用户指南

AWS Amplify 托管

AWS Amplify 托管: 用户指南

Table of Contents

欢迎来到 AWS Amplify 托管

Amplify Hosting 提供了基于 Git 的工作流，用于托管持续部署的全栈无服务器 Web 应用程序。Amplify 会将您的应用程序部署到 AWS 全球内容分发网络 (CDN)。这份用户指南提供开始使用 Amplify Hosting 所需的信息。

支持的框架

Amplify Hosting 支持许多常用 SSR 框架、单页面应用程序（SPA）框架和静态网站生成器，包括以下各项。

SSR 框架

- Next.js
- Nuxt
- 将 Astro 与社区适配器结合使用
- 将 SvelteKit 与社区适配器结合使用
- 任何带自定义适配器的 SSR 框架

SPA 框架

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

静态网站生成器

- Eleventy
- Gatsby
- Hugo
- Jekyll
- VuePress

Amplify Hosting 功能

功能分支

通过连接新分支来管理前端和后端的生产环境和暂存环境。

自定义域

将您的应用程序连接到自定义域。

拉取请求预览

在代码审核期间预览更改。

End-to-end 测试

通过 end-to-end测试提高应用质量。

受密码保护的分支

密码可保护 Web 应用程序，因此您可以处理新功能而不使它们可公开访问。

重定向和重写

设置重写和重定向，以维护 SEO 排名并根据您的客户端应用程序要求路由流量。

原子部署

通过确保仅在整个部署完成后更新您的 Web 应用程序，原子部署消除了维护时段。这消除了文件无法正确上传的情况。

开始使用 Amplify Hosting

要开始使用 Amplify Hosting，请参阅 开始将应用程序部署到 Amplify Hosting 教程。完成本教程后，您将知道如何连接 Git 存储库（GitHub、、或 AWS CodeCommit）中的 Web 应用程序 BitBucket GitLab，并通过持续部署将其部署到 Amplify Hosting。

构建后端

AWS Amplify Gen 2 引入了一种 TypeScript基于代码优先的开发者体验，用于定义后端。如需了解如何使用 Amplify Gen 2 构建后端并将其连接到应用程序，请参阅《Amplify 文档》中的构建和连接后端。

如需进一步了解 Amplify Gen 2 的代码优先方法，请参阅“AWS Workshop Studio”网站上的 Amplify Gen 2 讲习会。在这份综合教程中，您将使用 React 和 Next.js 构建无服务器应用程序，并学习如何使用 Amplify Gen 2 数据和身份验证库以及 Amplify UI 为应用程序添加功能。

如果您正在寻找有关使用 CLI 和 Amplify Studio 为第 1 代应用程序构建后端的文档，请参阅第 1 代 Amplify 文档中的构建和连接后端。

Amplify Hosting 定价

AWS Amplify 仅向您收取用量费用。有关更多信息，请参阅 AWS Amplify 定价。

开始将应用程序部署到 Amplify Hosting

为帮助您了解 Amplify Hosting 的工作原理，以下教程将为您详细介绍如何构建和部署使用 Amplify 支持的常用 SSR 框架创建的应用程序。

教程

- 将 Next.js 应用程序部署到 Amplify Hosting
- 将 Nuxt.js 应用程序部署到 Amplify Hosting
- 将 Astro.js 应用程序部署到 Amplify Hosting
- 将 SvelteKit 应用程序部署到 Amplify 托管

将 Next.js 应用程序部署到 Amplify Hosting

本教程将为您详细介绍从 Git 存储库构建和部署 Next.js 应用程序的过程。

开始本教程之前，请完成以下先决条件。

注册获取 AWS 账户

如果您还不是 AWS 客户，则需要按照在线说明进行创建。 AWS 账户注册后，您就可以访问 Amplify 和其他可与您的应用程序配合使用的 AWS 服务。

创建 应用程序

按照 Next.js 文档中的create-next-app说明，创建用于本教程的基本 Next.js 应用程序。

创建 Git 存储库

Amplify 支持 GitHub Bitbucket 和 GitLab。 AWS CodeCommit将 `create-next-app` 应用程序推送到 Git 存储库。

步骤 1：连接 Git 存储库

在此步骤中，您将 Git 存储库中的 Next.js 应用程序连接到 Amplify Hosting。

将应用程序连接到 Git 存储库

1. 打开 Amplify 控制台。

2. 如果您要在当前区域中部署第一个应用程序，则默认情况下，您将从 AWS Amplify 服务页面开始。

   选择页面顶部的部署应用程序。

3. 在开始使用 Amplify 进行构建页面中选择您的 Git 存储库提供商，然后选择下一步。

   对于 GitHub 存储库，Amplify 使用 GitHub 应用程序功能来授权 Amplify 访问权限。有关安装和授权 GitHub应用程序的更多信息，请参阅设置 Amplify 对仓库的访问权限 GitHub。

   > **Note**
   >
   > 在你使用 Bitbucket、 GitLab、或 Amplify 控制台授权后， AWS CodeCommit Amplify 会从存储库提供者那里获取访问令牌，但它不会将该令牌存储在服务器上。 AWS Amplify 仅使用安装在特定存储库中的部署密钥访问存储库。

4. 在添加存储库分支页面上，执行以下操作：

   a. 选择需要连接的存储库的名称。

   b. 选择需要连接的存储库分支的名称。

   c. 选择下一步。

步骤 2：确认构建设置

Amplify 将自动检测要为您部署的分支运行的构建命令序列。在此步骤中，您将审核并确认构建设置。

确认应用程序构建设置的方法

1. 在应用程序设置页面中找到构建设置部分。

   验证前端构建命令和构建输出目录是否正确。对于此 Next.js 示例应用程序，构建输出目录设置为 `.next`。

2. 添加服务角色的过程会有所不同，具体取决于您要创建新角色还是使用现有角色。

   - 若要创建新角色：
     - 请选择创建和使用新的服务角色。
   - 若要使用现有角色：

     a. 选择使用现有角色。

b. 在服务角色列表中选择要使用的角色。

3. 选择下一步。

第 3 步：部署应用程序

在此步骤中，您将应用程序部署到 AWS 全球内容分发网络 (CDN)。

保存和部署应用程序

1. 在审核页面上，确认您的存储库详细信息和应用程序设置正确无误。
2. 选择保存并部署。您的前端构建通常需要 1 到 2 分钟，但可能因应用程序大小而异。
3. 完成部署时，您可以使用 `amplifyapp.com` 默认域链接查看应用程序。

> **Note**
>
> 为增强 Amplify 应用程序的安全性，已将 amplifyapp.com 域注册到公共后缀列表 (PSL)。为进一步增强安全性，如果您需要在 Amplify 应用程序的默认域名中设置敏感 Cookie，我们建议您使用带 `__Host-` 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造 (CSRF) 攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

步骤 4：（可选）清理资源

如果不再需要为本教程部署的应用程序，您可以将其删除。此步骤有助于确保不会为未使用的资源付费。

删除应用程序的方法

1. 在导航窗格中的应用程序设置菜单中，选择常规设置。
2. 在常规设置页面上选择删除应用程序。
3. 在确认窗口中，输入 **`delete`**。然后选择删除应用程序。

为应用程序添加功能

现在，您已经将一个应用程序部署到了 Amplify，您可以探索托管应用程序可用的以下功能。

环境变量

应用程序通常在运行时需要配置信息。这些配置可以是数据库连接详细信息、API 密钥或参数。环境变量提供了在构建时公开这些配置的方法。有关更多信息，请参阅 Environment variables。

自定义域

在本教程中，Amplify 将您的应用托管在默认 amplifyapp.com 域上，URL 示例：https://branch-name.d1m7bkiki6tdw1.amplifyapp.com。当您将应用程序连接到自定义域时，用户会看到您的应用程序托管在自定义网址上，例如 https://www.example.com。有关更多信息，请参阅设置自定义域。

拉取请求预览

Web 拉取请求预览为团队提供了一种在将代码合并到生产或集成分支之前预览拉取请求 (PRs) 中更改的方法。有关更多信息，请参阅拉取请求的 Web 预览。

管理多个环境

要了解 Amplify 如何使用功能分支和 GitFlow 工作流程来支持多个部署，请参阅功能分支部署和团队工作流程。

将 Nuxt.js 应用程序部署到 Amplify Hosting

按照以下说明，将 Nuxt.js 应用程序部署到 Amplify Hosting。Nuxt 已使用 Nitro 服务器实现了一个预设适配器。因此，您不需要任何额外的配置即可部署 Nuxt 项目。

将 Nuxt 应用程序部署到 Amplify Hosting 的方法

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 在所有应用程序页面中，选择创建新应用程序。
3. 在开始使用 Amplify 进行构建页面中选择您的 Git 存储库提供商，然后选择下一步。
4. 在添加存储库分支页面上，执行以下操作：
   a. 选择需要连接的存储库的名称。
   b. 选择需要连接的存储库分支的名称。
   c. 选择下一步。
5. 如果您希望 Amplify 能够将应用程序日志传输到 Amazon L CloudWatch ogs，则必须在控制台中明确启用此功能。打开高级设置部分，随后在服务器端渲染（SSR）部署部分中选择启用 SSR 应用程序日志。

6. 选择下一步。
7. 在查看页面上，选择保存并部署。

将 Astro.js 应用程序部署到 Amplify Hosting

按照以下说明，将 Astro.js 应用程序部署到 Amplify Hosting。您可以使用现有应用程序，也可以使用 Astro 提供的正式示例之一创建新手入门应用程序。要创建新手入门应用程序，请参阅《Astro 文档》中的使用主题或新手入门模板。

要将带 SSR 的 Astro 站点部署到 Amplify Hosting，您必须向应用程序添加适配器。我们不维护 Amplify 拥有、适用于 Astro 框架的适配器。本教程使用社区成员创建的 `astro-aws-amplify` 适配器。这个适配器可在 github 上找到。 com/alexnguyennz/astro-aws-amplify 在网站上放大。 GitHub AWS 不维护此适配器。

将 Astro 应用程序部署到 Amplify Hosting 的方法

1. 在本地计算机上，导航到需要部署的 Astro 应用程序。
2. 若要安装适配器，请打开一个终端窗口，然后运行以下命令。此示例使用 github 上提供的社区适配器。 com/alexnguyennz/astro-aws-amplify。你可以*astro-aws-amplify*用你正在使用的适配器的名称替换。

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

3. 在 Astro 应用程序的项目文件夹中，打开 `astro.config.mjs` 文件。更新文件以添加适配器。文件应该呈现以下状态。

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

import sitemap from '@astrojs/sitemap';

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

4. 提交更改，并将项目推送到 Git 存储库。

   您现在已准备好将 Astro 应用程序部署到 Amplify。

5. 登录 AWS 管理控制台 并打开 Amplify 控制台。
6. 在所有应用程序页面中，选择创建新应用程序。
7. 在开始使用 Amplify 进行构建页面中选择您的 Git 存储库提供商，然后选择下一步。
8. 在添加存储库分支页面上，执行以下操作：

   a. 选择需要连接的存储库的名称。

   b. 选择需要连接的存储库分支的名称。

   c. 选择下一步。

9. 在应用程序设置页面中找到构建设置部分。对于构建输出目录，请输入 **.amplify-hosting**。
10. 您还必须在构建规范中更新应用程序的前端构建命令。要打开构建规范，请选择编辑 YML 文件。
11. 在 amplify.yml 文件中，找到前端构建命令部分。输入 **mv node_modules ./.amplify-hosting/compute/default**

    您的构建设置文件应如下所示：

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

12. 选择保存。

13. 如果您希望 Amplify 能够将应用程序日志传输到 Amazon L CloudWatch ogs，则必须在控制台中明确启用此功能。打开高级设置部分，随后在服务器端渲染（SSR）部署部分中选择启用 SSR 应用程序日志。
14. 选择下一步。
15. 在查看页面上，选择保存并部署。

将 SvelteKit 应用程序部署到 Amplify 托管

按照以下说明将 SvelteKit 应用程序部署到 Amplify Hosting。您可以使用自己的应用程序，也可以创建新手入门应用程序。有关更多信息，请参阅SvelteKit 文档中的创建项目。

要将带有 SSR 的 SvelteKit 应用程序部署到 Amplify Hosting，您必须在项目中添加适配器。我们不为该框架维护 Amplify 自有的适配器。 SvelteKit 在此示例中，我们使用的是社区成员创建的 `amplify-adapter`。该适配器可在 github 上找到。 com/gzimbron/amplify- GitHub 网站上的适配器。 AWS 不维护此适配器。

将 SvelteKit 应用程序部署到 Amplify Hosting

1. 在您的本地计算机上，导航到要部署的 SvelteKit 应用程序。
2. 若要安装适配器，请打开一个终端窗口，然后运行以下命令。此示例使用 github 上提供的社区适配器。 com/gzimbron/amplify-适配器。如果您使用的是其他社区适配器，请*`amplify-adapter`*替换为适配器的名称。

```
npm install amplify-adapter
```

3. 在 SvelteKit 应用程序的项目文件夹中，打开该`svelte.config.js`文件。编辑文件以使用`amplify-adapter`或*`'amplify-adapter'`*替换为适配器的名称。文件应该呈现以下状态。

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


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

        kit: {
```

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

export default config;
```

4. 提交更改，并将应用程序推送到 Git 存储库。
5. 现在，您可以将您的 SvelteKit 应用程序部署到 Amplify 了。

   登录 AWS 管理控制台 并打开 Amplify 控制台。
6. 在所有应用程序页面中，选择创建新应用程序。
7. 在开始使用 Amplify 进行构建页面中选择您的 Git 存储库提供商，然后选择下一步。
8. 在添加存储库分支页面上，执行以下操作：

   a. 选择需要连接的存储库的名称。

   b. 选择需要连接的存储库分支的名称。

   c. 选择下一步。
9. 在应用程序设置页面中找到构建设置部分。对于构建输出目录，请输入 **build**。
10. 您还必须在构建规范中更新应用程序的前端构建命令。要打开构建规范，请选择编辑 YML 文件。
11. 在 amplify.yml 文件中，找到前端构建命令部分。输入 **- cd build/compute/default/** 和 **- npm i --production**。

    您的构建设置文件应如下所示：

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

```
                - 'cd build/compute/default/'
                - 'npm i --production'

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

12. 选择保存。
13. 如果您希望 Amplify 能够将应用程序日志传输到 Amazon L CloudWatch ogs，则必须在控制台中明确启用此功能。打开高级设置部分，随后在服务器端渲染（SSR）部署部分中选择启用 SSR 应用程序日志。
14. 选择下一步。
15. 在查看页面上，选择保存并部署。

使用 Amplify Hosting 部署在服务器端渲染的应用程序

您可以使用部署和托管使用 AWS Amplify 服务器端渲染 (SSR) 的 Web 应用程序。Amplify Hosting 会自动检测使用 Next.js 框架创建的应用程序，您无需在 AWS 管理控制台中执行任何手动配置。

Amplify 还支持任何基于 JavaScript 的 SSR 框架，其开源构建适配器可将应用程序的构建输出转换为 Amplify Hosting 期望的目录结构。例如，您可以通过安装可用的适配器来部署使用 Nuxt、Astro 和 SvelteKit框架创建的应用程序。

高级用户可使用部署规范来创建构建适配器或配置构建后脚本。

您只需通过最低限度的配置即可将以下框架部署到 Amplify Hosting。

Next.js

- Amplify 无需适配器即可支持 Next.js 15 个应用程序。要开始使用，请参阅Amplify 对 Next.js 的支持。

Nuxt.js

- Amplify 支持利用预设适配器进行 Nuxt.js 应用程序部署。要开始使用，请参阅Amplify 对 Nuxt.js 的支持。

Astro.js

- Amplify 支持利用社区适配器进行 Astro.js 应用程序部署。要开始使用，请参阅Amplify 对 Astro.js 的支持。

SvelteKit

- Amplify 支持使用社区适配器的 SvelteKit 应用程序部署。要开始使用，请参阅Amplify 对以下各项的支持 SvelteKit。

开源适配器

- 使用开源适配器 – 如需使用上述列表中未列出的适配器的说明，请参阅为任意 SSR 框架使用开源适配器。
- 构建框架适配器 – 如果框架作者想要集成框架提供的功能，则可使用 Amplify Hosting 部署规范来配置构建输出，使其符合 Amplify 期望的结构。有关更多信息，请参阅使用 Amplify Hosting 部署规范配置构建输出。
- 配置构建后脚本 – 您可以根据特定场景的需要，使用 Amplify Hosting 部署规范来操作构建输出。有关更多信息，请参阅使用 Amplify Hosting 部署规范配置构建输出。有关示例，请参阅使用部署清单部署 Express 服务器。

主题

Amplify 对 Next.js 的支持

Amplify 支持部署和托管使用 Next.js 创建的服务器端渲染（SSR）Web 应用程序。Next.js 是一个用于开发 SPAs 的 React 框架 JavaScript。您可以部署使用 Next.js 15 及更低版本的 Next.js 构建，且具有映像优化和中间件等功能的应用程序。

开发人员可以使用 Next.js 将静态网站生成 (SSG) 和 SSR 融合到一个项目中。SSG 页面在构建时预渲染，SSR 页面在请求时预渲染。

预渲染可以增强性能和搜索引擎优化。由于 Next.js 会预渲染服务器上的所有页面，因此每个页面的 HTML 内容在到达客户端的浏览器时就已准备就绪。这一内容的加载速度也更快。加载速度加快可以改善最终用户的网站体验，对网站的 SEO 排名产生积极影响。预渲染还使搜索引擎机器人能够轻松查找和爬取网站的 HTML 内容，从而改进 SEO。

Next.js 为衡量各种性能指标提供了内置的分析支持，例如首字节时间 (TTFB) 和首次内容绘制 (FCP)。有关 Next.js 的更多信息，请参阅 Next.js 网站上的开始使用。

Next.js 功能支持

对于使用 Next.js 版本 12 至 15 构建的应用程序，服务器端渲染（SSR）完全由 Amplify Hosting 计算管理。

如果您在 2022 年 11 月发布 Amplify Hosting 计算功能之前在 Amplify 上部署了 Next.js 应用程序，则该应用程序使用的是 Amplify 之前的 SSR 提供程序 Classic（仅限 Next.js 11）。Amplify Hosting 计算不支持使用 Next.js 11 或更早版本创建的应用程序。我们强烈建议您将 Next.js 11 应用程序迁移至 Amplify Hosting 计算托管 SSR 提供商。

以下列表描述了 Amplify Hosting 计算 SSR 提供商支持的具体功能。

支持的特征

- 服务器端渲染的页面 (SSR)
- 静态页面
- API 路由
- 动态路由
- 捕获所有路由
- 静态生成 (SSG)
- 增量静态再生成 (ISR)
- 国际化 (i18n) 子路径路由
- 国际化 (i18n) 域名路由
- 国际化 (i18n) 自动区域检测
- 中间件
- 环境变量
- 图像优化
- Next.js 13 应用程序目录

不支持的功能

- 边缘 API 路由（不支持边缘中间件）
- 按需增量静态再生成 (ISR)
- Next.js 流式处理
- 基于静态资产和经优化的图像运行中间件
- 收到 `unstable_after` 响应后执行代码（Next.js 15 中发布的实验性功能）

Next.js 图像

最大图像输出大小不能超过 4.3 MB。您可以将更大的图像文件存储在某个地方，然后使用 Next.js Image 组件调整尺寸并将其优化为 Webp 或 AVIF 格式，再以较小的尺寸提供。

注意，Next.js 文档建议您安装 Sharp 图像处理模块，使图像优化能够在生产环境中正常运行。但是，Amplify 部署不需要如此。Amplify 会自动为您部署 Sharp。

将 Next.js SSR 应用程序部署到 Amplify

默认情况下，Amplify 使用 Amplify Hosting 的计算服务（支持 Next.js 版本 12 至 15）来部署新 SSR 应用程序。Amplify Hosting 计算可完全管理部署 SSR 应用程序所需的资源。您的 Amplify 账户中在 2022 年 11 月 17 日之前部署的 SSR 应用程序使用的是 Classic（仅限 Next.js 11）SSR 提供商。

我们强烈建议您将使用 Classic（仅限 Next.js 11）SSR 的应用程序迁移至 Amplify Hosting 计算 SSR 提供商。Amplify 不会自动为您执行迁移。您必须手动迁移应用程序，然后启动新版本完成更新。有关说明，请参阅将 Next.js 11 SSR 应用程序迁移至 Amplify Hosting 计算。

请按照以下说明操作以部署新的 Next.js SSR 应用程序。

使用 Amplify Hosting 计算 SSR 提供商将 SSR 应用程序部署到 Amplify

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 在所有应用程序页面中，选择创建新应用程序。
3. 在开始使用 Amplify 进行构建页面中选择您的 Git 存储库提供商，然后选择下一步。
4. 在添加存储库分支页面上，执行以下操作：
   a. 在最近更新的存储库列表中，选择要连接的存储库的名称。
   b. 在分支列表中，选择要连接的存储库分支的名称。
   c. 选择下一步。
5. 该应用程序需要一个 IAM 服务角色，Amplify 在代表您调用其他服务时会代入该角色。您可以允许 Amplify Hosting 计算自动为您创建服务角色，也可以指定您已创建的角色。
   - 允许 Amplify 自动创建角色并将其附加到您的应用程序的方法：
     - 请选择创建和使用新的服务角色。
   - 附加您之前创建的服务角色的方法：
     a. 选择使用现有服务角色。
     b. 从列表中选择需要使用的角色。
6. 选择下一步。
7. 在查看页面上，选择保存并部署。

Package.json 文件设置

部署 Next.js 应用程序时，Amplify 会在 `package.json` 文件中检查该应用程序的构建脚本，以确定应用程序类型。

以下是适用于 Next.js 应用程序的构建脚本示例。构建脚本 `"next build"` 表示该应用程序同时支持 SSG 和 SSR 页面。此构建脚本也用于仅使用 Next.js 14 或更高版本的 SSG 应用程序。

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

以下是适用于 Next.js 13 或之前版本的 SSG 应用程序的构建脚本示例。构建脚本 `"next build && next export"` 表明该应用程序仅支持 SSG 页面。

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

Amplify Next.js SSR 应用程序的构建设置

在检查应用程序的 `package.json` 文件后，Amplify 会检查该应用程序的构建设置。您可以将构建设置保存在 Amplify 控制台中，也可以保存在存储库根目录下的 `amplify.yml` 文件中。有关更多信息，请参阅 配置 Amplify 应用程序的构建设置。

如果 Amplify 检测到您部署的是 Next.js SSR 应用程序，但不存在任何 `amplify.yml` 文件，则它会为该应用程序生成构建规范并将 `baseDirectory` 设置为 `.next`。如果您部署的是存在 `amplify.yml` 文件的应用程序，则该文件中的构建设置会覆盖控制台中的所有构建设置。因此，您必须在文件中将 `baseDirectory` 手动设置为 `.next`。

以下是将 `baseDirectory` 设置为 `.next` 的应用程序的构建设置示例。这表明构建构件适用于支持 SSG 和 SSR 页面的 Next.js 应用程序。

```
version: 1
frontend:
  phases:
```

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

适用于 Next.js 13 或之前版本的 SSG 应用程序的 Amplify 构建设置

如果 Amplify 检测到您在部署 Next.js 13 或之前版本的 SSG 应用程序，则会为该应用程序生成构建规范并将 `baseDirectory` 设置为 `out`。如果您部署的是存在 `amplify.yml` 文件的应用程序，则必须在文件中将 `baseDirectory` 手动设置为 `out`。`out` 目录是 Next.js 为存储导出的静态资源而创建的默认文件夹。配置应用程序的编译规范设置时，请更改 `baseDirectory` 文件夹的名称，以匹配您的应用程序配置。

以下是应用程序构建设置的示例，其中 `baseDirectory` 设置为 `out`，指明该构建构件适用于仅支持 SSG 页面的 Next.js 13 或之前版本的应用程序。

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

Next.js 14 或更高版本 SSG 应用程序的 Amplify 构建设置

在 Next.js 14 版本中，`next export` 命令已被弃用，并替换为 `next.config.js` 文件中的 `output: 'export'`，以启用静态导出。如果您在控制台部署仅限 Next.js 14 SSG 的应用程序，Amplify 会为该应用程序生成规范并将 `baseDirectory` 设置为 `.next`。如果您部署的是存在 `amplify.yml` 文件的应用程序，则必须在文件中将 `baseDirectory` 手动设置为 `.next`。这与 Amplify 为同时支持 SSG 和 SSR 页面的 Next.js `WEB_COMPUTE` 应用程序使用的 `baseDirectory` 设置相同。

以下是适用于仅限 Next.js 14 SSG 的应用程序构建设置的示例，其中的 `baseDirectory`设置为 `.next`。

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

将 Next.js 11 SSR 应用程序迁移至 Amplify Hosting 计算

部署新的 Next.js 应用程序时，默认情况下 Amplify 使用支持的 Next.js 最新版本。目前，Amplify Hosting 计算 SSR 提供商支持 Next.js 版本 15。

Amplify 控制台会检测您账户中在 2022 年 11 月发布 Amplify Hosting 计算服务（全面支持 Next.js 版本 12 至 15）之前部署的应用程序。控制台会显示一个信息横幅，标识使用 Amplify 以前的 SSR 提供商 Classic（仅限 Next.js 11）部署的带有分支的应用程序。我们强烈建议您将应用程序迁移至 Amplify Hosting 计算 SSR 提供商。

如果要将托管的 Next.js 11 应用程序更新到 Next.js 12 或更高版本，则在触发部署时可能会出现 `"target" property is no longer supported` 错误。在这种情况下，您必须迁移到 Amplify Hosting 计算服务。

您必须同时手动迁移应用程序及其所有生产分支。应用程序不能同时包含 Classic（仅限 Next.js 11）和 Next.js 12 或更新版本的分支。

按照以下说明将应用程序迁移至 Amplify Hosting 计算 SSR 提供商。

将应用程序迁移至 Amplify Hosting 计算 SSR 提供商

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要迁移的 Next.js 应用程序。

> **Note**
>
> 在 Amplify 控制台中迁移应用程序之前，必须先将应用程序的 package.json 文件更新，以使用 Next.js 12 或更高版本。

3. 在导航窗格中，依次选择应用程序设置、常规。
4. 如果应用程序有通过 Classic（仅限 Next.js 11）SSR 提供商部署的分支，则控制台会在应用程序主页上显示横幅。在横幅上，选择迁移。
5. 在迁移确认窗口中，选择三条语句并选择迁移。
6. Amplify 将构建并重新部署您的应用程序以完成迁移。

恢复 SSR 迁移

当您部署 Next.js 应用程序时，Amplify Hosting 会检测应用程序中的设置并为该应用程序设置内部平台值。有三个有效平台值。SSG 应用程序设置为平台值 `WEB`。使用 Next.js 版本 11 的 SSR 应用程序设置为平台值 `WEB_DYNAMIC`。Next.js 12 或更高版本的 SSR 应用程序设置为平台值 `WEB_COMPUTE`。

当您按照上一节中的说明迁移应用程序时，Amplify 会将应用程序的平台值从 `WEB_DYNAMIC` 更改为 `WEB_COMPUTE`。向 Amplify Hosting 计算完成迁移后，您无法在控制台中恢复迁移。要恢复迁移，必须使用 AWS Command Line Interface 将应用程序的平台值改回 `WEB_DYNAMIC`。打开终端窗口并输入以下命令，使用您的唯一信息更新应用程序 ID 和区域。

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

为静态 Next.js 应用程序添加 SSR 功能

您可以向部署有 Amplify 的现有静态 (SSG) Next.js 应用程序添加 SSR 功能。在开始将 SSG 应用程序转换为 SSR 之前，请将应用程序更新为使用 Next.js 12 或更高版本，并添加 SSR 功能。然后，您需要执行以下步骤。

1. 使用 AWS Command Line Interface 更改应用程序的平台类型。
2. 向应用程序添加服务角色。
3. 更新应用程序构建设置中的输出目录。
4. 更新应用程序的 `package.json` 文件以表明该应用程序使用 SSR。

更新平台

有三个平台类型的有效值。SSG 应用程序设置为平台类型 `WEB`。使用 Next.js 版本 11 的 SSR 应用程序设置为平台类型 `WEB_DYNAMIC`。对于借助由 Amplify Hosting 计算托管的 SSR 部署到 Next.js 12 或更新版本的应用程序，平台类型设置为 `WEB_COMPUTE`。

当您将应用程序部署为 SSG 应用时，Amplify 会将平台类型设置为 `WEB`。使用 AWS CLI 将您的应用程序的平台更改为`WEB_COMPUTE`。打开终端窗口，输入以下命令，使用您的应用程序 ID 和区域更新红色文本。

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

添加服务角色

服务角色是 Amplify 在代表您调用其他服务时扮演的 AWS Identity and Access Management (IAM) 角色。按照以下步骤向已部署 Amplify 的 SSG 应用程序添加服务角色。

添加服务角色

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 如果您尚未在 Amplify 账户中创建服务角色，请参阅添加服务角色以完成此先决条件。
3. 选择要向其添加服务角色的静态 Next.js 应用程序。
4. 在导航窗格中，依次选择应用程序设置、常规。
5. 在详情页上，选择编辑
6. 对于服务角色，请选择现有服务角色的名称或您在步骤 2 中创建的服务角色的名称。
7. 选择保存。

更新构建设置

在重新部署具有 SSR 功能的应用程序之前，必须更新应用程序的构建设置，将输出目录设置为 `.next`。您可以在 Amplify 控制台或存储库中存储的 `amplify.yml` 文件中编辑构建设置。有关更多信息，请参阅 配置 Amplify 应用程序的构建设置。

以下是将 `baseDirectory` 设置为 `.next` 的应用程序的构建设置示例。

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

更新 package.json 文件

添加服务角色并更新构建设置后，更新应用程序的 `package.json` 文件。如下例所示，将构建脚本设置为 `"next build"`，以指示 Next.js 应用程序同时支持 SSG 和 SSR 页面。

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

Amplify 会检测存储库中 `package.json` 文件的更改，并重新部署具有 SSR 功能的应用程序。

令服务器端运行时可以访问环境变量

Amplify Hosting 支持在 Amplify 控制台的项目配置中设置环境变量，从而为应用程序的构建添加环境变量。

但是，默认情况下，Next.js 服务器组件无权访问这些环境变量。此行为意在保护您的应用程序在构建阶段使用的环境变量所存储的任何密钥。

要使 Next.js 可以访问特定的环境变量，您可以修改 Amplify 构建规范文件，在 Next.js 可识别的环境文件中设置环境变量。这样 Amplify 就能够在构建应用程序之前加载这些环境变量。

> **Important**
>
> 我们强烈建议您不要在环境变量中存储任何凭证、机密或敏感信息，因为任何有权访问部署构件的用户都可以读取这些信息。
> 要授予您的 SSR 计算函数访问 AWS 资源的权限，我们建议使用 IAM 角色。

以下构建规范示例演示了如何在构建命令部分中添加环境变量。

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

在此示例中，构建命令部分包含两个在应用程序的构建运行之前将环境变量写入 `.env.production` 文件的命令。Amplify Hosting 允许您的应用程序在接收流量时访问这些变量。

上例中，构建命令部分的下一行演示了如何从构建环境中获取特定变量，并将其添加到 `.env.production` 文件中。

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

如果您的构建环境中存在这些变量，则 .env.production 文件将包含以下环境变量。

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

上例中，构建命令部分的下一行演示了如何向 .env.production 文件中添加带有特定前缀的环境变量。此示例中添加的变量都带有前缀 NEXT_PUBLIC_。

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

如果构建环境中存在多个带有 NEXT_PUBLIC_ 前缀的变量，则 .env.production 文件将如下所示。

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

适用于 monorepos 的 SSR 环境变量

如果您在 monorepo 中部署 SSR 应用程序，并希望使特定环境变量可供 Next.js 访问，则必须使用应用程序根目录作为 .env.production 文件的前缀。以下适用于 Next.js 的构建规范示例位于一个 Nx monorepo 中，演示了如何在构建命令部分中添加环境变量。

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

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

上例中，构建命令部分的后续行演示了如何从构建环境中获取特定变量，并针对应用程序根目录为 apps/app 的 monorepo 中的应用程序将其添加到 .env.production 文件中。

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

在单一存储库中部署 Next.js 应用程序

Amplify 支持通用单一存储库中的应用程序，以及使用 npm 工作空间、pnpm 工作空间、Yarn 工作空间、Nx 和 Turborepo 创建的单一存储库中的应用程序。当您部署应用程序时，Amplify 会自动检测您所使用的单一存储库构建框架。Amplify 会自动为 npm 工作空间、Yarn 工作空间或 Nx 中的应用程序应用构建设置。Turborepo 和 pnpm 应用程序需要额外的配置。有关更多信息，请参阅 配置 monorepo 构建设置。

有关 Nx 的详细示例，请参阅在 AWS Amplify Hosting 上使用 Nx 在 Next.js 应用程序之间共享代码博客文章。

Amplify 对 Nuxt.js 的支持

Nuxt 是用于通过 Vue.js 创建全栈 Web 应用程序的框架。

适配器

您可以在零配置的情况下使用预设适配器将 Nuxt.js 应用程序部署到 Amplify。有关此适配器的更多信息，请参阅 Nuxt 文档。

教程

要了解如何将 Nuxt.js 应用程序部署到 Amplify，请参阅将 Nuxt.js 应用程序部署到 Amplify Hosting。

演示

有关视频演示，请参阅 Nuxt 托管在几分钟内实现零配置（开 AWS启 YouTube）。

Amplify 对 Astro.js 的支持

Astro 是一种 Web 框架，用于创建内容驱动的 Web 应用程序。

适配器

您可以使用社区适配器将 Astro.js 应用程序部署到 Amplify。我们不维护 Amplify 拥有、适用于 Astro 框架的适配器。但是，github 上有适配器。 com/alexnguyennz/astro-aws-amplify 在网站上放大。 GitHub 此适配器由社区成员创建，并非由 AWS维护。

教程

要了解如何将 Astro 应用程序部署到 Amplify，请参阅将 Astro.js 应用程序部署到 Amplify Hosting。

演示

有关视频演示，请参阅如何在 Amazon Web Services YouTube 频道 AWS上部署 Astro 网站。

Amplify 对以下各项的支持 SvelteKit

SvelteKit 是一个用于使用 Svelte 创建完整堆栈 Web 应用程序的框架。

适配器

您可以使用社区适配器将 SvelteKit 应用程序部署到 Amplify。我们不为该框架维护 Amplify 自有的适配器。 SvelteKit 但是，github 上有适配器。 com/gzimbron/amplify- GitHub 网站上的适配器。此适配器由社区成员创建，并非由 AWS维护。

教程

要了解如何将 SvelteKit 应用程序部署到 Amplify，请参阅。将 SvelteKit 应用程序部署到 Amplify 托管

演示

有关视频演示，请参阅如何在 Amazon Web Services YouTube 频道 AWS上部署 SvelteKit 网站（使用 API）。

将 SSR 应用程序部署到 Amplify

您可以按照这些说明操作，以部署使用任何框架创建的应用程序，只要此类框架具有符合 Amplify 期望的构建输出的部署捆绑包即可。如果您正在部署 Next.js 应用程序，则不需要适配器。

如果您正在部署使用框架适配器的 SSR 应用程序，则必须首先安装和配置适配器。有关说明，请参阅为任意 SSR 框架使用开源适配器。

将 SSR 应用程序部署到 Amplify Hosting

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 在所有应用程序页面中，选择创建新应用程序。
3. 在开始使用 Amplify 进行构建页面中选择您的 Git 存储库提供商，然后选择下一步。
4. 在添加存储库分支页面上，执行以下操作：
   a. 选择需要连接的存储库的名称。
   b. 选择需要连接的存储库分支的名称。
   c. 选择下一步。
5. 在应用程序设置页面上，Amplify 会自动检测 Next.js SSR 应用程序。

   如果您要部署的 SSR 应用程序使用其他框架的适配器，则必须明确启用 Amazon CloudWatch Logs。打开高级设置部分，随后在服务器端渲染（SSR）部署部分中选择启用 SSR 应用程序日志。
6. 该应用程序需要一个 Amplify 代入的 IAM 服务角色来向 AWS 账户传送日志。

   添加服务角色的过程会有所不同，具体取决于您要创建新角色还是使用现有角色。
   - 若要创建新角色：
     - 请选择创建和使用新的服务角色。
   - 若要使用现有角色：
     a. 选择使用现有角色。
     b. 在服务角色列表中选择要使用的角色。
7. 选择下一步。
8. 在查看页面上，选择保存并部署。

支持的 SSR 功能

本部分提供的信息说明了 Amplify 对 SSR 功能的支持情况。

Amplify 提供 Node.js 版本支持，可匹配用于构建应用程序的 Node.js 版本。

Amplify 提供内置图像优化功能，支持所有 SSR 应用程序。如果您不想使用默认图像优化功能，则可以实现自定义图像优化加载程序。

主题

Node.js 版本支持 Next.js 应用程序

当 Amplify 构建和部署 Next.js 计算应用程序时，它使用的 Node.js 运行时版本与用于构建该应用程序的主要 Node.js 版本相匹配。

> **Note**
>
> 从 2025 年 9 月 15 日起，Amplify Hosting 将不再支持 Node.js 14、Node.js 16 和 Node.js 18 运行时。支持的运行时包括 Node.js 20 和 Node.js 22。

您可以在 Amplify 控制台的实时包覆盖功能中指定要使用的 Node.js 版本。有关配置实时包更新的更多信息，请参阅在构建映像中使用特定程序包和依赖项版本。您也可以使用其他机制（例如 nvm 命令）指定 Node.js 版本。如果未指定版本，Amplify 默认使用 Amplify 构建容器使用的当前版本。

SSR 应用程序的图像优化

Amplify Hosting 提供内置图像优化功能，支持所有 SSR 应用程序。借助 Amplify 的图像优化，您可以为访问这些图像的设备提供格式、尺寸和分辨率正确的高质量图像，同时保持尽可能小的文件大小。

目前，您可以使用 Next.js Image 组件按需优化图像，也可以实施自定义图像加载器。如果您使用的是 Next.js 13 或更高版本，则无需采取任何进一步措施即可使用 Amplify 的图像优化功能。如果您正在实现自定义加载程序，请参阅下面的使用自定义图像加载程序主题。

使用自定义图像加载器

如果您使用自定义图像加载器，Amplify 会检测到应用程序 `next.config.js` 文件中的加载器，并且不会使用内置的图像优化功能。有关 Next.js 支持的自定义加载器的更多信息，请参阅 Next.js Image 文档。

SSR 应用程序的 Amazon CloudWatch 日志

Amplify 将有关您的 SSR CloudWatch 运行时的信息发送到您的 Amazon Logs。 AWS 账户当您部署 SSR 应用程序时，Amplify 需要一个 IAM 服务角色，Amplify 在代表您调用其他服务时代入该角色。您可以允许 Amplify Hosting 计算自动为您创建服务角色，也可以指定您已创建的角色。

如果您选择允许 Amplify 为您创建 IAM 角色，则该角色将已经拥有创建 CloudWatch 日志的权限。如果您创建自己的 IAM 角色，则需要在策略中添加以下权限以允许 Amplify 访问亚马逊 CloudWatch 日志。

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

有关服务角色的更多信息，请参阅 添加具有后端资源部署权限的服务角色。

Amplify Next.js 11 SSR 支持

如果您在 2022 年 11 月 17 日 Amplify Hosting 计算发布之前在 Amplify 上部署了 Next.js 应用程序，那么您的应用程序使用的是 Amplify 之前的 SSR 提供商 Classic（仅限 Next.js 11）。本节中的文档仅适用于使用 Classic（仅限 Next.js 11）SSR 提供商部署的应用程序。

> **Note**
>
> 我们强烈建议您将 Next.js 11 应用程序迁移至 Amplify Hosting 计算托管 SSR 提供商。有关更多信息，请参阅 将 Next.js 11 SSR 应用程序迁移至 Amplify Hosting 计算。

下表描述了 Amplify Classic（仅限 Next.js 11）SSR 提供商支持的具体功能。

支持的特征

- 服务器端渲染的页面 (SSR)

- 静态页面
- API 路由
- 动态路由
- 捕获所有路由
- 静态生成 (SSG)
- 增量静态再生成 (ISR)
- 国际化 (i18n) 子路径路由
- 环境变量

不支持的功能

- 图像优化
- 按需增量静态再生成 (ISR)
- 国际化 (i18n) 域名路由
- 国际化 (i18n) 自动区域检测
- 中间件
- 边缘中间件
- 边缘 API 路由

Next.js 11 SSR 应用程序的定价

在部署 Next.js 11 SSR 应用程序时，Amplify 会在 AWS 您的账户中创建额外的后端资源，包括：

- Amazon Simple Storage Service (Amazon S3) 存储桶，为应用程序的静态资产存储资源。有关 Amazon S3 的费用信息，请参阅 Amazon S3 定价。
- 用于为应用程序提供服务的Amazon CloudFront 发行版。有关 CloudFront 费用的信息，请参阅 Amazon CloudFront 定价。
- 四个 Lambda @Edge 函数用于自定义所 CloudFront 提供的内容。

AWS Identity and Access Management Next.js 11 SSR 应用程序的权限

Amplify 需要 AWS Identity and Access Management (IAM) 权限才能部署 SSR 应用程序。对于 SSR 应用程序，Amplify 会部署诸如亚马逊 S3 存储桶、分配、函数Lambda@Edge、 CloudFront 亚马逊

SQS 队列（如果使用 ISR）和 IAM 角色等资源。如果没有所需的最低权限，则在尝试部署 SSR 应用程序时会出现 `Access Denied` 错误。要向 Amplify 提供所需的权限，您必须指定服务角色。

要创建 Amplify 控制台在代表您调用其他服务时代入的(IAM) 服务角色，请参见 添加具有后端资源部署权限的服务角色。这些说明演示了如何创建一个附加 `AdministratorAccess-Amplify` 托管策略的角色。

`AdministratorAccess-Amplify`托管策略提供对多种 AWS 服务的访问权限，包括 IAM 操作。应将其视为与该`AdministratorAccess`策略一样强大。此策略提供的权限超出了部署 SSR 应用程序所需的权限。

建议您遵循以下最佳实践授予最低权限并减少授予服务角色的权限。您可以创建自己的客户托管 IAM policy，仅授予部署 SSR 应用程序所需的权限，而不必向您的服务角色授予管理员访问权限。有关如何创建客户管理型策略的说明，请参阅 IAM 用户指南中创建 IAM policy。

如果您想创建自己的策略，请参阅以下部署 SSR 应用程序所需的最低权限列表。

```
acm:DescribeCertificate
acm:DescribeCertificate
acm:ListCertificates
acm:RequestCertificate
cloudfront:CreateCloudFrontOriginAccessIdentity
cloudfront:CreateDistribution
cloudfront:CreateInvalidation
cloudfront:GetDistribution
cloudfront:GetDistributionConfig
cloudfront:ListCloudFrontOriginAccessIdentities
cloudfront:ListDistributions
cloudfront:ListDistributionsByLambdaFunction
cloudfront:ListDistributionsByWebACLId
cloudfront:ListFieldLevelEncryptionConfigs
cloudfront:ListFieldLevelEncryptionProfiles
cloudfront:ListInvalidations
cloudfront:ListPublicKeys
cloudfront:ListStreamingDistributions
cloudfront:UpdateDistribution
cloudfront:TagResource
cloudfront:UntagResource
cloudfront:ListTagsForResource
iam:AttachRolePolicy
iam:CreateRole
iam:CreateServiceLinkedRole
```

```
iam:GetRole
iam:PutRolePolicy
iam:PassRole
lambda:CreateFunction
lambda:EnableReplication
lambda:DeleteFunction
lambda:GetFunction
lambda:GetFunctionConfiguration
lambda:PublishVersion
lambda:UpdateFunctionCode
lambda:UpdateFunctionConfiguration
lambda:ListTags
lambda:TagResource
lambda:UntagResource
route53:ChangeResourceRecordSets
route53:ListHostedZonesByName
route53:ListResourceRecordSets
s3:CreateBucket
s3:GetAccelerateConfiguration
s3:GetObject
s3:ListBucket
s3:PutAccelerateConfiguration
s3:PutBucketPolicy
s3:PutObject
s3:PutBucketTagging
s3:GetBucketTagging
lambda:ListEventSourceMappings
lambda:CreateEventSourceMapping
iam:UpdateAssumeRolePolicy
iam:DeleteRolePolicy
sqs:CreateQueue           // SQS only needed if using ISR feature
sqs:DeleteQueue
sqs:GetQueueAttributes
sqs:SetQueueAttributes
amplify:GetApp
amplify:GetBranch
amplify:UpdateApp
amplify:UpdateBranch
```

Next.js 11 SSR 部署的问题排查

如果您在使用 Amplify 部署 Classic（仅限 Next.js 11）SSR 应用程序时遇到意外问题，请查看以下问题排查主题。

主题

我的应用程序的输出目录被覆盖

使用 Amplify 部署的 Next.js 应用程序的输出目录必须设置为 `.next`。如果应用程序的输出目录被覆盖，请检查 `next.config.js` 文件。要将生成输出目录设置默认为 `.next`，请从文件中删除以下行：

```
distDir: 'build'
```

确认构建设置中的输出目录设置为 `.next`。有关查看应用程序构建设置的信息，请参阅 配置 Amplify 应用程序的构建设置。

以下是将 `baseDirectory` 设置为 `.next` 的应用程序的构建设置示例。

```
version: 1
frontend:
  phases:
    preBuild:
      commands:
        - npm ci
    build:
      commands:
        - npm run build
```

```
  artifacts:
    baseDirectory: .next
    files:
      - '**/*'
  cache:
    paths:
      - node_modules/**/*
```

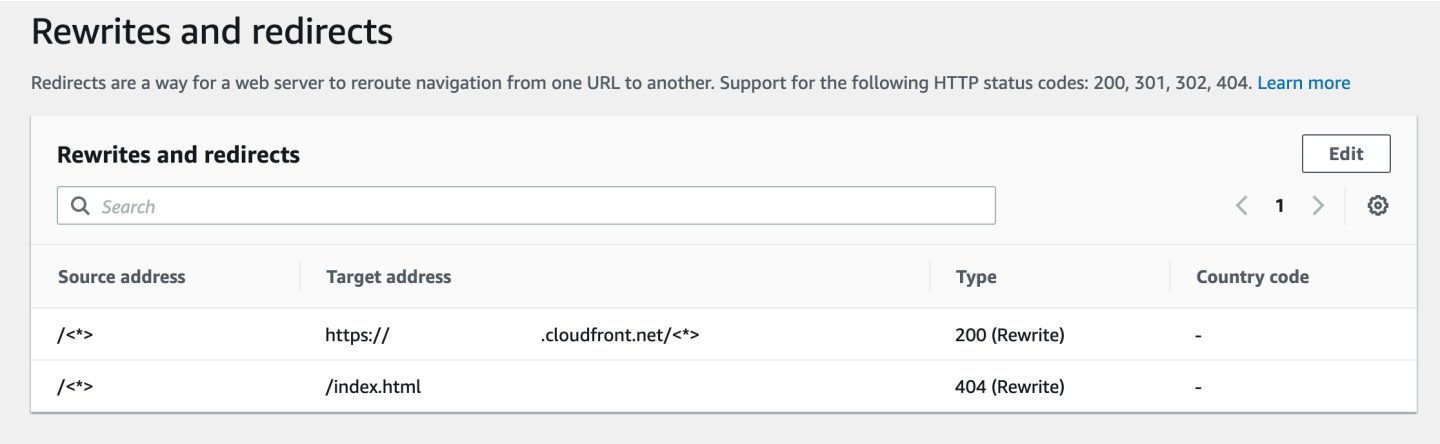
我在部署 SSR 网站后收到 404 错误

如果您在部署站点后遇到 404 错误，问题可能是由您的输出目录被覆盖所致。要检查您的 `next.config.js` 文件并验证应用程序构建规范中的构建输出目录是否正确，请按照上一主题 我的应用程序的输出目录被覆盖 中的步骤操作。

我的应用程序缺少 CloudFront SSR 发行版的重写规则

在您部署 SSR 应用程序时，Amplify 会为您的 SSR 发行版创建一条重写规则。 CloudFront 如果您无法在网络浏览器中访问您的应用程序，请在 Amplify 控制台中验证您的应用程序是否存在 CloudFront 重写规则。如果缺失，您可以手动添加或重新部署您的应用程序。

要在 Amplify 控制台中查看或编辑应用程序的重写和重定向规则，请在导航窗格中依次选择应用程序设置、重写和重定向。以下屏幕截图例举了 Amplify 在部署 SSR 应用时为您创建的重写规则。请注意，在此示例中，存在 CloudFront 重写规则。



我的应用程序太大，无法部署

Amplify 将 SSR 部署的大小限制在 50 MB 以内。如果您尝试将 Next.js SSR 应用程序部署到 Amplify 但出现了 `RequestEntityTooLargeException` 错误，则说明您的应用程序太大，无法部署。您可以尝试通过向 `next.config.js` 文件中添加缓存清理代码来解决此问题。

以下是 `next.config.js` 文件中执行缓存清理的代码示例。

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

我的构建失败并出现内存不足错误

Next.js 允许您缓存构建构件，以提高后续构建的性能。此外，Amplify 的 AWS CodeBuild 容器会代表您压缩此缓存并将其上传到 Amazon S3，以提高后续构建性能。这可能导致您的构建失败，出现内存不足错误。

执行以下操作以防止您的应用程序在构建阶段超过内存限制。首先，从构建设置的 cache.paths 部分中删除 `.next/cache/**/*`。接下来，从您的构建设置文件中移除 `NODE_OPTIONS` 环境变量。相反，在 Amplify 控制台中设置 `NODE_OPTIONS` 环境变量，定义节点最大内存限制。有关在 Amplify 控制台中设置环境变量的更多信息，请参阅 设置环境变量。

完成这些更改后，重新尝试构建。如果成功了，重新将 `.next/cache/**/*` 添加到构建设置文件的 cache.paths 部分。

有关用于提高构建性能的 Next.js 缓存配置的更多信息，请参阅 Next.js CodeBuild 网站上的 AWS。

我的应用程序同时具有 SSR 和 SSG 分支

您无法部署同时具有 SSR 和 SSG 分支的应用程序。如果您需要同时部署 SSR 和 SSG 分支，则必须部署一个仅使用 SSR 分支的应用程序和另一个仅使用 SSG 分支的应用程序。

我的应用程序将静态文件存储在带有保留路径的文件夹中

Next.js 可以从存储在项目根目录中的名为 `public` 的文件夹中提供静态文件。当您使用 Amplify 部署和托管 Next.js 应用程序时，您的项目不能包含带有路径 `public/static` 的文件夹。Amplify 保留了分发应用程序时使用的 `public/static` 路径。如果您的应用包含此路径，则必须先重命名 `static` 文件夹，才能使用 Amplify 进行部署。

我的申请已达到上 CloudFront 限

CloudFront 服务配额将您的 AWS 账户限制为 25 个附带 Lambda @Edge 函数的分配。如果您超过此配额，则可以从您的账户中删除任何未使用的 CloudFront 分配，也可以申请增加配额。有关更多信息，请参阅《服务配额用户指南》中的 Requesting a quota increase。

在美国东部（弗吉尼亚州北部）区域中创建 Lambda@Edge 函数

当你部署 Next.js 应用程序时，Amplify 会创建 Lambda @Edge 函数来自定义交付的内容。CloudFront Lambda@Edge 函数应在美国东部（弗吉尼亚州北部）区域创建，而不是您应用程序的部署区域。此为 Lambda@Edge 限制。有关 Lambda @Edge 函数的更多信息，请参阅亚马逊 CloudFront 开发者指南中的边缘函数限制。

我的 Next.js 应用程序使用了不支持的功能

使用 Amplify 部署的应用程序支持 Next.js 11 之前的主要版本。有关 Amplify 支持和不支持的 Next.js 功能的详细列表，请参阅 supported features。

当您部署新的 Next.js 应用程序时，Amplify 默认使用支持的最新版本 Next.js。如果您有一款使用旧版本 Next.js 部署到 Amplify 的 Next.js 应用程序，则可以将该应用程序迁移至 Amplify Hosting 计算 SSR 提供商。有关说明，请参阅将 Next.js 11 SSR 应用程序迁移至 Amplify Hosting 计算。

我的 Next.js 应用程序中的图像无法加载

使用 `next/image` 组件向 Next.js 应用程序添加图像时，图像的大小不能超过 1 MB。将应用程序部署到 Amplify 时，大于 1 MB 的图像将返回 503 错误。这是由 Lambda@Edge 限制引起的，它将 Lambda 函数生成的响应（包括标头和正文）的大小限制为 1 MB。

1 MB 的限制适用于应用程序中的其他构件，例如 PDF 和文档文件。

不支持的区域

Amplify 不支持在所有 AWS Amplify 可用地区部署 Classic（仅限 Next.js 11）SSR 应用程序。以下地区不支持 Classic（仅限 Next.js 11）SSR：欧洲地区（米兰）eu-south-1、中东（巴林）me-south-1 和亚太地区（香港）ap-east-1。

SSR 部署的问题排查

如果您在使用 Amplify Hosting 计算部署 SSR 应用程序时遇到意外问题，请参阅“Amplify 问题排查”一章中的为在服务器端渲染的应用程序排查问题。

高级：开源适配器

框架作者可使用基于文件系统的部署规范来开发针对其特定框架定制的开源构建适配器。这些适配器会将应用程序的构建输出转换为符合 Amplify Hosting 预期目录结构的部署包。此部署包将包含托管应用程序所需的所有必要文件和资产，包括运行时配置，例如路由规则。

如果您不使用框架，则可以开发自己的解决方案来生成 Amplify 期望的构建输出。

主题

使用 Amplify Hosting 部署规范配置构建输出

Amplify Hosting 部署规范是基于文件系统的规范，它定义了便于部署到 Amplify Hosting 的目录结构。框架可以生成这种预期的目录结构作为其构建命令的输出，使框架能够利用 Amplify Hosting 的服务基元。Amplify Hosting 了解部署包的结构并相应地对其进行部署。

有关解释如何使用部署规范的视频演示，请参阅如何在 Amazon Web Services YouTube 频道 AWS Amplify上使用托管任何网站。

以下是 Amplify 期望部署包采用的文件夹结构示例。简而言之，它有一个名为 `static` 的文件夹、一个名为 `compute` 的文件夹和一个名为 `deploy-manifest.json` 的部署清单文件。

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

```
#   #   ### entry.xxx.js
#   ### favicon.ico
#   ### robots.txt
### deploy-manifest.json
```

Amplify SSR 基元支持

Amplify Hosting 部署规范定义了与以下基元紧密映射的合约。

静态资产

为框架提供托管静态文件的功能。

计算

使框架能够在端口 3000 上运行 Node.js HTTP 服务器。

图像优化

为框架提供在运行时优化图像的服务。

路由规则

为框架提供一种将传入请求路径映射到特定目标的机制。

.amplify-hosting/static 目录

必须将本应通过应用程序 URL 提供的所有可公开访问的静态文件放在 `.amplify-hosting/static` 目录中。此目录中的文件通过静态资产基元提供。

可以在应用程序 URL 的根 (/) 处访问静态文件，而无需对其内容、文件名或扩展名进行任何更改。此外，子目录保留在 URL 结构中，并出现在文件名之前。例如，`.amplify-hosting/static/favicon.ico` 将从 `https://myAppId.amplify-hostingapp.com/favicon.ico` 中提供，`.amplify-hosting/static/_nuxt/main.js` 将从 `https://myAppId.amplify-hostingapp.com/_nuxt/main.js` 中提供

如果框架支持修改应用程序基本路径的功能，则它必须在 `.amplify-hosting/static` 目录内的静态资产前面加上基本路径。例如，如果基本路径是 `/folder1/folder2`，则名为 `main.css` 的静态资源的构建输出将是 `.amplify-hosting/static/folder1/folder2/main.css`。

.amplify-hosting/compute 目录

单个计算资源由 `default` 目录中包含的名为 `.amplify-hosting/compute` 的单个子目录表示。路径是 `.amplify-hosting/compute/default`。此计算资源映射到 Amplify Hosting 的计算基元。

子目录 default 的内容必须符合以下规则。

- 文件必须存在于子目录 default 的根目录中，才能作为计算资源的入口点。
- 入口点文件必须是 Node.js 模块，并且它必须启动一个在端口 3000 上侦听的 HTTP 服务器。
- 您可以将其他文件放在 default 子目录中，并从入口点文件中的代码中引用它们。
- 子目录的内容必须是自包含的。入口点模块中的代码不能引用子目录之外的任何模块。请注意，框架可以以任何方式捆绑其 HTTP 服务器。如果可以在子目录中使用 node server.js 命令（其中 server.js is 是入口文件的名称）启动计算过程，则 Amplify 认为目录结构符合部署规范。

Amplify Hosting 将 default 子目录中的所有文件捆绑并部署到预置的计算资源中。每个计算资源被分配 512 MB 的临时存储。此存储不在执行实例之间共享，而是在同一执行实例中的后续调用之间共享。执行实例的最大执行时间限制为 15 分钟，并且执行实例中唯一可写的路径是 /tmp 目录。每个计算资源捆绑包的未压缩大小不能超过 220 MB。例如，未压缩的 .amplify/compute/default 子目录不能超过 220 MB。

.amplify-hosting/deploy-manifest.json 文件

使用 deploy-manifest.json 文件存储部署的配置详细信息和元数据。deploy-manifest.json 文件必须至少包含一个 version 属性、指定了捕获所有路由的 routes 属性以及以及指定了框架元数据的 framework 属性。

以下对象定义演示了部署清单的配置。

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

以下主题描述了部署清单中每个属性的详细信息和用法。

使用版本属性

version 属性定义了您正在实施的部署规范的版本。目前，Amplify Hosting 部署规范的唯一版本是版本 1。以下 JSON 示例说明了如何使用 version 属性。

```
"version": 1
```

使用路由属性

routes 属性使框架能够利用 Amplify Hosting 路由规则基元。路由规则提供了一种机制，用于将传入的请求路径路由到部署包中的特定目标。路由规则仅规定传入请求的目的地，并在通过重写和重定向规则转换请求后应用。有关 Amplify Hosting 如何处理重写和重定向的更多信息，请参阅为 Amplify 应用程序设置重定向和重写

路由规则不会重写或转换请求。如果传入的请求与路由的路径模式匹配，则该请求将按原样路由到路径的目标。

routes 数组中指定的路由规则必须符合以下规则。

- 必须指定捕获所有路由。捕获所有路由具有匹配所有传入请求的 /* 模式。
- routes 数组最多可以包含 25 个项目。
- 您必须指定 Static 路由或 Compute 路由。
- 如果指定 Static 路由，则 .amplify-hosting/static 目录必须存在。
- 如果指定 Compute 路由，则 .amplify-hosting/compute 目录必须存在。
- 如果指定 ImageOptimization 路由，则还必须指定 Compute 路由。这是必要操作，因为纯静态应用程序尚不支持图像优化。

以下对象定义演示了 Route 对象的配置。

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

以下列表描述了 Route 对象的属性。

Key	Type	必需	说明
path	字符串	是	定义与传入请求路径（不包括查询字符串）相匹配的模式。 路径最大长度为 255 个字符。

Key	Type	必需	说明
			路径必须以正斜杠 / 开头。 路径可以包含以下任何字符：[A-Z]、[a-z]、[0-9]、[_-.*$/~"'@:+]。 对于模式匹配，仅支持以下通配符： • * (匹配 0 或多个字符) • /* 模式被称为捕获所有模式，它将匹配所有传入的请求。
target	Target	是	一个对象，用于定义将匹配的请求路由到的目标。 如果指定了 Compute 路由，则必须存在相应的 ComputeResource 。 如果指定了 ImageOptimization ，则还必须指定 imageSettings 。

Key	Type	必需	说明
回退	Target	否	一个对象，用于定义原始目标返回 404 错误时要回退到的目标。 指定路径的 `fallback` 种类和 `target` 种类不能相同。例如，不允许从 `Static` 回退到 `Static`。只有没有正文的 GET 请求才支持回退。如果请求中存在正文，则将在回退期间将其丢弃。

以下对象定义演示了 `Target` 对象的配置。

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

以下列表描述了 `Target` 对象的属性。

Key	Type	必需	说明
类型	Targetkind	是	定义目标类型的 `enum`。有效值包括 `Static`、`Compute` 和 `ImageOpti mization` 。

Key	Type	必需	说明
src	字符串	对于 Compute，为是 对于其他基元，为否	一个字符串，用于指定包含基元的可执行代码的部署包中的子目录的名称。仅对计算基元有效，且为必填项。 该值必须指向部署包中存在的计算资源之一。目前，此字段唯一支持的值为 default。

Key	Type	必需	说明
cacheControl	字符串	否	一个字符串，它指定要应用于响应的 Cache-Control 标头的值。仅对静态和 ImageOptimization基元有效。 指定的值会被自定义标头覆盖。有关 Amplify Hosting 客户标头的更多信息，请参阅为 Amplify 应用程序设置自定义 HTTP 标头。 **Note** 此 Cache-Control 标头仅适用于状态码设置为 200（OK）的成功响应。

以下对象定义演示了 `TargetKind` 枚举的用法。

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

以下列表指定了 `TargetKind` 枚举的有效值。

静态

将请求路由到静态资产基元。

计算

将请求路由到计算基元。

ImageOptimization

将请求路由到图像优化基元。

以下 JSON 示例说明了如何使用指定了多个路由规则的 `routes` 属性。

```
"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"
      }
    }
  ]
```

有关在部署清单中指定路由规则的详细信息，请参阅配置路由规则的最佳实践。

使用 computeResources 属性

computeResources 属性使框架能够提供有关预置计算资源的元数据。每个计算资源都必须有与之关联的相应路由。

以下对象定义演示了 ComputeResource 对象的用法。

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

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

以下列表描述了 ComputeResource 对象的属性。

Key	Type	必需	说明
name	字符串	是	指定计算资源的名称。名称必须

Key	Type	必需	说明
			与 .amplify-hosting/compute directory 内的子目录名称匹配。 对于部署规范的版本 1，唯一的有效值为 default。
运行时	ComputeRuntime	是	定义预置计算资源的运行时。 有效值为 nodejs20.x 和 nodejs22.x 。
entrypoint	字符串	是	为指定的计算资源指定代码将从中运行的启动文件的名称。该文件必须存在于代表计算资源的子目录中。

您的目录结构必须与以下结构类似。

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

computeResource 属性的 JSON 如下所示。

```
"computeResources": [
    {
      "name": "default",
      "runtime": "nodejs20.x",
```

```
      "entrypoint": "index.js",
    }
  ]
```

使用 imageSettings 属性

imageSettings 属性使框架能够自定义图像优化基元的行为，该基元在运行时提供图像的按需优化。

以下对象定义演示了 ImageSettings 对象的用法。

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

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

以下列表描述了 ImageSettings 对象的属性。

Key	Type	必需	说明
尺寸	Number[]	是	支持的图像宽度数组。
域	String[]	是	允许使用图像优化的外部域的数组。将数组留空，仅允许部署域使用图像优化。
remotePatterns	RemotePattern[]	是	允许使用图像优化的外部模式的数组。与域类似，但通过正则表达式（regex）提供了更多控制。

Key	Type	必需	说明
格式	ImageFormat[]	是	允许的输出图像格式的数组。
minimumCacheTTL	数字	是	优化图像的缓存时长（以秒为单位）。
dangerouslyAllowSVG	布尔值	是	允许 SVG 输入图像 URLs。默认情况下，出于安全考虑，此功能处于禁用状态。

以下对象定义演示了 `RemotePattern` 对象的用法。

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

以下列表描述了 `RemotePattern` 对象的属性。

Key	Type	必需	说明
protocol	字符串	否	允许的远程模式的协议。唯一有效值为 `https`。
hostname	字符串	是	允许的远程模式的主机名。 您可以指定文本或通配符。单个 `*` 匹配单个子域。双 `**` 匹配任意数量的子域。在仅指定 `**` 的情况

Key	Type	必需	说明
			下，Amplify 不允许使用笼统通配符。
端口	字符串	否	允许的远程模式的端口。
pathname	字符串	否	允许的远程模式的路径名称。

以下示例说明了 `imageSettings` 属性。

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

使用框架属性

使用 `framework` 属性来指定框架元数据。

以下对象定义演示了 `FrameworkMetadata` 对象的配置。

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

以下列表描述了 FrameworkMetadata 对象的属性。

Key	Type	必需	说明
name	字符串	是	框架的名称。
版本	字符串	是	框架的版本。 它必须是有效的语义版本控制（semver）字符串。

配置路由规则的最佳实践

路由规则提供了一种机制，用于将传入的请求路径路由到部署包中的特定目标。在部署捆绑包中，框架作者可以将部署到以下任一目标的文件发送到构建输出：

- 静态资源基元 – 文件包含在 .amplify-hosting/static 目录中。
- 计算基元 – 文件包含在 .amplify-hosting/compute/default 目录中。

框架作者还在部署清单文件中提供了一系列路由规则。数组中的每条规则都按顺序与传入的请求进行匹配，直到完成匹配为止。当存在匹配规则时，请求会被路由到匹配规则中指定的目标。或者，可以为每条规则指定一个回退目标。如果原始目标返回 404 错误，则会将请求路由到回退目标。

部署规范要求遍历顺序中的最后一条规则是捕获所有规则。使用 /* 路径指定了捕获所有规则。如果传入的请求与路由规则数组中先前的任何路由都不匹配，则该请求将被路由到捕获所有规则目标。

对于像 Nuxt.js 这样的 SSR 框架，捕获所有规则目标必须是计算基元。这是因为 SSR 应用程序具有服务器端渲染的页面，这些页面的路由在构建时是不可预测的。例如，如果 Nuxt.js 应用程序在 [slug] 处有一个页面，则其中 /blog/[slug] 是动态路由参数。捕获所有规则目标是将请求路由到这些页面的唯一方法。

相比之下，可以使用特定的路径模式来定位构建时已知的路由。例如，Nuxt.js 从 /_nuxt 路径中提供静态资产。这意味着 /_nuxt/* 路径可以由特定的路由规则来定向，该规则将请求路由到静态资产基元。

公共文件夹路由

大多数 SSR 框架提供了从 public 文件夹提供可变静态资产的能力。favicon.ico 和 robots.txt 之类的文件通常保存在 public 文件夹中，并通过应用程序的根 URL 提供。例如，favicon.ico 文件是从 https://example.com/favicon.ico 提供的。请注意，这些文件没有可预测的路径模式。它们几乎完全由文件名决定。在 public 文件夹中定位文件的唯一方法是使用捕获所有路由。但是，捕获所有路由目标必须是计算基元。

我们建议使用以下方法之一来管理 public 文件夹。

1. 使用路径模式来定位包含文件扩展名的请求路径。例如，您可以使用 /*.* 定位所有包含文件扩展名的请求路径。

   请注意，这种方法可能不可靠。例如，如果 public 文件夹内有没有文件扩展名的文件，则此规则不针对这些文件。使用这种方法需要注意的另一个问题是，应用程序的页面名称中可能有句点。例如，/blog/2021/01/01/hello.world 处的页面将被 /*.* 规则将定位。这并不理想，因为该页面不是静态资产。但是，您可以在此规则中添加回退目标，以确保当静态基元出现 404 错误时，请求会回退到计算基元。

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

2. 在构建时识别 public 文件夹中的文件，并为每个文件发出路由规则。这种方法不可扩展，因为部署规范规定了 25 条规则的限制。

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

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

3. 建议您的框架用户将所有可变的静态资源存储在 public 文件夹内的子文件夹中。

   在以下示例中，用户可以将所有可变的静态资产存储在 public/assets 文件夹中。然后，可以使用带有路径模式 /assets/* 的路由规则来定位 public/assets 文件夹内所有可变的静态资产。

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

4. 为捕获所有路由指定一个静态回退。这种方法的缺点将在下一节“捕获所有回退路由”中详细介绍。

捕获所有回退路由

对于 Nuxt.js 等 SSR 框架，例如为计算基元目标指定了捕获所有路由，框架作者可以考虑为捕获所有路由指定静态回退以解决 public 文件夹路由问题。但是，这种类型的路由规则会破坏服务器端渲染的 404 页面。例如，如果最终用户访问了一个不存在的页面，则应用程序会呈现一个状态码为 404 的 404 页面。但是，如果捕获所有路由具有静态回退，则不会呈现 404 页面。取而代之的是，请求会回退到静态基元中，但最终仍会显示 404 状态码，但是 404 页面无法渲染。

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

基本路径路由

提供修改应用程序基本路径功能的框架应预先设置 `.amplify-hosting/static` 目录内静态资产的基本路径。例如，如果基本路径是 `/folder1/folder2`，则名为 main.css 的静态资源的构建输出将是 `.amplify-hosting/static/folder1/folder2/main.css`。

这意味着还需要更新路由规则以反映基本路径。例如，如果基本路径是 `/folder1/folder2`，则 `public` 文件夹中静态资产的路由规则将如下所示。

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

同样，服务器端路由也需要在它们前面加上基本路径。例如，如果基本路径是 `/folder1/folder2`，则 `/api` 路由的路由规则将如下所示。

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

但是，不应将基本路径置于捕获所有路由之前。例如，如果基本路径是 `/folder1/folder2`，则捕获所有路由将保持如下所示。

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

Nuxt.js 路由示例

以下是 Nuxt 应用程序的示例 `deploy-manifest.json` 文件，演示了如何指定路由规则。

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

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

以下是 Nuxt 的示例 `deploy-manifest.json` 文件，演示了如何指定包括基本路径在内的路由规则。

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

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

有关使用 routes 属性的更多信息，请参阅使用路由属性。

使用部署清单部署 Express 服务器

此示例说明了如何使用 Amplify Hosting 部署规范部署基本的 Express 服务器。您可以利用提供的部署清单来指定路由、计算资源和其他配置。

先在本地设置 Express 服务器，然后再部署到 Amplify Hosting

1. 为您的项目创建一个新目录并安装 Express 和 Typescript。

```
mkdir express-app
cd express-app

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

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

2. 在项目的根目录中添加一个包含以下内容的 `tsconfig.json` 文件。

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

3. 在项目根目录中创建一个名为 `src` 的目录。
4. 在 `src` 目录中创建 `index.ts` 文件。这将是启动 Express 服务器的应用程序的入口点。应将服务器配置为在端口 3000 上侦听。

```
// 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. 在您的 package.json 文件中添加以下脚本。

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

6. 在项目根目录中创建一个名为 public 的目录。然后使用以下内容创建名为 hello-world.txt 的文件。

```
Hello world!
```

7. 在项目的根目录中添加一个包含以下内容的 .gitignore 文件。

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

设置 Amplify 部署清单

1. 在项目根目录中创建一个名为 deploy-manifest.json 的文件。
2. 复制并在 deploy-manifest.json 文件中粘贴以下清单。

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

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

清单描述了 Amplify Hosting 应如何处理您的应用程序的部署。主要设置如下。

- version – 表示您正在使用的部署规范的版本。
- framework – 调整此设置以指定您的 Express 服务器设置。
- imageSettings – 除非您正在处理图像优化，否则此部分对于 Express 服务器来说是可选选项。
- routes – 这些路由对于将流量引导到应用程序的正确部分至关重要。`"kind": "Compute"` 路由将流量引导到您的服务器逻辑。

- computeResources – 使用此部分来指定 Express 服务器的运行时和入口点。

接下来，设置一个构建后脚本，用于将已构建的应用程序构件移动到 `.amplify-hosting` 部署包中。目录结构符合 Amplify Hosting 部署规范。

设置构建后脚本

1. 在项目根目录中创建一个名为 `bin` 的目录。
2. 在 `postbuild.sh` 目录中创建一个名为 `bin` 的文件。将以下内容添加到 `postbuild.sh` 文件。

```
#!/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. 在 `package.json` 文件中添加 `postbuild` 脚本。文件应该呈现以下状态。

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

4. 要构建应用程序，请运行以下命令。

```
npm run build
```

5. （可选）调整您的 Express 路由。您可以修改部署清单中的路由，使其适合您的 Express 服务器。例如，如果 `public` 目录中没有任何静态资产，则可能只需要指向 Compute 的捕获所有路由 `"path": "/*"`。这取决于您的设置。

最终目录结构应如下所示。

```
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
```

部署服务器

1. 将您的代码推送到您的 Git 存储库，然后将您的应用程序部署到 Amplify Hosting。
2. 将您的构建设置更新为指向 `baseDirectory` 到 `.amplify-hosting`，如下所示。在构建过程中，Amplify 将检测 `.amplify-hosting` 目录中的清单文件并按照配置部署您的 Express 服务器。

```
version: 1
frontend:
  phases:
    preBuild:
      commands:
        - nvm use 18
```

```
        - npm install
    build:
      commands:
        - npm run build
  artifacts:
    baseDirectory: .amplify-hosting
    files:
      - '**/*'
```

3. 要验证您的部署是否成功以及服务器是否正常运行，请通过 Amplify Hosting 提供的默认 URL 访问您的应用程序。

框架作者的图像优化集成

框架作者可以使用 Amplify Hosting 部署规范来集成 Amplify 的图像优化功能。要启用图像优化，您的部署清单必须包含针对图像优化服务的路由规则。以下示例演示了如何配置路由规则。

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

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

有关使用部署规范配置图像优化设置的更多信息，请参阅 使用 Amplify Hosting 部署规范配置构建输出。

了解图像优化 API

图像优化可以在运行时通过 Amplify 应用程序的域 URL 在路由规则定义的路径上调用。

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

图像优化对图像施加了以下规则。

- Amplify 无法优化 GIF、APNG 和 SVG 格式，也无法将其转换为其他格式。
- 除非启用 `dangerouslyAllowSVG` 设置，否则不提供 SVG 图像。
- 源图像的宽度或高度不能超过 11 MB 或 9000 像素。
- 经过优化的图像的大小限制为 4 MB。
- HTTPS 是唯一支持通过远程获取图像的协议 URLs。

HTTP 标头

Accept 请求 HTTP 标头用于指定客户端（通常是 Web 浏览器）允许的图像格式，以 MIME 类型表示。图像优化服务将尝试将图像转换为指定格式。为此标头指定的值将比格式查询参数具有更高的优先级。例如，Accept 标头的有效值为 `image/png, image/webp, */*` 。Amplify 部署清单中指定的格式设置会将格式限制为列表中的格式。即使 Accept 标头要求使用特定的格式，如果该格式不在允许列表中，它也会被忽略。

URI 请求参数

下表介绍了的图像优化的 URI 请求参数。

查询参数	Type	必需	说明	示例
url	字符串	是	源图像的相对路径或绝对 URL。对于远程 URL，HTTPS 是唯一受支持的协议。值必须为 URL 编码。	`?url=https%3A%2F%2Fwww.example.com%2Fbuffalo.png`
width	数字	是	以像素为单位的优化图像宽度。	`?width=800`
height	数字	否	以像素为单位的优化图像高度。如果未指定，则将自动缩放图像以匹配宽度。	`?height=600`

查询参数	Type	必需	说明	示例
fit	枚举 值：cover、cont	否	如何调整图像大小以适应指定的宽度和高度。	?width=800&height=600&fit=cover
position	枚举 值：center、top	否	fit 为 cover 或 contain 时要使用的位置。	?fit=contain&position=centre
trim	数字	否	修剪所有边缘的像素，这些像素包含与左上角像素的指定背景颜色相似的值。	?trim=50
扩展	对象	否	使用从最近的边缘像素派生的颜色将像素添加到图像的边缘。格式为 {top}_{right}_{bottom}_{left} ，其中每个值都是要添加的像素数。	?extend=10_0_5_0
extract	对象	否	将图像裁剪到由顶部、左侧、宽度和高度分隔的指定矩形。格式为 {left}_{top}_{width}_{right}，其中每个值都是要裁剪的像素数。	?extract=10_0_5_0

查询参数	Type	必需	说明	示例
format	字符串	否	优化图像所需的输出格式。	`?format=webp`
quality	数字	否	图像的质量，从 1 到 100。仅在转换图像格式时使用。	`?quality=50`
rotate	数字	否	按指定角度（以度数为单位）旋转图像。	`?rotate=45`
flip	布尔值	否	在 x 轴上垂直（上下方向）镜像图像。此操作总是发生在旋转之前（如果有）。	`?flip`
flop	布尔值	否	在 y 轴上水平（左右方向）镜像图像。此操作总是发生在旋转之前（如果有）。	`?flop`
sharpen	数字	否	锐化可增强图像中边缘的清晰度。有效值在 0.000001 到 10 之间。	`?sharpen=1`
median	数字	否	应用中值滤波。这样可以去除噪点或平滑图像的边缘。	`?sharpen=3`

查询参数	Type	必需	说明	示例
blur	数字	否	应用指定 sigma 的高斯模糊。有效值介于 0.3 到 1,000 之间。	`?blur=20`
gamma	数字	否	应用伽玛校正以改善调整大小的图像的感知亮度。值必须在 1.0 到 3.0 之间。	`?gamma=1`
negate	布尔值	否	反转图像的颜色。	`?negate`
normalize	布尔值	否	通过拉伸其亮度以覆盖整个动态范围来增强图像对比度。	`?normalize`
threshold	数字	否	如果图像中的任何像素的强度小于指定的阈值，则将其替换为黑色像素。或者，如果它大于阈值，则使用白色像素。有效值在 0 到 255 之间。	`?threshold=155`
tint	字符串	否	使用提供的 RGB 对图像进行着色，同时保持图像的亮度。	`?tint=#7743CE`

查询参数	Type	必需	说明	示例
grayscale	布尔值	否	将图像转换为灰度（黑色和白色）。	`?grayscale`

响应状态代码

下表介绍了图像优化的响应状态代码。

Success – HTTP 状态代码 200

请求已成功完成。

BadRequest -HTTP 状态码 400

- 输入查询参数的指定不正确。
- 远程 URL 未在 `remotePatterns` 设置中列为允许。
- 远程 URL 无法解析为图像。
- `sizes` 设置中未将请求的宽度或高度列为允许值。
- 请求的图像是 SVG，但 `dangerouslyAllowSvg` 设置已禁用该格式。

Not Found – HTTP 状态代码 404

找不到源图像。

Content too large – HTTP 状态代码 413

源图像或优化的图像超过了允许的最大字节大小。

了解经优化的图像缓存

Amplify Hosting 会将经过优化的图像缓存在我们的 CDN 上，以便后续对具有相同查询参数的同一张图像的请求可以从缓存中获取。缓存存续时间（TTL）由 `Cache-Control` 标头控制。下表介绍了用于指定 `Cache-Control` 标头的选项。

- 在以图像优化为目标的路由规则中使用 `Cache-Control` 键。
- 使用 Amplify 应用程序中定义的自定义标头。
- 对于远程图像，将使用远程图像返回的 `Cache-Control` 标头。

图像优化设置中指定的 minimumCacheTTL 定义了 Cache-Control max-age 指令的下限。例如，如果远程图像 URL 以 Cache-Control s-max-age=10 响应，但值 minimumCacheTTL 为 60，则使用 60。

为任意 SSR 框架使用开源适配器

您可以使用为与 Amplify Hosting 集成而创建的任何 SSR 框架构建适配器。每个提供适配器的框架都决定了该适配器的配置方式以及如何连接到其构建过程。通常，您会将适配器作为 npm 开发依赖项进行安装。

使用框架创建应用程序后，请使用该框架的文档来学习如何安装 Amplify Hosting 适配器并在应用程序的配置文件中对其进行配置。

接下来，在项目的根目录中创建一个 amplify.yml 文件。在 amplify.yml 文件中，将 baseDirectory 设置为应用程序的构建输出目录。框架在构建过程中运行适配器，将输出转换为 Amplify Hosting 部署包。

构建输出目录的名称可以是任何名称，但 .amplify-hosting 文件名有具体意义。Amplify 首先会查找一个定义为 baseDirectory 的目录。如果存在，Amplify 会在那里查找构建输出。如果目录不存在，Amplify 会在 .amplify-hosting 中查找构建输出，即使客户尚未定义该输出也是如此。

以下是应用程序的构建设置示例。将 baseDirectory 设置为 .amplify-hosting 表示构建输出位于 .amplify-hosting 文件夹中。只要 .amplify-hosting 文件夹的内容符合 Amplify Hosting 部署规范，应用程序就会成功部署。

```
version: 1
frontend:
  preBuild:
    commands:
      - npm install
  build:
    commands:
      - npm run build
  artifacts:
    baseDirectory: .amplify-hosting
```

将您的应用程序配置为使用框架适配器后，您可以将其部署到 Amplify Hosting。有关详细说明，请参阅 [将 SSR 应用程序部署到 Amplify]

将 Amazon S3 存储桶中的静态网站部署到 Amplify

只需点击几下，您就可以利用 Amplify Hosting 与 Amazon S3 之间的集成，托管 S3 中存储的静态网站内容。部署到 Amplify Hosting 可为您提供以下优势和功能。

- 自动部署到由以下设备提供支持的全球可用 AWS 内容分发网络 (CDN) CloudFront
- HTTPS 支持
- 使用 Amplify 控制台轻松将您的网站连接到自定义域
- 自带自定义 SSL 证书
- 使用内置的访问日志和 CloudWatch 指标监控您的网站
- 为网站设置密码保护
- 在 Amplify 控制台中创建重定向与重写规则

您可以从 Amplify 控制台 AWS CLI、或. 开始部署过程。 AWS SDKs您只能从您自己账户中的 Amazon S3 通用存储桶部署到 Amplify。Amplify 不支持跨账户 S3 存储桶访问。

当您将应用程序从 Amazon S3 通用存储桶部署到 Amplify 托管时， AWS 费用将基于 Amplify 的定价模型。有关更多信息，请参阅AWS Amplify 定价。

> **Important**
>
> Amplify Hosting 并非在所有可用 Amazon S3 AWS 区域 的地方都可用。若要将静态网站部署到 Amplify Hosting，包含您的网站的 Amazon S3 通用存储桶必须位于可以使用 Amplify 的区域之中。有关可以使用 Amplify 的区域列表，请参阅《Amazon Web Services 一般参考》中的 Amplify 端点。

请参阅以下主题，了解如何将静态网站从 Amazon S3 部署和更新到 Amplify Hosting。

主题

- 使用 Amplify 控制台部署 S3 中存储的静态网站
- S3使用创建存储桶策略以部署静态网站 AWS SDKs
- 更新从 S3 存储桶部署到 Amplify 的静态网站
- 更新 S3 部署以使用存储桶和前缀，而非 .zip 文件

使用 Amplify 控制台部署 S3 中存储的静态网站

请按照以下说明操作，以使用 Amplify 控制台部署 Amazon S3 通用存储桶内存储的新静态网站。

使用 Amplify 控制台部署 Amazon S3 通用存储桶内存储的静态网站的方法

1. 登录 AWS 管理控制台 并打开 Amplify 控制台，网址为。https://console.aws.amazon.com/amplify/
2. 在所有应用程序页面中，选择创建新应用程序。
3. 在开始使用 Amplify 构建页面中，选择不使用 Git 部署。
4. 选择下一步。
5. 在开始手动部署页面上执行以下操作。
   a. 对于应用程序名称，输入您的应用程序的名称。
   b. 对于分支名称，输入需要部署的分支的名称。
6. 对于方法，选择 Amazon S3。
7. 对于要托管的对象的 S3 位置，选择浏览。选择要使用的 Amazon S3 通用存储桶，然后选择选择前缀。
8. 选择保存并部署。

S3使用创建存储桶策略以部署静态网站 AWS SDKs

您可以使用将静态网站从 Amazon S3 部署 AWS SDKs 到 Amplify Hosting。如果您使用软件开发工具包部署网站，则必须创建自己的存储桶策略，以授予 Amplify Hosting 检索S3 存储桶中对象的权限。

有关如何创建存储桶策略的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的适用于 Amazon S3 的存储桶策略。

以下示例存储桶策略授予 Amplify Hosting 列出存储分区和检索指定的 AWS 账户 Amplify 应用程序 ID 和分支的存储桶对象的权限。

要使用此示例，请执行以下操作：

- *amzn-s3-demo-website-bucket/prefix*替换为您网站存储桶的名称和前缀。
- *111122223333*用你的 AWS 账户 身份证替换。
- *region-id*替换为 Amplify 应用程序所在的，例如。 AWS 区域 **us-east-1**

- *app_id*用你的 Amplify 应用程序 ID 替换。可在 Amplify 控制台中获得此信息。
- *branch_name*替换为您的分支名称。

> **Note**
>
> 在您的存储桶策略中，aws:SourceArn 必须是经过 URL 编码（百分比编码）的分支 ARN。

JSON

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

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

更新从 S3 存储桶部署到 Amplify 的静态网站

如果您为 Amplify 上托管的通用 S3 存储桶中的静态网站更新了任何对象，则必须将应用程序重新部署到 Amplify Hosting，这样才能使更改生效。Amplify Hosting 不会自动检测对 S3 存储桶的更改。我们建议您使用 AWS Command Line Interface （CLI）更新网站。

将更新同步到 S3

对网站的项目文件进行更改后，使用以下 s3 sync 命令，将您对本地源目录作出的更改与目标 Amazon S3 通用存储桶同步。要使用此示例，请*<source>*替换为本地目录的*<target>*名称和 Amazon S3 存储桶的名称。

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

将网站重新部署到 Amplify Hosting

使用以下 amplify start-deployment 命令，将 Amazon S3 存储桶中经过更新的应用程序重新部署到 Amplify Hosting。要使用此示例，请*<app_id>*替换为您的 Amplify 应用程序的 ID、*<branch_name>*分支的名称以及*s3://amzn-s3-demo-website-bucket/prefix*存储S3桶和前缀。

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

更新 S3 部署以使用存储桶和前缀，而非 .zip 文件

如果您已通过 Amazon S3 通用存储桶中的 .zip 文件将现有静态网站部署到 Amplify Hosting，则可以更新应用程序部署，以使用包含待托管对象的存储桶名称和前缀。这种类型的部署无需将包含构建输出压缩内容的单独文件上传到存储桶。

将静态网站从 .zip 文件迁移到存储桶内容

1. 登录 AWS 管理控制台 并打开 Amplify 控制台，网址为。https://console.aws.amazon.com/amplify/
2. 在所有应用程序页面上，选择要从使用 .zip 文件迁移到直接使用应用程序文件的手动部署应用程序的名称。
3. 在应用程序的概览页面上，选择部署更新。
4. 对于部署更新页面上的方法，选择 Amazon S3。
5. 对于要托管的对象的 S3 位置，选择浏览。选择要使用的存储桶，然后选择选择前缀。
6. 选择保存并部署。

在无 Git 存储库的情况下将应用程序部署到 Amplify

通过手动部署，您可以在不连接 Git 提供商的情况下，用 Amplify Hosting 发布您的 Web 应用程序。您可以从桌面拖放 ZIP 压缩文件夹，然后在几秒钟内即可托管您的网站。或者，您可以引用 Amazon S3 存储桶中的资产或指定文件存储位置的公共网址。

> **Note**
>
> 由于 Amazon S3 复制操作限制的原因，手动部署的最大 .zip 文件大小限制为 5GB。如果您有任何构建超过此大小，请考虑将其分解为较小的存档文件或使用其他部署方法。

对于 Amazon S3，您还可以设置 AWS Lambda 触发器，以便在每次上传新资产时更新您的网站。有关设置此场景的更多详细信息，请参阅将存储在 Amazon S3、Dropbox 或桌面上的文件部署到 AWS Amplify 控制台博客文章。

Amplify Hosting 不支持手动部署服务器端渲染 (SSR) 应用程序。有关更多信息，请参阅 使用 Amplify Hosting 部署在服务器端渲染的应用程序。

拖放式手动部署

使用拖放方式手动部署应用程序

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 在右上角，选择创建新应用程序。
3. 在开始使用 Amplify 构建页面中，选择不使用 Git 部署。然后选择下一步。
4. 在开始手动部署页面的应用程序名称中，输入您的应用程序的名称。
5. 在分支名称中，输入一个有意义的名称，例如 **`development`** 或 **`production`**。
6. 在方法中，选择拖放。
7. 您可以将文件夹从桌面拖放到拖放区域，也可以使用选择 .zip 文件夹从计算机中选择文件。您拖放或选择的文件必须是包含构建输出内容的 zip 压缩文件夹。
8. 选择保存并部署。

Amazon S3 或 URL 式手动部署

> **Note**
>
> 如果您从 S3 部署静态网站，则以下过程要求您将包含构建输出内容的 zip 压缩文件夹上传到 S3 存储桶。我们建议您使用存储桶名称和前缀直接从 S3 部署静态网站。有关此简化过程的更多信息，请参阅将 Amazon S3 存储桶中的静态网站部署到 Amplify。

从 Amazon S3 或公共网址手动部署应用程序

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 在右上角，选择创建新应用程序。
3. 在开始使用 Amplify 构建页面中，选择不使用 Git 部署。然后选择下一步。
4. 在开始手动部署页面的应用程序名称中，输入您的应用程序的名称。
5. 在分支名称中，输入一个有意义的名称，例如 **`development`** 或 **`production`**。
6. 对于方法，选择 Amazon S3 或任何网址。
7. 上传文件的过程取决于上传方法。
   - Amazon S3
     a. 对于 S3 location of objects to host，选择浏览 S3。随后从列表中选择 Amazon S3 存储桶的名称。必须为您选择的存储桶启用访问控制列表 (ACLs)。有关更多信息，请参阅 手动部署的 Amazon S3 存储桶访问权限问题排查。
     b. 选择要部署的 .zip 文件的名称。
     c. 选择选择前缀。
   - 任何网址
     - 在资源网址中，输入要部署的 .zip 文件的网址。
8. 选择保存并部署。

> **Note**
>
> 创建 zip 压缩文件夹时，请务必压缩构建输出的内容，而不是压缩顶级文件夹。例如，如果您的生成输出生成了一个名为 build 或 public 的文件夹，请先导航到该文件夹，选择所有内容，

然后从那里压缩。如果不这样做，您将看到“访问被拒绝”错误，因为无法正确初始化站点的根目录。

手动部署的 Amazon S3 存储桶访问权限问题排查

创建 Amazon S3 存储桶时，您可以使用其 Amazon S3 对象所有权设置来控制该存储桶是启用还是禁用访问控制列表 (ACLs)。要将应用程序从 Amazon S3 存储桶手动部署到 Amplify， ACLs 必须在该存储桶上启用。

如果您在从 Amazon S3 存储桶部署时`AccessControlList`遇到错误，则表示该存储桶是在 ACLs 禁用状态创建的，您必须在 Amazon S3 控制台中将其启用。有关说明，请参阅 Amazon Simple Storage Service 用户指南中的在现有存储桶上设置对象所有权。

管理 Amplify 应用程序的构建配置

您可以自定义 Amplify 部署的构建设置和配置。部署应用程序时，Amplify 会自动检测前端框架和相关构建设置。您可以在应用程序的构建规范（buildspec）中自定义构建设置，以添加环境变量、运行构建命令和指定构建依赖项。

Amplify 的默认构建映像预装了多个软件包和依赖项，但您也可以使用实时软件包更新功能指定特定版本，或者始终确保安装了最新版本。如果您在使用 Amplify 的默认容器进行构建期间需要花很长时间来安装特定的依赖项，则可以创建自己的自定义构建映像。您还可以自定义构建实例的大小，从而为应用程序部署提供所需的 CPU、内存和磁盘空间资源。

每次提交到 Git 存储库以及每次新部署时，都会自动启动构建。您可以设置传入 Webhook 功能来启动构建，而不提交到 Git 存储库。

借助构建通知功能，您可以向团队成员分享有关构建成功和失败的信息。

主题

配置 Amplify 应用程序的构建设置

当您部署应用程序时，Amplify 会检查 Git 存储库中的应用程序 `package.json` 文件，以自动检测前端框架和相关的构建设置。您可以使用以下选项来存储应用程序构建设置：

- 在 Amplify 控制台中保存构建设置 – Amplify 控制台会自动检测构建设置并将其保存，以便可以通过 Amplify 控制台访问它们。Amplify 会将这些设置应用于所有分支，除非存储库中存储有 `amplify.yml` 文件。
- 将构建设置保存在您的存储库中：下载 `amplify.yml` 文件并将其添加到存储库的根位置。

> **Note**
>
> 只有将应用程序设置为持续部署并连接到 git 存储库时，构建设置才会显示在 Amplify 控制台的托管菜单中。有关此类部署的说明，请参阅开始使用。

构建规范参考

Amplify 应用程序的构建规范（buildspec）是 Amplify 用来运行构建的 YAML 设置和构建命令集合。下表描述了这些设置及其使用方式。

版本

Amplify YAML 版本号。

appRoot

此应用程序所在存储库中的路径。除非定义了多个应用程序，否则忽略。

env

向此部分中添加环境变量。您还可以使用控制台添加环境变量。

后端

在持续部署过程中，运行 Amplify CLI 命令来预置后端、更新 Lambda 函数或 GraphQL 架构。

frontend

运行前端构建命令。

测试

在测试阶段运行命令。了解如何为应用程序添加测试。

构建阶段

前端和后端都具有三个阶段，表示在每个构建序列中运行的命令。

- preBuild：预构建脚本在实际构建启动之前、Amplify 安装依赖项之后运行。
- build – 您的构建命令。
- postBuild – 构建后脚本在构建已完成并且 Amplify 已将所有必要项目都复制到输出目录后运行。

buildpath

用于运行构建程序的路径。Amplify 使用此路径来查找您的构建构件。如果您没有指定路径，Amplify 会使用单一存储库应用程序根目录，例如 `apps/app`。

artifacts>base-directory

您的构建构建所位于的目录。

artifacts>files

指定要部署的构件中的文件。输入 **/* 以包含所有文件。

cache

指定构建时的依赖项，例如 node_modules 文件夹。在首次构建期间，此处提供的路径将被缓存。在后续构建中，Amplify 会在运行您的命令之前将缓存恢复到相同的路径。

Amplify 认为所有提供的缓存路径都是相对于项目根目录的。但是，Amplify 不允许在项目根目录之外进行遍历。例如，假设您指定了某个绝对路径，则构建将会成功且不会出现错误，但不会缓存该路径。

构建规范的 YAML 语法参考

以下构建规范示例演示了基本的 YAML 语法。

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

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

编辑构建规范

您可以在 Amplify 控制台中编辑构建规范（buildspec），从而自定义应用程序的构建设置。这些构建设置将应用于您的应用程序中的所有分支，在 Git 存储库中保存了 `amplify.yml` 文件的分支除外。

在 Amplify 控制台中编辑构建设置

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要为其编辑构建设置的应用程序。
3. 在导航窗格中，依次选择托管和构建设置。
4. 在构建设置页面的应用程序构建规范部分中选择编辑。

5. 在编辑构建规范窗口中输入您的更新。
6. 选择保存。

您可以使用以下主题中所述示例来为特定场景更新构建设置。

主题

使用脚本设置特定于分支的构建设置

您可以使用 bash shell 脚本设置特定于分支的构建设置。例如，如果分支名称为 main，则以下脚本使用系统环境变量 $ AWS_BRANCH 运行一组命令；如果分支名称为 dev，则使用另一组命令。

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

设置命令以导航到子文件夹

对于单一存储库，用户希望能够通过 cd 命令进入文件夹以运行构建。运行 cd 命令后，它将应用于构建的所有阶段，因此您无需在单独的阶段中重复该命令。

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

为 Gen 1 应用程序部署带有前端的后端

> **Note**
>
> 此部分仅适用于 Amplify Gen 1 应用程序。Gen 1 后端使用 Amplify Studio 和 Amplify 命令行界面（CLI）创建。

该 `amplifyPush` 命令是一个帮助程序脚本，可以帮助您进行后端部署。下面的构建设置自动为当前分支确定正确的待部署后端环境。

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

设置输出文件夹

以下构建设置将输出目录设置为公用文件夹。

```
frontend:
  phases:
```

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

在构建过程中安装软件包

您可以在构建过程中使用 npm 或 yarn 命令来安装软件包。

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

使用私有 npm 注册表

您可以在构建设置中添加对私有注册表的引用或将其添加为环境变量。

```
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
```

安装操作系统软件包

Amplify 的 AL2023 图像使用名为的非特权用户运行您的代码。amplifyAmplify 授予此用户使用 Linux sudo 命令运行操作系统命令的权限。如果要为缺失的依赖项安装操作系统软件包，则可以使用 yum 和 rpm 及 sudo 等命令。

以下示例构建部分演示了使用 sudo 命令安装操作系统程序包的语法。

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

为每个构建设置键值存储

envCache 在构建时提供键/值存储。envCache 中存储的值只能在构建期间进行修改并可在下次构建时重复使用。使用 envCache，我们可以存储有关已部署环境的信息并将其提供给连续构建中的构建容器。与存储在 envCache 中的值不同，构建期间对环境变量的更改不会持久保存，无法在将来的构建中使用。

示例用法：

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

为某个提交跳过构建

要跳过对特定提交的自动构建，请在提交消息的末尾添加文本 [skip-cd]。

关闭每次提交的自动构建

您可以配置 Amplify，以停用在每次代码提交时自动构建。要进行设置，请选择应用程序设置、分支设置，然后找到列出所有已连接分支的分支部分。选择一个分支，然后选择操作、禁用自动构建。以该分支为目标的新提交操作将不再启动新的构建。

配置基于 diff 的前端构建和部署

您可以将 Amplify 配置为使用基于 diff 的前端构建。如果启用，则默认情况下，Amplify 会在每次构建开始时尝试对您的 appRoot 或 /src/ 文件夹运行 diff。如果 Amplify 未发现任何差异，它将跳过前端构建、测试（如果已配置）和部署步骤，并且不会更新托管的应用程序。

配置基于 diff 的前端构建和部署

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要为其配置基于 diff 的前端构建和部署的应用程序。
3. 在导航窗格中，依次选择托管和环境变量。

4. 在环境变量部分中，选择管理变量。
5. 配置环境变量的过程会有所不同，具体取决于您是启用还是禁用基于 diff 的前端构建和部署。
   - 启用基于 diff 的前端构建和部署
     a. 在管理变量部分，请在变量下输入 AMPLIFY_DIFF_DEPLOY。
     b. 对于值，请输入 true。
   - 禁用基于 diff 的前端构建和部署
     - 请执行以下操作之一：
       - 在管理变量部分，找到 AMPLIFY_DIFF_DEPLOY。对于值，请输入 false。
       - 删除 AMPLIFY_DIFF_DEPLOY 环境变量。
6. 选择保存。

或者，您可以将环境变量 AMPLIFY_DIFF_DEPLOY_ROOT 设置为使用存储库根目录的相对路径来覆盖默认路径，例如 dist。

为 Gen 1 应用程序配置基于 diff 的后端构建

> **Note**
> 此部分仅适用于 Amplify Gen 1 应用程序。Gen 1 后端使用 Amplify Studio 和 Amplify 命令行界面（CLI）创建。

您可以使用环境变量 AMPLIFY_DIFF_BACKEND 将 Amplify Hosting 配置为使用基于 diff 的后端构建。当您启用基于 diff 的后端构建时，Amplify 会在每次构建开始时尝试对存储库中的 amplify 文件夹运行 diff。如果 Amplify 没有发现任何差异，它将跳过后端构建步骤，并且不会更新您的后端资源。如果您的项目存储库中没有 amplify 文件夹，Amplify 会忽略 AMPLIFY_DIFF_BACKEND 环境变量的值。

如果您当前在后端阶段的构建设置中指定了自定义命令，则有条件的后端构建将不起作用。如果要运行这些自定义命令，则必须将其移至应用程序 amplify.yml 文件中构建设置的前端阶段。

配置基于 diff 的后端构建

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要为其配置基于 diff 的后端构建的应用程序。
3. 在导航窗格中，依次选择托管和环境变量。

4. 在环境变量部分中，选择管理变量。
5. 配置环境变量的过程会有所不同，具体取决于您是启用还是禁用基于 diff 的后端构建。
   - 启用基于 diff 的后端构建
     a. 在管理变量部分，请在变量下输入 AMPLIFY_DIFF_BACKEND。
     b. 对于值，请输入 true。
   - 禁用基于 diff 的后端构建
     - 请执行以下操作之一：
       - 在管理变量部分，找到 AMPLIFY_DIFF_BACKEND。对于值，请输入 false。
       - 删除 AMPLIFY_DIFF_BACKEND 环境变量。
6. 选择保存。

配置 monorepo 构建设置

当您在单个存储库中存储多个项目或微服务时，存储库称为单一存储库。您可以使用 Amplify Hosting 在单一存储库中部署应用程序，而无需创建多个构建配置或分支配置。

Amplify 支持通用单一存储库中的应用程序，以及使用 npm 工作空间、pnpm 工作空间、Yarn 工作空间、Nx 和 Turborepo 创建的单一存储库中的应用程序。当部署应用程序时，Amplify 会自动检测您所使用的单一存储库构建工具。Amplify 会自动为 npm 工作空间、Yarn 工作空间或 Nx 中的应用程序应用构建设置。Turborepo 和 pnpm 应用程序需要额外的配置。有关更多信息，请参阅 配置 Turborepo 和 pnpm 单一存储库应用程序。

您可以在 Amplify 控制台中保存单一存储库的构建设置，也可以下载 amplify.yml 文件并将其添加到存储库的根目录中。Amplify 会将保存在控制台中的设置应用于您的所有分支，除非它在您的存储库中找到了 amplify.yml 文件。当 amplify.yml 文件存在时，其设置会覆盖 Amplify 控制台中保存的所有构建设置。

Monorepo 构建规范的 YAML 语法参考

单一存储库构建规范的 YAML 语法不同于含有单个应用程序的存储库的 YAML 语法。对于单一存储库，您可以在应用程序列表中声明各个项目。您必须为在单一存储库构建规范中声明的各个应用程序提供以下附加 appRoot 密钥：

appRoot

应用程序启动所在的存储库中的根目录。此键必须存在，且其值须与 AMPLIFY_MONOREPO_APP_ROOT 环境变量相同。有关设置此环境变量的说明，请参阅 设置 AMPLIFY_MONOREPO_APP_ROOT 环境变量。

以下单一存储库构建规范示例演示了如何在同一个存储库中声明多个 Amplify 应用程序。applications 列表中声明了两个应用程序，即 react-app 和 angular-app。每个应用程序的 appRoot 键表示该应用程序位于存储库的 apps 根文件夹中。

buildpath 属性已设置为 /，以从单一存储库项目根目录运行和构建应用程序。baseDirectory 属性是 buildpath 的相对路径。

单一存储库构建规范的 YAML 语法

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

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

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

使用以下示例构建规范的应用程序将在项目根目录下构建，相关构件将位于 /packages/nextjs-app/.next 中。

```
applications:
  - frontend:
      buildPath: '/'  # run install and build from monorepo project root
      phases:
```

```
        preBuild:
          commands:
            - npm install
        build:
          commands:
            - npm run build --workspace=nextjs-app
      artifacts:
        baseDirectory: packages/nextjs-app/.next
        files:
          - '**/*'
      cache:
        paths:
          - node_modules/**/*
    appRoot: packages/nextjs-app
```

设置 AMPLIFY_MONOREPO_APP_ROOT 环境变量

部署存储在单一存储库中的应用程序时，该应用程序的 AMPLIFY_MONOREPO_APP_ROOT 环境变量必须与应用程序根目录路径（存储库根目录的相对路径）具有相同的值。例如，一个单一存储库 ExampleMonorepo 的根文件夹名为 apps，其中包含 app1、app2 和 app3，其目录结构如下：

```
ExampleMonorepo
  apps
    app1
    app2
    app3
```

在此示例中，app1 的 AMPLIFY_MONOREPO_APP_ROOT 环境变量的值为 apps/app1。

当您使用 Amplify 控制台部署单一存储库应用程序时，控制台会使用您为应用程序根目录路径指定的值自动设置 AMPLIFY_MONOREPO_APP_ROOT 环境变量。但是，如果您的 monorepo 应用程序已存在于 Amplify 中或者是使用部署的 AWS CloudFormation，则必须在 Amplify 控制台的 “AMPLIFY_MONOREPO_APP_ROOT环境变量” 部分中手动设置环境变量。

在部署期间自动设置 AMPLIFY_MONOREPO_APP_ROOT 环境变量

以下说明演示了如何使用 Amplify 控制台部署单一存储库应用程序。Amplify 使用您在控制台中指定的应用程序根文件夹自动设置 AMPLIFY_MONOREPO_APP_ROOT 环境变量。

使用 Amplify 控制台部署单一存储库应用程序

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。

2. 在右上角选择创建新应用程序。
3. 在开始使用 Amplify 进行构建页面中选择您的 Git 提供商，然后选择下一步。
4. 在添加存储库分支页面上，执行以下操作：
   a. 从列表中选择您的存储库名称。
   b. 选择要使用的分支名称。
   c. 选择我的应用程序是 monorepo
   d. 在单一存储库中输入您应用程序的路径，例如 **apps/app1**。
   e. 选择下一步。
5. 在应用程序设置页面上，您可以使用默认设置，也可以自定义应用程序的构建设置。在环境变量部分中，Amplify 会将 AMPLIFY_MONOREPO_APP_ROOT 设置为您在步骤 4d 中指定的路径。
6. 选择下一步。
7. 在查看页面上，选择保存并部署。

为现有应用程序设置 AMPLIFY_MONOREPO_APP_ROOT 环境变量

按照以下说明为已部署到 Amplify 或使用创建的应用程序手动设置AMPLIFY_MONOREPO_APP_ROOT环境变量。 CloudFormation

为现有应用程序设置 AMPLIFY_MONOREPO_APP_ROOT 环境变量

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要为其设置环境变量的应用程序的名称。
3. 在导航窗格中，依次选择托管和环境变量。
4. 在环境变量页面，选择管理变量。
5. 在变量管理器部分，执行以下操作：
   a. 选择新增。
   b. 对于变量，请输入密钥 AMPLIFY_MONOREPO_APP_ROOT。
   c. 对于值，请输入应用程序的路径，例如 **apps/app1**。
   d. 对于分支，默认情况下 Amplify 会将环境变量应用于所有分支。
6. 选择保存。

配置 Turborepo 和 pnpm 单一存储库应用程序

Turborepo 和 pnpm 工作空间单一存储库构建工具从 `.npmrc` 文件中获取配置信息。部署使用这些工具之一创建的单一存储库应用程序时，您的项目根目录中必须有一个 `.npmrc` 文件。

在 `.npmrc` 文件中，将用于安装 Node 软件包的链接器设置为 `hoisted`。您可以将以下一行复制到您的文件。

```
node-linker=hoisted
```

有关 `.npmrc` 文件和设置的更多信息，请参阅 pnpm 文档中的 pnpm.npmrc。

Pnpm 不包含在 Amplify 的默认构建容器中。对于 pnpm 工作空间和 Turborepo 应用程序，必须在应用程序构建设置的 `preBuild` 阶段添加一条安装 pnpm 的命令。

以下示例摘自构建规范，其中显示了一个包含 pnpm 安装命令的 `preBuild` 阶段。

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

自定义构建映像

您可以使用自定义构建映像，为 Amplify 应用程序提供自定义的构建环境。如果您在使用 Amplify 的默认容器进行构建期间需要花很长时间来安装特定的依赖项，则可以创建自己的 Docker 映像并在构建期间引用该映像。映像可以在公有 Amazon Elastic Container Registry 上托管。

要使自定义构建映像作为 Amplify 构建映像使用，它必须满足以下要求。

自定义构建映像要求

1. 支持 GNU C 库 (glibc) 的 Linux 发行版，例如 Amazon Linux，专为 x86-64 架构编译。
2. cURL：当我们启动您的自定义映像时，我们会将构建运行程序下载到您的容器中，因此我们需要有 cURL。如果缺少此依赖项，则构建将立即失败而没有任何输出，因为我们的构建运行程序无法生成任何输出。

3. Git：为了克隆您的 Git 存储库，我们要求在映像中安装 Git。如果缺少此依赖项，克隆存储库步骤将会失败。
4. OpenSSH：为了安全地克隆您的存储库，我们要求 OpenSSH 在构建期间临时设置 SSH 密钥。OpenSSH 包提供了构建运行程序执行此操作所需的命令。
5. Bash 和 Bourne Shell：这两个实用程序用于在构建时运行命令。如果未安装它们，您的构建可能会在开始之前失败。
6. Node.JS+NPM：我们的构建运行程序未安装 Node。相反，它依赖于安装在映象中的 Node 和 NPM。只有需要 NPM 程序包或节点特定命令的构建才需要这样做。但是，我们强烈建议安装它们，因为当它们存在时，Amplify 构建运行程序可以使用这些工具来改善构建执行。当您为 Hugo 设置覆盖时，Amplify 的包覆盖功能使用 NPM 来安装 Hugo 扩展包。

以下包不是必需的，但我们强烈建议您创建它。

1. NVM (Node Version Manager)：如果您需要处理不同版本的 Node，我们建议您安装此版本管理器。当您设置覆盖时，Amplify 的软件包覆盖功能会使用 NVM 在每次构建之前更改 Node.js 版本。
2. Wget：Amplify 可以在构建过程中使用 Wget 实用程序下载文件。我们建议您将其安装在您的自定义图像中。
3. Tar：Amplify 可以在构建过程中使用 Tar 实用程序解压缩下载的文件。我们建议您将其安装在您的自定义图像中。

为应用程序配置自定义构建映像

使用以下步骤，在 Amplify 控制台中为应用程序配置自定义构建映像。

配置托管在 Amazon ECR 中的自定义构建映像

1. 要使用 Docker 映像设置 Amazon ECR Public 存储库，请参阅 Amazon ECR Public 用户指南中的入门。
2. 登录 AWS 管理控制台 并打开 Amplify 控制台。
3. 选择要为其配置自定义构建映像的应用程序。
4. 在导航窗格中，依次选择托管和构建设置。
5. 在构建设置页面上，请在构建映像设置中选择编辑。
6. 在编辑构建映像设置页面中，展开构建映像菜单，然后选择自定义构建映像。

7. 输入您在第一步中创建的 Amazon ECR Public 存储库的名称。这就是您的构建映像的托管位置。例如，如果您的存储库的名称为 ecr-examplerepo，您可以输入 **`public.ecr.aws/xxxxxxxx/ecr-examplerepo`**。
8. 选择保存。

在构建映像中使用特定程序包和依赖项版本

利用实时程序包更新，您可以指定程序包版本和依赖项以便在 Amplify 的默认构建映像中使用。默认构建映像附带了几个预安装的程序包和依赖项（例如 Hugo、Amplify CLI、Yarn 等）。利用实时程序包更新，您可以覆盖这些依赖项的版本并指定特定版本，或者始终确保安装了最新版本。

如果启用了实时程序包更新，则在运行构建之前，构建运行程序会首先更新（或降级）指定的依赖项。这将与更新依赖项所花费的时间成比例增加构建时间，但好处是可以确保使用相同版本的依赖项来构建您的应用程序。

> **Warning**
>
> 将 Node.js 版本设置为最新会导致构建失败。相反，您必须指定确切的 Node.js 版本，例如 `18`、`21.5` 或 `v0.1.2`。

配置实时软件包更新

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要为其配置实时软件包更新的应用程序。
3. 在导航窗格中，依次选择托管和构建设置。
4. 在构建设置页面上，请在构建映像设置中选择编辑。
5. 在编辑构建映像设置页面的实时程序包更新列表中，选择新增。
6. 对于程序包，选择要覆盖的依赖项。
7. 对于版本，要么保留默认最新版本，要么输入依赖项的特定版本。如果使用最新，依赖项将始终升级到可用的最新版本。
8. 选择保存。

为 Amplify 应用程序配置构建实例

Amplify Hosting 提供可配置的构建实例大小，使您能够为应用程序的构建实例提供所需的 CPU、内存和磁盘空间资源。在此功能发布之前，Amplify 提供了 8 GiB 内存和 4 v 的固定大小的构建实例配置。CPUs

Amplify 支持三种构建实例类型：Standard、Large 和 XLarge。如果未指定实例类型，Amplify 将使用默认的 Standard 实例。您可以使用 Amplify 控制台 AWS CLI、或，为应用程序配置构建实例类型。SDKs

每种构建实例类型的成本均按构建分钟数计算。有关定价的详细信息，请参阅 AWS Amplify 定价。

下表描述了每种构建实例类型的计算规范：

构建实例类型	v CPUs	内存	磁盘空间
Standard	4 v CPUs	8 GiB	128 GB
Large	8 v CPUs	16 GiB	128 GB
XLarge	36 v CPUs	72 GiB	256GB

主题

了解构建实例类型

构建实例类型设置是在应用程序级别配置的，并适用于应用程序的所有分支。构建实例类型的关键细节如下：

- 您为应用程序配置的构建实例类型会自动适用于自动创建的分支和拉取请求预览。
- 并发任务服务配额适用于您的所有构建实例类型 AWS 账户。例如，假设您的并发作业上限为五个，则在您的 AWS 账户中最多可以为所有实例类型运行 5 个构建作业。

- 每种构建实例类型的成本均按构建分钟数计算。构建实例分配过程可能需要额外的开销时间，然后才能开始构建。特别是对于较大的实例 XLarge，由于这种开销时间，您的构建可能会在构建开始之前遇到延迟。不过您只需按实际构建时间付费，无需为开销时间付费。

您可以在创建新应用程序时配置构建实例类型，也可以为现有应用程序更新实例类型。有关在 Amplify 控制台中配置此设置的说明，请参阅在 Amplify 控制台中配置构建实例类型。您也可以使用更新此设置 SDKs。有关更多信息，请参阅 Amplify API 参考UpdateApp APIs 中的CreateApp、和。

如果您账户的现有应用程序是在可自定义构建实例类型功能发布之前创建的，则这些应用程序使用默认的 Standard 实例类型。更新现有应用程序的构建实例类型时，任何在更新之前已加入队列或正在运行的构建都将使用先前配置的构建实例类型。例如，假设您的一个现有应用程序已将其 `main` 分支部署到 Amplify，然后您将该应用程序的构建实例类型从标准更新为大型，则您从 `main` 分支启动的所有新构建都将使用大型构建实例类型。但您在更新构建实例类型时正在运行的任何构建，都将继续使用标准实例运行。

在 Amplify 控制台中配置构建实例类型

创建新的 Amplify 应用程序时，请使用以下过程配置构建实例类型。

为新应用程序配置构建实例类型

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 在所有应用程序页面中，选择创建新应用程序。
3. 在开始使用 Amplify 进行构建页面中选择您的 Git 存储库提供商，然后选择下一步。
4. 在添加存储库分支页面上，执行以下操作：

   a. 在最近更新的存储库列表中，选择要连接的存储库的名称。

   b. 在分支列表中，选择要连接的存储库分支的名称。

   c. 选择下一步。
5. 在应用程序设置页面中打开高级设置部分。
6. 对于构建实例类型，从列表中选择所需的实例类型。
7. 如果要部署基于 Node.js 运行时的应用程序，请配置堆内存大小以有效利用大型实例类型。您可以通过在应用程序设置页面上设置环境变量或更新构建设置来完成此操作。有关更多信息，请参阅配置应用程序的堆内存以利用大型实例类型。

   - 设置环境变量

     a. 在高级设置的环境变量部分，选择新增。

b. 对于键，输入 **NODE_OPTIONS**。

c. 对于值，请输入 `--max-old-space-size=`*memory_size_in_mb*。*memory_size_in_mb*替换为所需的堆内存大小（以兆字节为单位）。

- 更新构建设置

a. 在设置部分中，选择编辑 YML 文件。

b. 将以下命令添加到 preBuild 阶段。*memory_size_in_mb*替换为所需的堆内存大小（以兆字节为单位）。

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

c. 选择保存。

8. 选择下一步。
9. 在查看页面上，选择保存并部署。

按以下过程为现有应用程序配置构建实例类型。

为现有应用程序配置构建实例类型

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要为其配置构建实例类型的应用程序。
3. 在导航窗格中，依次选择 Hosting、构建设置。
4. 在构建设置页面的高级设置部分中，选择编辑。
5. 在编辑设置中，对于构建实例类型，从列表中选择所需的实例类型。
6. 选择保存。此更改将在您下次部署应用程序时生效。
7. (可选）要立即部署更新后的应用程序，请执行以下操作：

a. 在导航窗格中，选择概述。

b. 在应用程序的概述页面上，选择要重新部署的分支。

c. 在部署页面上，选择一个部署，例如最近的部署。然后选择重新部署此版本。这时将会开始新的部署。

d. 部署完成后，应用程序的构建设置将显示该分支正在使用更新后的构建实例类型。

配置应用程序的堆内存以利用大型实例类型

如果要构建内存密集型应用程序，请按照本节的说明了解如何配置应用程序以使用大型实例类型。编程语言和框架通常依赖于在运行时分配动态内存（也称为堆内存）来管理应用程序的内存需求。堆内存由运行时环境请求并由主机操作系统分配。默认情况下，运行时环境会强制执行应用程序可用的最大堆大小限制。这意味着，即使主机操作系统或容器的可用内存量更大，也不会为应用程序提供超出堆大小的额外内存。

例如， JavaScript Node.js v8 运行时环境强制使用默认堆大小限制，该限制取决于多个因素，包括主机内存大小。Standard 和 Large 构建实例的默认 Node.js 堆大小为 2096 MB，XLarge 实例的默认堆大小为 4144 MB。因此，在任何 Amplify 编译实例类型上使用默认 Node.js 堆大小构建内存要求为 6000 MB 的应用程序都将由于错误而导致构建失败。 out-of-memory

要解决 Node.js 默认堆大小的内存限制，您可以执行以下操作之一：

- 将 Amplify 应用程序中 `NODE_OPTIONS` 环境变量设置为值 `--max-old-space-size=`*`memory_size_in_mb`*。对于 `memory_size_in_mb`，请指定所需的堆内存大小（以兆字节为单位）。

  有关说明，请参阅设置环境变量。
- 将以下命令添加到 Amplify 应用程序构建规范的 `preBuild` 阶段。

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

  您可以在 Amplify 控制台中更新构建规范，也可在项目存储库中该应用程序的 `amplify.yml` 文件中更新。有关说明，请参阅配置 Amplify 应用程序的构建设置。

  以下示例 Amplify 构建规范将 Node.js 堆内存大小设置为 7000 MB，用于构建一个 React 前端应用程序：

  ```
  version: 1
  frontend:
    phases:
      preBuild:
        commands:
          # Set the heap size to 7000 MB
          - export NODE_OPTIONS='--max-old-space-size=7000'
          # To check the heap size memory limit in MB
          - node -e "console.log('Total available heap size (MB):',
   v8.getHeapStatistics().heap_size_limit / 1024 / 1024)"
  ```

```
        - npm ci --cache .npm --prefer-offline
    build:
      commands:
        - npm run build
  artifacts:
    baseDirectory: build
    files:
      - '**/*'
  cache:
    paths:
      - .npm/**/*
```

要有效利用大型实例类型，必须配置足够的堆内存大小。如果为内存密集型应用程序配置较小的堆大小，则可能会导致构建失败。应用程序的构建日志可能不会直接指示 out-of-memory错误，因为应用程序运行时可能会意外崩溃。如果将堆大小配置为与主机内存一样大，则可能会导致主机操作系统交换或终止其他进程，并可能导致构建过程中断。作为参考，Node.js 建议在内存约为 2000 MB 的计算机上将最大堆大小设置为 1536 MB，以便为其他用途预留一些内存。

最佳堆大小取决于应用程序的需求和资源使用情况。如果遇到 out-of-memory错误，请从适度的堆大小开始，然后根据需要逐渐增加堆大小。作为指导原则，我们建议将 Standard 实例类型的起始堆大小设为 6000 MB，将 Large 实例类型的起始堆大小设为 12000 MB，将 XLarge 实例类型的起始堆大小设为 60000 MB。

创建传入 Webhook 以开始构建

在 Amplify 控制台中设置传入 Webhook 以开始构建，而无需将代码提交到 Git 存储库。您可以将无外设 CMS 工具（例如 Contentful 或 GraphCMS）与 Webhook 结合使用，以触发内容更改时的构建，或者使用 Zapier 等服务执行每日构建。

创建传入 webhook

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要为其创建 webhook 的应用程序。
3. 在导航窗格中，依次选择托管和构建设置。
4. 在构建设）页面上，向下滚动到传入 webhook 部分，然后选择创建 webhook。
5. 在创建 webhook 对话框中，执行以下操作：

   a. 在 webhook 名称中输入 webhook 的名称。

b. 在要构建的分支中，选择要在传入 webhook 请求基础上构建的分支。

c. 选择创建 Webhook。

6. 在 传入 webhook 部分中，执行下列操作之一：

   - 复制 Webhook 网址并将其提供给无外设 CMS 工具或其他服务，以发起构建。
   - 在终端窗口中运行 curl 命令，以开始新构建。

为构建设置电子邮件通知

您可以为 AWS Amplify 应用程序设置电子邮件通知，以便在构建成功或失败时提醒利益相关者或团队成员。Amplify Hosting 在您的账户中创建一个 Amazon Simple Notification Service (SNS) 主题，并使用它来配置电子邮件通知。可以将通知配置为应用于 Amplify 应用的所有分支或特定分支。

设置电子邮件通知

使用以下步骤为 Amplify 应用的所有分支或特定分支设置电子邮件通知。

为 Amplify 应用程序设置电子邮件通知

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要为其设置电子邮件通知的应用程序。
3. 在导航窗格中选择托管、构建通知。在构建通知页面上，选择管理通知。
4. 在管理通知页面上选择新增。
5. 请执行以下操作之一：

   - 要为单个分支发送通知，请在电子邮件中输入要向其发送通知的电子邮件地址。对于分支，请选择要为其发送通知的分支的名称。
   - 要为所有已连接的分支发送通知，请在电子邮件中输入要向其发送通知的电子邮件地址。对于分支，请选择所有分支。

6. 选择保存。

连接自定义域

您可以将通过 Amplify Hosting 部署的应用程序连接到自定义域。当您使用 Amplify 部署 Web 应用程序时，Amplify 会在默认 `amplifyapp.com` 域上为您托管此应用程序，并使用 `https://branch-name.d1m7bkiki6tdw1.amplifyapp.com` 这样的 URL。当您将应用程序连接到自定义域时，用户会看到您的应用程序托管在自定义网址上，例如 `https://www.example.com`。

您可以通过认可的域名注册商（例如 Amazon Route 53 或）购买自定义域名 GoDaddy。Route 53 是 Amazon 的域名系统 (DNS) Web 服务。有关使用 Route 53 的更多信息，请参阅什么是 Amazon Route 53？。要获取认可的第三方注册商的列表，请参阅 ICANN 网站上的获得认可的注册商目录。

设置自定义域时，您可以使用 Amplify 为您预配的默认托管证书，也可以使用您自己的自定义证书。您可以随时更改为域使用的证书。有关管理证书的详细信息，请参阅使用 SSL/TLS 证书。

在继续设置自定义域之前，请确认您已满足以下先决条件。

- 您拥有注册的域名。
- 您拥有由颁发或导入的证书 AWS Certificate Manager。
- 您已将应用程序部署到 Amplify Hosting。

  有关完成此步骤的更多信息，请参阅开始将应用程序部署到 Amplify Hosting。
- 您具备域和 DNS 术语的基础知识。

  有关域和 DNS 的更多信息，请参阅 了解 DNS 术语和概念。

> **Warning**
>
> 当使用已经或以前与同一地区其他 AWS 账户中的不同 Amplify 应用程序关联的域名发起对 Amplify 应用程序的 DomainAssociation 请求时，这被视为跨账户域关联。跨账户域关联请求需要手动验证。如果您想继续进行跨账户域名关联，请联系 AWS 支持寻求帮助。

主题

- 添加由第三方 DNS 提供商管理的自定义域
- 更新由管理的域的 DNS 记录 GoDaddy
- 更新域的 SSL/TLS 证书
- 管理子域
- 设置通配符子域
- 为 Amazon Route 53 自定义域设置自动子域
- 自定义域问题排查

了解 DNS 术语和概念

如果不熟悉与域名系统 (DNS) 相关的术语和概念，以下主题可以帮助您了解添加自定义域的程序。

DNS 术语

以下是 DNS 常用术语列表。它们可以帮助您了解添加自定义域的程序。

CNAME

别名记录 (CNAME) 是一种 DNS 记录，用于掩盖一组网页的域名，使其看起来像是位于其他地方。别名记录将子域指向完全限定域名 (FQDN)。例如，您可以创建一个新的别名记录以将子域 www.example.com（其中 www 是子域）映射到在 Amplify 控制台中分配给您应用程序的 FQDN 域 branch-name.d1m7bkiki6tdw1.cloudfront.net。

ANAME

ANAME 记录类似于 CNAME 记录，但位于根级别。ANAME 会将您的域的根目录指向 FQDN。上述 FQDN 指向一个 IP 地址。

名称服务器

名称服务器是 Internet 上的服务器，专门处理有关域名各种服务位置的查询。如果在 Amazon Route 53 中设置域，则您的域已被分配了名称服务器列表。

NS 记录

NS 记录指向查找您的域详情的名称服务器。

DNS 验证

域名系统 (DNS) 就像一本电话簿，它将用户可读的域名转换为便于计算机使用的 IP 地址。当您在浏览器中键入 **`https://google.com`** 时，会在 DNS 提供商处执行查找操作，以查找托管此网站的服务器的 IP 地址。

DNS 提供商包含域及其对应 IP 地址的记录。最常用的 DNS 记录是 CNAME、ANAME 和 NS 记录。

Amplify 使用别名记录验证您自己的自定义域。如果您使用 Route53 托管您的域，则系统会代表您进行验证。但是，如果您使用第三方提供商（例如）托管域名 GoDaddy，则必须手动更新域名的 DNS 设置并添加 Amplify 提供的新 CNAME 记录。

自定义域的激活流程

> **Warning**
>
> 当使用已经或以前与同一地区其他 AWS 账户中的不同 Amplify 应用程序关联的域名发起对 Amplify 应用程序的 DomainAssociation 请求时，这被视为跨账户域关联。跨账户域关联请求需要手动验证。如果您想继续进行跨账户域名关联，请联系 AWS 支持寻求帮助。

在 Amplify 控制台中将 Amplify 应用程序与自定义域连接时，Amplify 必须完成几个步骤，之后您才能使用自定义域查看您的应用程序。下表详细描述了域设置和激活过程中的每个步骤。

SSL/TLS 创建

如果您使用的是托管证书，请 AWS Amplify 颁发 SSL/TLS 证书以设置安全的自定义域。

SSL/TLS 配置和验证

在颁发托管证书之前，Amplify 将验证您是否是域的所有者。对于由 Amazon Route 53 管理的域，Amplify 会自动更新 DNS 验证记录。对于在 Route 53 以外管理的域，您必须通过第三方 DNS 提供商将 Amplify 控制台中提供的 DNS 验证记录手动添加到您的域中。

如果您使用自定义证书，则您需要负责验证域所有权。

域激活

域验证成功。对于在 Route 53 以外管理的域，您需要通过第三方 DNS 提供商将 Amplify 控制台中提供的 CNAME 别名记录手动添加到您的域中。

使用 SSL/TLS 证书

SSL/TLS 证书是一种数字文档，允许 Web 浏览器使用安全 SSL/TLS 协议识别和建立与网站的加密网络连接。设置自定义域时，您可以使用 Amplify 为您预配的默认托管证书，也可以使用您自己的自定义证书。

使用托管证书，Amplify 会为连接到您的应用程序的所有域名 SSL/TLS 颁发证书，以便通过 HTTPS/2 保护所有流量。由 AWS Certificate Manager (ACM) 生成的默认证书有效期为 13 个月，只要您的应用程序由 Amplify 托管，就会自动续订。

> **Warning**
>
> 如果您的域提供商在 DNS 设置中修改或删除了 CNAME 别名记录验证记录，Amplify 将无法续订证书。您必须在 Amplify 控制台中删除并重新添加该域。

要使用自定义证书，您必须首先从您自选的第三方证书颁发机构获取证书。Amplify Hosting 支持两种类型的证书：RSA（非对称加密算法）和 ECDSA（椭圆曲线数字签名算法）。每种证书类型都必须符合以下要求。

RSA 证书

- Amplify Hosting 支持 1024 位、2048 位、3072 位和 4096 位 RSA 密钥。
- AWS Certificate Manager (ACM) 颁发包含最多 2048 位密钥的 RSA 证书。
- 要使用 3072 位或 4096 位 RSA 证书，请从外部获取证书并将其导入到 ACM 之中。然后，它将可以与 Amplify Hosting 一起使用。

ECDSA 证书

- Amplify Hosting 支持 256 位密钥。
- 使用 prime256v1 椭圆曲线为 Amplify Hosting 获取 ECDSA 证书。

获得证书后，将其导入 AWS Certificate Manager。ACM 是一项服务，可让您轻松预置、管理和部署公有和私有 SSL/TLS 证书，以便与内部连接的资源一起 AWS 服务 使用。请确保您在美国东部（弗吉尼亚州北部）（us-east-1）区域请求或导入证书。

确保您的自定义证书涵盖您计划添加的全部子域。您可以在域名开头处使用通配符来覆盖多个子域。例如，如果您的域是 `example.com`，则可以包含通配符域 `*.example.com`。这将涵盖 `product.example.com` 和 `api.example.com` 等子域名。

在 ACM 中提供自定义证书后，您就可以在域名设置过程中选择此证书。有关将证书导入到 AWS Certificate Manager的说明，请参阅《AWS Certificate Manager 用户指南》中的将证书导入到 AWS Certificate Manager。

如果您在 ACM 中续订或重新导入自定义证书，Amplify 会刷新与自定义域关联的证书数据。对于导入的证书，ACM 不会自动管理续订。您负责续订并重新导入自定义证书。

您可以随时更改为一个域使用的证书。例如，您可以从默认托管证书切换到自定义证书，或从自定义证书更改为托管证书。此外，您可以将所用的自定义证书更改为其他自定义证书。有关更新证书的说明，请参阅更新域的 SSL/TLS 证书。

添加由 Amazon Route 53 管理的自定义域

Amazon Route 53 是一种可用性高、可扩展性强的 DNS 服务。有关更多信息，请参阅《Amazon Route 53 开发人员指南》中的 Amazon Route 53。如果您已经拥有 Route 53 域，请按照以下说明，将您的自定义域连接到 Amplify 应用程序。

添加由 Route 53 管理的自定义域

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要连接到自定义域的应用程序。
3. 在导航窗格中，依次选择托管、自定义域。
4. 在自定义域页面上，选择添加域。
5. 输入您的根域的名称。例如，如果您的域名是 https://example.com，请输入**`example.com`**。

   在您开始键入时，您已经在 Route 53 中管理的根域将出现在列表中。您可以从列表中选择需要使用的域。如果您还没有该域的所有权且该域可用，则可以在 Amazon Route 53 中购买该域。
6. 输入域名后，选择配置域。
7. 默认情况下，Amplify 会自动为您的域创建两个子域条目。例如，如果您的域名是 example.com，您将看到子域名，https://www.example.com并https://example.com设置了从根域名到 www 子域名的重定向。

   （可选）如果只想添加子域，可以修改默认配置。要更改默认配置，请从导航窗格中选择重写和重定向，随后配置您的域。

8. 选择要使用的 SSL/TLS 证书。您可以使用 Amplify 为您提供的默认托管证书，也可以使用已导入的自定义第三方证书。 AWS Certificate Manager
   - 使用默认 Amplify 托管证书。
     - 选择 Amplify 托管证书。
   - 使用自定义第三方证书。
     a. 选择自定义 SSL 证书。
     b. 从列表中选择需要使用的证书。
9. 选择添加域。

> **Note**
>
> DNS 最多可能需要 24 小时才能传播并颁发证书。如需解决出现的错误的帮助，请参阅 自定义域问题排查。

添加由第三方 DNS 提供商管理的自定义域

如果未使用 Amazon Route 53 来管理您的域，则可以为使用 Amplify 部署的应用程序添加一个由第三方 DNS 提供商管理的自定义域。

如果您正在使用 GoDaddy，the section called “更新由管理的域的 DNS 记录 GoDaddy”请参阅，了解该提供商的特定说明。

添加由第三方 DNS 提供商管理的自定义域

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要向其添加自定义域的应用程序。
3. 在导航窗格中，依次选择托管、自定义域。
4. 在自定义域页面上，选择添加域。
5. 输入您的根域的名称。例如，如果您的域名是 https://example.com，请输入**`example.com`**。
6. Amplify 检测到您并未使用 Route 53 域，因此您可以选择在 Route 53 中创建托管区。
   - 在 Route 53 中创建一个托管区的方法
     a. 选择在 Route 53 中创建托管区。
     b. 选择配置域。

c. 托管区名称服务器会显示在控制台中。转到您的 DNS 提供商的网站，将名称服务器添加到 DNS 设置中。

d. 选择我已将上述名称服务器添加到我的域注册机构。

e. 继续执行步骤 7。

- 通过手动配置继续的方法

a. 选择手动配置

b. 选择配置域。

c. 继续执行步骤 7。

7. 默认情况下，Amplify 会自动为您的域创建两个子域条目。例如，如果您的域名是 example.com，您将看到子域名，https://www.example.com并https://example.com设置了从根域名到 www 子域名的重定向。

   （可选）如果只想添加子域，可以修改默认配置。要更改默认配置，请从导航窗格中选择重写和重定向，并配置您的域。

8. 选择要使用的 SSL/TLS 证书。您可以使用 Amplify 为您提供的默认托管证书，也可以使用已导入的自定义第三方证书。 AWS Certificate Manager

   - 使用默认 Amplify 托管证书。

     - 选择 Amplify 托管证书。

   - 使用自定义第三方证书。

     a. 选择自定义 SSL 证书。

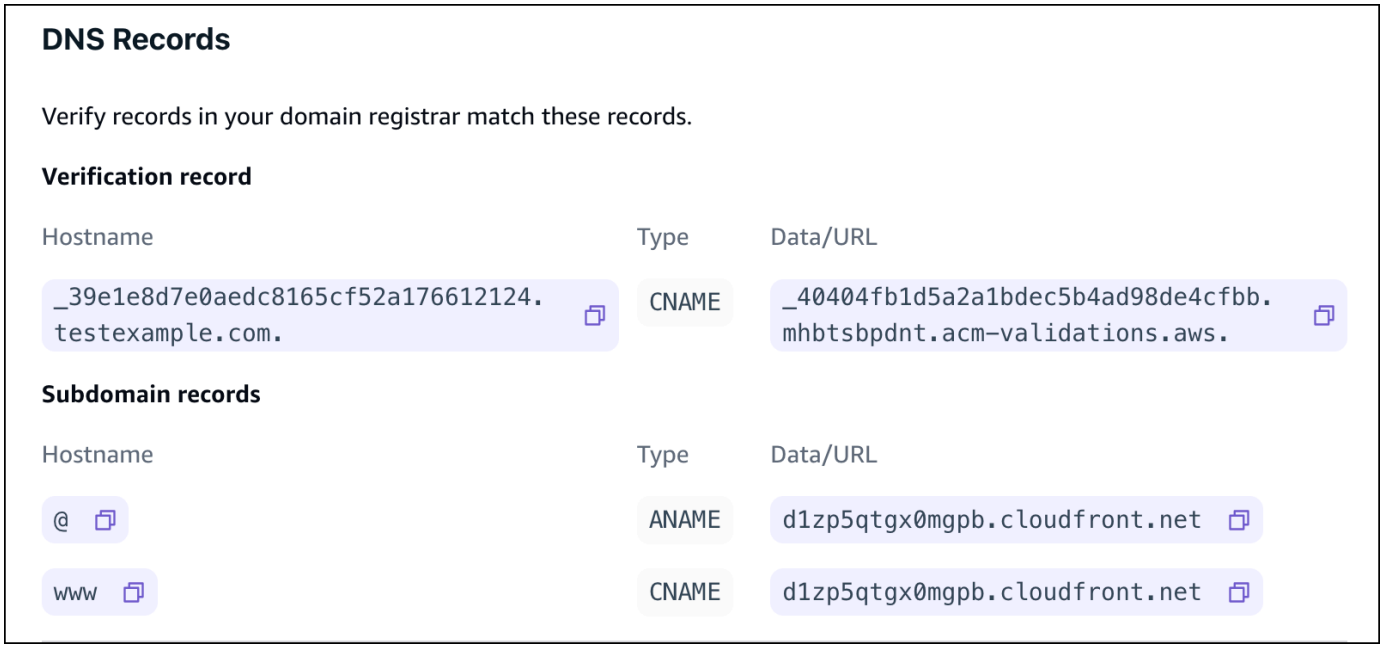
     b. 从列表中选择需要使用的证书。

9. 选择添加域。

10. 如果您在步骤 6 中选择了在 Route 53 中创建托管区，请继续执行步骤 15。

    如果您选择了手动配置，则在步骤 6 中必须通过第三方域提供商更新您的 DNS 记录。

    在操作菜单上，选择查看 DNS 记录。以下屏幕截图展示了在控制台中显示的 DNS 记录。



11. 请执行以下操作之一：

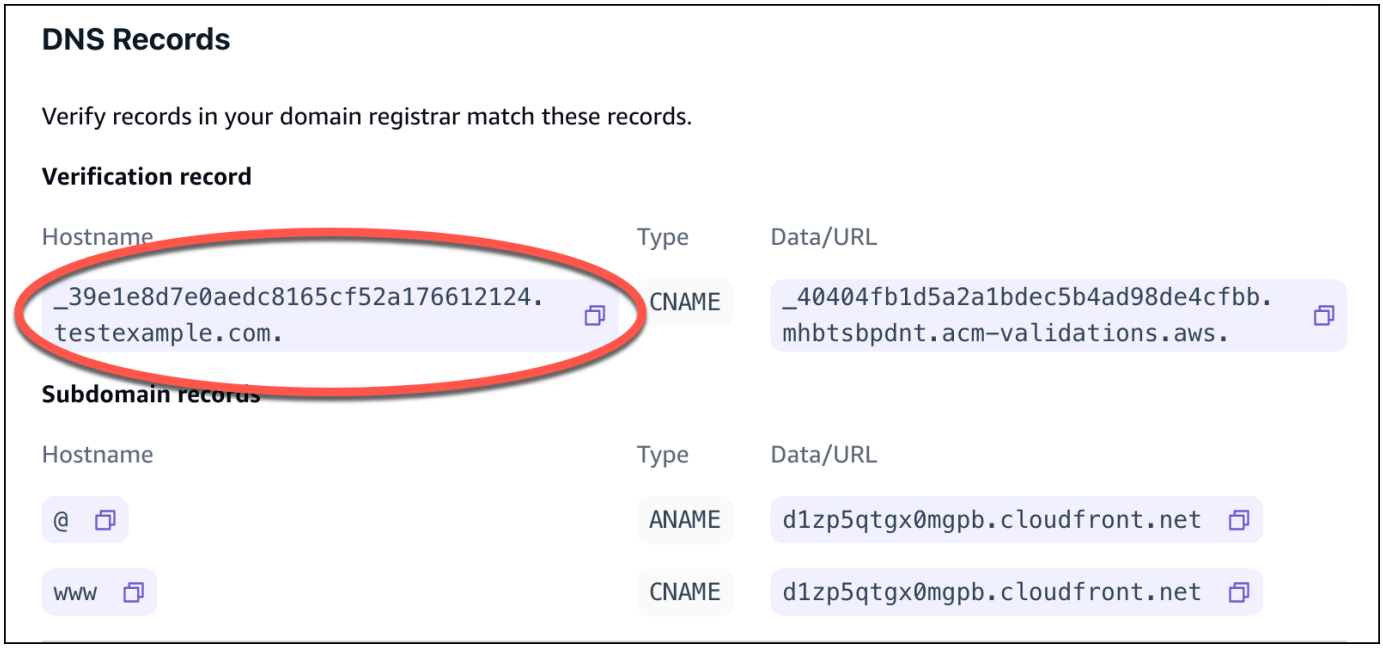
    - 如果您正在使用 GoDaddy，请转至更新由管理的域的 DNS 记录 GoDaddy。
    - 如果您使用的是其他第三方 DNS 提供商，请转到此流程的下一步。

12. 前往您的 DNS 提供商的网站，登录您的账户，然后找到域的 DNS 管理设置。您将配置两条 CNAME 别名记录。

13. 将第一个 CNAME 记录配置为将您的子域指向 AWS 验证服务器。

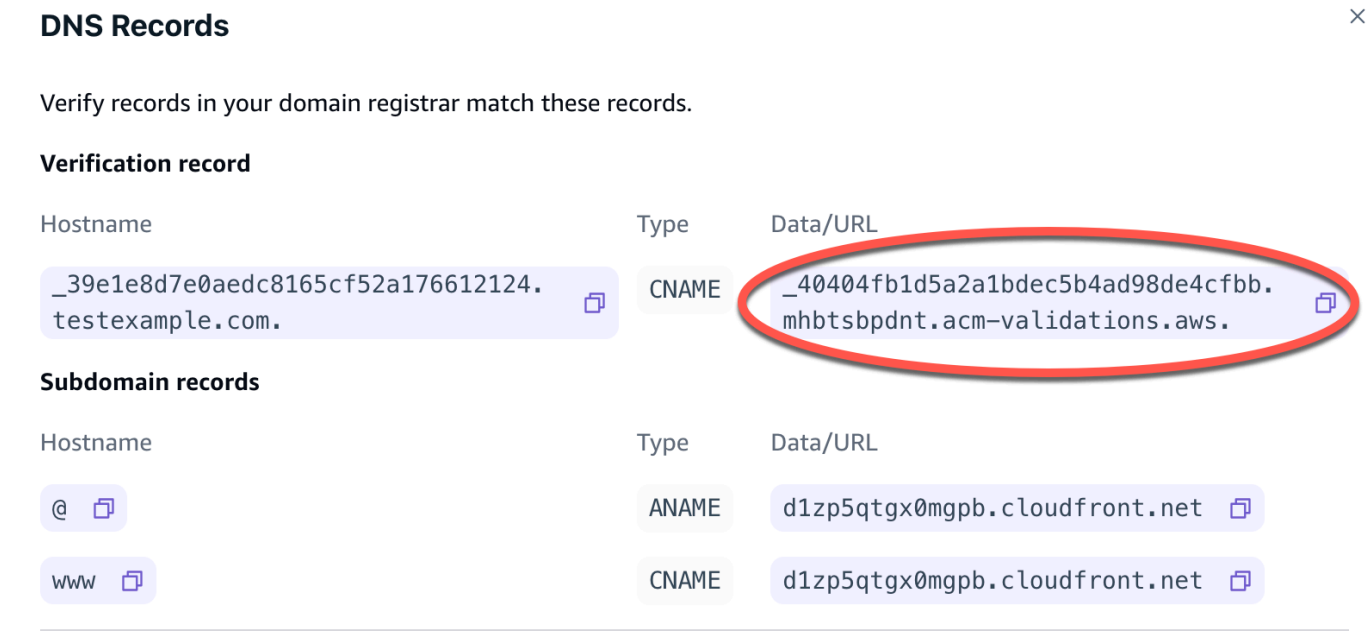
    如果 Amplify 控制台显示用于验证您的子域所有权的 DNS 记录，例如 _c3e2d7eaf1e656b73f46cd6980fdc0e.example.com，则仅为 CNAME 别名记录子域名输入 **`_c3e2d7eaf1e656b73f46cd6980fdc0e`**。

    以下屏幕截图显示了要使用的验证记录所在的位置。



如果 Amplify 控制台显示 ACM 验证服务器记录，例如 _cjhwou20vhu2exampleuw20vuyb2ovb9.j9s73ucn9vy.acm-validations.aws，则为 CNAME 别名记录值输入 **`_cjhwou20vhu2exampleuw20vuyb2ovb9.j9s73ucn9vy.acm-validations.aws`**。

以下屏幕截图显示了要使用的 ACM 验证记录所在的位置。



Amplify 使用这些信息来验证您的域名所有权并为您的域名生成 SSL/TLS 证书。在 Amplify 控制台验证您的域的所有权后，将使用 HTTPS/2 传送所有流量。

> **Note**
>
> 由 AWS Certificate Manager (ACM) 生成的默认 Amplify 证书有效期为 13 个月，只要您的应用程序由 Amplify 托管，该证书就会自动续订。如果别名记录验证记录已被修改或删除，Amplify 将无法续订证书。您必须在 Amplify 控制台中删除并重新添加该域。

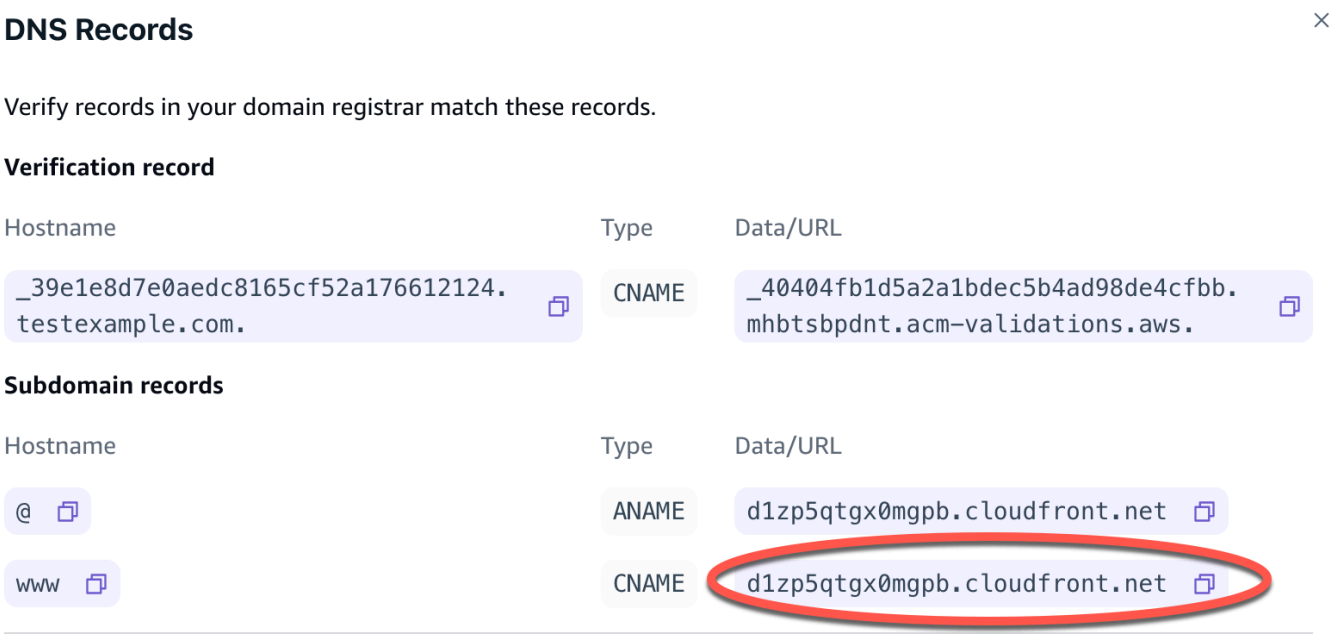
> **Important**
>
> 在 Amplify 控制台中添加自定义域后，请尽快执行此步骤，这一点很重要。 AWS Certificate Manager (ACM) 立即开始尝试验证所有权。随着时间的推移，检查的频率会降低。如果您在创建应用程序几小时后添加或更新别名记录，则可能会导致您的应用程序陷入待验证状态。

14. 配置第二条 CNAME 别名记录，将您的子域指向 Amplify 域。例如，如果您的子域为 www.example.com，请为子域名输入 www。

    如果 Amplify 控制台将您的应用程序的域显示为 d111111abcdef8.cloudfront.net，请为 Amplify 域输入 **`d111111abcdef8.cloudfront.net`**。

    如果您有生产流量，我们建议您在 Amplify 控制台中的域状态显示为可用后更新此别名记录。

    以下屏幕截图显示了要使用的域名记录所在的位置。



15. 将 ANAME/ALIAS 记录配置为指向应用程序的根域（例如 https://example.com）。ANAME 记录将域根指向一个主机名。如果您有生产流量，我们建议您在 Amplify 控制台中的域状态显示为可用后更新别名记录。对于不 ANAME/ALIAS 支持的 DNS 提供商，我们强烈建议您将 DNS 迁移到 Route 53。有关更多信息，请参阅将 Amazon Route 53 配置为 DNS 服务。

> **Note**
>
> 域所有权的验证和第三方域的 DNS 传播可能需要长达 48 小时。如需帮助解决出现的错误，请参阅自定义域问题排查。

更新由管理的域的 DNS 记录 GoDaddy

如果 GoDaddy 是您的 DNS 提供商，请按照以下说明在 GoDaddy 用户界面中更新您的 DNS 记录，完成将 Amplify 应用程序连接到您的 GoDaddy域名。

添加由管理的自定义域名 GoDaddy

1. 在使用更新 DNS 记录之前 GoDaddy，请先完成该过程的第一步到第九步the section called “添加由第三方 DNS 提供商管理的自定义域”。
2. 登录您的 GoDaddy 账户。
3. 在您的域列表中，找到要添加的域，然后选择管理 DNS。
4. 在 DNS 页面上，在 D NS 记录部分 GoDaddy 显示您的域名的记录列表。您需要添加两条新的别名记录。
5. 创建第一条别名记录，将您的子域指向 Amplify 域。

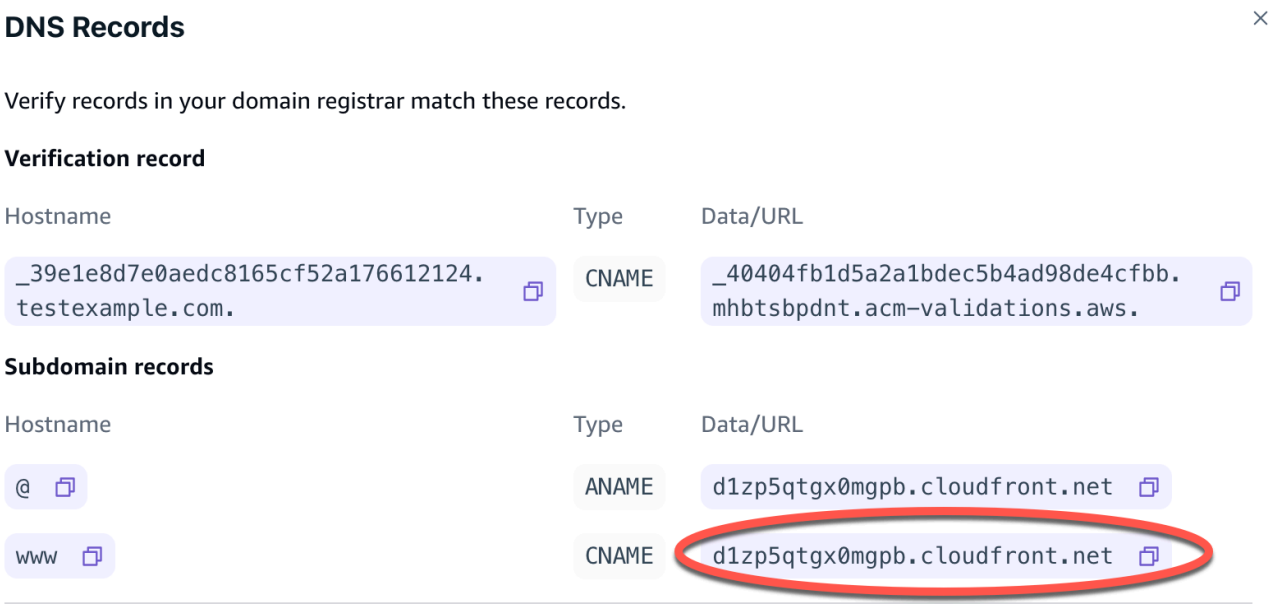
   a. 在 DNS 记录部分，选择添加新记录。

   b. 对于类型，选择 CNAME。

   c. 对于名称，仅输入子域。例如，如果您的子域为 www.example.com，请在名称中输入 www。

   d. 对于值，请在 Amplify 控制台中查看您的 DNS 记录，然后输入值。如果 Amplify 控制台将您的应用程序的域显示为 d111111abcdef8.cloudfront.net，请为值输入 **`d111111abcdef8.cloudfront.net`**。

      以下屏幕截图显示了要使用的域名记录所在的位置。



e. 选择保存。

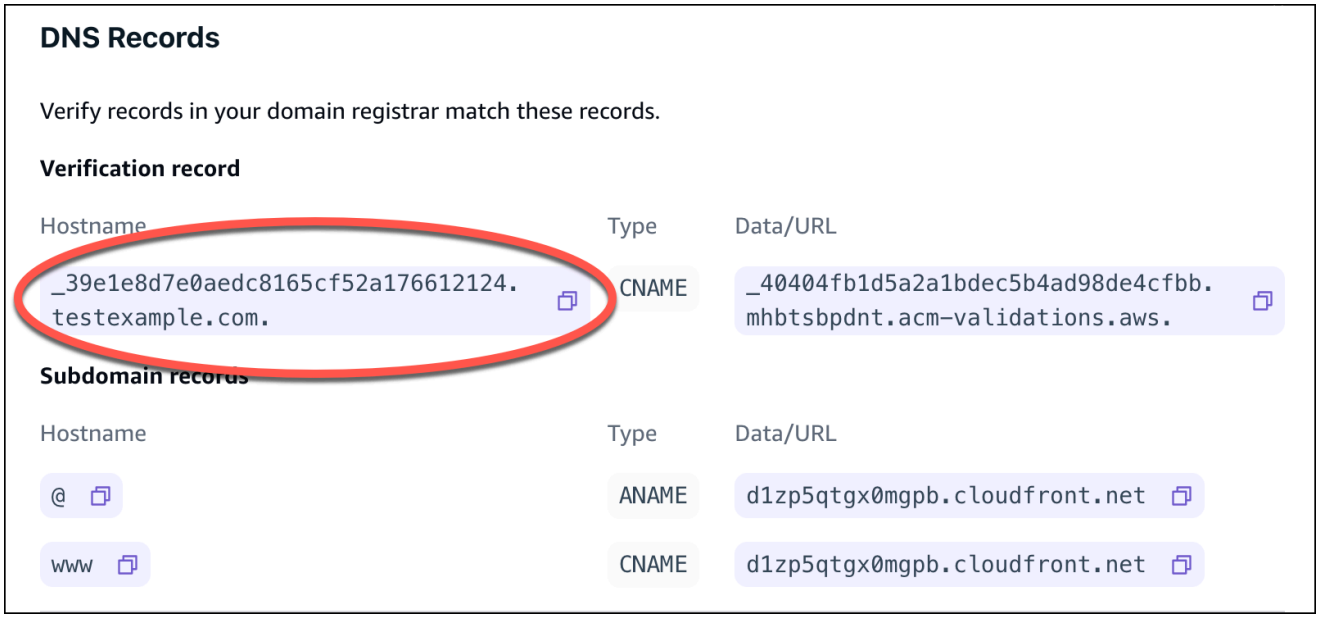
6. 创建第二个 CNAME 记录以指向 AWS Certificate Manager (ACM) 验证服务器。单个经过验证的 ACM 会为您的域生成 SSL/TLS 证书。

   a. 对于类型，选择 CNAME。

   b. 对于名称，请输入子域。

      例如，如果 Amplify 控制台中用于验证子域所有权的 DNS 记录为 _c3e2d7eaf1e656b73f46cd6980fdc0e.example.com，则在名称中仅输入 **_c3e2d7eaf1e656b73f46cd6980fdc0e**。

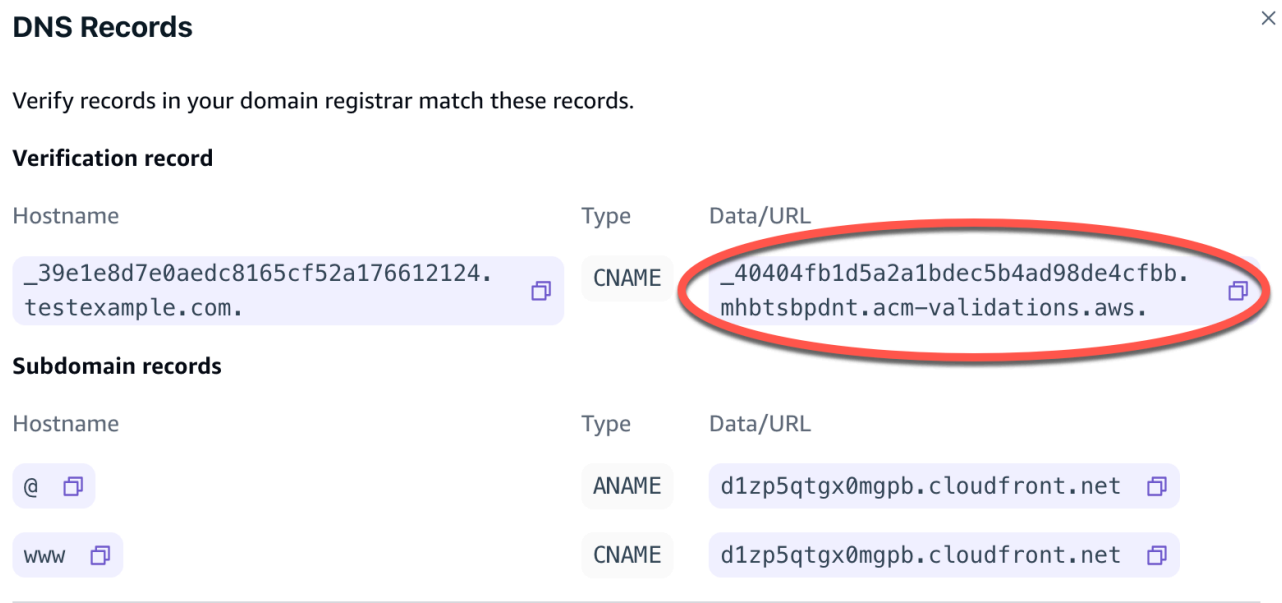
      以下屏幕截图显示了要使用的验证记录所在的位置。



c. 对于值，请输入 ACM 验证证书。

例如，如果验证服务器为 _cjhwou20vhu2exampleuw20vuyb2ovb9.j9s73ucn9vy.acm-validations.aws，请在值中输入 _cjhwou20vhu2exampleuw20vuyb2ovb9.j9s73ucn9vy.acm-validations.aws。

以下屏幕截图显示了要使用的 ACM 验证记录所在的位置。



d. 选择保存。

> **Note**
>
> 由 AWS Certificate Manager (ACM) 生成的默认 Amplify 证书有效期为 13 个月，只要您的应用程序由 Amplify 托管，该证书就会自动续订。如果别名记录验证记录已被修改或删除，Amplify 将无法续订证书。您必须在 Amplify 控制台中删除并重新添加该域。

7. 子域名不需要执行此步骤。 GoDaddy 不支持ANAME/ALIAS records. For DNS providers that do not have ANAME/ALIAS支持，我们强烈建议将您的 DNS 迁移到 Amazon Route 53。有关更多信息，请参阅将 Amazon Route 53 配置为 DNS 服务。

   如果您想继续 GoDaddy 作为提供商并更新根域，请添加转发并设置域名转发：

   a. 在 DNS 页面上，找到页面顶部的菜单并选择转发。
   b. 在域部分中，选择添加转发。
   c. 选择 http://，然后为目标 URL 输入要转发至的子域名称（例如，www.example.com）。
   d. 对于转发类型，请选择临时 (302)。
   e. 选择保存。

更新域的 SSL/TLS 证书

您可以随时更改域名正在使用的 SSL/TLS 证书。例如，您可从使用托管证书更改为使用自定义证书。需要管理证书及其到期通知时，此方法将非常实用。您还可以更改为域使用的自定义证书。对 SSL 证书的更改不会导致活动域出现任何停机时间。有关证书的更多信息，请参阅使用 SSL/TLS 证书。

使用以下过程更新为域使用的证书类型或自定义证书。

更新域证书的方法

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要更新的应用程序。
3. 在导航窗格中，依次选择托管、自定义域。
4. 在自定义域页面上，选择域配置。
5. 在您的域的详细信息页面上，找到自定义 SSL 证书部分。更新证书的步骤因您要执行的更改类型而异。
   - 从自定义证书更改为默认 Amplify 托管证书的方法

- 选择 Amplify 托管证书。

- 从托管证书更改为自定义证书的方法

  a. 选择自定义 SSL 证书。

  b. 从列表中选择需要使用的证书。

- 将一种自定义证书更改为其他自定义证书

  - 对于自定义 SSL 证书，请从列表中选择要使用的新证书。

6. 选择保存。域的状态详细信息将指明，Amplify 已发起托管证书的 SSL 创建过程或自定义证书的配置过程。

管理子域

子域是网址中出现在域名之前的部分。例如，www 为 www.amazon.com 的子域，aws 为 aws.amazon.com 的子域。如果已经拥有一个生产网站，您可能只想连接一个子域。子域也可以是多级的，例如 beta.alpha.example.com 有多级子域 beta.alpha。

仅添加子域

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要向其中添加子域的应用程序。
3. 在导航窗格中依次选择域、自定义域。
4. 在自定义域页面上，选择添加域。
5. 输入您的根域名称，然后选择配置域。例如，如果您的域名是 https://example.com，请输入 example.com。
6. 选择排除根目录并修改子域名称。例如，如果域为 example.com，则可以将其修改为仅添加子域 alpha。
7. 选择添加域。

添加多级子域

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要向其中添加多级子域的应用程序。
3. 在导航窗格中依次选择域、自定义域。

4. 在自定义域页面上，选择添加域。
5. 输入带有子域的域的名称，选择排除根目录，然后修改子域以添加新的级别。

   例如，如果您有一个名为 alpha.example.com 的域，并且想要创建一个多级子域 beta.alpha.example.com，则需要输入 beta 作为子域值。
6. 选择添加域。

添加或编辑子域

向应用程序添加自定义域后，您可以编辑现有子域或添加新的子域。

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择您要为其管理子域的应用程序。
3. 在导航窗格中依次选择域、自定义域。
4. 在自定义域页面上，选择域配置。
5. 在子域部分中，您可以根据需要编辑现有的子域。
6. （可选）要添加新的子域，请选择新增。
7. 选择保存。

设置通配符子域

Amplify Hosting 现在支持通配符子域。通配符子域是一个通用的子域，您可以借此将现存和不存在的子域指向应用程序的特定分支。当使用通配符将应用程序中的所有子域关联到特定分支时，您可以向任何子域中的应用程序用户提供相同的内容，而无需单独配置每个子域。

要创建通配符子域，请指定星号 (*) 作为子域名。例如，如果您为应用程序的特定分支指定通配符子域 `*.example.com`，则任何以 example.com 结尾的网址都将路由到该分支。在这种情况下，对 `dev.example.com` 和 `prod.example.com` 的请求将路由到 `*.example.com` 子域。

请注意，Amplify 仅支持自定义域的通配符子域。您不能在默认 `amplifyapp.com` 域中使用此功能。

以下要求适用于通配符子域：

- 子域只能用星号 (*) 指定。
- 您不能使用通配符替换子域名的一部分，例如：*domain.example.com。
- 您不能替换某个域名中间的子域名，例如：subdomain.*.example.com。

- 默认情况下，Amplify 配置的所有证书均涵盖自定义域的所有子域。

添加或删除通配符子域

向应用程序添加自定义域后，您可以为应用程序分支添加通配符子域。

1. 登录 AWS 管理控制台 并打开 Amplify 托管控制台。
2. 选择要为其管理通配符子域的应用程序。
3. 在导航窗格中依次选择域、自定义域。
4. 在自定义域页面上，选择域配置。
5. 在子域部分中，您可以添加或删除通配符子域。
   - 添加新的通配符子域
     a. 选择新增。
     b. 对于子域，请输入 *。
     c. 对于您的应用程序分支，请从列表中选择一个分支名称。
     d. 选择保存。
   - 删除通配符子域
     a. 选择子域名旁边的删除。通往未明确配置的子域的流量停止，Amplify Hosting 会为这些请求返回 404 状态码。
     b. 选择保存。

为 Amazon Route 53 自定义域设置自动子域

将应用程序连接到 Route 53 中的自定义域后，Amplify 允许您自动为新连接的分支创建子域。例如，如果您连接开发分支，Amplify 可以自动创建 dev.exampledomain.com。当您删除分支时，所有关联的子域都将自动删除。

为新连接的分支设置自动创建子域

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择连接到 Route 53 中管理的自定义域的应用程序。
3. 在导航窗格中依次选择域、自定义域。
4. 在自定义域页面上，选择域配置。

5. 在自动创建子域部分中，启用此功能。

> **Note**
>
> 此功能仅适用于根域，例如 exampledomain.com。如果您的域已经是子域，例如 dev.exampledomain.com，则 Amplify 控制台不会显示此复选框。

带子域的网页预览

按照上述说明启用子域自动检测后，您的应用程序的拉取请求 Web 预览也可通过自动创建的子域进行访问。拉取请求关闭后，关联的分支和子域将自动删除。有关为拉取请求设置网页预览的更多信息，请参阅 拉取请求的 Web 预览。

自定义域问题排查

如果您在 AWS Amplify 控制台中向应用程序添加自定义域时遇到问题，请查阅 Amplify 问题处理章节中的自定义域问题排查。如果您在其中找不到问题的解决方案，请联系 支持。有关更多信息，请参阅 AWS 支持 用户指南中的创建支持案例。

Amplify 托管式网站的防火墙支持

Amplify 托管网站的防火墙支持使您能够通过与直接集成来保护您的 Web 应用程序。 AWS WAF AWS WAF 允许您配置一组名为 Web 访问控制列表 (Web ACL) 的规则，这些规则根据您定义的可自定义 Web 安全规则和条件允许、阻止或监控（计数）Web 请求。将 Amplify 应用与集成后 AWS WAF，您可以更好地控制和了解应用接受的 HTTP 流量。要了解更多信息 AWS WAF，请参阅《AWS WAF 开发者指南》中的AWS WAF 工作原理。

Amplify Hosting 的所有 AWS 区域 运营场所都支持防火墙。这种整合属于 AWS WAF 全球资源，类似于 CloudFront。网页 ACLs 可以连接到多个 Amplify Hosting 应用程序，但它们必须位于同一个区域。

您可以使用 AWS WAF 保护您的 Amplify 应用免受常见网络漏洞的侵害，例如 SQL 注入和跨站脚本。这些威胁可能会影响应用程序的可用性和性能，损害安全性或消耗过多的资源。例如，您可以创建规则来允许或阻止以下请求：来自指定 IP 地址范围的请求；来自 CIDR 数据块的请求；源自特定国家或地区的请求；包含非预期 SQL 代码或脚本攻击的请求。

您还可以创建与 HTTP 标头、方法、查询字符串、URI 和请求正文中的指定字符串或正则表达式模式匹配的规则（限制为前 8 KB）。此外，您可以创建规则来阻止来自特定用户代理、机器人程序和内容抓取程序的事件。例如，您可以使用基于速率的规则来指定每个客户端 IP 在尾随的、不断更新的 5 分钟期间内允许的 Web 请求数。

要详细了解支持的规则类型和其他 AWS WAF 功能，请参阅AWS WAF 开发者指南和 AWS WAF API 参考。

> **Important**
>
> 安全是双方共同承担 AWS 的责任。 AWS WAF 并不是所有互联网安全问题的解决方案，您必须对其进行配置以满足您的安全与合规性目标。为了帮助您了解在使用时如何应用分担责任模型 AWS WAF，请参阅AWS WAF 服务使用过程中的安全性。

主题

- 在中启 AWS WAF 用 Amplify 应用程序 AWS 管理控制台
- 将 Web ACL 与 Amplify 应用程序取消关联
- AWS WAF 使用 Amplify 应用程序启用 AWS CDK
- Amplify 如何与之集成 AWS WAF

- Amplify 应用程序的防火墙定价

在中启 AWS WAF 用 Amplify 应用程序 AWS 管理控制台

您可以在 Amplify 控制台或主机中为 Amplify 应用程序启用 AWS WAF 保护。 AWS WAF

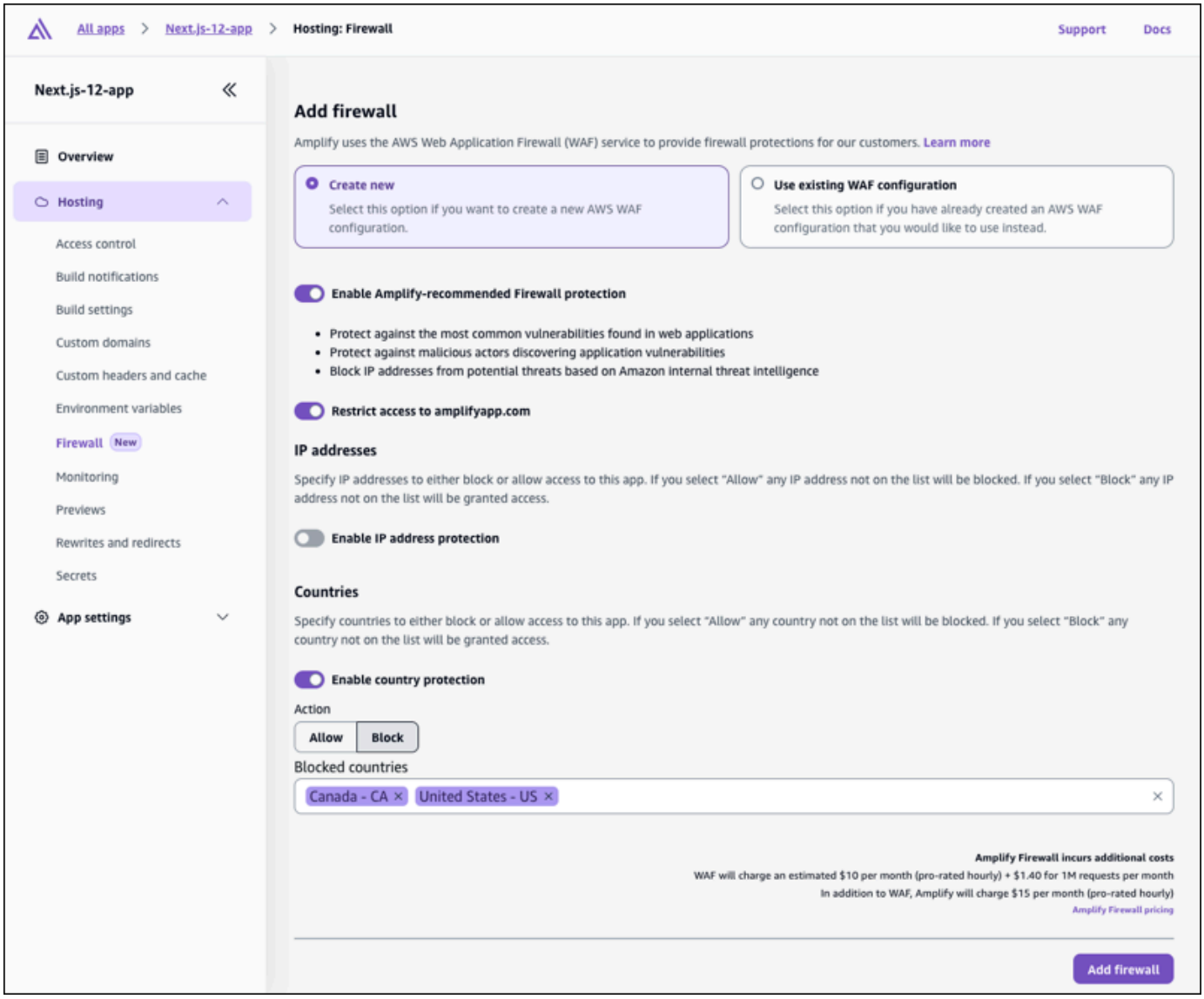
- Amplify 控制台 — 您可以在 Amplify 控制台中将 AWS WAF 网页 ACL 关联到您的应用程序，从而为现有 Amplify 应用程序启用防火墙功能。使用一键保护功能创建 Web ACL，其中包含我们认为是适用于大多数应用程序的最佳实践的预配置规则。您可以选择按 IP 地址和国家/地区自定义访问权限。本节中的操作说明介绍了如何设置一键保护功能。
- AWS WAF 控制台-使用您在控制台中创建的预配置的 Web ACL， AWS WAF 或者使用创建。 AWS WAF APIs您必须创建要与全球 (CloudFront) 区域中的 Amplify 应用程序关联的网站 ACLs 。您 ACLs 可能已经存在区域网络 AWS 账户，但它们与 Amplify 不兼容。有关入门说明，请参阅《AWS WAF 开发人员指南》中的设置 AWS WAF 及其组件。

使用以下步骤在 Amplify 控制台中 AWS WAF 为现有应用程序启用。

AWS WAF 为现有 Amplify 应用程序启用

1. 登录 AWS 管理控制台 并打开 Amplify 控制台，网址为。https://console.aws.amazon.com/amplify/
2. 在所有应用程序页面上，选择要启用防火墙功能的已部署应用程序的名称。
3. 在导航窗格中依次选择 Hosting、防火墙。

   以下屏幕截图展示了如何在 Amplify 控制台中导航到添加防火墙页面。



4. 在 “添加防火墙” 页面上，您的操作将取决于您是要创建新 AWS WAF 配置还是使用现有配置。

   - 创建新 AWS WAF 配置。

     a. 选择 新建。

     b. 您也可启用以下任意配置之一：

        i. 打开启用 Amplify 推荐的防火墙保护。

        ii. 开启限制对 amplifyapp.com 的访问，以防止通过默认 Amplify 域访问您的应用程序。

        iii. 对于 IP 地址，请开启启用 IP 地址保护。

A. 要指定有访问权限的 IP 地址并阻止所有其他地址，则对于操作，请选择允许。要指定将被阻止的 IP 地址并允许所有其他地址访问，请选择阻止。

B. 对于 IP 版本，请选择IPV4或IPV6。

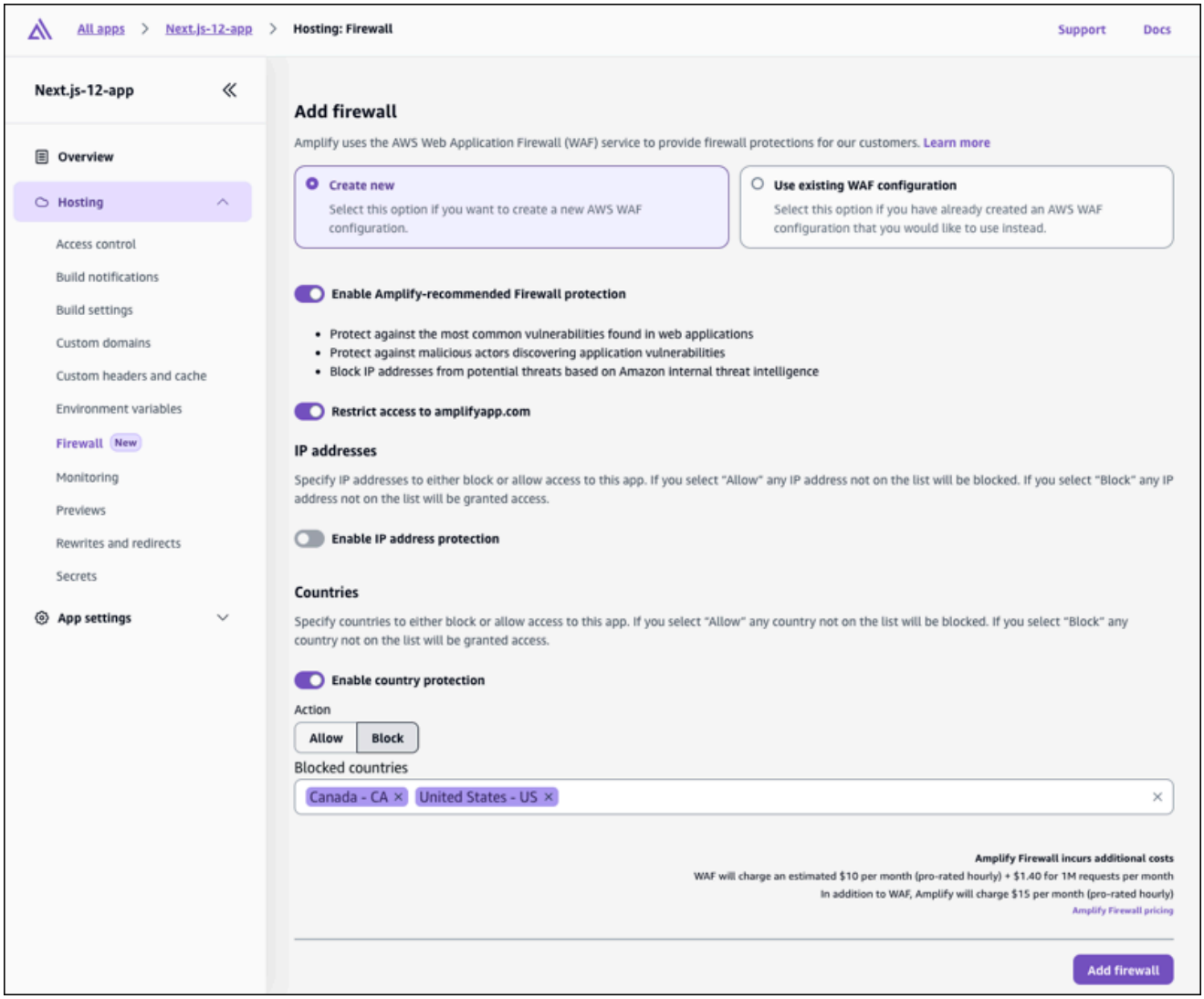
C. 在 IP 地址文本框中，以 CIDR 格式输入要允许或阻止的 IP 地址，每行一个地址。

iv. 对于国家/地区，打开启用国家/地区保护。

A. 要指定有访问权限的国家/地区并阻止所有其他国家/地区，则对于操作请选择允许。要指定将被阻止的国家/地区并允许所有其他国家/地区访问，请选择阻止。

B. 对于国家/地区，请从列表中选择要允许或阻止的国家/地区。

以下屏幕截图演示了如何为应用程序启用新 AWS WAF 配置。



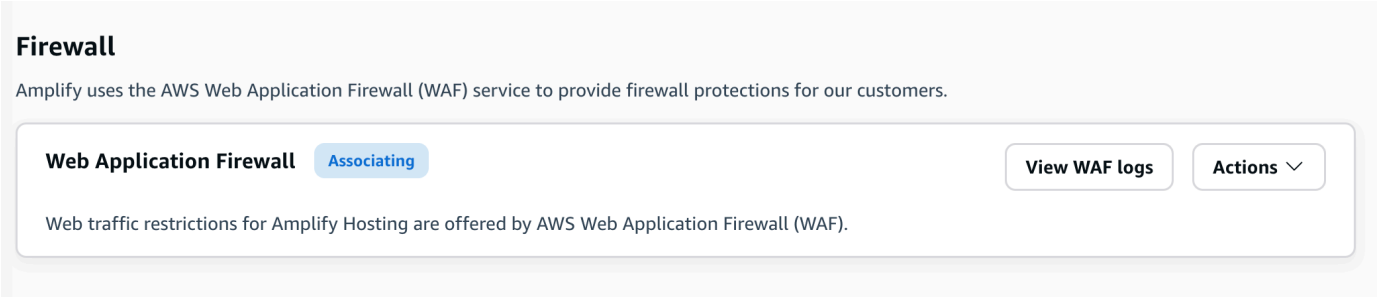
- 使用现有 AWS WAF 配置。

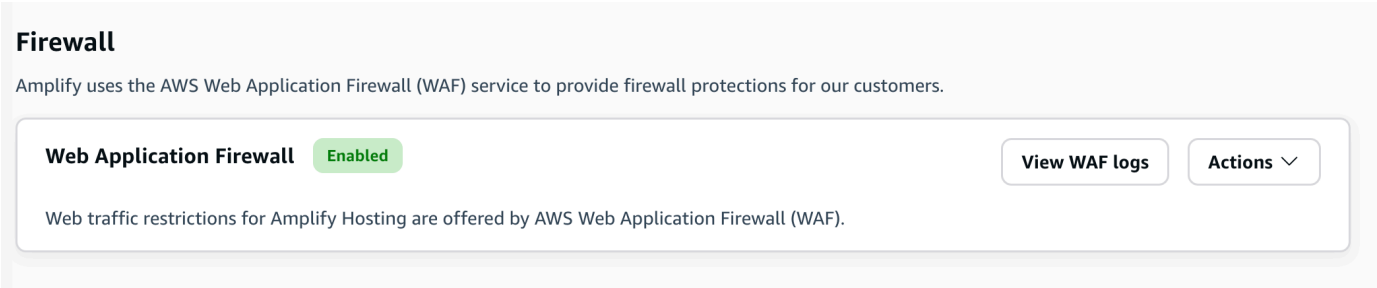
  a. 选择 “使用现有 AWS WAF 配置”。

  b. 从您的 Web ACLs 列表中选择已保存的配置 AWS 账户。 AWS WAF 与 Amplify 应用关联的网页 ACL 必须在全球 (CloudFront) 区域创建。您 ACLs 可能已经存在区域网络 AWS 账户，但它们与 Amplify 不兼容。

5. 选择添加防火墙。
6. 在 “防火墙” 页面上，将显示 “关联” 状态，表示 AWS WAF 设置正在传播中。此过程完成后，状态将变为已启用。

   以下屏幕截图显示了 Amplify 控制台中的防火墙进度状态，指示何时 AWS WAF 配置为 “关联并启用”。





将 Web ACL 与 Amplify 应用程序取消关联

您无法删除已关联到 Amplify 应用程序的 Web ACL。您必须首先在 Amplify 控制台中将该 Web ACL 与应用程序取消关联，然后才能在 AWS WAF 控制台中将其删除。

将 Web ACL 与 Amplify 应用程序取消关联

1. 登录 AWS 管理控制台 并打开 Amplify 控制台，网址为。https://console.aws.amazon.com/amplify/
2. 在所有应用程序页面上，选择要与 Web ACL 取消关联的应用程序的名称。

3. 在导航窗格中依次选择 Hosting、防火墙。
4. 在防火墙页面上，依次选择操作、取消关联防火墙。
5. 在确认模式中，输入**disassociate**，然后选择取消关联防火墙。
6. 在 “防火墙” 页面上，将显示 “取消关联” 状态，表示 AWS WAF 设置正在传播中。

   该过程完成后，您可以在 AWS WAF 控制台中删除 Web ACL。

AWS WAF 使用 Amplify 应用程序启用 AWS CDK

您可以使用 AWS Cloud Development Kit (AWS CDK) 来启用 Ampli AWS WAF fy 应用程序。要了解有关使用 CDK 的更多信息，请参阅《AWS Cloud Development Kit (AWS CDK) 开发人员指南》中的 What is the CDK?。

以下 TypeScript 代码示例演示如何创建包含两个 CDK 堆栈的 AWS CDK 应用程序：一个用于 Amplify，一个用于。 AWS WAF请注意， AWS WAF 堆栈必须部署到美国东部（弗吉尼亚北部）(us-east-1) 区域。Amplify 应用程序堆栈可以部署到不同的区域。您必须创建要与全球 (CloudFront) 区域中的 Amplify 应用程序关联的网络 ACL。您 ACLs 可能已经存在区域网络 AWS 账户，但它们与 Amplify 不兼容。

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

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

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

export class WAFStack extends cdk.Stack {
```

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

    new wafv2.CfnWebACLAssociation(this, "WebACLAssociation", {
      resourceArn: props.appArn,
      webAclArn: webAcl.attrArn,
    });
  }
}

const app = new cdk.App();

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

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

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

Amplify 如何与之集成 AWS WAF

以下列表详细介绍了如何与防火墙支持集成，以及在创建网页并将其与 AWS WAF Amplify ACLs 应用程序关联时需要考虑的限制。

- 您可以 AWS WAF 为任何类型的 Amplify 应用程序启用。这包括任何支持的框架、服务器端渲染 (SSR) 应用程序和完全静态的站点。 AWS WAF 支持 Amplify 第 1 代和第 2 代应用程序。
- 您必须创建要与全球 (CloudFront) 区域中的 Amplify 应用程序关联的网站 ACLs 。您 ACLs 可能已经存在区域网络 AWS 账户，但它们与 Amplify 不兼容。
- Web ACL 和 Amplify 应用程序必须在同一 AWS 账户中创建。您可以使用 AWS Firewall Manager 在多个之间复制 AWS WAF 规则 AWS 账户，以简化组织规则的集中化和分布在多个组织中的过程 AWS 账户。有关更多信息，请参阅《 AWS WAF 开发人员指南》中的 AWS Firewall Manager。
- 您可以在同一 AWS 账户中的多个 Amplify 应用程序之间共享同一 Web ACL。所有应用程序都必须位于同一区域。
- 将 Web ACL 关联到 Amplify 应用程序时，默认情况下该 Web ACL 会附加到该应用程序中的每个分支。当您创建新分支时，这些分支将会关联该 Web ACL。
- 将 Web ACL 关联到 Amplify 应用程序时，将会自动关联到该应用程序的所有域。不过您可以使用主机标头匹配规则，来配置适用于单独域名的规则。
- 您无法删除已关联到 Amplify 应用程序的 Web ACL。在 AWS WAF 控制台中删除 Web ACL 之前，需要将其与应用程序解除关联。

Amplify Web ACL 资源策略

为允许 Amplify 访问您的 Web ACL，系统会在关联过程中间将一个资源策略附加到该 Web ACL。Amplify 会自动构造此资源策略，但你可以使用 API 进行查看。 AWS WAFV2 GetPermissionPolicy需要具备以下 IAM 权限才能将 Web ACL 关联到 Amplify 应用程序。

- 放大：AssociateWeb前交叉韧带
- wafv2：前交叉韧带 AssociateWeb
- wafv2：PutPermissionPolicy
- wafv2：GetPermissionPolicy

Amplify 应用程序的防火墙定价

在 Amplify 应用程序 AWS WAF 上实现的成本是根据以下两个组成部分计算的：

- AWS WAF 使用量 — 将根据 AWS WAF 定价模式向您收取 AWS WAF 使用费。 AWS WAF 费用基于您创建的 Web 访问控制列表 (Web ACLs)、每个 Web ACL 添加的规则数量以及收到的 Web 请求数量。有关定价的详细信息，请参阅 AWS WAF 定价。

- Amplify Hosting 集成费用：将 Web ACL 附加到 Amplify 应用程序后，您需要为每个应用程序每月支付 15.00 美元的费用。该费用按比例按小时计收。

功能分支部署和团队工作流程

Amplify Hosting 旨在与功能分支和 GitFlow 工作流程配合使用。每次在存储库中连接新分支时，Amplify 都会使用 Git 分支来创建新部署。连接第一个分支后，可创建额外的功能分支。

向应用程序添加分支的方法

1. 选择要向其中添加分支的应用程序。
2. 选择应用程序设置，再选择分支设置。
3. 在分支设置页面中选择添加分支。
4. 从您的存储库中选择一个分支。
5. 选择添加分支。
6. 重新部署应用程序。

添加分支后，您的应用程序在 Amplify 默认域中将有两个部署可用，例如https://main.appid.amplifyapp.com和。https://dev.appid.amplifyapp.com这可能有所不同 team-to-team，但通常主分支会跟踪发布代码，并且是您的生产分支。开发分支用作集成分支来测试新功能。这样，Beta 版测试人员可以在开发分支部署上测试未发布的功能，而不会影响主分支部署中的任何生产最终用户。

主题

具有全栈 Amplify Gen 2 应用程序的团队工作流程

AWS Amplify Gen 2 引入了一种 TypeScript基于代码优先的开发者体验，用于定义后端。如需了解 Amplify Gen 2 应用程序的全栈工作流程，请参阅《Amplify 文档》中的全栈工作流程。

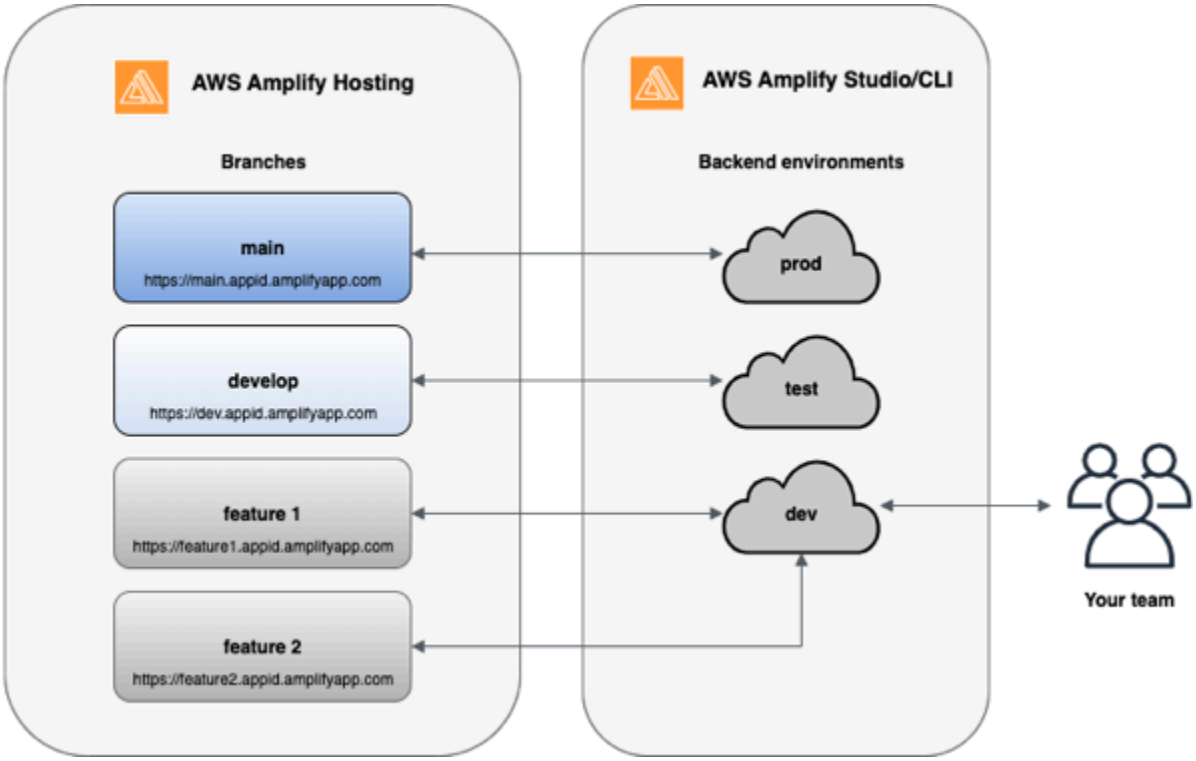
具有全栈 Amplify Gen 1 应用程序的团队工作流程

功能分支部署由前端和可选的后端环境组成。对前端进行构建并部署到全局内容分发网络 (CDN)，而后端由 Amplify Studio 或 Amplify CLI 部署到 AWS。如需了解如何设置此部署场景，请参阅为应用程序构建后端。

Amplify Hosting 在您的功能分支部署中持续部署 GraphQL 和 APIs Lambda 函数等后端资源。您可以使用以下分支模型在 Amplify Hosting 上部署后端和前端。

功能分支工作流程

- 使用 Amplify Studio 创建 prod、test 和 dev 后端环境。
- 将生产后端映射到主分支。
- 将测试后端映射到开发分支。
- 团队成员可以使用设备后端环境来测试各个功能分支。



1. 安装 Amplify CLI 以初始化一个新的 Amplify 项目。

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

2. 为项目初始化 prod 后端环境。如果您没有项目，请使用引导程序工具（如 create-react-app或 Gatsby）创建一个项目。

```
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. 添加 test 和 dev 后端环境。

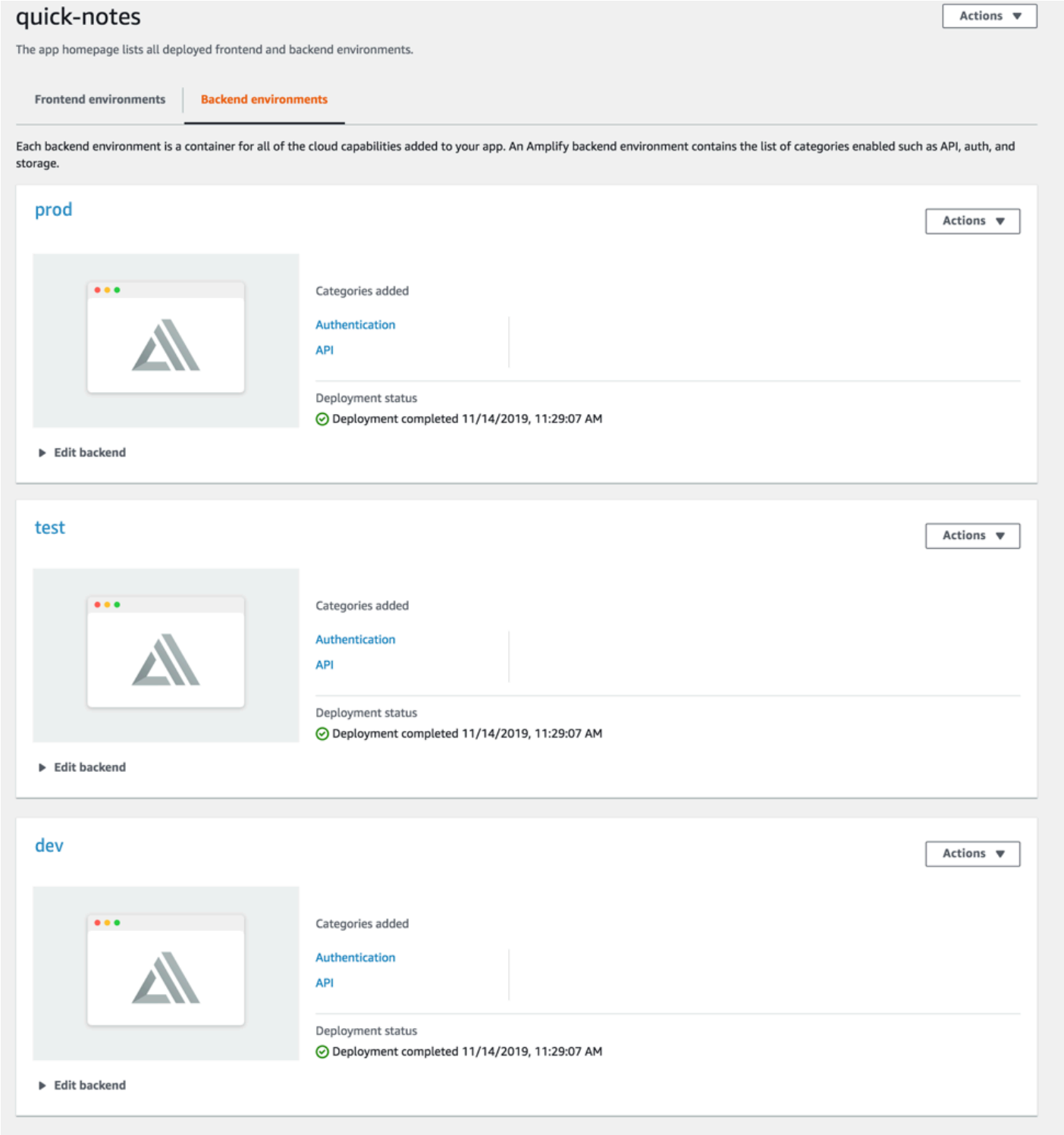
```
amplify env add
 ? Do you want to use an existing environment? (Y/n): n
 ? Enter a name for the environment: test
...
amplify push

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

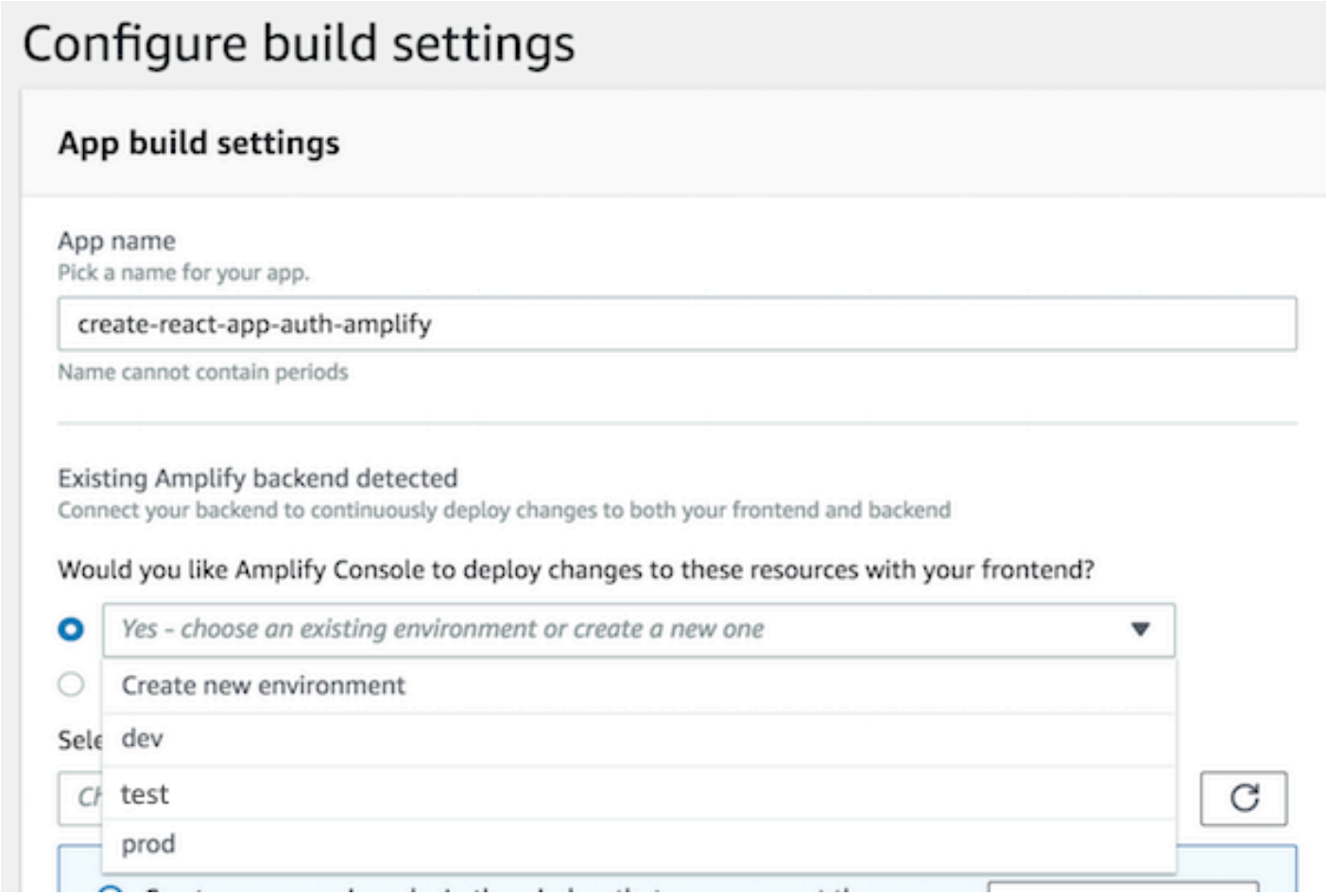
4. 将代码推送到您选择的 Git 存储库（在此示例中，我们假设您已推送到主存储库）。

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

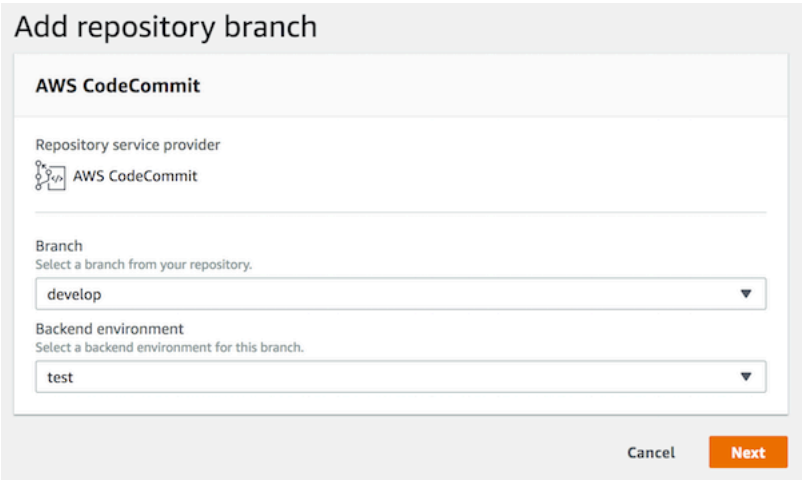
5. 访问中的 Amplify， AWS 管理控制台 查看您当前的后端环境。从导航位置向上导航一级，查看在后端环境选项卡中创建的所有后端环境的列表。



6. 切换到前端环境选项卡，连接存储库提供程序和主分支。
7. 在构建设置屏幕中，选择现有的后端环境，以便在主分支上设置持续部署。从列表中选择生产并将服务角色授予 Amplify。选择保存并部署。构建完成后，您将在上获得主分支部署https://main.appid.amplifyapp.com。



8. 在 Amplify 中连接开发分支（此时假设开发和主分支相同）。选择 test 后端环境。



9. Amplify 现已设置完毕。您可以开始处理功能分支中的新功能。使用本地工作站的 dev 后端环境添加后端功能。

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

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

10处理完功能后，提交代码，创建拉取请求以在内部进行审核。

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

11要预览更改的工作情况，请转到 Amplify 控制台并连接功能分支。注意：如果您的系统上 AWS CLI 安装了（不是 Amplify CLI），则可以直接从终端连接分支。您可以通过转到 App settings (应用程序设置) > General (常规) > AppARN: arn:aws:amplify:<region>:<region>:apps/<appid> 找到您的 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您可以访问您的功能，https://newinternet.appid.amplifyapp.com以便与队友共享。如果一切正常，则将 PR 合并到开发分支。

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

13这将启动一个构建，该版本将更新 Amplify 中的后端和前端，分支部署在。https://dev.appid.amplifyapp.com您可以与内部利益相关者共享此链接，以便他们可以查看新功能。

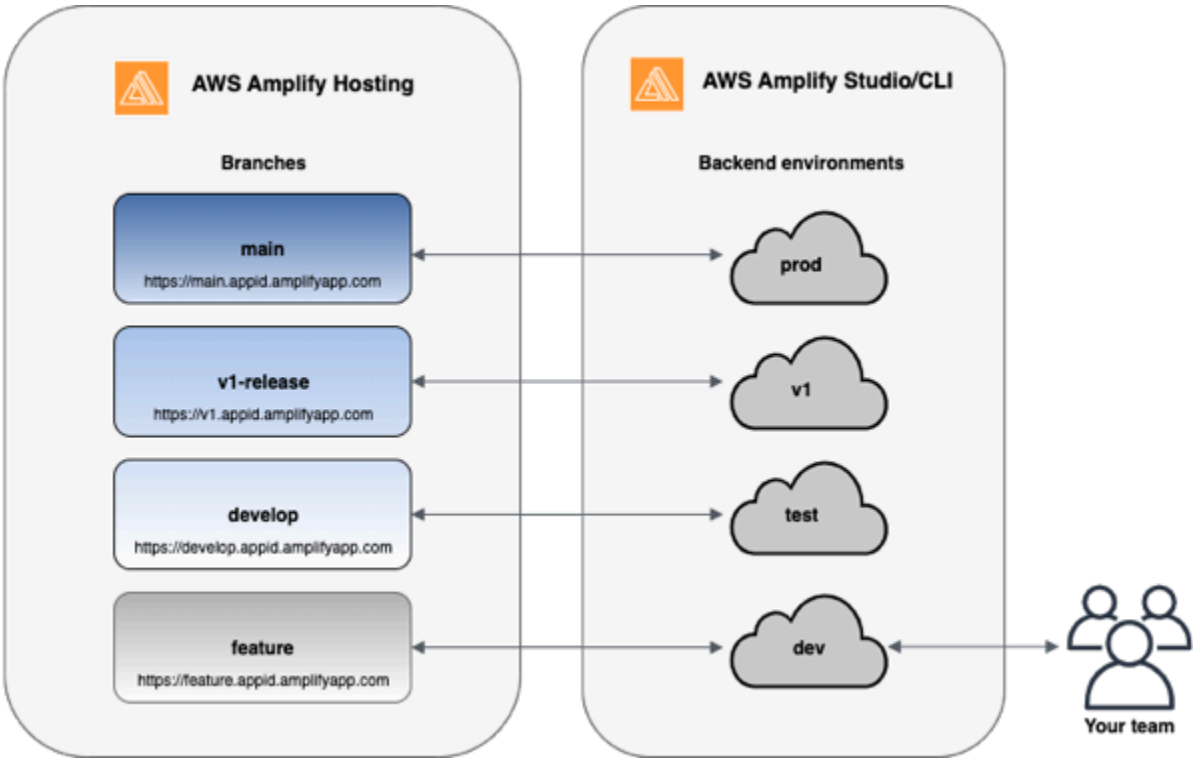
14从 Git 和 Amplify 中删除您的功能分支，并从云中删除后端环境（您始终可以通过运行“amplify env checkout prod”并运行“amplify env add”来启动新的环境）。

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

GitFlow 工作流程

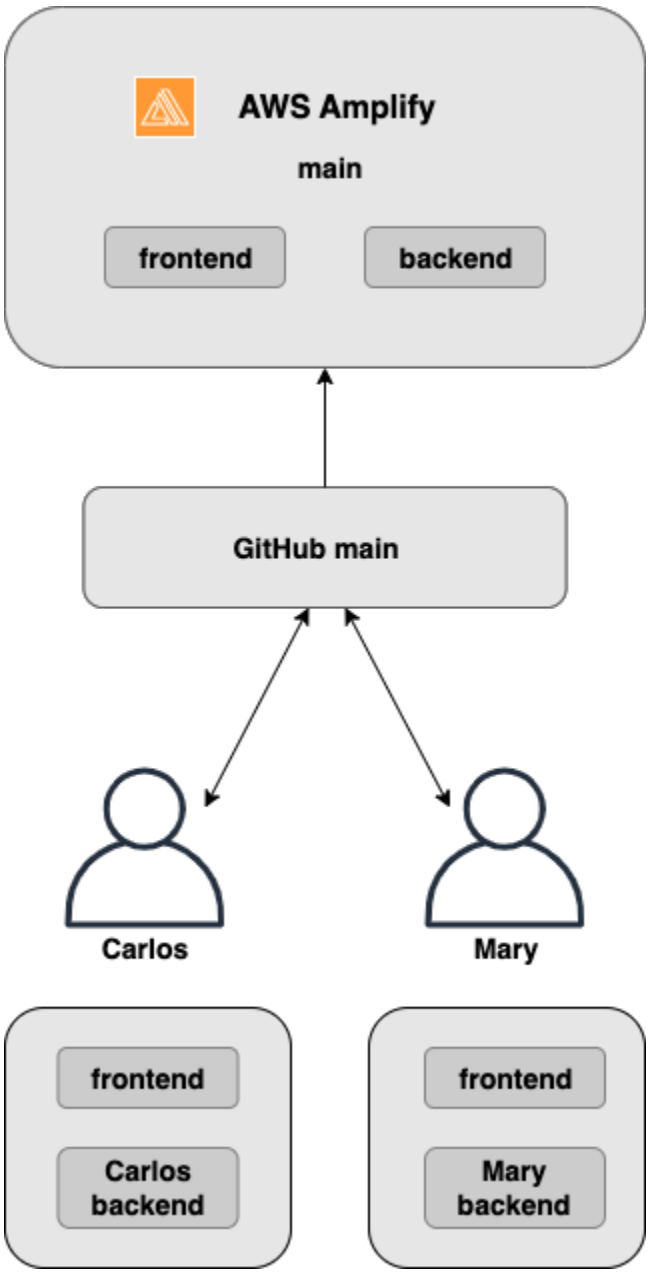
GitFlow 使用两个分支来记录项目的历史。主分支仅跟踪发布代码，开发分支用作新功能的集成分支。GitFlow 通过将新开发与已完成的工作隔离开来简化并行开发。在功能分支中完成新开发（例如功能和非紧急错误修复）。当开发人员确信代码已准备好发布时，功能分支将合并回集成开发分支。仅将来自发布分支和修补程序分支的合并提交到主分支（以修复紧急错误）。

下图显示了推荐的设置 GitFlow。您可以按照上面功能分支工作流程部分中描述的相同过程进行操作。



每个开发人员的沙盒

- 团队中的每个开发人员在云中创建独立于其本地计算机的沙盒环境。这使开发人员能够彼此隔离地进行工作，不会覆盖其他团队成员的更改。
- Amplify 中的每个分支都有自己的后端。这将确保 Amplify 使用 Git 存储库作为从其部署更改的单一信任源，而不是依赖团队中的开发人员手动将其后端或前端从本地计算机推送到生产。



1. 安装 Amplify CLI 以初始化一个新的 Amplify 项目。

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

2. 为项目初始化 mary 后端环境。如果您没有项目，请使用引导程序工具（如 create-react-app或 Gatsby）创建一个项目。

```
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. 将代码推送到您选择的 Git 存储库（在此示例中，我们假设您已推送到主存储库）。

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

4. 将您的 repo > main 连接到 Amplify。
5. Amplify 控制台将检测 Amplify CLI 创建的后端环境。从下拉列表中选择创建新环境并将服务角色授予 Amplify。选择保存并部署。构建完成后，您将获得一个可用的主分支部署，https://main.appid.amplifyapp.com其中包含一个链接到该分支的新后端环境。
6. 在 Amplify 中连接开发分支（此时假设开发和主分支相同），然后选择创建。

基于模式的功能分支部署

使用基于模式的分支部署，您可以自动将与特定模式匹配的分支部署到 Amplify。使用功能分支或发布 GitFlow 工作流程的产品团队现在可以定义模式，例如**`release**`**将以 “发布” 开头的 Git 分支自动部署到可共享的 URL。

1. 选择应用程序设置，再选择分支设置。
2. 在分支设置页面上，选择编辑。
3. 选择分支自动检测，将与模式集匹配的分支自动连接到 Amplify。
4. 在分支自动检测 – 模式框中，输入自动部署分支的模式。
   - **`*`**：部署存储库中的所有分支。
   - **`release*`** – 部署所有以“release”一词开头的分支。
   - **`release*/`**：部署匹配“release/”模式的所有分支。
   - 将多个模式指定为逗号分隔的列表。例如 **`release*, feature*`**。
5. 选择分支自动检测访问控制，为自动创建的所有分支设置自动密码保护。
6. 对于使用 Amplify 后端构建的 Gen 1 应用程序，您可以选择为每个连接的分支创建新环境，或者将所有分支指向一个现有后端。
7. 选择保存。

针对连接到自定义域名的应用程序进行基于模式的功能分支部署

针对连接到 Amazon Route 53 自定义域的应用程序使用基于模式的功能分支部署。

- 有关设置基于模式的功能分支部署的说明，请参阅 为 Amazon Route 53 自定义域设置自动子域
- 有关将 Amplify 应用程序连接到 Route 53 中托管的自定义域名的说明，请参阅 添加由 Amazon Route 53 管理的自定义域
- 有关使用 Route 53 的更多信息，请参阅什么是 Amazon Route 53？。

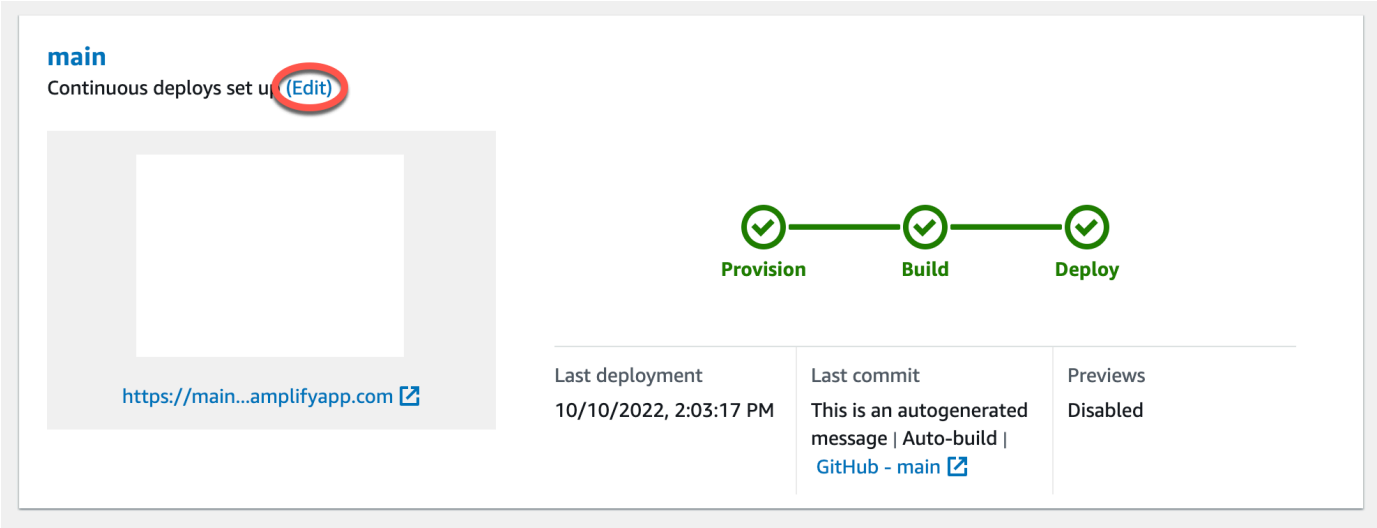
构建时自动生成 Amplify 配置（仅限 Gen 1 应用程序）

> **Note**
>
> 本节中的信息仅适用于 Gen 1 应用程序。如果您想自动部署 Gen 2 应用程序功能分支中的基础架构和应用程序代码更改，请参阅《Amplify 文档》中的全栈分支部署

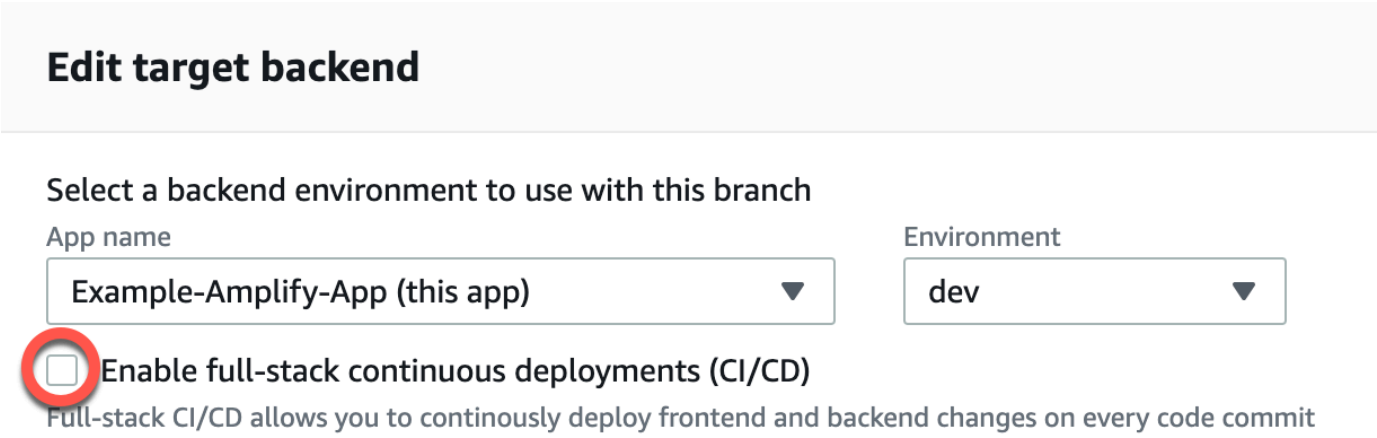
Amplify 支持在构建时为 Gen 1 应用程序自动生成 Amplify 配置 `aws-exports.js` 文件。通过关闭全栈 CI/CD 部署，可以让您的应用程序自动生成 `aws-exports.js` 文件，并确保在构建时不会对后端进行更新。

在构建时自动生成 **`aws-exports.js`**

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要编辑的应用程序。
3. 选择托管环境选项卡。
4. 找到要编辑的分支并选择编辑。



5. 在编辑目标后端页面上，取消选中启用全栈连续部署 (CI/CD) 以关闭此后端的全栈 CI/CD。



6. 选择现有服务角色以授予 Amplify 更改应用程序后端所需的权限。如果您需要创建服务角色，请选择创建角色。有关创建服务角色的更多信息，请参阅添加具有后端资源部署权限的服务角色。
7. 选择保存。Amplify 会在下次构建应用程序时应用这些更改。

有条件的后端构建（仅限 Gen 1 应用程序）

> **Note**
>
> 本节中的信息仅适用于 Gen 1 应用程序。Amplify Gen 2 引入了 TypeScript基于代码的开发者体验。因此，Gen 2 后端不需要此功能。

Amplify 支持在 Gen 1 应用程序的所有分支上进行有条件的后端构建。要配置有条件的后端构建，请将 AMPLIFY_DIFF_BACKEND 环境变量设置为 `true`。启用有条件的后端构建将有助于加快只对前端进行更改的构建速度。

当启用基于差异的后端构建时，在每次构建开始时，Amplify 都会尝试对存储库中的 `amplify` 文件夹运行 diff。如果 Amplify 没有发现任何差异，它将跳过后端构建步骤，并且不会更新您的后端资源。如果您的项目存储库中没有 `amplify` 文件夹，Amplify 会忽略 AMPLIFY_DIFF_BACKEND 环境变量的值。有关设置 AMPLIFY_DIFF_BACKEND 环境变量的说明，请参阅 为 Gen 1 应用程序配置基于 diff 的后端构建。

如果您当前在后端阶段的构建设置中指定了自定义命令，则有条件的后端构建将不起作用。如果要运行这些自定义命令，则必须将其移至应用程序 `amplify.yml` 文件中构建设置的前端阶段。有关更新 `amplify.yml` 文件的更多信息，请参阅 构建规范参考。

跨应用程序使用 Amplify 后端（仅限 Gen 1 应用程序）

> **Note**
>
> 本节中的信息仅适用于 Gen 1 应用程序。如果您想共享 Gen 2 应用程序的后端资源，请参阅《Amplify 文档》中的跨分支共享资源

Amplify 使您能够在给定区域的所有 Gen 1 应用程序中重复使用现有后端环境。您可以在创建新应用程序、将新分支连接到现有应用程序或更新现有前端以指向其他后端环境时执行此操作。

创建新应用程序时重复使用后端

在创建新的 Amplify 应用程序时重复使用后端

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 要创建一个用于此示例的新后端，请执行以下操作：
   a. 在导航窗格中，选择所有应用程序。
   b. 选择新建应用程序和构建应用程序。
   c. 输入您的应用程序名称，例如 **`Example-Amplify-App`**。
   d. 选择确认部署。
3. 要将前端连接到您的新后端，请选择托管环境选项卡。

4. 选择您的 Git 提供商，然后选择连接分支。
5. 在添加存储库分支页面上，对于最近更新的存储库，请选择您的存储库名称。对于分支，请从存储库中选择要连接的分支。
6. 在构建设置页面上，执行以下操作：
   a. 对于应用程序名称，请选择要用于添加后端环境的应用程序。您可以选择当前应用程序或当前区域中的其他任何应用程序。
   b. 对于环境，请选择要添加的后端环境的名称。您可以使用现有环境或创建新环境。
   c. 默认情况下，全栈 CI/CD 处于关闭状态。关闭全栈 CI/CD 会导致应用程序仅在拉取模式下运行。在构建时，Amplify 只会自动生成 `aws-exports.js` 文件，而不修改您的后端环境。
   d. 选择现有服务角色以授予 Amplify 更改应用程序后端所需的权限。如果您需要创建服务角色，请选择创建角色。有关创建服务角色的更多信息，请参阅添加具有后端资源部署权限的服务角色。
   e. 选择下一步。
7. 选择保存并部署。

将分支连接到现有应用程序时重复使用后端

在将分支连接到现有 Amplify 应用时重复使用后端

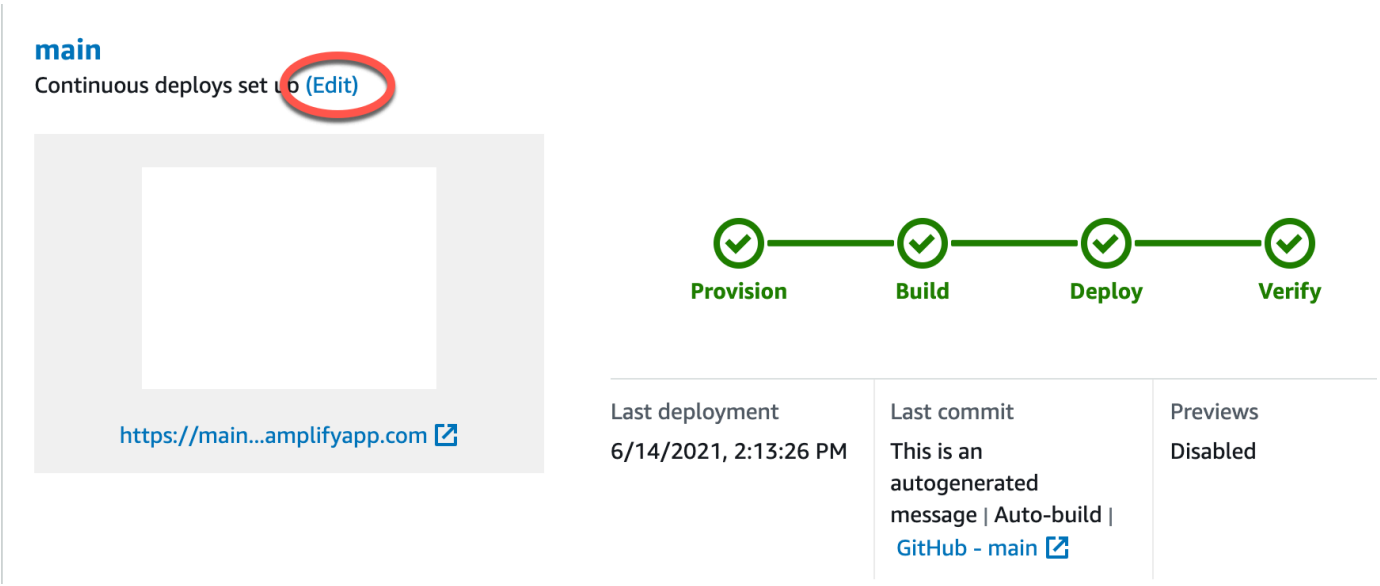
1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要连接新分支的应用程序。
3. 在导航窗格中，依次选择应用程序设置、常规。
4. 在分支部分，选择连接分支。
5. 在添加存储库分支页面上，对于分支，请从存储库中选择要连接的分支。
6. 对于应用程序名称，请选择要用于添加后端环境的应用程序。您可以选择当前应用程序或当前区域中的其他任何应用程序。
7. 对于环境，请选择要添加的后端环境的名称。您可以使用现有环境或创建新环境。
8. 如果需要设置服务角色以授予 Amplify 更改应用程序后端所需的权限，则控制台会提示您执行此任务。有关创建服务角色的更多信息，请参阅添加具有后端资源部署权限的服务角色。
9. 默认情况下，全栈 CI/CD 处于关闭状态。关闭全栈 CI/CD 会导致应用程序在仅拉取模式下运行。在构建时，Amplify 只会自动生成 `aws-exports.js` 文件，而不修改您的后端环境。
10. 选择下一步。

11. 选择保存并部署。

编辑现有前端以指向其他后端

编辑前端 Amplify 应用以指向其他后端

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要为其编辑后端的应用程序。
3. 选择托管环境选项卡。
4. 找到要编辑的分支并选择编辑。



5. 在选择要用于此分支的后端环境页面上，对于应用程序名称，请选择要为其编辑后端环境的前端应用程序。您可以选择当前应用程序或当前区域中的其他任何应用程序。
6. 对于后端环境，请选择要添加的后端环境的名称。
7. 默认情况下，全栈 CI/CD 处于启用状态。取消选中此选项可关闭此后端 CI/CD 的全栈功能。关闭全栈 CI/CD 会导致应用程序在仅拉取模式下运行。在构建时，Amplify 只会自动生成 `aws-exports.js` 文件，而不修改后端环境。
8. 选择保存。Amplify 会在下次构建应用程序时应用这些更改。

为应用程序构建后端

借 AWS Amplify 助，您可以构建包含数据、身份验证、存储和部署到的前端托管的全栈应用程序。AWS

AWS Amplify Gen 2 引入了一种 TypeScript基于代码优先的开发者体验，用于定义后端。如需了解如何使用 Amplify Gen 2 构建后端并将其连接到应用程序，请参阅《Amplify 文档》中的构建和连接后端。

如果您正在寻找有关使用 CLI 和 Amplify Studio 为 Gen 1 应用程序构建后端的文档，请参阅《Gen 1 Amplify 文档》https://docs.amplify.aws/gen1/react/build-a-backend/中的构建和连接后端。

主题

为 Gen 2 应用程序创建后端

有关指导您完成使用 TypeScript基于后端的 Amplify Gen 2 全栈应用程序创建步骤的教程，请参阅 Amplify 文档中的入门。

为 Gen 1 应用程序创建后端

在本教程中，您将使用 Amplify 设置全栈 CI/CD 工作流程。您将在 Amplify Hosting 上部署前端应用程序。然后您将使用 Amplify Studio 创建一个后端。最后，您将云后端连接到前端应用程序。

先决条件

开始本教程之前，请完成以下先决条件。

注册获取 AWS 账户

如果您还不是 AWS 客户，则需要按照在线说明进行创建。 AWS 账户注册后，您就可以访问 Amplify 和其他可与您的应用程序配合使用的 AWS 服务。

创建 Git 存储库

Amplify 支持 GitHub Bitbucket 和 GitLab。 AWS CodeCommit将您的应用程序推送到 Git 存储库。

安装 Amplify 命令行界面（CLI）

有关说明，请参阅Amplify Framework 文档中的安装 Amplify CLI。

第 1 步：部署前端

如果您在 git 存储库中有一个现有的前端应用程序要用于此示例，则可以继续按照部署前端应用程序的说明进行操作。

如果您需要创建用于此示例的新前端应用程序，可以按照《创建 React 应用程序文档》中的创建 React 应用程序说明进行操作。

部署前端应用程序

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 在所有应用程序页面上，选择新建应用程序，然后在右上角选择托管 Web 应用程序。
3. 选择您的 GitHub、Bitbucket 或 AWS CodeCommit 存储库提供商 GitLab，然后选择 “继续”。
4. Amplify 授权访问您的 git 存储库。对于 GitHub 存储库，Amplify 现在使用 GitHub 应用程序功能来授权 Amplify 访问权限。

   有关安装和授权 GitHub 应用程序的更多信息，请参阅设置 Amplify 对仓库的访问权限 GitHub。
5. 在添加存储库分支页面上，执行以下操作：

   a. 在最近更新的存储库列表中，选择要连接的存储库的名称。

   b. 在分支列表中，选择要连接的存储库分支的名称。

   c. 选择下一步。
6. 在配置构建设置页面上，选择下一步。
7. 在查看页面上，选择保存并部署。完成部署时，您可以在 `amplifyapp.com` 默认域上查看应用程序。

> **Note**
>
> 为增强 Amplify 应用程序的安全性，已将 amplifyapp.com 域注册到公共后缀列表 (PSL)。为进一步增强安全性，如果您需要在 Amplify 应用程序的默认域名中设置敏感 Cookie，我们建议您使用带 `__Host-` 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造 (CSRF) 攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

步骤 2：创建后端

现在，您已经在 Amplify Hosting 上部署了前端应用程序，可以创建后端了。使用以下说明创建带有简单数据库和 GraphQL API 端点的后端。

创建后端

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 在所有应用程序页面上，选择您在步骤 1 中创建的应用程序。
3. 在应用程序主页上，选择后端环境选项卡，然后选择开始。这将启动默认暂存 环境的设置过程。
4. 设置完成后，选择 启动 Studio 以访问 Amplify Studio 中的暂存后端环境。

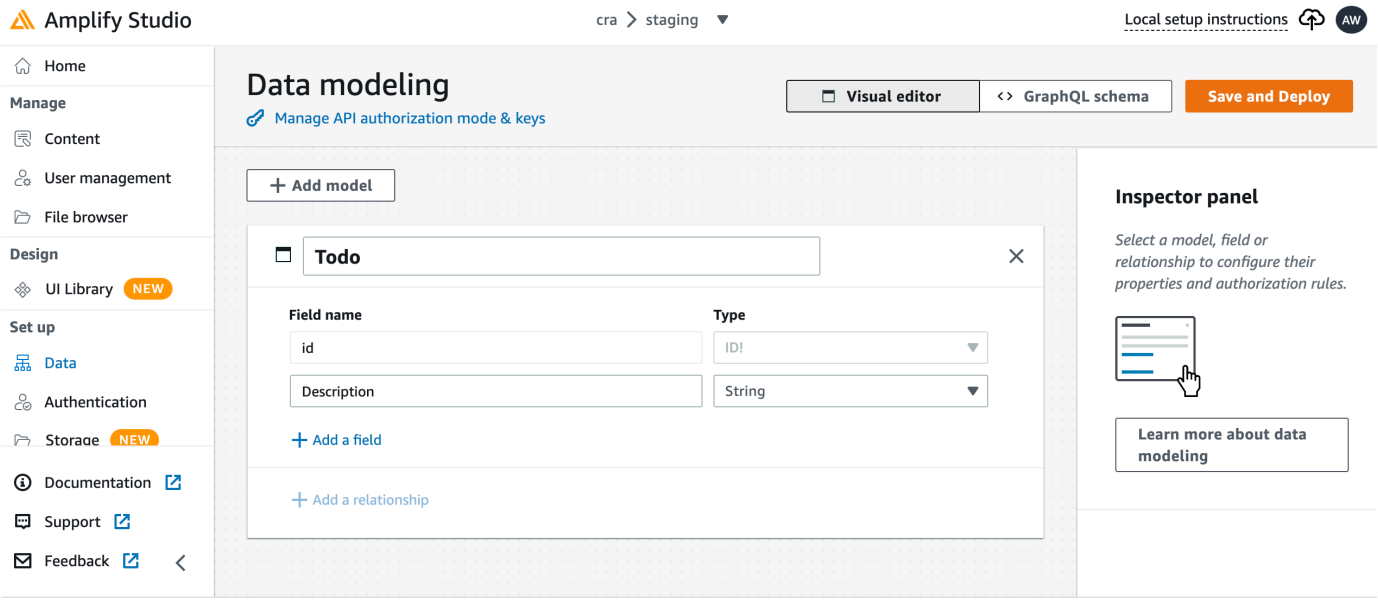
Amplify Studio 是一个可视化界面，用于创建和管理您的后端并加快前端用户界面的开发。有关 Amplify Studio 的更多信息，请参阅 Amplify Studio 文档。

使用以下说明，使用 Amplify Studio 可视化后端生成器界面可以创建简单的数据库。

创建数据模型

1. 在应用程序暂存环境的主页上，选择创建数据模型。这将打开数据模型设计器。
2. 在数据建模页面上，选择添加模型。
3. 对于标题，输入 **`Todo`**。
4. 选择添加字段。
5. 对于字段名称，输入 **`Description`**。

   以下屏幕截图是设计器中数据模型的外观示例。



6. 选择保存并部署。
7. 返回 Amplify Hosting 控制台，暂存环境的部署将在进行中。

在部署过程中，Amplify Studio 会在后端创建所有必需的 AWS 资源，包括用于访问数据的 GraphQL API 和用于托管待办事项的 Amazon DynamoDB AWS AppSync 表。Amplify CloudFormation 用于部署您的后端，这使您可以将后端定义存储为。 infrastructure-as-code

第 3 步：将后端连接至前端

现在，您已经部署了前端并创建了包含数据模型的云后端，您需要将它们连接起来。按照以下说明，使用 Amplify CLI 将您的后端定义下拉到本地应用程序项目中。

将云后端连接到本地前端

1. 打开终端窗口并导航到本地项目的根目录。
2. 在终端窗口中运行以下命令，将红色文本替换为项目的唯一应用程序 ID 和后端环境名称。

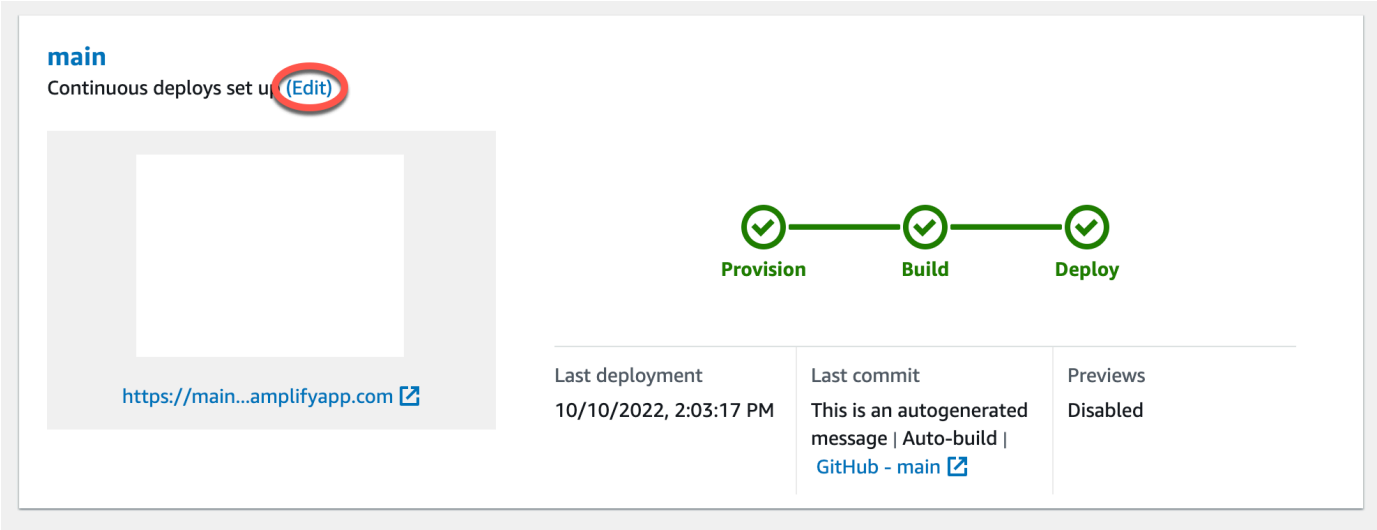
```
amplify pull --appId abcd1234 --envName staging
```

3. 按照终端窗口中的说明完成项目设置。

现在，您可以配置生成过程以将后端添加到持续部署工作流程中。按照以下说明在 Amplify Hosting 控制台中将前端分支与后端连接起来。

连接前端应用程序分支和云后端

1. 在应用程序主页上，选择托管环境选项卡。
2. 找到主分支并选择编辑。



3. 在编辑目标后端窗口中，为环境选择要连接的后端的名称。在此示例中，选择您在步骤 2中创建的暂存后端。

   默认情况下，全栈 CI/CD 处于启用状态。取消选中此选项可关闭此后端 CI/CD 的全栈功能。关闭全栈 CI/CD 会导致应用程序在仅拉取模式下运行。在构建时，Amplify 只会自动生成 `aws-exports.js` 文件，而不修改您的后端环境。
4. 接下来，您必须设置服务角色，以授予 Amplify 更改应用程序后端所需的权限。您可以使用现有的服务角色或创建新的服务角色。有关说明，请参阅添加具有后端资源部署权限的服务角色。
5. 添加服务角色后，返回编辑目标后端窗口，然后选择保存。
6. 要完成将暂存后端与前端应用程序主分支的连接，请对项目执行新的构建。

   请执行以下操作之一：

   - 从您的 git 存储库中，推送一些代码以在 Amplify 控制台中启动构建。
   - 在 Amplify 控制台中，导航到应用程序的构建详情页面，然后选择重新部署此版本。

后续步骤

设置功能分支部署

按照我们推荐的工作流程，在多个后端环境中设置功能分支部署。

在 Amplify Studio 中创建前端用户界面

使用 Studio 使用一组界面组件构建您的前端 ready-to-use用户界面，然后将其连接到您的应用程序后端。有关更多信息和教程，请参阅 Amplify Framework 文档中的 Amplify Studio 用户指南。

高级部署功能

本章介绍了可增强 Amplify Hosting 工作流的高级部署功能。这些功能提供了额外的控制和功能，有助团队更有效地管理部署，确保代码质量，并在整个开发生命周期中保持安全性。

了解如何使用密码身份验证来保护功能分支，限制对尚未发布的功能的访问。为拉取请求启用 Web 预览，以便在将代码合并到生产分支 URLs 之前，在唯一预览中查看更改。在将代码推送到生产环境之前，使用 Cypress 框架设置 end-to-end测试以捕捉回归。尽管已不再提供“部署到 Amplify”按钮功能，但您仍然可以使用 Amplify Hosting 直接从存储库轻松部署应用程序。

主题

限制访问 Amplify 应用程序分支的权限

如果您正在开发未发布的功能，则可以对功能分支进行密码保护，从而仅限特定用户访问。在分支上设置访问控制后，当用户尝试访问分支的网址时，系统会提示他们输入用户名和密码。

您可以设置适用于单个分支的密码，也可以设置适用于所有已连接分支的全局密码。如果在分支和全局级别都启用了访问控制，则分支级别密码优先于全局（应用程序）级别密码。

Amplify 会对尝试访问受密码保护的资源的失败请求进行节流。这种行为可保护应用程序免受字典攻击，或者其他读取受访问控制保护的数据的尝试。

使用以下步骤设置密码，以限制对 Amplify 应用程序分支的访问。

在功能分支上设置密码

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要为其设置功能分支密码的应用程序。
3. 在导航窗格中依次选择托管和访问控制。
4. 在访问控制设置部分，选择管理访问。
5. 在管理访问控制页面上，执行以下任一操作。

- 设置适用于所有已连接分支的用户名和密码的方法
  - 启用管理所有分支的访问权限。例如，如果您连接了主要、开发和功能分支，则可以为所有分支应用相同的用户名和密码。
- 设置适用于个别分支的用户名和密码的方法
  a. 禁用管理所有分支的访问权限。
  b. 找到要管理的分支。在访问设置中选择需要受限密码。
  c. 对于用户名，输入用户名。
  d. 对于密码，输入密码。
- 选择保存。

6. 如果您正在管理服务器端渲染 (SSR) 应用程序的访问控制权限，请通过从 Git 存储库执行新构建来重新部署该应用程序。要让 Amplify 应用您的访问控制权限设置，就必须执行此步骤。

拉取请求的 Web 预览

Web 预览为开发和质量保证 (QA) 团队提供了一种在将代码合并到生产或集成分支之前预览拉取请求 (PRs) 中更改的方法。拉取请求可以让您将已推送到存储库中某个分支的更改告诉给其他人。打开拉取请求后，您可以与合作者讨论和查看潜在的更改，并在更改合并到基础分支之前添加后续提交。

Web 预览会将向存储库发出的每个拉取请求部署到一个唯一的预览网址，该网址与您的主网站使用的网址完全不同。对于使用 Amplify CLI 或 Amplify Studio 配置了后端环境的应用程序，每个拉取请求（仅限私有 Git 存储库）都会创建一个临时后端，该后端在 PR 关闭时会被删除。

当您的应用开启 Web 预览时，每个 PR 都会计入每个应用程序 50 个分支的 Amplify 配额。为避免超过此配额，请务必关闭您的 PRs。有关限额的更多信息，请参阅Amplify 托管服务限额。

> **Note**
>
> 当前，当 AWS CodeCommit 用作存储库提供者时，`AWS_PULL_REQUEST_ID`环境变量不可用。

Web 预览安全性

为确保安全，您可以在所有使用私有存储库的应用程序上启用 Web 预览，但不能在所有使用公有存储库的应用程序上启用 Web 预览。如果您的 Git 存储库是公开的，则只能为不需要 IAM 服务角色的应

用程序设置预览。例如，带有后端的应用程序和部署到 WEB_COMPUTE 托管平台的应用程序需要一个 IAM 服务角色。因此，如果这些类型的应用程序存储库是公开的，则无法为其启用 Web 预览。Amplify 强制执行此限制是为了防止第三方提交使用您的应用程序的 IAM 角色权限运行的任意代码。

使用 SSR 计算角色为公有存储库中的应用程序启用 Web 预览后，您需要仔细管理有权访问该角色的分支。我们建议不要使用应用程序级别的角色，而应在分支级别附加计算角色。这可让您仅向需要访问特定资源的分支授予访问权限。有关更多信息，请参阅 添加 SSR 计算角色以允许访问资源 AWS。

为拉取请求启用 Web 预览

对于存储在存储 GitHub 库中的应用程序，网络预览使用 Amplify GitHub 应用程序访问存储库。如果您要在之前使用存储库部署的现有 Amplify 应用程序上启用网页预览，则必须先将该应用程序迁移为 OAuth 使用 Amplify 应用程序。 GitHub GitHub 有关迁移说明，请参阅 将现有OAuth应用程序迁移到 Amplif GitHub y 应用程序。

为拉取请求启用 Web 预览

1. 选择托管，然后选择预览。

   > **Note**
   >
   > 只有将应用程序设置为持续部署并连接到 git 存储库时，预览才会显示在应用程序设置菜单中。有关此类部署的说明，请参阅现有代码入门。

2. 仅限 GitHub 存储库，请执行以下操作在您的账户中安装和授权 Amplify GitHub 应用程序：

   a. 在 “安装 GitHub 应用程序以启用预览” 窗口中，选择 “安装 GitHub 应用程序”。

   b. 选择要在其中配置 Amplify 应用程序 GitHub的 GitHub 账户。

   c. GitHub.com 打开一个页面，用于为您的账户配置存储库权限。

   d. 请执行以下操作之一：

      - 要将安装应用于所有存储库，请选择所有存储库。
      - 要将安装限于您所选择的特定存储库，请选择仅选择存储库。确保在您选择的存储库中包含您要为其启用 Web 预览的应用程序的存储库。

   e. 选择保存

3. 为存储库启用预览后，返回 Amplify 控制台为特定分支启用预览。在预览页面上，从列表中选择一个分支，然后选择编辑设置。

4. 在管理预览设置页面中，启用拉取请求预览。然后，选择 Confirm（确认）。

5. 对于全栈应用程序，执行以下操作之一：
   - 选择为每个拉取请求创建新的后端环境。此选项使您能够在不影响生产的情况下测试更改。
   - 选择将此分支的所有拉取请求指向现有环境。
6. 选择确认。

下次您为分支提交拉取请求时，Amplify 会构建您的 PR 并将其部署到预览网址。拉取请求关闭后，将删除预览网址，并删除与拉取请求关联的所有临时后端环境。仅限 GitHub 仓库，您可以直接通过 GitHub 账户中的拉取请求访问网址的预览。

使用子域名进行 Web 预览访问

对于连接到 Amazon Route 53 管理的自定义域名的 Amplify 应用程序，可以使用子域名访问拉取请求的 Web 预览。拉取请求关闭后，与拉取请求关联的分支和子域名会被自动删除。在为应用程序设置基于模式的功能分支部署后，这是 Web 预览的默认行为。有关设置自动子域的说明，请参阅 为 Amazon Route 53 自定义域设置自动子域。

为你的 Amplify 应用程序设置 end-to-end赛普拉斯测试

你可以在 Amplify 应用的测试阶段运行 end-to-end (E2E) 测试，以便在将代码推送到生产环境之前捕捉回归情况。可以在构建规范 YAML 中配置测试阶段。目前，您只能在构建期间运行 Cypress 测试框架。

Cypress 是一个 JavaScript基于测试框架，允许您在浏览器上运行 E2E 测试。有关演示如何设置 E2E 测试的教程，请参阅博客文章使用 Amplify 为全栈部署 CI/CD 运行 end-to-end Cypress 测试。

为现有 Amplify 应用程序添加 Cypress 测试

您可以在 Amplify 控制台中更新应用程序的构建设置，从而向现有应用程序添加 Cypress 测试。构建规范 YAML 包含一系列构建命令和相关设置，Amplify 会使用这些命令和设置来运行构建。使用 `test` 步骤在构建时运行任何测试命令。对于 E2E 测试，Amplify Hosting 提供了与 Cypress 的更深层次集成，允许您为测试生成用户界面报告。

下面的列表介绍了测试设置及其使用方式。

预测试

安装运行 Cypress 测试所需的依赖项。Amplify Hosting 使用 mochawesome 生成报告来查看您的测试结果，然后等待在构建期间设置本地服务器。

测试

运行 cypress 命令，使用 mochawesome 执行测试。

测试后

mochawesome 报告是根据输出 JSON 生成的。请注意，如果您使用的是 Yarn，则必须在静默模式下运行此命令才能生成 mochawesome 报告。对于 Yarn，您可使用以下命令。

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

构件>baseDirectory

运行测试的目录。

文物> configFilePath

生成的测试报告数据。

artifacts>files

可供下载的生成构件（屏幕截图和视频）。

以下示例摘自构建规范 `amplify.yml` 文件，其中展示了如何向您的应用程序添加 Cypress 测试。

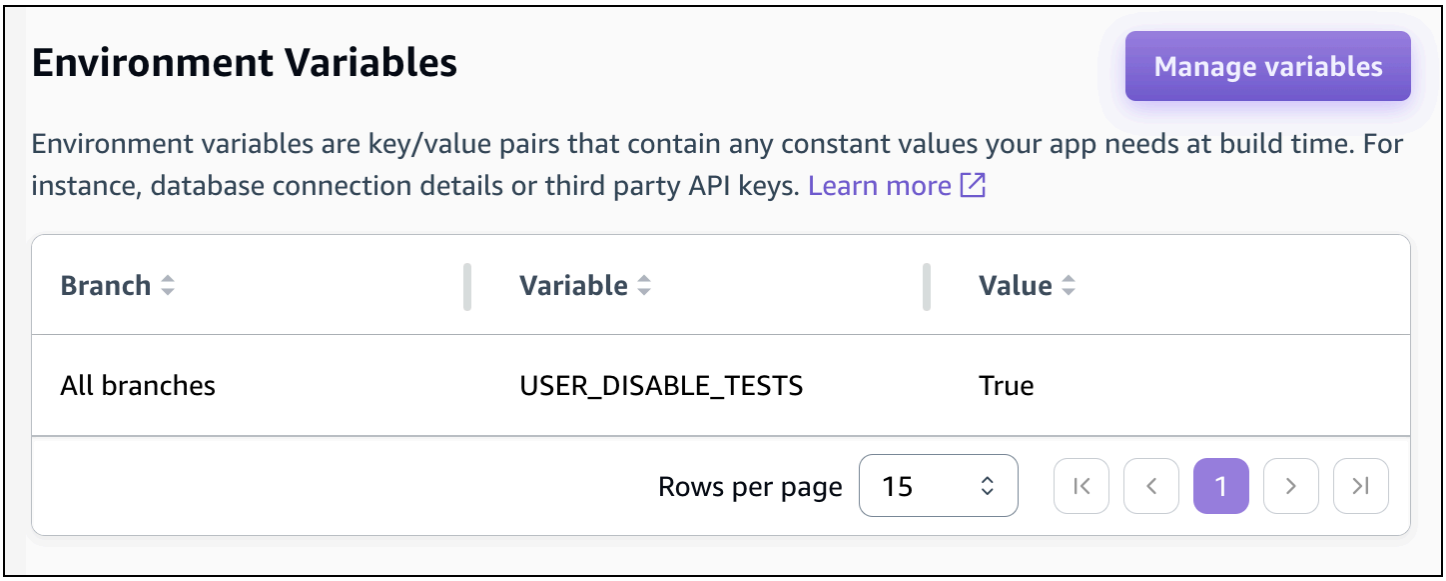
```
test:
  phases:
    preTest:
      commands:
        - npm ci
        - npm install -g pm2
        - npm install -g wait-on
        - npm install mocha mochawesome mochawesome-merge mochawesome-report-generator
        - pm2 start npm -- start
        - wait-on http://localhost:3000
    test:
      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'
```

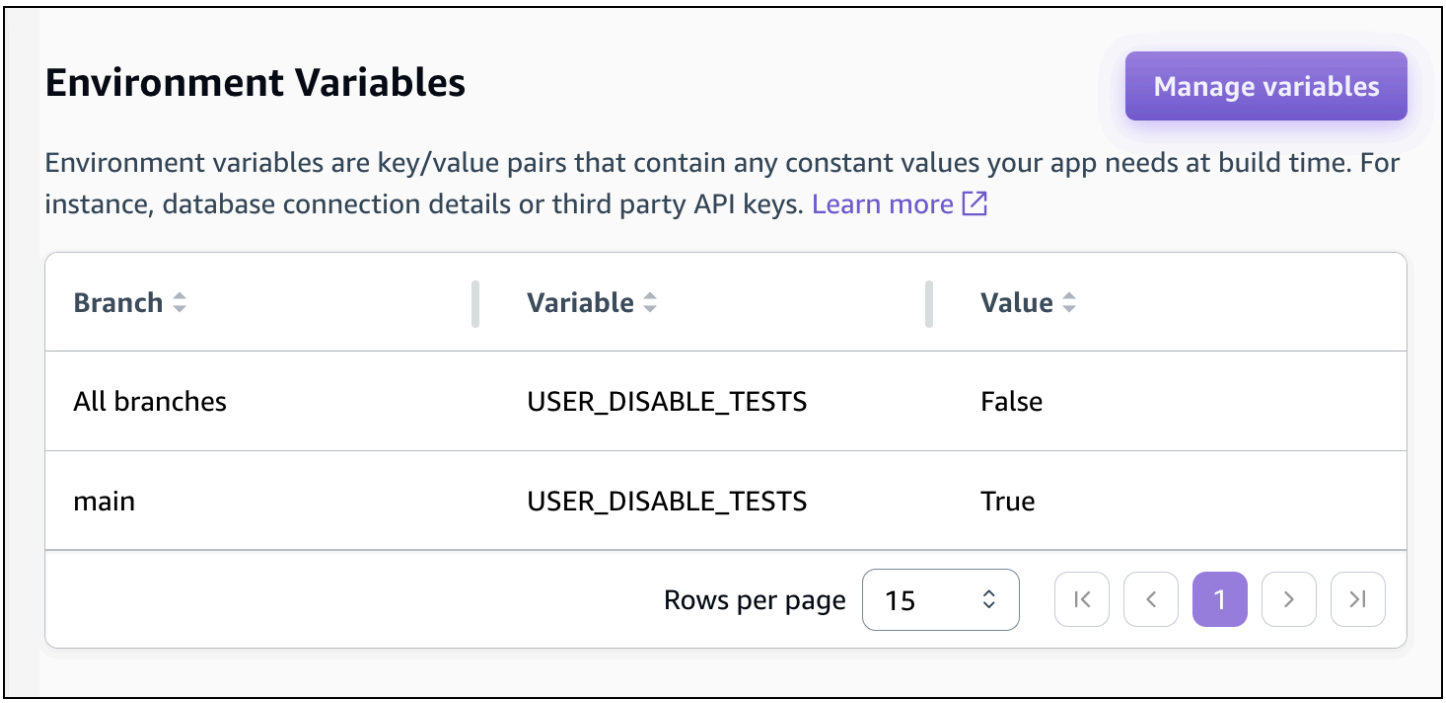
为 Amplify 应用程序或分支关闭测试

将测试配置添加到您的 `amplify.yml` 构建设置后，在每个分支上为每个构建运行 `test` 步骤。如果您想全局禁止测试的运行，或者只对特定分支运行测试，则可以使用 USER_DISABLE_TESTS 环境变量而不修改构建设置。

要全局禁用所有分支的测试，请为所有分支添加值为 `true` 的环境变量 USER_DISABLE_TESTS。以下屏幕截图显示了 Amplify 控制台中的环境变量部分，其中所有分支都禁用了测试。



要禁用特定分支的测试，请为所有分支添加值为 `false` 的环境变量 USER_DISABLE_TESTS，然后为要禁用的每个分支添加一个值为 `true` 分支覆盖。在以下屏幕截图中，主分支上禁用了测试，而其他所有分支启用了测试。



使用此变量禁用测试将导致在构建期间完全跳过测试步骤。要重新启用测试，请将此值设置为 `false` 或删除环境变量。

使用 “部署到 Amplify” 按钮共享项目 GitHub

> **Important**
>
> 不再提供使用部署到 Amplify Hosting 按钮进行一键部署的功能。要从存储库进行部署，请在 Amplify Hosting 中创建一个新应用程序。有关说明，请参阅开始将应用程序部署到 Amplify Hosting。

使用 “部署到 Amplify Hosting” 按钮，您可以公开或在团队内部共享 GitHub 项目。以下是按钮的图像：

将“部署到 Amplify Hosting”按钮添加到您的存储库或博客

将该按钮添加到您的 GitHub README.md 文件、博客文章或任何其他呈现 HTML 的标记页面。该按钮拥有以下两个组件：

1. 位于网址 https://oneclick.amplifyapp.com/button.svg 处的 SVG 图片
2. 带有仓库链接的 Amplify 控制台网址。 GitHub 您可以复制存储库的网址（例如 https://github.com/username/repository），也可以提供指向特定文件夹的深层链接（例如https://github.com/username/repository/tree/branchname/folder）。Amplify Hosting 将在您的存储库中部署默认分支。连接应用程序后，可以连接其他分支。

使用以下示例将按钮添加到 markdown 文件中，例如您的 GitHub README.md。将 https://github.com/username/repository 替换为存储库的网址。

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

使用以下示例将按钮添加到任何 HTML 文档。将 https://github.com/username/repository 替换为存储库的网址。

```
<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>
```

为 Amplify 应用程序设置重定向和重写

重定向使 Web 服务器能够将导航从一个网址重新路由到另一个网址。使用重定向的常见原因包括：自定义网址的外观、避开失效链接、移动应用程序或站点的托管位置而不更改其地址以及将所请求的网址更改为 Web 应用程序需要的形式。

了解 Amplify 支持的重定向

Amplify 在控制台中支持以下重定向类型。

永久重定向 (301)

301 重定向旨在持续更改 Web 地址的目标。原始地址的搜索引擎排名历史记录应用于新目标地址。重定向发生在客户端上，因此浏览器导航栏在重定向后显示目标地址。

使用 301 重定向的常见原因包括：

- 为在页面地址更改时避开失效链接。
- 为在用户在地址中出现可预测拼写错误时避开失效链接。

临时重定向 (302)

302 重定向旨在临时更改 Web 地址的目标。原始地址的搜索引擎排名历史记录不应用于新目标地址。重定向发生在客户端上，因此浏览器导航栏在重定向后显示目标地址。

使用 302 重定向的常见原因包括：

- 为提供绕道目标，同时对原始地址进行修复。
- 提供用于 A/B 比较用户界面的测试页面。

> **Note**
>
> 如果您的应用程序返回意外的 302 响应，则该错误很可能是由于您对应用程序的重定向和自定义标头配置所做的更改所致。要解决此问题，请验证您的自定义标头是否有效，然后为您的应用程序重新启用默认 404 重写规则。

重写 (200)

200 重定向（重写）旨在显示来自目标地址的内容，就像它是从原始地址提供的一样。搜索引擎排名历史记录继续应用于原始地址。重定向发生在服务器端上，因此浏览器导航栏在重定向后显示原始地址。使用 200 重定向的常见原因包括：

- 为将整个站点重定向到新的托管位置而不更改站点的地址。
- 为将流向单页 Web 应用程序 (SPA) 的所有流量重定向到其 index.html 页面以由客户端路由器功能处理。

找不到 (404)

当请求指向的地址不存在时，会发生 404 重定向。将显示目标页面 404，而不显示请求的页面。发生 404 重定向的常见原因包括：

- 为在用户输入错误网址时避开失效链接消息。
- 为使对不存在的 Web 应用程序页面的请求指向其 index.html 页面以由客户端路由器功能处理。

了解重定向顺序

重定向按照从列表顶部向下的顺序应用。确保您的排序具有您想要的效果。例如，以下重定向顺序会导致对 /docs/ 下的给定路径的所有请求都重定向到 /documents/ 下的同一路径，但 /docs/specific-filename.html 除外，它会重定向到 /documents/different-filename.html：

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

以下重定向顺序会忽略 specific-filename.html 到 different-filename.html 的重定向：

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

了解 Amplify 如何转发查询参数

您可以使用查询参数来更好地控制网址匹配。Amplify 会将所有查询参数转发到 301 和 302 重定向的目标路径，但以下情况除外：

- 如果原始地址包含设置为特定值的查询字符串，则 Amplify 不会转发查询参数。在这种情况下，重定向仅适用于向具有指定查询值的目标网址发出的请求。

- 如果匹配规则的目标地址包含查询参数，则不会转发查询参数。例如，如果重定向的目标地址为 `https://example-target.com?q=someParam`，则不会传递查询参数。

在 Amplify 控制台中创建和编辑重定向

您可以在 Amplify 控制台中为应用程序创建和编辑重定向。在开始之前，您将需要有关重定向各个部分的以下信息。

原始地址

用户请求的地址。

目标地址

实际提供用户看到的内容的地址。

重定向类型

类型包括永久重定向 (301)、临时重定向 (302)、重写 (200) 或找不到 (404)。

一个由两个字母组成的国家/地区代码（可选）

这个值可以用于按地理区域细分用户对应用程序的体验。

在 Amplify 控制台中创建重定向

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要为其创建重定向的应用程序。
3. 在导航窗格中，依次选择托管、重写和重定向。
4. 在重写和重定向页面上选择管理重定向。
5. 在重写和重定向 JSON 编辑器中，手动添加或更新重定向。
   a. 对于 `source`，请指定用户请求的原始地址。
   b. 对于 `status`，请指定重定向的类型。
   c. 对于 `target`，请指定向用户呈现内容的目标地址。
   d. （可选）对于 `condition`，请输入一个由两个字母组成的国家/地区代码条件。
6. 选择保存。

重定向和重写示例参考

本节提供了有关多种常见重定向情景的示例。

Important

特定于域的重定向不支持 source 字段中的路径组件。
支持：

- `"source": "https://example.com"` （自动附加路径）

不支持：

- `"source": "https://example.com/specific-path"`
- 目前不支持使用 domain+path 组合的规则。

替代模式
对于特定于域的路径重定向，请使用：

1. 仅指定域的单独规则（系统会自动附加路径）
2. 带条件逻辑的仅指定路径的规则
3. 多规则的组合

您可以通过以下示例，来了解在 Amplify 控制台 JSON 编辑器中创建自己的重定向和重写的 JSON 语法。

Note

原始地址域匹配不区分大小写。

主题

- 占位符
- 查询字符串和路径参数
- 基于区域的重定向
- 在重定向和重写中使用通配符表达式

简单重定向和重写

您可以使用以下示例将特定页面永久地重定向到新地址。

原始地址	目标地址	重定向类型	国家/地区代码
/original.html	/destination.html	permanent redirect (301)	

JSON 格式

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

您可以使用以下示例将一个文件夹下的任何路径都重定向到不同文件夹下的同一路径。

原始地址	目标地址	重定向类型	国家/地区代码
/docs/<*>	/documents/<*>	permanent redirect (301)	

JSON 格式

```
[
  {
    "source": "/docs/<*>",
```

```
    "status": "301",
    "target": "/documents/<*>",
    "condition": null
  }
]
```

您可以使用以下示例以重写的方式将所有流量都重定向到 index.html。在此情景中，重写使用户看来他们到达了原始地址。

原始地址	目标地址	重定向类型	国家/地区代码
/<*>	/index.html	rewrite (200)	

JSON 格式

```
[
  {
    "source": "/<*>",
    "status": "200",
    "target": "/index.html",
    "condition": null
  }
]
```

您可以使用以下示例来通过重写更改显示给用户的子域。

原始地址	目标地址	重定向类型	国家/地区代码
https://mydomain.com	https://www.mydomain.com	rewrite (200)	

JSON 格式

```
[
  {
    "source": "https://mydomain.com",
    "status": "200", "target": "https://www.mydomain.com",
```

```
    "condition": null
  }
]
```

您可以使用以下示例重定向到带有路径前缀的其他域。

原始地址	目标地址	重定向类型	国家/地区代码
https://mydomain.com	https://www.mydomain.com/documents	temporary redirect (302)	

JSON 格式

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

您可以使用以下示例将某个找不到的文件夹下的路径重定向到自定义 404 页面。

原始地址	目标地址	重定向类型	国家/地区代码
/<*>	/404.html	not found (404)	

JSON 格式

```
[
  {
    "source": "/<*>",
    "status": "404",
    "target": "/404.html",
    "condition": null
```

```
  }
]
```

> **Important**
>
> 不支持在基于域的源规则中使用路径组件（例如 "https://domain.com/path"），这会导致该规则被忽略且不会返回错误。

单页 Web 应用程序 (SPA) 的重定向

大多数 SPA 框架都支持 HTML5 History.pushState () 在不启动服务器请求的情况下更改浏览器位置。这适用于从根（或 /index.html）开始的用户，但不适用于直接导航到任何其他页面的用户。

以下示例使用正则表达式将所有文件设置到 index.html 的 200 重写，但正则表达式中指定的特定文件扩展名除外。

原始地址	目标地址	重定向类型	国家/地区代码
</^[^.]+$\|\.(?!(css\|gif\|ico\|jpg\|js\|png\|txt\|svg\|woff\|woff2\|ttf\|map\|json\|webp)$)([^.]+$)/>	/index.html	200	

JSON 格式

```
[
  {
    "source": "</^[^.]+$|\.(?!(css|gif|ico|jpg|js|png|txt|svg|woff|woff2|ttf|map|json|
webp)$)([^.]+$)/>",
    "status": "200",
    "target": "/index.html",
    "condition": null
  }
]
```

反向代理重写

以下示例对来自其他位置的代理内容使用重写以便在用户看来域未更改。HTTPS 是唯一支持使用反向代理的协议。

原始地址	目标地址	重定向类型	国家/地区代码
/images/<*>	https://images.otherdomain.com/<*>	rewrite (200)	

JSON 格式

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

尾随斜线然后清理 URLs

为创建清洁网址结构（如 about 而不是 about.html），静态站点生成器（如 Hugo）会为具有 index.html (/about/index.html) 的页面生成目录。Amplify 在需要时 URLs 通过添加尾部斜杠来自动创建干净的效果。下表重点介绍了几种不同情景：

用户在浏览器中的输入	地址栏中的网址	提供的文档
/about	/about	/about.html
/about (when about.html returns 404)	/about/	/about/index.html
/about/	/about/	/about/index.html

占位符

您可以使用以下示例将一个文件夹结构中的路径重定向到另一个文件夹中的匹配结构。

原始地址	目标地址	重定向类型	国家/地区代码
/docs/<year>/<month>/<date>/<itemid>	/documents/<year>/<month>/<date>/<itemid>	permanent redirect (301)	

JSON 格式

```
[
  {
    "source":  "/docs/<year>/<month>/<date>/<itemid>",
    "status": "301",
    "target": "/documents/<year>/<month>/<date>/<itemid>",
               "condition": null
   }
]
```

查询字符串和路径参数

> **Warning**
>
> 请勿在路径或查询参数中 URLs 包含机密、凭证或敏感数据。这些值可在 Amplify 应用的访问日志中以明文方式查看。

您可以使用以下示例将路径重定向到其名称与原始地址中的查询字符串元素值匹配的文件夹：

原始地址	目标地址	重定向类型	国家/地区代码
/docs?id=<my-blog-id-value	/documents/<my-blog-post-id-value>	permanent redirect (301)	

JSON 格式

```
[
  {
    "source": "/docs?id=<my-blog-id-value>",
    "status": "301",
    "target": "/documents/<my-blog-id-value>",
    "condition": null
  }
]
```

> **Note**
>
> Amplify 会将所有查询字符串参数转发到 301 和 302 重定向的目标路径。但是，如果原始地址包含设置为特定值的查询字符串（如本示例所示），Amplify 不会转发查询参数。在这种情况下，重定向仅适用于向具有指定查询值 `id` 的目标地址发出的请求。

您可以使用以下示例将在文件夹结构的给定级别上找不到的所有路径都重定向到指定文件夹中的 index.html。

原始地址	目标地址	重定向类型	国家/地区代码
`/documents/<folder>/<child-folder>/<grand-child-folder>`	`/documents/index.html`	`not found (404)`	

JSON 格式

```
[
  {
    "source": "/documents/<x>/<y>/<z>",
    "status": "404",
    "target": "/documents/index.html",
    "condition": null
  }
```

```
]
```

基于区域的重定向

您可以使用以下示例来基于区域重定向请求。

原始地址	目标地址	重定向类型	国家/地区代码
/documents	/documents/us/	temporary redirect (302)	<US>

JSON 格式

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

在重定向和重写中使用通配符表达式

您可以在原始地址中为重定向或重写使用通配符表达式 <*>。必须将表达式放在原始地址的末尾，而且该表达式必须唯一。Amplify 会忽略包含多个通配符表达式的原始地址，或者将其用在不同的放置中。

以下是使用通配符表达式的有效重定向示例。

原始地址	目标地址	重定向类型	国家/地区代码
/docs/<*>	/documents/<*>	permanent redirect (301)	

以下两个示例展示了使用通配符表达式的无效重定向。

原始地址	目标地址	重定向类型	国家/地区代码
/docs/<*>/content	/documents/<*>/content	permanent redirect (301)	
/docs/<*>/content/<*>	/documents/<*>/content/<*>	permanent redirect (301)	

在 Amplify 应用程序中使用环境变量

环境变量是可以将其添加到应用程序设置中以使其可供 Amplify Hosting 使用的键/值对。作为最佳实践，您可以使用环境变量来公开应用程序配置数据。您添加的所有环境变量都经过加密以防止恶意访问。

Amplify 会对您创建的环境变量强制实施如下约束。

- Amplify 不允许您创建带有 AWS 前缀的环境变量名称。此前缀仅为 Amplify 内部使用而预留。
- 环境变量的值长度不能超过 5500 个字符。

> **Important**
>
> 请勿使用环境变量存储密钥。对于 Gen 2 应用程序，请使用 Amplify 控制台中的密钥管理功能。有关更多信息，请参阅《Amplify 文档》中的密钥和环境变量。对于第 1 代应用程序，将密钥存储在使用 P AWS Systems Manager arameter Store 创建的环境密钥中。有关更多信息，请参阅 管理环境密钥。

Amplify 环境变量参考

Amplify 控制台默认可访问以下环境变量。

变量名称	说明	示例值
_BUILD_TIMEOUT	构建超时时间（以分钟为单位）。 最小值为 5。 最大值为 120。	30
_LIVE_UPDATES	该工具将升级到最新版本。	[{"name":"Amplify CLI","pkg":"@aws-amplify/cli","type":"npm","version":"latest"}]

变量名称	说明	示例值
USER_DISABLE_TESTS	在构建过程中会跳过测试步骤。您可以禁用应用程序中所有分支或特定分支的测试。 此环境变量用于在构建阶段执行测试的应用程序。有关设置此变量的更多信息，请参阅 为 Amplify 应用程序或分支关闭测试。	`true`
AWS_APP_ID	当前构建的应用程序 ID	`abcd1234`
AWS_BRANCH	当前构建的分支名称	`main`, `develop`, `beta`, `v2.0`
AWS_BRANCH_ARN	当前版本的 Amazon 资源名称 (ARN)	`aws:arn:amplify:us-west-2:123456789012:appname/branch/...`
AWS_CLONE_URL	用于提取 Git 存储库内容的克隆网址	`git@github.com:<user-name>/<repo-name>.git`
AWS_COMMIT_ID	当前构建的提交 ID 用于重新构建的“HEAD”	`abcd1234`
AWS_JOB_ID	当前构建的作业 ID。 这包括填充的一些“0”，因此它总是具有相同的长度。	`0000000001`

变量名称	说明	示例值
AWS_PULL_REQUEST_ID	拉取请求 Web 预览版本的拉取请求 ID。 当 AWS CodeCommit 用作存储库提供程序时，此环境变量不可用。	1
AWS_PULL_REQUEST_SOURCE_BRANCH	在 Amplify 控制台中提交到应用程序分支的拉取请求预览的功能分支名称。	`featureA`
AWS_PULL_REQUEST_DESTINATION_BRANCH	在 Amplify 控制台中，功能分支拉取请求所提交到的应用程序分支的名称。	`main`
AMPLIFY_AMAZON_CLIENT_ID	Amazon 客户端 ID	`123456`
AMPLIFY_AMAZON_CLIENT_SECRET	Amazon 客户端密钥	`example123456`
AMPLIFY_FACEBOOK_CLIENT_ID	Facebook 客户端 ID	`123456`
AMPLIFY_FACEBOOK_CLIENT_SECRET	Facebook 客户端密钥	`example123456`
AMPLIFY_GOOGLE_CLIENT_ID	Google 客户端 ID	`123456`
AMPLIFY_GOOGLE_CLIENT_SECRET	Google 客户端密钥	`example123456`
AMPLIFY_DIFF_DEPLOY	启用或禁用基于 diff 的前端部署。有关更多信息，请参阅配置基于 diff 的前端构建和部署。	`true`

变量名称	说明	示例值
AMPLIFY_DIFF_DEPLOY_ROOT	用于比较基于 diff 的前端部署的路径，相对于存储库的根目录。	`dist`
AMPLIFY_DIFF_BACKEND	启用或禁用基于 diff 的后端构建。这仅适用于 Gen 1 应用程序。有关更多信息，请参阅 为 Gen 1 应用程序配置基于 diff 的后端构建。	`true`
AMPLIFY_BACKEND_PULL_ONLY	Amplify 管理此环境变量。这仅适用于 Gen 1 应用程序。有关更多信息，请参阅 编辑现有前端以指向其他后端。	`true`
AMPLIFY_BACKEND_APP_ID	Amplify 管理此环境变量。这仅适用于 Gen 1 应用程序。有关更多信息，请参阅 编辑现有前端以指向其他后端。	`abcd1234`
AMPLIFY_SKIP_BACKEND_BUILD	如果您的构建规范中没有后端部分，并且想要禁用后端构建，请将此环境变量设置为 `true`。这仅适用于 Gen 1 应用程序。	`true`
AMPLIFY_ENABLE_DEBUG_OUTPUT	将此变量设置为 `true`，即可在日志中打印堆栈跟踪。这对于调试后端构建错误非常有用。	`true`
AMPLIFY_MONOREPO_APP_ROOT	用于指定单一存储库应用程序的应用程序根目录的路径，相对于存储库的根目录。	`apps/react-app`

变量名称	说明	示例值
AMPLIFY_USERPOOL_ID	为进行身份验证而导入的 Amazon Cognito 用户群体的 ID	`us-west-2_example`
AMPLIFY_WEBCLIENT_ID	Web 应用程序要使用的应用程序客户端的 ID 必须将应用程序客户端配置为可以访问由 AMPLIFY_USERPOOL_ID 环境变量指定的 Amazon Cognito 用户群体。	`123456`
AMPLIFY_NATIVECLIENT_ID	原生应用程序要使用的应用程序客户端的 ID 必须将应用程序客户端配置为可以访问由 AMPLIFY_USERPOOL_ID 环境变量指定的 Amazon Cognito 用户群体。	`123456`
AMPLIFY_IDENTITYPOOL_ID	Amazon Cognito 身份池的 ID	`example-identitypool-id`
AMPLIFY_PERMISSIONS_BOUNDARY_ARN	用作权限边界的 IAM policy 的 ARN，适用于 Amplify 创建的所有 IAM 角色。	`arn:aws:iam::123456789012:policy/example-policy`
AMPLIFY_DESTRUCTIVE_UPDATES	将此环境变量设置为 true，以允许使用可能导致数据丢失的模式操作更新 GraphQL API。	`true`

> **Note**
>
> `AMPLIFY_AMAZON_CLIENT_ID`和`AMPLIFY_AMAZON_CLIENT_SECRET`环境变量是 OAuth 令牌，而不是 AWS 访问密钥和密钥。

前端框架环境变量

如果您使用支持自有环境变量的前端框架开发应用程序，请务必了解这些变量与您在 Amplify 控制台中配置的环境变量不同。例如，React（前缀 REACT_APP）和 Gatsby（前缀 GATSBY）使您能够创建运行时环境变量，这些框架会自动将这些变量捆绑到您的前端生产构建中。要了解使用这些环境变量存储值的效果，请参阅您所使用的前端框架的文档。

将敏感值（例如 API 密钥）存储在这些以前端框架为前缀的环境变量中并不是最佳做法，因此强烈建议不要这样做。

设置环境变量

按照以下说明，在 Amplify 控制台中为应用程序设置环境变量。

> **Note**
>
> 只有将应用程序设置为持续部署并连接到 git 存储库时，环境变量才会显示在 Amplify 控制台的应用程序设置菜单中。有关此类部署的说明，请参阅现有代码入门。

设置环境变量

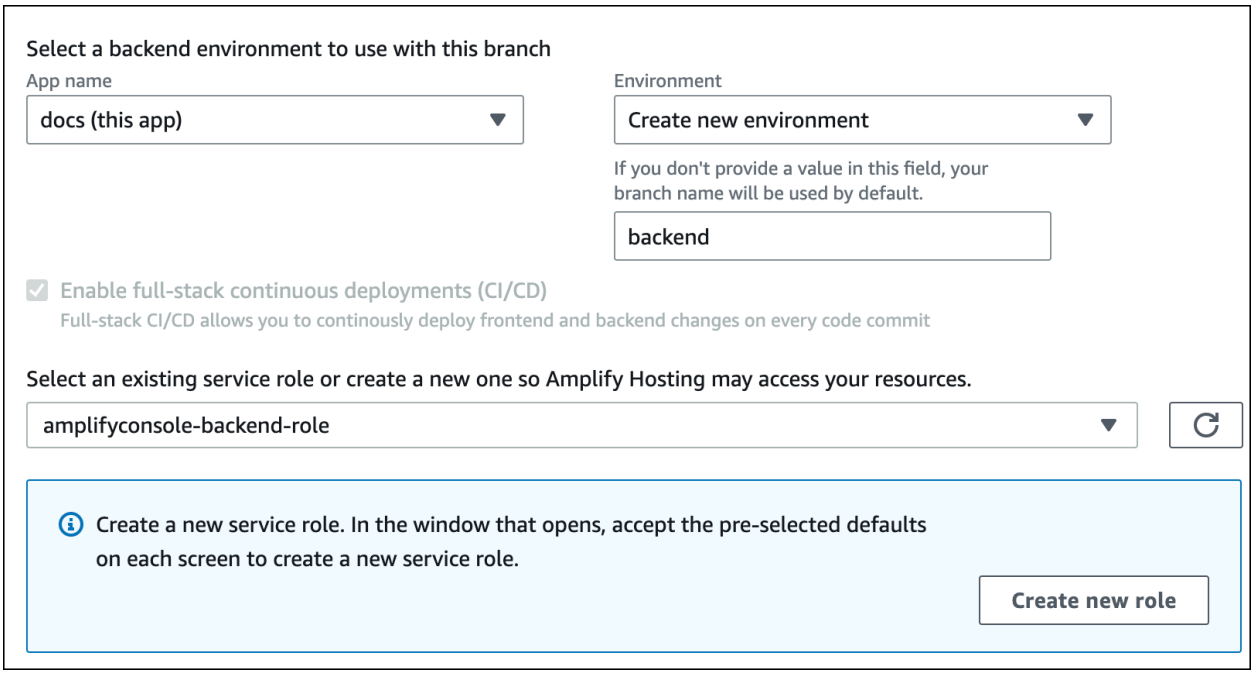
1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 在 Amplify 控制台中，选择托管，然后选择环境变量。
3. 在环境变量页面，选择管理变量。
4. 对于变量，请输入您的密钥。对于值，请输入您的值。默认情况下，Amplify 会将环境变量应用于所有分支，因此在连接新分支时不必重新输入变量。
5. （可选）要专门为某个分支自定义环境变量，请按如下方式添加分支覆盖：
   a. 选择操作，然后选择添加变量覆盖。
   b. 您现在具有一组特定于分支的环境变量。
6. 选择保存。

使用社交登录的身份验证参数创建新的后端环境

将分支连接到应用程序

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 根据您是将分支连接到新应用程序还是现有应用程序，分支与应用程序的连接过程会有所不同。
   - 将分支连接到新应用程序

     a. 在构建设置页面上，找到选择要用于此分支的后端环境部分。对于环境，请选择创建新环境，然后输入您后端环境的名称。以下屏幕截图显示了构建设置页面的选择要用于此分支的后端环境部分，其中输入了 **backend** 作为后端环境名称。



     b. 展开构建设置页面上的高级设置部分，为社交登录密钥添加环境变量。例如，**AMPLIFY_FACEBOOK_CLIENT_SECRET** 是一个有效的环境变量。有关默认可用的 Amplify 系统环境变量列表，请参阅 Amplify 环境变量参考 中的表格。

   - 将分支连接到现有应用程序

     a. 如果您要将新分支连接到现有应用程序，请在连接此分支之前设置社交登录环境变量。在导航窗格中，依次选择应用程序设置、环境变量。

     b. 在环境变量部分中，选择管理变量。

     c. 在管理变量部分中，选择添加变量。

     d. 对于变量（密钥），请输入您的客户端 ID。对于值，输入您的客户端密钥。

     e. 选择保存。

管理环境密钥

随着 Amplify Gen 2 的发布，环境密钥的工作流程得到了简化，可在 Amplify 控制台中集中管理密钥和环境变量。有关设置和访问 Amplify Gen 2 应用程序密钥的说明，请参阅《Amplify 文档》中的密钥和环境变量。

对于 Gen 1 应用程序，环境密钥与环境变量类似，但前者是可加密的 AWS Systems Manager Parameter Store 密钥值对。某些值必须加密，例如 Amplify 的使用 Apple 私钥登录。

用于 AWS Systems Manager 为 Amplify Gen 1 应用程序设置环境密钥

按照以下说明使用控制台为第 1 代 Amplify 应用程序设置环境密钥。 AWS Systems Manager

设置环境密钥

1. 登录 AWS 管理控制台 并打开AWS Systems Manager 控制台。
2. 在导航窗格中，依次选择应用程序管理、参数存储。
3. 在 AWS Systems Manager Parameter Store 页面上，选择创建参数。
4. 在创建参数页面上，在参数详情部分中，执行以下操作：
   a. 在名称中，以格式 **`/amplify/{your_app_id}/{your_backend_environment_name}/{your_parameter_name}`** 输入参数。
   b. 对于类型，选择 SecureString。
   c. 对于 KMS 密钥来源，请选择我的当前账户以使用账户的默认密钥。
   d. 对于值，请输入要加密的密钥值。
5. 选择创建参数。

> **Note**
>
> Amplify 只能访问 `/amplify/{your_app_id}/{your_backend_environment_name}` 在特定构建环境下的密钥。必须指定默认值 AWS KMS key 才能允许 Amplify 解密该值。

访问 Gen 1 应用程序的环境密钥

第 1 代应用程序的环境密钥以 JSON 字符串的形式存储在 `process.env.secrets` 中。

Amplify 环境密钥参考

按照格式 `/amplify/{your_app_id}/{your_backend_environment_name}/AMPLIFY_SIWA_CLIENT_ID` 指定 Systems Manager 参数。

您可以使用 Amplify 控制台默认可访问的以下环境密钥。

变量名称	说明	示例值
AMPLIFY_SIWA_CLIENT_ID	通过 Apple 客户端 ID 登录	`com.yourapp.auth`
AMPLIFY_SIWA_TEAM_ID	通过 Apple 团队 ID 登录	`ABCD123`
AMPLIFY_SIWA_KEY_ID	通过 Apple 密钥 ID 登录	`ABCD123`
AMPLIFY_SIWA_PRIVATE_KEY	通过 Apple 私钥登录	-----开始私钥----- ****..... -----结束私钥-----

为 Amplify 应用程序设置自定义 HTTP 标头

使用自定义 HTTP 标头可以让您能够指定每个 HTTP 响应的标头。响应标头可用于调试、安全和信息等用途。您可以在 Amplify 控制台中指定标头，也可以通过下载和编辑应用程序的 `customHttp.yml` 文件并将其保存在项目的根目录中来指定标头。有关详细步骤，请参阅 设置自定义标头。

以前，为应用程序指定自定义 HTTP 标头的方法是在 Amplify 控制台中编辑构建规范（buildspec），或者下载并更新 `amplify.yml` 文件并将其保存在项目的根目录中。我们强烈推荐将以这种方式指定的自定义标头从 buildspec 和 `amplify.yml` 文件中迁移出来。有关说明，请参阅将自定义标头从构建规范和 amplify.yml 中迁移出来。

主题

- 自定义标头 YAML 参考
- 设置自定义标头
- 将自定义标头从构建规范和 amplify.yml 中迁移出来
- Monorepo 自定义标头要求

自定义标头 YAML 参考

使用以下 YAML 格式指定自定义标头：

```
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'
```

对于 monorepo，请使用以下 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'
```

当您向应用程序添加自定义标头时，您需要为以下内容指定自己的值：

模式

应用于与模式匹配的所有 URL 文件路径的标头。

标头

定义与文件模式相匹配的标头。

键

自定义标头的名称。

值

自定义标头的值。

要了解有关 HTTP 标头的更多信息，请参阅 Mozilla 的 HTTP 标头列表。

设置自定义标头

有两种方法可以为 Amplify 应用程序指定自定义 HTTP 标头。您可以在 Amplify 控制台中指定标头，也可以通过下载和编辑应用程序的 `customHttp.yml` 文件并将其保存在项目的根目录中来指定标头。

要在控制台中为应用程序设置并保存自定义标头

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。

2. 选择要为其设置自定义标头的应用程序。
3. 在导航窗格中依次选择托管和自定义标头。
4. 在自定义标头页面中选择编辑。
5. 在编辑自定义标头窗口中，使用自定义标头 YAML 格式输入自定义标头的信息。

   a. 对于 `pattern`，输入要匹配的模式。

   b. 对于 `key`，输入自定义标头的名称。

   c. 对于 `value`，输入自定义标头的值。
6. 选择保存。
7. 重新部署应用程序以应用新的自定义标头。
   - 对于 CI/CD 应用程序，请导航到要部署的分支并选择重新部署此版本。您也可以从 Git 存储库中执行新构建。
   - 对于手动部署应用程序，请在 Amplify 控制台中再次部署该应用程序。

要在存储库根目录中为应用程序设置并保存自定义标头

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要为其设置自定义标头的应用程序。
3. 在导航窗格中依次选择托管和自定义标头。
4. 在自定义标头页面中选择下载 YML。
5. 在您选择的代码编辑器中打开下载的 `customHttp.yml` 文件，然后使用自定义标头 YAML 格式输入自定义标头的信息。

   a. 对于 `pattern`，输入要匹配的模式。

   b. 对于 `key`，输入自定义标头的名称。

   c. 对于 `value`，输入自定义标头的值。
6. 将编辑后的 `customHttp.yml` 文件保存在项目的根目录中。如果您使用的是 monorepo，请将 `customHttp.yml` 文件保存在存储库的根目录中。
7. 重新部署应用程序以应用新的自定义标头。
   - 对于 CI/CD 应用程序，请从包含新`customHttp.yml`文件的 Git 存储库中执行新构建。
   - 对于手动部署应用程序，请在 Amplify 控制台中再次部署应用程序，并将新 `customHttp.yml` 文件与您上传的构件一起包括在内。

> **Note**
>
> 在 `customHttp.yml` 文件中设置并部署在应用程序根目录中的自定义标头，将覆盖在 Amplify 控制台的自定义标头部分中定义的自定义标头。

安全自定义标头示例

自定义安全标头可以强制执行 HTTPS，防止 XSS 攻击，并保护您的浏览器免受点击劫持。使用以下 YAML 语法将自定义安全标头应用于您的应用程序。

```
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'"
```

设置 Cache-Control 自定义标头

使用 Amplify 托管的应用程序会保留源站发送的 `Cache-Control` 标头，除非您使用自己的自定义标头将其覆盖。Amplify 只会为带有 `200 OK` 状态码的成功响应应用 Cache-Control 自定义标头。这样可以防止系统缓存错误的响应，并将其提供给发出相同请求的其他用户。

您可以手动调整 `s-maxage` 指令，以便更好地控制应用程序的性能和部署可用性。例如，要延长内容在边缘缓存的时间长度，您可以通过更新 `s-maxage` 到比默认 600 秒（10 分钟）更长的值来手动延长生存时间 (TTL)。

要指定 `s-maxage` 的自定义值，请使用以下 YAML 格式。此示例将关联内容保留在边缘缓存 3600 秒 (1 小时)。

```
customHeaders:
  - pattern: '/img/*'
```

```
    headers:
      - key: 'Cache-Control'
        value: 's-maxage=3600'
```

有关使用标头控制应用程序性能的更多信息，请参阅 使用 Cache-Control 标头提高应用程序性能。

将自定义标头从构建规范和 amplify.yml 中迁移出来

以前，要为应用程序指定自定义 HTTP 标头，要么在 Amplify 控制台中编辑构建规范，要么下载并更新 `amplify.yml` 文件并将其保存到项目的根目录中。强烈建议您将自定义标头从构建规范和 `amplify.yml` 文件中迁移出来。

在 Amplify 控制台的自定义标头部分中指定自定义标头，或者通过下载和编辑 `customHttp.yml` 文件来指定自定义标头。

迁移 Amplify 控制台中存储的自定义标头

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要执行自定义标头迁移的应用程序。
3. 在导航窗格中，依次选择托管和构建设置。在应用程序构建规范部分，您可以查看应用程序的构建规范。
4. 选择下载以保存当前构建规范的副本。稍后如果需要恢复任何设置，您可以引用此副本。
5. 下载完成后，选择编辑。
6. 请记下文件中的自定义标头信息，因为稍后将在步骤 9 中使用这些信息。在编辑窗口中，从文件中删除所有自定义标头，然后选择保存。
7. 在导航窗格中，依次选择托管、自定义标头。
8. 在自定义标头页面中选择编辑。
9. 在编辑自定义标头窗口中，输入您在步骤 6 中删除的自定义标头的信息。
10. 选择保存。
11. 重新部署您想要将新自定义标头应用到的任何分支。

将自定义标头从 amplify.yml 迁移到 customHttp.yml

1. 导航到当前部署在应用程序根目录中的 `amplify.yml` 文件。
2. 在选定的代码编辑器中打开 `amplify.yml` 文件。

3. 请记下文件中的自定义标头信息，因为稍后将在步骤 8 中使用这些信息。删除文件中的自定义标头。保存并关闭 文件。
4. 登录 AWS 管理控制台 并打开 Amplify 控制台。
5. 选择要为其设置自定义标头的应用程序。
6. 在导航窗格中，依次选择托管、自定义标头。
7. 在自定义标头页面中选择下载。
8. 在您选择的代码编辑器中打开下载的 `customHttp.yml` 文件，然后输入您在步骤 3 中从 `amplify.yml` 中删除的自定义标头的信息。
9. 将编辑后的 `customHttp.yml` 文件保存在项目的根目录中。如果您使用的是 monorepo，请将文件保存在存储库的根目录中。
10. 重新部署应用程序以应用新的自定义标头。
    - 对于 CI/CD 应用程序，请从包含新`customHttp.yml`文件的 Git 存储库中执行新构建。
    - 对于手动部署应用程序，请在 Amplify 控制台中再次部署该应用程序，并添加包含您上传的构件的新 `customHttp.yml` 文件。

> **Note**
>
> 在 `customHttp.yml` 文件中设置并部署在应用程序根目录中的自定义标头，将覆盖在 Amplify 控制台的自定义标头部分中定义的自定义标头。

Monorepo 自定义标头要求

在 monorepo 中为应用程序指定自定义标头时，请注意以下设置要求：

- monorepo 有一种特定的 YAML 格式。有关正确的语法，请参见 自定义标头 YAML 参考。
- 您可以使用 Amplify 控制台的自定义标头部分，为 monorepo 中的应用程序指定自定义标头。您必须重新部署应用程序才能应用新的自定义标头。
- 作为使用控制台的替代方法，您可以在 `customHttp.yml` 文件中为 monorepo 中的应用程序指定自定义标头。您必须将 `customHttp.yml` 文件保存在存储库的根目录中，然后重新部署应用程序以应用新的自定义标头。`customHttp.yml` 文件中指定的自定义标头会覆盖使用 Amplify 控制台的自定义标头部分指定的任何自定义标头。

管理应用程序的缓存配置

Amplify 使用亚马逊 CloudFront 来管理您的托管应用程序的缓存配置。缓存配置会应用于每个应用程序，以实现优化，获得最佳性能。

2024 年 8 月 13 日，Amplify 发布了对于应用程序缓存效率的改进。有关更多信息，请参阅 CDN 缓存改进，通过 AWS Amplify 托管提高应用程序性能。

下表汇总了 Amplify 对缓存改进发布前后的特定缓存行为的支持。

缓存行为	之前的支持	使用缓存改进时
您可以在 Amplify 控制台或 `customHeaders.yaml` 文件中为应用程序添加自定义标头。您可以覆盖的标头之一是 `Cache-Control`。有关更多信息，请参阅 为 Amplify 应用程序设置自定义 HTTP 标头。	是	是
Amplify 以您在 `customHeaders.yaml` 文件中定义的 `Cache-Control` 标头为准，它们优先于 Amplify 的默认缓存设置。	支持	是
Amplify 以应用程序框架中为动态路由（例如 Next.js SSR 路由）设置的 `Cache-Control` 标头为准。如果在应用程序的 `customHeaders.yaml` 文件中设置了 `Cache-Control` 标头，则该标头优先于 `next.config.js` 文件中的设置。	支持	是

缓存行为	之前的支持	使用缓存改进时
每次部署新的 CI/CD 应用程序都会清除缓存。	支持	是
您可以为应用程序启用性能模式。	是	否 性能模式设置在 Amplify 控制台中已不再可用。但您可以创建用于设置 `s-maxage` 指令的 `Cache-Control` 标头。有关说明，请参阅使用 Cache-Control 标头提高应用程序性能。

下表列出了特定缓存设置的默认值更改。

缓存设置	先前的默认值	缓存改进后的默认值
静态资产的缓存持续时间	2 秒	一年
反向代理响应的缓存持续时间	2 秒	零秒（不缓存）
最大生存时间（TTL）	十分钟	一年

有关 Amplify 如何确定为某个应用程序应用的缓存配置的更多信息，以及有关管理缓存键配置的说明，请参阅以下主题。

主题

- Amplify 如何将缓存配置应用于应用程序
- 管理缓存键 Cookie
- 使用 Cache-Control 标头提高应用程序性能

Amplify 如何将缓存配置应用于应用程序

为了管理应用程序的缓存，Amplify 通过检查应用程序的平台类型和重写规则来确定所提供的内容类型。对于 `Compute` 应用程序，Amplify 还会检查部署清单中的路由规则。

> **Note**
>
> 应用程序的平台类型由 Amplify Hosting 在部署期间进行设置。SSG（静态）应用程序设置为平台类型 `WEB`。SSR 应用程序（Next.js 12 或更高版本）设置为平台类型 `WEB_COMPUTE`。

Amplify 识别以下四种类型的内容，并应用指定托管式缓存策略。

静态

使用 `WEB` 平台的应用程序提供的内容，或 `WEB_COMPUTE` 应用程序中的静态路由。

此内容使用 Amplify-StaticContent 缓存策略。

图像优化

`WEB_COMPUTE` 应用程序中的 `ImageOptimization` 路由提供的图像。

此内容使用 Amplify-ImageOptimization 缓存策略。

计算

`WEB_COMPUTE` 应用程序中的 `Compute` 路由提供的内容。这包括所有服务器端渲染（SSR）的内容。

此内容使用 Amplify-Default 或 Amplify-DefaultNoCookies 缓存策略，具体取决于在 Amplify App 上设置的 `cacheConfig.type` 值。

反向代理

由与反向代理重写自定义规则匹配的路径提供的内容。有关创建此自定义规则的更多信息，请参阅《使用重定向》一章中的反向代理重写。

此内容使用 Amplify-Default 或 Amplify-DefaultNoCookies 缓存策略，具体取决于在 Amplify App 上设置的 `cacheConfig.type` 值。

了解 Amplify 的托管式缓存策略

Amplify 使用以下预定义的托管式缓存策略来优化您的托管应用程序的默认缓存配置。

- Amplify-Default
- Amplify-DefaultNoCookies

- Amplify-ImageOptimization
- Amplify-StaticContent

Amplify-Default 托管式缓存策略设置

在 CloudFront 控制台中查看此政策

此策略适合用于作为 AWS Amplify Web 应用程序的源。

此策略包含以下设置：

- 最小 TTL：0 秒
- 最大 TTL：31536000 秒（一年）
- 原定设置 TTL：0 秒
- 缓存键中包含的标头：
  - `Authorization`
  - `Accept`
  - `CloudFront-Viewer-Country`
  - `Host`
- 缓存键中包含的 Cookie：包含所有 Cookie。
- 缓存键中包含的查询字符串：包含所有查询字符串。
- 缓存压缩对象设置：Gzip 和 Brotli 已启用。

Amplify-DefaultNoCookies 托管缓存策略设置

在 CloudFront 控制台中查看此政策

此策略适合用于作为 AWS Amplify Web 应用程序的源。

此策略包含以下设置：

- 最小 TTL：0 秒
- 最大 TTL：31536000 秒（一年）
- 原定设置 TTL：0 秒
- 缓存键中包含的标头：
  - `Authorization`

  - `Accept`
  - `CloudFront-Viewer-Country`
  - `Host`
- 缓存键中包含的 Cookie：不包含任何 Cookie。
- 缓存键中包含的查询字符串：包含所有查询字符串。
- 缓存压缩对象设置：Gzip 和 Brotli 已启用。

Amplify-ImageOptimization 托管缓存策略设置

在 CloudFront 控制台中查看此政策

此策略适合用于作为 AWS Amplify Web 应用程序的源。

此策略包含以下设置：

- 最小 TTL：0 秒
- 最大 TTL：31536000 秒（一年）
- 原定设置 TTL：0 秒
- 缓存键中包含的标头：
  - `Authorization`
  - `Accept`
  - `Host`
- 缓存键中包含的 Cookie：不包含任何 Cookie。
- 缓存键中包含的查询字符串：包含所有查询字符串。
- 缓存压缩对象设置：Gzip 和 Brotli 已启用。

Amplify-StaticContent 托管缓存策略设置

在 CloudFront 控制台中查看此政策

此策略适合用于作为 AWS Amplify Web 应用程序的源。

此策略包含以下设置：

- 最小 TTL：0 秒
- 最大 TTL：31536000 秒（一年）

- 原定设置 TTL：0 秒
- 缓存键中包含的标头：
  - `Authorization`
  - `Host`
- 缓存键中包含的 Cookie：不包含任何 Cookie。
- 缓存键中包含的查询字符串：不包含任何查询字符串。
- 缓存压缩对象设置：Gzip 和 Brotli 已启用。

管理缓存键 Cookie

将应用程序部署到 Amplify 时，您可以选择是在缓存键中包含还是排除 Cookie。在 Amplify 控制台中，使用缓存键设置切换开关，在自定义标头和缓存页面上指定此设置。有关说明，请参阅[在缓存键中包含或排除 Cookie](#)。

在缓存键中包含 Cookie

使用此设置，Amplify 会根据所提供的内容类型，自动为您的应用程序选择最佳缓存配置。必须明确选择这种缓存配置类型。

如果您使用的是 SDKs 或 AWS CLI，则此设置对应于使用`CreateApp`或`AMPLIFY_MANAGED`的设置`cacheConfig.typeUpdateApp` APIs。

在缓存键中排除 Cookie

这是默认缓存配置。此缓存配置类似于 `AMPLIFY_MANAGED` 配置，不同之处在于它会从缓存键中排除所有 Cookie。

选择从缓存键中排除 Cookie 可以提高缓存性能。但在选择此缓存配置之前，请务必考虑您的应用程序是否使用 Cookie 来提供动态内容。

如果您使用 SDKs 或 AWS CLI，则此设置对应于`AMPLIFY_MANAGED_NO_COOKIES`使用`CreateApp`或`cacheConfig.type`将设置为`UpdateApp` APIs。

有关缓存密钥的更多信息，请参阅 Amazon CloudFront 开发者指南中的[了解缓存密钥](#)；。

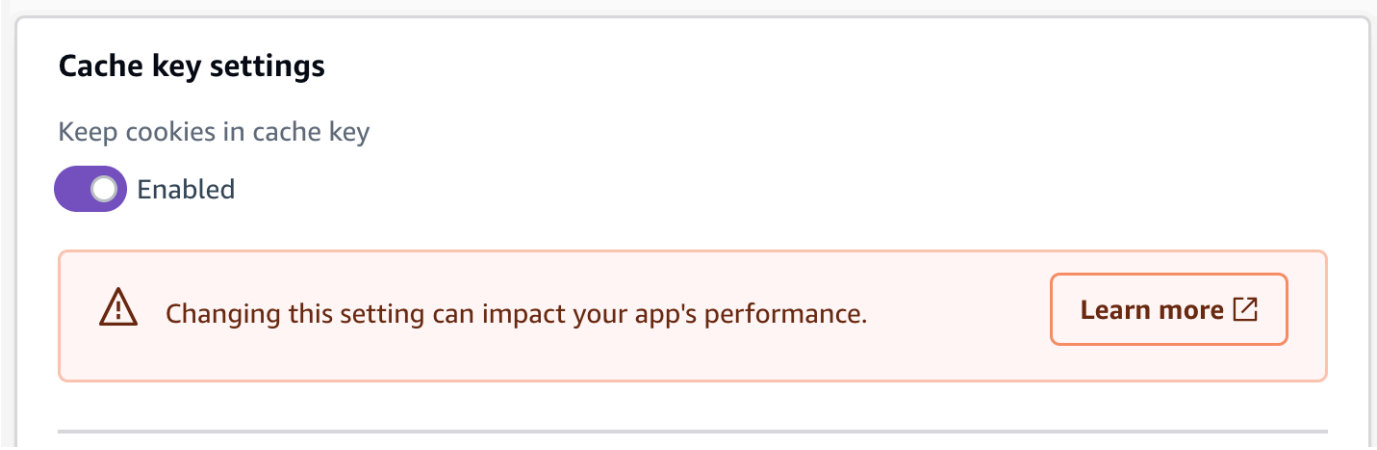
在缓存键中包含或排除 Cookie

你可以在 Amplify 控制台中为应用程序设置缓存密钥 cookie 配置 SDKs，或者。 AWS CLI

使用以下步骤，通过 Amplify 控制台指定在部署新应用程序时，是在缓存键中包含还是排除 Cookie。

将应用程序部署到 Amplify 时设置缓存键 Cookie 配置

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 在所有应用程序页面中，选择创建新应用程序。
3. 在开始使用 Amplify 进行构建页面中选择您的 Git 存储库提供商，然后选择下一步。
4. 在添加存储库分支页面上，执行以下操作：
   a. 选择需要连接的存储库的名称。
   b. 选择需要连接的存储库分支的名称。
   c. 选择下一步。
5. 如果应用程序需要 IAM 服务角色，您可以允许 Amplify Hosting 计算自动为您创建服务角色，也可以指定您已创建的角色。
   - 允许 Amplify 自动创建角色并将其附加到您的应用程序的方法：
     - 请选择创建和使用新的服务角色。
   - 附加您之前创建的服务角色的方法：
     a. 选择使用现有服务角色。
     b. 从列表中选择需要使用的角色。
6. 选择高级设置，然后找到缓存键设置部分。
7. 选择在缓存键中保留 Cookie 或从缓存键中删除 Cookie。以下屏幕截图显示了控制台中的缓存键设置切换开关。



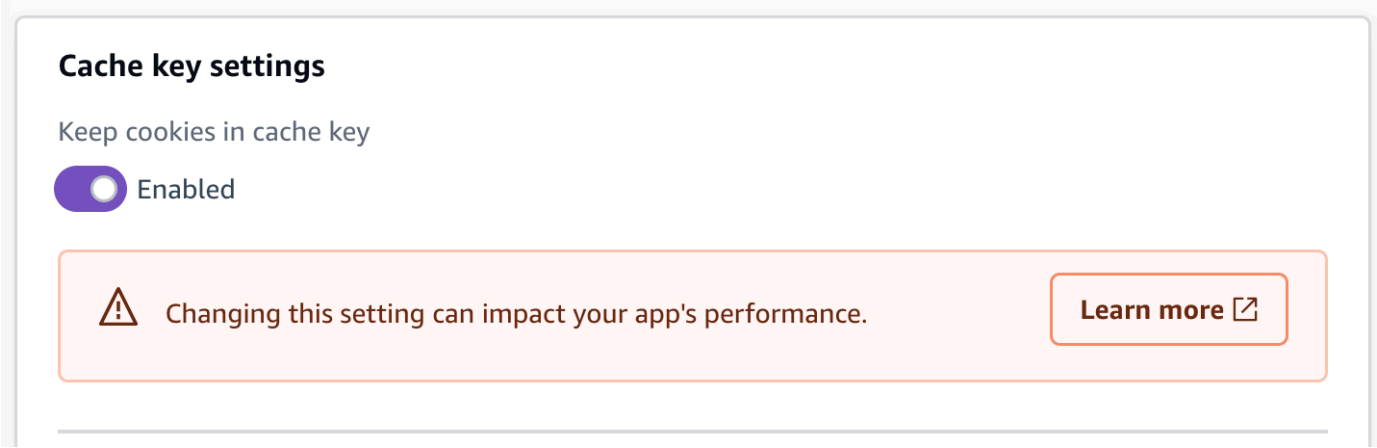
8. 选择下一步。
9. 在查看页面上，选择保存并部署。

更改应用程序的缓存键 Cookie 配置

您可以为已部署到 Amplify 的应用程序更改缓存键 Cookie 配置。使用以下步骤，通过 Amplify 控制台更改是在缓存键中包含还是排除 Cookie。

更改已部署应用程序的缓存键 Cookie 配置的方法

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 在所有应用程序页面上，选择要更新的应用程序。
3. 在导航窗格中依次选择托管和自定义标头和缓存。
4. 在自定义标头和缓存页面上，找到缓存键设置部分，然后选择编辑。
5. 选择在缓存键中保留 Cookie 或从缓存键中删除 Cookie。以下屏幕截图显示了控制台中的缓存键设置切换开关。



6. 选择保存。

使用 Cache-Control 标头提高应用程序性能

Amplify 的默认托管架构优化了托管性能和部署可用性之间的平衡。对于大多数客户，我们均建议使用默认架构。

如果您需要更精细地控制应用程序的性能，则可以手动设置 HTTP `Cache-Control` 标头，从而将在内容分发网络（CDN）边缘处缓存的内容保留更长的时间间隔，以此优化托管性能。

HTTP `Cache-Control` 标头的 `max-age` 和 `s-maxage` 指令会影响应用程序的内容缓存持续时间。`max-age` 指令告诉浏览器在从原始服务器中刷新内容之前希望在缓存中保留内容的时间长度（以秒为单位）。`s-maxage` 指令覆盖 `max-age`，让您指定在原始服务器中刷新内容之前希望内容在 CDN 边缘保留内容的时间长度（以秒为单位）。

使用 Amplify 托管的应用程序会保留源站发送的 Cache-Control 标头，除非您使用自己的自定义标头将其覆盖。Amplify 只会为带有 200 OK 状态码的成功响应应用 Cache-Control 自定义标头。这样可以防止系统缓存错误的响应，并将其提供给发出相同请求的其他用户。

您可以手动调整 s-maxage 指令，以便更好地控制应用程序的性能和部署可用性。例如，要更改内容在边缘缓存的时间长度，您可以将 s-maxage 更新为到默认 31536000 秒（一年）以外的值，从而手动设置生存时间（TTL）。

您可以在 Amplify 控制台的自定义标头部分为应用程序定义自定义标头。有关 YAML 格式的示例，请参阅 设置 Cache-Control 自定义标头。

使用以下步骤设置 s-maxage 指令，使内容在 CDN 边缘缓存 24 小时。

设置自定义 Cache-Control 标头

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要为其设置自定义标头的应用程序。
3. 在导航窗格中，依次选择托管、自定义标头。
4. 在自定义标头页面中选择编辑。
5. 在编辑自定义标头窗口中，按如下方式输入自定义标头的信息：
   a. 对于 pattern，输入所有路径的 **`**/*`**。
   b. 对于 key，输入 **`Cache-Control`**。
   c. 对于 value，输入 **`s-maxage=86400`**。
6. 选择保存。
7. 重新部署应用程序以应用新的自定义标头。

Amplify 部署的倾斜保护

Amplify 应用程序可以使用部署倾斜保护功能来消除 Web 应用程序中客户端和服务器之间的版本倾斜问题。对 Amplify 应用程序应用倾斜保护后，无论何时进行部署，都可以确保客户端始终与正确版本的服务器端资源进行交互。

版本倾斜是 Web 开发者面临的一个常见挑战。当 Web 浏览器运行应用程序的过时版本而服务器运行新版本时，就会发生这种情况。这种差异可能会导致不可预测的行为和错误，并损害应用程序的用户体验。Amplify 部署倾斜保护功能会使在 Web 浏览器上运行的客户端固定为特定的部署。这样可以确保 Amplify 始终提供该特定部署的资源，从而保持客户端和服务器的同步。

随着您不断新部署，Amplify 的倾斜保护功能可以减少应用程序用户遇到的错误。此外还可以减少管理向后和向前兼容性问题所需的时间，从而提升开发者体验。

倾斜保护功能详情：

支持的应用程序类型

> 您可以为使用 Amplify 支持的任何框架创建的静态和 SSR 应用程序添加倾斜保护。应用程序可以从 Git 存储库部署，也可以手动部署。
>
> 您无法为部署到 `WEB_DYNAMIC` 平台（Next.js 版本 11 或更早版本）的应用程序添加倾斜保护。

Duration

> 对于静态应用程序，Amplify 提供为期一周的部署倾斜保护。对于 SSR 应用程序，我们保证为不超过八个早期的部署提供倾斜保护。

成本

> 向应用程序添加倾斜保护不会产生额外成本。

性能注意事项

> 为应用程序启用倾斜保护后，Amplify 必须更新其 CDN 缓存配置。因此在启用倾斜保护后，预计首次部署最长会需要十分钟时间。

主题

为 Amplify 应用程序配置部署倾斜保护

您可以使用 Amplify 控制台、 AWS Command Line Interface或，为应用程序添加或移除部署偏斜保护。 SDKs此功能将在分支级别应用。只有在为分支启用倾斜保护后进行的新部署才会受到倾斜保护。

要使用或添加或移除部署偏差保护 SDKs，请使用`CreateBranch.enableSkewProtection`和`UpdateBranch.enableSkewProtection`字段。AWS CLI 有关更多信息，请参阅 Amplify API 参考文档UpdateBranch中的CreateBranch和。

如果要移除特定的部署以不再为其提供服务，请使用 `DeleteJob` API。有关更多信息，请参阅 Amplify API 参考文档DeleteJob中的。

目前只能为已部署到 Amplify Hosting 的应用程序启用倾斜保护。请按照以下说明操作，通过 Amplify 控制台为分支添加倾斜保护。

为 Amplify 应用程序的分支启用倾斜保护

1. 登录 AWS 管理控制台 并打开 Amplify 控制台，网址为。https://console.aws.amazon.com/amplify/
2. 在所有应用程序页面上，选择要启用倾斜保护的已部署应用程序的名称。
3. 在导航窗格中，依次选择应用程序设置和分支设置。
4. 在分支部分中，选择需要更新的分支的名称。
5. 在操作菜单上，选择启用倾斜保护。
6. 在确认窗口中选择确认。现在已为该分支启用倾斜保护。
7. 重新部署应用程序的分支。只有在启用倾斜保护之后进行的部署才会受到倾斜保护。

请按照以下说明操作，通过 Amplify 控制台为应用程序的分支移除倾斜保护。

为 Amplify 应用程序的分支移除倾斜保护

1. 登录 AWS 管理控制台 并打开 Amplify 控制台，网址为。https://console.aws.amazon.com/amplify/
2. 在所有应用程序页面上，选择要为其移除倾斜保护的已部署应用程序的名称。
3. 在导航窗格中，依次选择应用程序设置和分支设置。
4. 在分支部分中，选择需要更新的分支的名称。
5. 在操作菜单上，选择禁用倾斜保护。现在已为该分支禁用倾斜保护，将只提供最新内容。

倾斜保护的工作原理

在大多数情况下，_dpl cookie 的默认行为即可满足您的倾斜保护需求。但对于以下高级应用场景，建议使用 X-Amplify-Dpl 标头和 dpl 查询参数启用倾斜保护。

- 同时在多个浏览器选项卡中加载您的网站
- 使用服务 Worker 节点

在确定要提供给客户端的内容时，Amplify 会按以下顺序评估传入的请求：

1. **X-Amplify-Dpl** 标头：应用程序可以使用此标头将请求定向到特定的 Amplify 部署。可以使用 process.env.AWS_AMPLIFY_DEPLOYMENT_ID 的值来设置此请求标头。
2. **dpl** 查询参数：对于对指纹资源（.js 和.css 文件）的请求，Next.js 应用程序会自动设置 _dpl 查询参数。
3. _dpl cookie：这是所有受倾斜保护的应用程序的默认设置。对于特定的浏览器，会为每个与域交互的浏览器选项卡或实例发送相同的 Cookie。

   请注意，如果不同的浏览器选项卡加载了同一个网站的不同版本，则所有选项卡都会共享 _dpl cookie。对于此场景，使用 _dpl cookie 无法实现完全的倾斜保护，而应考虑使用 X-Amplify-Dpl 标头进行倾斜保护。

X-Amplify-Dpl 标题示例

以下示例演示了通过 X-Amplify-Dpl 标头来访问倾斜保护功能的 Next.js SSR 页面的代码。该页面根据其一个 api 路由来呈现其内容。要提供给 api 路由的部署是使用 X-Amplify-Dpl 标头来指定的，其值设置为 process.env.AWS_AMPLIFY_DEPLOYMENT_ID。

```
import { useEffect, useState } from 'react';

export default function MyPage({deploymentId}) {
    const [data, setData] = useState(null);

    useEffect(() => {
        fetch('/api/hello', {
            headers: {
                'X-Amplify-Dpl': process.env.AWS_AMPLIFY_DEPLOYMENT_ID
            },
        })
```

```
        .then(res => res.json())
        .then(data => setData(data))
        .catch(error => console.error("error", error))
    }, []);

    return <div>
        {data ? JSON.stringify(data) : "Loading ... " }
    </div>
}
```

监控 Amplify 应用程序

AWS Amplify 提供以下用于监控托管应用程序的功能：

- CloudWatch 指标 — Amplify 通过亚马逊发布指标 CloudWatch ，您可以使用这些指标来监控应用程序的流量、错误、数据传输和延迟。
- 访问日志：Amplify 提供访问日志，其中包含向应用程序发出请求的详细信息。
- CloudTrail 日志 — Amplify 与之集成 AWS CloudTrail ，可记录用户、角色或 AWS 服务在 Amplify 中执行的操作。您可以在 CloudTrail 控制台中查看这些事件。

主题

使用亚马逊监控 Amplify 应用程序 CloudWatch

AWS Amplify 已与 Amazon 集成 CloudWatch，使您能够近乎实时地监控 Amplify 应用程序的指标，并创建警报，以便在指标超过您设置的阈值时发送通知。有关该 CloudWatch 服务工作原理的更多信息，请参阅 Amazon CloudWatch 用户指南。

支持的 CloudWatch 指标

Amplify 支持`AWS/AmplifyHosting`命名空间中的七个 CloudWatch 指标，用于监控应用的流量、错误、数据传输、延迟和请求令牌。这些指标每隔一分钟汇总一次。 CloudWatch 监控指标是免费的，不计入CloudWatch 服务配额。

下表中介绍了各项受支持的指标，并列出了最相关的统计数据。并非所有可用的统计数据都适用于每个指标。

指标	说明
请求	您的应用程序收到的查看器请求总数。 最相关的统计数据是 Sum。使用 Sum 统计数据可获取请求总数。

指标	说明
BytesDownloaded	查看器为 GET、HEAD 和 OPTIONS 请求从您的应用程序传出（下载）的数据总量，以字节为单位。 最相关的统计数据是 Sum。
BytesUploaded	传输到应用程序（上传）的任何请求的数据总量，包括表头在内，以字节为单位。 Amplify 不会针对应用程序中上传的数据向您收取费用。 最相关的统计数据是 Sum。
4xxErrors	在 HTTP 状态码 400-499 范围内返回错误的请求数。 最相关的统计数据是 Sum。使用 Sum 统计数据以得出这些错误的总出现次数。
5xxErrors	在 HTTP 状态码 500-599 范围内返回错误的请求数。 最相关的统计数据是 Sum。使用 Sum 统计数据以得出这些错误的总出现次数。
延迟	第一个字节的时间（以秒为单位）。这是 Amplify Hosting 收到请求到它将响应返回给网络之间所经过的总时间。这还不包括响应到达查看者设备时遇到的网络延迟。 最相关的统计数据是 Average、Maximum、Minimum、p10、p50、p90、p9 和 p100。 使用 Average 统计数据可评估预期延迟。

指标	说明
TokensConsumed	应用程序使用的请求令牌数。 Sum 统计数据指示总请求令牌消耗量。您可以将此统计数据与您当前的 Request tokens per second 服务配额进行比较，确定是否需要申请增加配额，以免在未来的高流量事件中潜在被节流。 Average 统计数据指示正常和高峰时段的请求令牌消耗量。通常，令牌消耗量越高，首字节时间（TTFB）将会更长。因此，您可以在评估应用程序的延迟时使用此统计数据。如果延迟较差，则可以改善下游 APIs 以减少令牌消耗，并避免在令牌消耗量超过应用程序的Request tokens per second服务配额时可能出现的限制。 有关 Request tokens per second 服务配额的更多信息，请参阅 Amplify 托管服务限额。

Amplify 提供以下 CloudWatch 指标维度。

维度	说明
应用程序	指标数据由应用程序提供。
AWS 账户	中提供了所有应用程序的指标数据 AWS 账户。

访问 CloudWatch 指标

您可以按照以下步骤直接从 Amplify 控制台访问 CloudWatch 指标。

> **Note**
>
> 您也可以在中 AWS 管理控制台 访问 CloudWatch 指标https://console.aws.amazon.com/cloudwatch/。

在 Amplify 控制台中访问指标

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要查看其指标的应用程序。
3. 在导航窗格中，依次选择监控和指标。

创建 CloudWatch 警报

您可以在 Amplify 控制台中创建 CloudWatch 警报，在满足特定条件时发送通知。警报会监视单个 CloudWatch 指标，并在该指标超过指定评估周期的阈值时发送 Amazon Simple Notification Service 通知。

您可以使用 CloudWatch控制台中的指标数学表达式或使用创建更高级的警报 CloudWatch APIs。例如，您可以创建一个警报，以在连续三个期间内的 4xxErrors 百分比超过 15% 时通知您。有关更多信息，请参阅 Amazon CloudWatch 用户指南中的基于指标数学表达式创建 CloudWatch 警报。

标准 CloudWatch 定价适用于警报。有关更多信息，请参阅 Amazon CloudWatch 定价。

使用以下步骤在 Amplify 控制台中创建一个警报。

为 Amplify 指标创建 CloudWatch 警报

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要设置警报的应用程序。
3. 在导航窗格中，依次选择监控和警报。
4. 在警报页面中，选择创建警报。
5. 在创建警报窗口中，按以下方式配置您的警报：
   a. 对于指标，从列表中选择要监控的指标的名称。
   b. 对于警报名称，输入有意义的警报名称。例如，如果您正在监视请求，则可以为警报命名 **`HighTraffic`**。名称只能包含 ASCII 字符。

c. 对于设置通知，执行以下操作之一：

- i. 选择新建以设置新 Amazon SNS 主题。
  ii. 对于电子邮箱地址中，输入通知收件人的电子邮箱地址。
  iii. 选择添加新电子邮箱地址以添加其他收件人。
- i. 选择现有以重复使用 Amazon SNS 主题。
  ii. 对于 SNS 主题，从列表中选择现有 Amazon SNS 主题的名称。

d. 对每当指标的统计数据，按如下方式设置警报的条件：

i. 指定指标是否必须大于、小于或等于阈值。
ii. 指定阈值。
iii. 指定为了调用警报而必须处于警报状态的连续评估期限数。
iv. 然后指定评估期限的时间长度。

e. 选择确认。

> **Note**
>
> 您指定的每个 Amazon SNS 收件人都会收到一封来自 AWS 通知的确认电子邮件。电子邮件包含一个链接，收件人必须遵循该链接以确认其订阅并接收通知。

访问 SSR 应用程序的 CloudWatch 日志

Amplify 将有关您的 SSR CloudWatch 运行时的信息发送到您的 Amazon Logs。 AWS 账户当您将 SSR 应用程序部署到 Amplify Hosting 计算时，该应用程序将需要一个 IAM 服务角色，以便 Amplify 在代表您调用其他服务时代入该角色。您可以允许 Amplify Hosting 计算自动为您创建服务角色，也可以指定您已创建的角色。

如果您选择允许 Amplify 为您创建 IAM 角色，则该角色将已经拥有创建 CloudWatch 日志的权限。如果您创建自己的 IAM 角色，则需要在策略中添加以下权限以允许 Amplify 访问亚马逊 CloudWatch 日志。

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

有关添加服务角色的更多信息，请参阅添加具有后端资源部署权限的服务角色。有关部署服务端渲染的应用程序的更多信息，请参阅 使用 Amplify Hosting 部署在服务器端渲染的应用程序。

你可以在 CloudWatch控制台或 Amplify 控制台中查看 SSR 应用程序的 Amplify Hosting 计算日志。按照以下说明操作，在 Amplify 控制台中查看日志。

在 Amplify 控制台中查看 SSR 应用程序的 CloudWatch 日志

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要查看其 CloudWatch 日志的 SSR 应用程序。
3. 在导航窗格中，依次选择监控和 Hosting 计算日志。
4. 在托管计算日志页面上，搜索并选择特定分支的 CloudWatch日志组。

检索和分析 Amplify 应用程序的访问日志

Amplify 会存储您在 Amplify 中托管的所有应用程序的访问日志。访问日志包含有关对托管应用程序做出的请求的信息。在您删除应用程序之前，Amplify 会保留此应用程序的所有访问日志。Amplify 控制台中提供了一个应用程序的所有访问记录。但访问日志的每个请求均限于您指定的两周时间段。

> **Warning**
>
> 请勿在路径或查询参数中 URLs 包含机密、凭证或敏感数据。这些值可在 Amplify 应用的访问日志中以明文方式查看。

Amplify 从不重复使用客户之间的 CloudFront 分配。Amplify 会提前创建 CloudFront发行版，这样在部署新应用程序时您就不必等待 CloudFront 分配的创建了。在将这些分配分配给 Amplify 应用程序之前，它们可能会收到来自机器人的流量。但是，它们被配置为在分配之前始终以未找到响应。如果您的应用程序的访问日志包含您创建应用程序之前一段时间的条目，则这些条目与该活动相关。

> **Important**
>
> 建议您使用日志来了解内容的请求性质，而不是作为所有请求的完整描述。Amplify 将尽力提供访问日志。特定请求的日志条目可能会在实际处理该请求之后很久才进行传输，而且极少数情况下，可能根本不会传输日志条目。如果访问日志中省略了某个日志条目，则访问日志中的条目数将与 AWS 账单和使用情况报告中显示的使用量不匹配。

检索应用程序的访问日志

使用以下步骤检索 Amplify 应用程序的访问日志。

查看访问日志

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择您要查看日志的应用程序。
3. 在导航窗格中，依次选择监控和访问日志。
4. 选择编辑时间范围。
5. 在编辑时间范围窗口中执行以下操作。
   a. 为开始日期指定要检索日志的两周间隔的第一天。
   b. 在开始时间中，选择第一天开始检索日志的时间。
   c. 选择确认。
6. Amplify 控制台在访问日志部分显示您指定时间范围内的日志。选择下载，以 CSV 格式保存日志。

分析访问日志

要分析访问日志，您可以将 CSV 文件存储在 Amazon S3 存储桶中。分析访问日志的一种方式是使用 Athena。Athena 是一项交互式查询服务，可以帮助您分析服务数据。 AWS 您可以按照此处的step-by-step 说明创建表格。创建表格后，您可以按以下方式查询数据。

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

使用记录 Amplify API 调用 AWS CloudTrail

AWS Amplify 与一项服务集成 AWS CloudTrail，该服务提供用户、角色或 AWS 服务在 Amplify 中执行的操作的记录。 CloudTrail 将 Amplify 的所有 API 调用捕获为事件。捕获的调用包括来自 Amplify 控制台的调用和对 Amplify API 操作的代码调用。如果您创建了跟踪，则可以允许将 CloudTrail事件持续传输到 Amazon S3 存储桶，包括 Amplify 的事件。如果您未配置跟踪，您仍然可以在 CloudTrail 控制

台的 “事件历史记录” 中查看最新的事件。利用 CloudTrail 收集到的信息，您可以确定向 Amplify 发出的请求、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。

要了解更多信息 CloudTrail，请参阅AWS CloudTrail 用户指南。

在 Amplify 中放大信息 CloudTrail

CloudTrail 默认情况下，您的 AWS 账户已启用。当 Amplify 中发生活动时，该活动会与其他 AWS 服务 CloudTrail 事件一起记录在事件历史记录中。您可以在 AWS 账户中查看、搜索和下载最新事件。有关更多信息，请参阅《AWS CloudTrail 用户指南》中的使用 CloudTrail 事件历史记录查看事件。

要持续记录您的 AWS 账户中的事件，包括 Amplify 的事件，请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪时，此跟踪应用于所有 AWS 区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅《AWS CloudTrail 用户指南》中的以下内容：

- 为您的 AWS 账户创建跟踪
- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个地区的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail 日志文件

所有 Amplify 操作均由《AWS Amplify 控制台 API 参考》、《Amplify 管理界面 API 参考》 CloudTrail 和《Am AWS plify UI Builder API 参考》记录并记录在案。例如，调用`DeleteApp`和`DeleteBackendEnvironment`操作会在 CloudTrail 日志文件中生成条目。`CreateApp`

每个事件或日志条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根或 AWS Identity and Access Management (IAM) 用户证书发出的。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是由其他 AWS 服务机构发出的。

有关更多信息，请参阅《CloudTrail 用户指南》中的 “用户身份”AWS CloudTrail 元素。

了解 Amplify 日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序出现。

以下示例显示了一个演示 AWS Amplify 控制台 API 参考`ListApps`操作的 CloudTrail 日志条目。


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::444455556666:user/Mary_Major",
        "accountId": "444455556666",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Mary_Major",
        "sessionContext": {
            "sessionIssuer": {},
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-01-12T05:48:10Z"
            }
        }
    },
    "eventTime": "2021-01-12T06:47:29Z",
    "eventSource": "amplify.amazonaws.com",
    "eventName": "ListApps",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.255",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.898
Linux/4.9.230-0.1.ac.223.84.332.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.275-b01
java/1.8.0_275 vendor/Oracle_Corporation",
    "requestParameters": {
        "maxResults": "100"
    },
    "responseElements": null,
    "requestID": "1c026d0b-3397-405a-95aa-aa43aexample",
    "eventID": "c5fca3fb-d148-4fa1-ba22-5fa63example",
    "readOnly": true,
    "eventType": "AwsApiCall",
```

```
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "444455556666"
}
```

以下示例显示了演示 “ AWS Amplify 管理员 UI API 参考” ListBackendJobs操作的 CloudTrail 日志条目。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::444455556666:user/Mary_Major",
        "accountId": "444455556666",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Mary_Major",
        "sessionContext": {
            "sessionIssuer": {},
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-01-13T00:47:25Z"
            }
        }
    },
    "eventTime": "2021-01-13T01:15:43Z",
    "eventSource": "amplifybackend.amazonaws.com",
    "eventName": "ListBackendJobs",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.255",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.898
 Linux/4.9.230-0.1.ac.223.84.332.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.275-b01
 java/1.8.0_275 vendor/Oracle_Corporation",
    "requestParameters": {
        "appId": "d23mv2oexample",
        "backendEnvironmentName": "staging"
    },
    "responseElements": {
        "jobs": [
            {
                "appId": "d23mv2oexample",
                "backendEnvironmentName": "staging",
```

```
                "jobId": "ed63e9b2-dd1b-4bf2-895b-3d5dcexample",
                "operation": "CreateBackendAuth",
                "status": "COMPLETED",
                "createTime": "1610499932490",
                "updateTime": "1610500140053"
            },
            {
                "appId": "d23mv2oexample",
                "backendEnvironmentName": "staging",
                "jobId": "06904b10-a795-49c1-92b7-185dfexample",
                "operation": "CreateBackend",
                "status": "COMPLETED",
                "createTime": "1610499657938",
                "updateTime": "1610499704458"
            }
        ],
        "appId": "d23mv2oexample",
        "backendEnvironmentName": "staging"
    },
    "requestID": "7adfabd6-98d5-4b11-bd39-c7deaexample",
    "eventID": "68769310-c96c-4789-a6bb-68b52example",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "444455556666"
}
```

将 IAM 角色与 Amplify 应用程序结合使用

IAM 角色是一种具有特定权限的 IAM 身份。角色所具有的权限决定了该身份在 AWS中可以执行和不可执行的操作。您可以在 AWS 账户 中创建 IAM 角色并使用这些角色向 Amplify Hosting 委派权限。有关角色的更多信息，请参阅《IAM 用户指南》中的 IAM 角色。

您可以使用以下类型的 IAM 角色向 Amplify Hosting 授予所需的权限，以代表您执行操作或运行访问其他 AWS 资源的计算代码。

IAM 服务角色

Amplify 会代入此角色来代表您执行操作。具有后端资源的应用程序需要此角色。

IAM SSR 计算角色

允许服务器端渲染（SSR）应用程序安全地访问特定的 AWS 资源。

IAM SSR CloudWatch 日志角色

当您部署 SSR 应用程序时，该应用程序需要一个 Amplify 代入的 IAM 服务角色才能允许 Amplify 访问亚马逊日志。 CloudWatch

主题

添加具有后端资源部署权限的服务角色

Amplify 需要使用前端部署后端资源的权限。可使用服务角色来实现此目的。服务角色是 AWS Identity and Access Management (IAM) 角色，它为 Amplify Hosting 提供代表您部署、创建和管理后端的权限。

创建需要 IAM 服务角色的新应用程序时，您可以允许 Amplify Hosting 自动为您创建服务角色，也可以指定您已创建的 IAM 角色。本节将介绍如何创建一个具有账户管理员权限，并显式允许直接访问 Amplify 应用程序部署、创建和管理后端所需的资源的 Amplify 服务角色。

在 IAM 控制台中创建 Amplify 服务角色

创建服务角色

1. 打开 IAM 控制台并从左侧导航栏中选择角色，然后选择创建角色。
2. 在选择可信实体页面上，选择 AWS 服务。在使用案例下，选择 Amplify - 后端部署，然后选择下一步。
3. 在添加权限页面上，选择下一步。
4. 在名称、查看和创建页面中，对于角色名称，输入一个有意义的名称，例如 **AmplifyConsoleServiceRole-AmplifyRole**。
5. 接受所有默认值，然后选择创建角色。
6. 返回 Amplify 控制台，将该角色附加到您的应用程序。
   - 如果您正在部署新应用程序，请执行以下操作：
     a. 刷新服务角色列表。
     b. 选择您刚刚创建的角色。在这个例子中，它应该看起来像 AmplifyConsoleServiceRole-AmplifyRole。
     c. 选择下一步，然后按照步骤完成应用程序部署。
   - 如果您具备现有应用程序，请执行以下操作：
     a. 在导航窗格中，依次选择应用程序设置、IAM 角色。
     b. 在 IAM 角色页面的服务角色部分中，选择编辑。
     c. 在服务角色页面的服务角色列表中，选择您刚刚创建的角色。
     d. 选择保存。
7. Amplify 现在有权为您的应用程序部署后端资源。

编辑服务角色的信任策略，以防止混淆代理

混淆代理问题是一个安全性问题，即不具有某操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。有关更多信息，请参阅 防止跨服务混淆代理。

目前，`Amplify-Backend Deployment` 服务角色的默认信任策略强制执行 `aws:SourceArn` 和 `aws:SourceAccount` 全局上下文条件键，以免出现混淆代理。但是，如果您之前曾在账户中创建过 `Amplify-Backend Deployment` 角色，就可以更新该角色的信任策略以添加这些条件，以免出现混淆代理。

使用以下示例来限制对账户中应用程序的访问。在示例中，将区域和应用程序 ID 替换为您自己的信息。

```
"Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:amplify:us-east-1:123456789012:apps/*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
```

有关使用编辑角色信任策略的说明 AWS 管理控制台，请参阅 IAM 用户指南中的修改角色（控制台）。

添加 SSR 计算角色以允许访问资源 AWS

通过这一集成，您可以为 Amplify SSR 计算服务分配 IAM 角色，从而让服务器端渲染（SSR）应用程序能够基于该角色的权限安全地访问特定的 AWS 资源。例如，根据分配的 IAM 角色中定义的权限，您可以允许应用程序的 SSR 计算函数安全地访问其他 AWS 服务 Amazon Bedrock 或资源，例如或 Amazon S3 存储桶。

IAM SSR 计算角色会提供临时凭证，因此无需在环境变量中对长期存在的安全凭证进行硬编码。使用 IAM SSR Compute 角色符合授予最低权限权限和尽可能使用短期证书 AWS 的安全最佳实践。

本节后面的说明将介绍如何创建具有自定义权限的策略并将该策略附加到角色。创建该角色时，必须附加一个向 Amplify 授予代入该角色的权限的自定义信任策略。如果未正确定义信任关系，尝试添加该角色时将会出现错误。以下自定义信任策略将向 Amplify 授予代入该角色的权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Principal": {
                "Service": [
```

```
                    "amplify.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

您可以使用 Amplify 控制台将中的 IAM 角色 AWS 账户 与现有 SSR 应用程序关联起来 AWS SDKs，或者。 AWS CLI您附加的角色会自动与 Amplify SSR 计算服务关联，从而向其授予您指定的访问其他资源的权限。 AWS 随着应用程序的需求逐渐变化，您只需修改附加的 IAM 角色，而无需重新部署应用程序。这不仅提供了灵活性，同时也减少了应用程序停机时间。

> **Important**
>
> 您负责根据自己的安全性和合规性目标配置应用程序。这包括管理您的 SSR 计算角色，为其配置支持您的使用案例所需的最低权限集。有关更多信息，请参阅 管理 IAM SSR 计算角色的安全性。

在 IAM 控制台中创建 SSR 计算角色

IAM SSR 计算角色必须首先已经存在于您的 AWS 账户中，然后才能将其附加到 Amplify 应用程序。本节将介绍如何创建 IAM 策略并将其附加到一个角色，以便 Amplify 代入该角色来访问特定的 AWS 资源。

我们建议您在创建 IAM 角色时遵循授予最低权限权限 AWS 的最佳实践。IAM SSR 计算角色只能从 SSR 计算函数调用，因此应仅向其授予运行代码所需的权限。

您可以使用 AWS 管理控制台 AWS CLI、或 SDKs 在 IAM 中创建策略。有关更多信息，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

以下说明演示了如何使用 IAM 控制台创建向 Amplify 计算服务授予若干权限的 IAM 策略。

使用 IAM 控制台 JSON 策略编辑器创建策略

1. 登录 AWS 管理控制台 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
2. 在左侧的导航窗格中，选择策略。
3. 选择创建策略。

4. 在策略编辑器部分，选择 JSON 选项。
5. 键入或粘贴一个 JSON 策略文档。
6. 向策略添加完权限后，选择下一步。
7. 在查看和创建页面上，为创建的策略键入策略名称和描述（可选）。查看此策略中定义的权限以查看策略授予的权限。
8. 选择创建策略可保存新策略。

创建策略之后，按照以下说明将该策略附加到 IAM 角色。

创建向特定资源授予 Amplify 权限的角色 AWS

1. 登录 AWS 管理控制台 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
2. 在控制台的导航窗格中，选择 Roles，然后选择 Create role。
3. 选择 Custom trust policy（自定义信任策略）角色类型。
4. 在自定义信任策略部分中，输入该角色的自定义信任策略。必须具有角色信任策略，其中定义了您信任可以代入该角色的主体。

   复制并粘贴以下信任策略，向 Amplify 服务授予代入该角色的权限。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "amplify.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

5. 解决策略验证过程中生成的任何安全警告、错误或常规警告，然后选择下一步。

6. 在添加权限页面上，搜索您在上一过程中所创建策略的名称，然后将选中该策略。然后选择下一步。
7. 在角色名称中，输入一个角色名称。角色名称在您的内部必须是唯一的 AWS 账户。它们不按大小写区分。例如，您无法同时创建名为 **PRODROLE** 和 **prodrole** 的角色。由于其他 AWS 资源可能会引用该角色，因此您无法在角色创建后对其名称进行编辑。
8. （可选）对于 Description（描述），输入新角色的描述。
9. （可选）在步骤 1：选择受信任的实体或步骤 2：添加权限部分中选择编辑，以编辑角色的自定义策略和权限。
10. 检查角色，然后选择创建角色。

将 IAM SSR 计算角色添加到 Amplify 应用程序

在中创建 IAM 角色后 AWS 账户，您可以在 Amplify 控制台中将其与 Amplify 控制台中的应用程序关联。

在 Amplify 控制台中将 IAM SSR 计算角色添加到应用程序

1. 登录 AWS 管理控制台 并打开 Amplify 控制台，网址为。https://console.aws.amazon.com/amplify/
2. 在所有应用程序页面上，选择要向其添加计算角色的应用程序的名称。
3. 在导航窗格中，依次选择应用程序设置、IAM 角色。
4. 在计算角色部分中，选择编辑。
5. 在默认角色列表中，搜索要附加的角色的名称，然后选中该角色。在此例中，您可以选择在上一过程中所创建角色的名称。默认情况下，您选中的角色将关联到应用程序的所有分支。

   如果未正确定义角色的信任关系，将会出现错误并且您将无法添加该角色。
6. （可选）如果应用程序位于公有存储库中，并且使用自动创建分支功能或启用了拉取请求 Web 预览功能，则我们不建议使用应用程序级别的角色。而应仅将计算角色附加到需要访问特定资源的分支。要覆盖默认的应用程序级别行为并将角色附加到特定的分支，请执行以下操作：

   a. 对于分支，选择要使用的分支的名称。

   b. 对于计算角色，选择要与该分支关联的角色的名称。
7. 选择保存。

管理 IAM SSR 计算角色的安全性

安全是双方 AWS 的共同责任。您负责根据自己的安全性和合规性目标配置应用程序。这包括管理您的 SSR 计算角色，为其配置支持您的使用案例所需的最低权限集。您指定的 SSR 计算角色凭证将在 SSR 函数的运行时中立即可用。如果 SSR 代码故意、由于错误或通过允许远程代码执行（RCE）暴露了这些凭证，则未经授权的用户可能会获得该 SSR 角色及其权限的访问权限。

当公有存储库中的应用程序使用 SSR 计算角色和自动分支创建功能，或为所有拉取请求启用 Web 预览时，您需要仔细管理有权访问该角色的分支。我们建议不要使用应用程序级别的角色，而应在分支级别附加计算角色。这可让您仅向需要访问特定资源的分支授予访问权限。

如果角色的凭证泄露，请执行以下操作移除对该角色的凭证的所有访问权限。

1. 撤销所有会话

   有关立即撤销对该角色的凭证的所有权限的说明，请参阅撤销 IAM 角色临时安全凭证。

2. 从 Amplify 控制台移除该角色

   此操作将立即生效。您不需要重新部署应用程序。

在 Amplify 控制台中删除计算角色

1. 登录 AWS 管理控制台 并打开 Amplify 控制台，网址为。https://console.aws.amazon.com/amplify/
2. 在所有应用程序页面上，选择要从中移除计算角色的应用程序的名称。
3. 在导航窗格中，依次选择应用程序设置、IAM 角色。
4. 在计算角色部分中，选择编辑。
5. 要删除默认角色，请选择角色名称右侧的 X。
6. 选择保存。

添加具有访问 CloudWatch 日志权限的服务角色

Amplify 将有关您的 SSR CloudWatch 运行时的信息发送到您的 Amazon Logs。 AWS 账户当您部署 SSR 应用程序时，Amplify 需要一个 IAM 服务角色，Amplify 在代表您调用其他服务时代入该角色。您可以允许 Amplify Hosting 计算自动为您创建服务角色，也可以指定您已创建的角色。

如果您选择允许 Amplify 为您创建 IAM 角色，则该角色将已经拥有创建 CloudWatch 日志的权限。如果您创建自己的 IAM 角色，则需要在策略中添加以下权限以允许 Amplify 访问亚马逊 CloudWatch 日志。

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

适用于 Git 存储库的统一 Webhook

在向 Git 存储库发送新的提交后，Amplify Hosting 使用 Webhook 自动启动构建。统一 Webhook 功能改进了 Amplify 与 Git 提供商的集成，使您能够将更多 Amplify 应用程序连接到单个存储库。使用统一 Webhook 后，对于存储库中所有关联的应用程序，Amplify 现在将在每个区域使用单个 Webhook。例如，假设存储库同时连接到美国东部（弗吉尼亚州北部）和美国西部（俄勒冈州）区域中的应用程序，则您将有两个统一 Webhook。

在此功能发布之前，Amplify 为关联到一个存储库的每个应用程序创建一个新的 Webhook。如果您在单个存储库中有多个应用程序，则可能会达到各 Git 提供商强制实施的 Webhook 限制，不再能够添加其他应用程序。这给使用单体存储库模式的团队尤其带来挑战，因为在单体存储库模式中，单个存储库中存在多个项目。

统一 Webook 具有以下优势：

- 克服 Git 提供商的 Webhook 限制：您可以根据需要将任意数量的 Amplify 应用程序连接到单个存储库。
- 增强对单体存储库模式的支持：使用单体存储库模式时，由于多个项目共享一个存储库，您可以获得更好的灵活性和更高的效率。
- 简化管理：使用单个存储库 Webhook 来管理多个 Amplify 应用程序，可降低复杂性，减少可能的故障点。
- 改进工作流集成：您可以将 Git 提供商分配的 Webhook 用于开发过程中的其他关键工作流。

统一 Webhook 入门

创建新应用程序

从 Git 存储库将新应用程序部署到 Amplify Hosting 时，系统会自动为您的存储库实施统一 Webhook 功能。有关创建新应用程序的说明，请参阅开始将应用程序部署到 Amplify Hosting。

更新现有应用程序

对于现有的 Amplify 应用程序，您必须将 Git 存储库重新连接到该应用程序，才能将现有 Webhook 替换为统一 Webhook。如果您已经达到了 Git 提供商允许的 Webhook 数量上限，则可能无法成功迁移到统一 Webhook。在这种情况下，请手动移除至少一个现有 Webhook，然后再重新连接。

您可以将一个存储库中的多个应用程序部署到不同的 AWS 区域。由于 Amplify 操作属于区域性操作，因此统一 Webhook 将仅对重新连接 Amplify 应用程序的区域中的 Webhook 有效。由于这一原因，您可能会在存储库中同时看到基于应用程序 ID 的 Webhook 和基于区域的统一 Webhook。

按照以下说明将现有 Amplify 应用程序迁移到统一 Webhook。

将现有 Amplify 应用程序迁移到统一 Webhook

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要迁移到统一 Webhook 的应用程序。
3. 在导航窗格中，依次选择应用程序设置和分支设置。
4. 在分支设置页面上，选择重新连接存储库。
5. 要验证是否成功迁移到统一 Webhook，请导航到 Git 存储库中的 Webhook 设置。您应会看到格式为 `https://amplify-webhooks.Region.amazonaws.com/git-provider` 的单个 Webhook URL。

Amplify 的安全保护

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用的合规计划 AWS Amplify，请参阅按合规计划划分的范围内的AWSAWS 服务按合规计划。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

该文档将帮助您了解如何在使用 Amplify 时应用责任共担模型。以下主题说明如何配置 Amplify 以实现您的安全性和合规性目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 Amplify 资源。

主题

适用于 Amplify 的身份和访问管理

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以通过身份验证（登录）和授权（具有权限）使用 Amplify 资源。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

受众

您的使用方式 AWS Identity and Access Management (IAM) 因您的角色而异：

- 服务用户：如果您无法访问功能，请从管理员处请求权限（请参阅Amplify 身份和访问问题排查）
- 服务管理员：确定用户访问权限并提交权限请求（请参阅Amplify 如何与 IAM 协同工作）
- IAM 管理员：编写用于管理访问权限的策略（请参阅适用于 Amplify 的基于身份的策略示例）

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份进行身份验证 AWS 账户根用户，或者通过担任 IAM 角色进行身份验证。

您可以使用来自身份源的证书 AWS IAM Identity Center（例如（IAM Identity Center）、单点登录身份验证或 Google/Facebook 证书，以联合身份登录。有关登录的更多信息，请参阅《AWS 登录 用户指南》中的如何登录您的 AWS 账户。

对于编程访问，AWS 提供 SDK 和 CLI 来对请求进行加密签名。有关更多信息，请参阅《IAM 用户指南》中的适用于 API 请求的AWS 签名版本 4。

AWS 账户 root 用户

创建时 AWS 账户，首先会有一个名为 AWS 账户 root 用户的登录身份，该身份可以完全访问所有资源 AWS 服务 和资源。我们强烈建议不要使用根用户进行日常任务。有关需要根用户凭证的任务，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户使用与身份提供商的联合身份验证才能 AWS 服务 使用临时证书进行访问。

联合身份是指来自您的企业目录、Web 身份提供商的用户 Directory Service ，或者 AWS 服务 使用来自身份源的凭据进行访问的用户。联合身份代入可提供临时凭证的角色。

要集中管理访问权限，建议使用。 AWS IAM Identity Center有关更多信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？。

IAM 用户和群组

IAM 用户是对某个人员或应用程序具有特定权限的一个身份。建议使用临时凭证，而非具有长期凭证的 IAM 用户。有关更多信息，请参阅 IAM 用户指南中的要求人类用户使用身份提供商的联合身份验证才能 AWS 使用临时证书进行访问。

IAM 组指定一组 IAM 用户，便于更轻松地对大量用户进行权限管理。有关更多信息，请参阅《IAM 用户指南》中的 IAM 用户使用案例。

IAM 角色

IAM 角色是具有特定权限的身份，可提供临时凭证。您可以通过从用户切换到 IAM 角色（控制台）或调用 AWS CLI 或 AWS API 操作来代入角色。有关更多信息，请参阅《IAM 用户指南》中的担任角色的方法。

IAM 角色对于联合用户访问、临时 IAM 用户权限、跨账户访问、跨服务访问以及在 Amazon EC2 上运行的应用程序非常有用。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略定义了与身份或资源关联时的权限。 AWS 在委托人提出请求时评估这些政策。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概述。

管理员使用策略，通过定义哪个主体可以在什么条件下对哪些资源执行哪些操作来指定谁有权访问什么。

默认情况下，用户和角色没有权限。IAM 管理员创建 IAM 策略并将其添加到角色中，然后用户可以担任这些角色。IAM 策略定义权限，与执行操作所用的方法无关。

基于身份的策略

基于身份的策略是您附加到身份（用户、组或角色）的 JSON 权限策略文档。这些策略控制身份可以执行什么操作、对哪些资源执行以及在什么条件下执行。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

基于身份的策略可以是内联策略（直接嵌入到单个身份中）或托管策略（附加到多个身份的独立策略）。要了解如何在托管策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。您必须在基于资源的策略中指定主体。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

其他策略类型

AWS 支持其他策略类型，这些策略类型可以设置更常见的策略类型授予的最大权限：

- 权限边界 – 设置基于身份的策略可以授予 IAM 实体的最大权限。有关更多信息，请参阅《 IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCPs)-在中指定组织或组织单位的最大权限 AWS Organizations。有关更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。
- 资源控制策略 (RCPs)-设置账户中资源的最大可用权限。有关更多信息，请参阅《AWS Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略 – 在为角色或联合用户创建临时会话时，作为参数传递的高级策略。有关更多信息，请参阅《IAM 用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

Amplify 如何与 IAM 协同工作

在使用 IAM 管理 Amplify 的访问权限之前，请了解哪些 IAM 功能可用于 Amplify。

可与 Amplify 一起使用的 IAM 功能

IAM 功能	Amplify 支持
基于身份的策略	是
基于资源的策略	否
策略操作	是
策略资源	是
策略条件键	是
ACLs	否
ABAC（策略中的标签）	部分
临时凭证	是
转发访问会话（FAS）	是
服务角色	是
服务关联角色	否

要全面了解 Amplify 和其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中与 IAM 配合使用的AWS 服务。

适用于 Amplify 的基于身份的策略

支持基于身份的策略：是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

适用于 Amplify 的基于身份的策略示例

要查看 Amplify 基于身份的策略的示例，请参阅 适用于 Amplify 的基于身份的策略示例。

Amplify 中的基于资源的策略

支持基于资源的策略：否

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户访问，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

Amplify 的策略行动

支持策略操作：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。在策略中包含操作以授予执行关联操作的权限。

有关 Amplify 操作的列表，请参阅服务授权参考中 AWS Amplify定义的操作。

Amplify 中的策略操作在操作前使用以下前缀：

```
amplify
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "amplify:action1",
      "amplify:action2"
         ]
```

要查看 Amplify 基于身份的策略的示例，请参阅 适用于 Amplify 的基于身份的策略示例。

Amplify 的策略资源

支持策略资源：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于不支持资源级权限的操作，请使用通配符 (*) 指示语句应用于所有资源。

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

有关 Amplify 资源类型及其列表 ARNs，请参阅《服务授权参考》 AWS Amplify中定义的资源类型。要了解可以在哪些操作中指定每个资源的 ARN，请参阅 AWS Amplify定义的操作。

要查看 Amplify 基于身份的策略的示例，请参阅 适用于 Amplify 的基于身份的策略示例。

Amplify 的策略条件密钥

支持特定于服务的策略条件键：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

`Condition` 元素根据定义的条件指定语句何时执行。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

有关 Amplify 条件密钥的列表，请参阅服务授权参考中的 AWS Amplify的条件密钥。要了解可以使用条件键的操作和资源，请参阅由定义的操作 AWS Amplify。

要查看 Amplify 基于身份的策略的示例，请参阅 适用于 Amplify 的基于身份的策略示例。

Amplify 中的访问控制列表 (ACLs)

支持 ACLs：否

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amplify 基于属性的访问权限控制 (ABAC)

支持 ABAC（策略中的标签）：部分支持

基于属性的访问权限控制（ABAC）是一种授权策略，该策略基于称为标签的属性来定义权限。您可以将标签附加到 IAM 实体和 AWS 资源，然后设计 ABAC 策略以允许在委托人的标签与资源上的标签匹配时进行操作。

要基于标签控制访问，您需要使用 `aws:ResourceTag/key-name`aws:RequestTag/*key-name* 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息，请参阅《IAM 用户指南》中的使用 ABAC 授权定义权限。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

将临时凭证用于 Amplify

支持临时凭证：是

临时证书提供对 AWS 资源的短期访问权限，并且是在您使用联合身份或切换角色时自动创建的。AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的临时安全凭证和使用 IAM 的。AWS 服务

Amplify 的转发访问会话

支持转发访问会话（FAS）：是

转发访问会话 (FAS) 使用调用主体的权限 AWS 服务，再加上 AWS 服务 向下游服务发出请求的请求。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

Amplify 的服务角色

支持服务角色：是

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。

> **Warning**
>
> 更改服务角色的权限可能会中断 Amplify 功能。仅当 Amplify 提供相关指导时才编辑服务角色。

Amplify 的服务相关角色

支持服务相关角色：否

服务相关角色是一种与服务相关联的 AWS 服务服务角色。服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务关联角色的权限。

有关创建或管理服务相关角色的详细信息，请参阅 IAM 用户指南中的能够与 IAM 搭配使用的AWS 服务。在表中查找服务相关角色列中包含 Yes 的表。选择是链接以查看该服务的服务相关角色文档。

适用于 Amplify 的基于身份的策略示例

默认情况下，用户和角色没有权限创建或修改 Amplify 资源。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略（控制台）。

有关 Amplify 定义的操作和资源类型（包括每种资源类型的格式）的详细信息，请参阅《服务授权参考》 AWS Amplify中的操作、资源和条件键。 ARNs

主题

- 策略最佳实践
- 使用 Amplify 控制台
- 允许用户查看他们自己的权限

策略最佳实践

基于身份的策略决定是否有人可以在你的账户中创建、访问或删除 Amplify 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的AWS 托管策略或工作职能的AWS 托管策略。
- 应用最低权限：在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限：您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性：IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言（JSON）和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM Access Analyzer 验证策略。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的使用 MFA 保护 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 Amplify 控制台

要访问 AWS Amplify 控制台，您必须拥有一组最低权限。这些权限必须允许您列出和查看有关您的 Amplify 资源的详细信息。 AWS 账户如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

随着 Amplify Studio 的发布，删除应用程序或后端需要同时具备 `amplify` 和 `amplifybackend` 权限。如果 IAM policy 仅提供 `amplify` 权限，则用户在尝试删除应用程序时会出现权限错误。如果您是编写策略的管理员，请确定向需要执行删除操作的用户授予的正确权限。

为确保用户和角色仍然可以使用 Amplify 控制台，还需要将 Amplify `ConsoleAccess` 或`ReadOnly` AWS 托管策略附加到实体。有关更多信息，请参阅《IAM 用户指南》中的为用户添加权限。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
    "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 的托管策略 AWS Amplify

AWS 托管策略是由创建和管理的独立策略 AWS。 AWS 托管策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住， AWS 托管策略可能不会为您的特定用例授予最低权限权限，因为它们可供所有 AWS 客户使用。我们建议通过定义特定于使用案例的客户管理型策略来进一步减少权限。

您无法更改 AWS 托管策略中定义的权限。如果 AWS 更新 AWS 托管策略中定义的权限，则更新会影响该策略所关联的所有委托人身份（用户、组和角色）。 AWS 最有可能在启动新的 API 或现有服务可以使用新 AWS 服务 的 API 操作时更新 AWS 托管策略。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

AWS 托管策略： AdministratorAccess-Amplify

您可以将 `AdministratorAccess-Amplify` 策略附加到 IAM 身份。Amplify 还会将此策略附加到服务角色，允许 Amplify 代表您执行操作。

在 Amplify 控制台中部署后端时，必须创建一个`Amplify-Backend Deployment`服务角色，Amplify 使用该角色来创建和管理资源。 AWS IAM 将 `AdministratorAccess-Amplify` 托管策略附加到 `Amplify-Backend Deployment` 服务角色。

此策略授予账户管理权限，同时明确允许直接访问 Amplify 应用创建和管理后端所需的资源。

权限详细信息

此策略提供对多种 AWS 服务的访问权限，包括 IAM 操作。这些操作允许使用 AWS Identity and Access Management 此策略的身份创建具有任何权限的其他身份。这允许升级权限，此策略应视为与 `AdministratorAccess` 策略一样强大。

该策略向所有资源授予 `iam:PassRole` 操作权限。这是支持 Amazon Cognito 用户群体配置所必需的。

要查看此策略的权限，请参阅《AWS 托管策略参考》中的 AdministratorAccess-Amplify。

AWS 托管策略： AmplifyBackendDeployFullAccess

您可以将 `AmplifyBackendDeployFullAccess` 策略附加到 IAM 身份。

此政策授予 Amplify 使用部署 Amplify 后端资源的完全访问权限。 AWS Cloud Development Kit (AWS CDK)权限将延迟到具有必要`AdministratorAccess`策略权限的 AWS CDK 角色。

权限详细信息

此策略包括执行以下操作的权限。

- Amplify – 检索有关已部署应用程序的元数据。
- CloudFormation – 创建、更新和删除 Amplify 托管的堆栈。
- SSM – 创建、更新和删除 Amplify 托管的 SSM Parameter Store String 和 SecureString 参数。
- AWS AppSync— 更新和检索 AWS AppSync 架构、解析器和函数资源。目的是支持 Gen 2 沙盒热交换功能。
- Lambda – 更新和检索 Amplify 托管函数的配置。目的是支持 Gen 2 沙盒热交换功能。

  检索 Lambda 函数的标签。其目的在于支持客户定义的 Lambda 函数。
- Amazon S3 – 检索 Amplify 部署资产。
- AWS Security Token Service— 允许 AWS Cloud Development Kit (AWS CDK) CLI 担任部署角色。
- Amazon RDS – 读取数据库实例、集群和代理的元数据。
- Amazon EC2 – 读取子网的可用区信息。
- CloudWatch Logs – 检索客户的 Lambda 函数的日志。目的是允许 Amplify 云开发沙盒环境将 Lambda 函数的日志流式传输到客户的终端。

要查看此策略的权限，请参阅《AWS 托管式策略参考》中的 AmplifyBackendDeployFullAccess。

Amplify 对托管策略的 AWS 更新

查看有关 Amplify AWS 托管政策自该服务开始跟踪这些变更以来这些更新的详细信息。要获得有关此页面更改的自动提示，请订阅 的文档历史记录 AWS Amplify 页面上的 RSS 源。

更改	描述	日期
AmplifyBackendDeployFullAccess：对现有策略的更新	添加对 logs:FilterLogEvents 资源的读取权限，以允许 Amplify 从创建自定义日志组的函数流式传输日志。这是对流式传输	2024 年 11 月 14 日

更改	描述	日期
	Lambda 函数日志的现有功能的扩展。	
AmplifyBackendDeployFullAccess：对现有策略的更新	添加对 `lambda:ListTags` 和 `logs:FilterLogEvents` 资源的读取权限，以支持客户定义的 Lambda 函数。这些权限允许 Amplify 云开发沙盒环境将 Lambda 函数的日志流式传输到客户的终端。	2024 年 7 月 18 日
AmplifyBackendDeployFullAccess：对现有策略的更新	添加对 `arn:aws:ssm:*:*:parameter/cdk-bootstrap/*` 资源的读取权限，允许 Amplify 检测客户账户中的 CDK 引导版本。	2024 年 5 月 31 日

更改	描述	日期
AmplifyBackendDeployFullAccess：对现有策略的更新	添加新的 `AmplifyDiscoverRDSVpcConfig` 策略声明，其中包含 Amazon RDS 和 Amazon EC2 只读权限，其范围由资源和账户条件决定。这些权限支持 Amplify Gen 2 `npx amplify generate schema-from-database` 命令，该命令允许客户基于现有 SQL 数据库生成 Typescript 数据架构。 添加 `rds:DescribeDBProxies` 、`rds:DescribeDBInstances` 、`rds:DescribeDBClusters` 、`rds:DescribeDBSubnetGroups` 和 `ec2:DescribeSubnets` 权限。该`npx amplify generate schema-from-database` 命令需要这些权限来检查指定的数据库主机是否托管在 Amazon RDS 中，并自动生成预置设置由 SQL 数据库支持的 AWS AppSync API 所需的其他资源所需的 Amazon VPC 配置。	2024 年 4 月 17 日

更改	描述	日期
AmplifyBackendDeployFullAccess：对现有策略的更新	添加 `cloudformation:DeleteStack` 策略操作，以支持在调用 `DeleteBranch` API 时删除堆栈。 添加 `lambda:GetFunction` 策略操作，以支持热交换功能。 添加 `lambda:UpdateFunctionConfiguration` 策略操作，以支持 Lambda 函数。	2024 年 4 月 5 日
AdministratorAccess-Amplify — 更新现有政策	添加`cloudformation:TagResource` 和`cloudformation:UnTagResource` 权限以支持调用 CloudFormation APIs。	2024 年 4 月 4 日
AmplifyBackendDeployFullAccess：对现有策略的更新	添加支持 AWS Cloud Development Kit (AWS CDK) 热交换的`lambda:InvokeFunction` 策略操作。直接调用 Lambda 函数来执行 Amazon S3 资产热交换。AWS CDK 添加 `lambda:UpdateFunctionCode` 策略操作，以支持热交换功能。	2024 年 1 月 2 日
AmplifyBackendDeployFullAccess：对现有策略的更新	添加策略操作以支持该 `UpdateApiKey` 操作。退出并重新启动沙盒后，要想在不删除资源的情况下成功部署应用程序，就必须执行此操作。	2023 年 11 月 17 日

更改	描述	日期
AmplifyBackendDeployFullAccess：对现有策略的更新	添加支持 Amplify 应用程序部署的 `amplify:GetBackendEnvironment` 权限。	2023 年 11 月 6 日
AmplifyBackendDeployFullAccess：新策略	Amplify 添加了一项新策略，该策略拥有部署 Amplify 后端资源所需的最低权限。	2023 年 10 月 8 日
AdministratorAccess-Amplify — 更新现有政策	添加 Amplify 命令行界面 (CLI) 所需的 `ecr:DescribeRepositories` 权限。	2023 年 6 月 1 日

更改	描述	日期
AdministratorAccess-Amplify — 更新现有政策	添加一项策略操作以支持从 AWS AppSync 资源中移除标签。 添加一项策略操作以支持 Amazon Polly 资源。 添加策略操作以支持更新 OpenSearch 域配置。 添加一项策略操作以支持从 AWS Identity and Access Management 角色中移除标签。 添加一项策略操作以支持从 Amazon DynamoDB 资源中移除标签。 向 `CLISDKCalls` 语句块中添加 `cloudfront:GetCloudFrontOriginAccessIdentity` 和 `cloudfront:GetCloudFrontOriginAccessIdentityConfig` 权限以支持 Amplify 发布和托管工作流程。 向 `CLIManageviaCFNPolicy` 语句块添加 `s3:PutBucketPublicAccessBlock` 权限，以允许 AWS CLI 支持 Amazon S3 安全最佳实践，即在内部存储桶上启用 Amazon S3 屏蔽公共访问权限功能。	2023 年 2 月 24 日

更改	描述	日期
	向`CLISDKCalls` 语句块添加`cloudformation:DescribeStacks` 权限以支持在 Amplify 后端处理器中重试时检索客户的 CloudFormation 堆栈，从而避免在堆栈更新时重复执行。 向 `CLICloudformationPolicy` 语句块添加 `cloudformation:ListStacks` 权限。需要此权限才能完全支持该 CloudFormation DescribeStacks 操作。	
AdministratorAccess-Amplify — 更新现有政策	添加政策操作，允许 Amplify 服务器端渲染功能将应用程序指标推送到客户的服务器端渲染 CloudWatch 中。 AWS 账户	2022 年 8 月 30 日
AdministratorAccess-Amplify — 更新现有政策	添加策略操作以屏蔽公共访问 Amplify 部署 Amazon S3 存储桶。	2022 年 4 月 27 日

更改	描述	日期
AdministratorAccess-Amplify — 更新现有政策	添加一项操作以允许客户删除其服务器端渲染 (SSR) 应用程序。这也允许成功删除相应的CloudFront分发。 添加一项操作以允许客户使用 Amplify CLI 指定不同的 Lambda 函数，从而处理来自现有事件源的事件。通过这些更改， AWS Lambda 将能够执行UpdateEventSourceMapping操作。	2022 年 4 月 17 日
AdministratorAccess-Amplify — 更新现有政策	添加一项策略操作以在所有资源上启用 Amplify UI Builder 操作。	2021 年 12 月 2 日
AdministratorAccess-Amplify — 更新现有政策	添加一项策略操作以支持使用社交身份提供商的 Amazon Cognito 身份验证功能。 添加一项策略操作以支持 Lambda 层。 添加一项策略操作以支持 Amplify 存储类别。	2021 年 11 月 8 日

更改	描述	日期
AdministratorAccess-Amplify — 更新现有政策	添加 Amazon Lex 操作以支持 Amplify 交互类别。 添加 Amazon Rekognition 操作以支持 Amplify 预测类别。 添加 Amazon Cognito 操作以支持 Amazon Cognito 用户群体上的 MFA 配置。 添加 CloudFormation 操作以获得支持 CloudFormation StackSets。 添加 Amazon Location Service 操作以支持 Amplify 地理位置类别。 添加 Lambda 操作以支持 Amplify 中的 Lambda 层。 添加 CloudWatch 日志操作以支持 CloudWatch 事件。 添加 Amazon S3 操作以支持 Amplify 存储类别。 添加策略操作以支持服务器端渲染 (SSR) 应用程序。	2021 年 9 月 27 日

更改	描述	日期
AdministratorAccess-Amplify — 更新现有政策	将所有 Amplify 操作合并为一项 `amplify:*` 操作。 添加 Amazon S3 操作以支持加密客户的 Amazon S3 存储桶。 添加 IAM 权限边界操作以支持启用了权限边界的 Amplify 应用程序。 添加 Amazon SNS 操作以支持查看起始电话号码，以及查看、创建、验证和删除目标电话号码。 Amplify Studio：添加亚马逊 Cognito AWS Lambda、IAM CloudFormation 和策略操作，以便在 Amplify 控制台和 Amplify Studio 中管理后端。 添加 AWS Systems Manager (SSM) 策略声明以管理 Amplify 环境密钥。 添加一个 CloudFormation `ListResources` 操作以支持 Amplify 应用程序的 Lambda 图层。	2021 年 7 月 28 日
Amplify 已开启跟踪更改	Amplify 开始跟踪其 AWS 托管策略的变更。	2021 年 7 月 28 日

Amplify 身份和访问问题排查

使用以下信息可帮助您诊断和修复在使用 Amplify 和 IAM 时可能遇到的常见问题。

主题

我无权在 Amplify 中执行操作

如果您收到错误提示，指明您无权执行某个操作，则必须更新策略以允许执行该操作。

当 mateojackson IAM 用户尝试使用控制台查看有关虚构 *my-example-widget* 资源的详细信息，但不拥有虚构 amplify:*GetWidget* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 amplify:GetWidget on resource: my-example-widget
```

在此情况下，必须更新 mateojackson 用户的策略，以允许使用 amplify:*GetWidget* 操作访问 *my-example-widget* 资源。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

随着 Amplify Studio 的发布，删除应用程序或后端需要同时具备 amplify 和 amplifybackend 权限。如果管理员编写了仅提供 amplify 权限的 IAM policy，则在尝试删除应用程序时会出现权限错误。

当 mateojackson IAM 用户试图使用控制台删除虚构 *example-amplify-app* 资源但没有 amplifybackend:*RemoveAllBackends* 权限时，会出现以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 amplifybackend;:RemoveAllBackends on resource: example-amplify-app
```

在这种情况下，Mateo 请求他的管理员更新其策略，以允许他使用 amplifybackend:*RemoveAllBackends* 操作访问 *example-amplify-app* 资源。

我无权执行 iam：PassRole

如果您收到一个错误，表明您无权执行 `iam:PassRole` 操作，则必须更新策略以允许您将角色传递给 Amplify。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

在名为 `marymajor` 的 IAM 用户尝试使用控制台在 Amplify 中执行操作时，将会出现以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

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

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许 AWS 账户之外的人访问我的 Amplify 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解 Amplify 是否支持这些功能，请参阅 Amplify 如何与 IAM 协同工作。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

Amplify 的数据保护

AWS Amplify 符合分担责任模式 AWS 分担责任模式，其中包括数据保护的法规和指导方针。 AWS 负责保护运行所有 AWS 服务的全球基础架构。 AWS 保持对托管在此基础架构上的数据的控制，包括用于处理客户内容和个人数据的安全配置控制。 AWS 客户和 APN 合作伙伴，无论是作为数据控制者还是数据处理者，都应对他们在 AWS 云端存储的任何个人数据负责。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这仅向每个用户授予履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 用于 SSL/TLS 与 AWS 资源通信。
- 使用设置 API 和用户活动日志 AWS CloudTrail。
- 使用 AWS 加密解决方案以及 AWS 服务中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的个人数据。

我们强烈建议您切勿将敏感的可识别信息（例如您客户的账号）放入自由格式字段（例如名称字段）。这包括您使用控制台、API 或使用 Amplify 或其他 AWS 服务时。 AWS CLI AWS SDKs您输入到 Amplify 或其他服务的任何数据都可能被选取以包含在诊断日志中。当您向外部服务器提供网址时，请勿在网址中包含凭证信息来验证您对该服务器的请求。

有关数据保护的更多信息，请参阅AWS 安全性博客 上的AWS 责任共担模式和 GDPR 博客文章。

静态加密

静态加密是指通过对存储数据进行加密来保护您的数据免受未经授权的访问。默认情况下，Amplify 使用由管理的 A AWS KMS keys mazon S3 对应用程序的构建项目进行加密。 AWS Key Management Service

Amplify CloudFront 使用亚马逊为您的客户提供您的应用程序。 CloudFront 用于边缘接入点 (POPs)，使用 SSDs 加密的 EBS 卷用于区域边缘缓存 (RECs)。Functi CloudFront ons 中的功能代码和配置始终以加密格式存储 SSDs 在边缘位置 POPs和使用的其他存储位置 CloudFront。

传输中加密

传输中加密是指在通信终端节点之间移动数据时，保护您的数据免遭拦截。默认情况下，Amplify Hosting 会对传输中数据提供加密。使用通过签名版本 4 签名流程签名的 SSL 连接来保护客户和 Amplify 之间以及 Amplify 与其下游依赖项之间的所有通信。所有 Amplify Hosting 端点都使用由 AWS 私有证书颁发机构管理的 SHA-256 证书。有关更多信息，请参阅签名版本 4 签名流程和什么是 AWS 私有证书颁发机构。

加密密钥管理

AWS Key Management Service (KMS) 是一项托管服务 AWS KMS keys，用于创建和控制用于加密客户数据的加密密钥。 AWS Amplify 代表客户生成和管理用于加密数据的加密密钥。没有可供您管理的加密密钥。

合规性验证 AWS Amplify

AWS Amplify 作为多个合规计划的一部分，第三方审计师对安全性和 AWS 合规性进行评估。其中包括 SOC、PCI、ISO、HIPAA、MTSC、C5、K-ISMS、ENS High、OSPAR、HITRUST CSF 和 FINMA。

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。有关您在使用时的合规责任的更多信息 AWS 服务，请参阅AWS 安全文档。

中的基础设施安全 AWS Amplify

作为一项托管服务 AWS Amplify ，受 AWS 全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅AWS 云安全。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 AWS 已发布的 API 调用通过网络访问 Amplify。客户端必须支持以下内容：

- 传输层安全性协议（TLS）。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密（PFS）的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

在 Amplify 中记录和监控安全事件

监控是维护 Amplify 和其他 AWS 解决方案的可靠性、可用性和性能的重要组成部分。 AWS 提供了以下监控工具，用于监视 Amplify、报告问题并在适当时自动采取措施：

- Amazon 会实时 CloudWatch监控您的 AWS 资源和您运行的应用程序 AWS。您可以收集和跟踪指标，创建自定义的控制面板，以及设置在某个指标达到指定阈值时通知您或采取措施的警报。例如，您可以 CloudWatch 跟踪亚马逊弹性计算云 (Amazon EC2) 实例的 CPU 使用率或其他指标，并在需要时自动启动新实例。有关在 Amplify 中使用 CloudWatch 指标和警报的更多信息，请参阅。监控 Amplify 应用程序
- Amazon CloudWatch Logs 允许您监控、存储和访问来自 Amazon EC2 实例和其他来源的日志文件。 AWS CloudTrail CloudWatch 日志可以监视日志文件中的信息，并在达到特定阈值时通知您。您还可以在高持久性存储中检索您的日志数据。有关更多信息，请参阅 Amazon CloudWatch 日志用户指南。
- AWS CloudTrail捕获由您的账户或代表您的 AWS 账户进行的 API 调用和相关事件，并将日志文件传输到您指定的亚马逊简单存储服务 (Amazon S3) Service 存储桶。您可以识别哪些用户和帐户拨打了电话 AWS、发出呼叫的源 IP 地址以及呼叫发生的时间。有关更多信息，请参阅 使用记录 Amplify API 调用 AWS CloudTrail。
- Amazon EventBridge 是一项无服务器事件总线服务，可以轻松地将您的应用程序与来自各种来源的数据连接起来。 EventBridge 提供来自您自己的应用程序、 Software-as-a-Service (SaaS) 应用程序和 AWS 服务的实时数据流，并将这些数据路由到目标，例如 AWS Lambda。这使您能够监控服务中发生的事件，并构建事件驱动的架构。有关更多信息，请参阅 Amazon EventBridge 用户指南。

防止跨服务混淆代理

混淆代理问题是一个安全性问题，即不具有某操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在中 AWS，跨服务模仿可能会导致混乱的副手问题。一个服务（呼叫服务）调用另一项服务（所谓的服务）时，可能会发生跨服务模拟。可以操纵调用服务，使用其权限以在其他情况下该服务不应有访问权限的方式对另一个客户的资源进行操作。为防止这种情况， AWS 提供可帮助您保护所有服务的数据的工具，而这些服务中的服务主体有权限访问账户中的资源。

我们建议在资源策略中使用`aws:SourceArn`和`aws:SourceAccount`全局条件上下文密钥来限制为资源 AWS Amplify 提供其他服务的权限。如果使用两个全局条件上下文键，在同一策略语句中使用时，`aws:SourceAccount` 值和 `aws:SourceArn` 值中的账户必须使用相同的账户 ID。

`aws:SourceArn` 的值必须为 Amplify 应用程序的分支 ARN。指定该值为 `arn:`*`Partition`*`:amplify:`*`Region`*`:`*`Account`*`:apps/`*`AppId`*`/branches/`*`BranchName` 格式。

防范混淆代理问题最有效的方法是使用 aws:SourceArn 全局条件上下文键和资源的完整 ARN。如果不知道资源的完整 ARN，或者正在指定多个资源，请针对 ARN 未知部分使用带有通配符（*）的 aws:SourceArn 全局上下文条件键。例如 arn:aws:*servicename*::*123456789012*:*。

以下示例显示了一个角色信任策略，您可以应用该策略来限制对账户中任何 Amplify 应用程序的访问，并防止出现混淆代理问题。要使用此策略，请将示例策略中的红色斜体文本替换为您自己的信息。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": [
          "amplify.me-south-1.amazonaws.com",
          "amplify.eu-south-1.amazonaws.com",
          "amplify.ap-east-1.amazonaws.com",
          "amplifybackend.amazonaws.com",
          "amplify.amazonaws.com"
        ]
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:amplify:us-east-1:123456789012:apps/*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

以下示例显示了一个角色信任策略，您可以应用该策略来限制对账户中指定 Amplify 应用程序的访问，并防止出现混淆代理问题。要使用此策略，请将示例策略中的红色斜体文本替换为您自己的信息。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": [
          "amplify.me-south-1.amazonaws.com",
          "amplify.eu-south-1.amazonaws.com",
          "amplify.ap-east-1.amazonaws.com",
          "amplifybackend.amazonaws.com",
          "amplify.amazonaws.com"
        ]
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:amplify:us-east-1:123456789012:apps/d123456789/
branches/*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

Amplify 的安全最佳实践

Amplify 提供了许多安全功能，供您在制定和实施自己的安全策略时参考。以下最佳实操是一般准则，并不代表完整的安全解决方案。这些最佳实践可能不适合您的环境或不满足您的环境要求，请将其视为有用的建议而不是惯例。

在 Amplify 默认域中使用 Cookie

当您使用 Amplify 部署 Web 应用程序时，Amplify 会将其托管在默认的 `amplifyapp.com` 域上。您可以在格式为 `https://branch-name.d1m7bkiki6tdw1.amplifyapp.com` 的网址上查看您的应用程序。

为增强 Amplify 应用程序的安全性，已将 amplifyapp.com 域注册到公共后缀列表 (PSL)。为进一步增强安全性，如果您需要在 Amplify 应用程序的默认域名中设置敏感 Cookie，我们建议您使用带 __Host- 前缀的 Cookie。这将有助于保护您的域，防范跨站点请求伪造 (CSRF) 攻击。要了解更多信息，请参阅 Mozilla 开发者网络中的 Set-Cookie 页面。

Amplify 托管服务限额

以下是 AWS Amplify 托管的服务配额。服务限额（以往也称为限制）是 AWS 账户的服务资源或操作的最大数量。

新增减少 AWS 账户 了应用程序和并发任务配额。 AWS 根据您的使用情况自动提高这些配额。您也可以请求提高限额。

服务限额控制台提供有关您的账户限额的信息。您可以使用服务限额控制台查看默认限额，并对可调整的限额请求增加限额。有关更多信息，请参阅《服务配额用户指南》中的 Requesting a quota increase。

Name	默认值	可调整	说明
应用程序	每个受支持的区域：25 个	是	在当前地区的此账户中，您可以在 A AWS mplify Console 中创建的最大应用程序数量。
每个应用程序的分支数	每个受支持的区域：50 个	否	您可以在当前区域中的此账户中为每个应用程序创建的分支的最大数量。
生成构件大小	每个受支持的区域：5 GB	否	应用程序生成构件的最大大小（以 GB 为单位）。构建工件由 AWS Amplify 控制台在构建后部署。
缓存构件大小	每个受支持的区域：5 GB	否	缓存构件的最大大小（以 GB 为单位）。
并发任务	每个受支持的区域：5 个	是	在当前区域内的此账户中，您可以创建的并发作业的最大数量。

Name	默认值	可调整	说明
每个应用程序的域数	每个受支持的区域：5 个	是	您可以在当前区域中的此账户中为每个应用程序创建的域的最大数量。
环境缓存构件大小	每个受支持的区域：5 GB	否	环境缓存构件的最大大小（以 GB 为单位）。
手动部署 ZIP 文件大小	每个受支持的区域：5 GB	否	手动部署 ZIP 文件的最大大小（以 GB 为单位）。
每小时最大应用程序创建次数	每个受支持的区域：25 个	否	在当前区域内，您每小时可以在此账户中在 AWS Amplify Console 中创建的最大应用程序数量。
每秒请求令牌数	每个支持的区域：2 万个	是	为一个应用程序每秒请求令牌的最大数量。Amplify Hosting 根据请求好用的资源量（处理时间和数据传输量）为这些请求分配令牌。
每个域的子域数	每个受支持的区域：50 个	否	您可以在当前区域中的此账户中为每个域创建的子域的最大数量。
每个应用程序的 Webhook 数	每个受支持的区域：50 个	是	您可以在当前区域中的此账户中为每个应用程序创建的 Webhook 的最大数量。

有关 Amplify 服务限额的更多信息，请参阅 AWS 一般参考 中的 AWS Amplify 端点和限额。

Amplify Hosting 问题排查

如果您在使用 Amplify Hosting 时遇到错误或部署问题，请查询本部分中的相关主题。

主题

Amplify Hosting 问题排查

以下信息可帮助您解决与 Amplify Hosting 相关的一般性问题。

主题

HTTP 429 状态码（请求过多）

Amplify 会根据传入请求耗费的处理时间和数据传输量来控制向您的网站发送的每秒请求数（RPS）。如果您的应用程序返回 HTTP 429 状态码，则传入请求将超过为您的应用程序分配的处理时间和数据传输量。此应用程序限制由 Amplify 的 REQUEST_TOKENS_PER_SECOND 服务配额管理。有关限额的更多信息，请参阅Amplify 托管服务限额。

如果修复此问题，我们建议您优化应用程序以缩短请求持续时间、降低数据传输量，以提高应用程序的 RPS。例如，在同样使用 20,000 个词元的情况下，与延迟超过 200 毫秒的页面相比，在 100 毫秒内响应且高度优化的 SSR 页面可支持更高的 RPS。

同样，返回大小为 1 MB 的响应的应用程序将比返回 250 KB 响应大小的应用程序消耗更多的词元数。

我们还建议您通过配置Cache-Control标头来利用 Amazon CloudFront 缓存，以最大限度地延长给定响应在缓存中的保留时间。从 CloudFront 缓存中处理的请求不计入速率限制。每个 CloudFront 分发每秒最多可处理 250,000 个请求，使您能够使用缓存将应用程序扩展得非常高。有关 CloudFront 缓存的更多信息，请参阅 Amazon CloudFront 开发者指南中的优化缓存和可用性。

Amplify 控制台未显示我应用程序的构建状态和上次更新时间

当您在 Amplify 控制台中导航到所有应用程序页面时，系统会为当前区域中的每个应用程序显示一个磁贴。如果您看不到应用程序的构建状态（例如已部署）和上次更新时间，则表示该应用程序没有关联的 `Production` 暂存区分支。

Amplify 使用 `ListApps` API 在控制台中列出应用程序。Amplify 使用 `ProductionBranch.status` 属性来显示构建状态，并使用 `ProductionBranch.lastDeployTime` 属性来显示上次更新时间。有关此 API 的更多信息，请参阅 Amplify Hosting API 文档ProductionBranch中的。

按照以下说明将 `Production` 暂存区关联到您应用程序的分支。

1. 登录 Amplify 控制台。
2. 在所有应用程序页面上，选择要更新的应用程序。
3. 在导航窗格中，依次选择应用程序设置和分支设置。
4. 在分支设置部分中，选择编辑。
5. 在生产分支中，选择要使用的分支名称。
6. 选择保存。
7. 返回所有应用程序页面。现在应该会显示您应用程序的构建状态和上次更新时间。

未为新拉取请求创建 Web 预览

借助 Web 预览功能，您可以在将拉取请求合并到集成分支之前预览拉取请求将执行的更改。对于向存储库发出的每个拉取请求，Web 预览会将其部署到一个与主网站所用 URL 不同的唯一预览 URL。

如果您已为应用程序开启网络预览，但不是为新应用程序创建的 PRs，请调查以下原因是否是导致问题的原因。

1. 检查您的应用程序是否已达到最大 Branches per app 服务配额。有关限额的更多信息，请参阅Amplify 托管服务限额。

   为遵循每个应用程序 50 个分支的默认配额，可考虑在应用程序中启用自动分支删除。这可以防止在您的账户中累积存储库中已不存在的分支。

2. 如果您使用的是公共 GitHub 仓库，并且您的 Amplify 应用程序附加了 IAM 服务角色，那么出于安全考虑，Amplify 不会创建预览。例如，带有后端的应用程序和部署到 WEB_COMPUTE 托管平台的应用程序需要一个 IAM 服务角色。因此，如果这些类型的应用程序存储库是公开的，则无法为其启用 Web 预览。

   要使网页预览适用于您的应用程序，您可以取消关联服务角色（如果应用程序没有后端或不是WEB_COMPUTE应用程序），也可以将 GitHub 存储库设为私有。

我的手动部署在 Amplify 控制台中停滞在待处理状态

通过手动部署，您可以在不连接 Git 提供商的情况下，用 Amplify Hosting 发布您的 Web 应用程序。您可以使用以下四种部署选项之一。

1. 将您的应用程序文件夹拖放到 Amplify 控制台中。
2. 将包含网站生成构件的 .zip 文件拖放到 Amplify 控制台中。
3. 将包含网站生成构件的 .zip 文件上传到某个 Amazon S3 存储桶，然后在 Amplify 控制台中将该存储桶连接到某个应用程序。
4. 在 Amplify 控制台中使用指向包含网站生成构件的 .zip 文件的公有 URL。

我们了解到在 Amplify 控制台中使用应用程序文件夹进行手动部署时，拖放功能存在问题。可能导致这些部署失败的原因如下。

- 出现暂时性网络问题。
- 在上传过程中，文件在本地出现更改。
- 浏览器会话尝试同时上传大量静态资源。

我们正在全力提高拖放上传功能的可靠性。在此期间，我们建议您使用 .zip 文件而不是拖放应用程序文件夹的功能。

我们强烈建议将 .zip 文件上传到 Amazon S3 存储桶，因为这样可以避免从 Amplify 控制台上传文件，并且可以提高手动部署的可靠性。Amplify 与 Amazon S3 的集成简化了这一过程。有关更多信息，请参阅 将 Amazon S3 存储桶中的静态网站部署到 Amplify。

我需要更新我应用程序的 Node.js 版本

对于使用 Node.js 版本 14、16 和 18 的应用程序，Amplify 将于 2025 年 9 月 15 日结束支持。在该日期之后，具体行为将取决于应用程序的类型：

- SSR 应用程序：使用已弃用的 Node.js 版本时会导致生成失败。在升级到 Node.js 20 或更高版本之前，您将无法部署更新。
- 非 SSR 应用程序：如果您通过 buildspec 或实时软件包更新手动安装已弃用的 Node.js 版本，则可以继续使用这些版本。

不论是哪个 Node.js 版本，已经部署的应用程序都将继续运行。

如果您使用的是 Amazon Linux 2023 构建映像，则默认支持 Node.js 版本 20。从 2025 年 9 月 15 日起，该 AL2023 映像将自动支持 Node.js 22，并将其默认 Node.js 版本从 18 更改为 22。

亚马逊 Linux 2 (AL2) 不自动支持 Node.js 版本 20 或更高版本。如果您目前正在使用 AL2，我们建议您切换到 AL2023。您可以在 Amplify 控制台中更改构建映像，也可以使用支持所指定 Node.js 版本的自定义构建映像。

在升级之前，我们建议您使用新分支来测试您的应用程序，验证其是否能够正常运行。

升级选项

Amplify 控制台

您可以使用 Amplify 控制台中的实时软件包更新功能来指定要使用的 Node.js 版本。有关说明，请参阅在构建映像中使用特定程序包和依赖项版本。

自定义构建映像

如果您使用的是自定义构建镜像，并且映像上安装了 NVM，则可以将 `nvm install 20` 添加到您的 Dockerfile 中。要详细了解有关自定义构建映像的要求和配置说明，请参阅自定义构建映像。

构建设置

您可以通过将 `nvm use` 命令添加到 preBuild 命令部分，从而指定要在应用程序的 `amplify.yml` 构建设置中使用的 Node.js 版本。有关更新应用程序构建设置的说明，请参阅配置 Amplify 应用程序的构建设置。

下例说明了如何自定义构建设置，以在名为 node-20 的测试分支上将默认 Node.js 版本设置为 Node.js 18 并升级到 Node.js 版本 20。

```
frontend:
  phases:
    preBuild:
      commands:
        - nvm use 18
        - if [ "${AWS_BRANCH}" = "node-20" ]; then nvm use 20; fi
```

> **Warning**
>
> 请注意，preBuild 命令会在实时软件包更新完成后运行。nvm use 命令指定的 Node.js 版本将会覆盖由实时软件包更新设置的 Node.js 版本。

Amazon Linux 2023 构建映像问题排查

以下信息可以帮助您解决亚马逊 Linux 2023 (AL2023) 版本映像的问题。

主题

我想使用 Python 运行时运行 Amplify 函数

现在，当您部署新应用程序时，Amplify Hosting 默认使用亚马逊 Linux 2023 版本映像。 AL2023 预装了 Python 版本 3.8、3.9、3.10 和 3.11。

为了向后兼容 Amazon Linux 2 镜像， AL2023 构建镜像预装了旧版 Python 的符号链接。

默认全局使用 Python 3.10 版。要使用特定 Python 版本构建函数，请在您的应用程序的编译规范文件中运行以下命令。

```
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
```

我想运行需要超级用户或 root 权限的命令

如果您在使用 Amazon Linux 2023 构建镜像，但在运行需要超级用户或 root 权限的系统命令时出错，则必须使用 Linux `sudo` 命令运行这些命令。例如，如果您在运行 `yum install -y gcc` 时遇到错误，请使用 `sudo yum install -y gcc`。

Amazon Linux 2 构建镜像使用根用户，但是 Amplify 的 AL2023 镜像使用自定义`amplify`用户运行你的代码。Amplify 授予此用户使用 Linux `sudo` 命令运行命令的权限。对于需要超级用户权限的命令，最佳实践是使用 `sudo`。

构建问题排查

如果您在创建或构建 Amplify 应用程序时遇到问题，请参阅本节中的相关主题以获取帮助。

主题

向我存储库发出的新提交未触发 Amplify 构建

如果向 Git 存储库发出的新提交没有触发 Amplify 构建，请检查您的 Webhook 是否仍存在于您的存储库中。如果存在，请检查 Webhook 请求的历史记录，看看是否出现任何失败。对于传入的 Webhook，Amplify 的有效载荷大小上限为 256 KB。如果您将提交推送到包含大量已更改文件的存储库，则可能会超过此限制并导致无法触发构建。

创建新应用程序时，我的存储库名称未在 Amplify 控制台中列出

在 Amplify 控制台中创建新应用程序时，您可以在添加存储库和分支页面上选择组织的可用存储库。如果目标存储库近期没有更新，则可能不会显示在列表中。如果组织有大量存储库，则可能会发生这种情况。要解决此问题，请先将提交推送到该存储库，然后在控制台中刷新存储库列表。这应会使该存储库显示。

我的构建失败了并返回 **Cannot find module aws-exports** 错误（仅限 Gen 1 应用程序）

如果您的应用程序在构建过程中找不到 aws-exports.js 文件，则会返回以下错误。

```
TS2307: Cannot find module 'aws-exports'
```

在构建后端期间，Amplify 命令行界面（CLI）会生成 aws-exports.js 文件。要解决此错误，您必须创建一个 aws-exports.js 文件以在构建过程中使用。将以下代码添加到构建规范中以创建该文件：

```
backend:
  phases:
    build:
      commands:
        - "# Execute Amplify CLI with the helper script"
        - amplifyPush --simple
```

有关 Amplify 应用程序构建规范设置的完整示例，请参阅构建规范的 YAML 语法参考。

我想覆盖构建超时值

默认构建超时为 30 分钟。您可以使用 _BUILD_TIMEOUT 环境变量来覆盖默认构建超时值。最小构建超时为 5 分钟。最大构建超时值为 120 分钟。

有关在 Amplify 控制台中设置应用程序环境变量的说明，请参阅设置环境变量。

自定义域问题排查

如果您在将自定义域连接到 Amplify 应用程序时遇到问题，请查阅本节中的主题以获取帮助。

如果您在这里找不到问题的解决方案，请联系 支持。有关更多信息，请参阅 AWS 支持 用户指南中的创建支持案例。

主题

我需要验证我的 CNAME 别名记录是否已解析

1. 通过第三方域提供商更新 DNS 记录后，您可以使用诸如 dig 之类的工具或免费网站（例如 https://www.whatsmydns.net/）来验证您的别名记录是否正确解析。以下屏幕截图演示了如何使用 whatsmydns.net 来查看该域 www.example.com 的别名记录。

2. 选择搜索，whatsmydns.net 会显示您的别名记录结果。以下屏幕截图例举了验证别名记录是否正确解析为 cloudfront.net 网址的结果列表。

我在第三方托管的域卡在了等待验证状态

1. 如果您的自定义域陷入待验证状态，请验证您的 CNAME 记录是否正在解析。有关执行此任务的说明，请参阅之前的问题排查主题如何验证我的 CNAME 解析。
2. 如果您的 CNAME 记录无法解析，请向您的域提供商确认 CNAME 条目存在于您的 DNS 设置中。

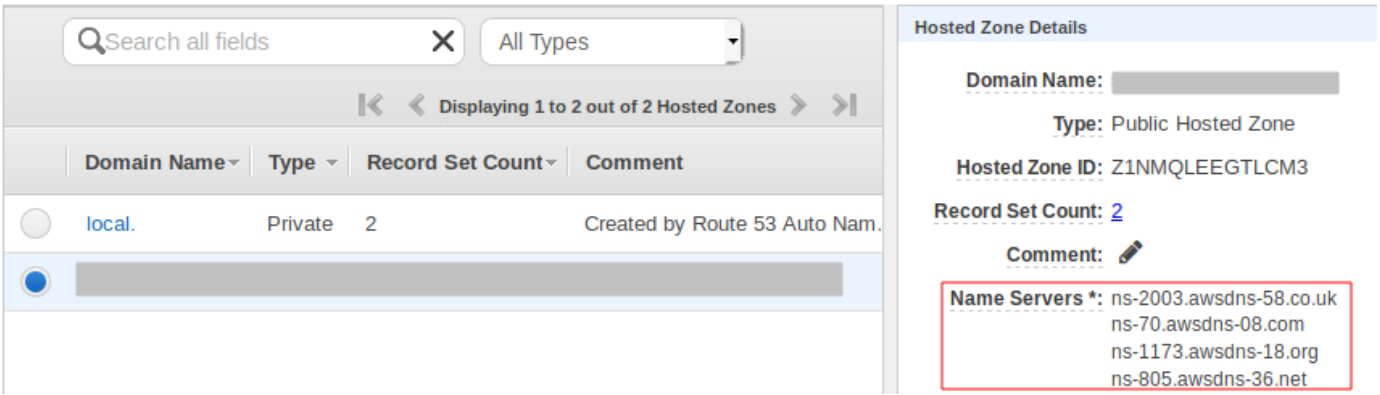
> **Important**
>
> 创建您的自定义域后，请务必立即更新 CNAME 记录。在 Amplify 控制台中创建应用程序后，每隔数分钟就会检查一次 CNAME 记录，以确定它是否已解析。如果一小时后仍未解决，则每隔数小时进行一次检查，这可能会导致您的域名在准备好使用方面会有所延迟。如果您在创建应用程序几小时后添加或更新了 CNAME 记录，则这很可能是您的应用程序陷入待验证状态的原因。

3. 如果已确认 CNAME 记录存在，那么您的 DNS 提供商可能存在问题。您可以联系 DNS 提供商，来诊断 DNS 验证 CNAME 未解析的原因，也可以将 DNS 迁移到 Route53。有关更多信息，请参阅将 Amazon Route 53 作为现有域的 DNS 服务。

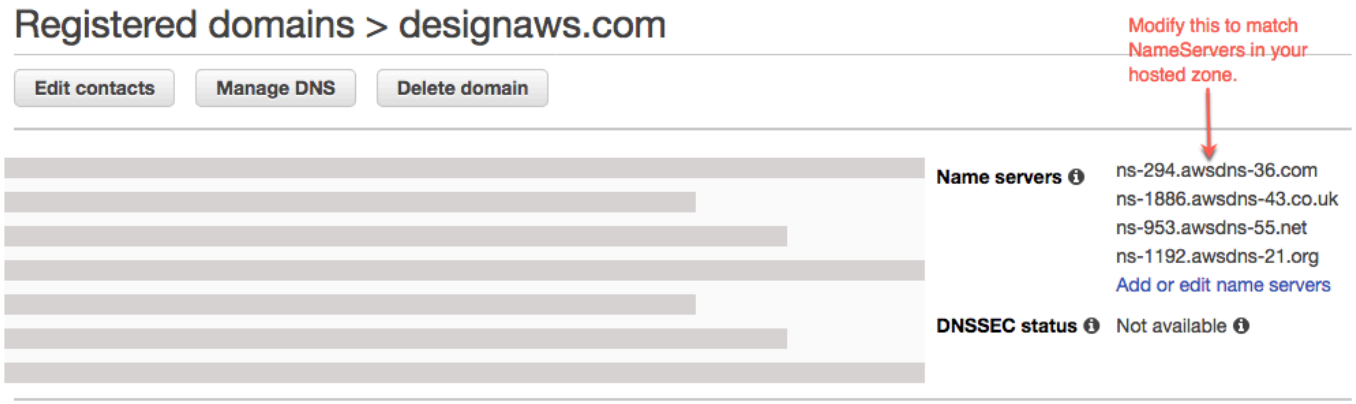
我在 Amazon Route 53 托管的域卡在了等待验证状态

如果您将域转移到 Amazon Route 53，则您的域可能与创建应用程序时由 Amplify 发布的名称服务器不同。执行以下步骤诊断出错的原因。

1. 登录 Amazon Route 53 控制台
2. 在导航窗格中，选择托管区域，然后选择要连接的域的名称。
3. 记录托管区域详细信息部分中的名称服务器值。您需要这些值来完成下一步。以下 Route 53 控制台屏幕截图在右下角处显示了名称服务器值的位置。



4. 在导航窗格中，选择 Registered domains。验证已注册域部分显示的名称服务器是否与您在上一步中在托管区域详细信息部分中记录的名称服务器值一致。如果不一致，请编辑名称服务器值，使其与托管区域中的值一致。以下 Route 53 控制台屏幕截图在右侧显示了名称服务器值的位置。



5. 如果这仍无法解决问题，请联系 支持。有关更多信息，请参阅 AWS 支持 用户指南中的创建支持案例。

包含多级子域的应用程序停滞在“待验证”状态

如果包含多级子域的应用程序在连接到第三方 DNS 提供商时停滞在待验证状态，则说明您的 DNS 记录格式可能存在问题。某些 DNS 提供商会自动将二级域（SLD）和顶级域（TLD）的域后缀添加到您的记录中。如果您也以包含 SLD 和 TLD 的格式指定域，则可能会导致域验证问题。

在连接域时，请先尝试使用 Amplify 提供的完整格式指定域名，例如 `_hash.docs.backend.example.com`。如果 SSL 配置停滞在待验证状态，请尝试移除记录中的 TLD 和 SLD。例如，假设完整格式为 `_hash.docs.backend.example.com`，则指定 `_hash.docs.backend`。请等待 15 到 30 分钟，以便完成记录传播。然后使用诸如 MX Toolbox 之类的工具来检查能否正常完成验证过程。

我的 DNS 提供商不支持使用完全限定域名的 A 记录

某些 DNS 提供商不支持使用完全限定域名（FQDN）的 A 记录（例如 `example.cloudfront.net`）。例如，Cloudflare A records 只能写入 IPv4 地址，不支持 FQDNs。要解决此限制，我们建议在 DNS 配置中使用 CNAME 记录，而不是 A records。

例如，以下 DNS 配置就使用了 A record。

```
A     | @ | ***.cloudfront.net
CNAME | www | ***.cloudfront.net
```

将其更改为以下 DNS 配置，从而仅使用 CNAME 记录。

```
CNAME | @ | ***.cloudfront.net
CNAME | www | ***.cloudfront.net
```

这种解决方法使您可以正确地将顶点域（@ 记录）指向诸如之类的服务 CloudFront，同时避免 Cloudflare 系统A records中 IPv4仅有的限制。

我收到一个 CNAMEAlreadyExistsException 错误

如果您遇到CNAMEAlreadyExistsException错误，则表示您尝试连接的其中一个主机名（子域名或顶点域）已部署到另一个 Ama CloudFront zon 分发中。导致错误的根源取决于当前使用的托管服务和 DNS 提供商。

CNAME别名（例如`example.com`或）一次`sub.example.com`只能与单个 CloudFront分配相关联。CNAMEAlreadyExistsException表示您的域名已与其他 CloudFront 分配相关联，无论是在同一个账户中 AWS 账户，还是可能在不同的账户中。在 Amplify Hosting 创建的新 CloudFront 发行版生效之前，必须先解除该域名与之前的发行版的关联。如果您或您的组织拥有多个帐户，则可能需要检查多个 AWS 账户帐户。

执行以下步骤来诊断CNAMEAlreadyExistsException错误的原因。

1. 登录 Amazon CloudFront 控制台并确认您没有将此域部署到其他分配。一次只能将CNAME一条记录附加到一个 CloudFront 发行版中。
2. 如果您之前已将该域部署到 CloudFront 分配中，则必须将其删除。
   a. 在左侧导航菜单中，选择分发。
   b. 选择要编辑的分发的名称。
   c. 选择通用选项卡。在设置部分中，选择编辑。
   d. 从备用域名 (ANAME) 中移除此域名。然后选择保存更改。
3. 确认当前 AWS 账户 或其他 CloudFront 版本中不存在使用此域的其他发行版 AWS 账户。如果不会导致任何当前正在运行的服务出现中断，则可尝试删除并重新创建托管区。
4. 查看此域是否与您拥有的其他 Amplify 应用程序相关联。如果是，请确保您没有尝试重用其中一个主机名。如果您正在将 `www.example.com` 用于其他应用程序，则不能将 `www.example.com` 用于您当前正在连接的应用程序。您可以使用其他子域，例如 `blog.example.com`。
5. 如果该域曾成功连接到其他应用程序，然后在过去一小时内被删除，请至少在一小时后重试。如果您在 6 小时后仍看到此异常，请与联系 支持。有关更多信息，请参阅 AWS 支持 用户指南中的创建支持案例。

6. 如果您通过 Route 53 管理您的域，请务必清理指向旧 CloudFront 分配的所有托管区域CNAME或ALIAS记录。
7. 完成上述步骤后，从 Amplify Hosting 中移除自定义域，然后在 Amplify 控制台中重新启动连接自定义域的工作流。

我收到需要额外验证错误

如果您收到 “需要额外验证” 错误，这意味着 AWS Certificate Manager (ACM) 需要其他信息来处理此证书申请。这可以用作防止欺诈措施，例如域名位于 Alexa 1000 强网站中时。要提供此必要信息，请使用支持中心联系 支持。如果您没有支持计划，请在 ACM 开发论坛中发布新话题。

> **Note**
>
> 您无法为 Amazon 拥有的域名 (例如以 amazonaws.com、cloudfront.net 或 elasticbeanstalk.com 结尾的域名) 请求证书。

我在网址上收到 404 错误 CloudFront

为了提供流量，Amplify Hosting 通过别名 CloudFront 记录指向网址。在将应用程序连接到自定义域名的过程中，Amplify 控制台会显示该应用程序的 CloudFront 网址。但是，您不能使用此 CloudFront URL 直接访问您的应用程序。它会返回 404 错误。您的应用程序只能使用 Amplify 应用网址（例如 `https://main.d5udybEXAMPLE.amplifyapp.com`）或您的自定义域（例如 `www.example.com`）进行解析。

Amplify 需要将请求路由到正确部署的分支，并使用主机名称来执行此操作。例如，您可以配置指向应用程序主线分支的域 `www.example.com`，也可以配置指向相同应用程序开发分支的域 `dev.example.com`。因此，您必须根据应用程序的配置子域来访问您的应用程序，这样 Amplify 才能相应地路由请求。

我在访问我的域时收到 SSL 证书或 HTTPS 错误

如果您使用第三方 DNS 提供商配置了证书颁发机构授权 (CAA) DNS 记录，则 AWS Certificate Manager (ACM) 可能无法更新或补发自定义域 SSL 证书的中间证书。要解决此问题，您需要添加一条 CAA 记录来信任 Amazon 证书颁发机构的至少一个域。以下过程描述了您需要执行的步骤。

添加 CAA 记录以信任 Amazon 证书颁发机构

1. 通过您的域提供商配置一条 CAA 记录，以信任 Amazon 证书颁发机构的至少一个域。有关配置 CAA 记录的更多信息，请参阅 AWS Certificate Manager 用户指南中的证书颁发机构授权 (CAA) 问题。
2. 使用以下方法之一更新 SSL 证书：
   - 使用 Amplify 控制台手动更新。

     > **Note**
     >
     > 此方法会导致您的自定义域宕机。

     a. 登录 AWS 管理控制台 并打开 Amplify 控制台。
     b. 选择要添加 CAA 记录的应用程序。
     c. 在导航窗格中，依次选择应用程序设置、域管理。
     d. 在域管理页面上，删除自定义域。
     e. 再次将您的应用程序连接到自定义域。此过程会颁发新的 SSL 证书，其中间证书现在可以由 ACM 管理。

        要将应用程序重新连接到您的自定义域，请使用与您正在使用的域提供商相对应的以下步骤之一。

        - 添加由 Amazon Route 53 管理的自定义域.
        - 添加由第三方 DNS 提供商管理的自定义域.
        - 更新由管理的域的 DNS 记录 GoDaddy.
   - 请联系 支持 以补发您的 SSL 证书。

域重定向中不支持路径组件

域重定向仅会匹配主机名部分。不支持在基于域的源规则中使用路径组件（例如 `"https://domain.com/path"`），这会导致该规则被忽略且不会返回错误。有关更多信息，请参阅 重定向和重写示例参考。

跨账户域名关联时出现了 400 错误

当使用已经或以前与同一地区其他 AWS 账户中的不同 Amplify 应用程序关联的域名发起对 Amplify 应用程序的 DomainAssociation 请求时，这被视为跨账户域关联。如果您收到此错误，则表示您正在尝试跨账户域名关联，需要手动验证。如果您想继续进行跨账户域名关联，请联系 AWS 支持寻求帮助。

为在服务器端渲染的应用程序排查问题

如果您在使用 Amplify Hosting 计算部署 SSR 应用程序时遇到意外问题，请查看以下问题排查主题。如果您在此处看不到问题的解决方案，请参阅 Amplify Ho GitHub sting Issues 存储库中的 SSR 网络计算疑难解答指南。

主题

我需要使用框架适配器的帮助

如果您在部署使用框架适配器的 SSR 应用程序时遇到问题，请参阅 为任意 SSR 框架使用开源适配器。

边缘 API 路由导致我的 Next.js 构建失败

目前，Amplify 不支持 Next.js 边缘 API 路由。使用 Amplify 托管应用程序时，必须使用非边缘 APIs 和中间件。

按需增量静态再生成不适用于我的应用程序

从版本 12.2.0 开始，Next.js 支持增量静态再生成 (ISR)，以手动清除特定页面的 Next.js 缓存。但是，Amplify 目前不支持按需 ISR。如果您的应用程序使用 Next.js 按需重新验证，则当您将应用程序部署到 Amplify 时，此功能将无法正常工作。

我的应用程序的构建输出超出了允许的大小上限

目前，Amplify 支持的 SSR 应用程序构建输出大小上限为 220 MB。如果您收到错误消息，指出您的应用程序构建输出的大小超过了允许的大小上限，则必须采取措施来将其缩小。

要缩小应用程序构建输出的大小，您可以检查应用程序的构建构件，并确定要更新或删除的任何大型依赖项。首先，将构建构建下载到本地计算机。然后检查目录的大小。例如，node_modules 目录可能包含由 Next.js 服务器运行时文件引用的二进制文件，例如 @swc 和 @esbuild。由于运行时不需要这些二进制文件，因此可以在构建之后将其删除。

按照以下说明下载应用程序的编译输出并使用 AWS Command Line Interface (CLI) 检查目录的大小。

下载并检查 Next.js 应用程序的构建输出

1. 打开终端窗口，并运行以下命令。将应用程序 ID、分支名称和作业 ID 更改为您自己的相应信息。对于作业 ID，请使用您正在研究的失败构建的构建编号。

```
aws amplify get-job --app-id abcd1234 --branch-name main --job-id 2
```

2. 在终端输出中，在 job、steps、stepName: "BUILD" 部分中找到预签名的项目 URL。以下示例输出中以红色突出显示 URL。

```
"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. 复制 URL 并将其粘贴到浏览器窗口中。artifacts.zip 文件将下载到您的本地计算机中。这是您的构建输出。
4. 运行 du 磁盘使用命令，以检查目录大小。以下示例命令会返回 compute 和 static 目录的大小。

```
du -csh compute static
```

以下输出示例中包含 compute 和 static 目录的大小信息。

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

5. 打开 compute 目录并找到 node_modules 文件夹。查看您的依赖项，寻找可以更新或移除以缩小文件夹的文件。
6. 如果您的应用程序包含运行时不需要的二进制文件，请在构建后将其删除，方法是将以下命令添加到应用程序 amplify.yml 文件的构建部分中。

```
- 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
```

以下示例展示了 amplify.yml 文件的构建命令部分，其中包含在运行生产构建之后添加的这些命令。

```
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
```

我的构建失败并出现内存不足错误

Next.js 允许您缓存构建构件，以提高后续构建的性能。此外，Amplify 的 AWS CodeBuild 容器会代表您压缩此缓存并将其上传到 Amazon S3，以提高后续构建性能。这可能导致您的构建失败，出现内存不足错误。

执行以下操作以防止您的应用程序在构建阶段超过内存限制。首先，从构建设置的 cache.paths 部分中删除 `.next/cache/**/*`。接下来，从您的构建设置文件中移除 `NODE_OPTIONS` 环境变量。相反，在 Amplify 控制台中设置 `NODE_OPTIONS` 环境变量，定义节点最大内存限制。有关在 Amplify 控制台中设置环境变量的更多信息，请参阅 设置环境变量。

完成这些更改后，重新尝试构建。如果成功了，重新将 `.next/cache/**/*` 添加到构建设置文件的 cache.paths 部分。

有关用于提高构建性能的 Next.js 缓存配置的更多信息，请参阅 Next.js CodeBuild 网站上的 AWS。

我的应用程序的 HTTP 响应太大

目前，Amplify 支持使用 Web 计算平台的 Next.js 12 及更新版本应用程序的最大响应大小为 5.72 MB。超过该限制的响应会返回 504 错误，且不向客户端发送任何内容。

如何在本地测量计算应用程序的启动时间？

按照以下说明确定 Next.js 12 或更 initialization/start 高版本的 Compute 应用程序的本地启动时间。您可以将本地应用程序的性能与在 Amplify Hosting 上的性能进行比较，并根据结果提高应用程序的性能。

在本地测量 Next.js 计算应用程序的初始化时间

1. 打开应用程序的 `next.config.js` 文件并将 `output` 选项设置为 `standalone`，如下所示。

```
** @type {import('next').NextConfig} */
const nextConfig = {
  // Other options
  output: "standalone",
};

module.exports = nextConfig;
```

2. 打开终端窗口，并运行以下命令来构建应用程序。

```
next build
```

3. 运行以下命令将 .next/static 文件夹复制到 .next/standalone/.next/static。

```
cp -r .next/static .next/standalone/.next/static
```

4. 运行以下命令将 public 文件夹复制到 .next/standalone/public。

```
cp -r public .next/standalone/public
```

5. 运行以下命令启动 Next.js 服务器。

```
node .next/standalone/server.js
```

6. 记下从运行第 5 步中的命令到服务器启动之间的间隔时间。当服务器在端口上侦听时，应会打印以下消息。

```
Listening on port 3000
```

7. 记下第 6 步中启动服务器后，加载任何其他模块使用的时间。例如，类似于 bugsnag 的库需要 10-12 秒才能完成加载。加载完成后，将会显示确认消息 [bugsnag] loaded。
8. 将第 6 步和第 7 步中的持续时间相加，此结果是您的计算应用程序的本地 initialization/start 运行时间。

我的构建失败了，并返回了“已弃用的 Node.js 版本”错误

问题：SSR 应用程序构建失败，并返回“Node.js 版本不受支持”错误。

```
# NODE.JS VERSION NOT SUPPORTED
================================================================================
Your application uses Node.js v18.x.x, which is no longer supported.
AWS Amplify Console has ended support for Node.js 14, Node.js 16 and Node.js 18.

To deploy your application, please upgrade to Node.js 20 or later.

For detailed migration guidelines, visit: https://docs.aws.amazon.com/amplify/latest/
userguide/troubleshooting-general.html#update-node-version
================================================================================
```

原因：SSR 应用程序是使用已弃用的 Node.js 版本（14.x、16.x 或 18.x）构建的。自 2025 年 9 月 15 日起，Amplify 会禁止部署在构建过程中使用这些已弃用版本的 SSR 应用程序。

更新构建环境以使用 Node.js 20 或更高版本。有关详细说明，请参阅我需要更新我应用程序的 Node.js 版本。

重定向和重写问题排查

如果在为 Amplify 应用程序设置重定向和重写时遇到问题，请参阅本节中的相关主题以获取帮助。

主题

即使具有 SPA 重定向规则，对某些路由的访问仍被拒绝。

如果您在访问某些具有 SPA 重定向规则的路由时遇到访问被拒绝错误，则可能是因为应用程序的构建设置中未正确设置 `baseDirectory`。例如，应用程序的前端构建到 `build` 目录时，构建设置也必须指向 `build` 目录。以下构建规范示例演示了此设置。

```
frontend:
  artifacts:
    baseDirectory: build
    files:
        - "**/*"
```

有关 Amplify 应用程序构建规范设置的完整示例，请参阅构建规范的 YAML 语法参考

我想设置 API 的反向代理

您可以使用以下 JSON 来设置动态端点的反向代理。

```
[
  {
    "source": "/documents/<*>",
    "target": "https://otherdomain/resource/<*>",
    "status": "200",
    "condition": null
  }
```

```
]
```

有关为 Amplify 应用程序设置第三方 API 反向代理的基本示例，请参阅反向代理重写。

缓存问题排查

如果某个 Amplify 应用程序遇到缓存问题，请参阅本节中的相关主题以获取帮助。

主题

- 我想减小应用程序的缓存大小
- 我想禁止某个应用程序读取缓存

我想减小应用程序的缓存大小

当您使用缓存时，可能会缓存构建之间未被清理的中间文件。缓存这些不常使用的文件会增加缓存的大小。为防止出现这种情况，您可以在应用程序构建规范的 cache 部分中使用 ! 指令将特定文件夹排除在缓存范围之外。

以下构建设置示例演示了如何使用 ! 指令来指定不需要缓存的文件夹。

```
cache:
  paths:
    - node_modules/**/*
    - "!node_modules/path/not/to/cache"
```

缓存 node_modules 文件夹时，默认情况下将会忽略 node_modules/.cache。

有关 Amplify 应用程序构建规范设置的完整示例，请参阅构建规范的 YAML 语法参考

我想禁止某个应用程序读取缓存

如果要禁止某个应用程序读取缓存，请从该应用程序的构建规范中移除缓存部分。

设置 Amplify 对仓库的访问权限 GitHub

Amplify 现在使用 GitHub 应用程序功能来授权 Amplify 对存储库的只读访问权限。 GitHub使用 Amplify GitHub 应用程序，可以对权限进行更精细的调整，使您能够仅向 Amplify 授予访问您指定的存储库的权限。要了解有关 GitHub应用程序的更多信息，请参阅 GitHub 网站上的关于 GitHub 应用程序。

当你连接存储在存储库中的新应用程序时，Am GitHub plify 默认使用 GitHub 该应用程序来访问存储库。但是，您之前从存储库连接的现有 Am GitHub plify 应用程序将用于 OAuth 访问。 CI/CD 将继续适用于这些应用程序，但我们强烈建议您将它们迁移到使用新的 Amplify GitHub 应用程序。

当您使用 Amplify 控制台部署新应用程序或迁移现有应用程序时，系统会自动将您定向到 Amplify 应用程序的安装位置。 GitHub 要手动访问应用程序的安装登录页面，请打开 Web 浏览器并按地区导航到该应用程序。使用格式`https://github.com/apps/aws-amplify-`*`REGION`*，*`REGION`*替换为要部署 Amplify 应用程序的区域。例如，要在美国西部（俄勒冈）地区安装 Amplify GitHub 应用程序，请导航至 https://github.com/apps/ aws-amplify-us-west -2。

主题

为新部署安装和授权 Amp GitHub lify 应用程序

当您使用存储库中的现有代码将新应用程序部署到 Am GitHub plify 时，请按照以下说明安装和授权该应用程序。 GitHub

安装和授权 Amplify 应用程序 GitHub

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 在所有应用程序页面中，选择新建应用程序，然后选择托管 Web 应用程序。
3. 在 “开始使用 Amplify Hosting” 页面上，选择 GitHub，然后选择继续。
4. 如果这是首次连接 GitHub 存储库，则会在您的浏览器中打开 GitHub .com 上的新页面，请求 AWS Amplify 在您的 GitHub账户中进行授权。选择授权。
5. 接下来，您必须在您的 GitHub 账户中安装 Amplify GitHub 应用程序。GitHub.com 上会打开一个页面，请求允许 AWS Amplify 在你的 GitHub账户中安装和授权。
6. 选择要在其中安装 Amplify 应用程序 GitHub 的 GitHub 账户。
7. 请执行以下操作之一：
   - 要将安装应用于所有存储库，请选择所有存储库。
   - 要将安装限于您所选择的特定存储库，请选择仅选择存储库。确保在您选择的存储库中包含要迁移的应用程序的存储库。

8. 选择安装和授权。
9. 您将被重定向到 Amplify 控制台中应用程序的添加存储库分支页面。
10. 在最近更新的存储库列表中，选择要连接的存储库的名称。
11. 在分支列表中，选择要连接的存储库分支的名称。
12. 选择下一步。
13. 在配置构建设置页面上，选择下一步。
14. 在查看页面上，选择保存并部署。

将现有OAuth应用程序迁移到 Amplif GitHub y 应用程序

您之前从 GitHub 存储库连接的现有 Amplify 应用程序 OAuth用于访问存储库。我们强烈建议您将这些应用程序迁移到使用 Amplify GitHub 应用程序。

按照以下说明迁移应用程序并删除 GitHub 账户中相应的 OAuth webhook。请注意，根据是否已安装 Amplify GitHub 应用程序，迁移过程会有所不同。迁移第一个应用程序并安装并授权该 GitHub 应用程序后，您只需要更新存储库权限即可进行后续应用程序迁移。

将应用程序从迁移 OAuth 到 GitHub 应用程序

1. 登录 AWS 管理控制台 并打开 Amplify 控制台。
2. 选择要迁移的应用程序。
3. 在应用程序的信息页面上，找到蓝色的 “迁移到我们的 GitHub应用程序” 消息，然后选择开始迁移。
4. 在 “安装和授权 GitHub 应用程序” 页面上，选择 “配置 GitHub 应用程序”。
5. 您的浏览器中将打开一个新页面（ GitHub.com ），请求 AWS Amplify 在您的 GitHub 账户中进行授权。选择授权。
6. 选择要在其中安装 Amplify 应用程序 GitHub 的 GitHub 账户。
7. 请执行以下操作之一：
   - 要将安装应用于所有存储库，请选择所有存储库。
   - 要将安装限于您所选择的特定存储库，请选择仅选择存储库。确保在您选择的存储库中包含要迁移的应用程序的存储库。
8. 选择安装和授权。
9. 您将被重定向到 GitHub Amplify 控制台中应用程序的 “安装和授权应用程序” 页面。如果 GitHub 授权成功，您将看到一条成功消息。选择下一步。

10. 在完成安装页面上，选择完成安装。此步骤将删除您现有的 webhook，并创建一个新的 webhook，然后完成迁移。

为 CloudFormation CLI 和 SDK 部署设置 Amplify GitHub 应用程序

您之前从 GitHub 存储库连接的现有 Amplify 应用程序 OAuth用于访问存储库。这可能包括您使用 Amplify 命令行界面 (CLI) 部署的应用程序 CloudFormation，或者。 SDKs我们强烈建议您将这些应用程序迁移到使用新的 Amplify GitHub 应用程序。迁移必须在 AWS 管理控制台的 Amplify 控制台中执行。有关说明，请参阅将现有OAuth应用程序迁移到 Amplif GitHub y 应用程序。

你可以使用 CloudFormation Amplify CLI 和 SDKs 来部署一个使用该应用程序访问存储库的新 Amplify GitHub 应用程序。此过程要求您首先在自己的 GitHub 账户中安装 Amplify GitHub 应用程序。接下来，您需要在您的 GitHub 账户中生成个人访问令牌。最后，部署应用程序并指定个人访问令牌。

在你的账户中安装 Amplify GitHub 应用程序

1. 打开网络浏览器，导航到要部署应用程序的 AWS 区域中的 GitHub Amplify 应用程序的安装位置。

   使用该格式`https://github.com/apps/aws-amplify-REGION/installations/new`，*REGION*替换为你自己的输入。例如，如果您要在美国西部（俄勒冈州）区域安装应用程序，请指定 `https://github.com/apps/aws-amplify-us-west-2/installations/new`。
2. 选择要在其中安装 Amplify 应用程序 GitHub 的 GitHub 账户。
3. 请执行以下操作之一：
   - 要将安装应用于所有存储库，请选择所有存储库。
   - 要将安装限于您所选择的特定存储库，请选择仅选择存储库。确保在您选择的存储库中包含要迁移的应用程序的存储库。
4. 选择安装。

在您的 GitHub 账户中生成个人访问令牌

1. 登录您的 GitHub 账户。
2. 在右上角，找到您的个人资料照片，然后从菜单中选择设置。
3. 在左侧导航菜单中，选择 SMTP 设置。
4. 在GitHub 应用程序页面的左侧导航菜单中，选择个人访问令牌。

5. 在个人访问令牌页面上，选择生成新令牌。
6. 在新建个人访问令牌页面上，请在注释中输入令牌的描述性名称。
7. 在选择范围部分，选择 admin:repo_hook。
8. 选择生成令牌。
9. 复制并保存个人访问令牌。当你使用 CLI 部署 Amplify 应用程序时，你需要提供它 CloudFormation，或者。 SDKs

在您的 GitHub 账户中安装 Amplify GitHub 应用程序并生成个人访问令牌后，您可以使用 Amplify CLI 或部署新应用程序。 CloudFormation SDKs使用 `accessToken` 字段指定您在上一步中创建的个人访问令牌。有关更多信息，请参阅 CreateAppAmplify API 参考和《AWS CloudFormation 用户AWS::Amplify::App指南》。

以下 CLI 命令部署了一个使用 GitHub 该应用程序访问仓库的新 Amplify 应用程序。用您自己的信息替换*myapp-using-githubapphttps://github.com/Myaccount/react-app*、和*MY_TOKEN*。

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

使用 Amplify GitHub 应用程序设置网页预览

Web 预览会将向 GitHub 仓库发出的每个拉取请求 (PR) 部署到一个唯一的预览 URL。预览现在使用 Amp GitHub lify 应用程序访问您的存储库。 GitHub 有关安装和授权 GitHub 应用程序进行网页预览的说明，请参阅。为拉取请求启用 Web 预览

AWS Amplify 托管参考

使用本节中的主题查找的详细参考资料 AWS Amplify。

主题

AWS CloudFormation 支持

使用 AWS CloudFormation 模板配置 Amplify 资源，实现可重复且可靠的 Web 应用部署。 AWS CloudFormation 提供了一种通用语言，供您描述和配置云环境中的所有基础设施资源，只需单击几下即可简化跨多个 AWS 账户 and/or 区域的部署。

有关 Amplify 托管，请参阅 Amplify 文档。 CloudFormation有关 Amplify Studio，请参阅 Amp lif y 用户界面生成器文档。 CloudFormation

AWS Command Line Interface 支持

使用通过命令行 AWS Command Line Interface 以编程方式创建 Amplify 应用程序。有关更多信息，请参阅 AWS CLI 文档。

资源标记支持

您可以使用 AWS Command Line Interface 来标记 Amplify 资源。有关更多信息，请参阅 AWS CLI 资源标签文档。

Amplify Hosting API

本参考提供了有关 Amplify Hosting API 操作和数据类型的描述。有关更多信息，请参阅 Amplify API 参考文档。

的文档历史记录 AWS Amplify

下表描述了自上次发布以来对文档所做的重要更改 AWS Amplify。

- 最近文档更新时间：2025 年 9 月 8 日

更改	描述	日期
更新了 Node.js 支持的版本	更新了使用 Amplify Hosting 部署在服务器端渲染的应用程序中有关 Node.js 支持的版本的信息。	2025 年 9 月 8 日
更新了“构建设置和配置”章节	更新了管理 Amplify 应用程序的构建配置章节来介绍新增的可配置构建实例类型功能，使您能够选择实例类型，从而为应用程序提供所需的 CPU、内存和磁盘空间资源。	2025 年 5 月 28 日
更新了“防火墙”章节	更新了Amplify 托管式网站的防火墙支持本章，描述了 Amplify 与集成的正式上市 (GA) AWS WAF，包括正式发布功能和定价结构。	2025 年 3 月 26 日
新增“倾斜保护”章节	增加了Amplify 部署的倾斜保护章节来介绍倾斜保护功能，该功能可消除 Amplify Web 应用程序中客户端与服务器之间的版本倾斜问题。	2025 年 3 月 10 日
更新了 Webhook 章节	增加了适用于 Git 存储库的统一 Webhook主题来介绍统一 Webhook 功能，让您能够使用统一的综合 Webhook 来管理	2025 年 3 月 10 日

更改	描述	日期
	关联到单个 Git 存储库的所有 Amplify 应用程序。	
新增添一个 SSR 计算角色以允许访问 AWS 资源主题	增加了添加 SSR 计算角色以允许访问资源 AWS主题来介绍如何创建 Amplify SSR 计算角色并将其关联到应用程序，从而授予 Amplify Compute 服务访问 AWS 资源的权限。	2025 年 2 月 17 日
全新 “使用 AWS WAF 保护你的 Amplify 应用程序” 章节	添加了Amplify 托管式网站的防火墙支持本章来介绍 Amplify 与 AWS WAF （预览版）的集成，通过该集成，您可以使用 Web 访问控制列表 (Web ACL) 保护您的 Web 应用程序。	2024 年 12 月 18 日
更新了托管策略主题	更新了 AWS 的托管策略 AWS Amplify 主题，以描述 Amplify AWS 托管策略的最新更改。	2024 年 11 月 14 日
更新了“Amplify 对 Next.js 的支持”主题	更新了 Amplify 对 Next.js 的支持主题，来介绍 Amplify 对 Next.js 版本 15 的支持。	2024 年 11 月 6 日
新增将 Amazon S3 存储桶中存储的静态网站部署到 Amplify 章节	添加了“将 Amazon S3 存储桶中的静态网站部署到 Amplify”章节，介绍 Amplify 与 Amazon S3 的全新集成，这让您只需点击几下，即可托管 S3 中存储的静态网站内容。	2024 年 10 月 16 日
新增管理缓存配置章节	添加了“管理应用程序的缓存配置”章节，介绍 Amplify 的默认缓存行为，及其如何将托管缓存策略应用于内容。	2024 年 8 月 13 日

更改	描述	日期
更新了托管策略主题	更新了 AWS 的托管策略 AWS Amplify 主题，以描述 Amplify AWS 托管策略的最新更改。	2024 年 7 月 18 日
更新了托管策略主题	更新了 AWS 的托管策略 AWS Amplify 主题，以描述 Amplify AWS 托管策略的最新更改。	2024 年 5 月 31 日
更新了托管策略主题	更新了 AWS 的托管策略 AWS Amplify 主题，以描述 Amplify AWS 托管策略的最新更改。	2024 年 4 月 17 日
更新了“开始使用”章节	更新了“开始将应用程序部署到 Amplify Hosting”章节，以使用教程中的 Next.js 示例应用程序。	2024 年 4 月 12 日
更新了托管策略主题	更新了 AWS 的托管策略 AWS Amplify 主题，以描述 Amplify AWS 托管策略的最新更改。	2024 年 4 月 5 日
更新了托管策略主题	更新了 AWS 的托管策略 AWS Amplify 主题，以描述 Amplify AWS 托管策略的最新更改。	2024 年 4 月 4 日
新增问题处理章节	添加了“Amplify Hosting 问题排查”章节，介绍如何修复将应用程序部署到 Amplify Hosting 时遇到的问题。	2024 年 4 月 2 日
对自定义 SSL/TLS 证书的新支持	在本连接自定义域章中添加了使用 SSL/TLS 证书主题，描述了将应用程序连接到自定义域时 Amplify 对自定义 SSL/TLS 证书的支持。	2024 年 2 月 20 日

更改	描述	日期
更新了托管策略主题	更新了 AWS 的托管策略 AWS Amplify 主题，以描述 Amplify AWS 托管策略的最新更改。	2024 年 1 月 2 日
对 SSR 框架的全新支持	更新了“使用 Amplify Hosting 部署在服务器端渲染的应用程序”主题，介绍 Amplify 对任何带有开源适配器的基于 JavaScript 的 SSR 框架的支持。	2023 年 11 月 19 日
推出对图像优化功能的全新支持	添加了“SSR 应用程序的图像优化”，介绍服务器端渲染应用程序对图像优化的内置支持。	2023 年 11 月 19 日
更新了托管策略主题	更新了 AWS 的托管策略 AWS Amplify 主题，以描述 Amplify AWS 托管策略的最新更改。	2023 年 11 月 17 日
更新了托管策略主题	更新了 AWS 的托管策略 AWS Amplify 主题，以描述 Amplify AWS 托管策略的最新更改。	2023 年 11 月 6 日
新的通配符子域名主题	添加了 设置通配符子域 主题以描述自定义域名上对通配符子域名的支持。	2023 年 11 月 6 日
新增托管策略	更新了AWS 的托管策略 AWS Amplify主题，描述了 Ampli AmplifyBackendDeployFullAccess AWS fy 的新托管策略。	2023 年 10 月 8 日

更改	描述	日期
对 monorepo 框架的新支持功能上线	更新了 配置 monorepo 构建设置 主题，以描述支持在使用 npm 工作空间、pnpm 工作空间、Yarn 工作空间、Nx 和 Turborepo 创建的 monorepos 中部署应用程序。	2023 年 6 月 19 日
更新了托管策略主题	更新了 AWS 的托管策略 AWS Amplify 主题，以描述 Amplify AWS 托管策略的最新更改。	2023 年 6 月 1 日
更新了托管策略主题	更新了 AWS 的托管策略 AWS Amplify 主题，以描述 Amplify AWS 托管策略的最新更改。	2023 年 2 月 24 日
更新了服务器端渲染章节	更新了 使用 Amplify Hosting 部署在服务器端渲染的应用程序 章节，以描述 Amplify 对 Next.js 版本 12 和 13 的支持的最新变化。	2022 年 11 月 17 日
更新了托管策略主题	更新了 AWS 的托管策略 AWS Amplify 主题，以描述 Amplify AWS 托管策略的最新更改。	2022 年 8 月 30 日
更新了托管策略主题	更新了 为应用程序构建后端 主题，以描述如何使用 Amplify Studio 部署后端。	2022 年 8 月 23 日
更新了托管策略主题	更新了 AWS 的托管策略 AWS Amplify 主题，以描述 Amplify AWS 托管策略的最新更改。	2022 年 4 月 27 日
更新了托管策略主题	更新了 AWS 的托管策略 AWS Amplify 主题，以描述 Amplify AWS 托管策略的最新更改。	2022 年 4 月 17 日

更改	描述	日期
新 GitHub 应用程序功能上线	添加了设置 Amplify 对仓库的访问权限 GitHub主题来描述用于授权 Amplify 访问仓库的新 GitHub 应用程序。 GitHub	2022 年 4 月 5 日
Amplify Studio 新功能上线	更新了 欢迎来到 AWS Amplify 托管 主题，以描述 Amplify Studio 的更新，该更新提供了可视化设计器，用于创建可以连接到后端数据的用户界面组件。	2021 年 12 月 2 日
更新了托管策略主题	更新了 AWS 的托管策略 AWS Amplify 主题，以描述为支持 Amplify Studio 而对 Amplify 的 AWS 托管策略的最新更改。	2021 年 12 月 2 日
更新了托管策略主题	更新了 AWS 的托管策略 AWS Amplify 主题，以描述 Amplify AWS 托管策略的最新更改。	2021 年 11 月 8 日
更新了托管策略主题	更新了 AWS 的托管策略 AWS Amplify 主题，以描述 Amplify AWS 托管策略的最新更改。	2021 年 9 月 27 日
新托管策略主题	添加了该AWS 的托管策略 AWS Amplify主题以描述 Amplify 的 AWS 托管策略以及这些策略的最新更改。	2021 年 7 月 28 日
更新了服务器端渲染章节	更新了 使用 Amplify Hosting 部署在服务器端渲染的应用程序 章节，以描述对 Next.js 版本 10.x.x 和 Next.js 版本 11 的新支持。	2021 年 7 月 22 日

更改	描述	日期
更新了配置构建设置章节	添加了 配置 monorepo 构建设置 主题，以描述在使用 Amplify 部署 monorepo 应用程序时如何配置构建设置和新的 `AMPLIFY_MONOREPO_APP_ROOT` 环境变量。	2021 年 7 月 20 日
更新了功能分支部署章节	添加了 构建时自动生成 Amplify 配置（仅限 Gen 1 应用程序） 主题，以描述如何在构建时自动生成 `aws-exports.js` 文件。添加了 有条件的后端构建（仅限 Gen 1 应用程序） 主题，以描述如何启用有条件的后端构建。添加了 跨应用程序使用 Amplify 后端（仅限 Gen 1 应用程序） 主题，以描述在创建新应用程序、将新分支连接到现有应用程序或更新现有前端以指向其他后端环境时如何重新使用现有后端。	2021 年 6 月 30 日
更新了安全性章节	添加了 Amplify 的数据保护 主题，以描述如何应用责任共担模型以及 Amplify 如何使用加密来保护您的静态和传输中的数据。	2021 年 6 月 3 日
对 SSR 功能的新支持上线	添加了 使用 Amplify Hosting 部署在服务器端渲染的应用程序 章节，介绍 Amplify 对使用服务器端渲染 (SSR) 且使用 Next.js 创建的 Web 应用程序的支持。	2021 年 5 月 18 日

更改	描述	日期
新增安全性章节	添加了 Amplify 的安全保护 章节，描述在使用 Amplify 时如何应用分担责任模型，以及如何配置 Amplify 以满足您的安全和合规目标。	2021 年 3 月 26 日
更新了自定义构建主题	更新了自定义构建映像和实时包更新主题，以描述如何配置托管在 Amazon Elastic Container Registry Public 中的自定义构建映像。	2021 年 3 月 12 日
更新了监控主题	更新了监控主题以描述如何访问 Amazon CloudWatch 指标数据和设置警报。	2021 年 2 月 2 日
新的 CloudTrail 日志主题	添加了 Logging Amplify API 调用使用 AWS CloudTrail主题来描述如何 AWS CloudTrail 捕获和记录 AWS Amplify 控制台 API 参考 AWS Amplify 和管理界面 API 参考的所有 API 操作。	2021 年 2 月 2 日
新的管理用户界面功能上线	更新了 欢迎来到 AWS Amplify 托管 主题，以描述新的管理用户界面，该界面为前端 Web 和移动开发者提供了一个可视化界面，让他们可以在 AWS 管理控制台外部创建和管理应用程序后端。	2020 年 12 月 1 日
新的性能模式功能上线	更新了管理应用程序性能主题，描述了如何启用性能模式进行优化以提高托管性能。	2020 年 11 月 4 日

更改	描述	日期
更新了自定义标头主题	更新了自定义标头主题，描述了如何使用控制台或编辑 YML 文件为 Amplify 应用程序定义自定义标头。	2020 年 10 月 28 日
全新自动子域名功能上线	添加了为 Route 53 自定义域名设置自动子域名主题，以描述如何对连接到 Amazon Route 53 自定义域名的应用程序使用基于模式的功能分支部署。添加了使用子域名进行 Web 预览访问主题，以描述如何将拉取请求中的 Web 预览设置为可通过子域名访问。	2020 年 6 月 20 日
新的通知主题	添加了通知主题，介绍如何为 Amplify 应用程序设置电子邮件通知，以便在构建成功或失败时提醒利益相关者或团队成员。	2020 年 6 月 20 日
更新了自定义域名主题	更新了连接自定义域主题，改进了在 Amazon Route 53 和 Google 域名中添加自定义域名的程序。 GoDaddy此更新还包括有关设置自定义域名的新故障排除信息。	2020 年 5 月 12 日
AWS Amplify 发布	此版本引入了 Amplify。	2018 年 11 月 26 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。