使用者指南

AWS Amplify 託管

AWS Amplify 託管: 使用者指南

Table of Contents

歡迎來到AWS Amplify託管

AWS Amplify是一組專門打造的工具和功能，可讓前端 Web 和行動開發人員快速輕鬆地在其上建置完整堆疊應用程式。AWSAmplify 提供兩種服務：Amplify Hosting 和 Amplify Studio。Amplify Hosting 提供了一個 Git 型的工作流程，可用來託管具有連續部署的全堆疊無伺服器 Web 應用程式。本用戶指南提供了開始使用 Amplify 託管所需的信息。

Amplify 託管功能

- Amplify 託管支持常見的 SPA 框架, 例如, 反應, 角, Vue.js, 離子, 和灰燼, 以及靜態網站生成器，如蓋茨比, 十一, 雨果, 和哲基爾. VuePress
- 通過連接新分支機構來管理前端和後端的生產和預備環境。請參閱功能分支部署。
- 將您的應用程式 Connect 到自訂網域。請參閱，設定自訂網域。
- 部署和託管 SSR Web 應用程式。Amplify 主機會自動檢測使用 Next.js 框架創建的應用程序。

  Amplify 還支持任何基於 Javascript 的 SSR 框架，其中包含開放源代碼構建適配器，可將應用程序的構建輸出轉換為 Amplify 託管期望的目錄結構。您可以使用介面卡來部署 Nuxt 應用程式以 Amplify。
- 透過設定提取要求預覽來預覽程式碼檢閱期間的變更。
- 通過端對端測試提高您的應用程序質量。請參閱，end-to-end 測試。
- 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。請參閱，限制存取。
- 設置重寫和重定向，以維護 SEO 排名並根據您的客戶端應用程序要求路由流量。請參閱，使用重新導向。
- 原子部署可確保 Web 應用程式僅在整個部署完成後才更新，以消除維護時間。如此便可減少檔案無法正確上傳的情況。

開始使用 Amplify 託管

要開始使用 Amplify 的託管功能，請參閱開始使用現有程式碼教程。完成教學課程後，您將能夠連接您的 git 儲存庫 (GitHub GitLab、 BitBucket Cloud 和AWS CodeCommit) 以設定持續部署。或者，您也可以開始使用其中一個完整堆疊持續部署範例。

Amplify 工作室

您可以從AWS Amplify控制台中訪問 Amplify 工作室。AWS Management ConsoleAmplify Studio 是視覺化的開發環境，可簡化可擴展、全堆疊 Web 和行動應用程式的建立作業。使用 Studio 使用一組 UI 組件構建您的前端 ready-to-use UI，創建一個應用程序後端，然後將兩者連接在一起。請參閱 Amplify 文件中的 Amplify 工作室的使用者指南。

Amplify 工作室功能

- 視覺化資料模型可讓您專注於特定網域的物件，而非雲端基礎架構。
- 為您的應用設置身份驗證。
- 功能強大且易於理解的授權。
- 我nfrastructure-as-code 配置了所有後端功能。AWS CloudFormation
- 與 Amplify 命令列介面 (CLI) 搭配使用。您在 Studio 中進行的所有更新都可以提取到 CLI 中。
- 透過電子郵件邀請使用者設定和管理後端。這些使用者也可以使用他們的電子郵件登入 Amplify CLI。
- 具有降價支持的內容管理。
- 管理應用程式的使用者和群組。
- 使用 Studio 的視覺化設計器來構建前端 UI 組件。從預先建置的 UI 元件庫中的數十種設計中進行選擇。
- 將由設計師建立的 Figma 原型導入到工作室作為反應代碼。
- 使用主題自定義前端 UI，以將全局樣式應用於應用程序的組件。
- 直接在 Studio 中配置和測試 UI 組件，以查看它們如何更新和顯示數據。
- 只需幾個簡單的步驟，即可將雲端連線的後端繫結至前端 UI。

開始使用 Amplify 工作室

您不需要AWS帳戶即可開始使用 Studio 來創建後端。如果沒有AWS帳戶，您可以開始在本地為後端建模數據。

使用AWS帳戶，您可以訪問擴展的 Studio 功能集來管理後端環境，以及用於創建可連接到應用程序後端的前端 UI 組件的可視化設計器。如需詳細資訊，請參閱 Amplify 文件中的入門。

現代水療網頁應用程

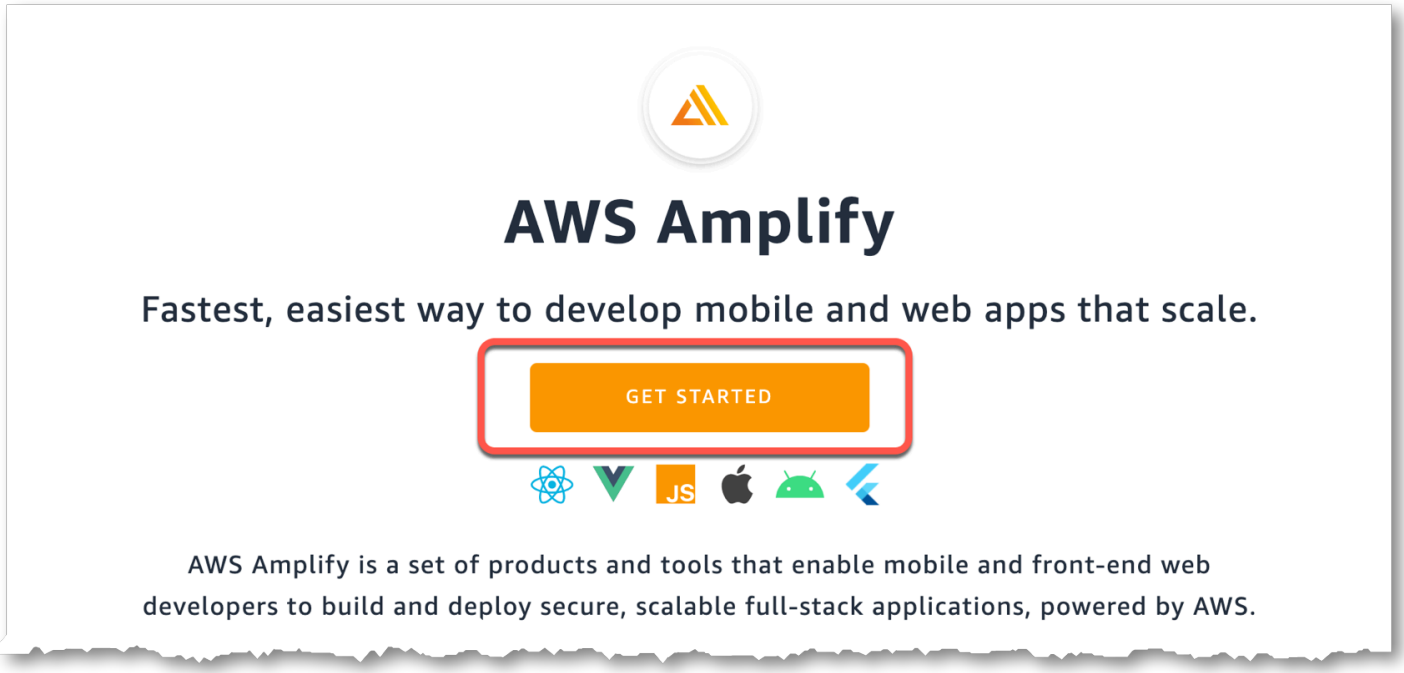
本用戶指南適用於對現代單頁 Web 應用程序（SPA）有基本了解的客戶。現代 Web 應用程序構建為 SPA，將所有應用程序組件打包到靜態文件中。傳統的客戶端-服務器 Web 體系結構導致不良的體驗; 每次點擊或搜索按鈕都需要往返服務器，重新渲染整個應用程序。現代 Web 應用程序通過將應用程序前端或用戶界面作為預構建的 HTML/JavaScript 文件有效地提供類似本地應用程序的用戶體驗，然後可以調用後端功能而無需重新加載頁面。

現代 Web 應用程序的功能通常分佈在多個地方，例如AWS Lambda數據庫，身份驗證服務，在瀏覽器中運行的前端代碼以及後端業務邏輯或在雲中運行的功能。這使得應用程式部署變得複雜且耗時，因為開發人員需要仔細協調前端和後端的部署，以避免部署或失敗。Amplify 可在單一工作流程中簡化前端和後端的部署作業。

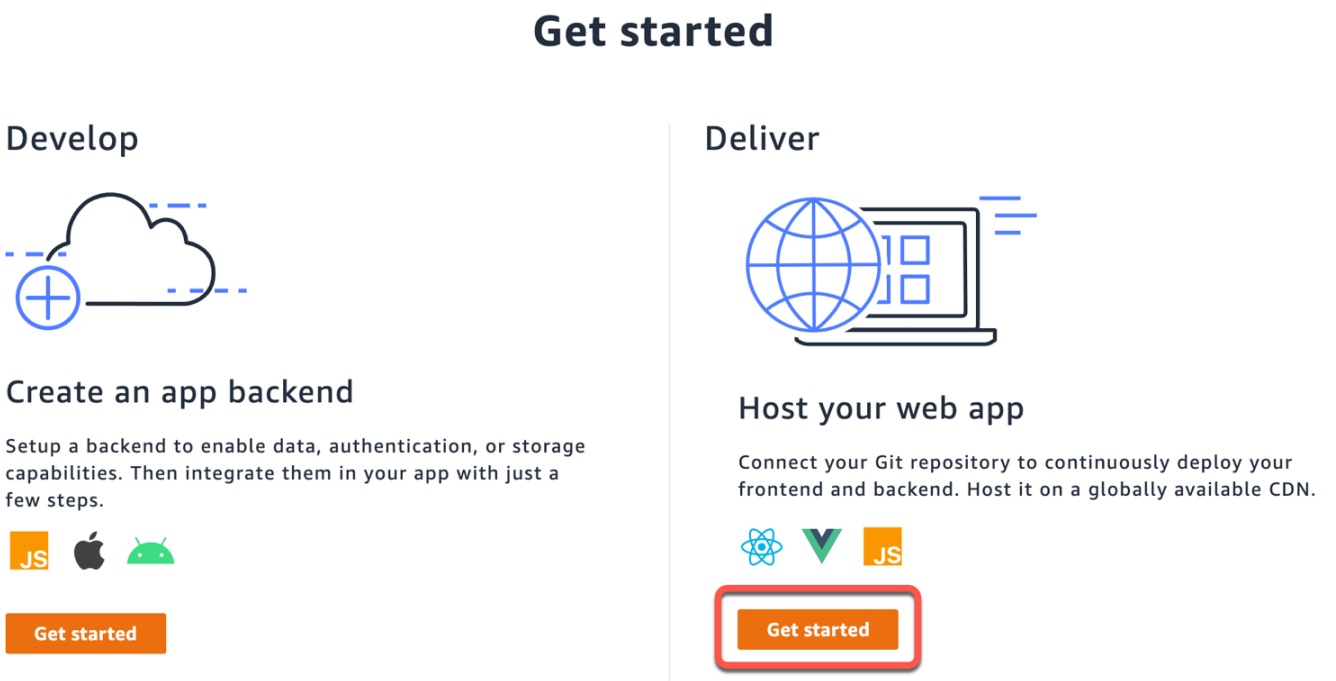
開始使用現有程式碼

在這個逐步解說中，您會了解如何持續建置、部署和託管現代 Web 應用程式。現代 Web 應用程序包括單頁應用程序（SPA）框架（例如，React，Angular 或 Vue）和靜態站點生成器（SSG）（例如雨果，哲基爾或蓋茨比）。Amplify 主機還支持使用服務器端渲染（SSR）並使用 Next.js 創建的 Web 應用程序。

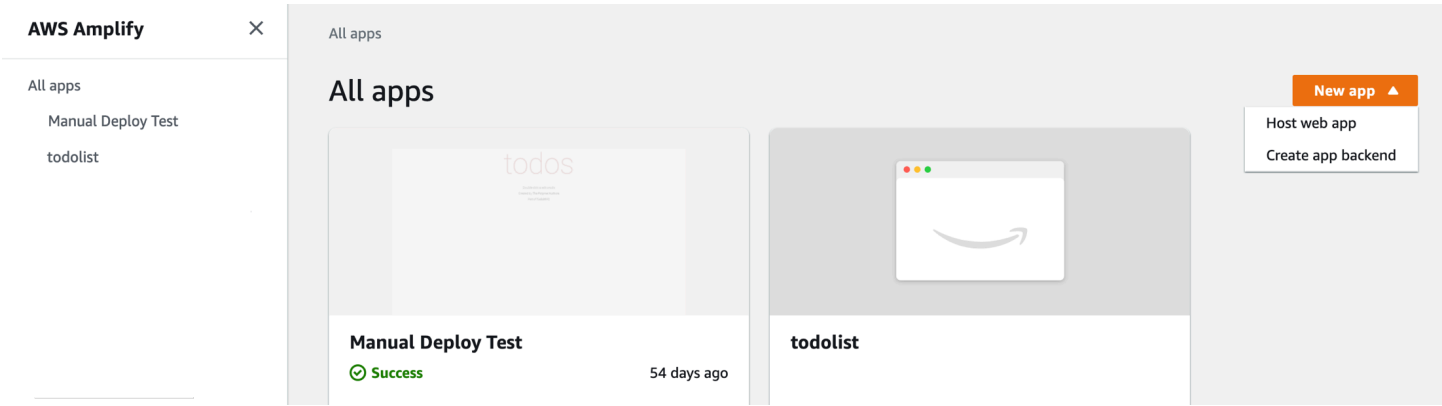
若要開始使用，請登入 Amplify 控制台。如果您是從AWS Amplify首頁開始，請選擇頁面頂端的 [開始使用]。



然後選擇「交付」下的「開始」。

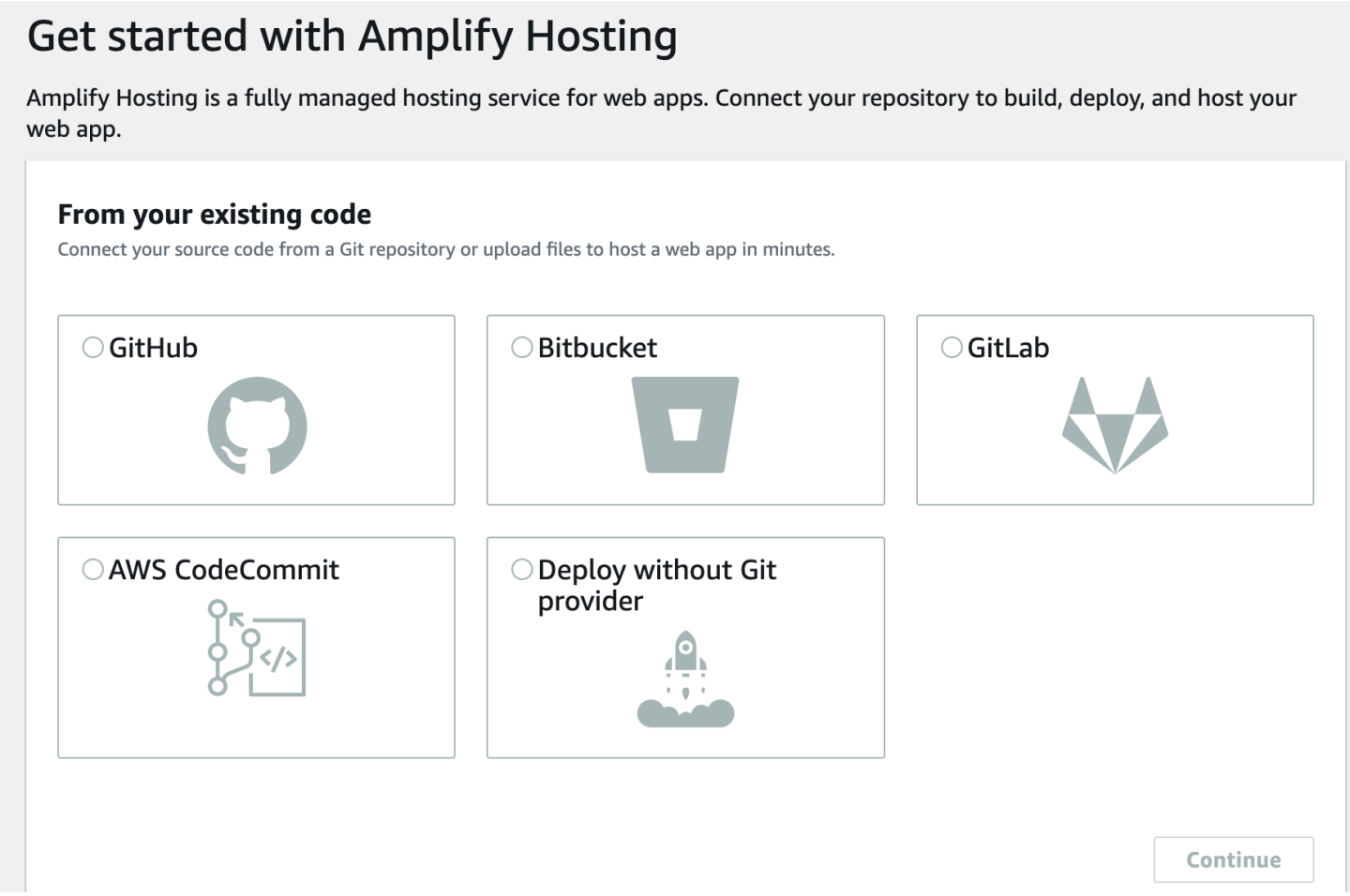


如果您是從 [所有應用程式] 頁面開始，請選擇 [新增應用程式]，然後選取右上角的 [主機 Web 應用程式]。



步驟 1：Connect 儲存庫

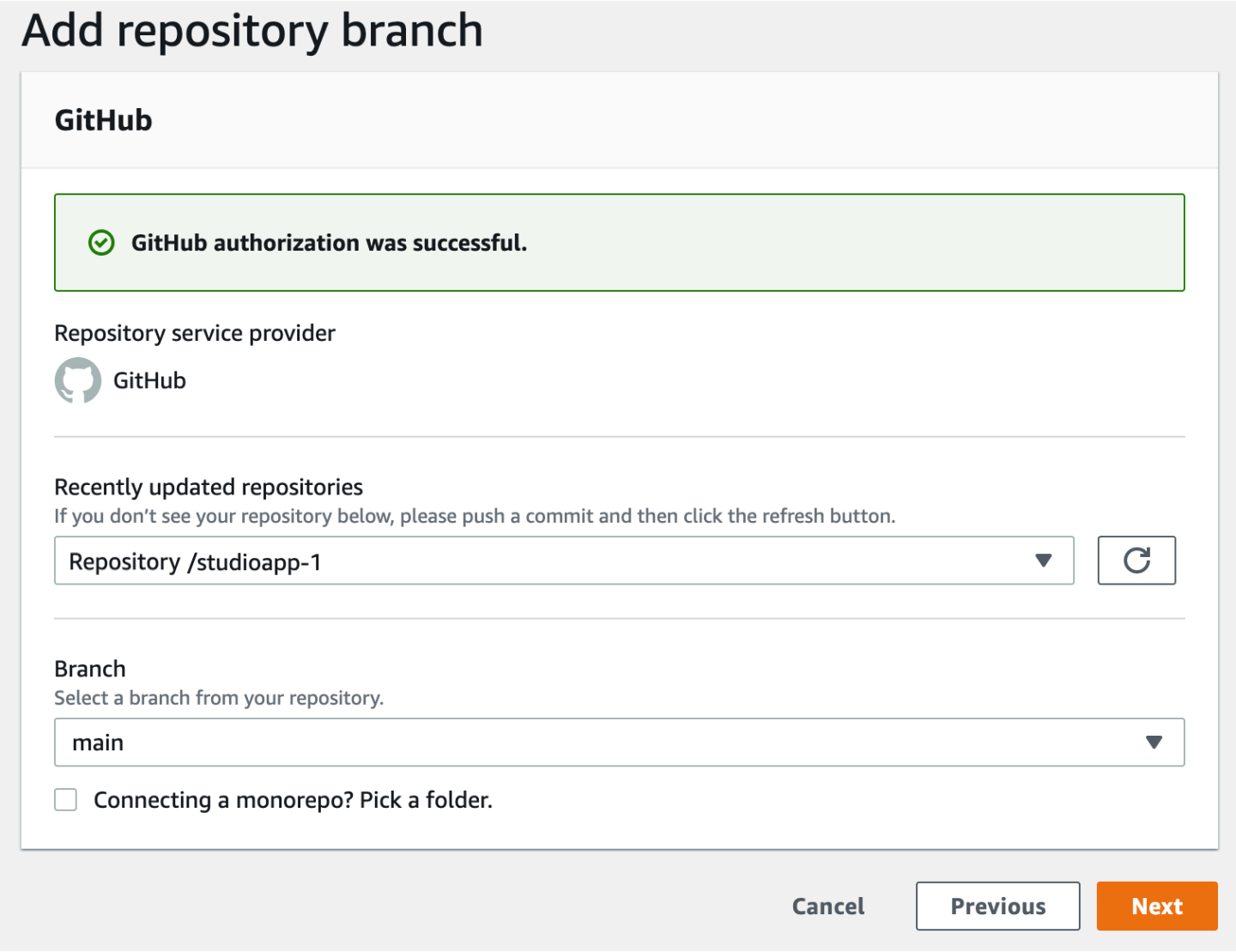
Connect 您的 GitHub，比特桶 GitLab，或AWS CodeCommit存儲庫。您也可以選擇手動上傳構建工件，而無需連接 Git 存儲庫。如需詳細資訊，請參閱手動部署。



在您使用 Bitbucket 授權 Amplify 主控台之後，或者 GitLabAWS CodeCommit，Amplify 會從儲存庫提供者擷取存取權杖，但不會將權杖儲存在伺服器上。AWSAmplify 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。

對於 GitHub 存儲庫，Amplify 現在使用應用 GitHub 程序功能來授權 Amplify 訪問權限。透過 Amplify GitHub 應用程式，權限會進行更微調，讓您僅授與您指定的存放庫的 Amplify 存取權。有關安裝和授權 GitHub 應用程序的更多信息，請參閱設定 Amplify GitHub 庫的存取。

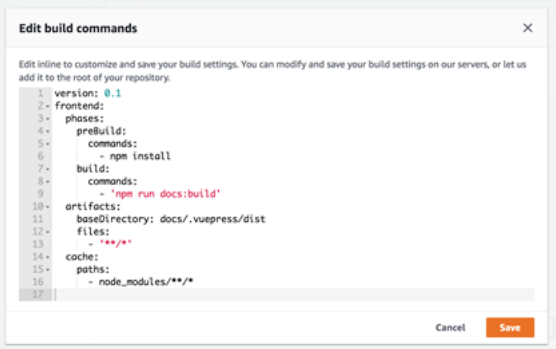
連接儲存庫服務提供者之後，請選擇儲存庫，然後選擇對應的分支來建置和部署。



步驟 2a：確認前端的構建設置

對於選定的分支，Amplify 會檢查您的存儲庫以自動檢測要運行的構建命令序列。

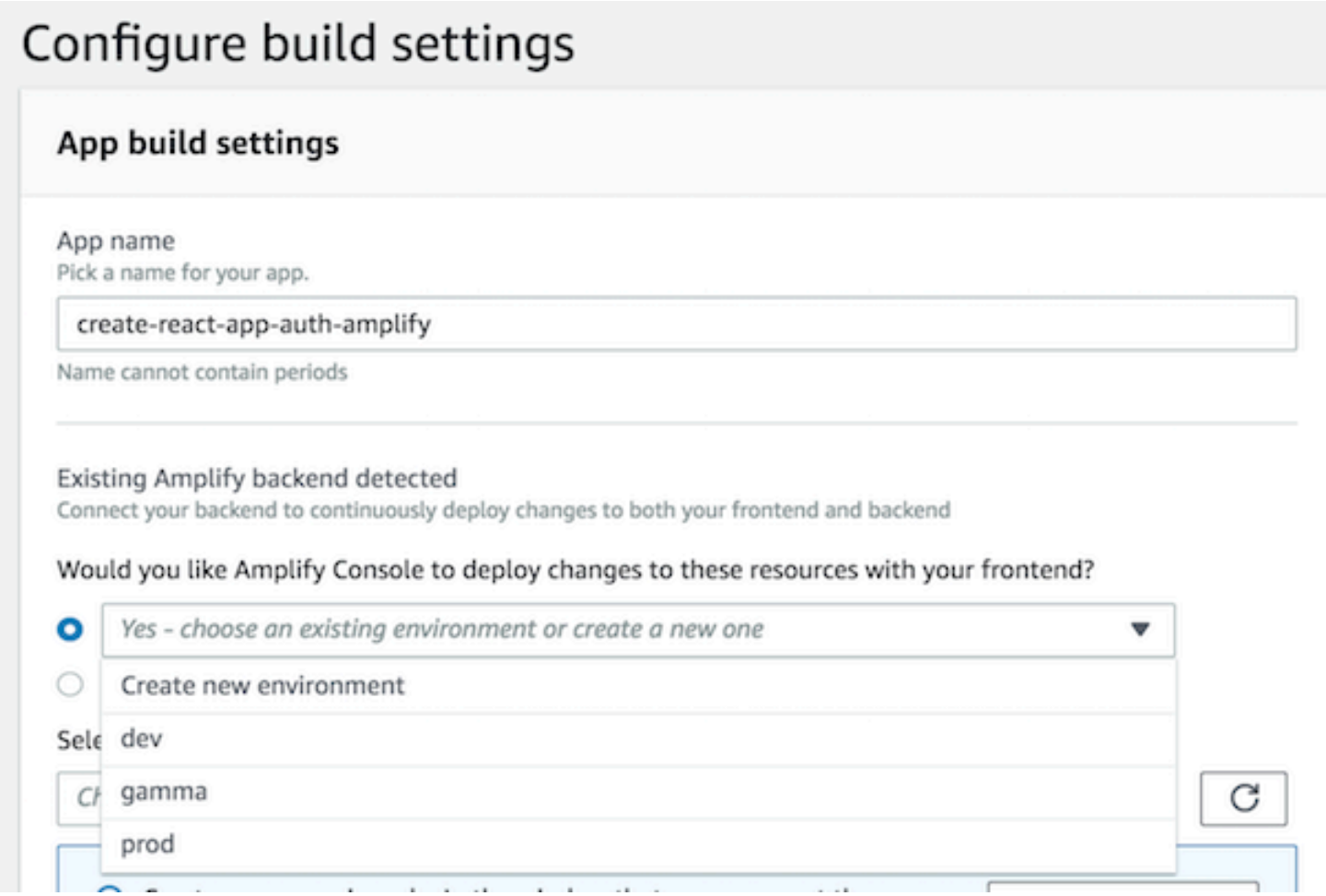
重要：驗證建置命令和建置輸出目錄 (也就是 artifacts > baseDirectory) 是準確的。如果您需要修改此資訊，請選擇 Edit (編輯) 以開啟 YML 編輯器。您可以將構建設置保存在我們的服務器上，也可以下載 YML 並將其添加到回購的根目錄中（對於壟斷，請將 YML 存儲在應用程序的根目錄中）。



如需詳細資訊，請參閱組建規格 YAML 語法。

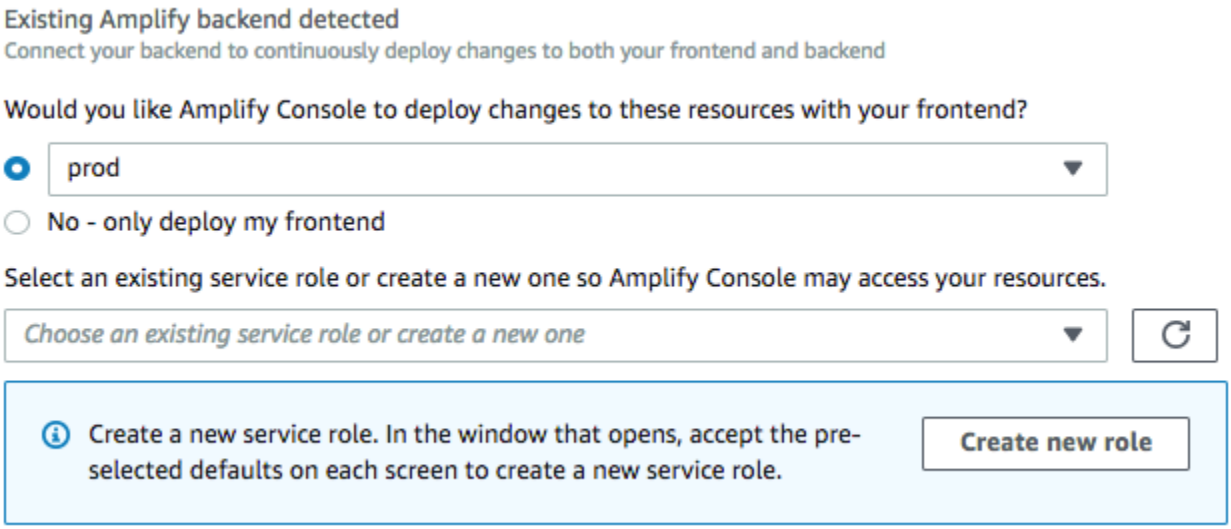
步驟 2b：確認後端的構建設置

如果您連線由 Amplify CLI v1.0+ 佈建的存放庫 (執行擴增-v 以尋找 CLI 版本)，則 Amplify 主機會在單一工作流程中部署或自動更新後端資源 (由 Amplify CLI 佈建的任何資源)，並使用前端組建。您可以選擇將現有後端環境指向分支，或建立全新的環境。如需 step-by-step 自學課程，請參閱開始使用完整堆疊部署。



若要在建置期間使用 Amplify CLI 部署後端功能，請建立或重複使用 AWS Identity and Access Management (IAM) 服務角色。IAM 角色是一種安全的方式，可授予 Amplify 權限以對帳戶中的資源採取行動。如需詳細說明，請參閱 新增服務角色。

備註：如果沒有啟用 IAM 服務角色，Amplify CLI 將無法執行。



步驟 2c：添加環境變量（可選）

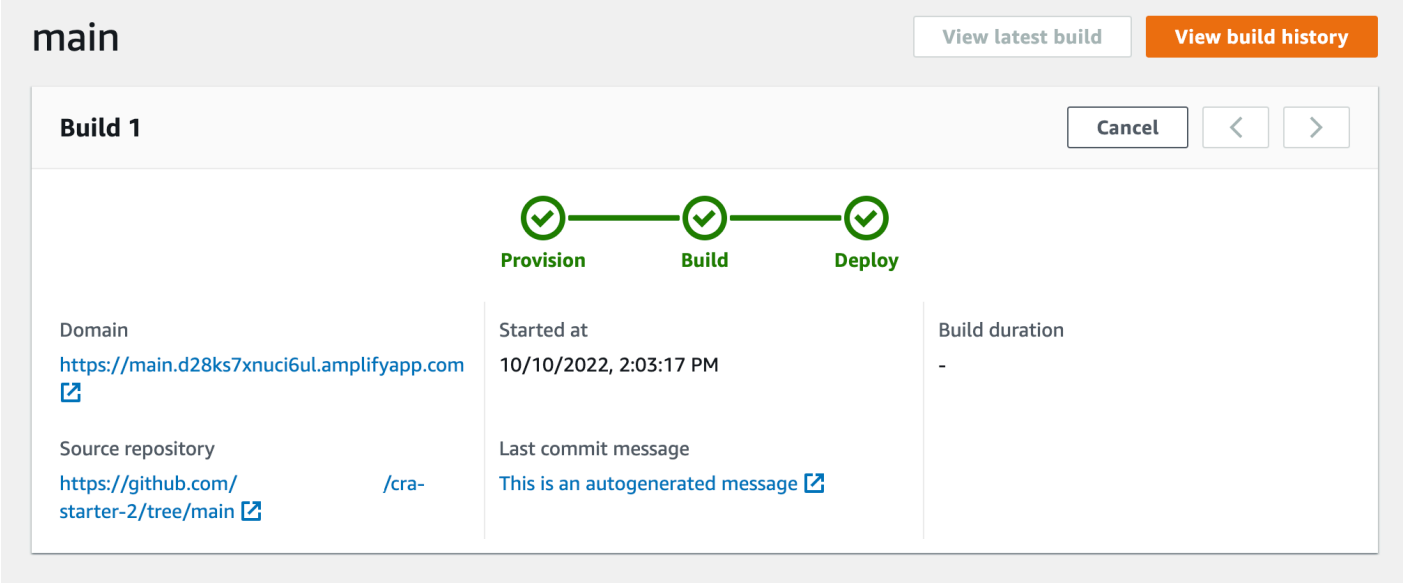
幾乎每個應用程式都需要在執行時間取得組態資訊。這些組態可以是資料庫連線詳細資訊、API 金鑰或是不同的參數。環境變量提供了一種在構建時公開這些配置的方法。

步驟 3：儲存並部署

檢閱您的所有設定，以確保一切都正確設定。選擇 [儲存並部署]，將您的 Web 應用程式部署到AWS全球內容傳遞網路 (CDN)。您的前端組建通常需要 1 到 2 分鐘，但可能會根據應用程式的大小而有所不同。

通過在分支部分選擇進度指示器來訪問構建日誌屏幕。建置有以下階段：

1. 佈建 - 您的建置環境是在具有 4 個 vCPU、7 GB 記憶體的主機上使用 Docker 映像設定。每個建置會有自己的主機執行個體，以確保所有資源安全地隔離。系統會顯示 Docker 檔案的內容，以確保預設的映像支援您的需求。
2. 建置 - 建置階段包含三個階段：設定 (複製儲存庫到容器)、部署後端 (執行 Amplify CLI 來部署後端資源)，以及建置前端 (建置您的前端成品)。
3. 部署-當構建完成時，所有成品都會部署到由 Amplify 託管管理的託管環境中。您可以在`amplifyapp.com`網域上檢視您的應用程式。每個部署不可分割 - 不可分割部署透過確保 Web 應用程式只有在整個部署完成後才會更新，因此不需維護時段。



> **Note**
>
> 為了增強您的 Amplify 應用程式的安全性，請在公用尾碼清單 (PSL) 中註冊了 amplifyapp.com 網域。為了進一步的安全性，如果您需要在 Amplify 應用程式的預設網域名稱中設定敏感性 Cookie，建議您使用__Host-前置詞的 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。

後續步驟

- 將自訂網域新增至您的應用程式
- 管理多個環境
- 合併前預覽提取請求

開始使用完整堆疊持續部署

Amplify Hosting 可讓開發人員使用 Amplify 架構建置應用程式，在每次程式碼提交時持續將更新部署到其後端和前端。使用 Amplify 主機，您可以在與前端程式碼相同的提交上，使用 Graphql/Rest API、驗證、分析和儲存裝置部署無伺服器後端。

在本教學課程中，您將使用「Amplify」來設定完整堆疊 CI/CD 工作流程。您將部署一個前端應用程序來 Amplify 託管。然後，您將創建使用 Amplify 工作室後端。最後，您將雲端後端連接到前端應用程序。

主題

必要條件

在開始本教程之前，您需要執行以下操作：

- 註冊 AWS 帳戶。打開 https://portal.aws.amazon.com/billing/signup#/start/email 以開始使用。
- 使用 git 儲存庫提供者建立帳戶 GitHub，例如 GitLab，Bitbucket 或 AWS CodeCommit。
- 安裝 Amplify 命令列介面 (CLI)。如需指示，請參閱擴大架構文件中的安裝 Amplify CLI。

步驟 1：部署前端

如果您要在此範例中使用的 git 儲存庫中有現有的前端應用程式，您可以繼續部署前端應用程式的指示。

如果你需要創建一個新的前端應用程序來用於這個例子，你可以按照創建 React 應用程序文檔中的創建反應應用程序說明進行操作。

部署前端應用程式

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 在 [所有應用程式] 頁面上，選擇 [新增應用程式]，然後選取右上角的 [主機 Web 應用程式]
3. 選取您的 GitHub、Bitbucket 或 AWS CodeCommit 儲存庫提供者 GitLab，然後選擇 [繼續]。
4. Amplify 授權訪問您的 git 存儲庫。對於 GitHub 存儲庫，Amplify 現在使用應用 GitHub 程序功能來授權 Amplify 訪問權限。

   有關安裝和授權 GitHub 應用程序的更多信息，請參閱設定 Amplify GitHub 庫的存取。
5. 在「新增儲存區域分支」頁面上執行下列動作：

   a. 在「最近更新的儲存庫」清單中，選取要連線的存放庫名稱。

   b. 在「分支」清單中，選取要連線的存放庫分支名稱。

   c. 選擇下一步。
6. 在 [設定組建設定] 頁面上，選擇 [下一步]。
7. 在「複查」頁面上，選擇「儲存並部署」。部署完成後，您可以在`amplifyapp.com`預設網域上檢視您的應用程式。

> **Note**
>
> 為了增強您的 Amplify 應用程式的安全性，請在公用尾碼清單 (PSL) 中註冊了 amplifyapp.com 網域。為了進一步的安全性，如果您需要在 Amplify 應用程式的預設網域名稱中設定敏感性 Cookie，建議您使用`__Host-`前置詞的 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，您已經將前端應用程序部署到 Amplify 託管，您可以創建一個後端。使用下列指示建立具有簡單資料庫和 GraphQL API 端點的後端。

若要建立後端

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
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」標籤，然後選擇「開始使用」。這會啟動預設暫存環境的設定程序。
4. 設定完成後，選擇啟動工作室以存取安培工作室中的預 Amplify 後端環境。

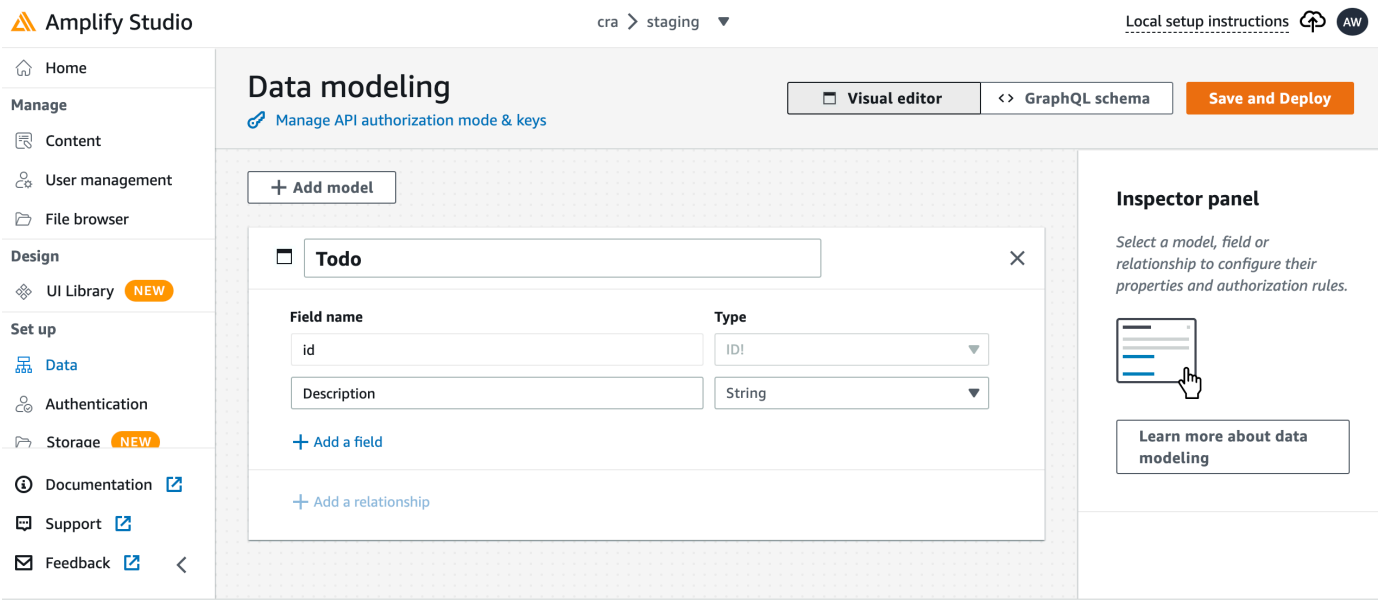
Amplify Studio 是一個可視化界面來創建和管理您的後端，並加速您的前端 UI 開發。如需有關 Amplify 工作室的詳細資訊，請參閱 Amplify 工作室文件。

使用下列指示建立使用 Amplify Studio 視覺化後端產生器介面的簡單資料庫。

建立資料模型

1. 在應用程式測試環境的首頁上，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. 對於標題，請輸入**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師中的外觀範例。



6. 選擇 [儲存並部署]。
7. 返回 Amplify 主控台，暫存環境部署將會進行中。

在部署期間，Amplify Studio 會在後端建立所有必要的 AWS 資源，包括用於存取資料的 AWS AppSync GraphQL API，以及用來託管待辦事項項目的 Amazon DynamoDB 表格。Amplify 用 AWS CloudFormation 於部署後端，這使您可以將後端定義存儲為 infrastructure-as-code.

步驟 3：將後端 Connect 到前端

現在，您已經部署了前端並創建了包含數據模型的雲端後端，您需要將它們連接起來。使用下列指示，透過 Amplify CLI 將您的後端定義拉到本機應用程式專案。

將雲端後端連線至本機前端

1. 打開終端窗口並導航到本地項目的根目錄。
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。

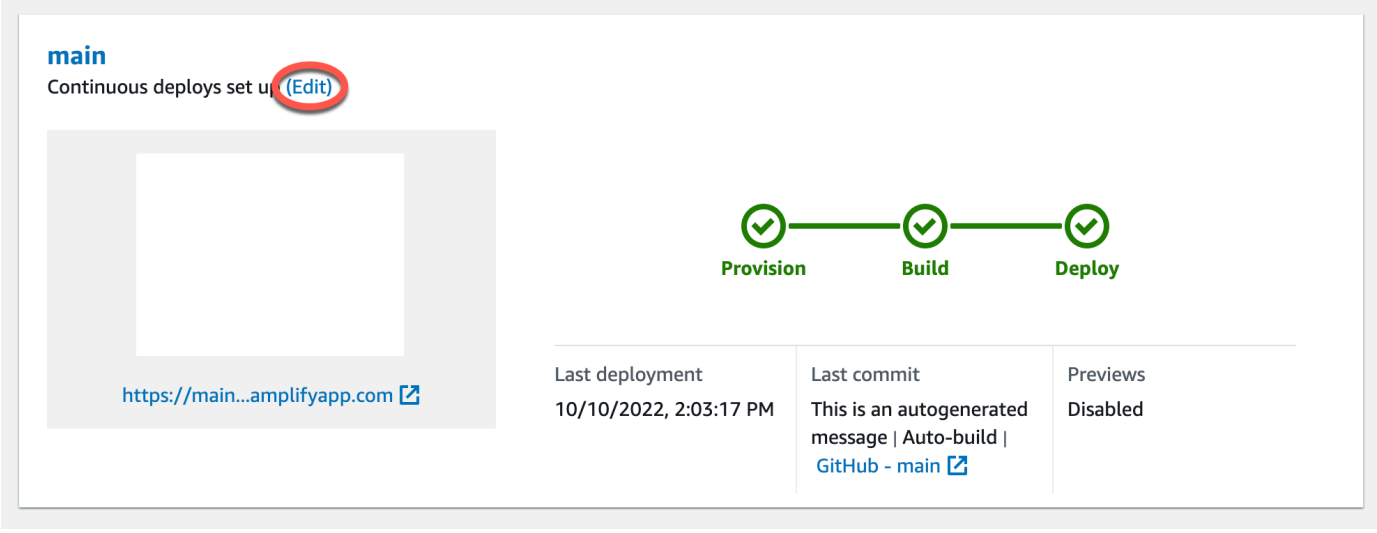
```
amplify pull --appId abcd1234 --envName staging
```

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，將後端新增至持續部署工作流程。使用以下說明將前端分支與 Amplify 主機控制台中的後端連接。

連接前端應用程式分支和雲端後端

1. 在應用程序首頁上，選擇託管環境選項卡。
2. 找到主分支，然後選擇 「編輯」。



3. 在「編輯目標後端」視窗中，對於「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。

   依預設，會啟用完整堆疊 CI/CD。取消勾選此選項可關閉此後端的完整堆疊 CI/CD。關閉全堆疊 CI/CD 會導致應用程式在僅提取模式下執行。在建置階段，Amplify 只會自動產生`aws-exports.js`檔案，而不會修改後端環境。
4. 接下來，您必須設定服務角色，以授予 Amplify 變更應用程式後端所需的權限。您可以使用現有的服務角色，也可以建立新的服務角色。如需說明，請參閱新增服務角色。
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. 要完成將登台後端連接到前端應用程序的主分支，請執行項目的新構建。

   執行以下任意一項：

   - 從您的 git 存儲庫中，推送一些代碼以在 Amplify 控制台中啟動構建。
   - 在 Amplify 主控台中，導覽至應用程式的組建詳細資料頁面，然後選擇 [重新部署此版本]。

後續步驟

設定功能分支部署

遵循我們建議的工作流程，在多個後端環境中設定功能分支部署。

在 Amplify 工作室中創建前端 UI

使用 Studio 使用一組 UI 組件構建您的前端 ready-to-use UI，然後將其連接到您的應用程序後端。如需詳細資訊和教學課程，請參閱擴增架構文件中的 Amplify Studio 的使用者指南。

使用 Amplify 主機部署伺服器端轉譯應用程

您可以用 AWS Amplify 來部署和託管使用伺服器端轉譯 (SSR) 的 Web 應用程式。Amplify 主機會自動偵測使用 Next.js 架構建立的應用程式，而且您不需要在 AWS Management Console. Amplify 還支持任何基於 Javascript 的 SSR 框架，其中包含開放源代碼構建適配器，可將應用程序的構建輸出轉換為 Amplify 託管期望的目錄結構。

若要瞭解 Amplify 如何支援 SSR，請檢閱下列主題。

主題

什麼是服務器端渲染

Amplify 支援部署和託管使用單一頁面應用程式 (SPA) 架構 (例如 React) 建立的靜態 Web 應用程式，以及使用靜態網站產生器 (SSG) (例如 Gatsby) 建立的應用程式。靜態 Web 應用程式包含儲存在內容傳遞網路 (CDN) 上的 JavaScript 檔案 (例如 HTML、CSS 和檔案) 的組合。當客戶端瀏覽器向網站發出請求時，服務器將帶有 HTTP 響應的頁面返回給客戶端，客戶端瀏覽器解釋該內容並將其顯示給用戶。

Amplify 也支援具有伺服器端轉譯 (SSR) 的 Web 應用程式。當客戶端向 SSR 頁面發送請求時，會在每個請求的服務器上創建頁面的 HTML。SSR 使開發人員能夠自定義每個請求和每個用戶的網站。此外，SSR 可以改善網站的性能和搜索引擎優化（SEO）。

Amplify 對 SSR 框架的支援

Amplify 主機支援任何 JavaScript 基於 SSR 架構，其部署套件包符合 Amplify 預期的建置輸出。Amplify Hosting 提供了一項部署規範，可將檔案和目錄結構標準化，以便針對任何 SSR 架構輸出應用程式組建。

架構作者可以使用以檔案系統為基礎的部署規格，開發針對其特定架構自訂的開放原始碼建置配接 這些適配器將應用程序的構建輸出轉換為符合 Amplify Hosting 預期目錄結構的部署包。此部署包將包括所有必要的文件和資產來託管應用程序，包括運行時配置，例如路由規則。

如果您不使用框架或框架適配器，則可以開發自己的解決方案來生成符合 Amplify Hosting 預期目錄結構的部署包。

Amplify 託管支持以下原語：靜態資產，計算，圖像優化和路由規則。您可以利用這些原語來部署具有更豐富功能的應用程序。如需每個基本元件的詳細資訊，請參閱Amplify SSR 原始支持。

您可以從下列案例中選擇，開始將 SSR 應用程式部署到 Amplify。

部署一個 Next.js 應用程式

Amplify 支援使用 Next.js 建立的應用程式，不需要在主控台中使用介面卡或手動設定。如需詳細資訊，請參閱 Amplify 對 Next.js SSR 的支援。

部署使用架構配接器的應用程式

您可以參考任何可用的開放原始碼架構配接器，將 SSR 應用程式部署到「Amplify 主機」。如需詳細資訊，請參閱 使用框架適配器。

Nuxt 架構可使用配接器。如需有關使用此轉接器的詳細資訊，請參閱 Nuxt 文件。

建置架構轉接器

想要整合架構提供的功能的架構作者可以使用 Amplify 主機部署規格來設定您的組建輸出，以符合 Amplify 預期的結構。如需詳細資訊，請參閱 使用部署資訊清單部署快速伺服器。

設定建置後指令碼

您可以使用 Amplify 主機部署規格，根據特定案例的需要操作組建輸出。如需詳細資訊，請參閱 使用「Amplify 主機」部署規格來設定組建輸出。如需範例，請參閱 使用部署資訊清單部署快速伺服器。

部署 SSR 應用程式以 Amplify

您可以使用本主題中的指示，部署使用符合 Amplify 預期之組建輸出的部署服務包的任何架構建立的應用程式。如果您要部署 Next.js 應用程式，則不需要介面卡。

如果您要部署使用架構介面卡的 SSR 應用程式，您必須先安裝並設定介面卡。如需說明，請參閱使用框架適配器。

若要部署 SSR 應用程式以 Amplify 主機

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 在 [所有應用程式] 頁面上，選擇 [新增應用程式]，然後選擇 [主機
3. 選取您的 GitHub、Bitbucket 或 AWS CodeCommit 儲存庫提供者 GitLab，然後選擇 [繼續]。
4. 在「新增儲存區域分支」頁面上，執行下列動作：
   a. 在「最近更新的儲存庫」清單中，選取要連線的存放庫名稱。
   b. 在「分支」清單中，選取要連線的存放庫分支名稱。
   c. 選擇下一步。
5. 在 [建置設定] 頁面上，[Amplify] 會自動偵測 Next.js SSR 應用程式。如果您部署的 SSR 應用程式使用另一個架構的適配器，則必須明確啟用 Amazon CloudWatch Logs。在伺服器端轉譯 (SSR) 部署區段中，選擇啟用 SSR 應用程式記錄。
6. 此應用程式需要 IAM 服務角色，Amplify 假設將記錄傳遞給您的 AWS 帳戶. 您可以允許 Amplify 託管自動為您創建服務角色，也可以指定已創建的角色。
   - 允許 Amplify 自動建立角色並將其附加至您的應用程式
     - 在「IAM 角色」區段中，選擇「建立並使用新的服務角色」。
   - 若要附加先前建立的服務角色
     a. 在「IAM 角色」區段中，選擇「使用現有的服務角色」。
     b. 從清單中選擇要使用的角色。
7. 選擇下一步。
8. 在「複查」頁面上，選擇「儲存並部署」。

使用框架適配器

您可以安裝和使用任何為與 Amplify 主機整合而建立的 SSR 架構建置配接器。每個提供轉接器的架構都會決定介面卡的配置方式，以及如何連接到其建置流程。一般而言，您會將介面卡安裝為 npm 開發相依性。

使用架構建立應用程式之後，請使用架構的文件來學習如何安裝 Amplify Hosting 配接器，並在應用程式的設定檔中進行設定。

接下來，在項目的根目錄中創建一個amplify.yml文件。在amplify.yml檔案中，將設定baseDirectory為應用程式的建置輸出目錄。架構會在建置程序期間執行介面卡，以將輸出轉換為 Amplify 主機部署服務包。

build 輸出目錄的名稱可以是任何名稱，但.amplify-hosting文件名具有重要意義。Amplify 首先尋找定義為. baseDirectory 如果存在，「Amplify」會在那裡尋找構建輸出。如果目錄不存在，Amplify 會尋找內部的組建輸出.amplify-hosting，即使客戶尚未定義該目錄也是如此。

以下是應用程序的構建設置的示例。設定baseDirectory為.amplify-hosting以指出組建輸出位於.amplify-hosting資料夾中。只要.amplify-hosting資料夾的內容符合 Amplify 主機部署規格，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
```

將應用程序配置為使用框架適配器後，您可以將其部署到 Amplify 託管。如需詳細說明，請參閱部署 SSR 應用程式以 Amplify。

使用「Amplify 主機」部署規格來設定組建輸出

使用「Amplify」部署規格，為您要與「擴 Amplify 主機」整合的 SSR 架構設定組建輸出。如果您是架構作者，則可以使用部署規格來瞭解如何建構 Amplify 預期的組建輸出。如果您不使用架構，您可以開發自己的解決方案，以產生 Amplify 預期的組建輸出。

Amplify 主機部署規範

Amplify 主機部署規格是以檔案系統為基礎的規格，可定義目錄結構，以促進 Amplify 主機的部署。框架可以生成此預期的目錄結構作為其構建命令的輸出，從而使框架能夠利用 Amplify Hosting 的服務原語。Amplify 主機了解部署包的結構並相應地進行部署。

以下是 Amplify 預期部署服務包的資料夾結構範例。在高層級，它有一個名為的資料夾static、一個名為的資料夾compute和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 原始支持

Amplify 主機部署規格定義了一份與下列原語密切對應的合約。

靜態資產

為框架提供託管靜態文件的能力。

運算

提供能夠在連接埠 3000 上執行 Node.js HTTP 伺服器的架構。

影像最佳化

為框架提供服務，以便在運行時優化圖像。

路由規則

為框架提供一種機制，以將傳入的請求路徑映射到特定目標。

該.amplify-hosting/static目錄

您必須將要從應用程式 URL 提供的所有可公開存取的靜態檔案放在.amplify-hosting/static目錄中。此目錄中的文件通過靜態資產原始提供。

靜態檔案可從應用程式 URL 的根 (/) 存取，而不會對其內容、檔案名稱或副檔名進行任何變更。此外，子目錄會保留在 URL 結構中，並顯示在檔案名稱之前。作為一個例子，.amplify-hosting/static/favicon.ico將從那裡提供服務，https://myAppId.amplify-hostingapp.com/favicon.ico並.amplify-hosting/static/_nuxt/main.js將從https://myAppId.amplify-hostingapp.com/_nuxt/main.js

如果框架支持修改應用程序的基本路徑的能力，它必須在目錄中的靜態資產前面添加基本路徑。.amplify-hosting/static例如，如果基本路徑是/folder1/folder2，那麼調用的靜態資產的構建輸出main.css將是.amplify-hosting/static/folder1/folder2/main.css。

該.amplify-hosting/compute目錄

單一計算資源由目錄default中包含的單一子目.amplify-hosting/compute錄表示。路徑是.amplify-hosting/compute/default。該計算資源映射到 Amplify 託管的計算原始。

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，才能做為計算資源的進入點。
- 入口點檔案必須是 Node.js 模組，而且它必須啟動接聽連接埠 3000 的 HTTP 伺服器。
- 您可以將其他檔案放置在default子目錄中，並從入口點檔案中的程式碼參照它們。
- 子目錄的內容必須是獨立的。入口點模塊中的代碼無法引用子目錄外的任何模塊。請注意，框架可以以他們想要的任何方式捆綁他們的 HTTP 服務器。如果可以從子目錄內使用node server.js命令 (其中server.js is是項目檔案的名稱) 啟動計算程序，則 Amplify 會將目錄結構視為符合部署規格。

Amplify 主機服務包，並將default子目錄內的所有檔案部署到佈建的計算資源。每個計算資源都會配置 512 MB 的暫時儲存空間。此儲存空間不會在執行個體之間共用，但會在相同執行執行個體內的後續叫用之間共用。執行執行個體的執行時間上限為 15 分鐘，而執行執行個體中唯一可寫入的路徑是目/tmp錄。每個計算資源服務包的壓縮大小不能超過 220 MB。例如，壓縮時.amplify/compute/default子目錄不能超過 220 MB。

.amplify-hosting/deploy-manifest.json 檔案

使用該deploy-manifest.json檔案儲存部署的規劃詳細資料和中繼資料。deploy-manifest.json檔案至少必須包含version屬性、指定全部捕捉路由的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料和用法。

使用版本屬性

version屬性會定義您所實作之部署規格的版本。目前，Amplify 主機部署規格的唯一版本是第 1 版。下面的 JSON 示例演示了version屬性的用法。

```
"version": 1
```

使用路由屬性

該routes屬性使框架能夠利用 Amplify 託管路由規則原始。路由規則提供一種機制，可將傳入的要求路徑路由至部署服務包中的特定目標。路由規則只會指定傳入要求的目的地，而且會在透過重新寫入和重新導向規則轉換要求之後套用。有關 Amplify 主機如何處理重寫和重定向的更多信息，請參閱。使用重定向

路由規則不會重寫或轉換請求。如果傳入的請求與路由的路徑模式匹配，則該請求將按原樣路由到路由的目標。

routes陣列中指定的路由規則必須符合下列規則。

- 必須指定「捕捉全部」路線。捕獲所有路由具有匹配所有傳入請求的/*模式。
- routes陣列最多可包含 25 個項目。
- 您必須指定Static路線或Compute路線。
- 如果您指定Static路由，則該.amplify-hosting/static目錄必須存在。

- 如果您指定Compute路由，則該.amplify-hosting/compute目錄必須存在。
- 如果您指定ImageOptimization路線，您還必須指定Compute路線。這是必要的，因為純靜態應用程式尚不支援影像最佳化。

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	描述
路徑	字串	是	定義匹配傳入請求路徑（不包括查詢字符串）的模式。 最大路徑長度為 255 個字元。 路徑必須以正斜線開頭/。 路徑可以包含下列任何字元：[A-Z]、[a-z]、[0-9]、[_-.*$/~ "'@: +]。 對於模式比對，僅支援下列萬用字元： • *（匹配 0 個或更多字符） • 該/*模式稱為全部捕獲模式，並將

金鑰	Type	必要	描述
			匹配所有傳入的請求。
目標	目標	是	定義要將匹配請求路由到的目標的對象。 如果指定了Compute路由，則ComputeResource 必須存在相應的路由。 如果指定了ImageOptimization 路由，也imageSettings 必須指定。
撤退	目標	否	定義要在原始目標傳回 404 錯誤時回復目標的目標的物件。 對於指定的路由，fallback種類和種類不能相同。target例如，不允許從Static到Static的後援。只有沒有主體的 GET 要求才支援回退。如果主體存在於請求中，它將在後備期間丟棄。

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	描述
種	目標實物	是	enum定義目標類型的。有效值為 Static、Compute 和 ImageOpti mization 。
src	字串	是的 Compute 否為其他原語	字串；指定部署套裝軟體中包含原始可執行程式碼的子目錄名稱。有效且僅適用於計算原始元件。 此值必須指向部署服務包中存在的其中一個計算資源。目前，此欄位唯一支援的值為default。
緩存控制	字串	否	字串；指定要套用至回應的快取控制標頭值。僅對靜態和ImageOptimization基元有效。 指定的值會由自訂標頭覆寫。如需「Amplify 主機」客戶標

金鑰	Type	必要	描述
			頭的詳細資訊，請參閱自訂標頭。 **Note** 此快取控制標頭僅適用於狀態碼設定為 200 (OK) 的成功回應。

下列物件定義示範`TargetKind`列舉的用法。

```
enum TargetKind {
  Static = "Static",
  Compute = "Compute",
  ImageOptimization = "ImageOptimization"
}
```

下面的列表指定`TargetKind`枚舉的有效值。

靜態

路由請求到靜態資產原始。

運算

將要求路由至計算原始元件。

ImageOptimization

將請求路由到圖像優化原始文件。

下列 JSON 範例示範具有指定多個路由規則之`routes`屬性的用法。

```
"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"
      }
    }
```

```
  ]
```

如需有關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作法

使用計算資源屬性

此computeResources屬性可讓架構提供有關已佈建之計算資源的中繼資料。每個計算資源都必須具有與其相關聯的對應路由。

下列物件定義示範ComputeResource物件的用法。

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

type ComputeRuntime = 'nodejs16.x' | 'nodejs18.x' | 'nodejs20.x';
```

下表說明ComputeResource物件的屬性。

金鑰	Type	必要	描述
name	字串	是	指定計算資源的名稱。名稱必須與中子目錄的名稱相符。.amplify-hosting/compute directory 對於部署規格的第 1 版，唯一的有效值為 default。
runtime	ComputeRuntime	是	定義佈建之計算資源的執行階段。

金鑰	Type	必要	描述
			有效值為 nodejs16.x 、nodejs18.x 和 nodejs20.x 。
入口點	字串	是	針對指定的計算資源，指定程式碼將從中執行的起始檔案名稱。檔案必須存在於代表計算資源的子目錄內。

如果您的目錄結構如下所示。

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

computeResource屬性的 JSON 看起來如下所示。

```
"computeResources": [
    {
      "name": "default",
      "runtime": "nodejs16.x",
      "entrypoint": "index.js",
    }
  ]
```

使用圖像設定屬性

該imageSettings屬性使框架能夠自定義圖像優化原始的行為，在運行時提供圖像的按需優化。

下列物件定義示範ImageSettings物件的用法。

```
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	描述
大小	編號 []	是	支援的影像寬度陣列。
domains	字符串 []	是	允許的外部網域陣列，可以使用影像最佳化。將陣列保留空白，只允許部署網域使用影像最佳化。
遠端模式	RemotePattern[]	是	允許的外部模式陣列，可以使用影像最佳化。與域類似，但使用正則表達式（regex）提供了更多控制。
格式	ImageFormat[]	是	允許的輸出圖像格式的數組。
最低金霉毒素	Number	是	最佳化影像的快取持續時間 (以秒為單位)。
危險性允許 SVG	Boolean	是	允許 SVG 輸入圖片網址。出於安全目的，默認情況下禁用此功能。

下列物件定義示範RemotePattern物件的用法。

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

下表說明RemotePattern物件的屬性。

金鑰	Type	必要	描述
protocol	字串	否	允許遠端病毒碼的通訊協定。 有效值為 http 或 https。
hostname	字串	是	允許遠端病毒碼的主機名稱。 您可以指定常值或萬用字元。單一 `*` 與單一子網域相符。雙「**`」符合任意數量的子網域。Amplify 不允許僅指定 `**` 的總括萬用字元。
port	字串	否	允許遠端病毒碼的連接埠。
路徑名稱	字串	否	允許遠端病毒碼的路徑名稱。

下面的例子演示了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
  }
```

使用框架屬性

使用framework屬性來指定框架元數據。

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	描述
name	字串	是	框架的名稱。
version	字串	是	框架的版本。

金鑰	Type	必要	描述
			它必須是一個有效的語義版本控制（semver）字符串。

設定路由規則的最佳作法

路由規則提供一種機制，可將傳入的要求路徑路由至部署服務包中的特定目標。在部署服務包中，框架作者可以將文件發送到部署到以下任一目標的構建輸出：

- 靜態資產原始-文件包含在目`.amplify-hosting/static`錄中。
- 計算原始 — 檔案包含在目`.amplify-hosting/compute/default`錄中。

架構作者也會在部署資訊清單檔案中提供路由規則陣列。陣列中的每個規則都會依序遍歷順序與傳入的要求進行比對，直到有相符項目為止。當有相符規則時，要求會路由至相符規則中指定的目標。您可以選擇性地為每個規則指定後援目標。如果原始目標傳回 404 錯誤，則會將要求路由至後援目標。

部署規格要求遍歷順序中的最後一個規則必須是全部捕捉規則。擷取`/*`全部規則會使用路徑指定。如果傳入的要求與路由規則陣列中的任何先前路由不相符，則要求會路由至 Catch-All 規則目標。

對於 SSR 框架Nuxt.js，所有捕獲的規則目標必須是計算原始。這是因為 SSR 應用程式具有伺服器端轉譯的頁面，其中包含在建置階段無法預測的路由。例如，如果一個Nuxt.js應用程序有一個頁面，`/blog/[slug]`其中`[slug]`是一個動態路由參數。「全部捕捉」規則目標是將要求路由傳送至這些頁面的唯一方法。

相反，特定路徑模式可用於定位在構建時已知的路由。例如，從`/_nuxt`路徑Nuxt.js提供靜態資源。這意味著該`/_nuxt/*`路徑可以通過將請求路由到靜態資產原始的特定路由規則定位。

公用資料夾路由

大多數 SSR 框架都提供了從`public`文件夾中提供可變靜態資產的功能。像`favicon.ico`和通常保存`robots.txt`在文件`public`夾中的文件，並從應用程序的根 URL 提供。例如，從提供`favicon.ico`檔案`https://example.com/favicon.ico`。請注意，這些檔案沒有可預測的路徑模式。它們幾乎完全由文件名決定。目標文件`public`夾中的文件的唯一方法是使用 Catch-All 路由。但是，捕獲所有路由目標必須是計算原始。

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要求路徑。例如，您可以使用指 `/*.*`定包含副檔名的所有要求路徑。

   請注意，這種方法可能是不可靠的。例如，如果public資料夾內有沒有副檔名的檔案，則這些檔案不會受到此規則的目標。使用這種方法需要注意的另一個問題是，應用程序可能具有名稱中帶有句點的頁面。例如，在的頁面/blog/2021/01/01/hello.world將成為/*.* 規則的目標。這並不理想，因為頁面不是靜態資產。但是，您可以向此規則添加一個後備目標，以確保當靜態基本原始存在 404 錯誤時，請求會退回到計算原始文件。

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

2. 在建置時識別資public料夾中的檔案，並為每個檔案發出路由規則。由於部署規格有 25 個規則的限制，因此無法擴充此方法。

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

3. 建議您的框架用戶將所有可變靜態資產存儲在文件夾內的子文件public夾中。

   在下面的例子中，用戶可以存儲public/assets文件夾中的所有可變靜態資產。然後，/assets/*可以使用路徑模式的路由規則來定位public/assets文件夾內的所有可變靜態資產。

   ```
   {
   ```

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

4. 指定捕捉全部路由的靜態後援。這種方法具有在下捕捉全部後備路由一節中更詳細描述的缺點。

捕捉全部後備路由

對於 SSR 框架，例如Nuxt.js，為計算原始目標指定了 Catch-All 路由，框架作者可能會考慮為 Catch-All 路由指定靜態後備，以解決文件夾路由問題。public不過，這種類型的路由規則會中斷伺服器端轉譯的 404 頁面。例如，如果一般使用者造訪不存在的頁面，應用程式會呈現 404 狀態碼為 404 的網頁。但是，如果捕獲所有路由具有靜態後備，則不會呈現 404 頁面。相反，請求回落到靜態原語，並且仍然以 404 狀態碼結束，但不會呈現 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之功能的架構預期會在目錄內的靜態資產前面加上基本路徑。.amplify-hosting/static例如，如果基本路徑是/folder1/folder2，則名為 main.css 的靜態資產的構建輸出將是.amplify-hosting/static/folder1/folder2/main.css。

這意味著還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，則資public料夾中靜態資產的路由規則將如下所示。

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

```
}
```

同樣地，伺服器端路由也需要在前面加上基底路徑。例如，如果基本路徑是/folder1/folder2，則路由的/api路由規則將如下所示。

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

但是，基準路徑不應附加在集合所有佈線的前面。例如，如果基準路徑為/folder1/folder2，則集成所有佈線將保持如下所示。

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

Nuxt.js 路由的例子

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": "nodejs18.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": "nodejs18.x"
    }
  ],
  "framework": {
    "name": "nuxt",
    "version": "3.8.1"
  }
}
```

如需有關使用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快速伺服器

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

在部署到 Amplify 主機之前，先在本地設置快速服務器

1. 為您的項目創建一個新的目錄，並安裝快遞和打字稿。

```
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. 使用以下內容將tsconfig.json文件添加到項目的根目錄中。

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

3. 創建一個在項目根目錄src中命名的目錄。

4. 在src目錄中創建一個index.ts文件。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. 將以下腳本添加到您的package.json文件中。

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

6. 創建一個public在項目根目錄中命名的目錄。然後創建一個具hello-world.txt有以下內容命名的文件。

```
Hello world!
```

7. 將具有以下內容的.gitignore文件添加到項目根目錄中。

```
.amplify-hosting
```

```
dist
node_modules
```

設定 Amplify 部署資訊清單

1. 創建一個deploy-manifest.json在項目根目錄中命名的文件。
2. 將以下清單複製並粘貼到您的deploy-manifest.json文件中。

```
{
  "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": "nodejs18.x",
      "entrypoint": "index.js"
    }
  ]
}
```

資訊清單描述了 Amplify 主機應如何處理應用程式的部署。主要設定如下。

- version — 指出您正在使用的部署規格版本。
- 框架 — 調整此值以指定您的Express服務器設置。
- 影像設定 — 除非您正在處理影像最佳化，否則此區段對於Express伺服器而言是選用的。
- 路線 — 這些對於將流量導向到應用程序的正確部分至關重要。`"kind": "Compute"`路由會將流量導向至您的伺服器邏輯。
- 計算資源 — 使用此區段可指定Express伺服器的執行階段和進入點。

接下來，設定建置後指令碼，將建置的應用程式成品移至部`.amplify-hosting`署服務包。目錄結構與「Amplify 主機」部署規格一致。

設定建置後指令碼

1. 創建一個在項目根目錄`bin`中命名的目錄。
2. 建立目錄`postbuild.sh`中名為的`bin`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腳本添加到文package.json件中。檔案應如下所示。

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

4. 運行以下命令來構建您的應用程序。

```
npm run build
```

5. (選擇性) 調整「快速」的路線。您可以修改部署資訊清單中的路由，以適合您的 Express 伺服器。例如，如果您在public目錄中沒有任何靜態資產，則可能只需要"path": "/*"引導至 Compute 的全部捕捉路由。這將取決於您的服務器的設置。

您的最終目錄結構應如下所示。

```
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. 將您的代碼推送到 Git 存儲庫，然後將您的應用程序部署到 Amplify 託管。
2. 將構建設置更新為指`baseDirectory`向`.amplify-hosting`如下。在建置期間，Amplify 會偵測`.amplify-hosting`目錄中的資訊清單檔案，並依照設定部署 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. 要驗證您的部署是否成功，並且您的服務器是否正常運行，請訪問您的應用程序由 Amplify Hosting 提供的默認 URL。

SSR 應用程式的影像優化

Amplify 託管提供支持所有 SSR 應用程序的內置圖像優化功能。借助 Amplify 的圖像優化，您可以為正在訪問它們的設備以正確的格式，尺寸和分辨率提供高質量的圖像，同時保持最小的文件大小。

目前，您可以使用 Next.js Image 元件隨選最佳化影像，也可以實作自訂映像載入程式。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即可使用放大的影像最佳化功能。如果您要實作自訂載入程式，請參閱使用自定義圖像加載器。

使用自定義圖像加載器

如果您使用自定義圖像加載器，Amplify 會檢測應用程序`next.config.js`文件中的加載器，並且不使用內置的圖像優化功能。如需 Next.js 支援之自訂載入程式的詳細資訊，請參閱 Next.js 映像檔文件。

框架作者的圖像優化集成

架構作者可以使用 Amplify 主機部署規格整合 Amplify 的影像最佳化功能。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影像最佳化設定的詳細資訊，請參閱Amplify 主機部署規範。

瞭解影像最佳化 API

您可以在執行階段透過 Amplify 應用程式的網域 URL，在路由規則所定義的路徑上叫用影像最佳化。

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

影像最佳化會在影像上施加下列規則。

- Amplify 無法優化 GIF，APNG 和 SVG 格式或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得超過 11 MB 或 9,000 像素。
- 最佳化影像的大小限制為 4 MB。
- HTTP 或 HTTPS 是唯一支援使用遠端 URL 擷取影像的通訊協定。

HTTP 標頭

接受請求 HTTP 標頭用於指定圖像格式，表示為 MIME 類型，由客戶端（通常是 Web 瀏覽器）允許。影像最佳化服務會嘗試將影像轉換為指定的格式。為此標頭指定的值將具有比 format 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 Accept 標頭要求特定格式，如果該格式不在允許列表中，它也將被忽略。

URI 請求參數

下表說明影像最佳化的 URI 要求參數。

查詢參數	Type	必要	描述	範例
url	字串	是	來源影像的相對路徑或絕對 URL。對於遠端網址，支援 http 和 https 通訊協定。值必須經過 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	最佳化影像的高度 (以像素為單位)。如果沒有指定，圖像將 auto 縮放以匹配寬度。	`?height=600`

查詢參數	Type	必要	描述	範例
適合	枚舉 值：cover，cont	否	如何調整圖像大小以適應指定的寬度和高度。	?width=800&height=600&fit=cover
position	枚舉 值：center，top	否	擬合為或時要使用的位cover置contai	?fit=contain&position=centre
trim	Number	否	修剪包含與左上角像素指定背景顏色類似值的所有邊緣的像素。	?trim=50
擴充	物件	否	使用衍生自最近邊緣像素的顏色，將像素新增至影像邊緣。格式是每{top}_{right}_{bottom}_{left} 個值都是要加入的像素數目的位置。	?extend=10_0_5_0
提取物	物件	否	將圖像裁剪為由頂部，左側，寬度和高度分隔的指定矩形。格式為 {左} _ {頂} _ {寬度} _ {右}，其中每個值是要裁剪的像素數。	?extract=10_0_5_0

查詢參數	Type	必要	描述	範例
格式	字串	否	最佳化影像所需的輸出格式。	`?format=webp`
quality	Number	否	圖像的質量，從 1 到 100。僅在轉換圖像格式時使用。	`?quality=50`
rotate	Number	否	以指定的角度 (以度數為單位) 旋轉影像。	`?rotate=45`
翻轉	Boolean	否	在 X 軸上垂直 (上下) 鏡射影像。這總是在旋轉之前發生（如果有的話）。	`?flip`
搖晃	Boolean	否	在 y 軸上水平 (左右) 鏡射影像。這總是在旋轉之前發生（如果有的話）。	`?flop`
削尖	Number	否	銳利化可增強影像中邊緣的清晰度。有效值介於 0.000001 和 10 之間。	`?sharpen=1`
median	Number	否	套用中值篩選。這樣可以消除雜訊或平滑影像的邊緣。	`?sharpen=3`

查詢參數	Type	必要	描述	範例
模糊	Number	否	套用指定 sigma 的高斯模糊效果。有效值介於 0.3 到 1,000 之間。	`?blur=20`
Gamma	Number	否	套用 Gamma 校正，以改善調整大小後影像的可感知亮度。值必須介於 1.0 和 3.0 之間。	`?gamma=1`
無效	Boolean	否	反轉影像的顏色。	`?negate`
正常化	Boolean	否	通過拉伸其亮度以覆蓋完整的動態範圍來增強圖像對比度。	`?normalize`
threshold	Number	否	如果影像中的任何像素亮度小於指定臨界值，則以黑色像素取代影像中的任何像素。或者，如果它大於閾值，則使用白色像素。有效值介於 0 到 255 之間。	`?threshold=155`
色調	字串	否	使用提供的 RGB 對影像進行色調，同時保留影像的亮度。	`?tint=#7743CE`

查詢參數	Type	必要	描述	範例
灰階	Boolean	否	將影像變成灰階 (黑白)。	`?grayscale`

響應狀態碼

下列清單說明影像最佳化的回應狀態碼。

成功-狀態碼 200

請求已成功填滿。

BadRequest -狀態碼 400

- 輸入查詢參數指定不正確。
- 遠端 URL 未列為`remotePatterns`設定中允許的狀態。
- 遠端 URL 無法解析為影像。
- 請求的寬度或高度未在`sizes`設置中允許列出。
- 請求的圖像是 SVG，但`dangerouslyAllowSvg`設置被禁用。

找不到-HTTP 狀態碼

找不到來源影像。

內容太大-HTTP 狀態碼 413

來源影像或最佳化影像超過允許的最大大小 (以位元組為單位)。

快取

Amplify Hosting 會在我們的 CDN 上緩存優化的圖像，以便從緩存中提供對同一圖像的後續請求，具有相同查詢參數。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。

- 使用以影像最佳化為目標的路由規則中的`Cache-Control`金鑰。
- 使用「Amplify」應用程式中定義的自訂標題。
- 對於遠端映像，會遵循遠端映像傳回的`Cache-Control`標頭。

在影像最佳化設定中minimumCacheTTL指定的定義了cache-control max-age指令的下限。例如，如果遠端影像 URL 以 a 回應cache-control s-max-age=10，但值minimumCacheTTL為 60，則會使用 60。

對 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，它會使用符合用來建置應用程式的主要版本Node.js的Node.js執行階段版本。

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。如需有關設定即時套件更新的詳細資訊，請參閱即時套件更新。您也可以使用其他機制 (例如指nvm令) 指定Node.js版本。如果您未指定版本，Amplify 預設會使用 Amplify 組建容器所使用的目前版本。

SSR 部署的疑難

如果您在使用 Amplify 主機運算部署 SSR 應用程式時遇到未預期的問題，請檢閱下列疑難排解主題。如果您在這裡找不到問題的解決方案，請參閱擴大主機問 GitHub 題存 Amplify 庫中的 SSR Web 計算疑難排解指南。

主題

- 您正在使用框架適配器
- 邊緣 API 路由會導致您的 Next.js 組建失敗
- 依需求增量靜態再生不適用於您的應用程式
- 您的應用程序的構建輸出超過允許的最大大小
- 您的構建失敗，並出現內存不足錯誤
- HTTP 回應大小太大

您正在使用框架適配器

如果您在部署使用架構介面卡的 SSR 應用程式時遇到問題，請參閱Amplify 對 SSR 框架的支援。

邊緣 API 路由會導致您的 Next.js 組建失敗

目前，Amplify 不支援 Next.js 邊緣 API 路由。使用 Amplify 託管應用程式時，您必須使用非邊緣 API 和中介軟體。

依需求增量靜態再生不適用於您的應用程式

從 12.2.0 版開始，Next.js 支援漸進靜態再生 (ISR)，以手動清除特定頁面的 Next.js 快取。不過，Amplify 目前不支援隨選 ISR。如果您的應用程式使用 Next.js 隨選重新驗證，則當您將應用程式部署到 Amplify 時，此功能將無法運作。

您的應用程序的構建輸出超過允許的最大大小

目前，「擴大」對 SSR 應用程式所支援的最大組建輸出大小為 220 MB。如果您收到錯誤消息，指出應用程序的構建輸出大小超過了允許的最大大小，則必須採取措施來減少它。

若要減少應用程式組建輸出的大小，您可以檢查應用程式的組建成品，並識別要更新或移除的任何大型相依性。首先，將構建工件下載到本地計算機。然後，檢查目錄的大小。例如，node_modules目錄可能包含 Next.js 伺服器執行階段檔案所參考@esbuild的二進位檔案，例如@swc和。由於這些二進製文件在運行時不是必需的，因此您可以在構建後刪除它們。

使用下列指示下載應用程式的組建輸出，並使用 AWS Command Line Interface (CLI) 檢查目錄的大小。

若要下載並檢查 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。將應用程序 ID，分支名稱和作業 ID 更改為您自己的信息。針對工作 ID，請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，於、stepName: "BUILD"區段中找出預先簽署的job成品 URL。stepsURL會在下列範例輸出中以紅色反白顯示。

   ```
   "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. 將 URL 複製並貼到瀏覽器視窗中。artifacts.zip檔案會下載到您的本機電腦。這是您的組建輸出。
4. 執行du磁碟使用指令以檢查目錄的大小。下列範例命令會傳回compute和static目錄的大小。

```
du -csh compute static
```

以下是包含compute和static目錄大小資訊的輸出範例。

```
 29M    compute
3.8M    static
 33M    total
```

5. 開啟目compute錄，然後找到資node_modules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. 如果您的應用程序包含運行時不需要的二進製文件，請在構建後通過將以下命令添加到應用程序amplify.yml文件的構建部分來刪除它們。

```
- 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檔案的建置指令區段範例，其中包含在執行生產組建之後新增這些命令。

```
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
```

您的構建失敗，並出現內存不足錯誤

Next.js 可讓您快取組建加工品，以改善後續組建的效能。此外，Amplify 的 AWS CodeBuild 容器會代表您壓縮此快取並上傳到 Amazon S3，以改善後續的建置效能。這可能會導致您的構建失敗，並出現內存不足錯誤。

執行下列動作，以防止您的應用程式在建置階段超出記憶體限制。首先，`.next/cache/**/*`從構建設置的緩存 .path 部分中刪除。接下來，從構建設置文件中刪除`NODE_OPTIONS`環境變量。請改為在 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。如需有關使用 Amplify 主控台設定環境變數的詳細資訊，請參閱設定環境變數。

進行這些更改後，請再次嘗試構建。如果成功，請將其添加`.next/cache/**/*`回構建設置文件的 cache.path 部分。

如需 Next.js 快取組態以提升建置效能的詳細資訊，請參閱 Next.js 網站 CodeBuild上的 AWS。

HTTP 回應大小太大

目前，使用網路運算平台的 Next.js 12 和 13 個應用程式所支援的最 Amplify 回應大小為 5.72 MB。超過該限制的響應返回 504 錯誤，沒有內容給客戶端。

Amplify 對 Next.js SSR 的支援

Amplify 支援僅使用 Next.js 建立的伺服器端轉譯 (SSR) 網路應用程式的部署和託管。Next.js 是一個反應框架用於開發水療中心與 JavaScript. 您可以部署使用 Next.js 13 建置的應用程式，其中包含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 頁面會在建置時預先轉譯，SSR 頁面會在要求時預先轉譯。

預渲染可以提高性能和搜索引擎優化。由於 Next.js 預先呈現伺服器上的所有頁面，因此每個頁面的 HTML 內容在到達用戶端的瀏覽器時都已準備就緒。此內容也可以加快載入速度。更快的加載時間可以改善最終用戶對網站的體驗，並對網站的 SEO 排名產生積極影響。預渲染還可以通過使搜索引擎機器人輕鬆查找和抓取網站的 HTML 內容來改善 SEO。

Next.js 提供用於測量各種效能指標的內建分析支援，例如第一個位元組的時間 (TTFB) 和第一個內容繪圖 (FCP)。如需 Next.js 的詳細資訊，請參閱 Next.js 網站開始使用。

Next.js 功能支援

Amplify 託管計算可完全管理使用 Next.js 12 和 13 構建的應用程序的服務器端渲染（SSR）。如果您在 Amplify 託管計算發布之前部署了 Next.js 應用程序以 Amplify，則您的應用程序正在使用放大的先前的 SSR 提供程序經典（僅限 Next.js 11）。Amplify 主機運算不支援使用 Next.js 版本 11 或更早版本建立的應用程式。我們強烈建議您將 Next.js 11 應用程式移轉至 Amplify 主機運算託管 SSR 提供者。

下列清單說明 Amplify 主機運算 SSR 提供者支援的特定功能。

支援的功能

- 伺服器端呈現頁面 (SSR)
- 靜態頁面
- API 路由
- 動態路線
- 捕捉所有路線
- 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
- 中介軟體
- 環境變數
- 影像最佳化
- Next.js 應用程式目錄

不支援的功能

- 邊緣 API 路由 (不支援邊緣中介軟體)
- 依需求增量靜態再生 (ISR)
- 國際化 (i18n) 自動語言環境偵測
- Next.js 流媒體
- 在靜態資產和最佳化影像上執行中介軟體

Next.js 圖片

圖像的最大輸出大小不能超過 4.3 MB。你可以有一個更大的圖像文件存儲在某個地方，並使用 Next.js 圖像組件調整大小和優化成一個 Webp 或 AVIF 格式，然後將其作為一個較小的尺寸。

請注意，Next.js 文件建議您安裝 Sharp 影像處理模組，以使影像最佳化能夠在生產環境中正常運作。但是，這對於擴大部署來說並不是必要的。Amplify 功能會自動為您部署夏普。

Next.js SSR 應用程式的定價

部署 Next.js 12 或更新版本的 SSR 應用程式時，Amplify 主機運算會管理為您部署 SSR 應用程式所需的資源。如需 Amplify 主機運算費用的相關資訊，請參閱AWS Amplify 定價。

使用 Amplify 功能部署 Next.js SSR 應用程式

默認情況下，Amplify 使用 Amplify 大託管的計算服務與 Next.js 12 和 13 的支持部署新的 SSR 應用程序。Amplify 託管計算可以完全管理部署 SSR 應用程序所需的資源。您在 2022 年 11 月 17 日之前部署的「Amplify」帳戶中的 SSR 應用程式正在使用傳統版 (僅限 Next.js 11) SSR 提供者。

我們強烈建議您將使用傳統版 (僅限 Next.js 11) SSR 的應用程式移轉至「Amplify 主機」運算 SSR 提供者。Amplify 不會為您執行自動遷移。您必須手動遷移應用程序，然後啟動新的組建才能完成更新。如需說明，請參閱遷移 Next.js 11 SSR 應用程式以 Amplify 主機運算。

使用下列指示來部署新的 SSR 應用程式。

使用 Amplify 主機運算 SSR 提供者部署 SSR 應用程式以 Amplify

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 在 [所有應用程式] 頁面上，選擇 [新增應用程式]，然後選擇 [主機
3. 選取您的 GitHub、Bitbucket 或 AWS CodeCommit 儲存庫提供者 GitLab，然後選擇 [繼續]。
4. 在「新增儲存區域分支」頁面上，執行下列動作：
   a. 在「最近更新的儲存庫」清單中，選取要連線的存放庫名稱。
   b. 在「分支」清單中，選取要連線的存放庫分支名稱。
   c. 選擇下一步。
5. 此應用程式需要 Amplify 代表您呼叫其他服務時所假設的 IAM 服務角色。您可以允許 Amplify 託管計算自動為您創建服務角色，也可以指定已創建的角色。

- 允許 Amplify 自動建立角色並將其附加至您的應用程式
  - 在「IAM 角色」區段中，選擇「建立並使用新的服務角色」。
- 若要附加先前建立的服務角色
  a. 在「IAM 角色」區段中，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「複查」頁面上，選擇「儲存並部署」。

封裝 .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，以偵測應用程式是 SSR 還是 SSG。

以下是 Next.js SSR 應用程式的建置指令碼範例。構建腳本`"next build"`指出該應用程序支持 SSG 和 SSR 頁面。

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

以下是 Next.js SSG 應用程式的建置指令碼範例。構建腳本`"next build && next export"`指出該應用程序僅支持 SSG 頁面。

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

Amplify 構建設置

檢查應用程式的`package.json`檔案以判斷是否要部署 SSG 或 SSR 應用程式之後，Amplify 會檢查應用程式的建置設定。您可以將組建設定儲存在 Amplify 主控台或儲存庫根目錄的`amplify.yml`檔案中。如需詳細資訊，請參閱 配置構建設置。

如果 Amplify 偵測到您正在部署 Next.js SSR 應用程式，且沒有任何amplify.yml檔案存在，它會產生應用程式的建置規格，並將設定為。baseDirectory .next如果您要部署存在檔案的應用程式，則amplify.yml檔案中的組建設定會覆寫主控台中的任何組建設定。因此，您必須手動將檔baseDirectory案.next中的設定為。

以下是設置為的應用程序的構建設置的示例.next。baseDirectory這表示建置加工品適用於支援 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/**/*
```

如果 Amplify 偵測到您正在部署 SSG 應用程式，它會為應用程式產生建置規格，並將其設定為。baseDirectory out如果您要部署存在amplify.yml檔案的應用程式，則必須在檔案out中手動baseDirectory將設定為。

以下是設置為的應用程序的構建設置的示例out。baseDirectory這表示組建加工品適用於僅支援 SSG 頁面的 Next.js 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/**/*
```

遷移 Next.js 11 SSR 應用程式以 Amplify 主機運算

當您部署新的 Next.js 應用程式時，預設情況下，Amplify 會使用最新的支援版本 Next.js。目前，Amplify 主機運算 SSR 提供者支援 Next.js 版本 13。

Amplify 主控台會偵測您帳戶中在 Amplify 主機運算服務發行之前部署的應用程式，並完整支援 Next.js 12 和 13。主控台會顯示資訊橫幅，識別使用 Amplify 先前的 SSR 提供者經典 (僅限 Next.js 11) 部署的分支的應用程式。我們強烈建議您將應用程式移轉至 Amplify 主機運算 SSR 提供者。

您必須同時手動遷移應用程式及其所有生產分支。一個應用程序不能同時包含傳統版（僅限 Next.js 11）和 Next.js 12 或 13 個分支。

使用下列指示將應用程式移轉至 Amplify 主機運算 SSR 提供者。

將應用程式移轉至 Amplify 主機運算 SSR 提供者

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 選擇您要遷移的 Next.js 應用程序。

   > **Note**
   >
   > 在 Amplify 控制台中遷移應用程式之前，您必須先更新應用程式的套件 .json 檔案，以使用 Next.js 版本 12 或 13。

3. 在導覽窗格中，選擇 [應用程式設定]、[一般]。
4. 如果應用程式已使用傳統 (僅限 Next.js 11) SSR 提供者部署分支，則主控台會在應用程式首頁上顯示橫幅。在橫幅上，選擇 [移轉]。
5. 在移轉確認視窗中，選取三個陳述式，然後選擇移轉。
6. Amplify 會建置並重新部署您的應用程式，以完成移轉作業。

還原 SSR 移轉

當您部署 Next.js 應用程式時，Amplify 主機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 應用程式已設定為平台值WEB。使用 Next.js 版本 11 的 SSR 應用程式會設定為平台值WEB_DYNAMIC。Next.js 12 或 13 SSR 應用程式已設定為平台值WEB_COMPUTE。

當您使用上一節中的指示移轉應用程式時，Amplify 會將應用程式的平台值從變更WEB_DYNAMIC為WEB_COMPUTE。遷移到 Amplify 主機計算完成後，您無法在控制台中還原遷移。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。打開終端機窗口並輸入以下命令，使用您的唯一信息更新應用程序 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。在您開始將 SSG 應用程式轉換為 SSR 之前，請先更新應用程式以使用 Next.js 版本 12 或 13，並新增 SSR 功能。然後，您將需要執行以下步驟。

1. 使用變 AWS Command Line Interface 更應用程式的平台類型。
2. 將服務角色新增至應用程式。
3. 在應用程序的構建設置中更新輸出目錄。
4. 更新應用程式的package.json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 版本 11 的 SSR 應用程式已設定為平台類型WEB_DYNAMIC。對於使用由 Amplify 主機運算管理的 SSR 部署至 Next.js 12 或 13 的應用程式，平台類型會設定為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 將您的應用程式的平台變更為WEB_COMPUTE。打開終端機窗口並輸入以下命令，使用您唯一的應用程序 ID 和 Region 更新紅色文本。

```
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所承擔的 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。

若要新增服務角色

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
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必要步驟。
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. 對於服務角色，請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。

更新構建設置

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組建設定，以便將輸出目錄設定為。`.next`您可以在 Amplify 控制台或存儲在存儲庫中的`amplify.yml`文件中編輯構建設置。若要取得更多資訊，請參閱配置構建設置。

以下是設置為的應用程序的構建設置的示例`.next`。`baseDirectory`

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

```
      - node_modules/**/*
```

更新封裝的 .json 檔案

新增服務角色並更新組建設定後，請更新應用程式的`package.json`檔案。如下列範例所示，將建置指令碼設定`"next build"`為指出 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 會偵測存放庫中`package.json`檔案的變更，並重新部署具有 SSR 功能的應用程式。

讓環境變數可供伺服器端執行階段存取

Amplify 託管支持通過在 Amplify 控制台的項目配置中進行設置來將環境變量添加到應用程序的構建中。不過，Next.js 伺服器元件預設無法存取這些環境變數。此行為是刻意保護應用程式在建置階段期間所使用之環境變數中儲存的任何密碼。

若要讓 Next.js 可存取特定的環境變數，您可以修改 Amplify 建置規格檔案，以便在 Next.js 可辨識的環境檔案中設定這些變數。這可讓 Amplify 在建置應用程式之前載入這些環境變數。以下構建規範示例演示瞭如何在構建命令部分中添加環境變量。

```
version: 1
frontend:
  phases:
    preBuild:
      commands:
        - npm ci
    build:
      commands:
        - env | grep -e DB_HOST -e DB_USER -e DB_PASS >> .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，build 命令區段包含兩個命令，這些命令會在應用程式建置執行之前將環境變數寫入.env.production檔案。Amplify 託管允許您的應用程序在應用程序接收流量時訪問這些變量。

前面示例中構建命令部分的以下行演示瞭如何從構建環境中獲取特定變量並將其添加到.env.production文件中。

```
- env | grep -e DB_HOST -e DB_USER -e DB_PASS >> .env.production
```

如果變數存在於您的建置環境中，.env.production檔案將包含下列環境變數。

```
DB_HOST=localhost
DB_USER=myuser
DB_PASS=mypassword
```

上述範例中建置指令區段的下列行示範如何將具有特定前置詞的環境變數加入至.env.production檔案。在此範例中，NEXT_PUBLIC_會加入具有前置字元的所有變數。

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

如果構建環境中存在多個帶有NEXT_PUBLIC_前綴的變量，則該.env.production文件將類似於以下內容。

```
NEXT_PUBLIC_ANALYTICS_ID=abcdefghijk
NEXT_PUBLIC_GRAPHQL_ENDPOINT=uowelalsmlsadf
NEXT_PUBLIC_SEARCH_KEY=asdfiojslf
NEXT_PUBLIC_SEARCH_ENDPOINT=https://search-url
```

壟斷的 SSR 環境變量

如果您要在 monorepo 中部署 SSR 應用程式，並且想要讓 Next.js 可以存取特定的環境變數，您必須在.env.production檔案前面加上您的應用程式根目錄。以下示例為 NX monorepo 中的 Next.js 應用程序構建規範演示瞭如何在構建命令部分中添加環境變量。

```
version: 1
```

```
applications:
  - frontend:
      phases:
        preBuild:
          commands:
            - npm ci
        build:
          commands:
            - env | grep -e DB_HOST -e DB_USER -e DB_PASS >> 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
```

前面示例中構建命令部分的以下幾行演示瞭如何從構建環境中獲取特定變量，並將它們添加到具有應用程序根目錄apps/app的 monorepo 中的應用程序的.env.production文件中。

```
- env | grep -e DB_HOST -e DB_USER -e DB_PASS >> apps/app/.env.production
- env | grep -e NEXT_PUBLIC_ >> apps/app/.env.production
```

在單一軟件庫中部署 Next.js 應用程序

Amplify 支持通用壟斷中的應用程序以及使用 npm 工作區，pnpm 工作區，Yarn 工作區，NX 和植物園創建的壟斷中的應用程序。當您部署應用程式時，Amplify 會自動偵測您正在使用的 MONOrepo 建置架構。Amplify 會自動套用建置設定，適用於 npm 工作區、Yarn 工作區或 NX 中的應用程式。請注意，pnpm 和植物園應用程序需要額外的配置。如需詳細資訊，請參閱 壟斷構建設置。

如需詳細的 NX 範例，請參閱在 AWS 上使用 NX 在 Next.js 應用程式之間共用程式碼部落格文章。

SSR 應用程式的 Amazon CloudWatch 日

Amplify 會將 Next.js 執行階段的相關資訊傳送至您 AWS 帳戶的. CloudWatch 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所假設的 IAM 服務角色。您可以允許 Amplify 託管計算自動為您創建服務角色，也可以指定已創建的角色。

如果您選擇允許 Amplify 為您建立 IAM 角色，則該角色已具有建立 CloudWatch 記錄的權限。如果您建立自己的 IAM 角色，則需要在政策中新增下列許可，以允許 Amplify 存取 Amazon CloudWatch 日誌。

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

如需服務角色的詳細資訊，請參閱新增服務角色。

Amplify Next.js 11 社会主义支持

如果您在 2022 年 11 月 17 日發行「Amplify 主機」計算之前部署了 Next.js 應用程式來放大，則您的應用程式正在使用 Amplify 先前的 SSR 提供者經典版 (僅限 Next.js 11)。本節中的文件僅適用於使用傳統 (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 應用程式移轉至 Amplify 主機運算託管 SSR 提供者。如需詳細資訊，請參閱 遷移 Next.js 11 SSR 應用程式以 Amplify 主機運算。

下列清單說明擴大傳統 (僅限 Next.js 11) SSR 提供者支援的特定功能。

支援的功能

- 伺服器端呈現頁面 (SSR)
- 靜態頁面
- API 路由
- 動態路線
- 捕捉所有路線
- 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

- 影像最佳化
- 依需求增量靜態再生 (ISR)
- 國際化 (i18n) 網域路由
- 國際化 (i18n) 自動語言環境偵測
- 中介軟體
- 邊緣中間件
- 邊緣 API 路由

應用程式定價 Next.js

部署 Next.js 11 SSR 應用程式時，Amplify 會在您的 AWS 帳戶中建立額外的後端資源，包括：

- 一種 Amazon Simple Storage Service (Amazon S3) 儲存貯體，用於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。
- 為應用程序提供服務的 Amazon CloudFront 分發。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 應用程式的權限 Next.js

Amplify 需要 AWS Identity and Access Management (IAM) 許可才能部署 SSR 應用程式。如果沒有必要的最低權限，您會在嘗試部署 SSR 應用程式時收到錯誤訊息。若要提供 Amplify 所需的權限，您必須指定服務角色。

若要建立 Amplify 代表您呼叫其他服務時假設的 IAM 服務角色，請參閱新增服務角色。這些指示示範如何建立附加`AdministratorAccess-Amplify`受管理原則的角色。

受`AdministratorAccess-Amplify`管政策可讓您存取多項 AWS 服務，包括 IAM 動作。而且應視為`AdministratorAccess`政策一樣強大。此原則提供的權限超過部署 SSR 應用程式所需的權限。

建議您遵循授與最低權限並降低授與服務角色的權限的最佳作法。您可以建立自己的客戶受管理 IAM 政策，以僅授予部署 SSR 應用程式所需的權限，而不是授予服務角色的管理員存取權限。如需建立客戶受管政策的指示，請參閱 IAM 使用者指南中的建立 IAM 政策。

如果您建立自己的原則，請參閱下列清單，列出部署 SSR 應用程式所需的最低權限。

```
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
cloudfront:DeleteDistribution
iam:AttachRolePolicy
iam:CreateRole
iam:CreateServiceLinkedRole
iam:GetRole
iam:PutRolePolicy
iam:PassRole
iam:UpdateAssumeRolePolicy
iam:DeleteRolePolicy
lambda:CreateFunction
lambda:EnableReplication
lambda:DeleteFunction
lambda:GetFunction
lambda:GetFunctionConfiguration
lambda:PublishVersion
lambda:UpdateFunctionCode
lambda:UpdateFunctionConfiguration
lambda:ListTags
lambda:TagResource
lambda:UntagResource
lambda:ListEventSourceMappings
```

```
lambda:CreateEventSourceMapping
route53:ChangeResourceRecordSets
route53:ListHostedZonesByName
route53:ListResourceRecordSets
s3:CreateBucket
s3:GetAccelerateConfiguration
s3:GetObject
s3:ListBucket
s3:PutAccelerateConfiguration
s3:PutBucketPolicy
s3:PutObject
s3:PutBucketTagging
s3:GetBucketTagging
sqs:CreateQueue
sqs:DeleteQueue
sqs:GetQueueAttributes
sqs:SetQueueAttributes
amplify:GetApp
amplify:GetBranch
amplify:UpdateApp
amplify:UpdateBranch
```

疑難排解 Next.js 11 SSR 部署

如果您在使用擴大功能部署傳統 (僅限 Next.js 11) SSR 應用程式時遇到未預期的問題，請檢閱下列疑難排解主題。

主題

- 您的輸出目錄被覆蓋
- 您在部署 SSR 網站後收到 404 錯誤
- 您的應用程序缺少 CloudFront SSR 發行版的重寫規則
- 您的應用程式太大而無法部署
- 您的構建失敗，並出現內存不足錯誤
- 您的應用程序同時具有 SSG 分支和 SSG 分支
- 您的應用程序將靜態文件存儲在具有保留路徑的文件夾中
- 您的應用程式已達到 CloudFront上限
- 環境變數不會傳遞至 Lambda 函數
- 在美國東部 (維吉尼亞北部) 區域建立 Lambda @Edge 函數

- Next.js 應用程式使用不支援的功能
- Next.js 應用程式中的影像無法載入
- 不支援地區

您的輸出目錄被覆蓋

使用 Amplify 功能部署之 Next.js 應用程式的輸出目錄必須設定為.next。如果您的應用程序的輸出目錄被覆蓋，請檢查該next.config.js文件。要將構建輸出目錄默認為.next，請從文件中刪除以下行：

```
distDir: 'build'
```

驗證輸出目錄是否已在構建設置.next中設置為。如需檢視應用程式組建設定的詳細資訊，請參閱配置構建設置。

以下是設置為的應用程序的構建設置的示例.next。baseDirectory

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

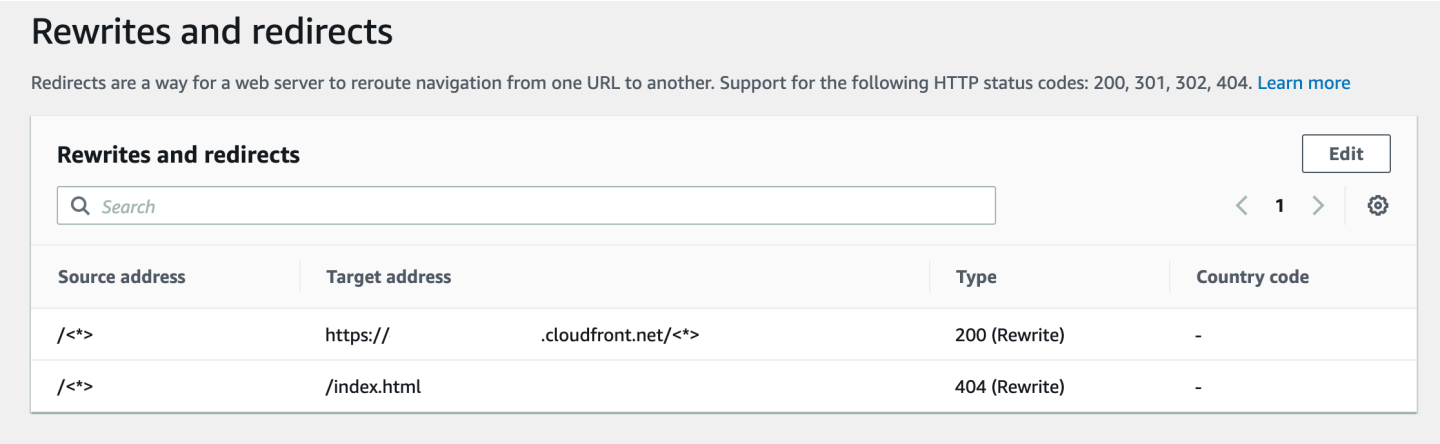
您在部署 SSR 網站後收到 404 錯誤

如果您在部署網站後收到 404 錯誤，可能是因為您的輸出目錄遭到覆寫所造成。若要檢查next.config.js檔案並驗證應用程式組建規格中的正確組建輸出目錄，請依照上一個主題中的步驟執行您的輸出目錄被覆蓋。

您的應用程序缺少 CloudFront SSR 發行版的重寫規則

當您部署 SSR 應用程式時，Amplify 會為您的 CloudFront SSR 發行版建立重寫規則。如果您無法在網頁瀏覽器中存取應用程式，請確認 Amplify 主控台中是否存在應用程式的 CloudFront重寫規則。如果遺失，您可以手動新增或重新部署應用程式。

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新寫入規則範例。請注意，在此範例中，存在 CloudFront 重新寫入規則。



您的應用程式太大而無法部署

「Amplify」會將 SSR 部署的大小限制為 50 MB。如果您嘗試部署 Next.js SSR 應用程式來 Amplify 大並出RequestEntityTooLargeException現錯誤，表示您的應用程式太大而無法部署。您可以嘗試將快取清理程式碼新增至next.config.js檔案來解決此問題。

以下是執行快取清理的next.config.js檔案中的程式碼範例。

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

您的構建失敗，並出現內存不足錯誤

Next.js 可讓您快取組建加工品，以改善後續組建的效能。此外，Amplify 的 AWS CodeBuild 容器會代表您壓縮此快取並上傳到 Amazon S3，以改善後續的建置效能。這可能會導致您的構建失敗，並出現內存不足錯誤。

執行下列動作，以防止您的應用程式在建置階段超出記憶體限制。首先，`.next/cache/**/*`從構建設置的緩存 .path 部分中刪除。接下來，從構建設置文件中刪除`NODE_OPTIONS`環境變量。請改為在 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。如需有關使用 Amplify 主控台設定環境變數的詳細資訊，請參閱設定環境變數。

進行這些更改後，請再次嘗試構建。如果成功，請將其添加`.next/cache/**/*`回構建設置文件的 cache.path 部分。

如需 Next.js 快取組態以提升建置效能的詳細資訊，請參閱 Next.js 網站 CodeBuild上的 AWS。

您的應用程序同時具有 SSG 分支和 SSG 分支

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

您的應用程序將靜態文件存儲在具有保留路徑的文件夾中

Next.js 可以從存儲在項目根目錄中`public`的名為的文件夾提供靜態文件。當您使用 Amplify 部署和主控 Next.js 應用程式時，您的專案無法包含路徑`public/static`的資料夾。Amplify 會保留散佈應用程式時使用的`public/static`路徑。如果您的應用程式包含此路徑，則必須在使用 Amplify 部署之前重新命名`static`資料夾。

您的應用程式已達到 CloudFront上限

CloudFront 服務配額可將您的 AWS 帳戶限制為 25 個具有附加 Lambda @Edge 函數的發佈。如果您超過此配額，您可以從帳戶中刪除任何未使用的 CloudFront 分配，也可以要求提高配額。如需詳細資訊，請參閱《Service Quotas 使用者指南》中的請求提高配額。

環境變數不會傳遞至 Lambda 函數

您在 SSR 應用程式的 Amplify 主控台中指定的環境變數不會傳遞至應用程式的 AWS Lambda 功能。如需如何新增可從 Lambda 函數參考的環境變數的詳細指示讓環境變數可供伺服器端執行階段存取，請參閱，。

在美國東部 (維吉尼亞北部) 區域建立 Lambda @Edge 函數

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) 區域建立的，而不是在部署應用程式的區域建立。這是一 Lambda 限制。@Edge 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。

Next.js 應用程式使用不支援的功能

使用 Amplify 能部署的應用程式支援 Next.js 主要版本至 11 版。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，預設情況下，Amplify 會使用最新受支援的 Next.js 版本。如果您有部署到使用較舊版本的 Next.js 擴增的現有 Next.js 應用程式，您可以將應用程式移轉至「Amplify 主機」運算 SSR 提供者。如需說明，請參閱遷移 Next.js 11 SSR 應用程式以 Amplify 主機運算。

Next.js 應用程式中的影像無法載入

當您使用`next/image`元件將影像新增至 Next.js 應用程式時，影像的大小不能超過 1 MB。當您將應用程式部署到 Amplify 時，大於 1 MB 的映像會傳回 503 錯誤。這是由 Lambda @Edge 限制所造成，該限制會將 Lambda 函數 (包括標頭和內文) 所產生的回應大小限制為 1 MB。

1 MB 的限制適用於應用程式中的其他成品，例如 PDF 和文件檔案。

不支援地區

Amplify 不支援傳統 (僅限 Next.js 11) SSR 應用程式部署在每個可用 Amplify 的 AWS 區域。以下地區不支援經典版 (僅限 Next.js 11) SSR：歐洲 (米蘭) 歐洲-南 1、中東 (巴林) 我南 -1 和亞太區域 (香港) ap-East -1。

設定自訂網域

您可以將使用 Amplify 託管部署的應用程序連接到自定義域。當您使用 Amplify 來部署您的 Web 應用程式時，Amplify 會在預設網amplifyapp.com域上為您代管該應用程式，並使用如. https://branch-name.d1m7bkiki6tdw1.amplifyapp.com 當您將應用程序連接到自定義域時，用戶會看到您的應用程序託管在自定義 URL 上，例如https://www.example.com。

您可以透過認可的網域註冊商 (例如 Amazon Route 53 或) 購買自訂網域 GoDaddy。路線 53 是亞馬遜的域名系統（DNS）網絡服務。有關使用路線 53 的更多信息，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。

當您設定自訂網域時，您可以使用 Amplify 為您佈建的預設受管理憑證，也可以使用自己的自訂憑證。您可以隨時變更網域使用中的憑證。如需有關管理憑證的詳細資訊，請參閱使用 SSL/TLS 憑證。

在繼續設定自訂網域之前，請確認您已符合下列先決條件。

- 您擁有註冊的網域名稱。
- 您擁有由發行或匯入的憑證 AWS Certificate Manager。
- 您已將應用程序部署到 Amplify 託管。

  如需有關完成此步驟的詳細資訊，請參閱開始使用現有程式碼。
- 您具備網域和 DNS 術語的基本知識。

  如需網域和 DNS 的詳細資訊，請參閱了解 DNS 術語和概念。

主題

- 萬用字元子網域
- 為 Amazon 路線 53 自定義域設置自動子域
- 排解自訂網域

了解 DNS 術語和概念

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

DNS 術語

以下是 DNS 通用術語的列表。它們可協助您瞭解新增自訂網域的程序。

CNAME

「規範記錄名稱」(CNAME) 是一種 DNS 記錄，可遮罩一組網頁的網域，並讓它們看起來好像位於其他地方。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 (其中 www 為子網域) 對應至 FQDN 網域分支名稱. d1m7bki6tdw1.cloudfront.net 在 A mplify 主控台中指派給您的應用程式。

ANAME

一個 ANAME 記錄就像一個 CNAME 記錄，但在根級別。一個 ANNAME 會將您的網域根目錄指向 FQDN。該 FQDN 指向 IP 位址。

名稱伺服器

名稱伺服器是網際網路上的伺服器，專門用於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，系統就會將名稱伺服器清單指派給您的網域。

NS 記錄

NS 記錄會指向查詢您網域詳細資料的名稱伺服器。

DNS 驗證

域名系統（DNS）就像一本電話簿，將人類可讀的域名轉換為計算機友好的 IP 地址。當您在瀏覽器中輸**`https://google.com`**入時，會在 DNS 提供者中執行查閱作業，以尋找託管網站之伺服器的 IP 位址。

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，名稱和 NS 記錄。

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

Amplify 託管自定義域激活過程

使用 Amplify Hosting 添加自定義域時，需要完成多個步驟，然後才能使用自定義域查看應用程序。下列清單說明網域設定程序中的每個步驟。

建立 SSL/TLS

如果您使用受管理的憑證，請核 AWS Amplify 發 SSL/TLS 憑證以設定安全的自訂網域。

SSL/TLS 組態設定與驗證

在發行受管理憑證之前，Amplify 會驗證您是網域的擁有者。對於由 Amazon 路線 53 管理的網域，Amplify 會自動更新 DNS 驗證記錄。對於在 Route 53 以外管理的網域，您必須透過第三方 DNS 提供者手動將 Amplify 主控台中提供的 DNS 驗證記錄新增至您的網域中。

如果您使用自訂憑證，您必須負責驗證網域擁有權。

域激活

已成功驗證網域。對於在 Route 53 以外管理的網域，您必須透過第三方 DNS 提供者，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 證書是一種數字文檔，允許網絡瀏覽器使用安全的 SSL/TLS 協議識別和建立與網站的加密網絡連接。當您設定自訂網域時，您可以使用 Amplify 為您佈建的預設受管理憑證，也可以使用自己的自訂憑證。

使用受管理的憑證時，Amplify 會為連接到應用程式的所有網域發行 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，只要您的應用程式是以 Amplify 代管，就會自動續訂。如果您的網域供應商在 DNS 設定中修改或刪除 CNAME 驗證記錄，則 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，您必須從您選擇的協力廠商憑證授權單位取得憑證。接下來，將憑證匯入 AWS Certificate Manager。ACM 是一項服務，可讓您輕鬆佈建、管理和部署公用和私有 SSL/TLS 憑證，以

便 AWS 服務 與內部連線資源搭配使用。請務必在美國東部 (維吉尼亞北部) (us-east-1) 區域申請或匯入憑證。

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 中提供您的自訂憑證之後，您就可以在網域設定程序中選取它。如需將憑證匯入的指示 AWS Certificate Manager，請參閱《AWS Certificate Manager 使用指南》 AWS Certificate Manager中的〈將憑證匯入到〉。

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。在匯入憑證的情況下，ACM 不會自動管理續約。您必須負責更新自訂憑證，然後再次匯入它們。

您可以隨時變更網域使用中的憑證。例如，您可以從預設受管理憑證切換為自訂憑證，或從自訂憑證變更為受管理憑證。此外，您可以將使用中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。

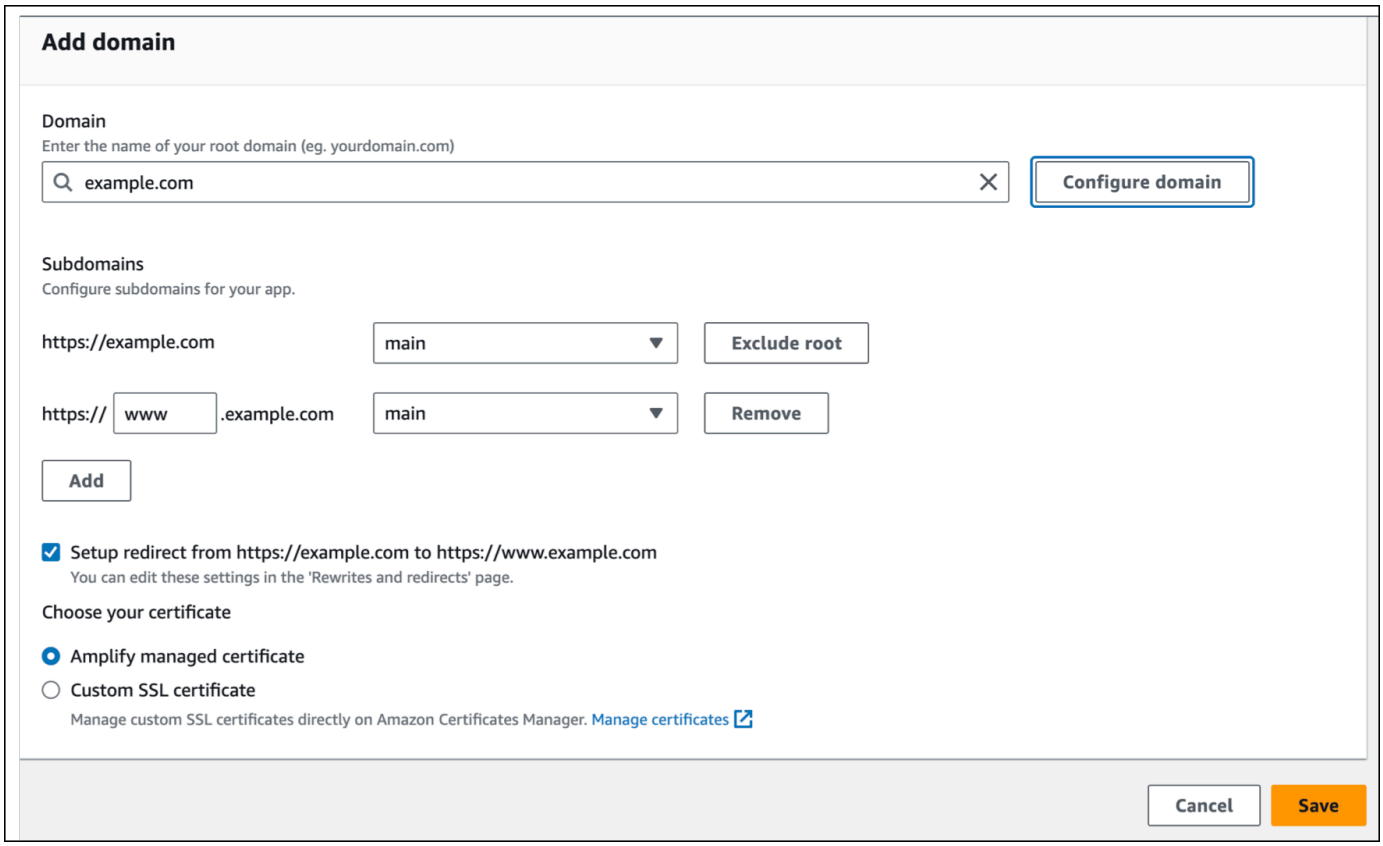
添加由 Amazon 路線 53 管理的自定義域

新增由 Route 53 管理的自訂網域

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 選擇您要連接到自定義域的應用程序。
3. 在功能窗格中，選擇 [應用程式設定]、[網域管理]。
4. 在 [網域管理] 頁面上，選擇 [新增網域]。
5. 在網域中，輸入您的根網域。例如，如果您的網域名稱是 https://example.com，請在「網域」中輸入 example.com。

   當您開始輸入時，您已經在 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。如果您尚未擁有該網域並且可以使用，則可以在 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，您將會看到子網域 https://www.example.com 和 https://example.com，其中包含從根網域設定到 www 子網域的重新導向。

(選擇性) 如果您只想新增子網域，則可以修改預設組態。若要變更預設設定，請從導覽窗格中選擇「重新寫入和重新導向」，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理憑證，也可以使用已匯入 AWS Certificate Manager的自訂協力廠商憑證。
   - 使用預設的「Amplify 受管理憑證」。
     - 選擇 Amplify 受管理憑證。
   - 使用自訂協力廠商憑證。
     a. 選擇「自訂 SSL 憑證」。
     b. 從清單中選取要使用的憑證。
9. 選擇儲存。

> **Note**
>
> DNS 最多可能需要 24 小時才能傳播和發行憑證。如需解決發生錯誤的說明，請參閱排解自訂網域。

新增由第三方 DNS 提供者管理的自訂網域

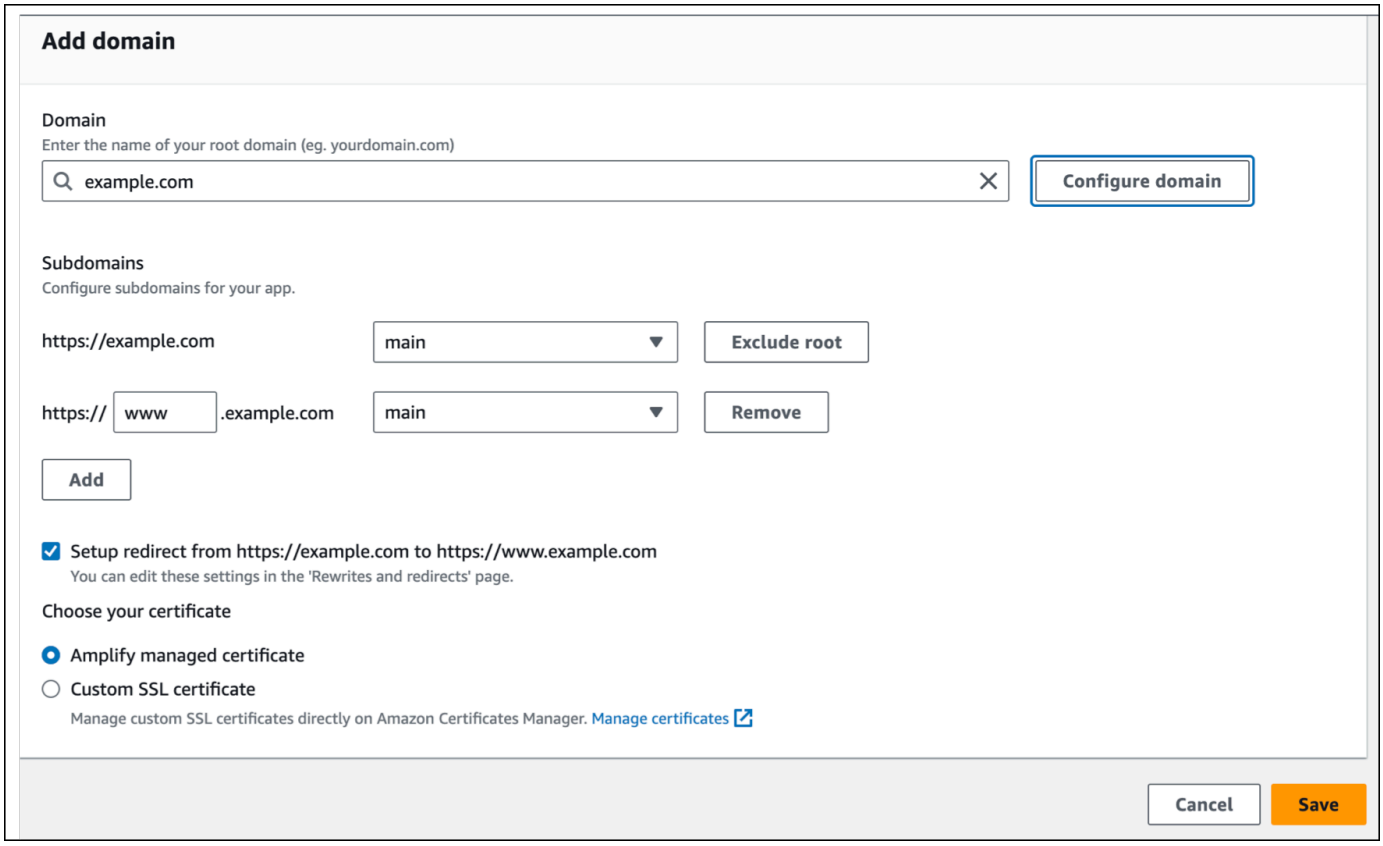
如果您不是使用 Amazon Route 53 來管理網域，您可以將由第三方 DNS 提供者管理的自訂網域新增至使用 Amplify 部署的應用程式。

如果您正在使用 GoDaddy 或 Google 網域，請參閱the section called “新增由管理的自訂網域 GoDaddy”或the section called “新增由 Google 網域管理的自訂網域”瞭解這些供應商的特定程序。

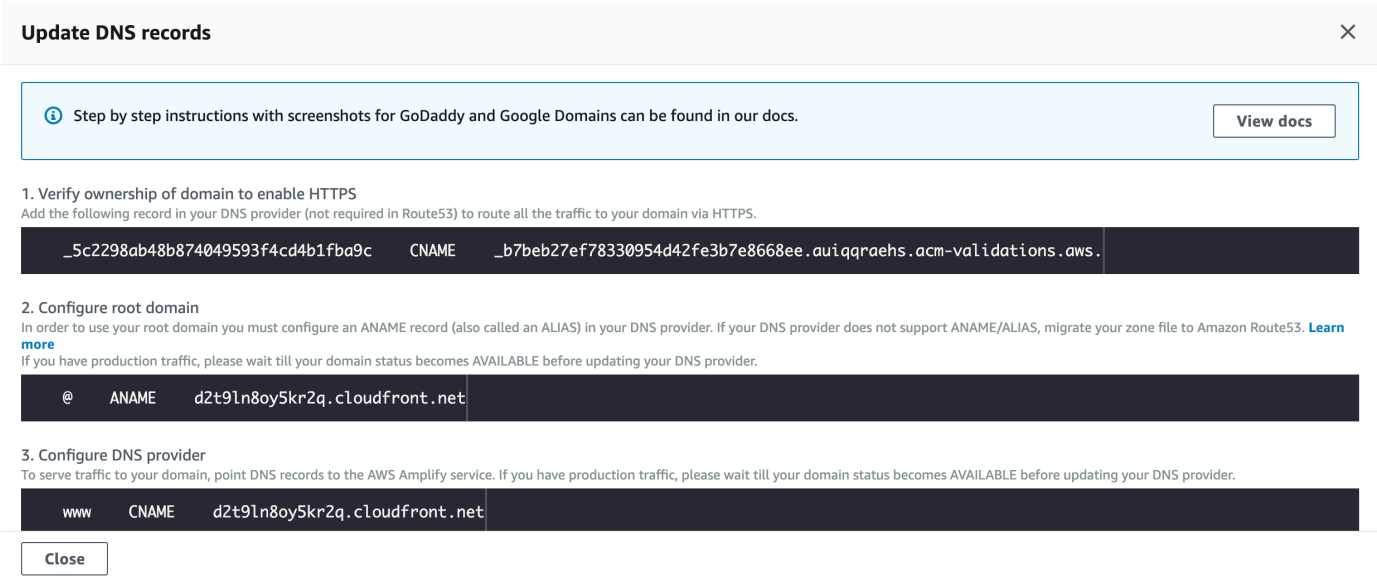
新增由第三方 DNS 提供者管理的自訂網域

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 選擇您要新增自訂網域的應用程式。
3. 在功能窗格中，選擇 [應用程式設定]、[網域管理]。
4. 在 [網域管理] 頁面上，選擇 [新增網域]。
5. 在 [網域] 中，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. 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，您將會看到子網域 https://www.example.com 和 https://example.com，其中包含從根網域設定到 www 子網域的重新導向。

   (選擇性) 如果您只想新增子網域，則可以修改預設組態。若要變更預設設定，請從導覽窗格中選擇 [重寫和重新導向]，然後設定您的網域。



7. 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理憑證，也可以使用已匯入 AWS Certificate Manager的自訂協力廠商憑證。
   - 使用預設的「Amplify 受管理憑證」。
     - 選擇 Amplify 受管理憑證。
   - 使用自訂協力廠商憑證。
     a. 選擇「自訂 SSL 憑證」。
     b. 從清單中選取要使用的憑證。
8. 選擇儲存。
9. 在 [動作] 功能表上，選擇 [檢視 DNS 記錄]。在下一個步驟中，您將使用這些 DNS 記錄與第三方網域供應商更新您的 DNS 記錄。



10. 執行以下任意一項：

    - 如果您正在使用 GoDaddy，請前往新增由管理的自訂網域 GoDaddy。
    - 如果您使用的是 Google 網域，請前往新增由 Google 網域管理的自訂網域。
    - 如果您使用不同的第三方 DNS 供應商，請前往此程序的下一個步驟。

11. 前往 DNS 供應商的網站，登入您的帳戶，然後找出網域的 DNS 管理設定。

12. 將 CNAME 設定為指向 AWS 驗證伺服器。例如，如果驗證伺服器是驗證伺服器，請輸入驗證伺服器。Amplify 會使用此資訊來驗證您的網域擁有權，並為您的網域產生 SSL/TLS 憑證。一旦擴大驗證您的域的所有權，所有流量都將使用 HTTPS/2 提供服務。

> **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以 Amplify 代管，就會自動續訂。如果 CNAME 驗證記錄已修改或刪除，則 Amplify 無法續訂憑證。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) 會立即開始嘗試驗證擁有權。隨著時間的推移，檢查變得不那麼頻繁。如果您在建立應用程式幾個小時後新增或更新 CNAME 記錄，這可能會導致您的應用程式卡在擱置驗證狀態。

13. 設定第二個 CNAME 記錄 (例如 https://*.example.com)，將您的子網域指向 Amplify 網域。如果您有生產流量，建議您在 Amplify 主控台中的網域狀態顯示為「可用」之後，更新此 CNAME 記錄。
14. 將「全名/別名」記錄設定為指向您網域的根網**amplifyapp**域 (例如 https://example.com)。ANAME 記錄會將您網域的根目錄指向主機名稱。如果您有生產流量，建議您在主控台中的網域狀態顯示為「可用」之後，更新 ANAME 記錄。對於沒有 ANAME/ALIAS 支援的 DNS 供應商，我們強烈建議您將 DNS 移轉至路由 53。如需詳細資訊，請參閱將 Amazon Route 53 設定為您的 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。如需協助解決發生的錯誤，請參閱疑難排解自訂網域。

新增由管理的自訂網域 GoDaddy

新增由管理的自訂網域 GoDaddy

1. 遵循程序中的步驟 1 到 9 the section called “新增由第三方 DNS 提供者管理的自訂網域”。
2. 登入您的 GoDaddy 帳戶。
3. 在您的網域清單中，找到要新增的網域，然後選擇 [管理 DNS]。
4. 在 DNS 頁面上，在「DNS 記錄」區段中 GoDaddy 顯示您網域的記錄清單。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. 在「類型」中選擇「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，請在「名稱」中輸入 www。
   d. 在「值」中查看 Amplify 主控台中的 DNS 記錄，然後輸入值。如果 Amplify 控制台將您的應用程序的域顯示為 xxxxxxxxxxxx.cloud前端網，請在「值」中輸入 xxxxxxxxxxxx.cloudfront.net。
   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 會為您的網域產生一個 SSL/TLS 憑證。

a. 在「類型」中選擇「CNAME」。

b. 在名稱中，輸入子網域。

例如，如果 Amplify 控制台中用於驗證您子網域擁有權的 DNS 記錄是 _c3e2d7e656b7f46f6980fdc0e 作為「名稱」，請僅輸入 _c3e2d7eaf1e 656f46f46cd6980fdc0e。

c. 在值中，輸入 ACM 驗證憑證。

例如，如果驗證伺服器是驗證伺服器，請針對值輸入示例。

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以 Amplify 代管，就會自動續訂。如果 CNAME 驗證記錄已修改或刪除，則 Amplify 無法續訂憑證。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「全名」/「別名」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 設定為您的 DNS 服務。

如果您想保留 GoDaddy 為您的供應商並更新根網域，請新增「轉寄」並設定網域轉址：

a. 在 [DNS] 頁面上，找出頁面頂端的功能表，然後選擇 [轉寄]。

b. 在「網域」區段中，選擇「新增轉址」。

c. 選擇 http://，然後輸入要轉寄至的子網域名稱 (例如，www.example.com) 做為「目的地網址」。

d. 針對 [轉寄類型]，選擇 [暫時 (302)]。

e. 選擇，保存。

新增由 Google 網域管理的自訂網域

新增由 Google 網域管理的自訂網域

1. 請遵循程序中的步驟 1 到 9 新增由協力廠商 DNS 提供者管理的自訂網域。

2. 在 https://domains.google.com 登入您的帳戶，然後在左側導覽窗格中選擇「我的網域」。
3. 在您的網域清單中，找到要新增的網域，然後選擇 [管理]。
4. 在左側導覽窗格中，選擇 [DNS]。Google 會顯示您網域的資源記錄。您需要新增兩個新的 CNAME 記錄。
5. 建立第一個 CNAME 記錄，將所有子網域指向 Amplify 網域，如下所示：

   a. 在「主機名稱」中，僅輸入子網域名稱。例如，如果您的子網域是 www.example.com，請輸入 www 做為主機名稱。

   b. 在「類型」中選擇「CNAME」。

   c. 在「資料」中，輸入 Amplify 主控台中可用的值。

      如果 Amplify 控制台將您的應用程式的網域顯示為網域，請輸入資料的網域。

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 會為您的網域產生一個 SSL/TLS 憑證。

   a. 在「主機名稱」中，輸入子網域。

      例如，如果 Amplify 控制台中用於驗證子網域擁有權的 DNS 記錄是 _c3e2d7e656b7f46f6980fdc0e 作為主機名稱，請僅輸入 _c3e2d7eaf1e 656f46f46cd6980fdc0e。

   b. 在「類型」中選擇「CNAME」。

   c. 在資料中，輸入 ACM 驗證憑證。

      例如，如果驗證伺服器是 _cf 1z2npwt9vz例9c1j4xzc9wl.2te3lym6kr.ACM-驗證。 中，輸入 _cf 1z2npwt9vz範例93c1j4xzc9wl.2te3iym6kr.ACM-驗證。 用於資料。

7. 選擇儲存。

> **Note**
>
> 預設的 Amplify; 憑證 AWS Certificate Manager (ACM) 產生的有效期為 13 個月，只要您的應用程式是以 Amplify 代管，就會自動續訂。如果 CNAME 驗證記錄已修改或刪除，則 Amplify 無法續訂憑證。您必須在 Amplify 主控台中刪除並再次新增網域。

8. Google 域名對於「全名/別名」記錄的支持已處於預覽狀態。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 設定為您的 DNS 服務。如果您想要將 Google 網域保留為您的供應商並更新根

網域，請設定子網域轉址。找到您 Google 網域的「網站」頁面。然後選擇轉發域並在 Web 轉發頁面上配置您的轉發。

> **Note**
>
> Google 網域的 DNS 設定更新最多可能需要 48 小時才會生效。如需解決發生錯誤的說明，請參閱疑難排解自訂網域。

更新網域的 SSL/TLS 憑證

您可以隨時變更網域使用中的 SSL/TLS 憑證。例如，您可以從使用受管理憑證變更為使用自訂憑證。您也可以變更網域使用的自訂憑證。如需有關憑證的詳細資訊，請參閱使用 SSL/TLS 憑證。

使用下列程序來更新網域所使用的憑證類型或自訂憑證。

更新網域憑證

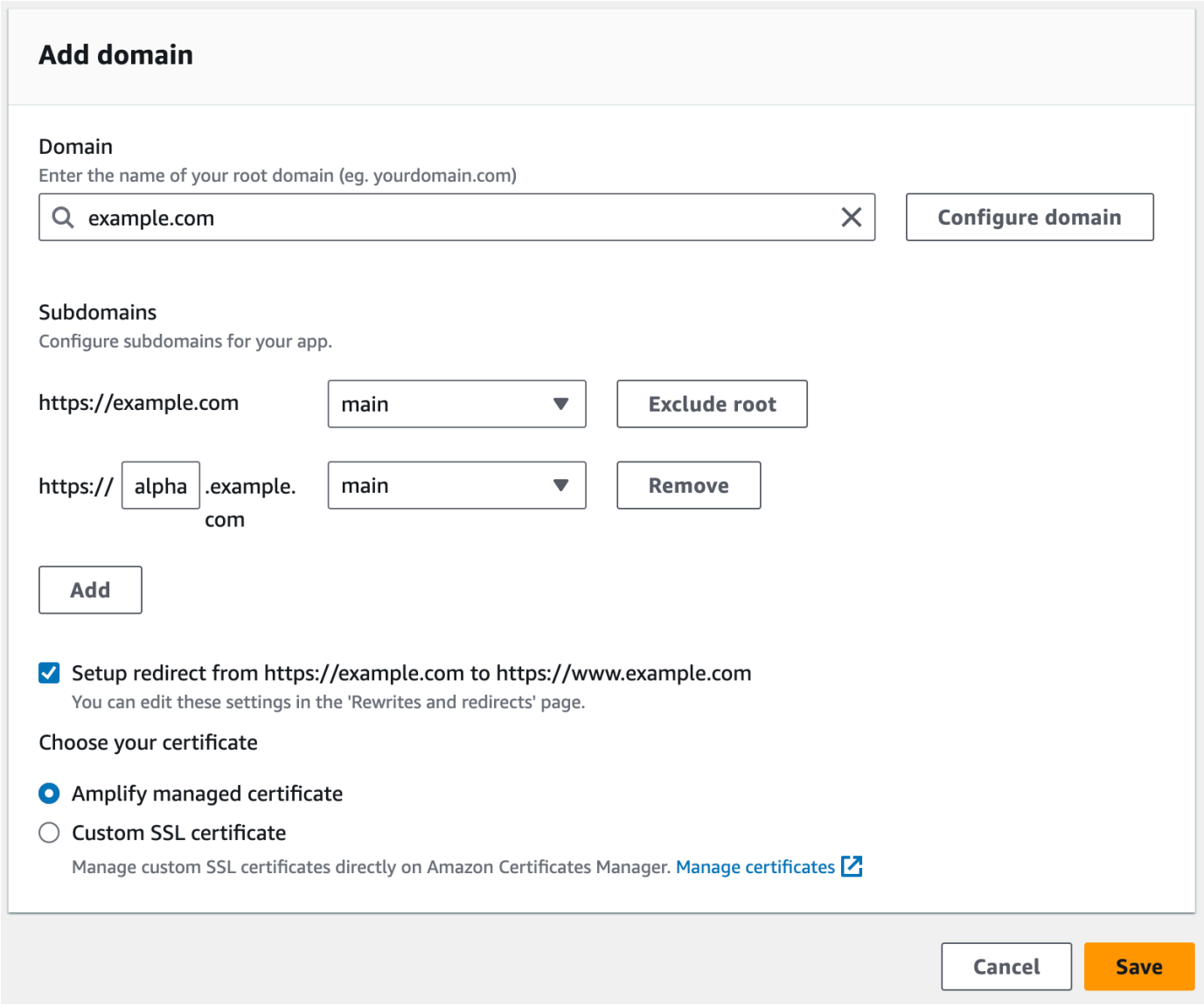
1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 選擇您要更新的應用程式。
3. 在功能窗格中，選擇 [應用程式設定]，然後選擇 [網域管理]。
4. 在 [網域管理] 頁面上，選擇 [管理網域]。
5. 在您網域的詳細資料頁面上，找到 [選擇您的憑證] 區段。根據您要進行的變更類型，更新憑證的程序會有所不同。
   - 從自訂憑證變更為預設的 Amplify 受管理憑證
     - 選擇 Amplify 受管理憑證。
   - 從受管理憑證變更為自訂憑證
     a. 選擇「自訂 SSL 憑證」。
     b. 從清單中選取要使用的憑證。
   - 將自訂憑證變更為不同的自訂憑證
     - 對於自訂 SSL 憑證，請從清單中選取要使用的新憑證。
6. 選擇更新。網域的狀態詳細資料會指出 Amplify 已針對受管理憑證啟動 SSL 建立程序，或是自訂憑證的組態程序。

管理子網域

子網域是顯示在網域名稱前面的 URL 部分。例如，萬維網是亞馬遜網站的子域名，而 AWS 是 aws.amazon.com 的子域名。如果您已經有一個生產網站，則可能只想連接一個子域。子網域也可以是多層次，例如，.alpha.example.com 擁有多層次的子網域是 .alpha。

僅新增子網域

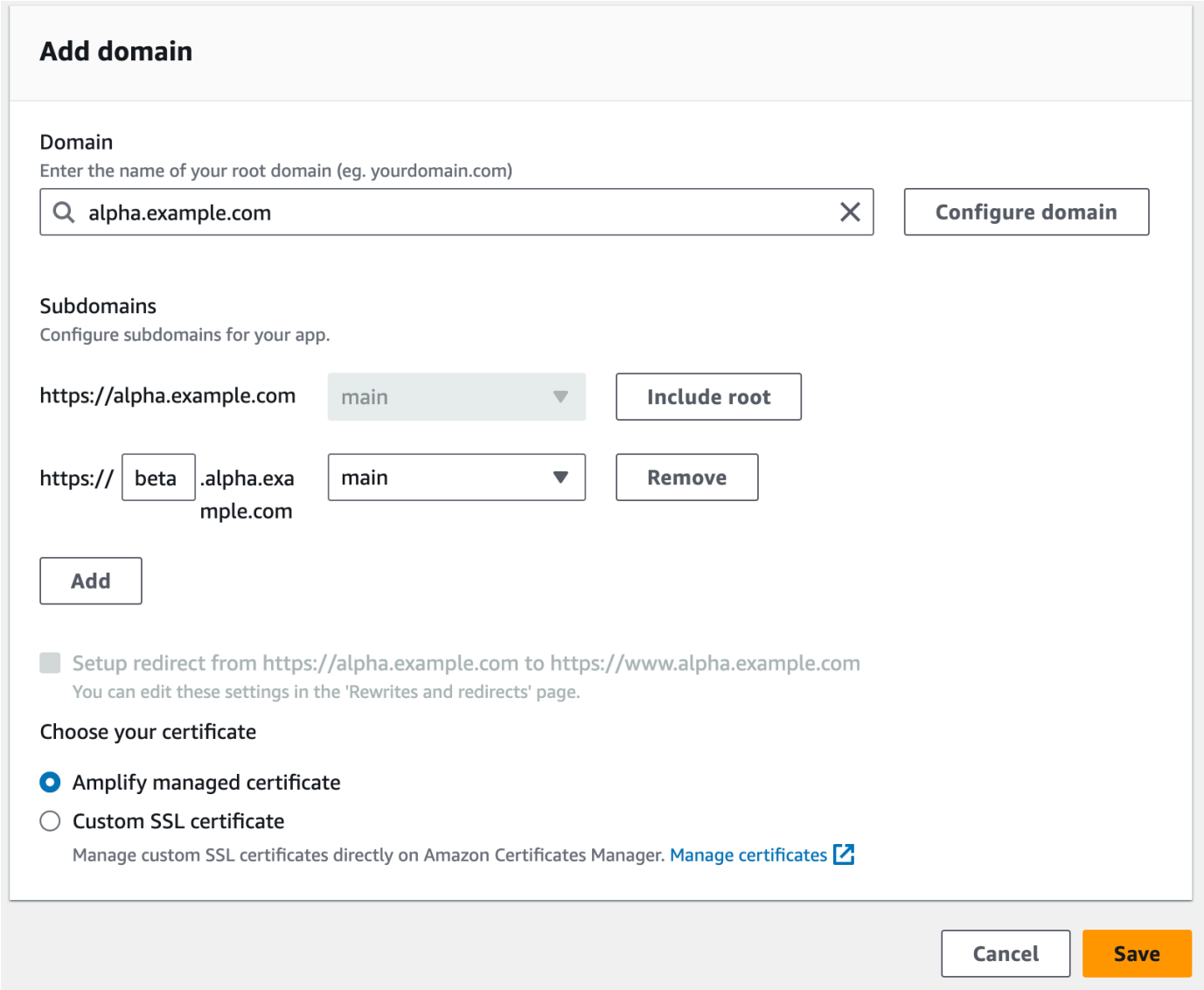
1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 選擇您要新增子網域的應用程式。
3. 在功能窗格中，選擇 [應用程式設定]，然後選擇 [網域管理]。
4. 在 [網域管理] 頁面上，選擇 [新增網域]。
5. 在 [網域] 中，輸入根網域的名稱，然後選擇 [設定網域]。例如，如果您的網域名稱是 https://example.com，請在「網域」中輸入 example.com。
6. 選擇排除根目錄，然後修改子網域的名稱。例如，如果域是 example.com，則可以將其修改為僅添加子域 alpha，如下面的屏幕截圖所示。



若要新增多層次子網域

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 選擇您要新增多層次子網域的應用程式。
3. 在功能窗格中，選擇 [應用程式設定]，然後選擇 [網域管理]。
4. 在 [網域管理] 頁面上，選擇 [新增網域]。
5. 在「網域」中，輸入具有子網域的網域名稱，選擇「排除根目錄」，然後修改子網域以新增層級。

   例如，如果您有一個名為 alpha.example.com 的網域，而且您想要建立一個多層次子網域名稱 beta.alpha.example.com，您可以輸入 beta 作為子網域值，如下列螢幕擷取畫面所示。



若要新增或編輯子網域

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子網域。

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 選擇您要管理子網域的應用程式。
3. 在功能窗格中，選擇 [應用程式設定]，然後選擇 [網域管理]。
4. 在 [網域管理] 頁面上，選擇 [管理網域]。
5. 在 「編輯網域」 中，您可以視需要編輯現有的子網域。
6. (選擇性) 若要新增子網域，請選擇「新增」。

7. 選擇 Update (更新) 以儲存您的設定。

萬用字元子網域

Amplify 託管現在支持通配符子域。萬用字元子網域是 Catch-All 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關聯至特定分支時，您可以在任何子網域中向應用程式的使用者提供相同的內容，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，如果您為應用程序的特定分支指定通配符子域`*.example.com`，則以 example.com 結尾的任何 URL 都將被路由到該分支。在此情況下，`prod.example.com`會將要求`dev.example.com`和的要求路由至`*.example.com`子網域。

請注意，Amplify 僅支援自訂網域的萬用字元子網域。您無法在預設`amplifyapp.com`網域中使用此功能。

下列需求適用於萬用字元子網域：

- 子網域名稱必須僅以星號 (*) 指定。
- 您無法使用萬用字元來取代部分子網域名稱，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：子網域 .*.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。

若要新增或刪除萬用字元子網域

將自定義域添加到應用程序後，您可以為應用程序分支添加通配符子域。

1. 登入 AWS Management Console 並開啟「Amplify 主機」主控台。
2. 選擇您要管理通配符子域的應用程序。
3. 在功能窗格中，選擇 [應用程式設定]，然後選擇 [網域管理]。
4. 在 [網域管理] 頁面上，選擇 [管理網域]。
5. 在編輯網域中，您可以新增或刪除萬用字元子網域。
    - 若要新增萬用字元子網域

        a. 選擇新增。

        b. 對於子網域，請輸入*****.

        c. 對於您的應用分支，請從列表中選擇分支名稱。

在以下示例屏幕截圖中，已為應用程序的dev分支創建了*.example.com通配符子域。

d. 選擇 Update (更新) 以儲存您的設定。

- 若要刪除萬用字元子網域

a. 選擇子網域名稱旁邊的「移除」。未明確配置的子域的流量停止，並且 Amplify 託管返回 404 狀態碼給這些請求。

b. 選擇 Update (更新) 以儲存您的設定。

為 Amazon 路線 53 自定義域設置自動子域

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，如果您連接您的開發分支，Amplify 可以自動創建 dev .exampledomain.com。當您刪除分支時，任何關聯的子網域都會自動刪除。

為新連線的分支設定自動子網域建立

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 選擇連接到 Route 53 中管理的自定義域的應用程序。
3. 在功能窗格中，選擇 [應用程式設定]，然後選擇 [網域管理]。
4. 在 [網域管理] 頁面上，選擇 [管理網域]。
5. 選取左下角的子網域自動偵測核取方塊。

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

具有子網域的網頁預覽

使用上述說明啟用子域自動檢測後，您的應用程序的提取請求 Web 預覽也將可以通過自動創建的子域訪問。關閉拉取要求時，會自動刪除相關聯的分支和子網域。如需有關為提取要求設定網頁預覽的詳細資訊，請參閱提取要求的網頁預覽。

排解自訂網域

如果在主 AWS Amplify 控台中將自訂網域新增至應用程式時遇到問題，請參閱本節中的下列主題以取得疑難排解說明。

如果您在這裡找不到問題的解決方案，請聯絡 AWS Support。如需詳細資訊，請參閱 AWS Support 使用者指南中的建立支援案例。

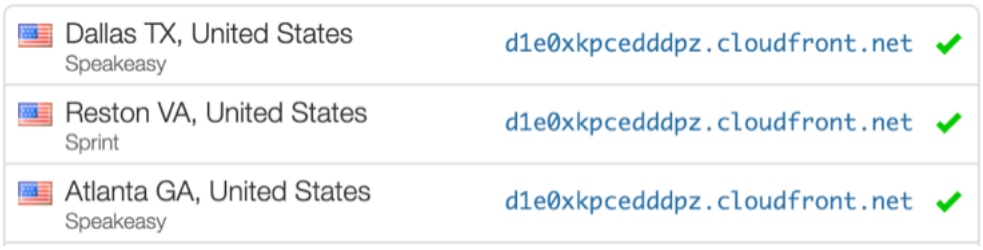
主題

- 如何確認 CNAME 是否解析？
- 我由第三方託管的網域停留在「等待驗證」狀態
- 我使用 Amazon Route 53 託管的域被卡在待驗證狀態
- 我收到一個 CNAME 錯誤 AlreadyExistsException
- 我收到「需要額外驗證」錯誤
- 我在 CloudFront 網址上收到 404 錯誤
- 我在訪問我的域名時收到 SSL 證書或 HTTPS 錯誤

如何確認 CNAME 是否解析？

1. 向第三方網域供應商更新 DNS 記錄後，您可以使用 dig 或免費網站 (例如 https://www.whatsmydns.net/) 來確認您的 CNAME 記錄是否正確解析。下列螢幕擷取畫面將示範如何使用您的 CNAME 記錄檢查網域名稱。

2. 選擇「搜索」，然後顯示 CNAME 的結果。下列螢幕擷取畫面是驗證 CNAME 是否正確解析為 Cloudfront.net 網址的結果清單範例。



我由第三方託管的網域停留在「等待驗證」狀態

1. 如果您的自訂網域停留在 [擱置驗證] 狀態，請確認您的CNAME記錄正在解析。如需執行此工作的指示，請參閱先前的疑難排CNAME解主題「如何驗證我已解決」。
2. 如果您的CNAME記錄無法解析，請向您的網域供應商確認該CNAME項目存在於您的 DNS 設定中。

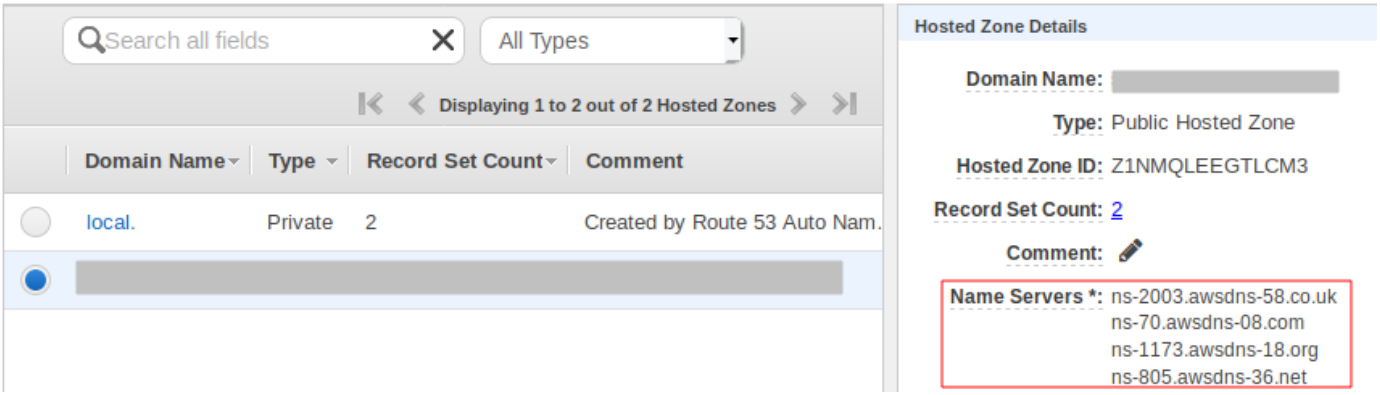
> **Important**
>
> 建立自訂網域後，請務必立即更新您的CNAME記錄。在 Amplify 控制台中創建應用程序後，系統會每隔幾分鐘檢查一次您的CNAME記錄，以確定是否可以解決。如果一小時後仍無法解決，則每隔幾個小時就會進行一次檢查，這可能會導致您的網域可以使用延遲。如果您在建立應用程式後幾個小時內新增或更新CNAME記錄，這是造成應用程式卡在「擱置驗證」狀態的最有可能原因。

3. 如果您已確認CNAME記錄存在，則可能是您的 DNS 供應商發生問題。您可以聯絡 DNS 提供者以診斷 DNS 驗證無法解析CNAME的原因，或者您可以將 DNS 遷移到 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。

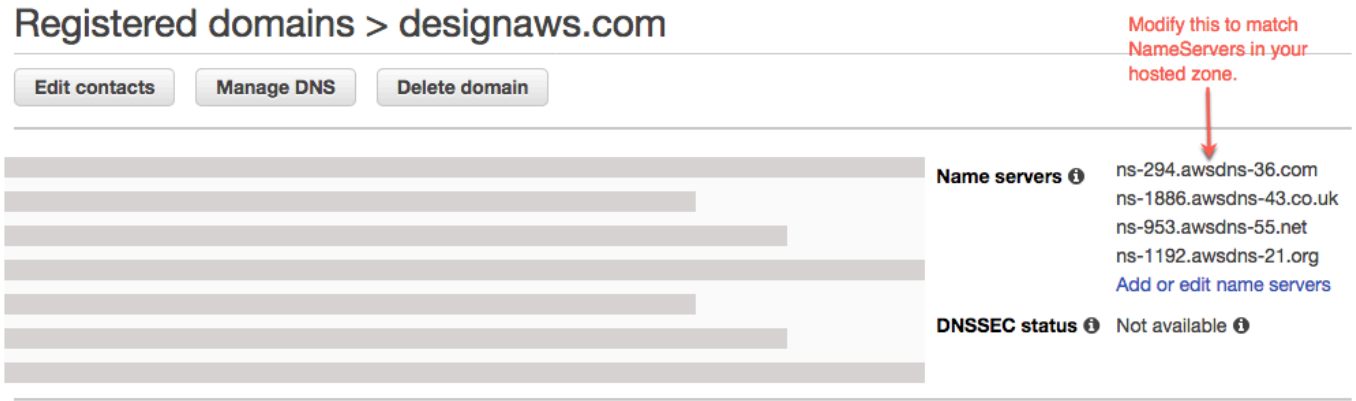
我使用 Amazon Route 53 託管的域被卡在待驗證狀態

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 Amplify 所發行的名稱伺服器不同。執行下列步驟來診斷錯誤的原因。

1. 登錄到亞 Amazon Route 53 控制台
2. 在功能窗格中，選擇 [託管區域]，然後選擇您要連線的網域名稱。
3. 記錄「託管區域詳細資料」區段中的名稱伺服器值。您需要這些值才能完成下一個步驟。Route 53 主控台的下列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)。確認顯示在 [已註冊的網域] 區段中的名稱伺服器與您在上一個步驟中從 [託管區域詳細資料] 區段中記錄的名稱伺服器值相符。如果它們不相符，請編輯名稱伺服器值以符合託管區域中的值。Route 53 主控台的下列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 AWS Support。如需詳細資訊，請參閱 AWS Support 使用者指南中的建立支援案例。

我收到一個 CNAME 錯誤 AlreadyExistsException

如果出現 CNAME AlreadyExistsException 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 分發。執行下列步驟來診斷錯誤的原因。

1. 登入 Amazon CloudFront 主控台並確認您沒有將此網域部署到任何其他分發。一次可以將單一 CNAME記錄附加到一個 CloudFront 分佈。
2. 如果您先前將網域部署到 CloudFront 發行版，則必須將其移除。
   a. 在左側導覽功能表中選擇 「分配」。
   b. 選取要編輯的發佈名稱。
   c. 選擇 [一般] 索引標籤。在 Settings (設定) 區段中，選擇 Edit (編輯)。
   d. 從替代網域名稱 (CNAME) 移除網域名稱。然後選擇，保存更改。
3. 檢查此網域是否已連線至您擁有的其他 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 用於其他應用程式，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，例如部落格。
4. 如果此網域已成功連線至其他應用程式，然後在最近一小時內刪除，請在至少一小時後再試一次。如果您在 6 小時後仍然看到此例外狀況，請聯絡 AWS Support。如需詳細資訊，請參閱 AWS Support 使用者指南中的建立支援案例。

我收到「需要額外驗證」錯誤

如果您收到「需要其他驗證」錯誤訊息，這表示 AWS Certificate Manager (ACM) 需要其他資訊來處理此憑證要求。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 AWS Support。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 網址上收到 404 錯誤

為了服務流量，Amplify 託管通過 CNAME 記錄指向 CloudFront URL。在將應用程式連線到自訂網域的過程中，Amplify 主控台會顯示應用程式的 CloudFront URL。但是，您無法使用此 CloudFront

URL 直接訪問您的應用程序。它返回一個 404 錯誤。您的應用程式只能使用 Amplify 應用程式 URL (例如`https://main.d5udybEXAMPLE.amplifyapp.com`，或您的自訂網域`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，您可以配置指向應用程序主線分支的域`www.example.com`，但也可以配置`dev.example.com`該域指向同一應用程序的 dev 分支。因此，您必須根據其配置的子域訪問應用程序，以便 Amplify 可以相應地路由請求。

我在訪問我的域名時收到 SSL 證書或 HTTPS 錯誤

如果您已使用協力廠商 DNS 提供者設定憑證授權 (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。要解決此問題，您需要添加 CAA 記錄以信任至少一個 Amazon 的證書頒發機構域。下列程序說明您需要執行的步驟。

若要新增 CAA 記錄以信任 Amazon 憑證授權單位

1. 與您的網域供應商設定 CAA 記錄，以信任至少一個 Amazon 的憑證授權機構網域。如需有關設定 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單位授權 (CAA) 問題。
2. 請使用下列其中一種方法來更新您的 SSL 憑證：
   - 使用 Amplify 控制台手動更新。

     > **Note**
     >
     > 此方法將導致您的自定義域停機時間。

     a. 登入 AWS Management Console 並開啟 Amplify 大控制台。
     b. 選擇您要添加 CAA 記錄的應用程序。
     c. 在功能窗格中，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. 再次將您的應用程序 Connect 到自定義域。此程序會發出新的 SSL 憑證，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，請使用與您使用的網域供應商相對應的下列程序之一。

        - 添加由 Amazon 路線 53 管理的自定義域.

    - 新增由第三方 DNS 提供者管理的自訂網域.
    - 新增由管理的自訂網域 GoDaddy.
    - 新增由 Google 網域管理的自訂網域.
- 請聯絡 AWS Support 以重新發行您的 SSL 憑證。

配置構建設置

當您使用 Amplify Hosting 部署應用程序時，它會通過檢查存儲庫中的`package.json`文件來自動檢測前端框架和關聯的構建設置。您可以使用以下選項來存儲應用程序的構建設置：

- 在 Amplify 控制台中保存構建設置-Amplify 控制台會自動檢測構建設置並保存它們，以便可以通過 Amplify 控制台訪問它們。Amplify 將這些設置應用於所有分支，除非存儲庫中存儲了一個`amplify.yml`文件。
- 將構建設置保存在存儲庫中-下載`amplify.yml`文件並將其添加到存儲庫的根目錄中。

您可以選擇 [應用程式設定]、[建置設定]，在 Amplify 主控台中編輯應用程式的建置設定。構建設置將應用於應用程序中的所有分支，但存儲庫中保存了`amplify.yml`文件的分支除外。

> **Note**
>
> 只有當應用程式設定為持續部署並連線到 git 儲存庫時，才能在 Amplify 主控台的 [應用程式設定] 功能表中看見建置設定。如需有關此部署類型的指示，請參閱開始使用現有程式碼。

建立規格指令和設定

組建規格 YAML 包含組建命令集合，以及 Amplify 用來執行組建的相關設定。下列清單說明這些設定及其使用方式。

version

Amplify YAML 版本號碼。

批准

此應用程式所在的儲存區域中的路徑。除非定義了多個應用程式，否則將

env

將環境變數新增至此區段。您也可以使用主控台新增環境變數。

後端

執行 Amplify CLI 命令以佈建後端、更新 Lambda 函數或 GraphQL 結構描述，做為持續部署的一部分。了解如何使用您的前端部署後端。

前端

運行前端構建命令。

test

在測試階段執行命令。了解如何將測試添加到您的應用程序。

建置階段

前端，後端和測試有三個階段，代表在構建的每個序列中運行的命令。

- 預生成-預生成腳本在實際構建開始之前運行，但 Amplify 後安裝依賴關係。
- build - 您的建置命令。
- PostBuild-建置後指令碼會在建置完成後執行，並且 Amplify 已將所有必要的成品複製到輸出目錄。

建置路徑

用來執行組建的路徑。Amplify 使用此路徑來尋找您的組建成品。例如，如果您沒有指定路徑，則 Amplify 會使用 monorepo 應用程式根目錄。apps/app

人造物 > 基本目錄

組建加工品所在的目錄。

人工因素 > 檔案

從您要部署的成品中指定檔案。輸入**/*以包含所有檔案。

快取

buildspec 的緩存字段用於緩存構建時依賴關係，例如 node_modules 文件夾，並根據客戶的應用程序內置的包管理器和框架自動建議。在第一次構建期間，這裡的任何路徑都會被緩存，並且在後續構建中，我們重新膨脹緩存並在可能的情況下使用這些緩存的依賴關係來加快構建時間。

下面的示例構建規範演示了基本的 YAML 語法：

建置規格 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
        - path
test:
  phases:
    preTest:
      commands:
        - *enter command*
    test:
      commands:
        - *enter command*
    postTest:
      commands:
        - *enter command*
  artifacts:
    files:
        - location
        - location
```

```
    configFilePath: *location*
    baseDirectory: *location*
```

分支特定的構建設置

您可以使用 bash shell 指令碼來配置分支特定的建置設定。例如，如果分支名稱為 main，則下列指令碼會使用系統環境變數 $AWS_ BRANCH 來執行一組命令，如果分支名稱為 dev，則使用不同的命令集。

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

導覽至子資料夾

對於龍斷，用戶希望能夠cd進入一個文件夾來運行構建。執行cd命令之後，它會套用至組建的所有階段，因此您不需要在不同階段重複指令。

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

使用前端部署後端

此命amplifyPush令是協助程式指令碼，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
```

設置輸出文件夾

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
```

將套件安裝為組建的一部分

您可以在建置期間使用npm或yarn指令來安裝套件。

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

使用私有 npm 註冊表

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 映像檔會以名為的非特權使用者執行您的程式碼。amplifyAmplify 會授與此使用者使用 Linux sudo 命令執行作業系統命令的權限。如果您想要安裝作業系統套件以尋找遺失的相依性，您可以使rpm用yum和sudo.

下面的示例構建部分演示了使用sudo命令安裝操作系統包的語法。

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

每個建置的索引鍵/值儲存體

在建置階段envCache提供索引鍵值儲存。儲存在中的值只envCache能在建置期間修改，而且可以在下一個組建中重複使用。使用envCache，我們可以儲存已部署環境的相關資訊，並在連續組建中將其提供給組建容器。與儲存在中的值不同envCache，組建期間對環境變數的變更不會保留在 future 的組建中。

使用範例：

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

跳過構建以進行提交

要在特定提交上跳過自動構建，請在提交消息末尾添加文本 [skip p-cd]。

停用自動組建

您可以將 Amplify 設定為停用每個程式碼提交時的自動建置。若要進行設定，請選擇 [應用程式設定]、[一般]，然後捲動至列出連線分支的 [分支] 區段。選取分支，然後選擇 [動作] > [停用 auto 建置]。如此一來，該分支後續遞交時不會再觸發新建置。

啟用或禁用基於差異的前端構建和部署

您可以配置 Amplify 使用基於差異的前端構建。如果啟用，則在每個構建啟動時，Amplify 會默認嘗試在您appRoot的或/src/文件夾上運行 diff。如果 Amplify 找不到任何差異，它會略過前端組建、測試 (如果已設定) 和部署步驟，而不會更新您的託管應用程式。

配置基於差異的前端構建和部署

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 選擇要配置基於差異的前端構建和部署的應用程序。
3. 在導覽窗格中，選擇 [應用程式設定] > [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因您啟用或停用差異型前端建置與部署而有所不同。
   - 啟用基於差異的前端構建和部署
     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`。
   - 禁用基於差異的前端構建和部署
     - 執行以下任意一項：
       - 在「管理變數」區段中，找出AMPLIFY_DIFF_DEPLOY。針對數值，輸入 `false`。
       - 移除環AMPLIFY_DIFF_DEPLOY境變數。
6. 選擇儲存。

或者，您可以設置AMPLIFY_DIFF_DEPLOY_ROOT環境變量，以使用相對於存儲庫根目錄的路徑覆蓋默認路徑，例如dist。

啟用或禁用基於差異的後端構建

您可以將 Amplify 託管配置為使用AMPLIFY_DIFF_BACKEND環境變量使用基於差異的後端構建。當您啟用基於差異的後端構建時，在每個構建開始時，Amplify 會嘗試對存儲庫中的amplify文件夾運行差異。如果 Amplify 沒有發現任何差異，它會跳過後端構建步驟，並且不會更新後端資源。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。

如果您當前在後端階段的構建設置中指定了自定義命令，則條件後端構建將無法正常工作。如果要運行這些自定義命令，則必須將它們移動到應用程序amplify.yml文件中構建設置的前端階段。

若要設定基於差異的後端組建

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 選擇要配置基於差異的後端構建的應用程序。
3. 在導覽窗格中，選擇 [應用程式設定] > [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因您啟用或停用差異型後端組建而有所不同。
   - 啟用基於差異的後端構建
     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`。
   - 若要停用基於差異的後端建置
     - 執行以下任意一項：
       - 在「管理變數」區段中，找出AMPLIFY_DIFF_BACKEND。針對數值，輸入 `false`。
       - 移除環AMPLIFY_DIFF_BACKEND境變數。
6. 選擇儲存。

壟斷構建設置

當您將多個項目或微服務存儲在單個存儲庫中時，它被稱為 monorepo。您可以使用 Amplify 託管在 monorepo 中部署應用程序，而無需創建多個構建配置或分支配置。

Amplify 支持通用專賣中的應用程序以及使用 npm 工作區，pnpm 工作區，Yarn 工作區，NX 和植物園創建的壟斷中的應用程序。當您部署應用程式時，Amplify 會自動偵測您正在使用的 MONOrepo 建置

工具。Amplify 會自動套用建置設定，適用於 npm 工作區、Yarn 工作區或 NX 中的應用程式。植物園和 pnpm 應用程序需要額外的配置。如需詳細資訊，請參閱 配置植物園和 PNPM 單回購應用程序。

您可以在 Amplify 控制台中保存 monorepo 的構建設置，也可以下載`amplify.yml`文件並將其添加到存儲庫的根目錄中。Amplify 將保存在控制台中的設置應用於所有分支，除非它在存儲庫中找到`amplify.yml`文件。存在`amplify.yml`檔案時，其設定會覆寫 Amplify 主控台中儲存的任何建置設定。

Monorepo 構建規範 YAML 語法

Monorepo 組建規格的 YAML 語法與包含單一應用程式的存放庫的 YAML 語法不同。對於 monorepo，您可以在應用程序列表中聲明每個項目。您必須為您在 monorepo 構建規範中聲明的每個應用程序提供以下附加`appRoot`密鑰：

批准

應用程式在儲存庫中啟動的根目錄。此機碼必須存在，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指示，請參閱設置放大器的環境變量。

下面的 monorepo 構建規範例演示瞭如何在同一個存儲庫中聲明多個 Amplify 應用程序。這兩個應用程式`react-app`、和`angular-app`會在`applications`清單中宣告。每個應用程序的`appRoot`密鑰表示該應用程序位於 repo 的`apps`根文件夾中。

該`buildpath`屬性設置`/`為從 monorepo 項目根目錄運行和構建應用程序。

Monorepo 構建規範 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*
```

設置放大器的環境變量

當您部署存儲在 monorepo 中的應用程序時，應用程序的AMPLIFY_MONOREPO_APP_ROOT環境變量必須與應用程序根目錄的路徑具有相同的值，相對於存儲庫的根目錄。例如，一個以名為ExampleMonorepo的根資料夾命名的 monorepoapps，其中包含、app1app2、和app3具有下列目錄結構：

```
ExampleMonorepo
  apps
    app1
    app2
    app3
```

在此範例中，的AMPLIFY_MONOREPO_APP_ROOT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，主控台會使用您為應用程式根目錄路徑指定的值自動設定AMPLIFY_MONOREPO_APP_ROOT環境變數。但是，如果您的 monorepo 應用程序已存在於 Amplify 中或使用部署 AWS CloudFormation，則必須在 Amplify 控制台的「AMPLIFY_MONOREPO_APP_ROOT環境變量」部分中手動設置環境變量。

在部署期間自動設定環境變數

下列指示示範如何使用 Amplify 主控台部署單一回購應用程式。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。

若要使用 Amplify 控制台部署單一回購應用程式

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 選擇右上角的新應用程序，主機 Web 應用程序。
3. 在「託管您的 Web 應用程式」頁面上，選擇您的 Git 提供者，然後選擇「繼續」。
4. 在「新增儲存區域分支」頁面上，執行下列動作：

    a. 從「最近更新的儲存庫」清單中選擇儲存庫的名稱。

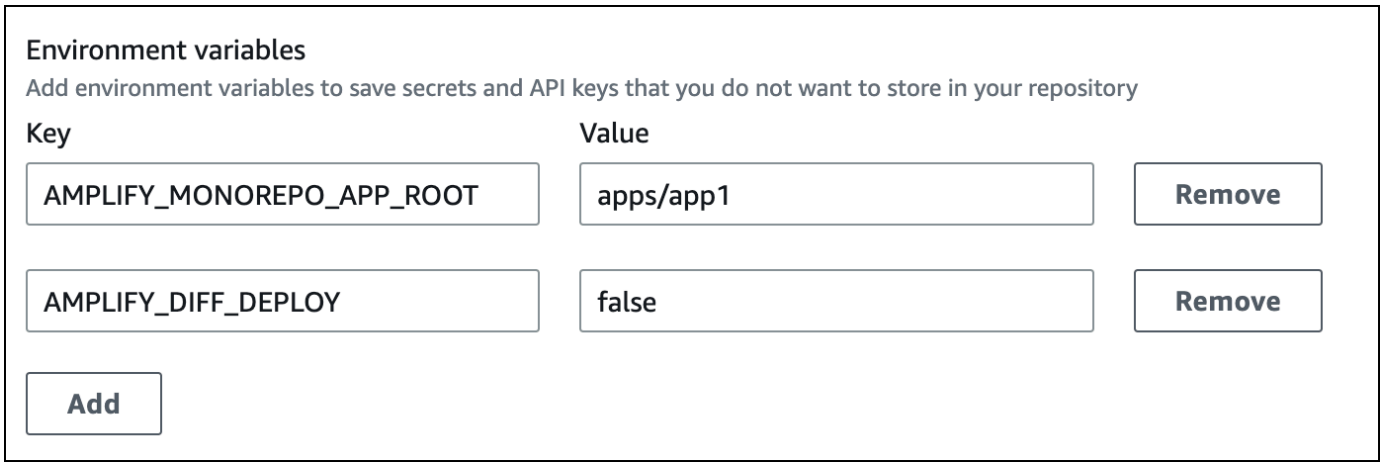
b. 在「分支」中，選擇要使用的分支名稱。

c. 選擇連接一個單據庫？ 選擇一個文件夾。

d. 在 monorepo 中輸入應用程式的路徑，**apps/app1**例如。

e. 選擇下一步。

5. 在「構建設置」 頁面上，您可以使用默認設置或自定義應用程序的構建設置。在下列範例螢幕擷取畫面中，使用您在步驟 4d 中指定的路徑，AMPLIFY_MONOREPO_APP_ROOT將 apps/app1「環境變數」區段中的「Amplify」設定為。



6. 選擇下一步。

7. 在「複查」頁面上，選擇「儲存並部署」。

為現有的應用程式設定應用程式的環境變數

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

若要為現有的應用程式設定環境變數

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。

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 [應用程式設定]，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，執行下列操作：

a. 選擇新增變數。

b. 在「變數」中，輸入金鑰AMPLIFY_MONOREPO_APP_ROOT。

c. 例如，在「值」中輸入應用程式的路徑**apps/app1**。

d. 對於「分支」，預設情況下，「Amplify」會將環境變數套用至所有分支。

6. 選擇儲存。

配置植物園和 PNPM 單回購應用程序

植物園和 pnpm 工作空間壟斷構建工具從文件中獲取配置信息。`.npmrc`當您部署使用這些工具之一創建的 monorepo 應用程序時，您的項目根目錄中必須有一個`.npmrc`文件。

在`.npmrc`檔案中，將`hoisted`用於安裝 Node 套件的連結器設定為。您可以將以下行複製到您的文件中。

```
node-linker=hoisted
```

如需有關*`.npmrc`*檔案和設定的詳細資訊，請參閱 pnpm 說明文件中的 pnpm .npmrc。

Pnpm 不包含在「Amplify」預設組建容器中。對於 pnpm 工作區和 Turborepo 應用程序，您必須添加一個命令以在應用程序的構建設置的`preBuild`階段安裝 pnpm。

下列組建規格摘錄的範例會顯示具有安裝 pnpm 命令的`preBuild`階段。

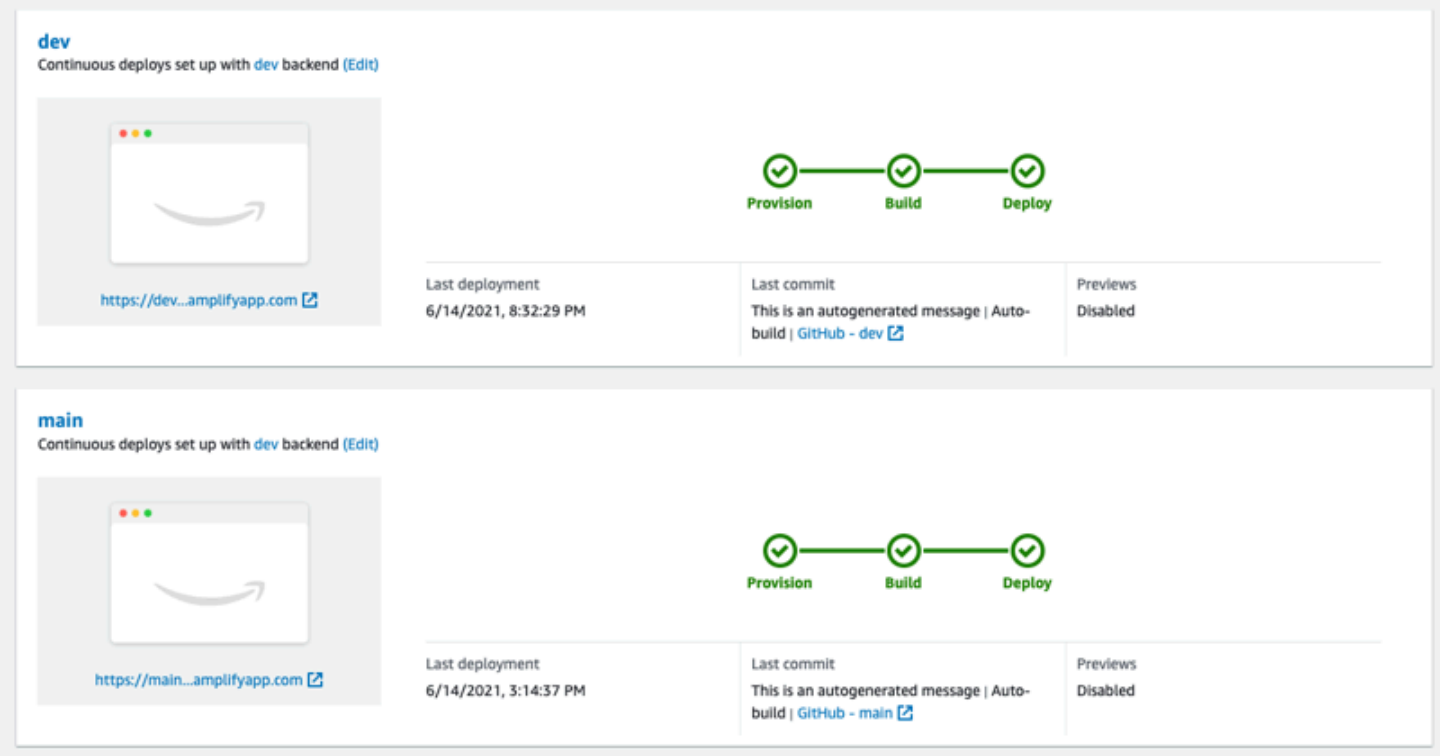
```
version: 1
applications:
  - frontend:
      phases:
        preBuild:
          commands:
            - npm install -g pnpm
```

功能分支部署和團隊工作流程

放大託管旨在與功能分支和工作GitFlow流程配合使用。每次開發人員在其儲存庫中連接新分支時，Amplify 都會利用 Git 分支來建立新的部署。連接第一個分支後，您就能透過新增分支來建立新的功能分支部署，如下所示：

1. 在分支清單頁面上，選擇 Connect branch (連接分支)。
2. 從儲存庫選擇分支。
3. 儲存並部署應用程式。

您的應用程式現在有兩個部署可在 https://main.appid.amplifyapp.com 和 https://dev.appid.amplifyapp.com。這可能與不同team-to-team，但通常主要分支跟踪發布代碼，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。這使 Beta 測試人員能夠在開發分支部署上測試未發布的功能，而不會影響主分支部署上的任何生產終端用戶。



主題

- 使用擴大後端環境的團隊工作流程
- 以模式為基礎的功能分支部署
- 自動生成放大配置的構建時間

- 條件式後端建置
- 跨應用程式使用擴大後端

使用擴大後端環境的團隊工作流程

功能分支部署由前端和可選的後端環境組成。前端是建置並部署到全球內容傳遞網路 (CDN)，而後端則由擴增工作室或擴增 CLI 部署到. AWS 如需有關此部署案例的詳細資訊，請參閱開始使用完整堆疊持續部署。

> **Note**
>
> 您可以輕鬆地在 Amplify 應用程式中重複使用放大後端環境。如需詳細資訊，請參閱跨應用程式使用擴大後端。

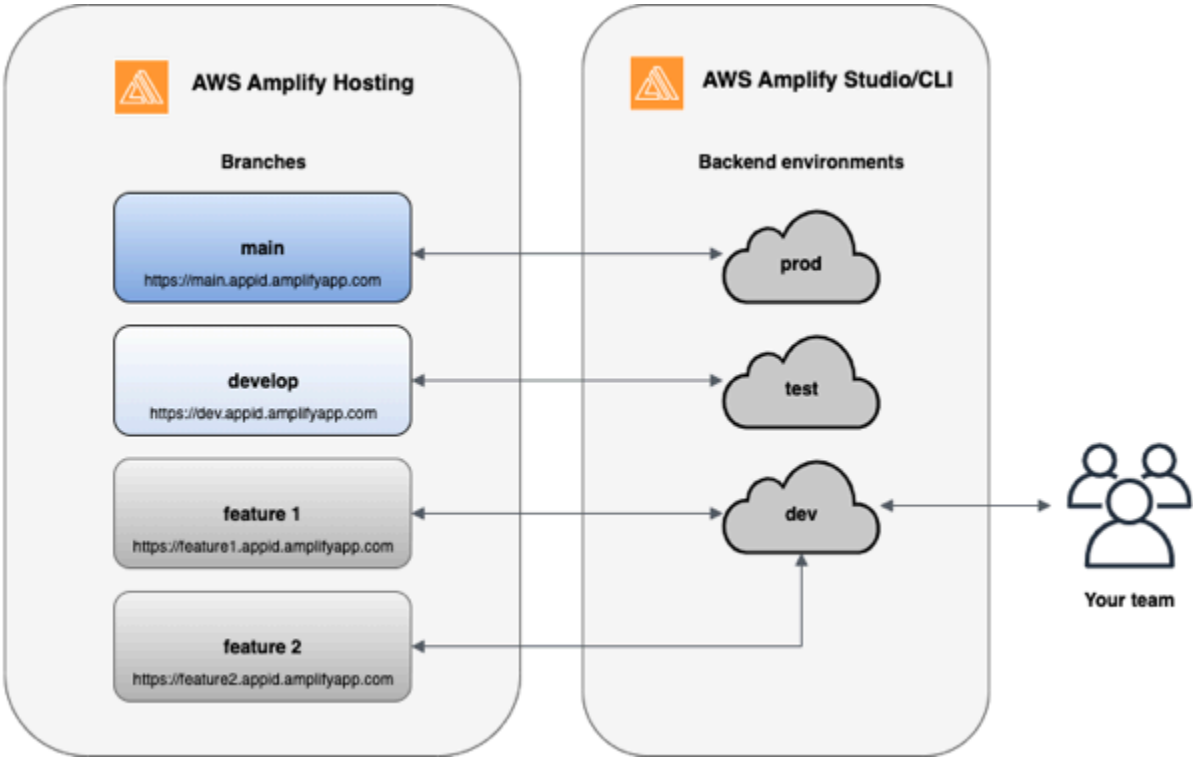
擴大主機可透過功能分支部署持續部署後端資源，例如 GraphQL API 和 Lambda 函數。您可以使用以下分支模型通過 Amplify 託管部署後端和前端。

主題

- 功能分支工作流程
- GitFlow 工作流程
- 每位開發人員的沙盒

功能分支工作流程

- 使用放大工作室或放大 CLI 建立產品、測試和開發後端環境。
- 將 prod 後端映射到主分支。
- 將測試後端映射到開發分支。
- 團隊成員可以使用 dev 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。如果您沒有項目，請使用create-react-app或 Gatsby 等引導工具創建一個項目。

```
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。

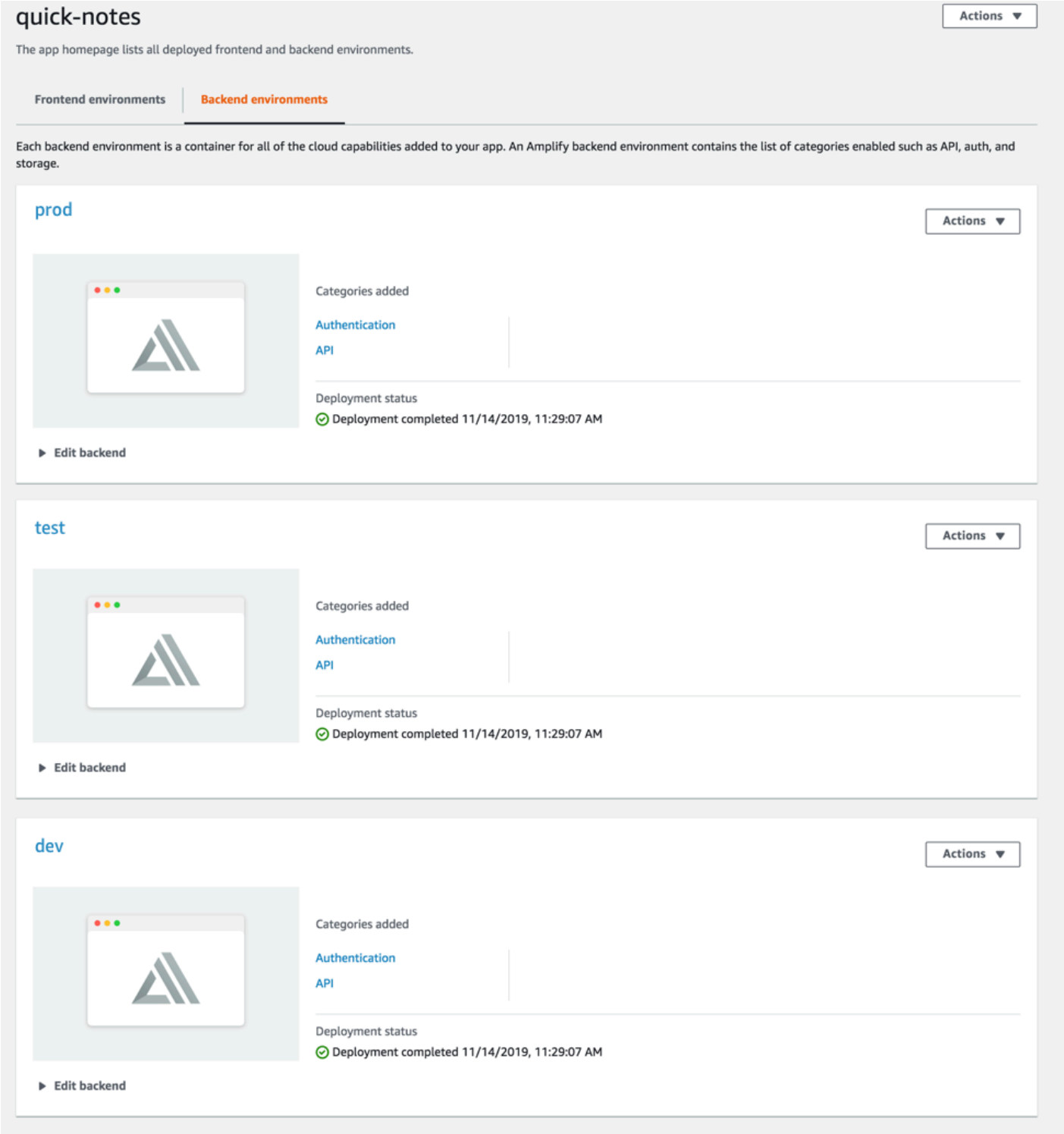
```
amplify env add
 ? Do you want to use an existing environment? (Y/n): n
 ? Enter a name for the environment: test
...
amplify push
```

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

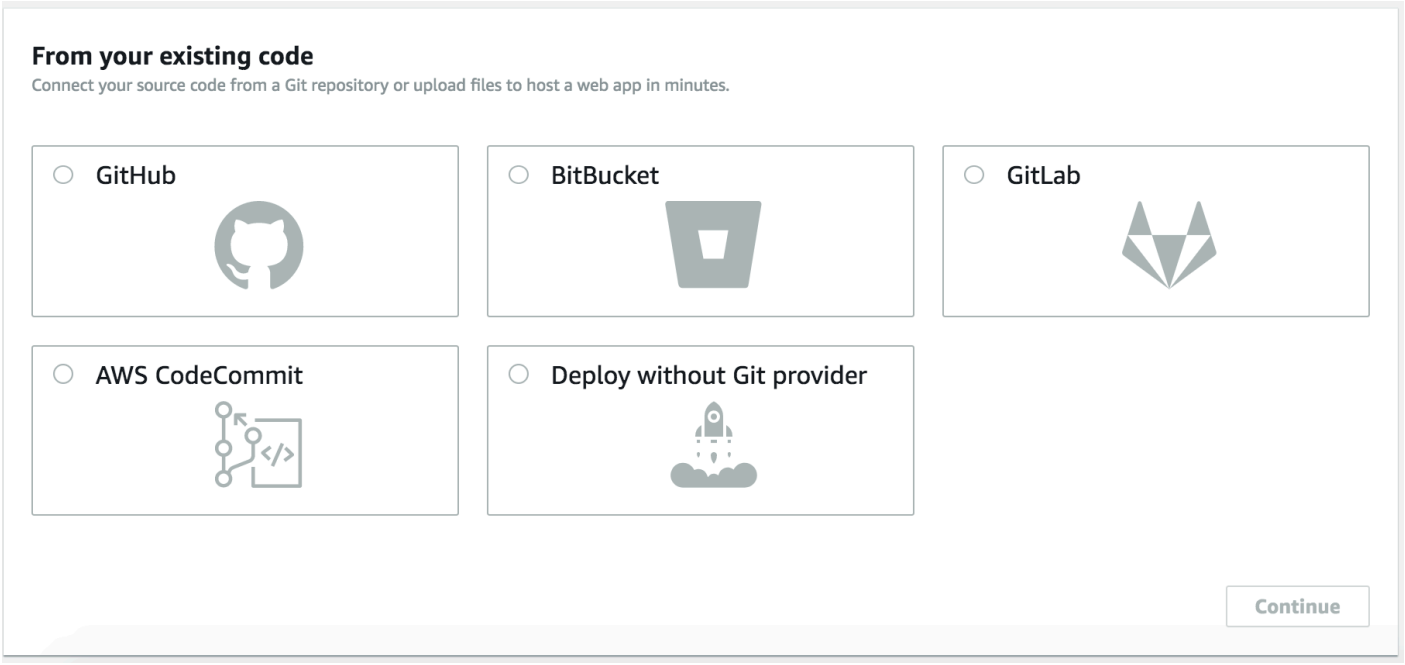
4. 將代碼推送到您選擇的 Git 存儲庫（在此示例中，我們將假設您推送到 main）。

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

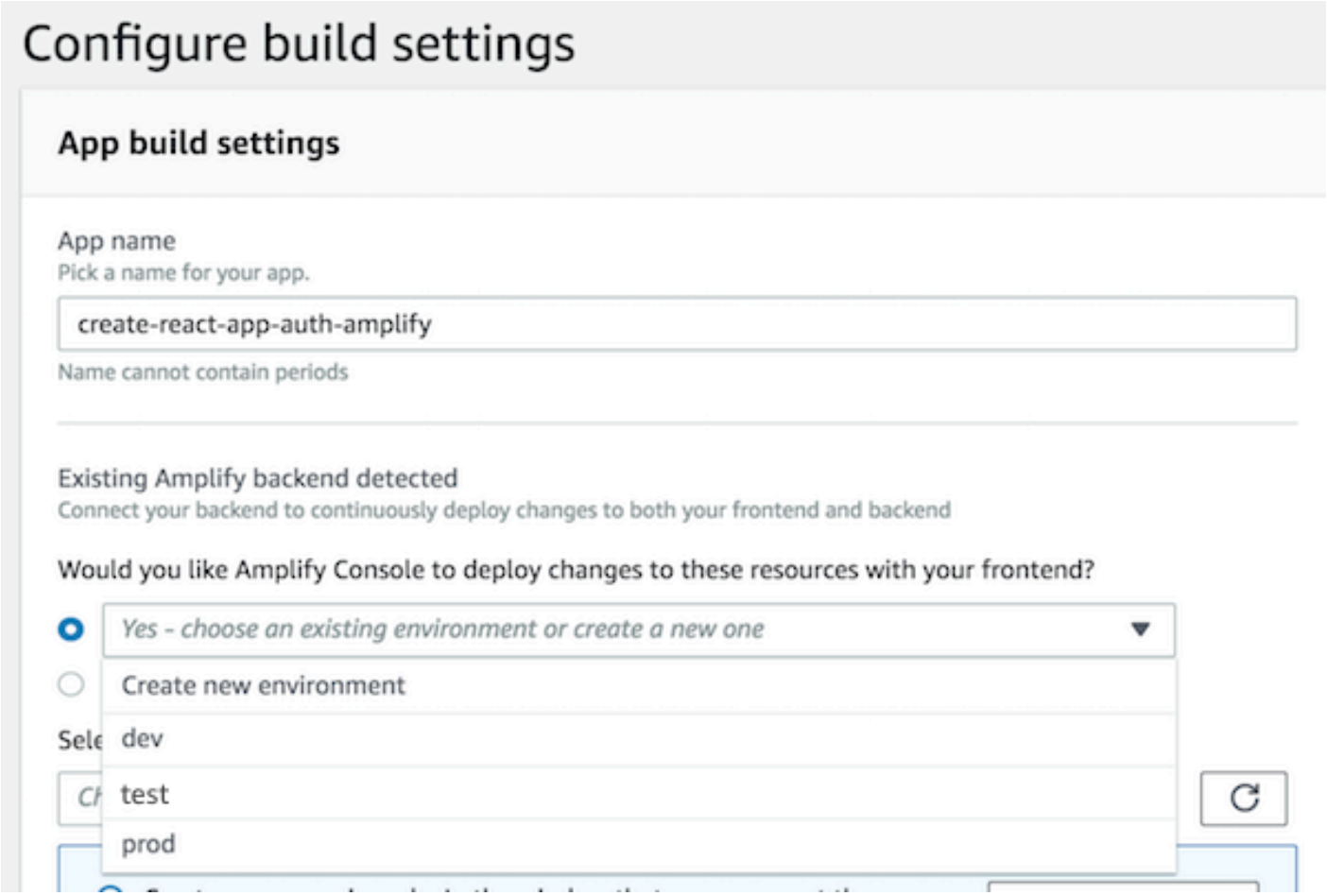
5. 請造訪中的「放大」以AWS Management Console查看您目前的後端環境。從階層連結導覽上一層，以檢視在「後端環境」索引標籤中建立的所有後端環境清單。



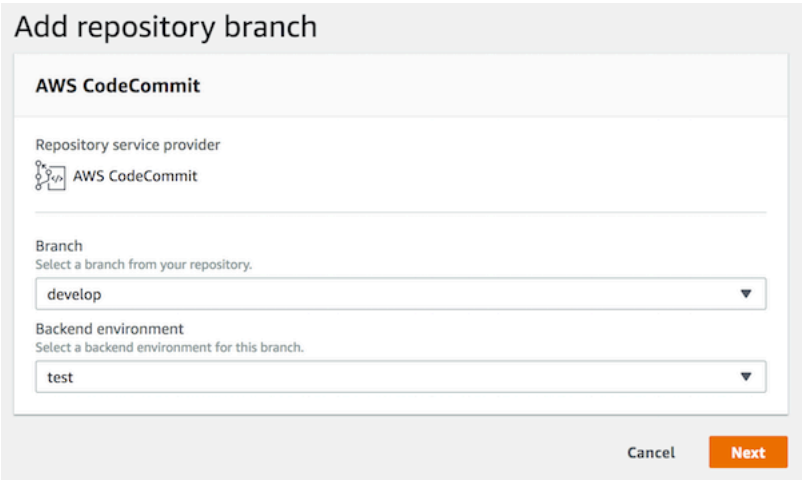
6. 切換到前端環境選項卡，並連接您的存儲庫提供程序和主分支。



7. 在構建設置屏幕中，選擇現有的後端環境以設置與主分支的持續部署。從下拉式清單中選擇 prod，並將服務角色授予「擴大」。選擇 Save and deploy (儲存並部署)。構建完成後，您將在 https://main.appid.amplifyapp.com 獲得可用的主要分支部署。



8. 在 Amplify 中連接開發分支（假設開發和主分支在這一點上是相同的）。選擇「test」後端環境。



9. 現在已設定「放大」。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.要預覽更改的外觀，請轉到 Amplify 控制台並連接您的功能分支。注意：如果您已經在系統上AWS CLI安裝了（而不是 Amplify CLI），則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.這將啟動一個構建，該構建將在 https://dev.appid.amplifyapp.com 上使用分支部署進行 Amplify 中更新後端以及前端。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 中刪除您的功能分支，並從雲中刪除後端環境（您始終可以通過運行「放大 env 結帳產品」並運行「放大 env 添加」來啟動一個新的環境）。

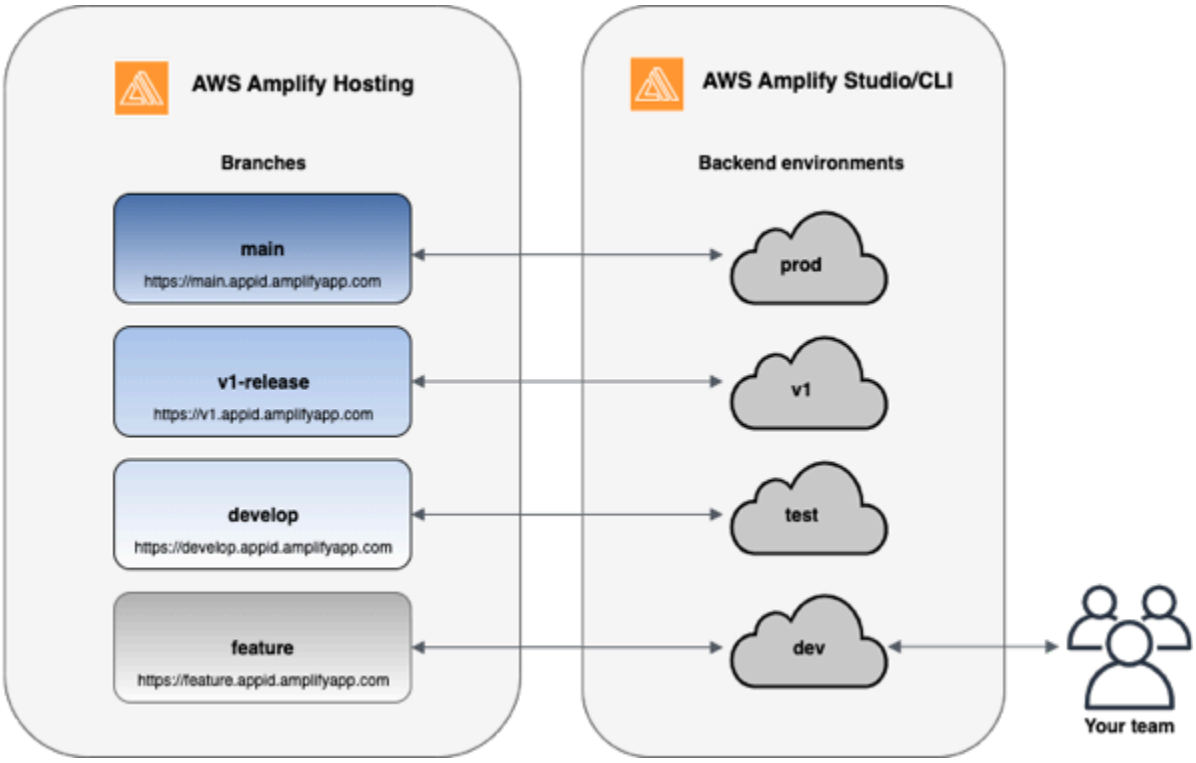
```
git push origin --delete newinternet
aws amplify delete-branch --app-id <appid> --branch-name <branchname>
amplify env remove dev
```

GitFlow 工作流程

GitFlow使用兩個分支來記錄專案的歷程。主分支僅跟踪發布代碼，並且開發分支用作新功能的集成分支。GitFlow通過將新的開發與已完成的工作隔離來簡化並行開發。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

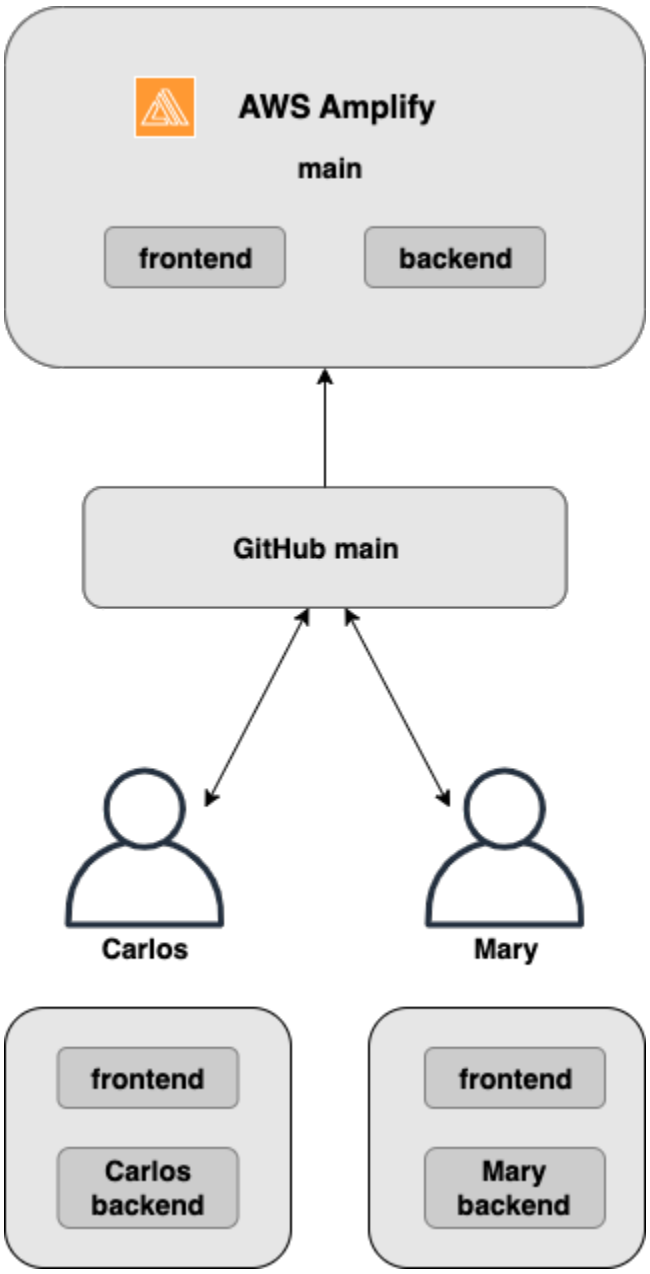
合併回整合「開發」分支。對主分支的唯一提交是來自發布分支和修補程序分支的合併（以修復緊急錯誤）。

下圖顯示建議使用的設定GitFlow。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。這使開發人員可以彼此隔離工作，而不會覆蓋其他團隊成員的更改。
- Amplify 中的每個分支都有自己的後端。這可確保 Amplify 會使用 Git 儲存庫做為單一事實來源來部署變更，而不是仰賴團隊的開發人員從本機電腦手動將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. 為您的項目初始化瑪麗後端環境。如果您沒有項目，請使用create-react-app或 Gatsby 等引導工具創建一個項目。

```
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. 將代碼推送到您選擇的 Git 存儲庫（在此示例中，我們將假設您推送到 main.

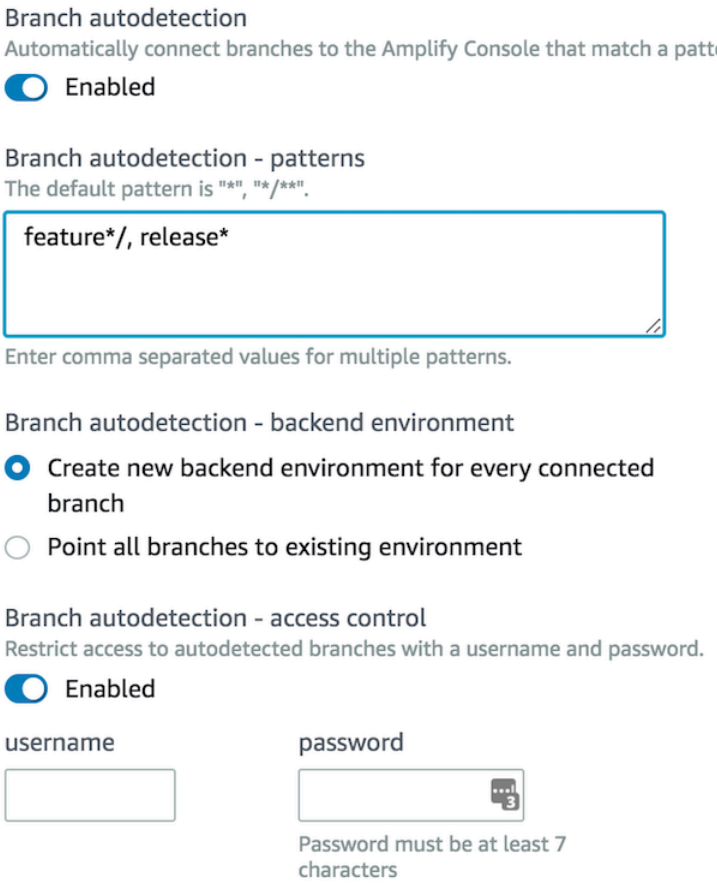
```
git commit -am 'Added mary sandbox'
git push origin main
```

4. 連接您的回購 > 主要放大。
5. 擴大控制台將檢測由放大 CLI 創建的後端環境。從下拉式清單中選擇 [建立新環境]，並將服務角色授與 Amplify。選擇 Save and deploy (儲存並部署)。構建完成後，您將在 https://main.appid.amplifyapp.com 獲得一個主要分支部署，其中包含鏈接到分支的新後端環境。
6. 在 Amplify 中連接開發分支（假設開發和主分支在這一點上是相同的），然後選擇創建新環境。建置完成後，您即可在 https://develop.appid.amplifyapp.com 取得開發分支部署，以及連接至該分支的新後端環境。

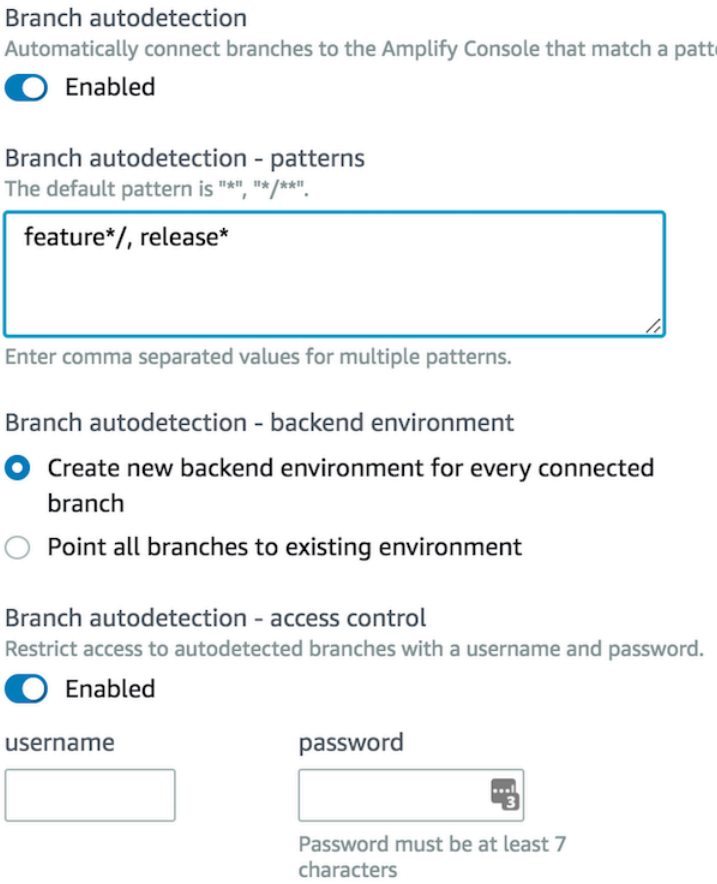
以模式為基礎的功能分支部署

以模式為基礎的分支部署可讓您自動部署符合特定模式的分支以擴增。在其發行版本中使用功能分支或GitFlow工作流程的產品團隊現在可以定義「釋放 **」之類的模式，以自動將以「發行」開頭的 Git 分支部署到可共享的 URL。此部落格文章說明搭配不同團隊工作流程使用本功能的方式。

1. 依序選擇 App settings > General > Edit (應用程式設定 > 一般 > 編輯)。
2. 將分支自動偵測開關切換為 Enabled (啟用)。



1. 定義模式以自動部署分支。
   - *****-部署存儲庫中的所有分支。
   - **`release*`**'— 部署以「釋放」一詞開頭的所有分支。
   - **`release*/`**— 部署符合「釋放/」模式的所有分支。
   - 在逗號分隔的清單中指定多個模式。例如：`release*, feature*`。
2. 將 Branch autodetection - access control (分支自動偵測 - 存取控制) 設定為 Enabled (啟用)，即可為自動建立的所有分支設定自動密碼保護功能。
3. 若應用程式是透過 Amplify 後端建置，則您可選擇建立新環境，或將所有分支指向現有後端。



連接到自定義域的應用程序基於模式的功能分支部署

對於連接到 Amazon Route 53 自訂網域的應用程式，您可以使用以模式為基礎的功能分支部署。

- 如需設定以模式為基礎的功能分支部署的指示，請參閱 為 Amazon 路線 53 自定義域設置自動子域
- 如需將 Amplify 應用程式連線至 Route 53 中管理的自訂網域的指示，請參閱 添加由 Amazon 路線 53 管理的自定義域
- 有關使用路線 53 的更多信息，請參閱什麼是亞馬遜路線 53。

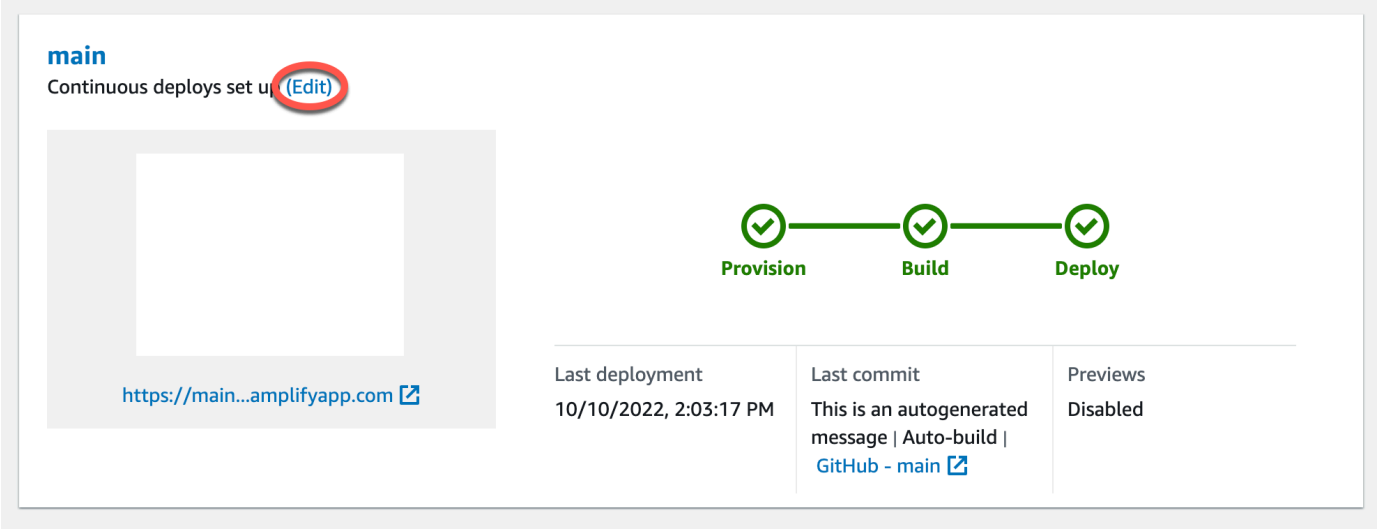
自動生成放大配置的構建時間

擴增支援自動建置時間產生擴增設定檔案。`aws-exports.js`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，並確保在建置階段不會對後端進行更新。

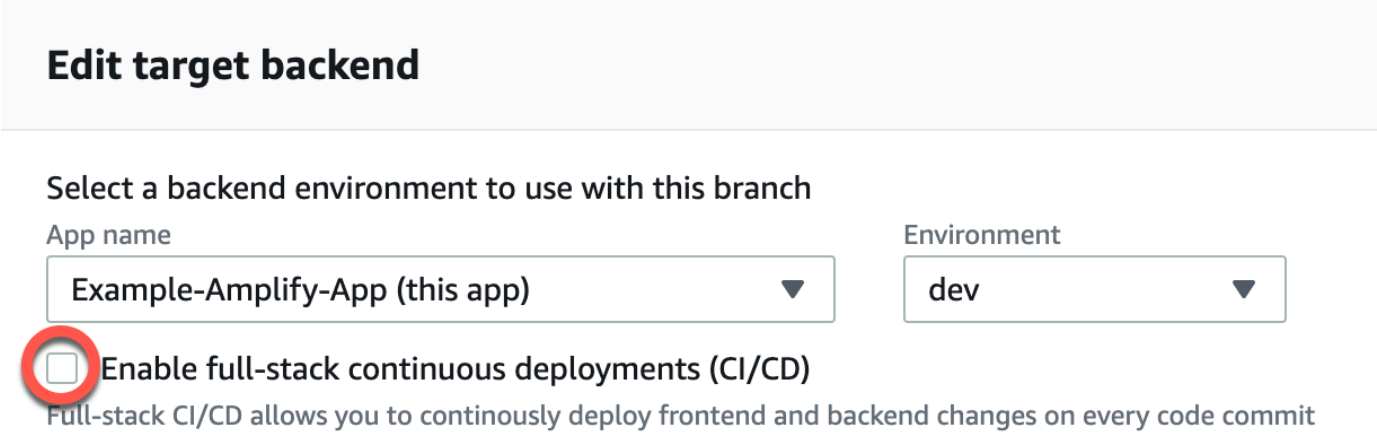
若要在建置時自動產**`aws-exports.js`**生

1. 登入AWS Management Console並開啟擴大控制台。

2. 選擇要編輯的應用程式。
3. 選擇託管環境選項卡。
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「啟用全堆疊連續部署 (CI/CD)」，以關閉此後端的全堆疊 CI/CD。



6. 選取現有的服務角色，將變更應用程式後端所需的權限授與 Amplify。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

Amplify 支援在應用程式中的所有分支上建置條件式後端。若要設定條件式後端組建，請將AMPLIFY_DIFF_BACKEND環境變數設定為true。啟用條件式後端組建將有助於加速僅對前端進行變更的建置。

當您啟用基於差異的後端構建時，在每個構建開始時，Amplify 會嘗試對存儲庫中的amplify文件夾運行差異。如果 Amplify 沒有發現任何差異，它會跳過後端構建步驟，並且不會更新後端資源。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，請參閱啟用或禁用基於差異的後端構建。

如果您當前在後端階段的構建設置中指定了自定義命令，則條件後端構建將無法正常工作。如果要運行這些自定義命令，則必須將它們移動到應用程序amplify.yml文件中構建設置的前端階段。如需更新amplify.yml檔案的更多資訊，請參閱建立規格指令和設定。

跨應用程式使用擴大後端

Amplify 可讓您輕鬆地在指定區域中的所有應用程式中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。

創建新的應用程序時重用後端

若要在建立新的 Amplify 應用程式時重複使用後端

1. 登入AWS Management Console並開啟擴大控制台。
2. 若要建立用於此範例的新後端，請執行下列動作：
   a. 在功能窗格中，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. 要將前端連接到新後端，請選擇託管環境選項卡。
4. 選擇您的 git 提供程序，然後選擇連接分支。
5. 在 [新增儲存庫分支] 頁面上，針對 [最近更新的儲存庫] 選擇您的存放庫名稱 對於「分支」，請從存放庫中選取要連線的分支。
6. 在 [組建] 設定中，頁面執行下列動作：

a. 針對應用程式名稱，選取要用於新增後端環境的應用程式。您可以選擇當前的應用程序或當前區域中的任何其他應用程序。

b. 在「環境」中，選取要新增的後端環境名稱。您可以使用既有環境或建立新環境。

c. 根據預設，全堆疊 CI/CD 處於關閉狀態。關閉完整堆疊 CI/CD 會導致應用程式在僅提取模式下執行。在建置階段，Amplify 只會自動產生`aws-exports.js`檔案，而不會修改後端環境。

d. 選取現有的服務角色，將變更應用程式後端所需的權限授與 Amplify。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)。

將分支連接到現有應用程序時重複使用後端

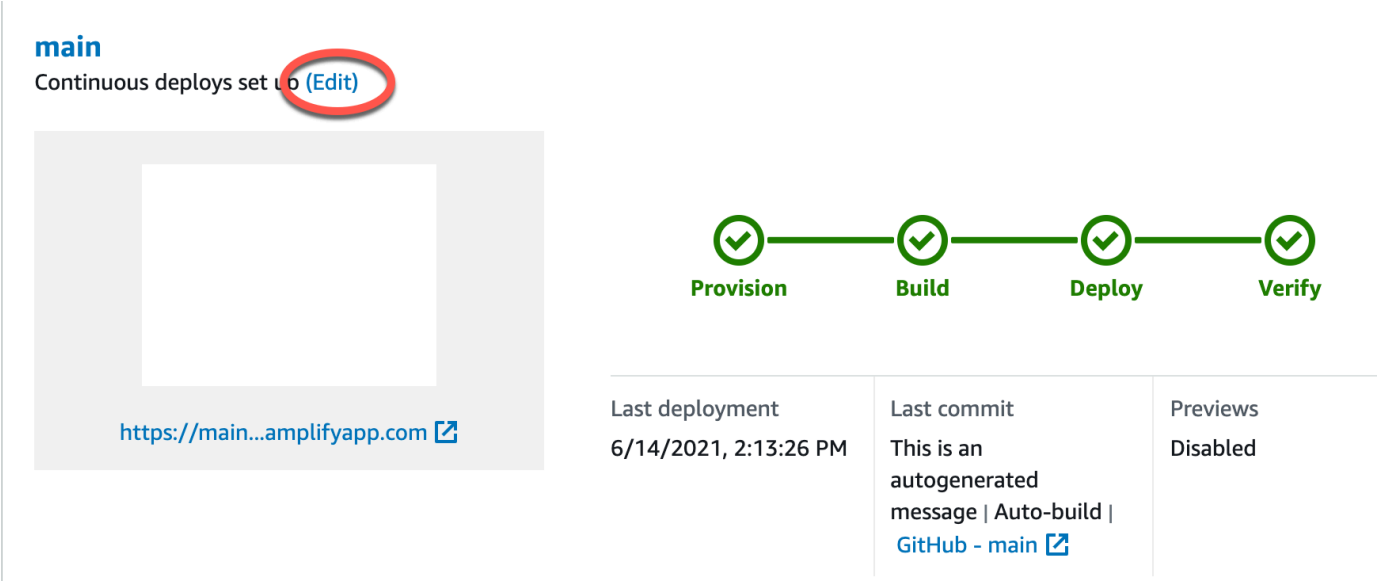
將分支連接到現有的 Amplify 應用程序時重複使用後端

1. 登入AWS Management Console並開啟擴大控制台。
2. 選擇要連接新分支的應用程序。
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「連線分支」。
5. 在「新增儲存區域分支」頁面上，對於「分支」，從儲存庫中選取要連線的分支。
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。您可以選擇當前的應用程序或當前區域中的任何其他應用程序。
7. 在「環境」中，選取要新增的後端環境名稱。您可以使用既有環境或建立新環境。
8. 如果您需要設定服務角色，以授與 Amplify 變更應用程式後端所需的權限，主控台會提示您執行此工作。如需建立服務角色的詳細資訊，請參閱新增服務角色。
9. 根據預設，全堆疊 CI/CD 處於關閉狀態。關閉完整堆疊 CI/CD 會導致應用程式在僅提取模式下執行。在建置階段，Amplify 只會自動產生`aws-exports.js`檔案，而不會修改後端環境。
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 Management Console並開啟擴大控制台。
2. 選擇要編輯後端的應用程式。
3. 選擇託管環境選項卡。
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。您可以選擇當前的應用程序或當前區域中的任何其他應用程序。
6. 在後端環境中，選取要新增的後端環境名稱。
7. 依預設，會啟用完整堆疊 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 會導致應用程式在僅提取模式下執行。在建置階段，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。

手動部署

手動部署可讓您使用 Amplify 主機發佈您的網路應用程式，而無需連線 Git 提供者。您可以從桌面拖放文件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用 URL。

對於 Amazon S3，您也可以設定AWS Lambda觸發器，以便在每次上傳新資產時更新您的網站。如需有關設定此案例的詳細資訊，請參閱將存放在 Amazon S3、Dropbox 或桌面上的檔案部落格文章部落格文章。AWS Amplify

Amplify 主機不支援手動部署伺服器端轉譯 (SSR) 應用程式。如需詳細資訊，請參閱使用 Amplify 主機部署伺服器端轉譯應用程。

拖放手動部署

使用拖放方式手動部署應用程式

1. 登入AWS Management Console並開啟 Amplify 大控制台。
2. 如何進入主機您的 Web 應用程序頁面取決於您是從 Amplify 首頁還是所有應用程序頁面開始。
   - 從「Amplify」首頁
     a. 選擇 Get started (開始使用)。
     b. 在 [傳送] 區段中，選擇 [開始使用]。
   - 從所有應用程式頁面
     - 在右上角，選擇 [新增應用程式]、[主機 Web 應用程式]
3. 在 [主機您的 Web 應用程式] 頁面上，選擇 [部署不含 Git 提供者]。然後選擇 Continue (繼續)。
4. 在 [啟動手動部署] 區段中，對於 [應用程式名稱]，輸入應用程式的名稱。
5. 對於「環境名稱」，請輸入有意義的環境名稱，例如**`development`**或**`production`**。
6. 在「方法」中，選擇「拖放」。
7. 您可以將檔案從桌面拖放到放置區域，或使用「選擇檔案」從電腦中選取檔案。您拖放或選取的檔案可以是包含網站根目錄的資料夾或 zip 檔案。
8. 選擇 Save and deploy (儲存並部署)。

Amazon S3 或 URL 手動部署

從 Amazon S3 手動部署應用程式或公有 URL

1. 登入AWS Management Console並開啟 Amplify 大控制台。
2. 在頁面頂端，選擇開始使用。
3. 在 [傳送] 區段中，選擇 [開始使用]。
4. 在 [主機您的 Web 應用程式] 頁面上，選擇 [部署不含 Git 提供者]。然後選擇 Continue (繼續)。
5. 在 [啟動手動部署] 區段中，對於 [應用程式名稱]，輸入應用程式的名稱。
6. 對於「環境名稱」，請輸入有意義的環境名稱，例如**`development`**或**`production`**。
7. 在「方法」中，選擇 Amazon S3 或任何 URL。
8. 上傳檔案的程序取決於上傳方法。
   - Amazon S3
     a. 對於儲存貯體，從清單中選 Amazon S3 的名稱。您必須為您選取的儲存貯體啟用存取控制清單 (ACL)。如需詳細資訊，請參閱診斷 Amazon S3 儲存貯體存取。
     b. 對於 Zip 檔案，請選取要部署的 zip 檔案名稱。
   - 任何網址
     - 對於資源 URL，請輸入要部署之壓縮檔案的 URL。
9. 選擇 Save and deploy (儲存並部署)。

> **Note**
> 當您創建 zip 文件夾時，請確保壓縮構建輸出的內容，而不是頂級文件夾。例如，如果您的構建輸出生成一個名為「build」或「public」的文件夾，請首先導航到該文件夾，選擇所有內容，然後從那裡壓縮它。如果不這樣做，您將看到「拒絕訪問」錯誤，因為網站的根目錄將無法正確初始化。

診斷 Amazon S3 儲存貯體存取

建立 Amazon S3 儲存貯體時，您可以用來控制 Amazon S3 存貯體的存取控制清單 (ACL)。若要手動部署應用程式以從 Amazon S3 儲存貯體 Amplify，必須在儲存貯體上啟用 ACL。

如果您在從 Amazon S3 儲存貯體部署時收到`AccessControlList`錯誤訊息，表示儲存貯體是在停用 ACL 的情況下建立的，您必須在 Amazon S3 主控台中啟用它們。如需指示，請參閱 Amazon 簡單儲存服務使用者指南中的對現有儲存貯體設定物件擁有權。

部署到放大按鈕

「部署到擴大主機」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「部署到擴大主機」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、部落格貼文或任何其他呈現 HTML 的標記頁面。此按鈕包含下列兩個元件：

1. 一個 SVG 圖像位於網址 `https://oneclick.amplifyapp.com/button.svg`
2. 擴大控制台 URL，其中包含指向您的GitHub存儲庫的鏈接。您可以複製儲存庫的 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`。放大託管將在您的存儲庫中部署默認分支。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降價檔案，例如 GitHub Readme.md。以存放庫的 URL 取`https://github.com/username/repository`代。

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

使用下面的例子將按鈕添加到任何 HTML 文檔。以存放庫的 URL 取`https://github.com/username/repository`代。

```
<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 GitHub 庫的存取

Amplify 現在使用「 GitHub 應用程式」功能來授權「Amplify」 GitHub 儲存庫的唯讀存取權。使用 Amplify GitHub 應用程式時，權限會進行更微調，讓您僅授與您指定的存放庫的 Amplify 存取權。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。

當您連接存儲在存儲 GitHub 庫中的新應用程序時，默認情況下 Amplify 使用該 GitHub 應用程序訪問存儲庫。不過，您先前從存放 GitHub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，但我們強烈建議您將其移轉為使用新的 Amplify GitHub 應用程式。

當您使用 Amplify 主控台部署新應用程式或移轉現有應用程式時，系統會自動將您導向至 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，請開啟網頁瀏覽器，然後依地區瀏覽至該應用程式。使用該格式`https://github.com/apps/aws-amplify-`*`REGION`*，將 REG*ION* 替換為您將部署 Amplify 應用程序的區域。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://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
- 將現有的OAuth應用程式移轉至 Amplify GitHub 應用程式
- 為AWS CloudFormation、CLI 和 SDK 部署設定 Amplify GitHub 應用程式
- 使用 Amplify GitHub 應用程式設定網頁預覽

為新部署安裝和授權 Amplify GitHub 應用程式

當您從存放 GitHub 庫中的現有代碼部署新的應用程序以 Amplify 時，請使用以下說明安裝和授權該 GitHub 應用程序。

安裝和授權 Amplify GitHub 應用程式

1. 登入AWS Management Console並開啟 Amplify 大控制台。
2. 從 [所有應用程式] 頁面中，選擇 [新增應用程式]，然後選擇 [主機
3. 在 [開始使用 Amplify 主機] 頁面上，選擇 GitHub，然後選擇 [繼續]。
4. 如果這是第一次連接 GitHub 存儲庫，則在 GitHub .com 上的瀏覽器中會打開一個新頁面，請求授權您AWS Amplify的 GitHub帳戶。選擇 Authorize (授權)。
5. 接下來，您必須在您的 GitHub 帳戶中安裝 Amplify GitHub 應用程序。在 Github.com 上會開啟一個頁面，要求授權AWS Amplify在您的 GitHub帳戶中安裝和授權。

6. 選擇您要安裝 Amplify GitHub 應用程式的 GitHub 帳戶。
7. 執行下列任意一項：
   - 若要將安裝套用至所有儲存庫，請選擇「所有儲存庫」。
   - 若要將安裝限制在您選取的特定儲存庫，請選擇僅選取儲存區域。請務必在您選取的存放庫中包含您要移轉之應用程式的存放庫。
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 [新增存放庫分支] 頁面。
10. 在「最近更新的儲存庫」清單中，選取要連線的存放庫名稱。
11. 在「分支」清單中，選取要連線的存放庫分支的名稱。
12. 選擇 Next (下一步)。
13. 在 [設定組建設定] 頁面上，選擇 [下一步]。
14. 在「複查」頁面上，選擇「儲存並部署」。

將現有的OAuth應用程式移轉至 Amplify GitHub 應用程式

您先前從儲存庫連線的現有 Amplify 應用程式會使用 OAuth 進行存放 GitHub 庫存取。強烈建議您遷移這些應用程式，以便使用 Amplify GitHub 應用程式。

使用以下說明遷移應用程序，並在您的 GitHub 帳戶中刪除其相應的 OAuth Webhook。請注意，移轉程序視是否已安裝 Amplify GitHub 應用程式而有所不同。遷移第一個應用程序並安裝並授權該 GitHub 應用程序後，您只需要更新後續應用程序遷移的存儲庫權限。

將應用程序從 OAuth 遷移到 GitHub 應用程序

1. 登入AWS Management Console並開啟 Amplify 大控制台。
2. 選擇您要遷移的應用程式。
3. 在應用程序的信息頁面上，找到藍色的遷移到我們的 GitHub應用程序消息，然後選擇開始遷移。
4. 在 [安裝並授權 GitHub 應用程式] 頁面上，選擇 [設定 GitHub 應用程式
5. 在 GitHub .com 上的瀏覽器中會開啟一個新頁面，要求授權您AWS Amplify的 GitHub 帳戶。選擇 Authorize (授權)。
6. 選擇您要安裝 Amplify GitHub 應用程式的 GitHub 帳戶。
7. 執行下列任意一項：
   - 若要將安裝套用至所有儲存庫，請選擇「所有儲存庫」。

- 若要將安裝限制在您選取的特定儲存庫，請選擇僅選取儲存區域。確保在您選擇的存儲庫中包含要遷移的應用程序的存儲庫。

8. 選擇「安裝和授權」。
9. 系統會將您重新導向至 Amplify 主控台中 GitHub 應用程式的「安裝和授權應用程式」頁面。如果 GitHub 授權成功，則會看到成功訊息。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，然後完成移轉。

為AWS CloudFormation、CLI 和 SDK 部署設定 Amplify GitHub 應用程式

您先前從儲存庫連線的現有 Amplify 應用程式會使用 OAuth 進行存放 GitHub 庫存取。這可能包括您使用 Amplify mand Line Interface (CLI) 或開發套件來部署的應用程式。AWS CloudFormation強烈建議您遷移這些應用程式，以便使用新的 Amplify GitHub 應用程式。移轉必須在中的 Amplify 主控台中執行AWS Management Console。如需相關指示，請參閱將現有的OAuth應用程式移轉至 Amplify GitHub 應用程式。

您可以使用AWS CloudFormation Amplify CLI 和 SDK 來部署新的 Amplify 應用程式，該應用程式使用應用 GitHub 程式進行存放庫存取。此程序需要您先在 GitHub 帳戶中安裝 Amplify GitHub 應用程式。接下來，您將需要在您的 GitHub 帳戶中生成個人訪問令牌。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

1. 開啟網頁瀏覽器，然後導覽至您要部署 GitHub應用程式的AWS區域中 Amplify 應用程式的安裝位置。

   *####https://github.com/apps/aws-amplify-REGION/installations/new#### ###### REGION#*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. 選擇您要安裝 Amplify GitHub 應用程式的 GitHub 帳戶。
3. 執行下列任意一項：
   - 若要將安裝套用至所有儲存庫，請選擇「所有儲存庫」。
   - 若要將安裝限制在您選取的特定儲存庫，請選擇僅選取儲存區域。請務必在您選取的存放庫中包含您要移轉之應用程式的存放庫。

4. 選擇 Install (安裝)。

在您的 GitHub 帳戶中生成個人訪問令牌

1. 登入您的 GitHub 帳戶。
2. 在右上角，找到您的個人資料照片，然後從菜單中選擇「設置」。
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 [個人存取權杖]。
5. 在 [個人存取權杖] 頁面上，選擇 [產生新權杖]。
6. 在 [新增個人存取權杖] 頁面上，針對備註輸入權杖的描述性名稱。
7. 在 [選取範圍] 區段中，選取 [管理員:repo_hook]。
8. 選擇 Generate token (產生字符)。
9. 複製並儲存個人存取權杖。當您部署具有 CLI 或 SDK 的 Amplify 應用程式時AWS 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權杖之後，您可以使用 Amplify CLI 或 SDK 部署新的應用程式。AWS CloudFormation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 CreateAppA mplify API 參考和AWS CloudFormation使用者指南AWS::Amplify::App中的。

下列 CLI 命令會部署使用應用程式進行存放庫存取權的新 Amplify GitHub 應用程式。用您自己的信息替換 *myapp-using-githubapp* ，*https://github.com/Myaccount/react-app* 和*####*。

```
aws amplify create-app --name myapp-using-githubapp --repository https://github.com/
Myaccount/react-app --access-token MY_TOKEN
```

使用 Amplify GitHub 應用程式設定網頁預覽

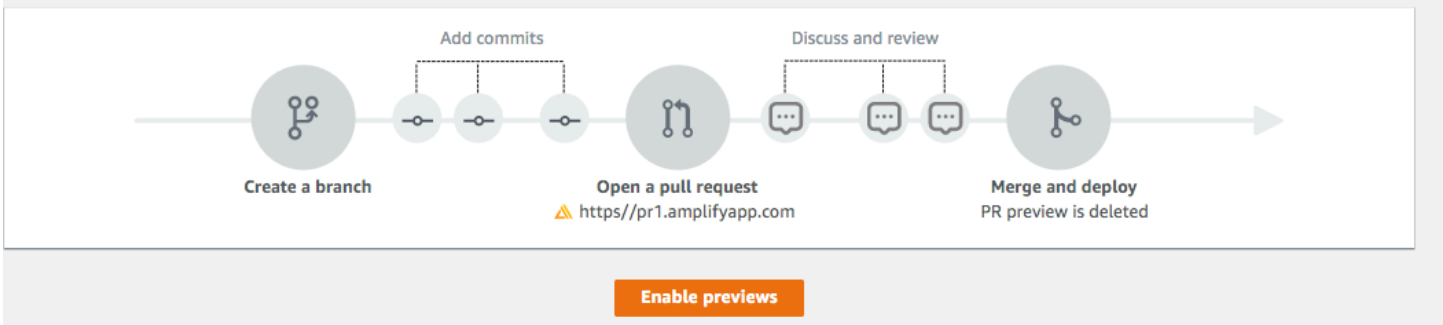
Web 預覽會將對 GitHub 儲存庫所做的每個提取要求 (PR) 部署到唯一的預覽 URL。預覽版現在可以使用 Amplify GitHub 應用程式來存取您的 GitHub 軟體庫。如需安裝和授權 GitHub 應用程式進行網頁預覽的說明，請參閱啟用網頁預覽。

提取要求的網頁預覽

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，預覽來自提取請求 (PR) 的變更的方法。提取要求可讓您告訴其他人您已推送至儲存庫中分支的變更。提取請求開啟後，您可以與協同合作者討論並檢閱潛在變更，並在將變更合併到基底分支之前新增後續提交。

> **Note**
>
> 目前，Amplify 的預覽分支支援 GitLab BitBucket、和AWS CodeCommit沒有完整的功能同位檢查 GitHub。`AWS_PULL_REQUEST_ID`環境變數只有在用 GitHub 作儲存庫提供者時才可用。

Web 預覽會將對儲存庫發出的每個提取請求部署到唯一的預覽 URL，該 URL 與您的主網站使用的 URL 完全不同。對於使用 Amplify CLI 或 Amplify Studio 佈建的後端環境的應用程式，每個提取要求 (僅限私人 Git 儲存庫) 都會啟動一個暫時的後端，該後端會在 PR 關閉時刪除。



> **Important**
>
> 基於安全性考量，您可以在所有具有私人儲存庫的應用程式上啟用網頁預覽，但不能在所有具有公開重新定義的應用程式上啟用 Web 如果您的 Git 儲存庫是公開的，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`主機平台的應用程式都需要 IAM 服務角色。因此，如果這些類型的應用程式存放庫是公開的，您就無法啟用 Web 預覽。
>
> Amplify 會強制執行此限制，以防止第三方提交可使用您應用程式的 IAM 角色許可執行的任意程式碼。

啟用網頁預覽

對於存儲在存儲 GitHub 庫中的應用程序，預覽使用 Amplify GitHub 應用程序進行存儲庫訪問。如果您要在先前使用 OAuth 存放 GitHub 庫部署的現有 Amplify 應用程式上啟用網頁預覽，則必須先移轉應用程式以使用 Amplify GitHub 應用程式。如需移轉指示，請參閱將現有的OAuth應用程式移轉至 Amplify GitHub 應用程式。

啟用提取要求的網頁預覽

1. 選擇 [應用程式設定]、[預覽]，然後選擇 [啟用

> **Note**
>
> 只有當應用程序設置為持續部署並連接到 git 存儲庫時，預覽才能在應用程序設置菜單中看到。如需有關此部署類型的指示，請參閱開始使用現有程式碼。

2. 僅對於 GitHub 存儲庫，請執行以下操作以在您的帳戶中安裝和授權 Amplify GitHub 應用程序：

   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. 選取您要在其中設定 Amplify GitHub 應用程式的 GitHub 帳戶。

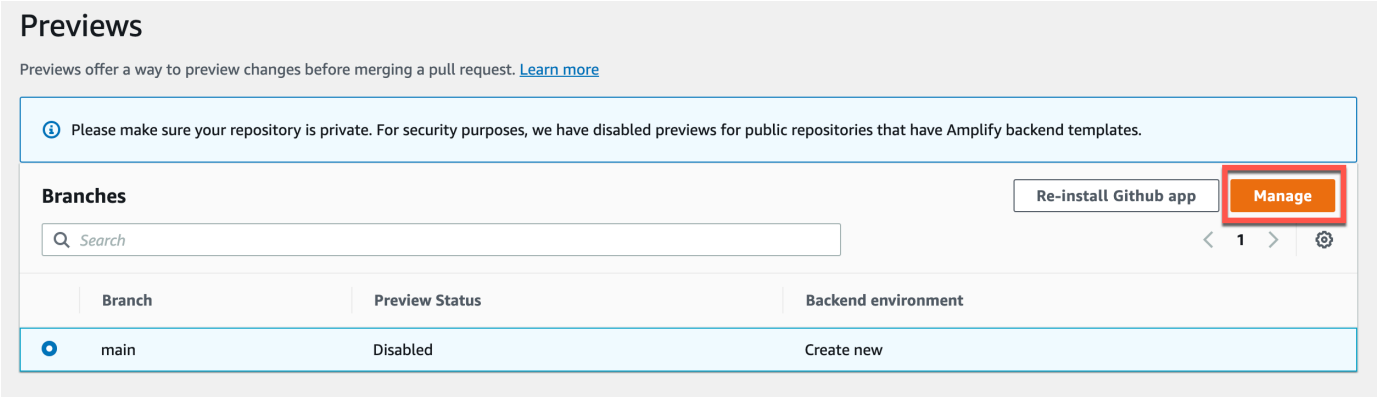
   c. Github.com 上會開啟一個頁面，以設定您帳戶的儲存庫權限。

   d. 執行下列任意一項：

      - 若要將安裝套用至所有儲存庫，請選擇「所有儲存庫」。
      - 若要將安裝限制在您選取的特定儲存庫，請選擇僅選取儲存區域。確保在您選擇的存儲庫中包含要啟用 Web 預覽的應用程序的存儲庫。

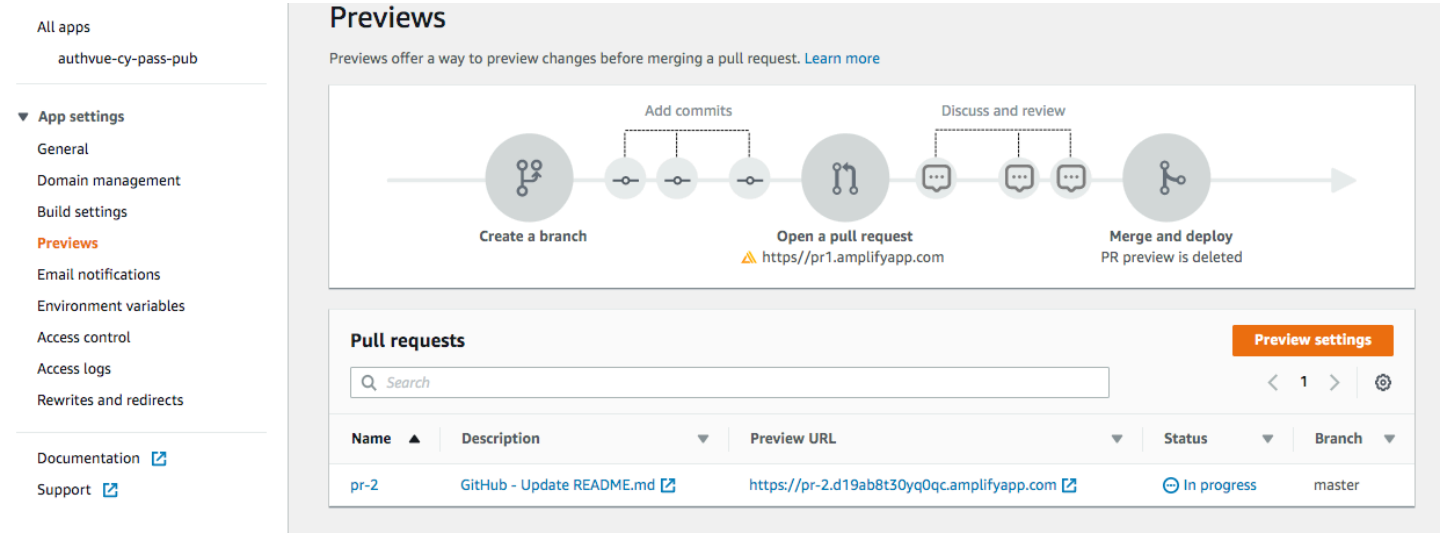
   e. 選擇 Save (儲存)

3. 啟用存放庫的預覽後，請返回 Amplify 控制台以啟用特定分支的預覽。在「預覽」頁面上，從清單中選取分支，然後選擇「管理」。

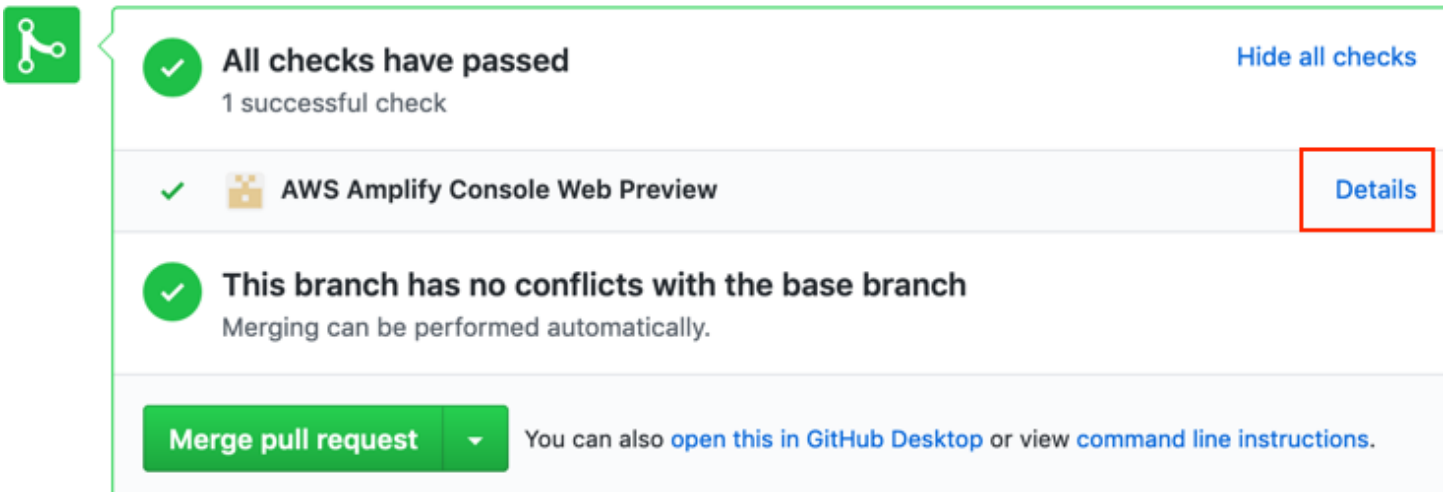


4. 在「管理分支預覽設定」視窗中，開啟「提取請求預覽」。
5. 如需如何設定全堆疊應用程式的說明：
   - 選擇，為每個提取請求創建新的後端環境。此選項可讓您在不影響生產環境的情況下測試變更。
   - 選擇將此分支的所有提取請求指向現有環境。
6. 選擇 Confirm (確認)。

下次您提交分支的提取請求時，Amplify 會構建您的 PR 並將其部署到預覽 URL。



僅適用於 GitHub 儲存庫，您可以直接從 GitHub 帳戶中的提取要求存取 URL 的預覽。



關閉提取請求後，預覽 URL 將被刪除，並刪除鏈接到提取請求的任何臨時後端環境。

使用子網域進行 Web 預覽存取

提取請求的網頁預覽可透過 Amplify 應用程式的子網域存取，該應用程式連接至 Amazon Route 53 所管理的自訂網域。當提取請求關閉時，與提取請求相關聯的分支和子網域會自動刪除。這是您為應用程式設定以模式為基礎的功能分支部署之後，網頁預覽的預設行為。如需如何設定自動子網域的說明，請參閱為 Amazon 路線 53 自定義域設置自動子域。

將端對端賽普拉斯測試添加到您的應用程序

您可以在 Amplify 應用程式的測試階段執行端對端 (E2E) 測試，以便在將程式碼推送至生產環境之前捕捉回歸。測試階段可以在組建規格 YAML 中進行配置。目前，您只能在建置期間執行 Cypress 測試架構。

教學課程：使用 Cypress 設定端對端測試

Cypress 是一個JavaScript基於測試框架，允許您在瀏覽器上運行 E2E 測試。如需示範如何設定 E2E 測試的教學課程，請參閱使用 Amplify 針對完整堆疊 CI/CD 部署執行端對端 Cypress 測試的部落格文章。

將測試新增至您現有的擴增應用程式

您可以通過在 Amplify 控制台中更新應用程序的構建設置，將 Cypress 測試添加到現有的應用程序。組建規格 YAML 包含組建命令集合，以及 Amplify 用來執行組建的相關設定。使用`test`步驟在構建時運行任何測試命令。對於 E2E 測試，放大託管提供了與賽普拉斯的更深入的集成，使您可以為測試生成 UI 報告。

下列清單說明測試設定及其使用方式。

預先測試

安裝運行賽普拉斯測試所需的依賴關係。放大託管使用 mochawesome 生成報告以查看您的測試結果，並等待在構建過程中設置本地主機服務器。

test

運行柏樹命令來使用 mochawesome 執行測試。

後期測試

該報告是從輸出 JSON 生成的。請注意，如果您使用的是 Yarn，您必須以靜音模式執行此指令，才能產生 mochawesome 報告。對於 Yarn，您可以使用以下命令。

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

人工藝品 > 基本目錄

執行測試的目錄。

人工藝品 > configFilePath

生成的測試報告數據。

人工因素 > 檔案

生成的工件（屏幕截圖和視頻）可供下載。

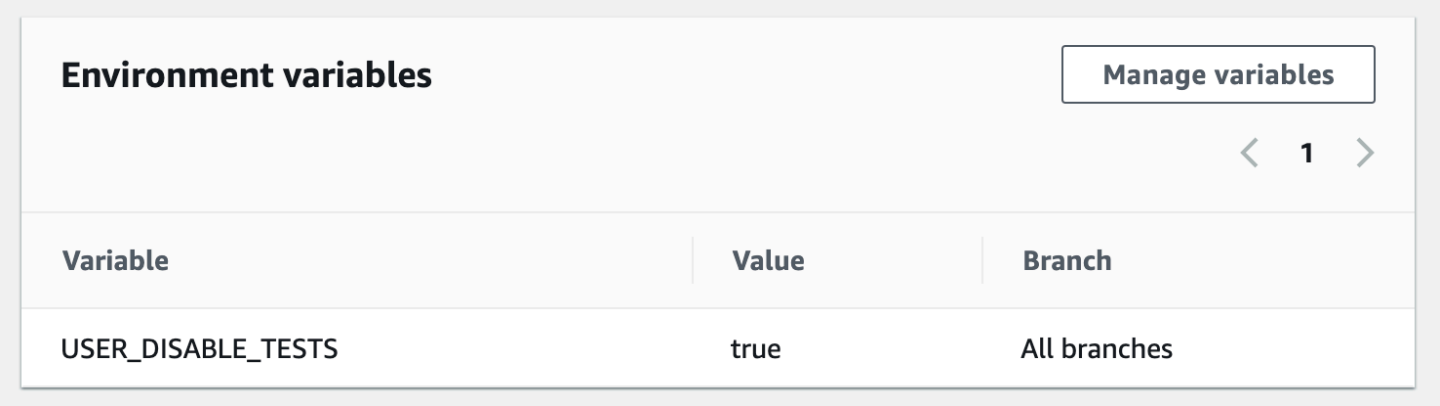
下列範例摘錄自組建規格amplify.yml檔案，說明如何將 Cypress 測試新增至您的應用程式。

```
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'
```

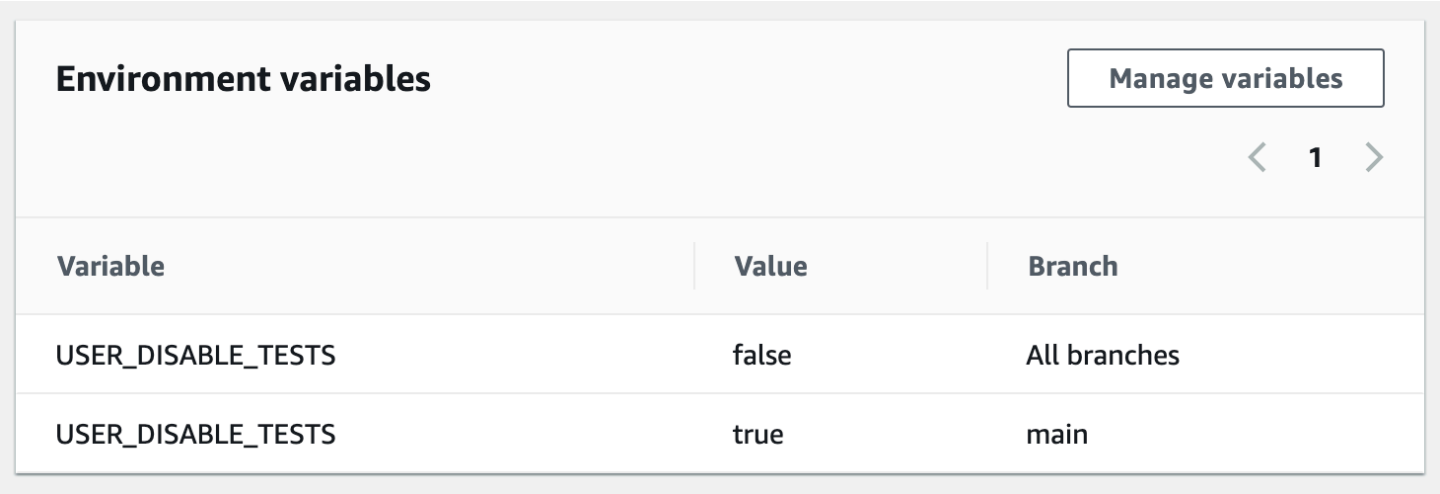
禁用測試

將測試配置添加到amplify.yml構建設置後，該test步驟會在每個分支上為每個構建運行。如果您想全局禁用運行測試，或者僅對特定分支運行測試，則可以使用USER_DISABLE_TESTS環境變量而無需修改構建設置。

要全局禁用所有分支的測試，請添加所有分支的值true為的USER_DISABLE_TESTS環境變量。下列螢幕擷取畫面顯示 Amplify 主控台中的 [環境變數] 區段，並針對所有分支停用測試。



要禁用特定分支的測試，請添加所有分支的值false為的USER_DISABLE_TESTS環境變量，然後為您要禁用的每個分支添加一個覆蓋，其值為true。在下面的屏幕截圖中，測試在主分支上被禁用，並為每隔一個分支啟用。



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。

使用重定向

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、移動應用程式或網站的主機位置而不變更其位址，以及將要求的 URL 變更為網路應用程式所需的表單。

重定向的類型

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。原始位址的搜尋引擎排名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。
- 為了提供用戶界面 A/B 比較的測試頁面。

> **Note**
>
> 如果您的應用返回意外 302 響應，則錯誤可能是由您對應用程序的重定向和自定義標題配置所做的更改引起的。要解決此問題，請驗證您的自定義標題是否有效，然後重新啟用應用程序的默認 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 重定向。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。

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。在開始之前，您將需要有關重定向部分的以下信息。

原始地址

使用者要求的位址。

一個目的地地址

實際提供使用者所看到內容的位址。

重定向類型

類型包括永久重定向（301），臨時重定向（302），重寫（200）或找不到（404）。

兩個字母的國家代碼（可選）

您可以包含在按地理區域劃分應用程式使用者體驗的值。

若要在 Amplify 控制台中建立重新導向

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。

2. 選擇您要為其建立重新導向的應用程式。
3. 在功能窗格中，選擇 [應用程式設定]，然後選擇 [重寫和重新導向]。
4. 在「重寫和重新導向」區段中，選擇 「編輯」。
5. 新增重新導向的程序會因您要個別新增規則或進行大量編輯而有所不同：
   - 若要建立個別重新導向，請選擇 [新增規則]。
     a. 針對來源位址，輸入使用者要求的原始位址。
     b. 在「目標位址」中，輸入將內容呈現給使用者的目的地位址。
     c. 在「類型」中，從清單中選擇重新導向的類型。
     d. (選擇性) 針對國家/地區代碼，輸入兩個字母的國碼條件。
   - 若要大量編輯重新導向，請選擇「開啟文字編輯
     - 在大量添加重寫和重定向 JSON 編輯器中手動添加或更新重定向。
6. 選擇 儲存。

重定向順序

重新導向會從清單的上方往下執行。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
```

查詢參數

您可以使用查詢參數來進一步控制 URL 相符項目。Amplify 會將 301 和 302 重新導向的所有查詢參數轉送至目的地路徑，但下列例外：

- 如果原始位址包含設定為特定值的查詢字串，Amplify 不會轉寄查詢參數。在此情況下，重新導向只會套用至具有指定查詢值之目的地 URL 的要求。

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

簡單的重定向和重寫

本節包含常見重新導向案例的範例程式碼。

> **Note**
>
> 原始位址網域比對不區分大小寫。

您可以使用以下範例程式碼，將特定頁面永久重新導向到新的地址。

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 [{"source": "/original.html", "status": "301", "target": "/destination.html", "condition": null}]

您可以使用以下範例程式碼，將資料夾下的任何路徑重新導向到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 [{"source": "/docs/<*>", "status": "301", "target": "/documents/<*>", "condition": null}]

您可以使用以下範例程式碼，將所有流量重新導向到 index.html 做為重寫。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 [{"source": "/<*>", "status": "200", "target": "/index.html", "condition": null}]

您可以使用以下範例程式碼，使用重寫來變更要向使用者顯示的子網域。

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 [{"source": "https://mydomain.com", "status": "200", "target": "https://www.mydomain.com", "condition": null}]

您可以使用下列範例程式碼，以路徑前置詞重新導向至不同的網域。

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 [{"source": "https://mydomain.com", "status": "302", "target": "https://www.mydomain.com/documents/", "condition": null}]

您可以使用下列範例程式碼，將無法找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 [{"source": "/<*>", "status": "404", "target": "/404.html", "condition": null}]

單頁網頁應用程式 (SPA) 的重新導向

大多數 SPA 架構支援使用 HTML5 history.pushState() 來變更瀏覽器位置，而不需觸發伺服器請求。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。

下列範例會使用規則運算式，將所有檔案的 200 重新寫入設定為 index.html，但規則運算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 [{"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

下列範例會使用重新寫入來代理其他位置的內容，讓使用者看到網域尚未變更。

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 [{"source": "/images/<*>", "status": "200", "target": "https://images.otherdomain.com/<*>", "condition": null}]

尾隨斜線和乾淨的 URL

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)。如果需要，「Amplify」會自動建立乾淨的 URL，方法是加入尾端斜線。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

您可以使用以下範例程式碼，將資料夾結構中的路徑重新導向到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 [{"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

您可以使用以下範例程式碼，將路徑重新導向至具有的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 [{"source": "/docs?id=<my-blog-id-value>", "status": "301", "target": "/documents/<my-blog-id-value>", "condition": null}]

> **Note**
>
> Amplify 會將 301 和 302 重新導向的所有查詢字串參數轉送至目標路徑。不過，如果原始位址包含設定為特定值的查詢字串 (如本範例所示)，Amplify 不會轉寄查詢參數。在此情況下，重新導向只會套用至具有指定查詢值之目的地位址的要求id。

您可以使用下列範例程式碼，將無法在指定資料夾結構層級找到的所有路徑，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 [{"source": "/documents/<x>/<y>/<z>", "status": "404", "target": "/documents/index.html", "condition": null}]

基於區域的重定向

您可以使用以下範例程式碼，根據區域來重新導向請求。

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 [{"source": "/documents", "status": "302", "target": "/documents/us/", "condition": "<US>"}]

重定向和重寫中的萬用字元運算式

您可以在原始位址中使用萬用字元運算式<*>，以進行重新導向或重新寫入。您必須將運算式放在原始位址的結尾，且必須是唯一的。Amplify 會忽略包含多個萬用字元運算式的原始位址，或在不同的位置使用它。

以下是使用萬用字元運算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)	

下列兩個範例會示範含萬用字元運算式的無效重新導

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)	

限制對分支的訪問

如果您正在使用未發行的功能，則可以密碼保護尚未準備好公開存取的功能分支。在分支上設定存取控制時，當使用者嘗試存取分支的 URL 時，系統會提示使用者輸入使用者名稱和密碼。

在功能分支上設定密碼的步驟

1. 登入AWS Management Console並開啟 Amplify 大控制台。
2. 選擇您要設置功能分支密碼的應用程序。
3. 在功能窗格中，選擇 [應用程式設定]，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. 在存取控制設定中執行下列任一項作業：
   - 若要設定套用至所有連線分支的使用者名稱和密碼，請開啟 [套用全域密碼]。例如，如果您連接了 main，dev 和功能分支，則可以使用全局密碼為所有分支設置相同的用戶名和密碼。
   - 若要將使用者名稱和密碼套用至個別分支，請關閉 [套用全域密碼]。對於您要為其設定唯一使用者名稱和密碼的分支，請選擇存取設定所需的限制密碼，然後輸入使用者名稱和密碼。
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 儲存庫執行新的組建來重新部署應用程式。需要執行此步驟，才能啟用「Amplify」才能套用存取控制設定。

環境變數

環境變量是鍵值對，您可以將其添加到應用程序的設置中，以使其可用於 Amplify 託管。最佳作法是，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 Rogue 存取。

Amplify 不允許您使用AWS前綴創建環境變量。此前綴僅保留用於 Amplify 內部使用。

> **Important**
>
> 不要使用環境變量來存儲密碼。將密碼儲存在使用 AWS Systems Manager 參數存放區建立的環境密碼中。如需詳細資訊，請參閱 環境秘密。

Amplify 環境變數

根據預設，您可以在 Amplify 主控台中存取下列環境變數。

變數名稱	描述	範例值
_BUILD_TIMEOUT	建置逾時持續時間 (分鐘)	`30`
_LIVE_UPDATES	該工具將升級到最新版本。	`[{"name":"Amplify CLI","pkg":"@aws-amplify/cli","type":"npm","version":"latest"}]`
USER_DISABLE_TESTS	建置期間會略過測試步驟。您可以禁用應用程序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。如需設定此變數的更多資訊，請參閱禁用測試。	`true`
AWS_APP_ID	目前建置的應用程式 ID	`abcd1234`

變數名稱	描述	範例值
AWS_BRANCH	目前建置的分支名稱	`main`, `develop`, `beta`, `v2.0`
AWS_BRANCH_ARN	當前構建的分支 Amazon 資源名稱（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	當前構建的提交 ID 用於重建的「頭」	`abcd1234`
AWS_JOB_ID	目前建置的任務 ID。 這包括一些 '0' 的填充，所以它總是具有相同的長度。	`0000000001`
AWS_PULL_REQUEST_ID	Web 預覽組建的提取要求識別碼。 此環境變數僅在用 GitHub 作儲存庫提供者時可用。	`1`
AMPLIFY_AMAZON_CLIENT_ID	Amazon 客戶端 ID	`123456`
AMPLIFY_AMAZON_CLIENT_SECRET	Amazon 客戶機密	`example123456`
AMPLIFY_FACEBOOK_CLIENT_ID	臉書用戶端識別碼	`123456`
AMPLIFY_FACEBOOK_CLIENT_SECRET	臉書用戶端的秘密	`example123456`

變數名稱	描述	範例值
AMPLIFY_GOOGLE_CLIENT_ID	谷歌客戶端 ID	123456
AMPLIFY_GOOGLE_CLIENT_SECRET	谷歌客戶機密	example123456
AMPLIFY_DIFF_DEPLOY	啟用或停用基於差異的前端部署。如需詳細資訊，請參閱 啟用或禁用基於差異的前端構建和部署。	true
AMPLIFY_DIFF_DEPLOY_ROOT	用於基於差異的前端部署比較的路徑，相對於儲存庫的根目錄。	dist
AMPLIFY_DIFF_BACKEND	啟用或禁用基於差異的後端構建。如需更多資訊，請參閱 啟用或禁用基於差異的後端構建	true
AMPLIFY_BACKEND_PULL_ONLY	Amplify 管理此環境變數。如需更多資訊，請參閱 編輯現有的前端以指向不同的後端	true
AMPLIFY_BACKEND_APP_ID	Amplify 管理此環境變數。如需更多資訊，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，而且想要停用後端組建，請將此環境變數設定為true。	true
AMPLIFY_ENABLE_DEBUG_OUTPUT	將此變數設定true為可在記錄檔中列印堆疊追蹤。這對於調試後端構建錯誤很有幫助。	true

變數名稱	描述	範例值
AMPLIFY_MONOREPO_APP_ROOT	用於指定 monorepo 應用程序的應用程序根目錄的路徑，相對於存儲庫的根目錄。	`apps/react-app`
AMPLIFY_USERPOOL_ID	已匯入進行驗證之 Amazon Cognito 使用者集區的識別碼	`us-west-2_example`
AMPLIFY_WEBCLIENT_ID	Web 應用程序使用的應用程序客戶端的 ID 必須設定應用程式用戶端，以存取由 AMPIFY_USERPOOL_ID 環境變數指定的 Amazon Cognito 使用者集區。	`123456`
AMPLIFY_NATIVECLIENT_ID	本機應用程序使用的應用程序客戶端的 ID 必須設定應用程式用戶端，以存取由 AMPIFY_USERPOOL_ID 環境變數指定的 Amazon Cognito 使用者集區。	`123456`
AMPLIFY_IDENTITYPOOL_ID	Amazon Cognito 身份池的 ID	`example-identitypool-id`
AMPLIFY_PERMISSIONS_BOUNDARY_ARN	IAM 政策的 ARN 用作許可界限，適用於 Amplify 建立的所有 IAM 角色。如需詳細資訊，請參閱 APL IFY 產生之角色的 IAM 許可界限。	`arn:aws:iam::123456789012:policy/example-policy`
AMPLIFY_DESTRUCTIVE_UPDATES	將此環境變數設定為 true，以允許使用可能造成資料遺失的結構描述作業來更新 GraphQL API。如需詳細資訊，請參閱更新結構描述。	`true`

> **Note**
>
> AMPLIFY_AMAZON_CLIENT_ID和AMPLIFY_AMAZON_CLIENT_SECRET環境變量是 OAuth 令牌，而不是 AWS 訪問密鑰和密鑰。

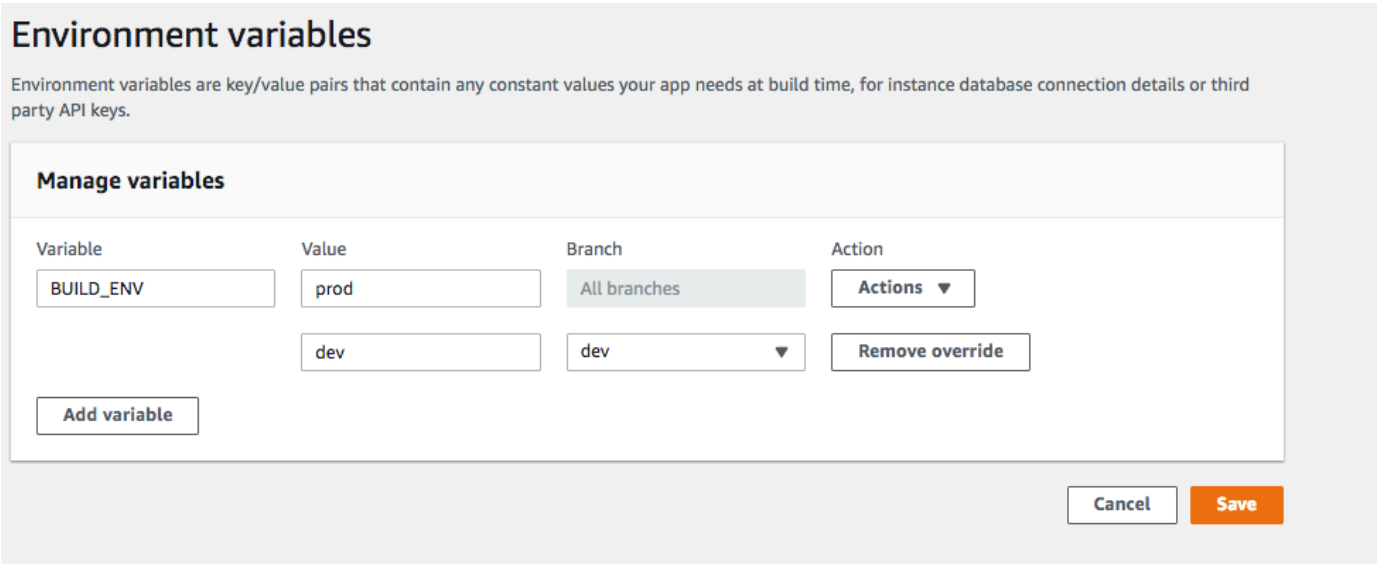
設定環境變數

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線到 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] 功能表中。如需有關此部署類型的指示，請參閱開始使用現有程式碼。

若要設定環境變數

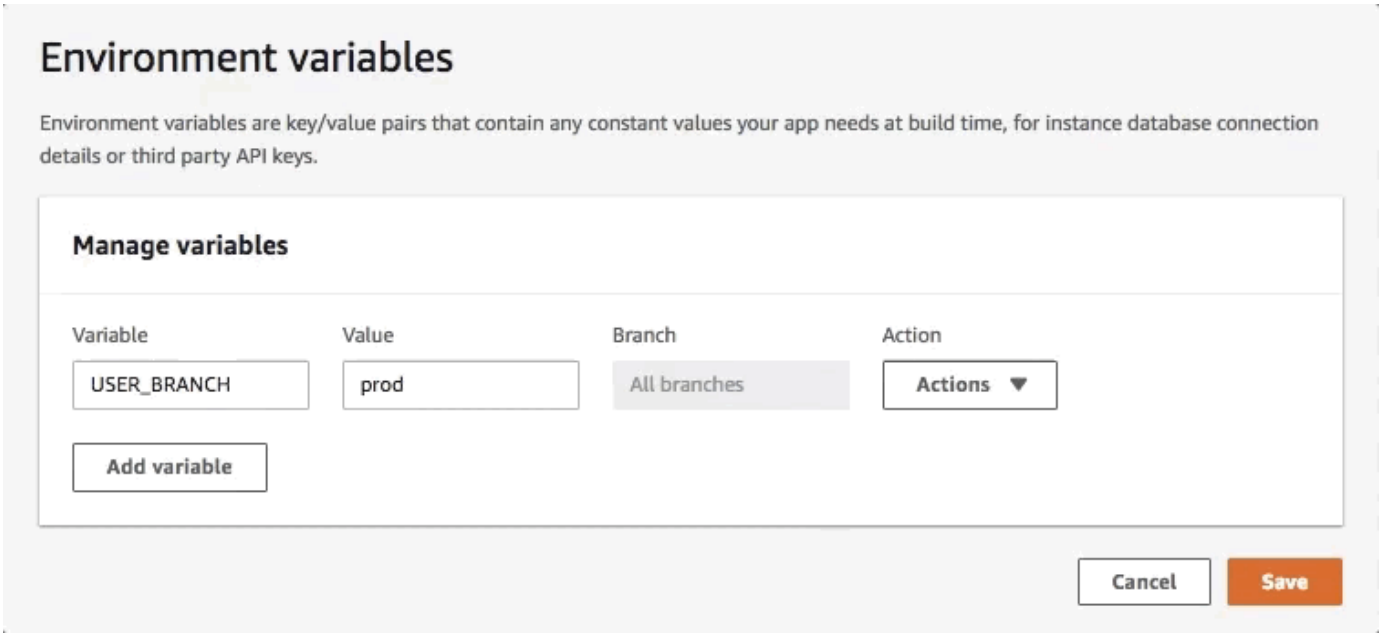
1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 在 Amplify 主控台中，選擇 [應用程式設定]，然後選擇 [環境變數]。
3. 在「環境變數」區段中，選擇「管理變數」。
4. 在「管理變數」區段的「變數」下，輸入金鑰。在「值」中，輸入您的值。根據預設，Amplify 會在所有分支上套用環境變數，因此在連接新分支時不必重新輸入變數。



5. (選擇性) 若要專門為分支自訂環境變數，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。

在構建時訪問環境變量

若要在建置期間存取環境變數，請編輯您的建置設定以在您的建置命令中包含環境變數。

若要編輯建置設定以包含環境變數

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 在 Amplify 主控台中，選擇「應用程式設定」，然後選擇「建置設定」
3. 在 [App 組建規格] 區段中，選擇 [編輯]。
4. 將環境變數新增至您的建置命令。您現在應該能夠在下一個建置期間存取環境變數。此範例會變更 npm 的行為 (BUILD_ENV)，並將外部服務的 API 權杖 (TWITCH_CLIENT_ID) 新增至環境檔案以供日後使用。

```
build:
  commands:
    - npm run build:$BUILD_ENV
```

```
    - echo "TWITCH_CLIENT_ID=$TWITCH_CLIENT_ID" >> backend/.env
```

構建配置中的每個命令都在 Bash shell 中運行。有關在 Bash 中使用環境變量的更多信息，請參閱《GNU Bash 手冊》中的外殼擴展。

讓環境變數可供伺服器端執行階段存取

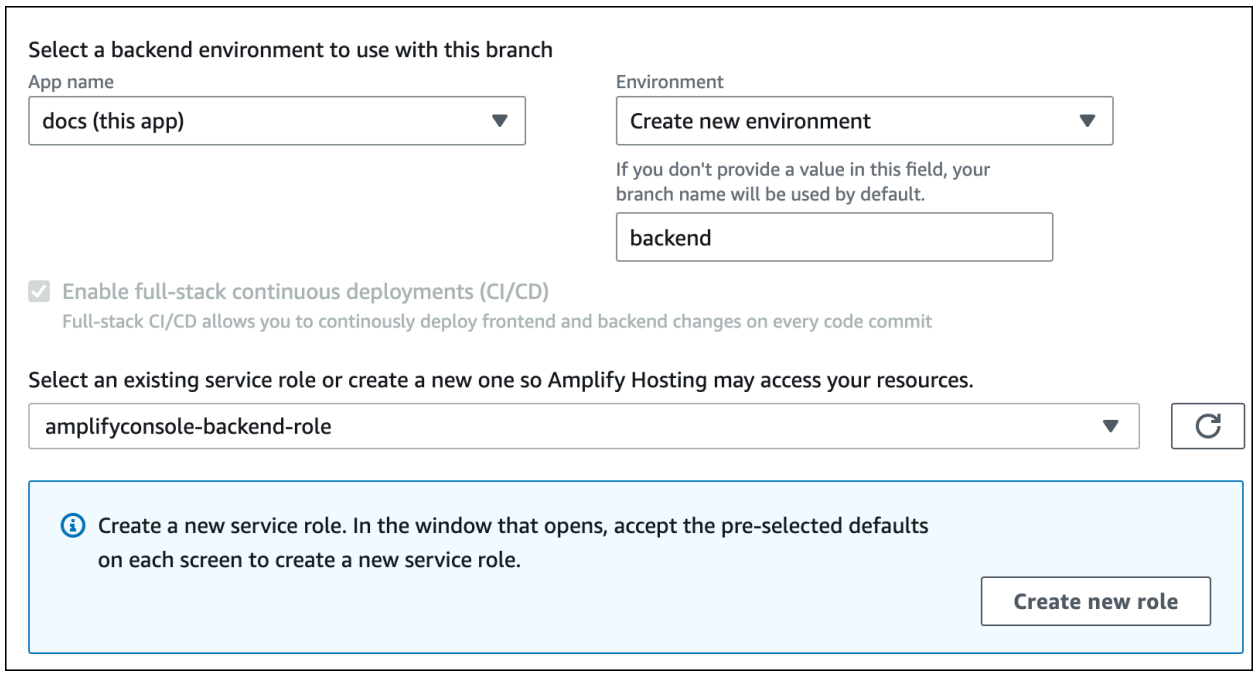
默認情況下，Next.js 服務器組件無法訪問應用程序的環境變量。此行為是刻意保護應用程式在建置階段期間所使用之環境變數中儲存的任何密碼。

若要讓 Next.js 可存取特定的環境變數，您必須修改 Amplify 建置規格檔案，以便在 Next.js 可辨識的環境檔案中設定環境變數。這可讓 Amplify 在建置應用程式之前載入環境變數。如需有關修改組建規格的詳細資訊，請參閱建置命令一節中如何新增環境變數的範例。

使用社交登錄的身份驗證參數創建新的後端環境

將分支連線至應用程式

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 將分支連接到應用程序的過程取決於您是將分支連接到新應用程序還是現有應用程序。
   - 將分支連接到新的應用程序
     a. 在 [組建設定] 頁面上，找到 [選取要與此分支搭配使用的後端環境] 區段。在「環境」中，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 [建置設定] 頁面的 [選取要與此分支搭配使用的後端環境] 區段，並**backend**輸入後端環境名稱。



b. 展開 [組建設定] 頁面上的 [進階設定] 區段，然後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Amplify 環境變數的表格。

- 將分支連接到現有的應用程序

a. 如果要將新分支連接到現有應用程序，請在連接分支之前設置社交登錄環境變量。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. 在變數 (金鑰) 中，輸入您的用戶端 ID。在「值」中，輸入您的用戶端密碼。

e. 選擇，保存。

前端框架環境變量

如果您使用支援自己環境變數的前端架構來開發應用程式，請務必瞭解這些變數與您在 Amplify 主控台中設定的環境變數不同。例如，React（前綴 REACT_APP）和 Gatsby（前綴為 GATSBY），使您能夠創建運行時環境變量，這些框架會自動捆綁到您的前端生產構建中。若要瞭解使用這些環境變數來儲存值的影響，請參閱您所使用之前端架構的文件。

在這些前端框架前綴的環境變量中存儲敏感值（例如 API 密鑰）並不是最佳實踐，因此不建議使用。如需針對此目的使用 Amplify 的建置時間環境變數的範例，請參閱在構建時訪問環境變量。

環境秘密

環境密碼與環境變數類似，但它們是可以加密的 AWS Systems Manager (SSM) 參數存放區金鑰值組。某些值必須加密，例如「使用 Apple 私鑰登錄」以進行「Amplify」。

設定環境密碼

請遵循下列指示，使用 AWS Systems Manager 主控台為 Amplify 應用程式設定環境密碼。

設定環境密碼

1. 登入 AWS Management Console 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 Parameter Store 頁面上，選擇建立參數。
4. 在「建立參數」頁面的「參數詳細資訊」段落中，執行下列動作：
   a. 在「名稱」中，以格式輸入參數**/amplify/{your_app_id}/{your_backend_environment_name}/{your_parameter_name}**。
   b. 針對 Type (類型)，選擇 SecureString。
   c. 對於 KMS 金鑰來源，請選擇 [我目前的帳戶] 以使用帳戶的預設金鑰。
   d. 在值中，輸入要加密的密碼值。
5. 選擇，創建參數。

> **Note**
>
> Amplify 只能存取特定環境組`/amplify/{your_app_id}/{your_backend_environment_name}`建下的金鑰。您必須指定預設值， AWS KMS key 才能允許「Amplify」解密值。

存取環境機密

在建置期間存取環境機密與存取環境變數類似，不同之處在於環境密碼會儲存`process.env.secrets`為 JSON 字串。

Amplify 環境機密

以格式指定「Systems Manager」參數`/amplify/{your_app_id}/{your_backend_environment_name}/AMPLIFY_SIWA_CLIENT_ID`。

您可以使用下列預設可在 Amplify 主控台中存取的環境密碼。

變數名稱	描述	範例值
AMPLIFY_SIWA_CLIENT_ID	使用蘋果客戶端 ID 登錄	`com.yourapp.auth`
AMPLIFY_SIWA_TEAM_ID	使用蘋果團隊 ID 登錄	`ABCD123`
AMPLIFY_SIWA_KEY_ID	使用蘋果密鑰 ID 登錄	`ABCD123`
AMPLIFY_SIWA_PRIVATE_KEY	使用蘋果私鑰登錄	----#-開始私鑰----- ****..... -----END 私鑰----

自訂標頭

自訂 HTTP 標頭可讓您指定每個 HTTP 回應的標頭。回應標頭可用在偵錯、安全和資訊方面。您可以在中指定標題AWS Management Console，或透過下載和編輯應用程式的`customHttp.yml`檔案並將其儲存在專案的根目錄中。如需詳細程序，請參閱設置自定義標題。

先前，透過編輯中的組建規格 (buildspec)，AWS Management Console或下載並更新檔案並將其儲存在專`amplify.yml`案的根目錄中，為應用程式指定自訂 HTTP 標頭。以這種方式指定的自定義標題應該從 buildspec 和文件中遷移出來。`amplify.yml`如需相關指示，請參閱移轉自訂標頭。

自定義標題 YAML 格式

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'
```

如果是單一儲存庫，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'
```

將自定義標題添加到應用程序時，您將為以下內容指定自己的值：

pattern

自訂標頭會套用符合該模式的所有 URL 檔案路徑。

標頭

定義與檔案模式相符的標頭。

key

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。

設置自定義標題

有兩種方法可以為AWS Amplify應用程序指定自定義 HTTP 標頭。您可以在中指定標題，AWS Management Console也可以透過下載和編輯應用程式的`customHttp.yml`檔案並將其儲存在專案的根目錄中來指定標題。

若要設定應用程式的自訂標題 AWS Management Console

1. 登入AWS Management Console並開啟擴大控制台。
2. 選擇要為其設置自定義標題的應用程序。
3. 在導覽窗格中，選擇 [應用程式設定] > [自訂標題]。
4. 在「自訂頁首規格」區段中，選擇 「編輯」。
5. 在 「編輯」視窗中，使用自訂標頭 YAML 格式輸入自訂標題的資訊。
   a. 對於`pattern`，輸入要符合的樣式。
   b. 在中`key`，輸入自訂標頭的名稱。

c. 在中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，請瀏覽至要部署的分支，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。

若要使用自訂 Httpp.yML 檔案設定自訂標頭

1. 登入AWS Management Console並開啟擴大控制台。
2. 選擇要為其設置自定義標題的應用程序。
3. 在導覽窗格中，選擇 [應用程式設定] > [自訂標題]。
4. 在「自訂頁首規格」區段中，選擇「下載」。
5. 在您選擇的代碼編輯器中打開下載的customHttp.yml文件，然後使用自定義標題 YAML 格式輸入自定義標題的信息。

   a. 對於pattern，輸入要符合的樣式。

   b. 在中key，輸入自訂標頭的名稱。

   c. 在中value，輸入自訂標頭的值。
6. 將編輯過的customHttp.yml檔案儲存在專案的根目錄中。如果您正在使用 monorepo，請將customHttp.yml文件保存在回購的根目錄中。
7. 重新部署應用程式以套用新的自訂標頭。
   - 如果是 CI/CD 應用程式，請從包含新檔案的 Git 儲存庫執行新customHttp.yml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檔案與您上傳的成品一起包含在內。

> **Note**
>
> 在customHttp.yml檔案中設定並部署在應用程式根目錄中的自訂標頭將覆寫在AWS Management Console.

移轉自訂標頭

之前，自訂 HTTP 標頭是透過編輯中的 buildspec，AWS Management Console或下載和更新檔案並將其儲存在專amplify.yml案的根目錄中來指定應用程式。強烈建議您將自訂標頭移轉出組建規格和檔案amplify.yml。

在的 [自訂標題] 區段中指定您的自訂標頭，AWS Management Console或透過下載和編輯customHttp.yml檔案。

移轉儲存在 Amplify 主控台中的自訂標頭

1. 登入AWS Management Console並開啟擴大控制台。
2. 選擇要在其上執行自定義標題遷移的應用程序。
3. 在導覽窗格中，選擇 [應用程式設定] > [建置設定]。在「應用程序構建規範」部分中，您可以查看應用程序的構建規格。
4. 選擇「下載」以儲存目前組建規格的副本。如果您稍後需要復原任何設定，可以參考此複本。
5. 下載完成時，選擇 [編輯]。
6. 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，刪除檔案中的任何自訂標題，然後選擇「儲存」。
7. 在導覽窗格中，選擇 [應用程式設定] > [自訂標題]。
8. 在「自訂頁首規格」區段中，選擇 「編輯」。
9. 在「編輯」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. 重新部署您想要套用新自訂標頭的任何分支。

若要將自訂標頭從放大的 .yml 移轉至自訂 Http.yML

1. 導航到應用程序根目錄中當前部署的amplify.yml文件。
2. amplify.yml在您選擇的程式碼編輯器中開啟。
3. 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 Management Console並開啟擴大控制台。
5. 選擇要為其設置自定義標題的應用程序。
6. 在導覽窗格中，選擇 [應用程式設定] > [自訂標題]。

7. 在「自訂頁首規格」區段中，選擇「下載」。
8. 在您選擇的程式碼編輯器中開啟下載的customHttp.yml檔案，然後輸入您amplify.yml在步驟 3 中刪除的自訂標頭資訊。
9. 將編輯過的customHttp.yml檔案儲存在專案的根目錄中。如果您正在使用 monorepo，請將文件保存在回購的根目錄中。
10. 重新部署應用程式以套用新的自訂標頭。
    - 如果是 CI/CD 應用程式，請從包含新檔案的 Git 儲存庫執行新customHttp.yml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，並包含您上傳的成品的新customHttp.yml檔案。

> **Note**
>
> 在customHttp.yml檔案中設定並部署在應用程式根目錄中的自訂標頭將覆寫在AWS Management Console.

自定義頁眉

當您在 monorepo 中為應用程序指定自定義標題時，請注意以下設置要求：

- 有一個特定的 YAML 格式為一個單一的。如需正確的語法，請參閱自定義標題 YAML 格式。
- 您可以使用的自訂標頭區段，為 monorepo 中的應用程式指定自訂標頭。AWS Management Console請注意，您必須重新部署應用程式才能套用新的自訂標頭。
- 作為使用控制台的替代方法，您可以在customHttp.yml文件中的 monorepo 中為應用程序指定自定義標題。您必須將customHttp.yml檔案儲存在軟體庫的根目錄中，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的自訂標頭區段指定的任何自訂標頭AWS Management Console。

安全性標頭範例

自訂安全標頭可讓您強制執行 HTTPS、防止 XSS 攻擊，以及防禦您的瀏覽器免於點擊劫持。使用下列 YAML 語法將自訂安全性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'"
```

快取控制標頭範例

您可以手動調整指s-maxage令，以更好地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快取的時間長度，您可以將存留時間 (TTL) 更新s-maxage為超過預設 600 秒 (10 分鐘) 的值，以手動方式增加存留時間 (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保留在邊緣快取 3600 秒 (一小時)。

```
customHeaders:
  - pattern: '/img/*'
    headers:
      - key: 'Cache-Control'
        value: 's-maxage=3600'
```

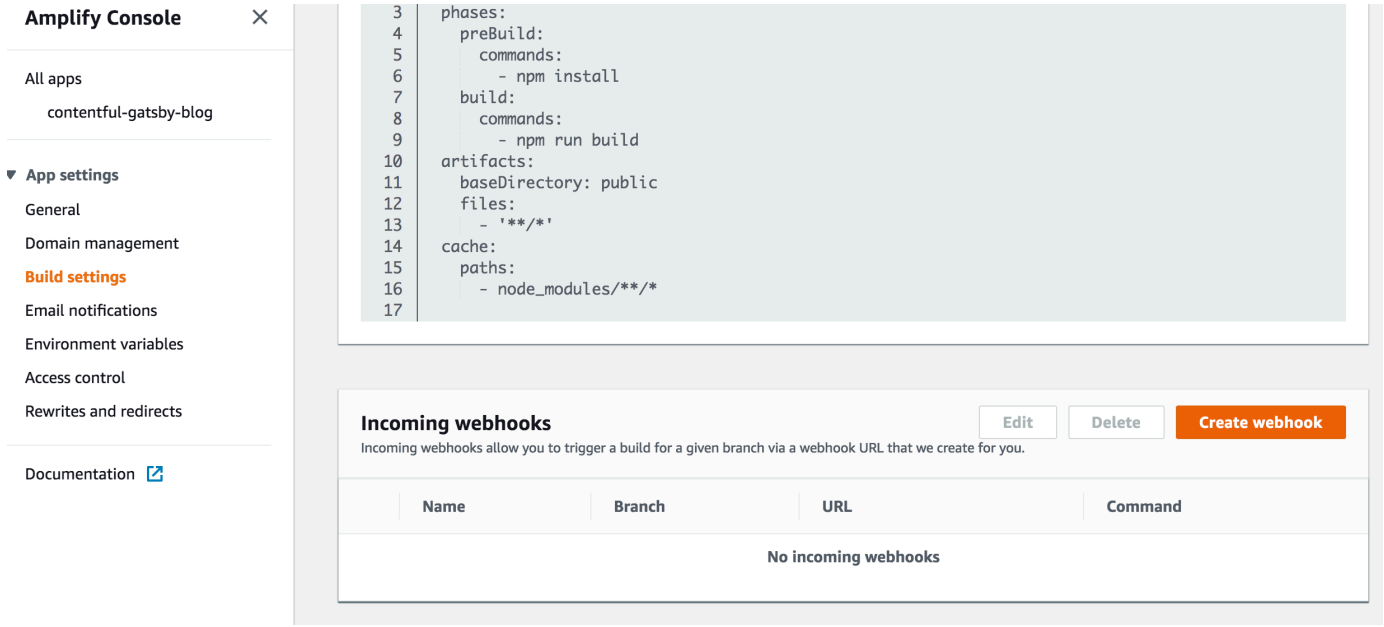
如需使用標頭控制應用程式效能的詳細資訊，請參閱使用標頭控制快取持續時間。

傳入網絡掛鉤

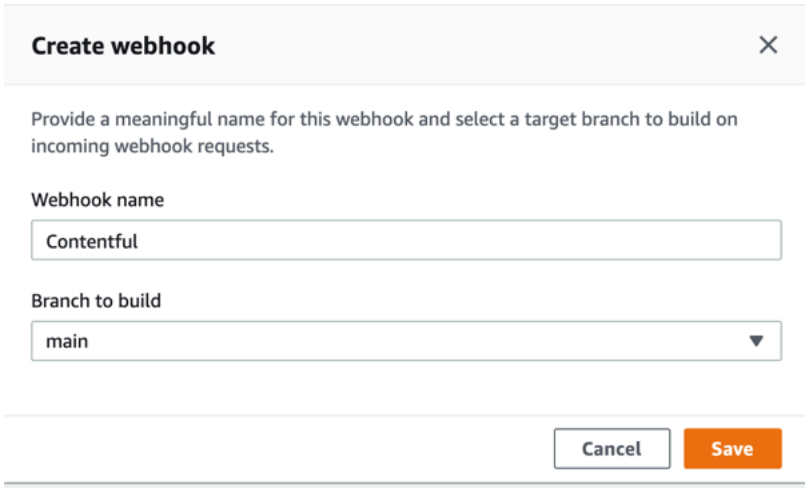
在 Amplify 控制台中設置傳入的 Webhook 以觸發構建，而無需將代碼提交到 Git 存儲庫。您可以將 webhook 觸發器與無頭 CMS 工具（例如內容或 GraphCMS）一起使用，在內容更改時啟動構建，或者使用 Zapier 之類的服務執行日常構建。

若要建立傳入的網路掛接

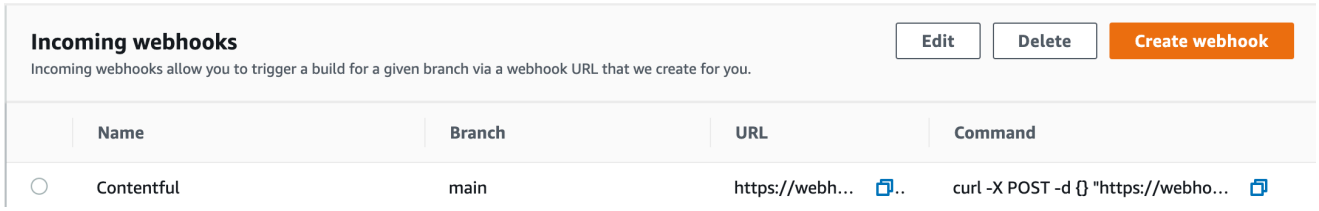
1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 選擇您要為其建立網路掛接的應用程式。
3. 在導覽窗格中，選擇 [建置設定]。
4. 在「構建設置」頁面上，向下滾動到「內送網絡掛鉤」部分，然後選擇「創建 web hook」。



5. 在「建立 Webhook」對話方塊中，執行下列動作：
   a. 針對網路掛接名稱，輸入網路掛接的名稱。
   b. 對於要構建的分支，請選擇要在傳入的 webhook 請求上構建的分支。
   c. 選擇儲存。



6. 在「內送網路掛接」區段中，執行下列其中一個動作：
   - 複製 webhook 網址並將其提供給無頭 CMS 工具或其他服務以觸發構建。
   - 在終端機視窗中執行 curl 指令以觸發新的組建。



監控

AWS Amplify 透過 Amazon 發出指標， CloudWatch 並提供存取日誌，其中包含對應用程式發出請求的詳細資訊。使用本節中的主題，瞭解如何使用這些指標和記錄來監視您的應用程式。

主題

使用監控 CloudWatch

AWS Amplify 與 Amazon 整合 CloudWatch，可讓您以近乎即時的速度監控 Amplify 應用程式的指標。您可以建立警示，以便在指標超過您設定的臨界值時傳送通知。如需有關 CloudWatch 服務運作方式的詳細資訊，請參閱 Amazon CloudWatch 使用者指南。

指標

Amplify 支援AWS/AmplifyHosting命名空間中的六個 CloudWatch 指標，用於監控應用程式的流量、錯誤、資料傳輸和延遲。這些測量結果會以一分鐘的間隔彙總。 CloudWatch 監控指標是免費的，不會計入CloudWatch 服務配額。

並非所有可用的統計資料都適用於每個量度。下表中最相關的統計資料列在每個測量結果的說明中。

指標	描述
請求	您的應用程式收到的檢視者要求總數。 最相關的統計數據是Sum。使用Sum統計資料來取得要求的總數。
BytesDownloaded	檢視者針對GET、和OPTIONS要求傳輸出應用程式 (下載) 的資料總量HEAD，以位元組為單位。 最相關的統計數據是Sum。
BytesUploaded	使用POST和PUT請求傳輸到您的應用程序（上傳）的數據總量（以字節為單位）。

指標	描述
	最相關的統計數據是Sum。
4XXErrors	傳回 HTTP 狀態碼 400-499 範圍內錯誤的要求數目。 最相關的統計數據是Sum。使用Sum統計資料來取得這些錯誤的總發生次數。
5XXErrors	在 HTTP 狀態碼 500-599 範圍內傳回錯誤的要求數目。 最相關的統計數據是Sum。使用Sum統計資料來取得這些錯誤的總發生次數。
Latency (延遲)	第一個位元組的時間 (以秒為單位)。這是 Amplify Hosting 接收請求到網絡返回響應之間的總時間。這不包括回應到達檢視者裝置時遇到的網路延遲。 最相關的統計資料是 `Average`、`Maximum`、`Minimum`、`p10`、`p50`、`p90`、`p95`、和`p100`。 使用`Average`統計資料來評估預期的延遲。

「Amplify」提供下列 CloudWatch 量度維度。

維度	描述
应用	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。

您可以在以下 AWS Management Console 位置存取 CloudWatch 指標：https://console.aws.amazon.com/cloudwatch/。或者，您也可以使用下列程序在 Amplify 主控台中存取指標。

若要在 Amplify 控制台中存取指標

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 選擇您要檢視其指標的應用程式。
3. 在功能窗格中，選擇 [應用程式設定]、[監控]。
4. 在「監督」頁面上，選擇測量結果。

警示

您可以在 Amplify 控制台中建立 CloudWatch 警報，以便在符合特定條件時傳送通知。警示會監視單一指 CloudWatch 標，並在指標超出指定評估期數的閾值時傳送 Amazon 簡單通知服務通知。

您可以在 CloudWatch主控台或使用 CloudWatch API 建立使用度量數學運算式的更進階警示。例如，您可以建立警示，在連續三個週期的百分比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度量數學運算式建立 CloudWatch 警示。

標準 CloudWatch 定價適用於警報。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。

建立 Amplify 量度的 CloudWatch 警示

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 選擇您要設定鬧鐘的應用程式。
3. 在功能窗格中，選擇 [應用程式設定]、[監控]。
4. 在 [監視] 頁面上，選擇 [警示]。
5. 選擇 Create alarm (建立警示)。
6. 在「建立警示」視窗中，依照下列方式設定您的警示：

    a. 在「測量結果」中，從清單中選擇要監督的測量結果名稱。
    b. 在警報名稱中，輸入有意義的警報名稱。例如，如果您正在監視請求，則可以命名警報**`HighTraffic`**。名稱只能包含 ASCII 字元。
    c. 對於「設定通知」，請執行下列其中一個動作：
        - i. 選擇「新增」以設定新的 Amazon SNS 主題。
          ii. 在「電子郵件地址」中，輸入通知收件者的電子郵件地址。
          iii. 選擇 [新增電子郵件地址] 以新增其他收件者。

- i. 選擇現有可重複使用 Amazon SNS 主題。
- ii. 對於 SNS 主題，請從清單中選取現有 Amazon SNS 主題的名稱。

d. 針對 「每當量度統計資料」，設定警示的條件，如下所示：

i. 指定測量結果必須大於、小於或等於臨界值。

ii. 指定閾值。

iii. 指定必須處於警示狀態才能觸發警示的連續評估週期數。

iv. 指定評估期間的時間長度。

e. 選擇 Create alarm (建立警示)。

> **Note**
>
> 您指定的每個 Amazon SNS 收件者都會收到來自 AWS 通知的確認電子郵件。電子郵件包含一個連結，收件者必須遵循該連結才能確認其訂閱並接收通知。

SSR 應用程式的 Amazon CloudWatch 日

Amplify 會將 Next.js 執行階段的相關資訊傳送至您 AWS 帳戶的. CloudWatch 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所假設的 IAM 服務角色。您可以允許 Amplify 託管計算自動為您創建服務角色，也可以指定已創建的角色。

如果您選擇允許 Amplify 為您建立 IAM 角色，則該角色已具有建立 CloudWatch 記錄的權限。如果您建立自己的 IAM 角色，則需要在政策中新增下列許可，以允許 Amplify 存取 Amazon CloudWatch 日誌。

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

如需服務角色的詳細資訊，請參閱新增服務角色。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 主機部署伺服器端轉譯應用程

存取日誌

擴大存儲您在 Amplify 中託管的所有應用程序的訪問日誌。存取記錄包含對託管應用程式發出要求的相關資訊。Amplify 會保留應用程式的所有存取記錄，直到您刪除應用程式為止。應用程式的所有存取記錄都可在 Amplify 主控台中取得。不過，每個個別的存取記錄要求都限制在您指定的兩週期間內。

Amplify 從不重複使用客戶之間的 CloudFront 分佈。Amplify 會事先建立發 CloudFront 行版，這樣您就不必等待在部署新應用程式時建立 CloudFront發行版。在將這些發行版分配給 Amplify 應用程序之前，它們可能會從機器人接收流量。但是，它們被配置為在分配之前始終響應為「未找到」。如果您在建立應用程式之前，應用程式的存取記錄包含一段時間內的項目，則這些項目與此活動相關。

> **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 會以最佳方式提供存取記錄。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。當存取記錄中省略記錄項目時，存取記錄檔中的項目數量將與 AWS 帳單和使用情況報告中顯示的使用量不符。

請使用下列程序擷取應用程式的存取記錄。

若要檢視存取記錄

1. 登入 AWS Management Console 並開啟 Amplify 大控制台。
2. 選擇您要檢視其存取記錄的應用程式。
3. 在功能窗格中，選擇 [應用程式設定]、[監控]。
4. 在 [監控] 頁面上，選擇 [存取記錄]。
5. 選擇編輯時間範圍。
6. 在 [編輯時間範圍] 視窗中，針對 [開始日期]，指定要擷取記錄之兩週間隔的第一天。在 [開始時間] 中，選擇第一天開始記錄擷取的時間。
7. Amplify 主控台會在 [存取記錄] 區段中顯示指定時間範圍的記錄。選擇「下載」，以 CSV 格式儲存記錄檔。

分析存取記錄

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。分析存取記錄的其中一種方法是使用 Athena。Athena 是一種互動式查詢服務，可協助您分析 AWS 服務的資料。您可以按照這裡的step-by-step 說明創建一個表。創建表格後，您可以按如下方式查詢數據。

```
SELECT SUM(bytes) AS total_bytes
FROM logs
WHERE "date" BETWEEN DATE '2018-06-09' AND DATE '2018-06-11'
LIMIT 100;
```

通知

您可以設置AWS Amplify應用程序的通知，以在構建成功或失敗時通知利益相關者或團隊成員。Amplify 託管會在您的帳戶中建立 Amazon 簡單通知服務 (SNS) 主題，並使用它來設定電子郵件通知。您可以設定通知套用至 Amplify 應用程式的所有分支或特定分支。

電子郵件通知

使用下列程序為 Amplify 應用程式的所有分支或特定分支設定電子郵件通知。

設定 Amplify 應用程式的電子郵件通知

1. 登入AWS Management Console並開啟擴大控制台。
2. 選擇您要設定電子郵件通知的應用程式。
3. 在功能窗格中，選擇 [應用程式設定]、[通知]，然後在 [電子郵件通知] 區段中選擇 [新增通知]。
4. 在「管理通知」區段中執行下列任一項作業：
   - 若要傳送單一分支的通知，對於「電子郵件」，請輸入要傳送通知的電子郵件地址。在「分支」中，選取要傳送通知的分支名稱。
   - 若要傳送所有連線分支機構的通知，對於「電子郵件」，請輸入要傳送通知的電子郵件地址。選擇「所有分支」做為「分支」。
5. 完成時，請選擇 Save (儲存)。

自定義構建映像和實時包更新

主題

自定義構建映像

您可以使用自訂組建映像，為 Amplify 應用程式提供自訂的建置環境。如果您在使用 Amplify 的預設容器進行組建期間安裝需要很長時間的特定相依性，您可以建立自己的 Docker 映像檔，並在組建期間參考它。圖像可以託管在 Amazon 彈性容器註冊表公共。

> **Note**
>
> 只有當應用程式設定為持續部署並連線到 git 儲存庫時，才能在 Amplify 主控台的 [應用程式設定] 功能表中看見建置設定。如需有關此部署類型的指示，請參閱開始使用現有程式碼。

自定義構建映像要求

若要讓自訂組建映像做為 Amplify 組建映像使用，它必須符合下列需求：

1. 支援 GNU C 程式庫 (glibc) 的 Linux 發行版本，例如 Amazon Linux，針對 x86-64 架構編譯。
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。如果缺少此依賴關係，則構建立即失敗而沒有任何輸出，因為我們的構建運行程序無法產生任何輸出。
3. Git：該影像上需安裝 Git，才能複製 Git 儲存庫。如果遺漏此相依性，「複製儲存區域」步驟將會失敗。
4. OpenSSH：為了安全地克隆您的存儲庫，我們需要 OpenSSH 在構建過程中臨時設置 SSH 密鑰。OpenSSH 套件會提供建置執行程式執行此作業所需的命令。
5. Bash 和伯恩殼牌：這兩個實用程序用於在構建時運行命令。如果沒有安裝它們，您的組建可能會在啟動之前失敗。
6. 節點 .j+NPM：我們的構建運行程序不會安裝節點。相反，它依賴於正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。但是，我們強烈建議您安裝它們，因為當它們存在時，Amplify 構建運行程序可以使用這些工具來改善構建執行。當您為 Hugo 設置覆蓋時，Amplify 的軟件包覆蓋功能使用 NPM 來安裝 Hugo 擴展的軟件包。

不需要下列套件，但我們強烈建議您安裝它們。

1. NVM (Node Version Manager)：我們建議您安裝此版本管理員，如果您需要處理不同版本的Node. 當您設定覆寫時，Amplify 的套件覆寫功能會使用NVM在每次建置之前變更 Node.js 版本。
2. Wget: Amplify 可以使用該Wget實用程序在構建過程中下載文件。我們建議您將其安裝在自訂映像檔中。
3. 焦點：Amplify 可以使用該Tar實用程序在構建過程中解壓縮下載的文件。我們建議您將其安裝在自訂映像檔中。

配置自定義構建映像

若要設定在 Amazon ECR 中託管的自訂建置映像

1. 請參閱 Amazon ECR 公開使用者指南中的入門，以設定具有泊塢視窗映像的 Amazon ECR 公用存放庫。
2. 登入AWS Management Console並開啟 Amplify 大控制台。
3. 選擇您要為其配置自定義構建映像的應用程序。
4. 在導覽窗格中，選擇 [應用程式設定] > [建置設定]。
5. 在 [組建設定] 頁面的 [組建映像設定] 區段中，選擇 [編輯]。
6. 在 [編輯組建映像設定] 對話方塊中，展開 [建置映像] 功能表，然後選擇 [建置映像]。
7. 輸入您在步驟 1 中建立的 Amazon ECR 公開存放庫的名稱。這是您的構建映像託管的位置。例如，如果您的軟件庫的名稱是 ecr-examplerepo，您可以輸入。**`public.ecr.aws/xxxxxxxx/ecr-examplerepo`**
8. 選擇儲存。

即時套件更新

即時套件更新可讓您指定要在 Amplify 預設組建映像中使用的套件版本和相依性。默認構建映像帶有幾個預先安裝的軟件包和依賴關係（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，或確定永遠安裝最新版本。

如果啟用了實時包更新，則在構建運行之前，構建運行程序首先更新（或降級）指定的依賴關係。這會增加構建時間與更新依賴關係所需的時間成正比，但好處是您可以確保使用相同版本的依賴項來構建您的應用程序。

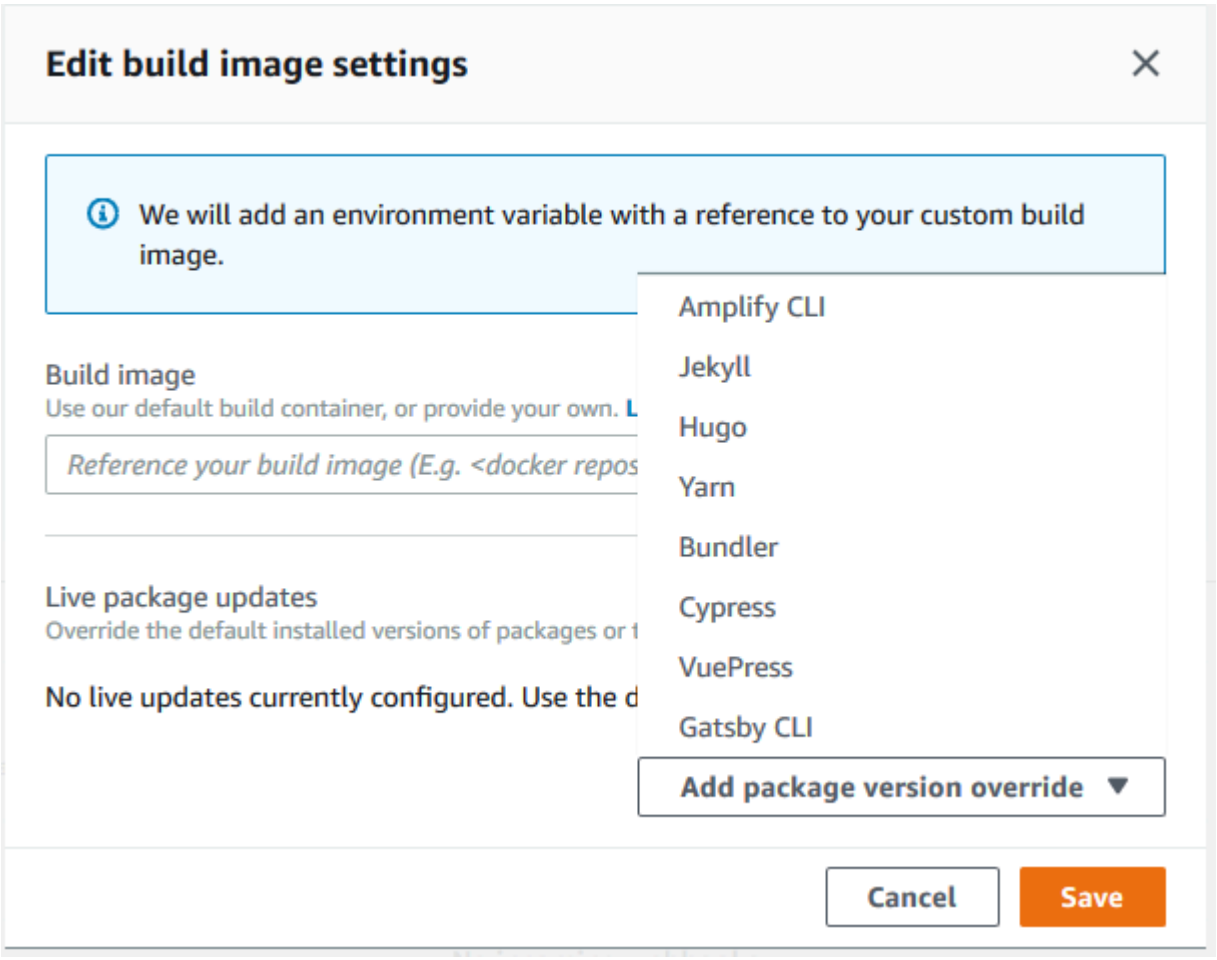
> **Warning**
>
> 將 Node.js 版本設定為最新版本會導致組建失敗。相反地，您必須指定完全相同的 Node.js 版本18，例如21.5、或v0.1.2。

設定即時套件更新

若要設定即時套件更新

1. 登入AWS Management Console並開啟 Amplify 大控制台。
2. 選擇您要設定即時套件更新的應用程式。
3. 在導覽窗格中，選擇 [應用程式設定] > [建置設定]。
4. 在 [組建設定] 頁面的 [組建映像設定] 區段中，選擇 [編輯]。
5. 在 [編輯組建映像設定] 對話方塊中，展開 [新增套件版本覆寫] 清單，然後選擇要變更的套件。

   下列螢幕擷取畫面顯示 [編輯組建映像設定] 對話方塊，其中包含 [新增套件版本覆寫] 清單



6. 對於「版本」，請保持最新的預設值，或輸入相依性的特定版本。如果您使用 latest，相依性會一律升級至可用的最新版本。
7. 選擇 Save (儲存)。

新增服務角色

Amplify 需要使用前端部署後端資源的權限。您會使用服務角色來達成此目標。服務角色是 Amplify 代表您呼叫其他服務時所承擔的 AWS Identity and Access Management (IAM) 角色。在本指南中，您將建立具有帳戶管理權限的 Amplify 服務角色，且明確性允許直接存取 Amplify 應用程式部署任何 Amplify Studio 或 CLI 資源所需的資源，以及建立和管理後端。如需有關擴增工作室的詳細資訊，請參閱 A mplify 文件中的入門。如需有關「Amplify CLI」的詳細資訊，請參閱 Amplify 文件中的「A mp lify CLI」。

步驟 1：登入 IAM 主控台

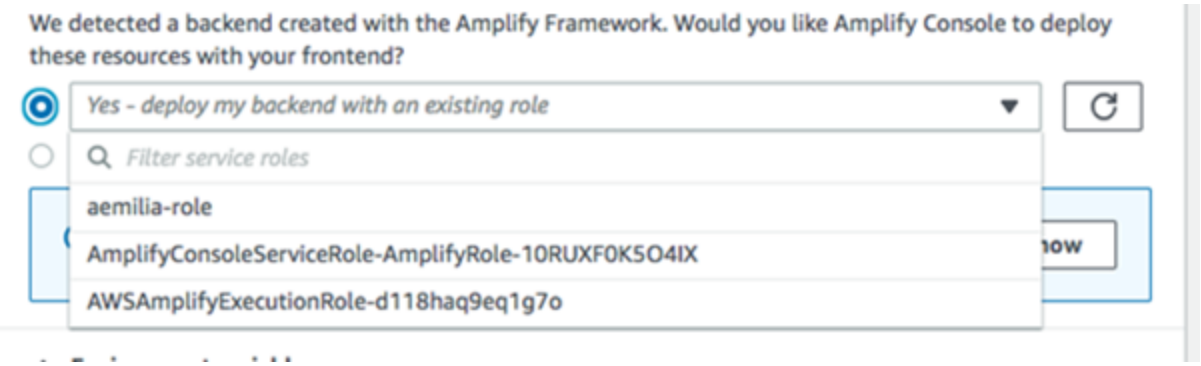
開啟 IAM 主控台並從左側導覽列選擇 [角色]，然後選擇 [建立角色]。

步驟 2：建立 Amplify 角色

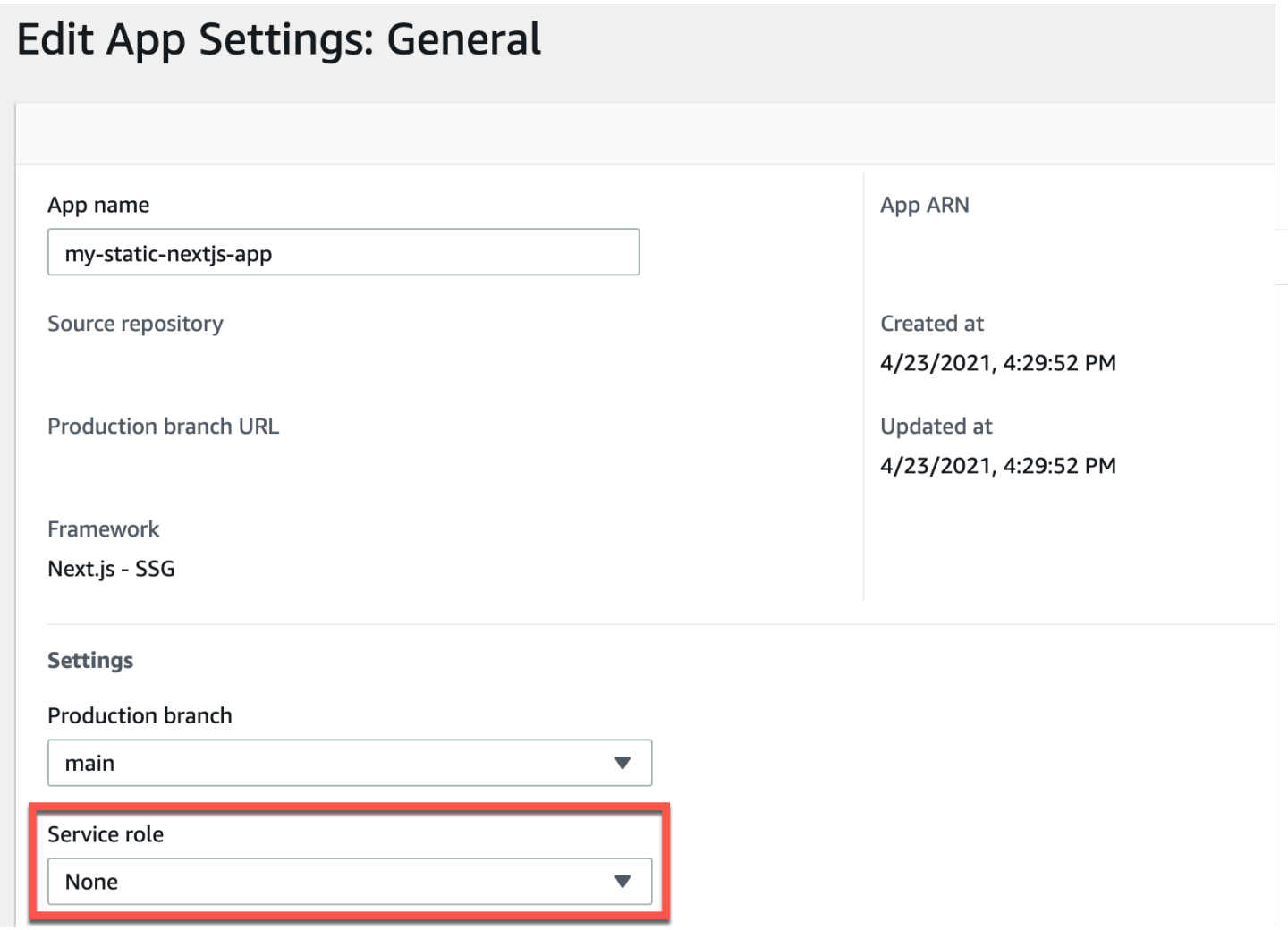
在角色選擇屏幕中找到 Amplify 並選擇擴大後端部署角色。接受所有預設值，並為您的角色選擇一個名稱，例如 AmplifyConsoleServiceRole-AmplifyRole。

步驟 3：返回 Amplify 控制台

開啟 Amplify 控制台。如果您正在部署新應用程式，請選擇 [重新整理]，然後選擇您剛建立的角色。它應該看起來像 AmplifyConsoleServiceRole-AmplifyRole。



如果您已具備現有應用程式，則可移至 App settings > General (應用程式設定 > 一般) 找到服務角色設定，然後在方塊右上角選擇 Edit (編輯)。從下拉式清單中選擇方才建立的服務角色，並選擇 Save (儲存)。



Amplify 主控台現在具有部署後端資源的權限。

預防混淆代理人

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原則會強制執行aws:SourceArn和aws:SourceAccount全域內容條件金鑰，以防止混淆副手。但是，如果您先前在帳戶中建立了Amplify-Backend Deployment角色，則可以更新角色的信任政策以新增這些條件，以防止混淆的副手。

請使用下列範例來限制帳戶中應用程式的存取權。將範例中的紅色斜體文字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指示AWS Management Console，請參閱 IAM 使用者指南中的修改角色 (主控台)。

管理應用程式效

Amplify 的預設託管架構可最佳化主機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。

對於需要更好地控制應用程序性能的高級客戶，Amplify 託管支持性能模式。效能模式可將內容保留在內容傳遞網路 (CDN) 邊緣的快取間隔，進而最佳化以提升託管效能。啟用效能模式時，主控組態或程式碼變更最多可能需要 10 分鐘的時間才能部署並可用。如需詳細資訊，請參閱 the section called “開啟效能模式”。

開啟效能模式

使用下列程序來為部署到 Amplify 主機的應用程式開啟效能模式。

啟用應用程式的效能模式

1. 登入AWS Management Console並開啟 Amplify 大控制台。
2. 選擇要啟用效能模式的應用程式。
3. 在導覽窗格中，選擇 [應用程式設定]、[一般]。
4. 在「一般」窗格中，向下捲動至「分支」區段。選取您要啟用效能模式的分支。
5. 選擇動作，啟用效能模式。
6. 在 [啟用效能模式] 對話方塊中，選擇 [啟用效能模式]。

使用標頭控制快取持續時間

HTTP `Cache-Control` 標頭`max-age`和`s-maxage`指令會影響應用程序的內容緩存持續時間。該`max-age`指令會告訴瀏覽器您希望內容在從原始伺服器重新整理之前保留在快取中的時間長度 (以秒為單位)。此`s-maxage`指示詞會覆寫`max-age`並讓您指定內容在 CDN 邊緣保留的時間長度 (以秒為單位)，再從原始伺服器重新整理內容。以 Amplify 代管的應用程式會尊重用戶端傳送的`Cache-Control`要求標頭，除非您定義的自訂標頭會覆寫這些要求標頭。

您可以手動調整指`s-maxage`令，以更好地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快取的時間長度，您可以將存留時間 (TTL) 更新`s-maxage`為超過預設 600 秒 (10 分鐘) 的值，以手動方式增加存留時間 (TTL)。

> **Note**
>
> 當應用程式的效能模式開啟時，Amplify 會增加您可以使用自訂標頭為應用程式設定的 TTL 上限，從 10 分鐘 (600 秒) 到一天 (86,400 秒)。Amplify 帽s-maxage，您可以在一天使用自定義標題設置。例如，如果您設定s-maxage為一週 (604,800 秒)，「Amplify」會使用一天的最大 TTL。

您可以在 Amplify 主控台的 [自訂標題] 區段中定義應用程式的自訂標題。如需YAML格式的範例，請參閱快取控制標頭範例。

使用記錄 Amplify API 呼叫叫叫AWS CloudTrail

AWS Amplify與整合AWS CloudTrail，這項服務可提供由使用者、角色或 Amplify 中AWS服務所採取之動作的記錄。 CloudTrail 擷取 Amplify API 呼叫叫叫叫叫叫叫叫叫做事件。擷取的呼叫包括從 Amplify 主控台進行的呼叫，以及針對 Amplify API 操作的程式碼呼叫。如果您建立追蹤記錄，就可以持續交付到 Amazon S3 S3 儲存貯體，包括 Amplify 的事件。 CloudTrail如果您不設定追蹤記錄，仍然可以透過 CloudTrail 主控台中的 Event history (事件歷史記錄) 檢視最新的事件。您可以使用 CloudTrail 收集的資訊來判斷向 Amplify 發出的請求，以及發出請求的 IP 地址、提出請求的對象、提出請求的時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱使AWS CloudTrail用者指南。

Amplify 資訊 CloudTrail

CloudTrail 您的AWS帳戶預設為啟用狀態。當 Amplify 中發生活動時，系統便會將該活動記錄至 CloudTrail 事件，並將其他AWS服務事件記錄到 Emplify 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使用指南中的檢視具有 CloudTrail 事件歷程記錄的事件。

如需您AWS帳戶中正在進行事件的記錄 (包括 Amplify 的事件)，請建立線索。線索能 CloudTrail 將日誌檔案交付到 Amazon S3 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。該追蹤會記錄來自 AWS 分割區中所有區域的事件，並將日誌檔案交付到您指定的 Amazon S3 儲存貯體。此外，您還能設定其他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，請參閱使AWS CloudTrail用者指南中的下列項目：

- 建立 AWS 帳戶的追蹤
- CloudTrail 支援的服務與整合
- 設定的 Amazon SNS SNS SNS SNS SNS SNS SNS CloudTrail
- 接收多個區域的 CloudTrail 日誌檔案及接收多個帳戶的 CloudTrail 日誌檔案

所有 Amplify 作業都會記錄 CloudTrail 並記錄在AWS Amplify主控台 API 參考、AWSAmplify 管理員 UI API 參考和 Amplify UI 產生器 API 參考資料中。例如，對`DeleteApp`和`DeleteBackendEnvironment`操作的`CreateApp`呼叫都會在 CloudTrail 日誌檔案中產生項目。

每一筆事件或日誌項目都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是否使用根或AWS Identity and Access Management (IAM) 使用者登入資料提出。
- 該請求是否使用了特定角色或聯合身分使用者的暫時安全登入資料。
- 是由另一個AWS服務提出的請求。

如需詳細資訊，請參閱《AWS CloudTrail使用者指南》中的「使用者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至您指定的 Amazon S3 儲存貯體。 CloudTrail 日誌檔案包含一個或多個日誌檔案項目。一個事件為任何來源提出的單一請求，並包含請求動作、請求的日期和時間、請求參數等資訊。 CloudTrail 日誌檔案並非依公有 API 呼叫的堆疊追蹤排序，因此不會以任何特定順序出現。

以下範例顯示的 CloudTrail 日誌項目會示範AWS Amplify主控台 API 呼叫引用ListApps操作：

```
{
    "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"
}
```

以下範例顯示的 CloudTrail 日誌項目會示範AWS Amplify AdUI API 呼叫 API 呼叫 API 呼叫引導[ListBackendJobs](#)操作：

```
{
    "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"
}
```

Amplify 中的安全性

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，這些架構是為了滿足對安全性最敏感的組織的需求而建置的。

安全是 AWS 與您之間共同承擔的責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端的安全性 — AWS 負責保護在 AWS 雲端中執行 AWS 服務的基礎架構。 AWS 還為您提供可以安全使用的服務。若要深入瞭解適用於的規範遵循計劃 AWS Amplify，請參閱合規計劃的AWS 服務範圍範圍)。
- 雲端中的安全性 — 您的責任取決於您使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件可協助您瞭解如何在使用 Amplify 時套用共同的責任模型。下列主題說明如何設定 Amplify 以符合安全性和合規性目標。您還將學習如何使用其他 AWS 服務來幫助您監控和保護 Amplify 資源。

主題

- Amplify Identity and Access Management
- Amplify 中的資料保護
- 符合性驗證 AWS Amplify
- 基礎架構安全性 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 的安全性最佳做法

Amplify Identity and Access Management

AWS Identity and Access Management (IAM) 可協助管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM 管理員控制哪些人可以通過身份驗證 (登入) 和授權 (具有權限) 來使用 Amplify 資源。IAM 是您可以使用的 AWS 服務 ，無需額外付費。

主題

- 物件

- 使用身分驗證
- 使用政策管理存取權
- Amplify 如何與 IAM 搭配使用
- Amplify 身分識別為基礎的原則範例
- AWS 受管理的政策 AWS Amplify
- 疑難排解 Amplify 身分和存取

物件

您使用 AWS Identity and Access Management (IAM) 的方式會有所不同，具體取決於您在「Amplify」中執行的工作。

服務使用者 — 如果您使用 Amplify 服務執行工作，則管理員會為您提供所需的認證和權限。當您使用更多 Amplify 功能來完成工作時，您可能需要其他權限。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法在「Amplify」中存取某個功能，請參閱疑難排解 Amplify 身分和存取。

服務管理員 — 如果您負責公司的 Amplify 資源，您可能擁有對 Amplify 的完整存取權。決定您的服務使用者應該存取哪些 Amplify 功能和資源是您的工作。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司如何將 IAM 與 Amplify 搭配使用，請參閱Amplify 如何與 IAM 搭配使用。

IAM 管理員 — 如果您是 IAM 管理員，您可能想要瞭解如何撰寫政策來管理 Amplify 存取權限的詳細資訊。若要檢視範例 Amplify 您可以在 IAM 中使用的身分型政策，請參閱。Amplify 身分識別為基礎的原則範例

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以 IAM 使用者身分或假設 IAM 角色進行驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM 身分中心) 使用者、貴公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料都是聯合身分識別的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以加密方式簽署要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的簽署 AWS API 請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。如需更多資訊，請參閱 AWS IAM Identity Center 使用者指南中的多重要素驗證和 IAM 使用者指南中的在 AWS中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱 IAM 使用者指南中的需要根使用者憑證的任務。

聯合身分

最佳作法是要求人類使用者 (包括需要系統管理員存取權的使用者) 使用與身分識別提供者的同盟，才能使用臨時登入資料進行存取 AWS 服務 。

聯合身分識別是來自企業使用者目錄的使用者、Web 身分識別提供者、Identi ty Center 目錄，或使用透過身分識別來源提供的認證進行存取 AWS 服務 的任何使用者。 AWS Directory Service同盟身分存取時 AWS 帳戶，他們會假設角色，而角色則提供臨時認證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步到自己身分識別來源中的一組使用者和群組，以便在所有應用程式 AWS 帳戶 和應用程式中使用。如需 IAM Identity Center 的相關資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱 IAM 使用者指南中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 內部具有特定許可的身分。它類似 IAM 使用者，但不與特定的人員相關聯。您可以切換角色，在中暫時擔任 IAM 角色。 AWS Management Console 您可以透過呼叫 AWS CLI 或 AWS API 作業或使用自訂 URL 來擔任角色。如需使用角色的方法更多相關資訊，請參閱 IAM 使用者指南中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 – 若要向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱 IAM 使用者指南中的為第三方身分提供者建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權 – 您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的委託人) 存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要了解跨帳戶存取權角色和資源型政策間的差異，請參閱 IAM 使用者指南中的 IAM 角色與資源類型政策的差異。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
  - 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。
  - 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

- 在 Amazon EC2 上執行的應用程式 — 您可以使用 IAM 角色來管理在 EC2 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並提供給其所有應用程式，請建立連接至執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需更多資訊，請參閱 IAM 使用者指南中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

若要了解是否要使用 IAM 角色或 IAM 使用者，請參閱 IAM 使用者指南中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以 JSON 文件的形式儲存在中。如需 JSON 政策文件結構和內容的更多相關資訊，請參閱 IAM 使用者指南中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱 IAM 使用者指南中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。若要了解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些委託人 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 和 Amazon VPC 是支援 ACL 的服務範例。 AWS WAF若要進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可範圍的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 實體許可範圍。
- 服務控制策略（SCP）— SCP 是 JSON 策略，用於指定中組織或組織單位（OU）的最大權限。AWS Organizations AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 限制成員帳戶中實體的權限，包括每個 AWS 帳戶根使用者帳戶。如需組織和 SCP 的更多相關資訊，請參閱 AWS Organizations 使用者指南中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需更多資訊，請參閱 IAM 使用者指南中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。要了解如何在涉及多個政策類型時 AWS 確定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

Amplify 如何與 IAM 搭配使用

在您使用 IAM 管理擴增的存取權限之前，請先了解哪些 IAM 功能可搭配 Amplify 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
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

若要深入瞭解 Amplify 和其他 AWS 服務如何搭配大多數 IAM 功能使用，請參閱 IAM 使用者指南中的搭配 IAM 使用的AWS 服務。

以身分識別為基礎的原則 Amplify

支援身分型政策	是

身分型政策是可以連接到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

Amplify 身分識別為基礎的原則範例

若要檢視 Amplify 大身分型原則的範例，請參閱。Amplify 身分識別為基礎的原則範例

Amplify 內部以資源為基礎的政策

支援以資源基礎的政策	否

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

若要啟用跨帳戶存取，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，作為資源型政策的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主體和資源位於不同時 AWS 帳戶，受信任帳戶中的 IAM 管理員也必須授與主體實體 (使用者或角色) 權限，才能存取資源。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 角色與資源型政策有何差異。

「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哪些內容。也就是說，哪個主體在什麼条件下可以對什麼資源執行哪些動作。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。原則動作通常與關聯的 AWS API 作業具有相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

如需「Amplify」動作的清單，請參閱服務授權參考 AWS Amplify中所定義的動作。

「Amplify」中的原則動作會在動作之前使用下列前置詞：

```
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 大身分型原則的範例，請參閱。Amplify 身分識別為基礎的原則範例

Amplify 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哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

`Resource` JSON 政策元素可指定要套用動作的物件。陳述式必須包含 `Resource` 或 `NotResource` 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

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

如需 Amplify 資源類型及其 ARN 的清單，請參閱服務授權參考 AWS Amplify中所定義的資源類型。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 大身分型原則的範例，請參閱。Amplify 身分識別為基礎的原則範例

Amplify 的原則條件金鑰

支援服務特定政策條件金鑰	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件金鑰，請參閱《IAM 使用者指南》中的AWS 全域條件內容金鑰。

如需 Amplify 條件金鑰的清單，請參閱服務授權參考 AWS Amplify中的條件金鑰。若要瞭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 大身分型原則的範例，請參閱。Amplify 身分識別為基礎的原則範例

Amplify 中的存取控制清單 (ACL)

支援 ACL	否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

具有 Amplify 功能的基於屬性的訪問控制（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，可根據屬性來定義許可。在中 AWS，這些屬性稱為標籤。您可以將標籤附加到 IAM 實體 (使用者或角色) 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

若要根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件金鑰，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的什麼是 ABAC?。如要查看含有設定 ABAC 步驟的教學課程，請參閱《IAM 使用者指南》中的使用屬性型存取控制 (ABAC)。

搭配 Amplify 使用臨時登入資料

支援臨時憑證	是

當您使用臨時憑據登錄時，某些 AWS 服務 不起作用。如需其他資訊，包括哪些 AWS 服務 與臨時登入資料搭配AWS 服務 使用，請參閱 IAM 使用者指南中的 IAM。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立暫時認證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱《IAM 使用者指南》中的切換至角色 (主控台)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

「Amplify」的轉寄存取工作階段

支援轉寄存取工作階段 (FAS)	是

當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。

「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。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。

> **Warning**
>
> 變更服務角色的權限可能會中斷 Amplify 功能。只有在 Amplify 提供指引時才編輯服務角色。

用於 Amplify 的服務連結角色

支援服務連結角色。	否

服務連結角色是一種連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需建立或管理服AWS 務連結角色的詳細資訊，請參閱 IAM 使用者指南中的與 IAM 搭配使用的服務。在表格中尋找服務，其中包含服務連結角色欄中的 Yes。選擇是連結以檢視該服務的服務連結角色文件。

Amplify 身分識別為基礎的原則範例

根據預設，使用者和角色沒有建立或修改 Amplify 資源的權限。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 來執行工作。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

如需 Amplify 定義的動作和資源類型的詳細資訊，包括每個資源類型的 ARN 格式，請參閱服務授權參考 AWS Amplify中的動作、資源和條件索引鍵。

主題

- 政策最佳實務
- 使用 Amplify 控制台
- 允許使用者檢視他們自己的許可

政策最佳實務

以身分識別為基礎的政策會決定某人是否可以建立、存取或刪除您帳戶中的 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 受管原則並邁向最低權限權限 — 若要開始授與使用者和工作負載的權限，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們在您的 AWS 帳戶. 建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低許可許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的權限。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。您也可以使用條件來授與對服務動作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需更多資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多因素身份驗證 (MFA) — 如果您的案例需要 IAM 使用者或根使用者 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。若要在呼叫 API 作業時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

使用 Amplify 控制台

若要存取 AWS Amplify 主控台，您必須擁有最少一組權限。這些權限必須允許您列出並檢視您 AWS 帳戶的. 如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

您不需要為僅對 AWS CLI 或 AWS API 進行呼叫的使用者允許最低主控台權限。反之，只需允許存取符合他們嘗試執行之 API 操作的動作就可以了。

隨著 Amplify Studio 的發布，刪除應用程序或後端需要amplify和amplifybackend權限。如果 IAM 政策僅提供amplify許可，則使用者在嘗試刪除應用程式時會收到許可錯誤。如果您是撰寫原則的管理員，請決定正確的權限，以授予需要執行刪除動作的使用者。

若要確保使用者和角色仍可使用 Amplify 主控台，請同時將 Amplify ConsoleAccess 或ReadOnly AWS 受管理的原則附加至實體。如需詳細資訊，請參閱《IAM 使用者指南》中的新增許可到使用者。

允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此原則包含在主控台上或以程式設計方式使用 AWS CLI 或 AWS API 完成此動作的權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
```

```
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS 受管理的政策 AWS Amplify

受 AWS 管理的策略是由建立和管理的獨立策略 AWS。 AWS 受管理的策略旨在為許多常見使用案例提供權限，以便您可以開始將權限指派給使用者、群組和角色。

請記住， AWS 受管理的政策可能不會為您的特定使用案例授與最低權限權限，因為這些權限可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的客戶管理政策，以便進一步減少許可。

您無法變更受 AWS 管理策略中定義的權限。如果 AWS 更新 AWS 受管理原則中定義的權限，則此更新會影響附加原則的所有主體識別 (使用者、群組和角色)。 AWS 當新的啟動或新 AWS 服務 的 API 操作可用於現有服務時，最有可能更新 AWS 受管理策略。

如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

AWS 受管政策：Ampli AdministratorAccess fy

您可將 `AdministratorAccess-Amplify` 政策連接到 IAM 身分。Amplify 也會將此原則附加至允許 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 資源的`Amplify-Backend Deployment`服務角色。IAM 會將受`AdministratorAccess-Amplify`管政策附加到`Amplify-Backend Deployment`服務角色。

此原則授予帳戶管理權限，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

此政策提供多種 AWS 服務的存取權，包括 IAM 動作。這些動作可讓具有此原則的身分識別用 AWS Identity and Access Management 來建立具有任何權限的其他身分。這允許權限提升，並且應將此原則視為與`AdministratorAccess`原則一樣強大。

此原則會授與所有資源的iam:PassRole動作權限。這是支援 Amazon Cognito 使用者集區組態的必要條件。

若要檢視此原則的權限，請參閱AWS 受管理原則參考中的 AdministratorAccess-Amplify。

AWS 受管理策略： AmplifyBackendDeployFullAccess

您可將 AmplifyBackendDeployFullAccess 政策連接到 IAM 身分。

此政策授予擴大完整存取權限，以透過部署 Amplify 後端資源 (AWS AppSync、Amazon Cognito、Amazon S3 和其他相關服務)。 AWS Cloud Development Kit (AWS CDK)權限會延遲至具有必要AdministratorAccess原則權限的 AWS CDK 角色。

若要檢視此原則的權限，請參閱AWS 受管理AmplifyBackendDeployFullAccess的策略參考中的。

Amplify AWS 受管理政策的更新

檢視有關 Amplify AWS 受管原則更新的詳細資料，因為此服務開始追蹤這些變更。如需有關此頁面變更的自動提醒，請訂閱 的文件歷史記錄 AWS Amplify 頁面的 RSS 摘要。

變更	描述	日期
AmplifyBackendDeployFullAccess – 更新現有政策	新增原cloudformation:DeleteStack 則動作以在呼叫 DeleteBranch API 時支援堆疊刪除。 新增lambda:GetFunction 原則動作以支援熱抽取功能。 新增政lambda:UpdateFunctionConfiguration 策動作以支援 Lambda 函數的更新。	2024年4月5 日
AdministratorAccess-Amplify- 更新到現有策略	新增cloudformation:TagResource 和cloudformation:UnTagResourc	2024年4月4日

變更	描述	日期
	e 權限以支援呼叫 AWS CloudFormation API。	
AmplifyBackendDeployFullAccess – 更新現有政策	新增`lambda:InvokeFunction` 原則動作以支援 AWS Cloud Development Kit (AWS CDK) 熱抽取功能。會直接 AWS CDK 呼叫 Lambda 函數以執行 Amazon S3 資產熱抽換。 新增`lambda:UpdateFunctionCode` 原則動作以支援熱抽取功能。	2024年1月02 日
AmplifyBackendDeployFullAccess – 更新現有政策	新增原則動作以支援 `UpdateApiKey` 作業。這是在退出並重新啟動沙箱而不刪除資源的情況下啟用成功的應用程序部署所必需的。	2023 年 11 月 17 日
AmplifyBackendDeployFullAccess – 更新現有政策	新增`amplify:GetBackendEnvironment` 權限以支援 Amplify 應用程式部署。	2023 年 11 月 6 日
AmplifyBackendDeployFullAccess – 新政策	Amplify 新增了具有部署 Amplify 後端資源所需的最低權限的新原則。	2023年10月8日
AdministratorAccess-Amplify-更新到現有策略	新增 Amplify 命令列介面 (CLI) 所需的`ecr:DescribeRepositories` 權限。	2023 年 6 月 1 日

變更	描述	日期
AdministratorAccess-Amplify-更新到現有策略	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。 新增原則動作以支援更新 OpenSearch 網域組態。 新增原則動作以支援從 AWS Identity and Access Management 角色移除標籤。 新增政策動作以支援從 Amazon DynamoDB 資源移除標籤。 將`cloudfront:GetCloudFrontOriginAccessIdentity` 和`cloudfront:GetCloudFrontOriginAccessIdentityConfig` 權限新增至`CLISDKCalls` 陳述式區塊，以支援 Amplify 發佈和裝載工作流程。 將`s3:PutBucketPublicAccessBlock` 權限新增至`CLIManageviaCFNPolicy` 陳述式區塊， AWS CLI 以允許在內部儲存貯體上啟用 Amazon S3 封鎖公用存取功能的 Amazon S3 安全最佳實務。	2023 年 2 月 24 日

變更	描述	日期
	將`cloudformation:DescribeStacks` 權限新增至`CLISDKCalls` 陳述式區塊，以支援在 Amplify 後端處理器中重試時擷取客戶的 AWS CloudFormation 堆疊，以避免在堆疊更新時重複執行。 將`cloudformation:ListStacks` 權限新增至`CLICloudformationPolicy` 陳述式區塊。需要此權限才能完全支援該 CloudFormation DescribeStacks 動作。	
AdministratorAccess-Amplify- 更新到現有策略	新增原則動作，以允許 Amplify 伺服器端轉譯功能將應用程式指標推送至 CloudWatch 客戶的 AWS 帳戶。	2022 年 8 月 30 日
AdministratorAccess-Amplify- 更新到現有策略	新增政策動作以封鎖對擴大部署 Amazon S3 儲存貯體的公開存取。	2022 年 4 月 27 日
AdministratorAccess-Amplify- 更新到現有策略	新增動作以允許客戶刪除其伺服器端轉譯 (SSR) 應用程式。這也允許成功刪除相應的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 處理來自現有事件來源的事件。通過這些更改， AWS Lambda 將能夠執行該UpdateEventSourceMapping操作。	2022年4月17日

變更	描述	日期
AdministratorAccess-Amplify - 更新到現有策略	新增原則動作以對所有資源啟用「Amplify UI 產生器」動作。	2021 年 12 月 2 日
AdministratorAccess-Amplify - 更新到現有策略	新增政策動作以支援使用社交身分供應商的 Amazon Cognito 身份驗證功能。 新增政策動作以支援 Lambda 層。 新增原則動作以支援「Amplify 儲存區」類別。	2021 年 11 月 8 日

變更	描述	日期
AdministratorAccess-Amplify-更新到現有策略	新增 Amazon Lex 動作以支援 Amplify 互動類別。 新增 Amazon Rekognition 動作以支援「A mplify 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 動作至支援 AWS CloudFormation StackSets。 新增 Amazon 定 Location Service 動作以支援「Amplify 地理」類別。 新增 Lambda 動作以支援 Amplify 增中的 Lambda 層。 添加 CloudWatch 日誌操作以支持 CloudWatch 事件。 新增 Amazon S3 動作以支援 Amplify 儲存類別。 新增原則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	日期
AdministratorAccess-Amplify-更新到現有策略	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的 Amazon S3 儲存貯體。 新增 IAM 權限界限動作，以支援已啟用權限界限的 Amplify 應用程式。 新增 Amazon SNS 動作以支援檢視原始電話號碼，以及檢視、建立、驗證和刪除目的地電話號碼。 Amplify 工作室：新增 Amazon Cognito AWS Lambda、IAM 和 AWS CloudFormation 政策動作，以便在擴大控制台和 Amplify 工作室中管理後端。 新增 AWS Systems Manager (SSM) 原則陳述式以管理 Amplify 環境密碼。 新增 AWS CloudFormation `ListResources` 動作以支援 Amplify 應用程式的 Lambda 層。	2021 年 7 月 28 日
Amplify 開始追蹤變更	Amplify 開始追蹤其 AWS 受管理原則的變更。	2021 年 7 月 28 日

疑難排解 Amplify 身分和存取

使用下列資訊可協助您診斷並修正使用 Amplify 和 IAM 時可能會遇到的常見問題。

主題

我沒有在「Amplify」中執行動作的授權

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 的發布，刪除應用程序或後端需要amplify和amplifybackend權限。如果管理員撰寫的 IAM 政策僅提供amplify許可，您將在嘗試刪除應用程式時收到許可錯誤。

當 mateojackson IAM 使用者嘗試使用主控台刪除虛構*example-amplify-app*資源但沒有amplifybackend:*RemoveAllBackends*權限時，就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。

我沒有授權執行 iam：PassRole

如果您收到未獲授權執行iam:PassRole動作的錯誤訊息，則必須更新您的原則，以允許您將角色傳遞給 Amplify。

有些 AWS 服務 允許您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為的 IAM 使用者`marymajor`嘗試使用主控台在 Amplify 中執行動作時，就會發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 `iam:PassRole` 動作。

如果您需要協助，請聯絡您的 AWS 系統管理員。您的管理員提供您的簽署憑證。

我想允許我 AWS 帳戶以外的人員存取我的 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：

- 若要瞭解「Amplify」是否支援這些功能，請參閱Amplify 如何與 IAM 搭配使用。
- 若要了解如何提供對您所擁有資源 AWS 帳戶 的存取權，請參閱《IAM 使用者指南》中您擁有的另一 AWS 帳戶 個 IAM 使用者提供存取權限。
- 若要了解如何將資源存取權提供給第三方 AWS 帳戶，請參閱 IAM 使用者指南中的提供第三方 AWS 帳戶 擁有的存取權。
- 若要了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 若要了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱 IAM 使用者指南中的 IAM 角色與資源型政策的差異。

Amplify 中的資料保護

AWS Amplify 符合共同責任模型 AWS 共同責任模，其中包括資料保護的法規與準則。 AWS 負責保護運行所有 AWS 服務的全球基礎設施。 AWS 保持對此基礎架構上託管的數據的控制，包括用於處理客戶內容和個人數據的安全配置控制。 AWS 作為資料控制者或資料處理者的客戶和 APN 合作夥伴負責放置在 AWS 雲端的任何個人資料。

基於資料保護目的，我們建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 登入資料並設定個別使用者。如此一來，每個使用者都只會獲得授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 資源進行通訊。
- 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 AWS 服務中的所有預設安全性控制。
- 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。這包括當您使用主控台、API 或 AWS SDK 使用 Amplify 或其他 AWS 服務時。 AWS CLI您在 Amplify 或其他服務中輸入的任何資料都可能會被拾取以包含在診斷記錄中。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。在預設情況下，Amplify 會使用 AWS KMS keys 由. AWS Key Management Service

Amplify 使用 Amazon 為您 CloudFront 的客戶提供您的應用程序服務。 CloudFront 使用針對邊緣位置存在點 (PoP) 加密的 SSD，並針對區域邊緣快取 (REC) 使用加密的 EBS 磁碟區。函數中的 CloudFront 功能代碼和配置始終以加密格式存儲在邊緣位置 POP 上的加密 SSD 上，以及其他使用的存儲位置 CloudFront。

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。默認情況下，Amplify 託管為傳輸中的數據提供加密。客戶與 Amplify 之間以及 Amplify 與其下游相依性之間的所有通訊，都會使用使用簽章第 4 版簽署程序簽署的 TLS 連線來保護。所有「Amplify 主機」端點均使用由 AWS Certificate Manager 私人憑證授權單位管理的 SHA-256 憑證。如需詳細資訊，請參閱簽章版本 4 簽署程序和什麼是 ACM PCA。

加密金鑰管理

AWS Key Management Service (KMS) 是用於建立和控制的受管服務 AWS KMS keys，也就是用來加密客戶資料的加密金鑰。 AWS Amplify 生成和管理加密密鑰，以代表客戶加密數據。沒有加密金鑰可供您管理。

符合性驗證 AWS Amplify

協力廠商稽核人員會評估其安全性與合規性， AWS Amplify 做為多個 AWS 合規計畫的一部分。這些措施包括晶片系統晶片、PCI、ISO、HIPAA、MTCS、C5、K-ISMS、ENS 高容量、生產管理系統 (CSF) 和金融管理局 (FINMA)。

若要瞭解 AWS 服務 是否屬於特定規範遵循方案的範圍內，請參閱AWS 服務 遵循規範計劃方案中的，並選擇您感興趣的合規方案。如需一般資訊，請參閱AWS 規範計劃AWS。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載中的報告中的 AWS Artifact。

您在使用時的合規責任取決 AWS 服務 於資料的敏感性、公司的合規目標以及適用的法律和法規。AWS 提供下列資源以協助遵循法規：

- 安全性與合規性快速入門指南 — 這些部署指南討論架構考量，並提供部署以安全性和合規性 AWS 為重點的基準環境的步驟。
- 在 Amazon Web Services 上架構 HIPAA 安全性與合規性 — 本白皮書說明公司如何使用建立符合 HIPAA 資格的應 AWS 用程式。

> **Note**
>
> 並非所有人 AWS 服務 都符合 HIPAA 資格。如需詳細資訊，請參閱 HIPAA 資格服務參照。

- AWS 合規資源AWS — 此工作簿和指南集合可能適用於您的產業和所在地。
- AWS 客戶合規指南 — 透過合規的角度瞭解共同的責任模式。這份指南總結了在多個架構 (包括美國國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO)) 中，保 AWS 服務 護指引並對應至安全控制的最佳實務。
- 使用AWS Config 開發人員指南中的規則評估資源 — 此 AWS Config 服務會評估您的資源組態符合內部實務、產業準則和法規的程度。

- AWS Security Hub— 這 AWS 服務 提供了內部安全狀態的全面視圖 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱 Security Hub controls reference。
- AWS Audit Manager— 這 AWS 服務 有助於您持續稽核您的 AWS 使用情況，以簡化您管理風險的方式，以及遵守法規和業界標準的方式。

基礎架構安全性 AWS Amplify

作為託管服務， AWS Amplify 受到 AWS 全球網絡安全的保護。有關 AWS 安全服務以及如何 AWS 保護基礎結構的詳細資訊，請參閱AWS 雲端安全 若要使用基礎架構安全性的最佳做法來設計您的 AWS 環境，請參閱安全性支柱架構良 AWS 好的架構中的基礎結構保護。

您可以使用 AWS 已發佈的 API 呼叫，透過網路存取「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。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

Amplify 中的安全事件記錄和監控

監控是維持 Amplify 和其他 AWS 解決方案的可靠性、可用性和效能的重要組成部分。 AWS 提供下列監控工具來觀看 Amplify、在發生錯誤時報告，並在適當時採取自動動作：

- Amazon 會即時 CloudWatch監控您的 AWS 資源和執行的應用程式 AWS。您可以收集和追蹤指標、建立自訂儀表板，以及設定警示以通知您或在特定量度達到您指定的閾值時採取動作。例如，您可以 CloudWatch 追蹤 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體的 CPU 使用率或其他指標，並在需要時自動啟動新執行個體。如需搭配 Amplify 使用 CloudWatch 量度和警示的詳細資訊，請參閱監控。
- Amazon CloudWatch 日誌可讓您從 Amazon EC2 執行個體和其他來源監控 AWS CloudTrail、存放和存取日誌檔。 CloudWatch 記錄檔可以監控記錄檔中的資訊，並在符合特定臨界值時通知您。您也可以將日誌資料存檔在高耐用性的儲存空間。如需詳細資訊，請參閱 Amazon CloudWatch 日誌使用者指南。

- AWS CloudTrail擷取您帳戶或代表您 AWS 帳戶發出的 API 呼叫和相關事件，並將日誌檔交付到您指定的 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 位址，以及呼叫發生的時間。如需詳細資訊，請參閱 使用記錄 Amplify API 呼叫叫叫AWS CloudTrail。
- Amazon EventBridge 是一種無伺服器事件匯流排服務，可讓您輕鬆地將應用程式與各種來源的資料連接起來。 EventBridge 從您自己的應用程式、S oftware-as-a 服務 (SaaS) 應用程式和 AWS 服務提供即時資料串流，並將資料路由到目標 (例如 AWS Lambda. 這可讓您監視發生在服務中的事件，並建置事件驅動的架構。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問題，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的副問題。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。

我們建議在資源策略中使用aws:SourceArn和aws:SourceAccount全域條件前後關聯索引鍵，以限制將其他服務 AWS Amplify 提供給資源的權限。如果同時使用全域條件內容索引鍵，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*:*。

下列範例顯示角色信任原則，您可以套用來限制帳戶中任何 Amplify 應用程式的存取權，並防止混淆的副問題。若要使用此原則，請以您自己的資訊取代範例原則中的紅色斜體文字。

```
{
  "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"
        }
      }
    }
}
```

下列範例顯示角色信任原則，您可以套用來限制帳戶中指定 Amplify 應用程式的存取權，並防止混淆的副問題。若要使用此原則，請以您自己的資訊取代範例原則中的紅色斜體文字。

```
{
  "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 的安全性最佳做法

Amplify 提供許多安全性功能，可在您開發和實作自己的安全性原則時考慮。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。

在 Amplify 預設網域中使用 Cookie

當您使用 Amplify 部署 Web 應用程式時，Amplify 會在預設`amplifyapp.com`網域上為您主控該應用程式。您可以在格式為的 URL 上查看您的應用程序`https://branch-name.d1m7bkiki6tdw1.amplifyapp.com`。

為了增強您的 Amplify 應用程式的安全性，請在公用尾碼清單 (PSL) 中註冊了 amplifyapp.com 網域。為了進一步的安全性，如果您需要在 Amplify 應用程式的預設網域名稱中設定敏感性 Cookie，建議您使用__Host-前置詞的 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

以下是主機的 AWS Amplify 服務配額。服務配額 (先前稱為限制) 是您的服務資源或作業的最大數目 AWS 帳戶。

新功 AWS 帳戶 能減少了應用程式和並行工作配額。 AWS 根據您的使用情況自動提高這些配額。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	描述
應用程式	每個受支援的區域：25	是	您可以在目前區域的這個帳戶中，在 AWS Amplify 主控台中建立的應用程式數目上限。
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)。組建成品會在組建之後由 AWS Amplify 主控台部署。
快取成品大小	所有受支援的區域：5 GB	否	快取加工品的大小上限 (以 GB 為單位)。
並行工作	每個受支援的區域：5	是	在目前「區域」中，您可以在此帳戶中建立的並行工作數目上限。

名稱	預設	可調整	描述
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。
環境快取加工品大小	所有受支援的區域：5 GB	否	環境快取人工因素的大小上限 (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)。
每小時最多建立應用程式	每個受支援的區域：25	否	在目前區域中，此帳戶每小時可在 AWS Amplify 主控台中建立的應用程式數目上限。
每秒請求令牌	每個受支援的區域：20,000	是	應用程式每秒要求權杖的最大數目。Amplify Hosting 根據它們消耗的資源量（處理時間和數據傳輸）將令牌分配給請求。
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。

如需有關「AWS 一般參考Amplify 服務配額」的詳細AWS Amplify 資訊，請參閱.

疑難排解 Amplify 託管

如果您在使用 Amplify 主機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

下列資訊可協助您疑難排解 Amplify 主機的一般問題。

主題

HTTP 429 狀態碼（請求過多）

Amplify 會根據傳入要求耗用的處理時間和資料傳輸，控制每秒要求 (RPS) 的數量。如果您的應用程式傳回 HTTP 429 狀態碼，傳入的要求就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 的REQUEST_TOKENS_PER_SECOND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，建議您將應用程式最佳化，以縮短要求持續時間和資料傳輸，以增加應用程式的 RPS。例如，與延遲超過 200 毫秒的頁面相比，使用相同 20,000 個字符，在 100 毫秒內回應的高度最佳化 SSR 頁面可支援更高的 RPS。

同樣地，傳回 1 MB 回應大小的應用程式會比傳回 250 KB 回應大小的應用程式消耗更多的 Token。

我們也建議您透過設定 CloudFront 快取控制標頭來利用 Amazon 快取，將指定回應保留在快取中的時間最大化。從 CloudFront 快取提供的要求不會計入速率限制。每個 CloudFront 分發最多每秒可處理 250,000 個請求，使您可以使用緩存擴展應用程序非常高。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。

Amazon 2023 建置映像問題疑難排解

下列資訊可協助您對 Amazon Linux 2023 (AL2023) 建置映像的問題進行疑難排解。

主題

如何使用 Python 運行時運行 Amplify 函數

現在，Amplify 託管在您部署新應用程式時，預設情況下會使用 Amazon Linux 2023 組建映像檔。AL2023 預先安裝了 Python 版本 3.8，3.9，3.10 和 3.11 版本。

為了與 Amazon Linux 2 映像的向後兼容性，AL2023 構建映像具有預先安裝的舊版 Python 的符號鏈接。因此，您不再需要使用 Amplify Hosting GitHub 常見問題解答中提供的說明更新應用程序構建規範中的構建命令。

默認情況下，Python 版本 3.10 在全局使用。要使用特定的 Python 版本構建函數，請在應用程序的構建規範文件中運行以下命令。

```
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
```

如何運行需要超級用戶或 root 權限的命令

如果您使用的是 Amazon Linux 2023 建置映像，並且在執行需要超級使用者或 root 權限的系統命令時收到錯誤訊息，您必須使用 Linux sudo 命令執行這些命令。例如，如果執行時發生錯誤`yum install -y gcc`，請使用`sudo yum install -y gcc`.

Amazon Linux 2 構建映像使用了根用戶，但擴增的 AL2023 映像與自定義用`amplify`戶一起運行您的代碼。Amplify 會授與此使用者使用 Linux sudo 命令執行命令的權限。最佳作法是用於需要超級使`sudo`用者權限的指令。

疑難排解自訂域

如果您在將自訂網域連接到 Amplify 應用程式時遇到問題，請參閱以排解自訂網域取得說明。

疑難排解伺服器端轉

如果您在將 SSR 應用程式部署到 Amplify 時遇到問題，請參閱以SSR 部署的疑難取得說明。

AWS Amplify託管參考

使用本節中的主題尋找詳細的參考資料AWS Amplify。

主題

AWS CloudFormation 支援

使用AWS CloudFormation用於佈建 Amplify 資源的範本，實現可重複且可靠的 Web 應用程式部署。AWS CloudFormation提供一種通用語言，供您描述和佈建雲端環境中的所有基礎架構資源，並簡化多個部署的作業AWS只需單擊幾下即可帳戶和/或地區。

對於放大託管，請參閱放大CloudFormation文件。對於放大工作室，請參閱放大 UI 生成器CloudFormation文件。

AWS Command Line Interface 支援

使用AWS Command Line Interface以程式設計方式從命令列建立擴大應用程式。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標記放大資源。如需詳細資訊，請參閱AWS CLI標籤資源文檔。

擴大託管服務 API

此參考提供「擴大主機 API」動作和資料類型的說明。如需詳細資訊，請參閱擴大 API 參考文件。

的文件歷史記錄 AWS Amplify

下表說明自上次發行版本以來文件的重要變更 AWS Amplify。

- 最新文件更新：2024 年 4 月 5 日

變更	描述	日期
更新了受管理的策略	已更新主AWS 受管理的政策 AWS Amplify題以說明 Amplify 之 AWS 受管理原則的最新變更。	2024年4月5 日
更新了受管理的策略	已更新主AWS 受管理的政策 AWS Amplify題以說明 Amplify 之 AWS 受管理原則的最新變更。	2024年4月4日
新增疑難排解章	已新增本疑難排解 Amplify 託管章，說明如何修正部署至 Amplify 主機的應用程式時遇到的問題。	2024年4月2日
自訂 SSL/TLS 憑證的新支援	將使用 SSL/TLS 憑證主題新增至設定自訂網域本章，說明將應用程式連接至自訂網域時對自訂 SSL/TLS 憑證的 Amplify 大支援。	2024年2月20日
更新了受管理的策略	已更新主AWS 受管理的政策 AWS Amplify題以說明 Amplify 之 AWS 受管理原則的最新變更。	2024年1月2日
對 SSR 框架的新支持	已新增主Amplify 對 SSR 框架的支援題，以說明 Amplify 對任何具有開放原始碼介面卡之	2023 年 11 月 19 日

變更	描述	日期
	JavaScript 型 SSR 架構的支援。	
對圖像優化功能推出的新支持	已新增主SSR 應用程式的影像優化題，說明伺服器端轉譯應用程式影像最佳化的內建支援。	2023 年 11 月 19 日
更新了受管理的策略	已更新主AWS 受管理的政策 AWS Amplify題以說明 Amplify 之 AWS 受管理原則的最新變更。	2023 年 11 月 17 日
更新了受管理的策略	已更新主AWS 受管理的政策 AWS Amplify題以說明 Amplify 之 AWS 受管理原則的最新變更。	2023 年 11 月 6 日
新增萬用字元子網域主題	已新增主萬用字元子網域題以說明對自訂網域上萬用字元子網域的支援。	2023 年 11 月 6 日
新的 受管政策	已更新主AWS 受管理的政策 AWS Amplify題以說明 Amplify 的新 AmplifyBackendDeployFullAccess AWS 受管理原則。	2023年10月8日
對壟斷框架功能推出的新支持	更新壟斷構建設置主題以說明在使用 npm 工作區、pnpm 工作區、Yarn 工作區、NX 和植物園建立的專賣中部署應用程式的支援。	2023 年 6 月 19 日

變更	描述	日期
更新了受管理的策略	已更新主AWS 受管理的政策 AWS Amplify題以說明 Amplify 之 AWS 受管理原則的最新變更。	2023 年 6 月 1 日
更新了受管理的策略	已更新主AWS 受管理的政策 AWS Amplify題以說明 Amplify 之 AWS 受管理原則的最新變更。	2023 年 2 月 24 日
更新服務器端渲染章	更新此使用 Amplify 主機部署伺服器端轉譯應用程章節以說明對於 Next.js 版本 12 和 13 的支援的最新變更。	2022 年 11 月 17 日
更新了受管理的策略	已更新主AWS 受管理的政策 AWS Amplify題以說明 Amplify 之 AWS 受管理原則的最新變更。	2022 年 8 月 30 日
更新了受管理的策略	已更新主開始使用完整堆疊持續部署題，以說明如何使用 Amplify Studio 部署後端。	2022 年 8 月 23 日
更新了受管理的策略	已更新主AWS 受管理的政策 AWS Amplify題以說明 Amplify 之 AWS 受管理原則的最新變更。	2022 年 4 月 27 日
更新了受管理的策略	已更新主AWS 受管理的政策 AWS Amplify題以說明 Amplify 之 AWS 受管理原則的最新變更。	2022年4月17日

變更	描述	日期
新 GitHub 應用程式功能啟動	已新增主設定 Amplify GitHub 庫的存取題，說明授權 Amplify 存取存放庫的 GitHub 新 GitHub 應用程式。	2022 年 4 月 5 日
新的 Amplify 工作室功能推出	已更新歡迎來到AWS Amplify 託管主題以說明 Amplify Studio 的更新，該更新可提供視覺化設計工具來建立可連線至後端資料的 UI 元件。	2021 年 12 月 2 日
更新了受管理的策略	已更新主AWS 受管理的政策 AWS Amplify題，以說明最近對擴增的 AWS 受管理原則所做的變更，以支援 Amplify Studio。	2021 年 12 月 2 日
更新了受管理的策略	已更新主AWS 受管理的政策 AWS Amplify題以說明 Amplify 之 AWS 受管理原則的最新變更。	2021 年 11 月 8 日
更新了受管理的策略	已更新主AWS 受管理的政策 AWS Amplify題以說明 Amplify 之 AWS 受管理原則的最新變更。	2021 年 9 月 27 日
新的受管理策略主題	已新增主AWS 受管理的政策 AWS Amplify題，說明 Amplify 的 AWS 受管理原則，以及這些原則的最近變更。	2021 年 7 月 28 日
更新了服務器端渲染章節	更新了本使用 Amplify 主機部署伺服器端轉譯應用程章以說明對 Next.js 版本 10 的新支援。 x. X 和 Next.js 版本 11.	2021 年 7 月 22 日

變更	描述	日期
更新了配置構建設置章節	已新增壟斷構建設置主題，說明在使用 Amplify 部署 monorepo 應用程式時，如何設定建置設定和新AMPLIFY_MONOREPO_APP_ROOT 環境變數。	2021 年 7 月 20 日
更新功能分支部署章節	已新增主自動生成放大配置的構建時間題，說明如何在建置階段自動產生aws-exports.js 檔案。已新增說明如何啟用條件式後端組建的條件式後端建置主題。已新增跨應用程式使用擴大後端主題，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
更新安全章節	新增主Amplify 中的資料保護題以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 功能推出全新支援	已新增此使用 Amplify 主機部署伺服器端轉譯應用程章節，說明使用伺服器端轉譯 (SSR) 並使用 Next.js 建立之 Web 應用程式的 Amplify 大支援。	2021 年 5 月 18 日

變更	描述	日期
新增安全性章節	已新增Amplify 中的安全性本章，說明如何在使用 Amplify 時套用共用責任模型，以及如何設定 Amplify 以符合安全性和合規性目標。	2021 年 3 月 26 日
更新的自訂組建主題	已更新自訂組建映像和即時套件更新主題，以說明如何設定在 Amazon Elastic 容器登錄公用中託管的自訂建置映像。	2021 年 3 月 12 日
更新的監控主題	更新了監控主題，以說明如何存取 Amazon CloudWatch 指標資料和設定警示。	2021 年 2 月 2 日
新增 CloudTrail 記錄主題	新增使用 AWS CloudTrail主題記錄 Amplify API 呼叫，以說明如何 AWS CloudTrail 擷取和記錄主 AWS Amplify 控台 API 參考和 AWS Amplify 管理員 UI API 參考的所有 API 動作。	2021 年 2 月 2 日
新的管理 UI 功能推出	已更新歡迎來到AWS Amplify託管主題以說明新的 Admin UI，該 UI 為前端 Web 和行動開發人員提供視覺化介面，以便在. AWS Management Console	2020 年 12 月 1 日
推出全新效能模式功能	更新了「管理應用程式效能」主題，以說明如何啟用效能模式以最佳化以加快託管效能。	2020 年 11 月 4 日

變更	描述	日期
更新了自定義標題主題	已更新自訂標頭主題，以說明如何使用主控台或編輯 YML 檔案來定義 Amplify 應用程式的自訂標題。	2020 年 10 月 28 日
新的 auto 子域功能啟動	新增為 Route 53 自訂網域設定自動子網域主題，以說明如何針對連接到 Amazon Route 53 自訂網域的應用程式使用以模式為基礎的功能分支部署。已新增具有子網域的 Web 預覽存取權主題，以說明如何從提取要求設定 Web 預覽，以便透過子網域存取。	二零二零年六月廿日
新增通知主題	已新增「通知」主題，說明如何為 Amplify 應用程式設定電子郵件通知，以在組建成功或失敗時提醒利益相關者或團隊成員。	二零二零年六月廿日
更新了自訂網域主題	更新了設定自訂網域主題，GoDaddy以改進在 Amazon 路線 53 和 Google 域中添加自定義域的程序。此更新也包含用於設定自訂網域的新疑難排解資訊。	2020 年 5 月 12 日
AWS Amplify 釋放	此版本引入「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。