME記錄一次可以連接到一個 CloudFront 分佈。
2. 如果您之前已將網域部署到 CloudFront 分佈，則必須將其移除。
   a. 在左側導覽功能表中選擇分佈。
   b. 選取要編輯的分佈名稱。
   c. 選擇一般索引標籤。在 Settings (設定) 區段中，選擇 Edit (編輯)。
   d. 從備用網域名稱 (CNAME) 中移除網域名稱。然後選擇儲存變更。
3. 確認目前 AWS 帳戶 或其他 中沒有使用此網域的其他 CloudFront 分佈 AWS 帳戶。如果它不會中斷任何目前正在執行的服務，請嘗試刪除並重新建立託管區域。
4. 檢查此網域是否連接到您擁有的不同 Amplify 應用程式。若是如此，確定您不是嘗試重複使用其中一個主機名稱。如果您將 `www.example.com`用於另一個應用程式，則無法將 `www.example.com`與您目前連線的應用程式搭配使用。您可以使用其他子網域，例如 `blog.example.com`。
5. 如果此網域已成功連線至另一個應用程式，然後在前一個小時內刪除，請在至少一小時過後再試一次。如果您在 6 小時後仍看到此例外狀況，請聯絡 支援。如需詳細資訊，請參閱 AWS 支援 使用者指南中的建立支援案例。

6. 如果您透過 Route 53 管理您的網域，請務必清除指向舊 CloudFront 分佈的任何託管區域CNAME或ALIAS記錄。
7. 完成上述步驟後，從 Amplify 託管移除自訂網域，然後重新開始工作流程，以在 Amplify 主控台中連接自訂網域。

我收到其他需要驗證的錯誤

如果您收到其他必要驗證錯誤，這表示 AWS Certificate Manager (ACM) 需要其他資訊才能處理此憑證請求。詐騙防護措施可能會發生此情況，例如，當網域排名在 Alexa 前 1000 名網站時。為了提供此資訊，請使用支援中心聯絡 支援。如果您沒有支援方案，請在 ACM 開發論壇中張貼新的討論主題。

> **Note**
>
> 您無法為 Amazon 擁有的網域名稱請求憑證，例如結尾為 amazonaws.com、cloudfront.net 或 elasticbeanstalk.com 的網域名稱。

我在 CloudFront URL 上收到 404 錯誤

為了提供流量，Amplify 託管會透過 CNAME 記錄指向 CloudFront URL。在將應用程式連線至自訂網域的過程中，Amplify 主控台會顯示應用程式的 CloudFront URL。不過，您無法使用此 CloudFront URL 直接存取您的應用程式。它傳回 404 錯誤。您的應用程式只會使用 Amplify 應用程式 URL （例如 `https://main.d5udybEXAMPLE.amplifyapp.com`或您的自訂網域 （例如 `www.example.com`) 來解析 。

Amplify 需要將請求路由到正確的部署分支，並使用主機名稱來執行此操作。例如，您可以設定`www.example.com`指向應用程式主線分支的網域，也可以設定`dev.example.com`指向相同應用程式開發分支的網域。因此，您必須根據應用程式設定的子網域來造訪應用程式，以便 Amplify 可以相應地路由請求。

我在造訪我的網域時收到 SSL 憑證或 HTTPS 錯誤

如果您使用第三方 DNS 供應商設定憑證授權機構授權 (CAA) DNS 記錄， AWS Certificate Manager (ACM) 可能無法更新或重新發行自訂網域 SSL 憑證的中繼憑證。若要解決此問題，您需要新增 CAA 記錄來信任至少一個 Amazon 憑證授權單位網域。下列程序說明您需要執行的步驟。

新增 CAA 記錄以信任 Amazon 憑證授權單位

1. 與您的網域提供者設定 CAA 記錄，以信任至少一個 Amazon 的憑證授權單位網域。如需設定 CAA 記錄的詳細資訊，請參閱AWS Certificate Manager 《 使用者指南》中的憑證授權機構授權 (CAA) 問題。
2. 使用下列其中一種方法來更新您的 SSL 憑證：
   - 使用 Amplify 主控台手動更新。

     > **Note**
     >
     > 此方法將導致自訂網域的停機時間。

     a. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
     b. 選擇您要新增 CAA 記錄的應用程式。
     c. 在導覽窗格中，選擇應用程式設定、網域管理。
     d. 在網域管理頁面上，刪除自訂網域。
     e. 再次將您的應用程式連線至自訂網域。此程序會發出新的 SSL 憑證，且其中繼憑證現在可以由 ACM 管理。

        若要將應用程式重新連線至您的自訂網域，請使用下列其中一個對應至您正在使用的網域提供者的程序。

        - 新增由 Amazon Route 53 管理的自訂網域.
        - 新增由第三方 DNS 供應商管理的自訂網域.
        - 更新由 GoDaddy 管理之網域的 DNS 記錄.
   - 請聯絡 支援 以重新發行您的 SSL 憑證。

網域重新導向中不支援的路徑元件

網域重新導向僅符合主機名稱部分。不支援以網域為基礎的來源規則（例如 "https://domain.com/path"）中的路徑元件，這會導致規則被忽略而不會發生錯誤。如需詳細資訊，請參閱重新導向和重寫範例參考。

我收到跨帳戶網域關聯的 400 錯誤

為 Amplify 應用程式啟動 DomainAssociation (Amplify 應用程式） 請求時，如果網域已經或先前已與相同區域中其他 AWS 帳戶中的不同 Amplify 應用程式相關聯，則這會被視為跨帳戶網域關聯。如果您收到此錯誤，表示您正在嘗試跨帳戶網域關聯，這需要手動驗證。如果您想要繼續進行跨帳戶網域關聯，請聯絡 AWS Support 尋求協助。

對伺服器端轉譯應用程式進行故障診斷

如果您在使用 Amplify 託管運算部署 SSR 應用程式時遇到意外問題，請檢閱下列疑難排解主題。如果您在這裡看不到問題的解決方案，請參閱 Amplify Hosting GitHub Issues 儲存庫中的 SSR Web 運算疑難排解指南。

主題

- 我需要使用架構轉接器的協助
- Edge API 路由導致我的 Next.js 建置失敗
- 隨需增量靜態復原不適用於我的應用程式
- 我應用程式的建置輸出超過允許的大小上限
- 我的建置失敗，並出現記憶體不足錯誤
- 我應用程式的 HTTP 回應大小太大
- 如何在本機測量運算應用程式的啟動時間？
- 我的建置失敗，並出現已棄用的 Node.js 版本錯誤

我需要使用架構轉接器的協助

如果您在部署使用架構轉接器的 SSR 應用程式時遇到問題，請參閱 對任何 SSR 架構使用開放原始碼轉接器。

Edge API 路由導致我的 Next.js 建置失敗

目前，Amplify 不支援 Next.js Edge API Routes。使用 Amplify 託管應用程式時，您必須使用非邊緣 APIs 和中介軟體。

隨需增量靜態復原不適用於我的應用程式

從 12.2.0 版開始，Next.js 支援增量靜態再生 (ISR)，以手動清除特定頁面的 Next.js 快取。不過，Amplify 目前不支援隨需 ISR。如果您的應用程式使用 Next.js 隨需重新驗證，當您將應用程式部署到 Amplify 時，此功能將無法運作。

我應用程式的建置輸出超過允許的大小上限

目前，Amplify 支援用於 SSR 應用程式的建置輸出大小上限為 220 MB。如果您收到錯誤訊息，指出您應用程式建置輸出的大小超過允許的大小上限，您必須採取步驟來減少它。

若要減少應用程式建置輸出的大小，您可以檢查應用程式的建置成品，並識別任何要更新或移除的大型相依性。首先，將建置成品下載到您的本機電腦。然後，檢查目錄的大小。例如，`node_modules`目錄可能包含二進位檔，例如 Next.js 伺服器執行期檔案所參考`@esbuild`的 `@swc`和 。由於執行時間不需要這些二進位檔，因此您可以在建置之後刪除它們。

使用下列指示下載應用程式的建置輸出，並使用 AWS Command Line Interface (CLI) 檢查目錄的大小。

下載並檢查 Next.js 應用程式的建置輸出

1. 開啟終端機視窗並執行下列命令。將應用程式 ID、分支名稱和任務 ID 變更為您自己的資訊。針對任務 ID，針對您正在調查的失敗組建使用組建編號。

   ```
   aws amplify get-job --app-id abcd1234 --branch-name main --job-id 2
   ```

2. 在終端機輸出中，找到 `job`區段中的預先簽章成品 URL`steps`。`stepName: "BUILD"`在下列範例輸出中，URL 會以紅色反白顯示。

   ```
   "job": {
       "summary": {
           "jobArn": "arn:aws:amplify:us-west-2:111122223333:apps/abcd1234/main/
   jobs/0000000002",
           "jobId": "2",
           "commitId": "HEAD",
           "commitTime": "2024-02-08T21:54:42.398000+00:00",
           "startTime": "2024-02-08T21:54:42.674000+00:00",
           "status": "SUCCEED",
           "endTime": "2024-02-08T22:03:58.071000+00:00"
       },
       "steps": [
   ```

```
        {
            "stepName": "BUILD",
            "startTime": "2024-02-08T21:54:42.693000+00:00",
            "status": "SUCCEED",
            "endTime": "2024-02-08T22:03:30.897000+00:00",
            "logUrl": "https://aws-amplify-prod-us-west-2-artifacts.s3.us-
west-2.amazonaws.com/abcd1234/main/0000000002/BUILD/log.txt?X-Amz-Security-
Token=IQoJb3JpZ2luX2V...Example
```

3. 複製 URL 並貼到瀏覽器視窗。artifacts.zip 檔案會下載到您的本機電腦。這是您的建置輸出。
4. 執行du磁碟用量命令來檢查目錄的大小。下列範例命令會傳回 compute和 static 目錄的大小。

```
du -csh compute static
```

以下是輸出的範例，其中包含 compute和 static目錄的大小資訊。

```
 29M    compute
3.8M    static
 33M    total
```

5. 開啟 compute目錄，並找到 node_modules 資料夾。檢閱您可以更新或移除的檔案相依性，以減少資料夾的大小。
6. 如果您的應用程式包含執行時間中不需要的二進位檔，請在建置之後將下列命令新增至應用程式amplify.yml檔案的建置區段，以將其刪除。

```
- rm -f node_modules/@swc/core-linux-x64-gnu/swc.linux-x64-gnu.node
- rm -f node_modules/@swc/core-linux-x64-musl/swc.linux-x64-musl.node
```

以下是 amplify.yml 檔案的建置命令區段範例，這些命令會在執行生產建置後新增。

```
frontend:
  phases:
    build:
      commands:
         -npm run build

         // After running a production build, delete the files
         - rm -f node_modules/@swc/core-linux-x64-gnu/swc.linux-x64-gnu.node
         - rm -f node_modules/@swc/core-linux-x64-musl/swc.linux-x64-musl.node
```

我的建置失敗，並出現記憶體不足錯誤

Next.js 可讓您快取建置成品，以改善後續建置的效能。此外，Amplify 的 AWS CodeBuild 容器會代您壓縮此快取並上傳至 Amazon S3，以改善後續建置效能。這可能會導致您的建置因記憶體不足錯誤而失敗。

執行下列動作，以防止您的應用程式在建置階段期間超過記憶體限制。首先，`.next/cache/**/*`從建置設定的 cache.paths 區段中移除 。接著，從您的建置設定檔案移除`NODE_OPTIONS`環境變數。反之，請在 Amplify 主控台中設定`NODE_OPTIONS`環境變數，以定義節點最大記憶體限制。如需使用 Amplify 主控台設定環境變數的詳細資訊，請參閱 設定環境變數。

進行這些變更後，請再次嘗試建置。如果成功，請將 `.next/cache/**/*`新增至建置設定檔案的 cache.paths 區段。

如需改善建置效能的 Next.js 快取組態詳細資訊，請參閱 Next.js 網站上的 AWS CodeBuild。

我應用程式的 HTTP 回應大小太大

目前，Amplify 支援使用 Web 運算平台的 Next.js 12 及更新版本的應用程式的回應大小上限為 5.72 MB。超過該限制的回應會傳回 504 個錯誤，而用戶端沒有內容。

如何在本機測量運算應用程式的啟動時間？

使用下列指示來判斷 Next.js 12 或更新版本運算應用程式的本機初始化/啟動時間。您可以在本機與 Amplify Hosting 上比較應用程式的效能，並使用結果來改善應用程式的效能。

在本機測量 Next.js 運算應用程式的初始化時間

1. 開啟應用程式的 `next.config.js` 檔案，並將 `output`選項設定為 `standalone` ，如下所示。

```
** @type {import('next').NextConfig} */
const nextConfig = {
  // Other options
  output: "standalone",
};

module.exports = nextConfig;
```

2. 開啟終端機視窗並執行下列命令來建置應用程式。

```
next build
```

3. 執行下列命令，將.next/static資料夾複製到 .next/standalone/.next/static。

```
cp -r .next/static .next/standalone/.next/static
```

4. 執行下列命令，將public資料夾複製到 .next/standalone/public。

```
cp -r public .next/standalone/public
```

5. 執行下列命令來啟動 Next.js 伺服器。

```
node .next/standalone/server.js
```

6. 請注意，在步驟 5 中執行命令和伺服器啟動之間需要多長時間。當伺服器接聽連接埠時，應列印下列訊息。

```
Listening on port 3000
```

7. 請注意，在步驟 6 中啟動伺服器後，任何其他模組載入所需的時間。例如，類似 的程式庫bugsnag需要 10-12 秒才能載入。載入後，會顯示確認訊息 [bugsnag] loaded。
8. 同時新增步驟 6 和步驟 7 的持續時間。此結果是運算應用程式的本機初始化/啟動時間。

我的建置失敗，並出現已棄用的 Node.js 版本錯誤

問題：您的 SSR 應用程式建置失敗，並出現不支援 Node.js 版本的錯誤。

```
# NODE.JS VERSION NOT SUPPORTED
================================================================================
Your application uses Node.js v18.x.x, which is no longer supported.
AWS Amplify Console has ended support for Node.js 14, Node.js 16 and Node.js 18.

To deploy your application, please upgrade to Node.js 20 or later.

For detailed migration guidelines, visit: https://docs.aws.amazon.com/amplify/latest/
userguide/troubleshooting-general.html#update-node-version
================================================================================
```

原因：您的 SSR 應用程式是使用已取代的 Node.js 版本 (14.x、16.x 或 18.x) 建置。自 2025 年 9 月 15 日起，Amplify 會封鎖在建置程序期間使用這些已棄用版本的 SSR 應用程式部署。

更新您的建置環境以使用 Node.js 20 或更新版本。如需詳細說明，請參閱 我需要更新應用程式的 Node.js 版本。

對重新導向和重寫進行故障診斷

如果您在設定 Amplify 應用程式的重新導向和重寫時遇到問題，請參閱本節中的主題以取得協助。

主題

- 即使使用 SPA 重新導向規則，也會拒絕特定路由的存取。
- 我想要設定 API 的反向代理

即使使用 SPA 重新導向規則，也會拒絕特定路由的存取。

如果您使用 SPA 重新導向規則取得特定路由的存取遭拒錯誤，`baseDirectory`可能無法在應用程式的建置設定中正確設定 。例如，如果您應用程式的前端建置至 `build`目錄，您的建置設定也必須指向 `build`目錄。下列建置規格範例示範此設定。

```
frontend:
  artifacts:
    baseDirectory: build
    files:
        - "**/*"
```

如需 Amplify 應用程式建置規格設定的完整範例，請參閱 組建規格 YAML 語法參考

我想要設定 API 的反向代理

您可以使用下列 JSON 來設定動態端點的反向代理。

```
[
  {
    "source": "/documents/<*>",
    "target": "https://otherdomain/resource/<*>",
    "status": "200",
    "condition": null
  }
]
```

如需為 Amplify 應用程式與第三方 API 建立反向代理的基本範例，請參閱 反向代理重寫。

對快取進行故障診斷

如果您遇到 Amplify 應用程式的快取問題，請參閱本節中的主題以取得協助。

主題

- 我想要減少應用程式快取的大小
- 我想要停用從應用程式的快取讀取

我想要減少應用程式快取的大小

如果您使用快取，您可能會快取未在組建之間清除的中繼檔案。快取這些不常使用的檔案會增加快取的大小。若要防止這種情況，您可以使用應用程式建置規格 cache區段中的 !指令，排除特定資料夾遭到快取。

下列建置設定範例示範如何使用 !指令來指定您不想快取的資料夾。

```
cache:
  paths:
    - node_modules/**/*
    - "!node_modules/path/not/to/cache"
```

當您快取 node_modules 資料夾時，預設node_modules/.cache會省略 。

如需 Amplify 應用程式建置規格設定的完整範例，請參閱 組建規格 YAML 語法參考

我想要停用從應用程式的快取讀取

如果您想要停用從應用程式的快取讀取，請從應用程式的建置規格中移除快取區段。

設定對 GitHub 儲存庫的 Amplify 存取權

Amplify 現在使用 GitHub 應用程式功能來授權 Amplify 對 GitHub 儲存庫的唯讀存取。使用 Amplify GitHub 應用程式時，許可會進行更精細的調校，讓您能夠僅將 Amplify 存取權授予您指定的儲存庫。若要進一步了解 GitHub 應用程式，請參閱 GitHub 網站上的關於 GitHub 應用程式。

當您連接存放在 GitHub 儲存庫的新應用程式時，根據預設，Amplify 會使用 GitHub 應用程式來存取儲存庫。不過，您先前從 GitHub 儲存庫連線的現有 Amplify 應用程式會使用 OAuth 進行存取。CI/CD 將繼續適用於這些應用程式，但強烈建議您遷移它們以使用新的 Amplify GitHub 應用程式。

當您使用 Amplify 主控台部署新應用程式或遷移現有應用程式時，系統會自動將您導向 Amplify GitHub 應用程式的安裝位置。若要手動存取應用程式的安裝登陸頁面，請開啟 Web 瀏覽器，然後依區域導覽至應用程式。使用 格式`https://github.com/apps/aws-amplify-REGION`，將 *REGION* 取代為您部署 Amplify 應用程式的區域。例如，若要在美國西部（奧勒岡）區域安裝 Amplify GitHub 應用程式，請導覽至 https：//https://github.com/apps/aws-amplify-us-west-2。

主題

- 為新部署安裝和授權 Amplify GitHub 應用程式
- 將現有OAuth應用程式遷移至 Amplify GitHub 應用程式
- 為 CloudFormation、 CLI 和 SDK 部署設定 Amplify GitHub 應用程式
- 使用 Amplify GitHub 應用程式設定 Web 預覽

為新部署安裝和授權 Amplify GitHub 應用程式

當您從 GitHub 儲存庫中的現有程式碼部署新的應用程式至 Amplify 時，請使用下列指示來安裝和授權 GitHub 應用程式。

安裝和授權 Amplify GitHub 應用程式

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 從所有應用程式頁面，選擇新增應用程式，然後選擇託管 Web 應用程式。
3. 在開始使用 Amplify 託管頁面上，選擇 GitHub，然後選擇繼續。
4. 如果這是第一次連接 GitHub 儲存庫，GitHub.com,請求在您的 GitHub AWS Amplify 帳戶中授權的許可。選擇 Authorize (授權)。
5. 接著，您必須在 GitHub 帳戶中安裝 Amplify GitHub 應用程式。在 Github.com 上開啟一個頁面，請求在您的 GitHub AWS Amplify 帳戶中安裝和授權的許可。
6. 選取您要安裝 Amplify GitHub 應用程式的 GitHub 帳戶。
7. 執行以下任意一項：
   - 若要將安裝套用至所有儲存庫，請選擇所有儲存庫。
   - 若要將安裝限制為您選取的特定儲存庫，請選擇僅選取儲存庫。請務必在您選取的儲存庫中包含您要遷移之應用程式的儲存庫。

8. 選擇安裝和授權。
9. 系統會將您重新導向至 Amplify 主控台中應用程式的新增儲存庫分支頁面。
10. 在最近更新的儲存庫清單中，選取要連線的儲存庫名稱。
11. 在分支清單中，選取要連線的儲存庫分支名稱。
12. 選擇下一步。
13. 在設定建置設定頁面上，選擇下一步。
14. 在檢閱頁面上，選擇儲存並部署。

將現有OAuth應用程式遷移至 Amplify GitHub 應用程式

您先前從 GitHub 儲存庫連線的現有 Amplify 應用程式會使用 OAuth 進行儲存庫存取。我們強烈建議您遷移這些應用程式以使用 Amplify GitHub 應用程式。

使用下列指示來遷移應用程式，並在 GitHub 帳戶中刪除其對應的 OAuth Webhook。請注意，遷移程序會根據 Amplify GitHub 應用程式是否已安裝而有所不同。遷移第一個應用程式並安裝和授權 GitHub 應用程式後，您只需要更新後續應用程式遷移的儲存庫許可。

將應用程式從 OAuth 遷移至 GitHub 應用程式

1. 登入 AWS 管理主控台 並開啟 Amplify 主控台。
2. 選擇您要遷移的應用程式。
3. 在應用程式的資訊頁面上，找到藍色遷移至我們的 GitHub 應用程式訊息，然後選擇開始遷移。
4. 在安裝和授權 GitHub 應用程式頁面上，選擇設定 GitHub 應用程式。
5. 在 GitHub.com,請求在您的 GitHub AWS Amplify 帳戶中授權的許可。選擇 Authorize (授權)。
6. 選取您要安裝 Amplify GitHub 應用程式的 GitHub 帳戶。
7. 執行以下任意一項：
   - 若要將安裝套用至所有儲存庫，請選擇所有儲存庫。
   - 若要將安裝限制為您選取的特定儲存庫，請選擇僅選取儲存庫。請務必在您選取的儲存庫中包含您要遷移之應用程式的儲存庫。
8. 選擇安裝和授權。
9. 系統會將您重新導向至 Amplify 主控台中應用程式的安裝和授權 GitHub 應用程式頁面。如果 GitHub 授權成功，您將會看到成功訊息。選擇下一步。
10. 在完成安裝頁面上，選擇完成安裝。此步驟會刪除您現有的 Webhook、建立新的 Webhook，並完成遷移。

為 CloudFormation、 CLI 和 SDK 部署設定 Amplify GitHub 應用程式

您先前從 GitHub 儲存庫連線的現有 Amplify 應用程式會使用 OAuth 進行儲存庫存取。這可能包括您使用 Amplify 命令列界面 (CLI) CloudFormation或 SDKs 部署的應用程式。我們強烈建議您遷移這些應用程式，以使用新的 Amplify GitHub 應用程式。遷移必須在 的 Amplify 主控台中執行 AWS 管理主控台。如需說明，請參閱將現有OAuth應用程式遷移至 Amplify GitHub 應用程式。

您可以使用 CloudFormation Amplify CLI 和 SDKs 來部署使用 GitHub 應用程式進行儲存庫存取的新 Amplify 應用程式。此程序需要您先在 GitHub 帳戶中安裝 Amplify GitHub 應用程式。接著，您需要在 GitHub 帳戶中產生個人存取字符。最後，部署應用程式並指定個人存取字符。

在您的帳戶中安裝 Amplify GitHub 應用程式

1. 開啟 Web 瀏覽器，並導覽至您將部署應用程式的 AWS 區域中 Amplify GitHub 應用程式的安裝位置。

   使用 格式`https://github.com/apps/aws-amplify-REGION/installations/new`，將 *`REGION`* 取代為您自己的輸入。例如，如果您在美國西部 （奧勒岡） 區域安裝應用程式，請指定 `https://github.com/apps/aws-amplify-us-west-2/installations/new`。
2. 選取您要安裝 Amplify GitHub 應用程式的 GitHub 帳戶。
3. 執行以下任意一項：
   - 若要將安裝套用至所有儲存庫，請選擇所有儲存庫。
   - 若要將安裝限制為您選取的特定儲存庫，請選擇僅選取儲存庫。請務必在您選取的儲存庫中包含您要遷移之應用程式的儲存庫。
4. 選擇 Install (安裝)。

在您的 GitHub 帳戶中產生個人存取字符

1. 登入您的 GitHub 帳戶。
2. 在右上角，找到您的設定檔相片，然後從功能表中選擇設定。
3. 在左側導覽功能表中，選擇開發人員設定。
4. 在 GitHub 應用程式頁面的左側導覽功能表中，選擇個人存取字符。
5. 在個人存取字符頁面上，選擇產生新字符。
6. 在新增個人存取字符頁面上，針對備註輸入字符的描述性名稱。
7. 在選取範圍區段中，選取 admin：repo_hook。
8. 選擇 Generate token (產生字符)。

9. 複製並儲存個人存取字符。當您使用 CLI 或 SDKs 部署 Amplify 應用程式時 CloudFormation，您將需要提供它。

在您的 GitHub 帳戶中安裝 Amplify GitHub 應用程式並產生個人存取字符後，您可以使用 Amplify CLI CloudFormation或 SDKs 部署新的應用程式。使用 `accessToken` 欄位來指定您在先前程序中建立的個人存取字符。如需詳細資訊，請參閱《Amplify API 參考》中的 CreateApp 和AWS CloudFormation《 使用者指南》中的 AWS::Amplify::App。

下列 CLI 命令會部署新的 Amplify 應用程式，該應用程式使用 GitHub 應用程式進行儲存庫存取。將 *myapp-using-githubapp*、*https://github.com/Myaccount/react-app* 和 *MY_TOKEN* 取代為您自己的資訊。

```
aws amplify create-app --name myapp-using-githubapp --repository https://github.com/
Myaccount/react-app --access-token MY_TOKEN
```

使用 Amplify GitHub 應用程式設定 Web 預覽

Web 預覽會將對 GitHub 儲存庫提出的每個提取請求 (PR) 部署至唯一的預覽 URL。預覽版現在使用 Amplify GitHub 應用程式來存取您的 GitHub 儲存庫。如需安裝和授權適用於 Web 預覽的 GitHub 應用程式的說明，請參閱 啟用提取請求的 Web 預覽 。

AWS Amplify 託管參考

使用本節中的主題來尋找 的詳細參考資料 AWS Amplify。

主題

- AWS CloudFormation 支援
- AWS Command Line Interface 支援
- 資源標記支援
- Amplify 託管 API

AWS CloudFormation 支援

使用 AWS CloudFormation 範本佈建 Amplify 資源，啟用可重複且可靠的 Web 應用程式部署。 AWS CloudFormation 提供通用語言，讓您描述和佈建雲端環境中的所有基礎設施資源，只需按幾下滑鼠，即可簡化跨多個 AWS 帳戶和/或區域的推展。

如需 Amplify 託管，請參閱 Amplify CloudFormation 文件。對於 Amplify Studio，請參閱 Amplify UI Builder CloudFormation 文件。

AWS Command Line Interface 支援

使用 從命令列 AWS Command Line Interface 以程式設計方式建立 Amplify 應用程式。如需詳細資訊，請參閱 AWS CLI 文件。

資源標記支援

您可以使用 AWS Command Line Interface 來標記 Amplify 資源。如需詳細資訊，請參閱 AWS CLI tag-resource 文件。

Amplify 託管 API

此參考提供 Amplify 託管 API 的動作和資料類型說明。如需詳細資訊，請參閱 Amplify API 參考文件。

的文件歷史記錄 AWS Amplify

下表說明文件自上次發行以來的重要變更 AWS Amplify。

- 文件最近更新時間：2025 年 9 月 8 日

變更	描述	Date
Node.js 支援版本的更新	更新使用 Amplify Hosting 部署伺服器端轉譯應用程式中 Node.js 支援版本的相關資訊。	2025 年 9 月 8 日
已更新建置設定和組態章節	更新管理 Amplify 應用程式的建置組態章節，說明新的可設定建置執行個體類型功能，可讓您選擇為應用程式提供所需 CPU、記憶體和磁碟空間資源的執行個體類型。	2025 年 5 月 28 日
已更新防火牆章節	更新Amplify 託管網站的防火牆支援章節，說明 Amplify 與整合的一般可用性 (GA) AWS WAF，包括 GA 功能和定價結構。	2025 年 3 月 26 日
新的偏斜保護章節	新增Amplify 部署的偏斜保護章節，說明可消除 Amplify Web 應用程式中用戶端與伺服器之間版本扭曲問題的扭曲保護功能。	2025 年 3 月 10 日
已更新 Webhooks 章節	新增Git 儲存庫的統一 Webhook主題來描述統一 Webhook 功能，該功能針對與單一 Git 儲存庫相關聯的所有	2025 年 3 月 10 日

變更	描述	Date
	Amplify 應用程式使用一個完整的 Webhook。	
新增 新增 SSR 運算角色以允許存取 AWS 資源主題	已新增新增 SSR 運算角色以允許存取 AWS 資源主題，說明如何建立 Amplify SSR Compute 角色並將其與應用程式建立關聯，以便讓 Amplify Compute 服務存取 AWS 資源。	2025 年 2 月 17 日
新增使用 AWS WAF 來保護您的 Amplify 應用程式章節	新增Amplify 託管網站的防火牆支援章節，說明 Amplify 與 AWS WAF （預覽） 的整合，可讓您使用 Web 存取控制清單 (Web ACL) 來保護 Web 應用程式。	2024 年 12 月 18 日
更新 受管政策主題	更新AWS 的 受管政策 AWS Amplify主題，說明 Amplify 受 AWS 管政策的最新變更。	2024 年 11 月 14 日
更新了對 Next.js 主題的 Amplify 支援	更新Next.js 的 Amplify 支援主題，說明 Amplify 對 Next.js 第 15 版的支援。	2024 年 11 月 6 日
新增從 Amazon S3 章節將靜態網站部署至 Amplify	新增從 Amazon S3 儲存貯體將靜態網站部署至 Amplify章節，說明 Amplify 與 Amazon S3 的新整合，讓您S3只要按幾下滑鼠就能託管存放在 上的靜態網站內容。	2024 年 10 月 16 日

變更	描述	Date
新增管理快取組態章節	新增管理應用程式的快取組態章節，說明 Amplify 的預設快取行為，以及如何將受管快取政策套用至內容。	2024 年 8 月 13 日
更新 受管政策主題	更新AWS 的 受管政策 AWS Amplify主題，說明 Amplify 受 AWS 管政策的最新變更。	2024 年 7 月 18 日
更新 受管政策主題	更新AWS 的 受管政策 AWS Amplify主題，說明 Amplify 受 AWS 管政策的最新變更。	2024 年 5 月 31 日
更新 受管政策主題	更新AWS 的 受管政策 AWS Amplify主題，說明 Amplify 受 AWS 管政策的最新變更。	2024 年 4 月 17 日
更新入門章節	更新開始將應用程式部署到 Amplify 託管章節以使用教學課程中的 Next.js 範例應用程式。	2024 年 4 月 12 日
更新 受管政策主題	更新AWS 的 受管政策 AWS Amplify主題，說明 Amplify 受 AWS 管政策的最新變更。	2024 年 4 月 5 日
更新 受管政策主題	更新AWS 的 受管政策 AWS Amplify主題，說明 Amplify 受 AWS 管政策的最新變更。	2024 年 4 月 4 日
新的故障診斷章節	新增Amplify 託管疑難排解章節，說明如何修正部署到 Amplify Hosting 的應用程式所遇到的問題。	2024 年 4 月 2 日

變更	描述	Date
對自訂 SSL/TLS 憑證的新支援	將使用 SSL/TLS 憑證主題新增至連接自訂網域章節，說明將應用程式連線至自訂網域時 Amplify 對自訂 SSL/TLS 憑證的支援。	2024 年 2 月 20 日
更新 受管政策主題	更新AWS 的 受管政策 AWS Amplify主題，說明 Amplify 受 AWS 管政策的最新變更。	2024 年 1 月 2 日
SSR 架構的新支援	更新使用 Amplify Hosting 部署伺服器端轉譯應用程式主題，說明 Amplify 支援具有開放原始碼轉接器的任何 Javascript 型 SSR 架構。	2023 年 11 月 19 日
映像最佳化功能啟動的新支援	新增SSR 應用程式的影像最佳化主題，說明伺服器端轉譯應用程式映像最佳化的內建支援。	2023 年 11 月 19 日
更新 受管政策主題	更新AWS 的 受管政策 AWS Amplify主題，說明 Amplify 受 AWS 管政策的最新變更。	2023 年 11 月 17 日
更新 受管政策主題	更新AWS 的 受管政策 AWS Amplify主題，說明 Amplify 受 AWS 管政策的最新變更。	2023 年 11 月 6 日
新的萬用字元子網域主題	新增設定萬用字元子網域主題，說明對自訂網域上萬用字元子網域的支援。	2023 年 11 月 6 日

變更	描述	Date
新的 受管政策	更新AWS 的 受管政策 AWS Amplify主題，說明 Amplify 的新 AmplifyBackendDeployFullAccess AWS 受管政策。	2023 年 10 月 8 日
對 monorepo 架構功能啟動的新支援	更新設定 monorepo 組建設定主題，說明使用 npm 工作區、pnpm 工作區、Yarn 工作區、Nx 和 Turborepo 建立的單一儲存庫中部署應用程式的支援。	2023 年 6 月 19 日
更新 受管政策主題	更新AWS 的 受管政策 AWS Amplify主題，說明 Amplify 受 AWS 管政策的最新變更。	2023 年 6 月 1 日
更新 受管政策主題	更新AWS 的 受管政策 AWS Amplify主題，說明 Amplify 受 AWS 管政策的最新變更。	2023 年 2 月 24 日
更新伺服器端轉譯章節	更新使用 Amplify Hosting 部署伺服器端轉譯應用程式章節，說明 Amplify 對 Next.js 第 12 版和第 13 版支援的最新變更。	2022 年 11 月 17 日
更新 受管政策主題	更新AWS 的 受管政策 AWS Amplify主題，說明 Amplify 受 AWS 管政策的最新變更。	2022 年 8 月 30 日
更新 受管政策主題	更新建置應用程式的後端主題，說明如何使用 Amplify Studio 部署後端。	2022 年 8 月 23 日

變更	描述	Date
更新 受管政策主題	更新AWS 的 受管政策 AWS Amplify主題，說明 Amplify 受 AWS 管政策的最新變更。	2022 年 4 月 27 日
更新 受管政策主題	更新AWS 的 受管政策 AWS Amplify主題，說明 Amplify 受 AWS 管政策的最新變更。	2022 年 4 月 17 日
新的 GitHub 應用程式功能啟動	新增設定對 GitHub 儲存庫的 Amplify 存取權主題來描述新的 GitHub 應用程式，以授權 Amplify 存取您的 GitHub 儲存庫。	2022 年 4 月 5 日
新的 Amplify Studio 功能啟動	更新歡迎使用 AWS Amplify 託管主題以描述 Amplify Studio 的更新，該更新提供視覺化設計工具來建立 UI 元件，您可以連接到後端資料。	2021 年 12 月 2 日
更新 受管政策主題	更新AWS 的 受管政策 AWS Amplify主題，說明 Amplify 受 AWS 管政策的最新變更，以支援 Amplify Studio。	2021 年 12 月 2 日
更新 受管政策主題	更新AWS 的 受管政策 AWS Amplify主題，說明 Amplify 受 AWS 管政策的最新變更。	2021 年 11 月 8 日
更新 受管政策主題	更新AWS 的 受管政策 AWS Amplify主題，說明 Amplify 受 AWS 管政策的最新變更。	2021 年 9 月 27 日

變更	描述	Date
新的 受管政策主題	新增AWS 的 受管政策 AWS Amplify主題，說明 Amplify 的 AWS 受管政策以及這些政策的最新變更。	2021 年 7 月 28 日
更新伺服器端轉譯章節	更新使用 Amplify Hosting 部署伺服器端轉譯應用程式章節，說明對 Next.js 10.x.x 版和 Next.js 11 版的新支援。	2021 年 7 月 22 日
已更新設定建置設定章節	新增 設定 monorepo 組建設定主題，說明如何在使用 Amplify 部署 monorepo 應用程式時設定建置設定和新的`AMPLIFY_MONOREPO_APP_ROOT` 環境變數。	2021 年 7 月 20 日
更新功能分支部署章節	新增 Amplify 組態的自動建置時間產生 （僅限第 1 代應用程式）主題，說明如何在建置時間自動產生`aws-exports.js` 檔案。新增 條件式後端建置 （僅限第 1 代應用程式）主題，說明如何啟用條件式後端建置。新增跨應用程式使用 Amplify 後端 （僅限第 1 代應用程式）主題，說明如何在建立新應用程式、將新分支連接至現有應用程式，或更新現有前端以指向不同的後端環境時重複使用現有後端。	2021 年 6 月 30 日

變更	描述	Date
已更新安全性章節	新增Amplify 中的資料保護主題，說明如何套用共同責任模型，以及 Amplify 如何使用加密來保護靜態和傳輸中的資料。	2021 年 6 月 3 日
SSR 功能啟動的新支援	新增使用 Amplify Hosting 部署伺服器端轉譯應用程式章節，說明 Amplify 對使用伺服器端轉譯 (SSR) 並使用 Next.js 建立之 Web 應用程式的支援。	2021 年 5 月 18 日
新增安全性章節	新增Amplify 的安全性章節，說明如何在使用 Amplify 時套用共同責任模型，以及如何設定 Amplify 以符合您的安全與合規目標。	2021 年 3 月 26 日
更新自訂建置主題	更新自訂建置映像和即時套件更新主題，說明如何設定 Amazon Elastic Container Registry Public 中託管的自訂建置映像。	2021 年 3 月 12 日
更新監控主題	更新監控主題，說明如何存取 Amazon CloudWatch 指標資料並設定警示。	2021 年 2 月 2 日
新的 CloudTrail 記錄主題	新增使用 主題記錄 Amplify API 呼叫 AWS CloudTrail，描述 如何 AWS CloudTrail 擷取和記錄 AWS Amplify 主控台 API 參考和 AWS Amplify Admin UI API 參考的所有 API 動作。	2021 年 2 月 2 日

變更	描述	Date
新的 Admin UI 功能啟動	更新歡迎使用 AWS Amplify 託管主題以描述新的 Admin UI，為前端 Web 和行動開發人員提供視覺化界面，以在 外部建立和管理應用程式後端 AWS 管理主控台。	2020 年 12 月 1 日
新的效能模式功能啟動	更新管理應用程式效能主題，以描述如何啟用效能模式以最佳化 ，以提高託管效能。	2020 年 11 月 4 日
更新自訂標頭主題	更新自訂標頭主題，說明如何使用主控台或編輯 YML 檔案來定義 Amplify 應用程式的自訂標頭。	2020 年 10 月 28 日
新的自動子網域功能啟動	新增設定 Route 53 自訂網域的自動子網域主題，以描述如何針對連線至 Amazon Route 53 自訂網域的應用程式使用模式型功能分支部署。已新增具有子網域的 Web 預覽存取主題，說明如何從提取請求設定 Web 預覽，以便透過子網域存取。	2020 年 6 月 20 日
新通知主題	新增通知主題，說明如何為 Amplify 應用程式設定電子郵件通知，以在建置成功或失敗時提醒利益相關者或團隊成員。	2020 年 6 月 20 日

變更	描述	Date
更新自訂網域主題	更新連接自訂網域主題，以改善在 Amazon Route 53、GoDaddy 和 Google Domains 中新增自訂網域的程序。此更新也包含設定自訂網域的新故障診斷資訊。	2020 年 5 月 12 日
AWS Amplify 版本	此版本推出 Amplify。	2018 年 11 月 26 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。