用户指南

AWS Amplify 托管

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Amplify 托管: 用户指南

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Table of Contents

什么是 AWS Amplify Hosting？ 1
Amplify Hosting 功能 1
开始使用 Amplify Hosting 1
Amplify Studio 2
Amplify Studio 功能 2
开始使用 Amplify Studio 2
现代 SPA Web 应用程序 3
入门 4
步骤 1：连接存储库 5
步骤 2a：确认前端的构建设置 7
步骤 2b：确认后端的构建设置 8
步骤 2c：添加环境变量（可选） 10
步骤 3：保存并部署 10
后续步骤 11
开始使用全栈部署 12
先决条件 12
第 1 步：部署前端 12
步骤 2：创建后端 13
第 3 步：将后端连接至前端 15
后续步骤 16
设置功能分支部署 16
在 Amplify Studio 中创建前端用户界面 16
服务器端渲染 (SSR) 17
什么是服务器端渲染 (SSR) 17
SSR 框架集成 17
将 SSR 应用程序部署到 Amplify 18
使用框架适配器 19
使用部署规范 20
部署规范 20
部署 Express 服务器 40
SSR 应用程序的图像优化 46
使用自定义图像加载器 47
框架作者的图像优化集成 47
了解图像优化 API 47

Node.js 版本支持 Next.js 应用程序 52
SSR 部署的问题排查 53
您正使用框架适配器 53
边缘 API 路由会导致您的 Next.js 构建失败 53
按需增量静态再生成不适用于您的应用程序 53
您的应用程序的编译输出超出了允许的最大大小 53
您的构建失败并出现内存不足错误 55
HTTP 响应大小太大 56
Amplify 对 Next.js SSR 的支持 56
Next.js 功能支持 56
Next.js SSR 应用程序的定价 57
使用 Amplify 部署 Next.js SSR 应用程序 58
将 Next.js 11 SSR 应用程序迁移至 Amplify Hosting 计算 60
为静态 Next.js 应用程序添加 SSR 功能 62
令服务器端运行时可以访问环境变量 64
在单一存储库中部署 Next.js 应用程序 66
SSR 应用程序的 Amazon CloudWatch 日志 66
Amplify Next.js 11 SSR 支持 67
设置自定义域 75
了解 DNS 术语和概念 76
DNS 术语 76
DNS 验证 76
Amplify Hosting 自定义域激活流程 77
使用 SSL/TLS 证书 77
添加在 Amazon Route 53 中管理的自定义域 78
添加由第三方 DNS 提供商管理的自定义域 80
添加由管理的自定义域名 GoDaddy 83
添加由 Google Domains 管理的自定义域 85
更新域名的 SSL/TLS 证书 86
管理子域 87
仅添加子域 87
添加多级子域 88
添加或编辑子域 89
通配符子域 90
添加或删除通配符子域 90
为 Amazon Route 53 自定义域设置自动子域 91

带子域的网页预览 92
自定义域问题排查 92
如何验证我的别名记录是否已解析？ 92
我在第三方托管的域卡在了等待验证状态 93
我在 Amazon Route 53 托管的域卡在了等待验证状态 93
我收到 CNAME 错误 AlreadyExistsException 94
我收到需要额外验证错误 95
我在网址上收到 404 错误 CloudFront 95
我在访问我的域时收到 SSL 证书或 HTTPS 错误 96
配置构建设置 97
构建规范的命令和设置 97
特定于分支的构建设置 100
导航到子文件夹 100
使用您的前端部署后端 100
设置输出文件夹 101
在构建过程中安装软件包 101
使用私有 npm 注册表 101
安装操作系统软件包 102
每个构建的键/值存储 102
跳过某个提交的构建 102
禁用自动构建 103
启用或禁用基于 diff 的前端构建和部署 103
启用或禁用基于 diff 的后端构建 104
单一存储库构建设置 104
单一存储库构建规范的 YAML 语法 105
设置 AMPLIFY_MONOREPO_APP_ROOT 环境变量 108
配置 Turborepo 和 pnpm 单一存储库应用程序 110
功能分支部署 111
使用 Amplify 后端环境的团队工作流程 112
功能分支工作流程 112
GitFlow 工作流程 118
每个开发人员的沙盒 119
基于模式的功能分支部署 121
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 112
构建时自动生成 Amplify 配置 123
有条件后端构建 124

跨应用程序使用 Amplify 后端 125
创建新应用程序时重复使用后端 125
将分支连接到现有应用程序时重复使用后端 126
编辑现有前端以指向其他后端 126
手动部署 128
拖放手动部署 128
Amazon S3 或 URL 手动部署 129
Amazon S3 存储桶访问故障排除 130
一键式部署按钮 131
将“部署到 Amplify Hosting”按钮添加到您的存储库或博客 131
设置 GitHub 访问权限 132
为新部署安装和授权 Amplify GitHub 应用程序 132
将现有 OAuth 应用程序迁移到 Amplify GitHub 应用程序 133
为 AWS CloudFormation、CLI 和 SDK 部署设置 Amplify GitHub 应用程序 134
使用 Amplify GitHub 应用程序设置网页预览 135
拉取请求的预览 136
启用 Web 预览 137
使用子域名进行 Web 预览访问 139
端到端测试 140
教程：设置 Cypress 端到端测试 140
向您的现有 Amplify 应用程序添加测试 140
禁用测试 142
使用重定向 143
重定向类型 143
创建和编辑重定向 144
重定向顺序 145
查询参数 145
简单重定向和重写 146
单页 Web 应用程序 (SPA) 的重定向 147
反向代理重写 148
结尾斜杠和清洁 URL 148
占位符 149
查询字符串和路径参数 149
基于区域的重定向 150
重定向和重写中的通配符表达式 150
限制访问 152

环境变量 153
Amplify 环境变量 153
设置环境变量 157
在构建时访问环境变量 158
令服务器端运行时可以访问环境变量 159
使用社交登录的身份验证参数创建新的后端环境 159
前端框架环境变量 160
环境密钥 161
设置环境密钥 161
访问环境密钥 161
Amplify 环境密钥 162
自定义标头 163
自定义标头 YAML 格式 163
设置自定义标头 164
迁移自定义标头 166
Monorepo 自定义标头 167
安全标头示例 167
缓存控制标头示例 168
传入 webhook 169
监控 171
使用监控 CloudWatch 171
指标 171
告警 173
SSR 应用程序的 Amazon CloudWatch 日志 174
访问日志 175
分析访问日志 176
通知 177
电子邮件通知 177
自定义构建 178
自定义构建映像 178
自定义构建映像要求 178
配置自定义构建映像 179
实时程序包更新 179
配置实时程序包更新 180
添加服务角色 182
步骤 1：登录 IAM 控制台 182

步骤 2：创建 Amplify 角色 182
步骤 3：返回到 Amplify 控制台 182
混淆代理问题防范 183
管理应用程序性能 185
开启性能模式 185
使用标头控制缓存时间长度 185
使用 AWS CloudTrail 记录 Amplify API 调用 187
CloudTrail 中的 Amplify 信息 187
了解 Amplify 日志文件条目 188
安全性 191
Identity and Access Management 191
受众 192
使用身份进行身份验证 192
使用策略管理访问 195
Amplify 如何与 IAM 协同工作 197
基于身份的策略示例 203
AWS 托管式策略 205
故障排除 214
数据保护 216
静态加密 216
传输中加密 217
加密密钥管理 217
合规性验证 217
基础设施安全性 218
日记账记录和监控 218
防止跨服务混淆座席 219
安全最佳实操 221
在 Amplify 默认域中使用 Cookie 221
配额 222
故障排除 224
一般性问题 224
HTTP 429 状态码（请求太多） 224
AL2023 构建镜像 225
如何使用 Python 运行时运行 Amplify 函数 225
如何运行需要超级用户或 root 权限的命令 226
自定义域 226

服务器端渲染 (SSR) 226
AWS Amplify Hosting 参考 227
AWS CloudFormation 支持 227
AWS Command Line Interface 支持 227
资源标记支持 227
Amplify Hosting API 227
文档历史记录 228
..... ccxxxv

欢迎来到 AWS Amplify Hosting

AWS Amplify 是一组专门构建的工具和功能，使前端 Web 和移动开发人员能够快速轻松地在 AWS 上构建全栈应用程序。Amplify 提供两项服务：Amplify Hosting 和 Amplify Studio。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。这份用户指南提供开始使用 Amplify Hosting 所需的信息。

Amplify Hosting 功能

- Amplify Hosting 支持常见的 SPA 框架，例如 React、Angular、Vue.js、Ionic 和 Ember，以及盖茨比、Eleventy、Hugo 和 Jekyll 等静态网站生成器。 VuePress
- 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。请参阅，功能分支部署。
- 将您的应用程序连接到自定义域。请参阅，设置自定义域名。
- 部署和托管 SSR 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。

  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 所期望的目录结构。适配器可用于将 Nuxt 应用程序部署到 Amplify。
- 通过设置拉取请求预览来预览代码审查期间的更改。
- 通过端到端测试提高应用程序质量。请参阅，end-to-end 测试。
- 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。请参阅，限制访问。
- 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。请参阅，使用重定向。
- 通过确保仅在整个部署完成后更新 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 的托管功能，请参阅 现有代码入门 教程。完成本教程后，您将能够连接 git 存储库（ GitHub GitLab、 BitBucket Cloud 和AWS CodeCommit ）以设置持续部署。或者，您可以开始使用一个全栈持续部署示例。

Amplify Studio

您可以从 AWS Management Console 中的 AWS Amplify 控制台访问 Amplify Studio。Amplify Studio 是一个直观的开发环境，可简化可扩展的全栈 Web 和移动应用程序的创建。使用 Studio 使用一组界面组件构建您的前端 ready-to-use 用户界面，创建应用程序后端，然后将两者连接在一起。请参阅Amplify 文档中的 Amplify Studio 用户指南。

Amplify Studio 功能

- 可视化数据建模让您能够专注于特定领域的对象，而不是云基础架构。
- 为您的应用程序设置身份验证。
- 功能强大且易于理解的授权。
- 我nfrastructure-as-code 使用配置了所有后端功能。AWS CloudFormation
- 与 Amplify Command Line Interface (CLI) 配合使用。您在 Studio 中所做的所有更新都可以导入到 CLI 中。
- 通过电子邮件邀请用户配置和管理后端。这些用户还可以使用他们的电子邮件登录 Amplify CLI。
- 支持降价的内容管理。
- 管理应用程序的用户和群组。
- 使用 Studio 的视觉设计器来构建前端用户界面 (UI) 组件。从预先构建的用户界面组件库中的数十种设计中进行选择。
- 将设计师构建的 Figma 原型作为 React 代码导入 Studio。
- 使用主题自定义前端用户界面，将全局样式应用于应用程序的组件。
- 直接在 Studio 中配置和测试您的用户界面组件，以了解它们如何更新和显示数据。
- 只需几个简单的步骤，即可将云端连接的后端绑定到前端用户界面。

开始使用 Amplify Studio

您无需 AWS 账户即可开始使用 Studio 创建后端。如果没有 AWS 账户，您就可以在本地开始为后端建模数据。

如果有 AWS 账户，您可以访问一组扩展的 Studio 功能来管理您的后端环境，也可以使用可视化设计器来创建可以连接到应用程序后端的前端用户界面组件。有关更多信息，请参阅 Amplify 文档中的入门。

现代 SPA Web 应用程序

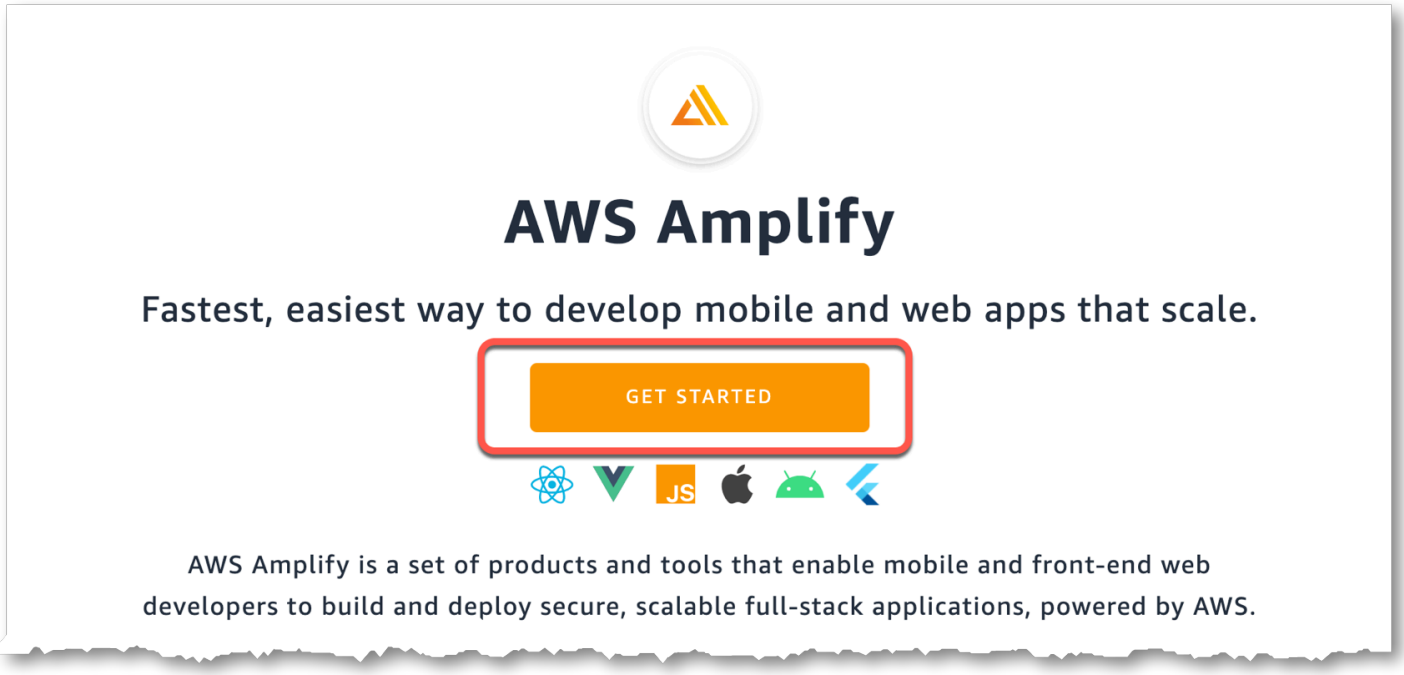
本用户指南适用于对现代单页 Web 应用程序 (SPA) 有基本了解的客户。现代 Web 应用程序构造为 SPA，将所有应用程序组件打包成静态文件。传统的客户端-服务器 Web 架构导致糟糕的体验；每次单击按钮或搜索都需要往返一次服务器，以重新呈现整个应用程序。现代 Web 应用程序通过将应用程序前端或用户界面作为预先构建的 HTML/ JavaScript 文件高效地提供给浏览器，然后无需重新加载页面即可调用后端功能，从而提供类似于原生应用程序的用户体验。

现代 Web 应用程序功能通常分布在多个位置，例如数据库、身份验证服务、在浏览器中运行的前端代码和后端业务逻辑或在云中运行的 AWS Lambda 函数。这使应用程序部署变得复杂且耗时，因为开发人员需要仔细协调前端和后端中的部署以避免部署了一部分或部署失败。Amplify 通过单个工作流简化了前端和后端的部署。

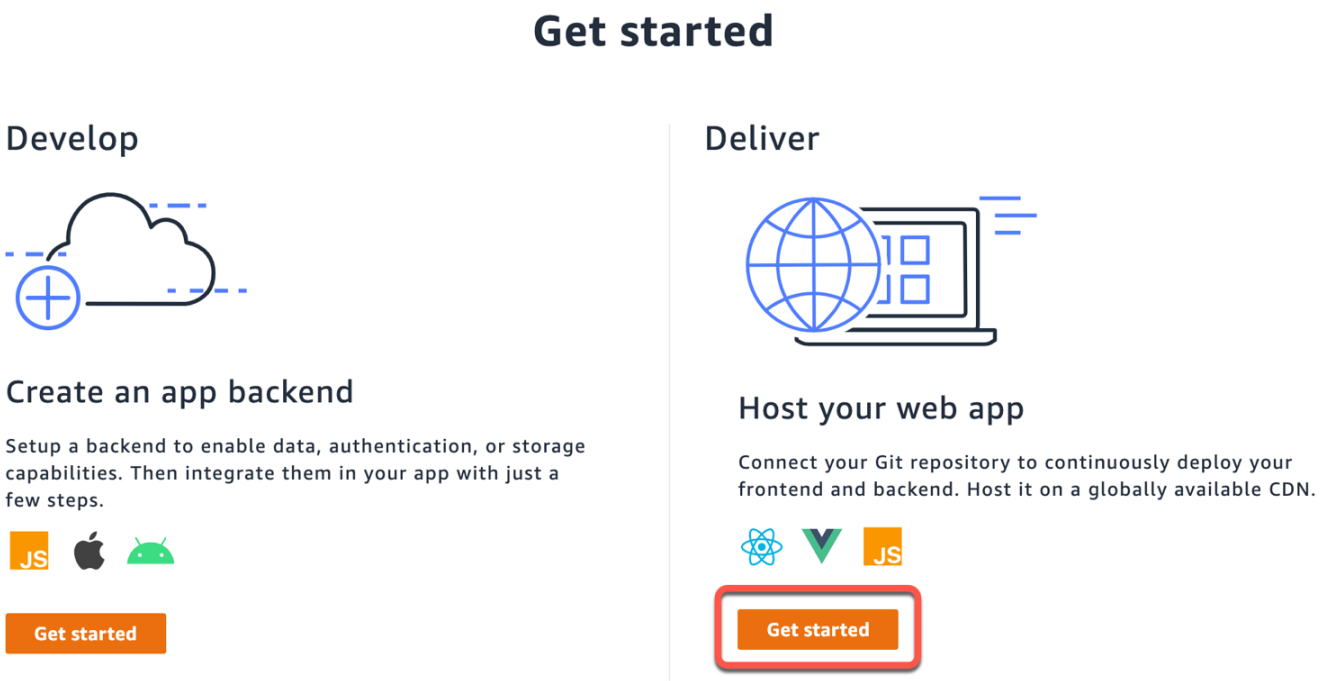
现有代码入门

在本演练中，您将了解如何持续构建、部署和托管现代 Web 应用程序。现代 Web 应用程序包括单页应用程序 (SPA) 框架（例如，React、Angular 或 Vue）和静态站点生成器 (SSG)（例如，Hugo、Jekyll 或 Gatsby）。Amplify Hosting 还支持使用服务器端渲染 (SSR) 且使用 Next.js 创建的 Web 应用程序。

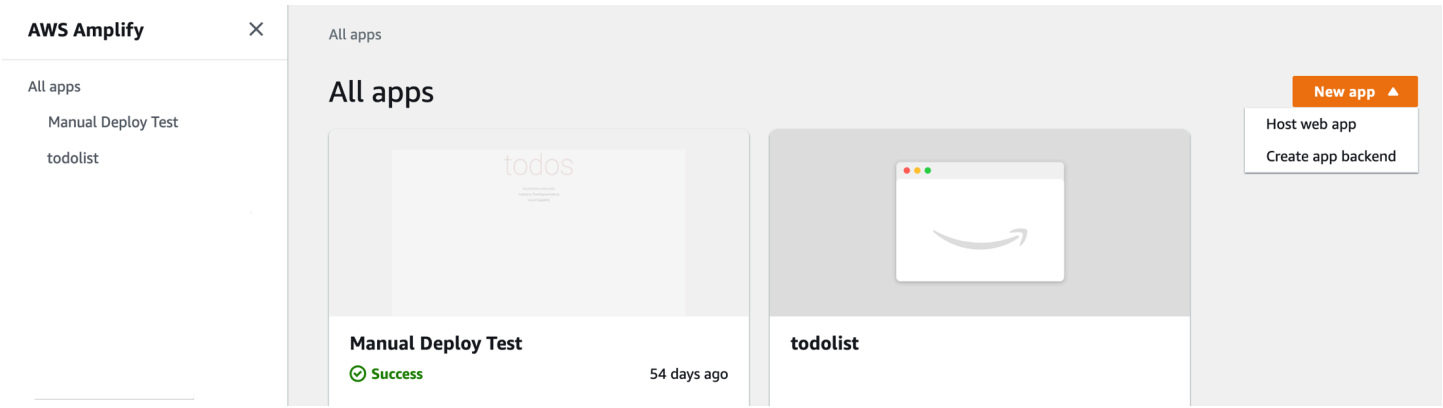
要开始使用，请登录 Amplify 控制台。如果从 AWS Amplify 主页开始，请选择页面顶部的入门。



然后选择交付下面的入门”。

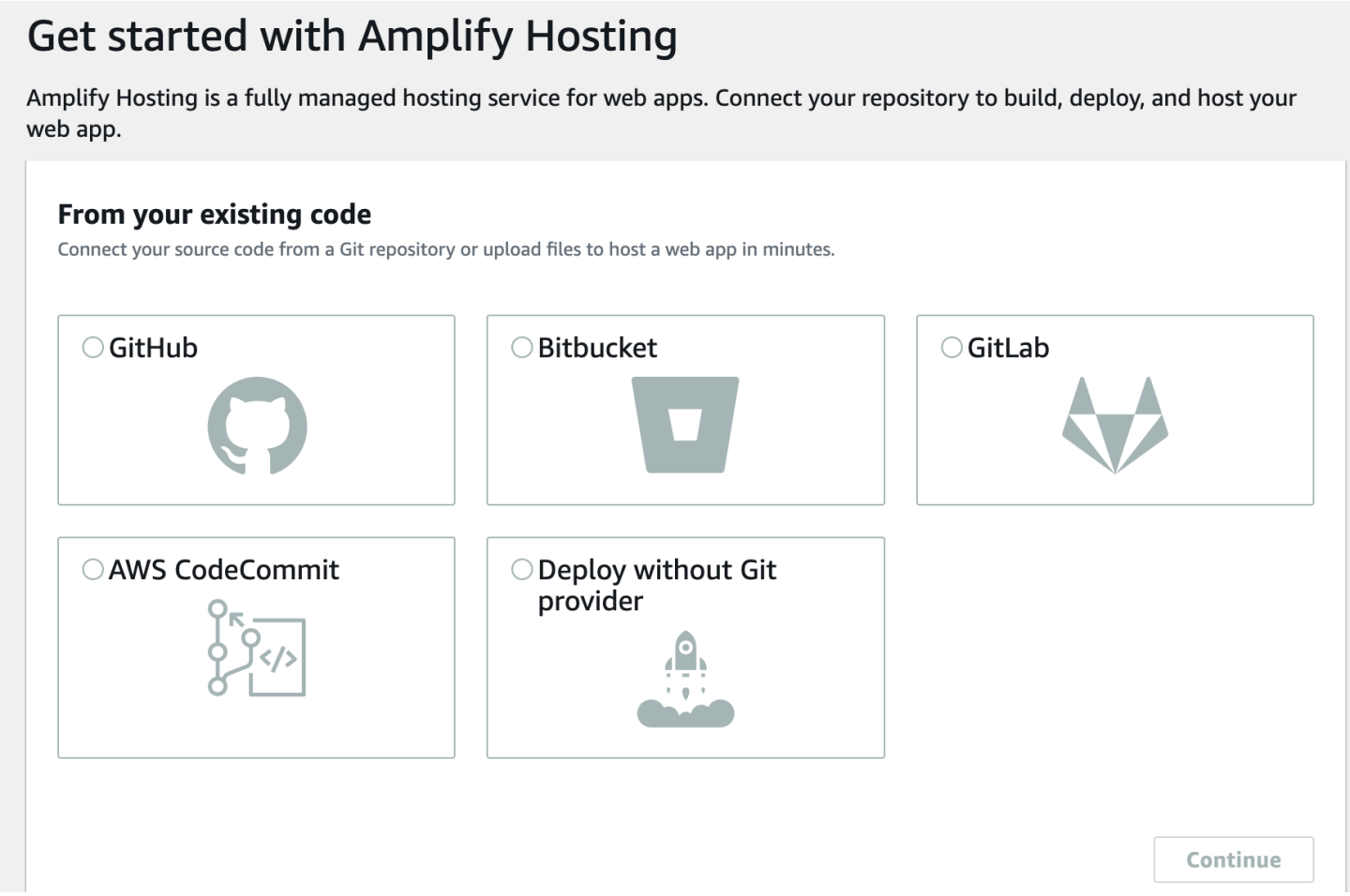


如果所有应用程序页面开始，请选择新建应用程序，然后选择右上角的托管 Web 应用程序。



步骤 1：连接存储库

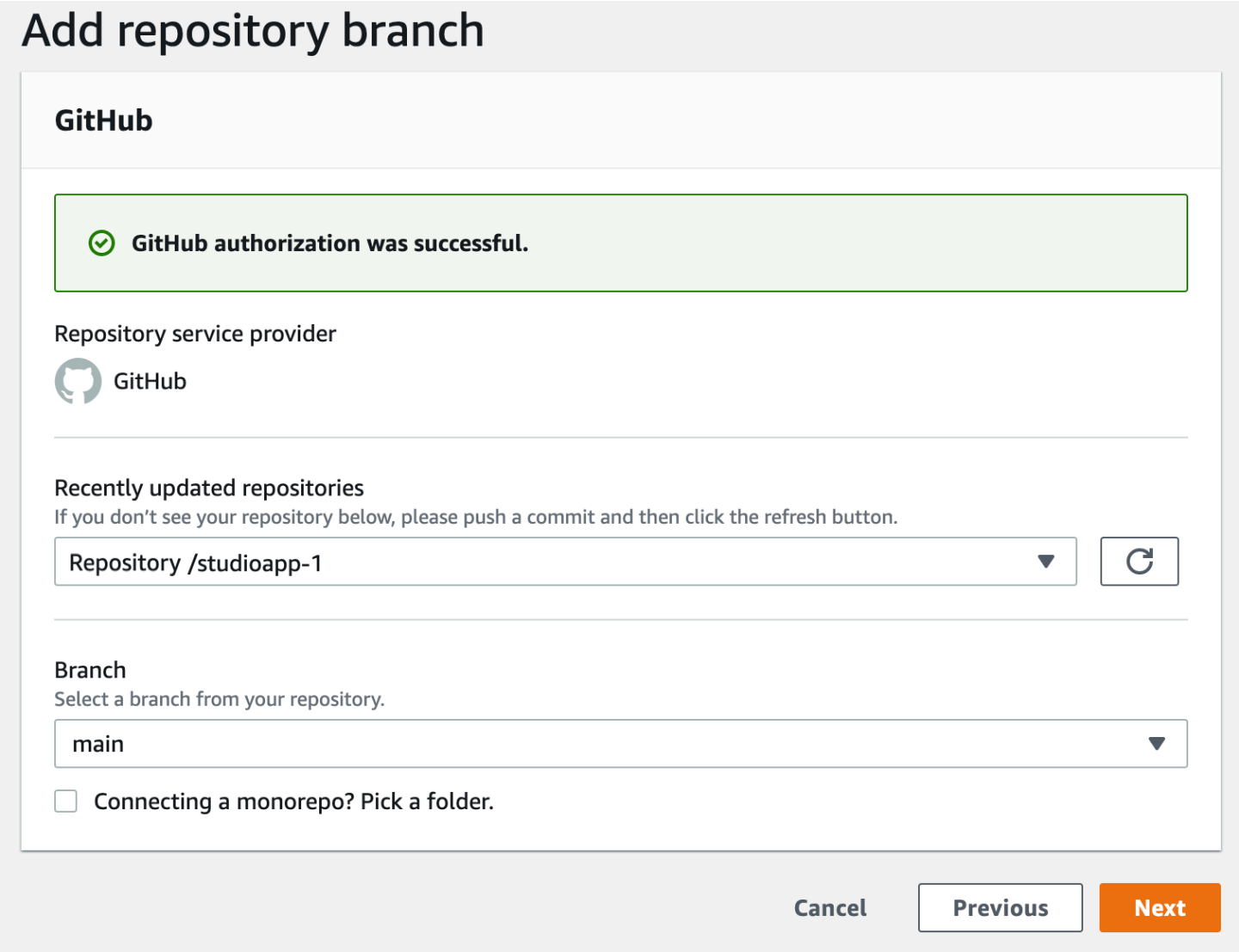
连接您的 GitHub、Bitbucket、GitLab 或 AWS CodeCommit 存储库。您还可以选择手动上传构建构件而无需连接 Git 存储库。有关更多信息，请参阅手动部署。



使用 Bitbucket、GitLab 或 AWS CodeCommit 授权 Amplify 控制台后，Amplify 会从存储库提供商处获取访问令牌，但不会将令牌存储在 AWS 服务器上。Amplify 仅使用安装在特定存储库中的部署密钥访问存储库。

对于 GitHub 存储库，Amplify 现在使用 GitHub 应用程序功能来授权访问 Amplify。使用 Amplify GitHub 应用程序，就可以对权限进行更精细的调整，使您能够授予 Amplify 仅访问指定存储库的权限。有关安装和授权 GitHub 应用程序的更多信息，请参阅 设置 Amplify 对 GitHub 存储库的访问权限。

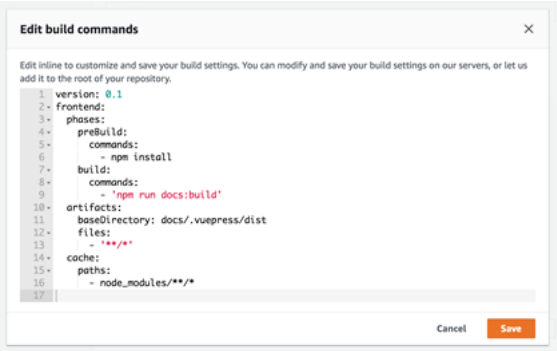
在您连接存储库服务提供商后，选择存储库，然后选择要构建并部署的对应分支。



步骤 2a：确认前端的构建设置

对于所选分支，Amplify 将检查您的存储库以自动检测要运行的构建命令序列。

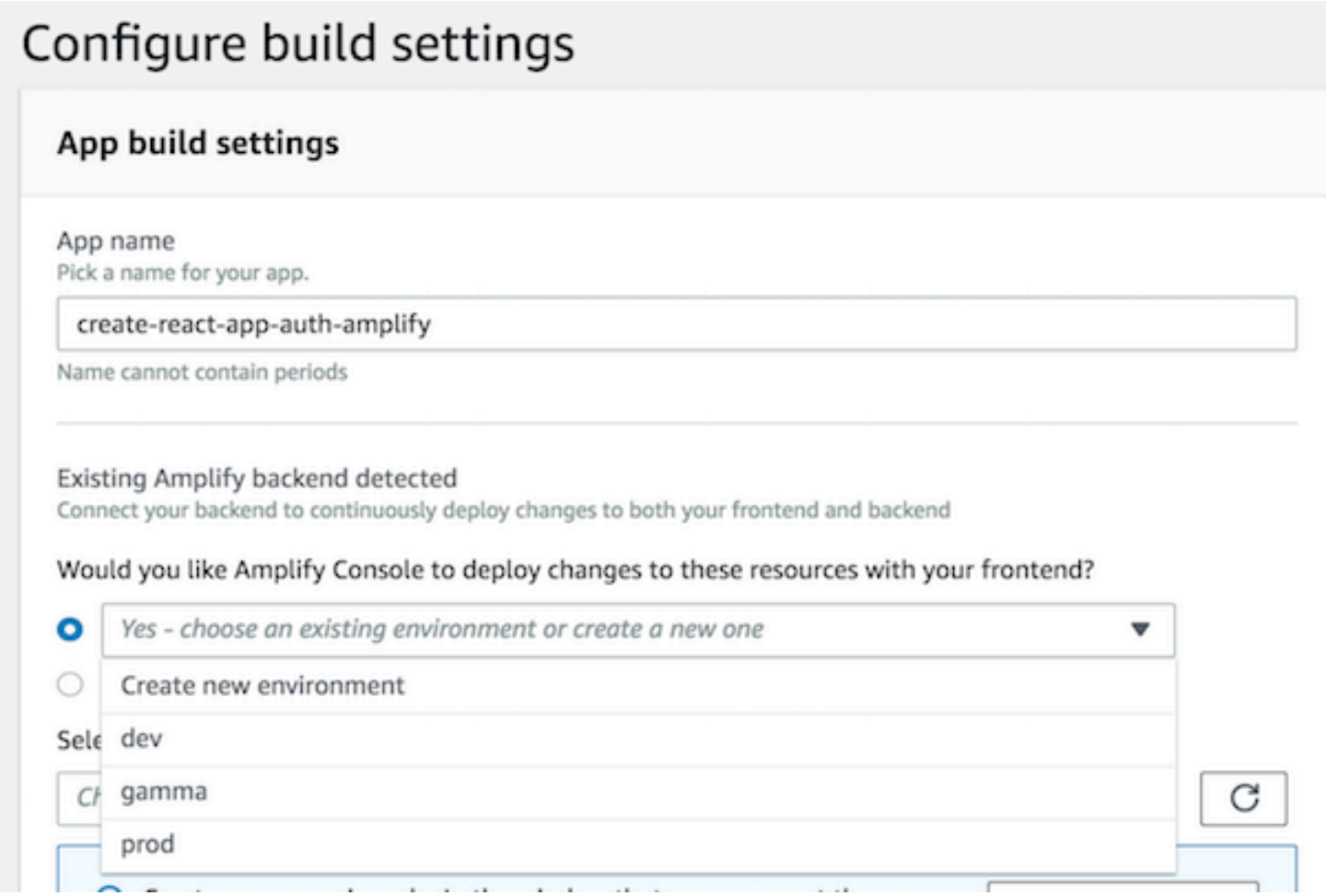
重要提示：验证构建命令和构建输出目录（即 artifacts > baseDirectory）是否准确。如果您需要修改此信息，请选择 Edit (编辑) 以打开 YML 编辑器。您可以将构建设置保存在我们的服务器上，也可以下载 YML 并将其添加到您的存储库的根位置（对于单一存储库，将 YML 存储在应用程序的根目录中）。



有关更多信息，请参阅构建规范 YAML 语法。

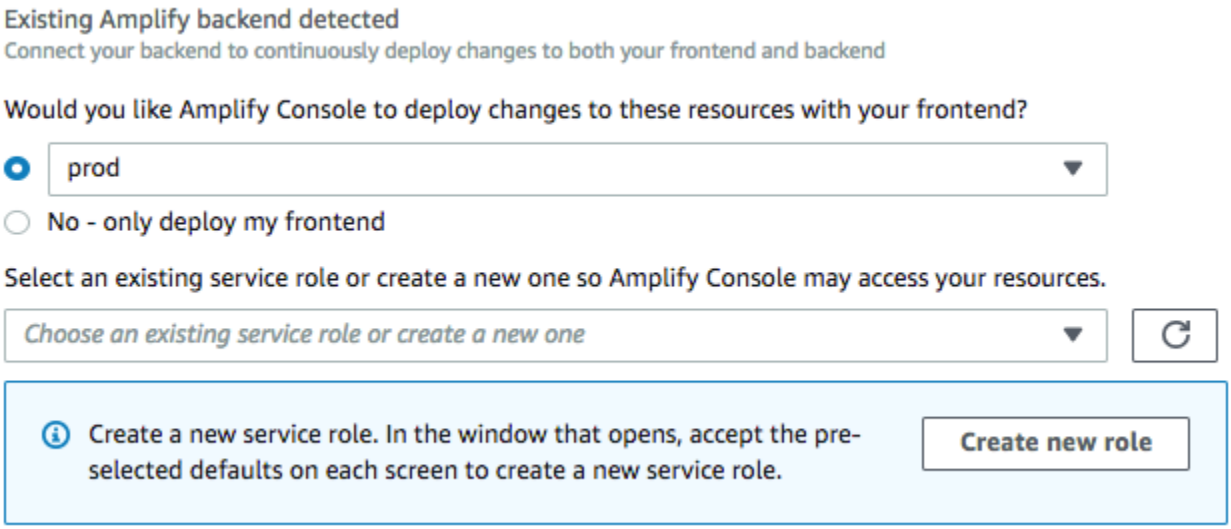
步骤 2b：确认后端的构建设置

如果您连接了 Amplify CLI v1.0+ 预配置的存储库（运行 amplify -v 可查找 CLI 版本），则 Amplify Hosting 将使用前端构建在单个工作流程中部署或自动更新后端资源（由 Amplify CLI 预配置的任何资源）。您可以选择将现有后端环境指向分支，或者创建一个全新的环境。有关分步教程，请参阅全栈部署入门。



要在构建过程中使用 Amplify CLI 部署后端功能，请创建或重用 AWS Identity and Access Management (IAM) 服务角色。IAM 角色是向 Amplify 授予处理您账户中的资源的权限的安全方式。有关详细说明，请参阅 添加服务角色。

注意：如果没有启用 IAM 服务角色，则 Amplify CLI 不会运行。



步骤 2c：添加环境变量（可选）

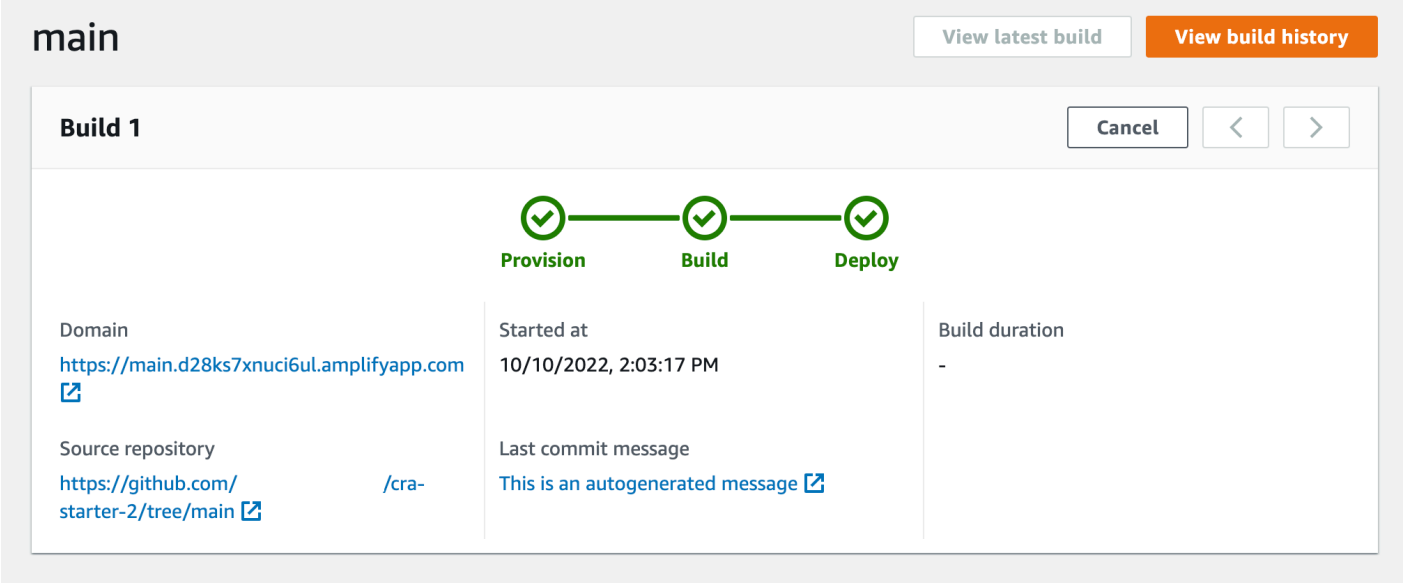
几乎每个应用程序在运行时都需要获取配置信息。这些配置可以是数据库连接详细信息、API 密钥或其他参数。环境变量提供了在构建时公开这些配置的方法。

步骤 3：保存并部署

复查您的所有设置以确保正确设置了所有内容。选择 保存并部署 以将您的 Web 应用程序部署到 AWS 全球内容分发网络 (CDN)。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。

通过选择分支部分的进度指示器来访问构建日志屏幕。构建具有以下阶段：

1. 预置 – 在具有 4 个 vCPU 和 7GB 内存的主机上使用 Docker 镜像设置您的构建环境。每个构建都获得自己的主机实例，从而确保所有资源都安全地隔离。将显示 Docker 文件的内容以确保默认镜像支持您的要求。
2. 构建 – 构建过程包括三个阶段：设置（将存储库克隆到容器中）、部署后端（运行 Amplify CLI 来部署后端资源）和构建前端（构建您的前端项目）。
3. 部署：当构建完成后，所有项目将部署到由 Amplify Hosting 管理的托管环境。您可以在 `amplifyapp.com` 域上查看您的应用程序。每个部署都是原子部署 – 通过确保仅在整个部署完成后更新 Web 应用程序，原子部署消除了维护时段。



> **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。

后续步骤

- 将自定义域添加到您的应用程序
- 管理多个环境
- 合并前预览拉取请求

开始使用全栈持续部署

Amplify Hosting 使开发人员可以使用 Amplify Framework 构建应用程序，在每次提交代码时向其后端和前端持续部署更新。借助 Amplify Hosting，您可以在与前端代码相同的提交中部署使用 Amplify Studio 创建的 GraphQL/REST API、身份验证、分析和存储的无服务器后端。

在本教程中，您将设置使用 Amplify 的全栈 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。

主题

- 先决条件
- 第 1 步：部署前端
- 步骤 2：创建后端
- 第 3 步：将后端连接至前端
- 后续步骤

先决条件

在开始本教程之前，您需要执行以下操作：

- 注册 AWS 账户。打开 https://portal.aws.amazon.com/billing/signup#/start/email 以便开始。
- 使用 git 存储库提供商（例如 Bitbucket GitHub GitLab、或 AWS CodeCommit）创建账户。
- 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。

如果你需要创建一个新的前端应用程序来用于此示例，你可以按照创建 React App 文档中的创建 React Ap p 说明进行操作。

部署前端应用程序

1. 登录 AWS Management Console 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 GitHub 存储库的访问权限。
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**
>
> 为了增强您的 Amplify 应用程序的安全性，amplifyapp.com 域名已在公共后缀列表 (PSL) 中注册。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 Management Console 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。

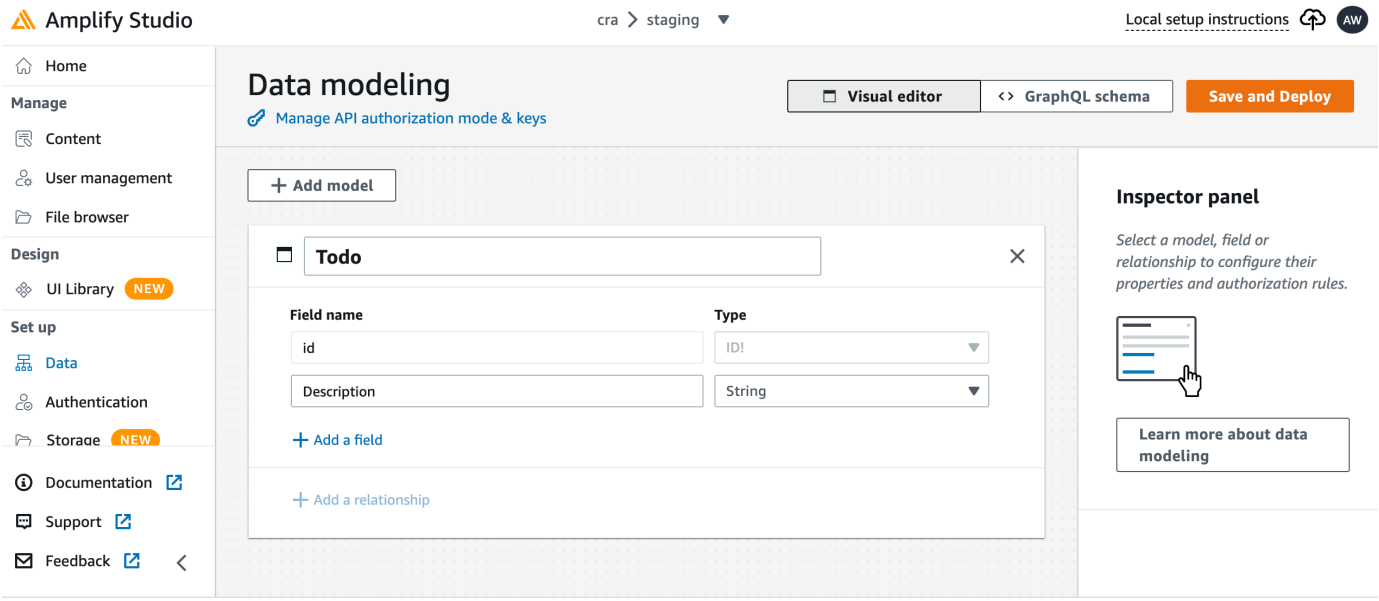
Amplify Studio 是一个可视化界面，用于创建和管理您的后端并加快前端用户界面的开发。有关 Amplify Studio 的更多信息，请参阅 Amplify Studio 文档。

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 AWS 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。

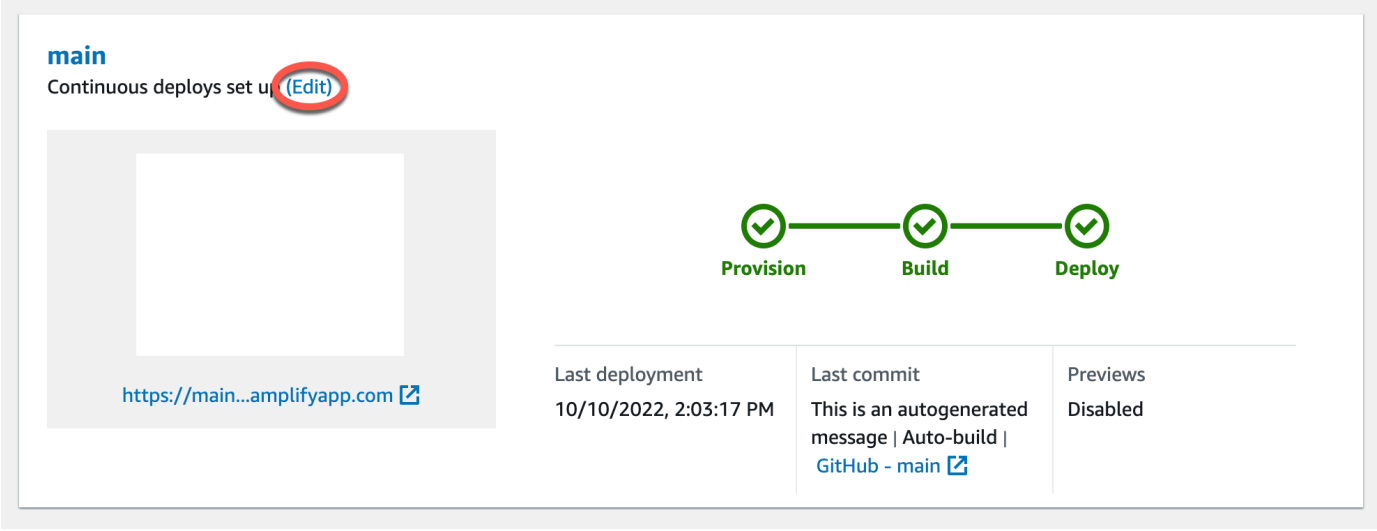
```
amplify pull --appId abcd1234 --envName staging
```

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的全栈 CI/CD。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，无需修改您的后端环境。

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。

使用 Amplify Hosting 部署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 Management Console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 所期望的目录结构。

要了解 Amplify 如何支持 SSR，请查看以下主题。

主题

- 什么是服务器端渲染 (SSR)
- 对 SSR 框架的 Amplify 支持
- 使用 Amplify Hosting 部署规范配置构建输出
- SSR 应用程序的图像优化
- Node.js 版本支持 Next.js 应用程序
- SSR 部署的问题排查
- Amplify 对 Next.js SSR 的支持

什么是服务器端渲染 (SSR)

Amplify 支持部署和托管使用单页应用程序（SPA）框架（例如 React）创建的静态 Web 应用程序，以及使用静态网站生成器（SSG）（例如 Gatsby）创建的应用程序。静态 Web 应用程序由存储在内容分发网络 (CDN) 上的 JavaScript 文件（例如 HTML、CSS 和文件）组合而成。当客户端浏览器向网站发出请求时，服务器会向客户端返回一个带有 HTTP 响应的页面，客户端浏览器会解释内容并将其显示给用户。

Amplify 还支持采用服务器端渲染（SSR）的 Web 应用程序。当客户端向 SSR 页面发送请求时，会在服务器上为每个请求创建该页面的 HTML。SSR 使开发人员能够根据请求和用户自定义网站。此外，SSR 可以增强网站性能和搜索引擎优化 (SEO)。

对 SSR 框架的 Amplify 支持

Amplify Hosting 支持任何 JavaScript 基于 SSR 框架，其部署包符合 Amplify 期望的构建输出。Amplify Hosting 提供了部署规范，该规范对任何 SSR 框架的应用程序构建输出文件和目录结构进行了标准化。

框架作者可以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。

如果您没有使用框架或框架适配器，则可以开发自己的解决方案来生成符合 Amplify Hosting 预期目录结构的部署包。

Amplify Hosting 支持以下基元：静态资产、计算、图像优化和路由规则。您可以利用这些基元来部署功能更丰富的应用程序。有关每个基元的详细信息，请参阅 Amplify SSR 基元支持。

您可以从以下场景中进行选择，开始将 SSR 应用程序部署到 Amplify。

部署 Next.js 应用程序

Amplify 支持使用 Next.js 创建的应用程序，无需在控制台中使用适配器或手动配置。有关更多信息，请参阅Amplify 对 Next.js SSR 的支持。

部署使用框架适配器的应用程序

您可以参考任何可用的开源框架适配器，将 SSR 应用程序部署到 Amplify Hosting。有关更多信息，请参阅使用框架适配器。

Nuxt 框架有一个适配器可用。有关使用此适配器的更多信息，请参阅 Nuxt 文档。

构建框架适配器

想要集成框架提供的功能的框架作者可以使用 Amplify Hosting 部署规范来配置您的构建输出，使其符合 Amplify 期望的结构。有关更多信息，请参阅使用部署清单部署 Express 服务器。

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。

将 SSR 应用程序部署到 Amplify

您可以使用本主题中的说明来部署使用任何框架创建的应用程序，该框架具有符合 Amplify 期望的构建输出的部署包。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，请参阅使用框架适配器。

将 SSR 应用程序部署到 Amplify Hosting

1. 登录 AWS Management Console 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. 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. 在构建设置页面上，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。在服务器端渲染（SSR）部署部分，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。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角色。
   - 允许 Amplify 自动创建角色并将其附加到您的应用程序
     - 在 IAM 角色部分，选择创建并使用新的服务角色。
   - 附加您之前创建的服务角色
     a. 在 IAM 角色部分，选择使用已有服务角色。
     b. 从列表中选择要使用的角色。
7. 选择下一步。
8. 在查看页面上，选择部署。

使用框架适配器

您可以安装和使用为与 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

使用 Amplify Hosting 部署规范配置构建输出

使用 Amplify 部署规范来配置要与 Amplify Hosting 集成的 SSR 框架的构建输出。如果您是框架作者，则可以使用部署规范来了解如何构建 Amplify 期望的构建输出。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。

Amplify Hosting 部署规范

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。

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。每个计算资源包的压缩大小不能超过 220 MB。例如，压缩后的 .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，请参阅使用重定向

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。

键	Type	必需	描述
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。 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

键	Type	必需	描述
			匹配所有传入的请求。
target	目标	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 。
回退	目标	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。

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

键	Type	必需	描述
			Amplify Hosting 客户标头的更多信息，请参阅自定义标头。 **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 = 'nodejs16.x' | 'nodejs18.x' | 'nodejs20.x';
```

以下列表描述了 ComputeResource 对象的属性。

键	Type	必需	描述
name	字符串	是	指定计算资源的名称。名称必须与 .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。
runtime	ComputeRuntime	是	定义预置计算资源的运行时。 有效值包括 nodejs16.

键	Type	必需	描述
			x 、nodejs18.x 和 nodejs20.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": "nodejs16.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。

键	Type	必需	描述
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。
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 输入图像网址。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?: 'http' | 'https';
  hostname: string;
  port?: string;
  pathname?: string;
}
```

以下列表描述了 RemotePattern 对象的属性。

键	Type	必需	描述
protocol	字符串	否	允许的远程模式的协议。 有效值为 http 或 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 `**` 的情况下，Amplify 不允许使用笼统通配符。
port	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。

键	Type	必需	描述
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": "nodejs18.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": "nodejs18.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": "nodejs18.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。

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。如果您正在实施自定义加载器，请参阅使用自定义图像加载器。

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。

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 Amplify Hosting 部署规范。

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。
- HTTP 或 HTTPS 是唯一支持使用远程 URL 获取图像的协议。

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	描述	示例
url	字符串	是	源图像的相对路径或绝对 URL。对于远程 URL，支持 http 和 https 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`
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`

查询参数	Type	必需	描述	示例
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`
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`

查询参数	Type	必需	描述	示例
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`
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`

查询参数	Type	必需	描述	示例
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`
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。

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。

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。

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，请参阅实时程序包更新。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 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

- 您正使用框架适配器
- 边缘 API 路由会导致您的 Next.js 构建失败
- 按需增量静态再生成不适用于您的应用程序
- 您的应用程序的编译输出超出了允许的最大大小
- 您的构建失败并出现内存不足错误
- HTTP 响应大小太大

您正使用框架适配器

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 对 SSR 框架的 Amplify 支持。

边缘 API 路由会导致您的 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 API 和中间件。

按需增量静态再生成不适用于您的应用程序

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。

您的应用程序的编译输出超出了允许的最大大小

目前，Amplify 为 SSR 应用程序支持的最大构建输出大小为 220 MB。如果您收到一条错误消息，指出您的应用程序编译输出的大小超过了允许的最大大小，则必须采取措施缩小该大小。

要缩小应用程序编译输出的大小，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，将构建工件下载到您的本地计算机。然后，检查目录的大小。例如，该node_modules目录可能包含由 Next.js 服务器运行时文件引@esbuild用的二进制文件，例如@swc和。由于运行时不需要这些二进制文件，因此可以在构建后将其删除。

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 应用程序的编译输出

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 更改为您自己的信息。对于任务 ID，请使用您正在调查的失败版本的内部版本号。

```
aws amplify get-job --app-id abcd1234 --branch-name main --job-id 2
```

2. 在终端输出中，在、、stepName: "BUILD"部分中找到预签名的构件 URL。job steps在以下示例输出中，URL 以红色突出显示。

```
"job": {
    "summary": {
        "jobArn": "arn:aws:amplify:us-west-2:111122223333:apps/abcd1234/main/
jobs/0000000002",
        "jobId": "2",
        "commitId": "HEAD",
        "commitTime": "2024-02-08T21:54:42.398000+00:00",
        "startTime": "2024-02-08T21:54:42.674000+00:00",
        "status": "SUCCEED",
        "endTime": "2024-02-08T22:03:58.071000+00:00"
    },
    "steps": [
        {
            "stepName": "BUILD",
            "startTime": "2024-02-08T21:54:42.693000+00:00",
            "status": "SUCCEED",
            "endTime": "2024-02-08T22:03:30.897000+00:00",
            "logUrl": "https://aws-amplify-prod-us-west-2-artifacts.s3.us-
west-2.amazonaws.com/abcd1234/main/0000000002/BUILD/log.txt?X-Amz-Security-
Token=IQoJb3JpZ2luX2V...Example
```

3. 将 URL 复制并粘贴到浏览器窗口中。artifacts.zip文件已下载到您的本地计算机。这是你的编译输出。

4. 运行du磁盘使用命令以检查目录的大小。以下示例命令返回compute和static目录的大小。

```
du -csh compute static
```

以下是输出示例，其中包含compute和static目录的大小信息。

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

5. 打开该compute目录，然后找到该node_modules文件夹。查看您的依赖关系，寻找可以更新或删除的文件，以减小文件夹的大小。
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。

```
- rm -f node_modules/@swc/core-linux-x64-gnu/swc.linux-x64-gnu.node
- rm -f node_modules/@swc/core-linux-x64-musl/swc.linux-x64-musl.node
```

以下是amplify.yml文件的构建命令部分的示例，其中包含在运行生产版本后添加的这些命令。

```
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
```

您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。

HTTP 响应大小太大

目前，Amplify 支持使用 Web 计算平台的 Next.js 12 和 13 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。

Amplify 对 Next.js SSR 的支持

Amplify 仅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 SPA 的 React 框架。 JavaScript您可以部署使用 Next.js 13 构建的具有图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

Amplify Hosting 计算可完全管理使用 Next.js 12 和 13 构建的应用程序的服务器端渲染（SSR）。如果您在 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
- 中间件
- 环境变量
- 图像优化
- Next.js 13 应用程序目录

不支持的 特征

- 边缘 API 路由（不支持边缘中间件）
- 按需增量静态再生成 (ISR)
- 国际化 (i18n) 自动区域检测
- Next.js 直播
- 在静态资产和经过优化的图像上运行中间件

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。

Next.js SSR 应用程序的定价

在部署 Next.js 12 或更高版本的 SSR 应用程序时，Amplify Hosting 计算会为您管理部署 SSR 应用程序所需的资源。有关 Amplify Hosting 计算费用的信息，请参阅 AWS Amplify 定价。

使用 Amplify 部署 Next.js SSR 应用程序

默认情况下，Amplify 使用 Amplify Hosting 的计算服务部署新的 SSR 应用程序，该服务支持 Next.js 12 和 13。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。

按照以下说明部署新的 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 Management Console 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. 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
     - 在 IAM 角色部分，选择创建并使用新的服务角色。
   - 附加您之前创建的服务角色
     a. 在 IAM 角色部分，选择使用已有服务角色。
     b. 从列表中选择要使用的角色。
6. 选择下一步。
7. 在查看页面上，选择部署。

package.json 文件设置

当部署 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，以检测该应用程序是 SSR 还是 SSG。

以下是 Next.js SSR 应用程序构建脚本的示例。构建脚本 `"next build"` 表示该应用程序同时支持 SSG 和 SSR 页面。

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

以下是 Next.js SSG 应用程序构建脚本的示例。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 构建设置

在检查应用程序的 `package.json` 文件以确定您部署的是 SSG 还是 SSR 应用程序后，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/**/*
```

如果 Amplify 检测到您部署的是 SSG 应用程序，则它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`。

以下是将 `baseDirectory` 设置为 `out` 的应用程序的构建设置示例。这表明构建构件适用于仅支持 SSG 页面的 Next.js 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/**/*
```

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 13。

Amplify 控制台会检测您账户中在 Amplify Hosting 计算服务发布之前部署的应用程序，该服务完全支持 Next.js 12 和 13。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。

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 13 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 Management Console 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 13。

3. 在导航窗格中的应用程序设置下，选择常规。
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 13 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 13，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`。对于通过由 Amplify Hosting 计算托管的 SSR 部署到 Next.js 12 或 13 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 Management Console 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. 在导航窗格中的应用程序设置下，选择常规。
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。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 DB_HOST -e DB_USER -e DB_PASS >> .env.production
        - env | grep -e NEXT_PUBLIC_ >> .env.production
        - npm run build
  artifacts:
    baseDirectory: .next
    files:
      - '**/*'
  cache:
    paths:
      - node_modules/**/*
      - .next/cache/**/*
```

在此示例中，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 DB_HOST -e DB_USER -e DB_PASS >> .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。

```
DB_HOST=localhost
```

```
DB_USER=myuser
DB_PASS=mypassword
```

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_SEARCH_KEY=asdfiojslf
NEXT_PUBLIC_SEARCH_ENDPOINT=https://search-url
```

monorepos 的 SSR 环境变量

如果您要在 monorepo 中部署 SSR 应用程序，并希望让 Next.js 可以访问特定的环境变量，则必须在.env.production文件前面加上应用程序根目录。以下 Nx monorepo 中 Next.js 应用程序的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
applications:
  - frontend:
      phases:
        preBuild:
          commands:
            - npm ci
        build:
          commands:
            - env | grep -e DB_HOST -e DB_USER -e DB_PASS >> apps/app/.env.production
            - env | grep -e NEXT_PUBLIC_ >> apps/app/.env.production
            - npx nx build app
      artifacts:
        baseDirectory: dist/apps/app/.next
        files:
          - '**/*'
      cache:
        paths:
```

```
          - node_modules/**/*
      buildPath: /
    appRoot: apps/app
```

前面示例中 “构建命令” 部分的以下几行演示了如何从构建环境中获取特定变量，并将其添加到具有应用程序根目录的 monorepo 中应用程序`.env.production`的文件中。`apps/app`

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

在单一存储库中部署 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。请注意，pnpm 和 Turborepo 应用程序需要额外的配置。有关更多信息，请参阅单一存储库构建设置。

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。

SSR 应用程序的 Amazon CloudWatch 日志

Amplify 将有关你的 Next.js 运行时的信息发送到你的 Amazon CloudWatch Logs。 AWS 账户部署 SSR 应用程序时，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担任该角色。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 特征

- 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。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。有关如何创建客户托管式策略的说明，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:ListCertificates
acm:RequestCertificate
cloudfront:CreateCloudFrontOriginAccessIdentity
cloudfront:CreateDistribution
cloudfront:CreateInvalidation
cloudfront:GetDistribution
cloudfront:GetDistributionConfig
cloudfront:ListCloudFrontOriginAccessIdentities
cloudfront:ListDistributions
cloudfront:ListDistributionsByLambdaFunction
cloudfront:ListDistributionsByWebACLId
```

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

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

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

- 您的输出目录已被覆盖
- 部署 SSR 网站后出现 404 错误
- 您的应用缺少 CloudFront SSR 发行版的重写规则
- 您的应用程序太大，无法部署
- 您的构建失败并出现内存不足错误
- 您的应用程序同时具有 SSR 和 SSG 分支
- 您的应用程序将静态文件存储在带有保留路径的文件夹中
- 您的应用程序已达到上 CloudFront限
- 环境变量不会传递至 Lambda 函数
- 在美国东部（弗吉尼亚州北部）区域中创建 Lambda@Edge 函数
- 您的 Next.js 应用程序使用了不支持的功能
- 您的 Next.js 应用程序中的图像无法加载
- 不支持的区域

您的输出目录已被覆盖

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/**/*
```

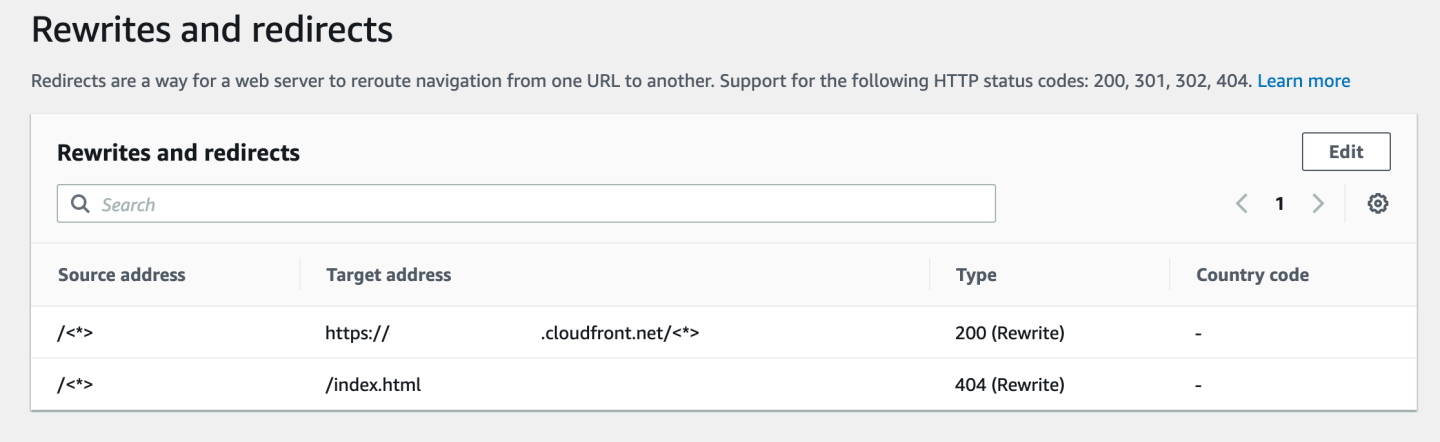
部署 SSR 网站后出现 404 错误

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 您的输出目录已被覆盖 中的步骤操作。

您的应用缺少 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，请在导航窗格中选择应用程序设置，然后选择重写和重定向。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。



您的应用程序太大，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
      },
}
```

您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。

您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。

您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。

您的应用程序已达到上 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，请参阅服务限额用户指南中的请求增加限额。

环境变量不会传递至 Lambda 函数

您在 Amplify 控制台中为 SSR 应用指定的环境变量不会传递到应用程序的函数中。 AWS Lambda 有关如何添加您可以从 Lambda 函数中引用的环境变量的详细说明，请参阅 令服务器端运行时可以访问环境变量。

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。

您的 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。

您的 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。

设置自定义域

您可以将通过 Amplify Hosting 部署的应用程序连接到自定义域。当你使用 Amplify 部署网络应用程序时，Amplify 会将它托管在默认网amplifyapp.com域上，网址为，例如。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。

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？。有关第三方认可的域名注册商名单，请参阅 ICANN 网站上的认证注册商名录。

设置自定义域名时，可以使用 Amplify 为您提供的默认托管证书，也可以使用自己的自定义证书。您可以随时更改该域正在使用的证书。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。

- 您拥有一个已注册的域名。
- 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 托管。

  有关完成此步骤的更多信息，请参阅现有代码入门。
- 您对域名和 DNS 术语有基本的了解。

  有关域和 DNS 的更多信息，请参阅 了解 DNS 术语和概念。

主题

- 了解 DNS 术语和概念
- 使用 SSL/TLS 证书
- 添加在 Amazon Route 53 中管理的自定义域
- 添加由第三方 DNS 提供商管理的自定义域
- 添加由管理的自定义域名 GoDaddy
- 添加由 Google Domains 管理的自定义域
- 更新域名的 SSL/TLS 证书
- 管理子域

- 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。

Amplify Hosting 自定义域激活流程

使用 Amplify Hosting 添加自定义域时，需要完成一系列步骤才能使用自定义域查看您的应用程序。以下列表描述了域设置过程中的每个步骤。

创建 SSL/TLS

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 会验证您是否是该域的所有者。对于由 Amazon Route 53 管理的域，Amplify 会自动更新 DNS 验证记录。对于 Route 53 以外管理的域名，您必须通过第三方 DNS 提供商手动将 Amplify 控制台中提供的 DNS 验证记录添加到您的域中。

如果您使用的是自定义证书，则需要负责验证域名所有权。

域激活

域验证成功。对于在 Route 53 之外管理的域名，您需要通过第三方 DNS 提供商手动将 Amplify 控制台中提供的别名记录添加到您的域中。

使用 SSL/TLS 证书

SSL/TLS 证书是一种数字文档，它允许 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。设置自定义域名时，可以使用 Amplify 为您提供的默认托管证书，也可以使用自己的自定义证书。

使用托管证书，Amplify 会为连接到您的应用程序的所有域名颁发 SSL/TLS 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。如果您的域名提供商在 DNS 设置中修改或删除了别名记录验证记录，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，必须从您选择的第三方证书颁发机构获取证书。接下来，将证书导入 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 AWS 服务 与内部连接的资源一起使用。请务必在美国东部（弗吉尼亚北部）(us-east-1) 地区申请或导入证书。

确保您的自定义证书涵盖您计划添加的所有子域名。您可以在域名开头使用通配符来覆盖多个子域名。例如，如果您的域名是`example.com`，则可以包含通配符域`*.example.com`。这将涵盖诸如`product.example.com`和`api.example.com`之类的子域名。

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

如果您在 ACM 中续订或重新导入自定义证书，Amplify 会刷新与您的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。您有责任续订您的自定义证书并重新导入它们。

您可以随时更改域名正在使用的证书。例如，您可以从默认托管证书切换到自定义证书，或者从自定义证书更改为托管证书。此外，您可以将正在使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。

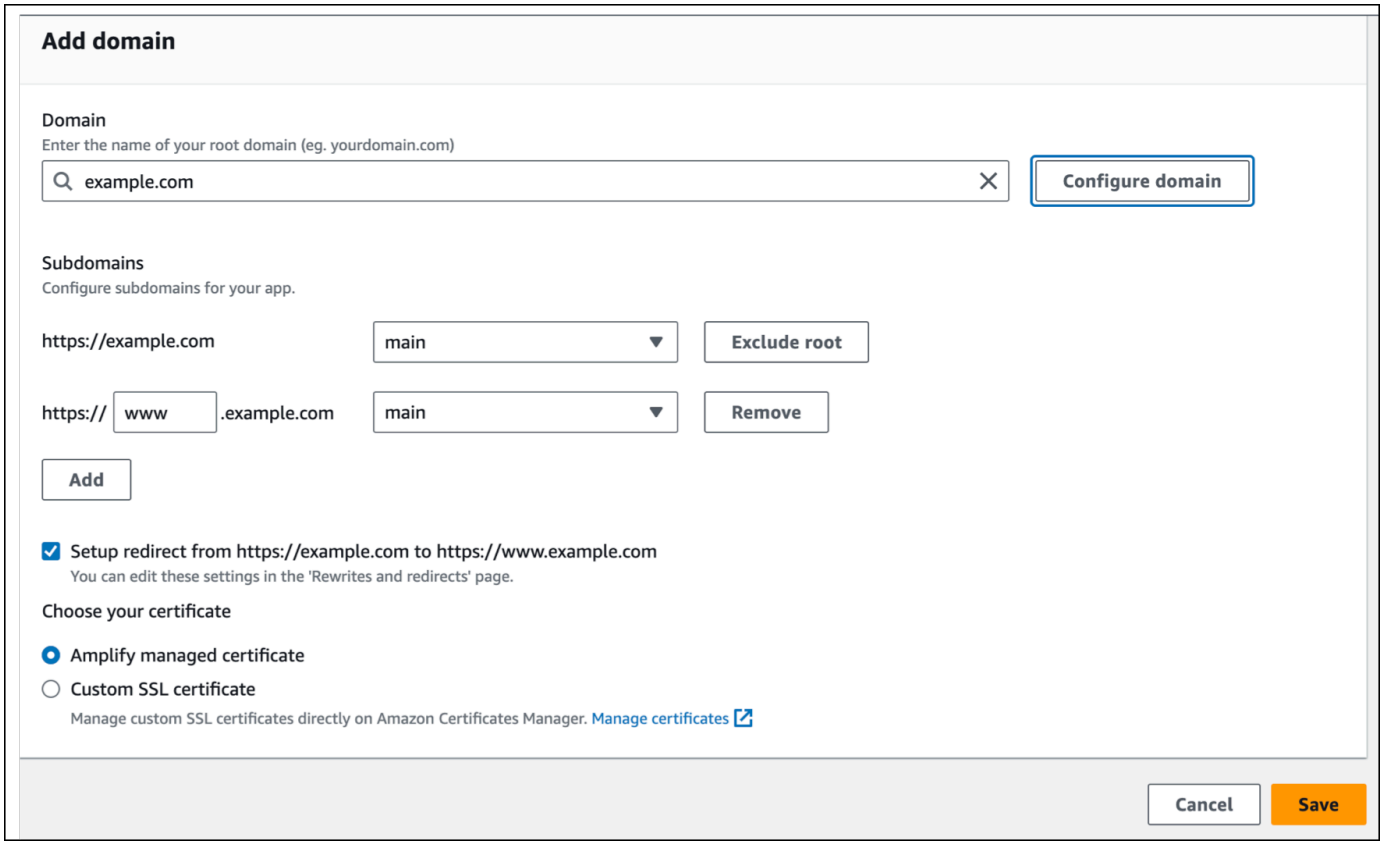
添加在 Amazon Route 53 中管理的自定义域

添加由 Route 53 管理的自定义域

1. 登录 AWS Management Console 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，依次选择应用程序设置、域管理。
4. 在域管理页面上，选择添加域。
5. 对于 “域”，输入您的根域。例如，如果您的域名为 https://example.com，请在域 中输入 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要使用的域名。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，选择配置域名。
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，您会看到子域 https://www.example.com 和 https://example.com，其重定向设置了从根域到 www 子域的重定向。

   （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”，然后配置您的域。



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
   - 使用默认的 Amplify 托管证书。
     - 选择 Amplify 托管证书。
   - 使用自定义的第三方证书。
     a. 选择自定义 SSL 证书。
     b. 从列表中选择要使用的证书。
9. 选择保存。

> **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

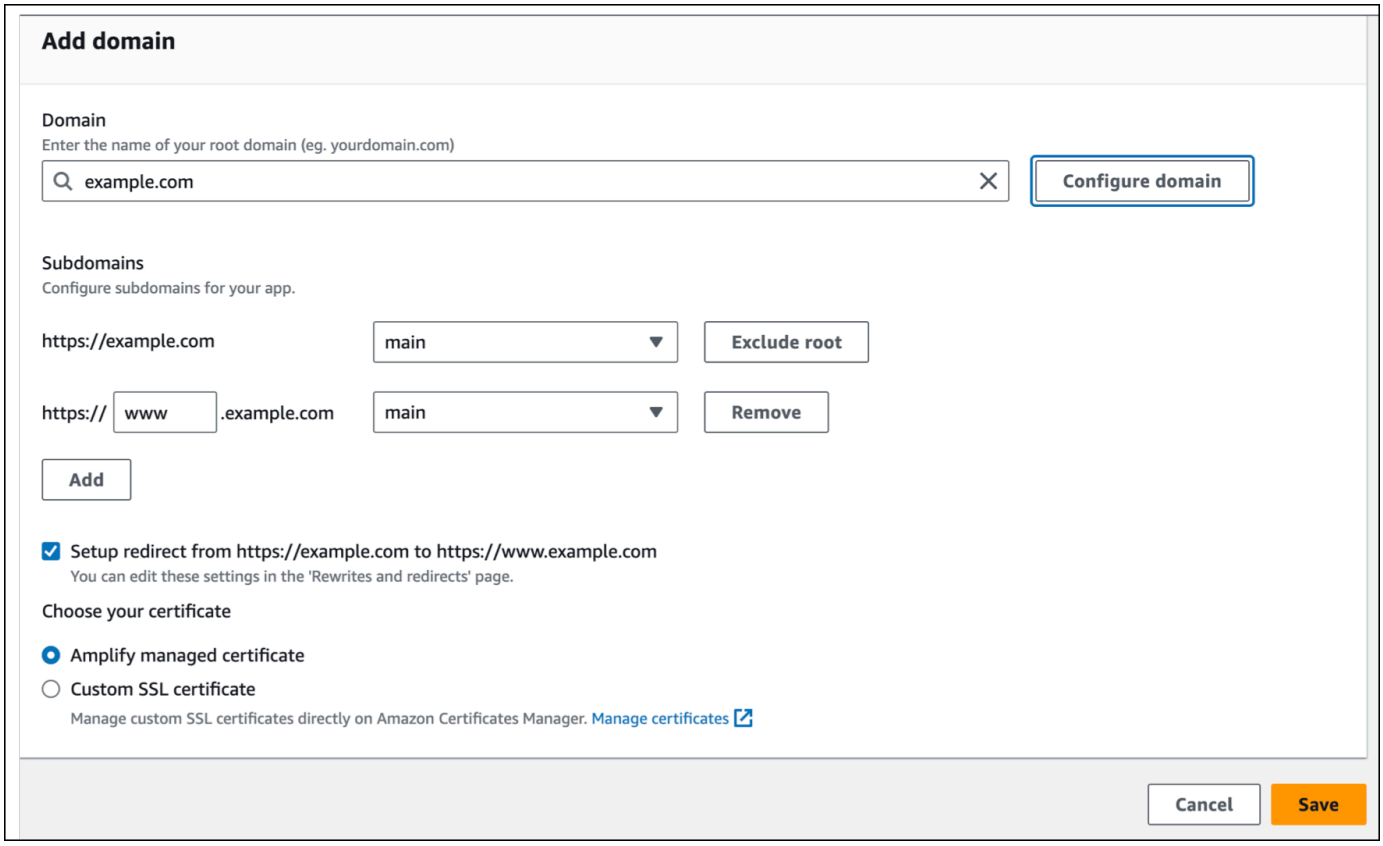
如果未使用 Amazon Route 53 来管理您的域，则可以为使用 Amplify 部署的应用程序添加一个由第三方 DNS 提供商管理的自定义域。

如果您使用的是 GoDaddy 或 Google 域名，the section called “添加由 Google Domains 管理的自定义域”请参阅the section called “添加由管理的自定义域名 GoDaddy”或，了解这些提供商的特定程序。

添加由第三方 DNS 提供商管理的自定义域

1. 登录 AWS Management Console 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，依次选择应用程序设置、域管理。
4. 在域管理页面上，选择添加域。
5. 对于域，请输入根域的名称，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. 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，您会看到子域 https://www.example.com 和 https://example.com，其重定向设置了从根域到 www 子域的重定向。

   （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”，然后配置您的域。



7. 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

   - 使用默认的 Amplify 托管证书。

     - 选择 Amplify 托管证书。

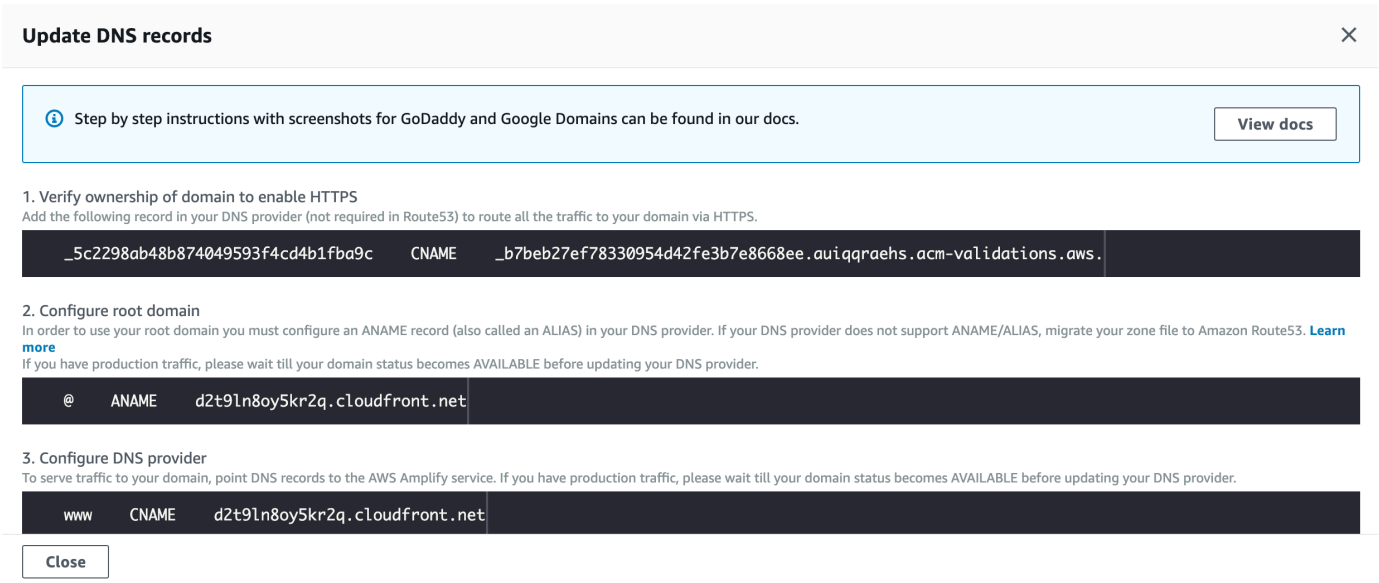
   - 使用自定义的第三方证书。

     a. 选择自定义 SSL 证书。

     b. 从列表中选择要使用的证书。

8. 选择保存。

9. 在操作菜单上，选择查看 DNS 记录。在下一步中，您将使用这些 DNS 记录向第三方域名提供商更新您的 DNS 记录。



10. 请执行以下操作之一：

    - 如果您正在使用 GoDaddy，请转至添加由管理的自定义域名 GoDaddy。
    - 如果您使用的是 Google Domains，请前往 添加由 Google Domains 管理的自定义域。
    - 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。

11. 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。

12. 将 CNAME 配置为指向 AWS 验证服务器。例如，如果验证服务器为 _cjhwou20vhu2exampleuw20vuyb2ovb9.j9s73ucn9vy.acm-validations.aws，请输入 _cjhwou20vhu2exampleuw20vuyb2ovb9.j9s73ucn9vy.acm-validations.aws。Amplify 使用此信息来验证您的域的所有权，并为您的域生成 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。

13. 配置第二条别名记录（例如 https://*.example.com），将您的子域指向 Amplify 域。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。
14. 将 ANAME/ALIAS 记录配置为指向您的域的根**amplifyapp**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。对于没有 ANAME/ALIAS 支持的 DNS 提供商，我们强烈建议将您的 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。

添加由管理的自定义域名 GoDaddy

添加由管理的自定义域名 GoDaddy

1. 按照程序的第一步到第九步进行操作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. 在您的域名列表中，找到要添加的域名，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 xxxxxxxxxxxxx.cloudfront.net，请在值中输入 xxxxxxxxxxxxx.cloudfront.net。

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。单个已验证 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 _c3e2d7eaf1e656b73f46cd 6980fdc0e。

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。

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 记录。对于没有 ANAME/ALIAS 支持的 DNS 提供商，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，然后选择转发。

b. 在域名部分中，选择添加转发。

c. 选择 http://，然后输入要转发到的子域名（例如 www.example.com）作为目标网址。

d. 对于转发类型，请选择临时 (302)。

e. 选择保存。

添加由 Google Domains 管理的自定义域

添加由 Google Domains 管理的自定义域

1. 按照程序的第一步到第九步添加由第三方 DNS 提供商管理的自定义域名。
2. 登录您在 https://domains.google.com 的账户，然后在左侧导航窗格中选择我的域。
3. 在您的域列表中，找到要添加的域，然后选择管理。
4. 在左侧导航窗格中，选择 DNS。Google 会显示您的域的资源记录。您需要添加两条新的别名记录。
5. 创建第一条别名记录，将所有子域指向 Amplify 域，如下所示：

   a. 对于主机名称，仅输入子域名。例如，如果您的子域为 www.example.com，请在主机名称中输入 www。

   b. 对于类型，选择 CNAME。

   c. 对于数据，请输入 Amplify 控制台中可用的值。

      如果 Amplify 控制台将您的应用程序的域显示为 d111111abcdef8.cloudfront.net，请在数据中输入 d111111abcdef8.cloudfront.net。
6. 创建第二个 CNAME 记录以指向 AWS Certificate Manager (ACM) 验证服务器。单个已验证 ACM 会为您的域生成 SSL/TLS 证书。

   a. 对于主机名称，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，则在主机名称中仅输入 _c3e2d7eaf1e656b73f46cd6980fdc0e。

   b. 对于类型，选择 CNAME。

   c. 对于数据，请输入 ACM 验证证书。

      例如，如果验证服务器为 _cf1z2npwt9vzexample93c1j4xzc92wl.2te3iym6kzr.acm-validations.aws，请在数据中输入 _cf1z2npwt9vzexample93c1j4xzc92wl.2te3iym6kzr.acm-validations.aws。
7. 选择保存。

> **Note**
>
> 由 AWS Certificate Manager (ACM) 生成的默认 Amplify; 证书的有效期为 13 个月，只要您的应用程序由 Amplify 托管，就会自动续订。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。

8. Google Domains 对 ANAME/ALIAS 记录的支持处于预览阶段。对于没有 ANAME/ALIAS 支持的 DNS 提供商，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。如果您要保留 Google Domains 作为您的提供商并更新根域，请添加转发并设置域转发。找到您的 Google 域的网站页面。然后选择转发域，并在 Web 转发页面上配置转发。

> **Note**
>
> Google 域的 DNS 设置更新最长需要 48 小时生效。如需帮助解决出现的错误，请参阅自定义域问题排查。

更新域名的 SSL/TLS 证书

您可以随时更改域名正在使用的 SSL/TLS 证书。例如，您可以从使用托管证书更改为使用自定义证书。您也可以更改该域正在使用的自定义证书。有关证书的更多信息，请参阅使用 SSL/TLS 证书。

使用以下步骤更新域中使用的证书类型或自定义证书。

更新域的证书

1. 登录 AWS Management Console 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，依次选择应用程序设置、域管理。
4. 在域名管理页面上，选择管理域名。
5. 在您的域名的详细信息页面上，找到 “选择您的证书” 部分。更新证书的步骤因您要更改的类型而异。
   - 从自定义证书更改为默认 Amplify 托管证书
     - 选择 Amplify 托管证书。

- 从托管证书更改为自定义证书

  a. 选择自定义 SSL 证书。

  b. 从列表中选择要使用的证书。

- 将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。

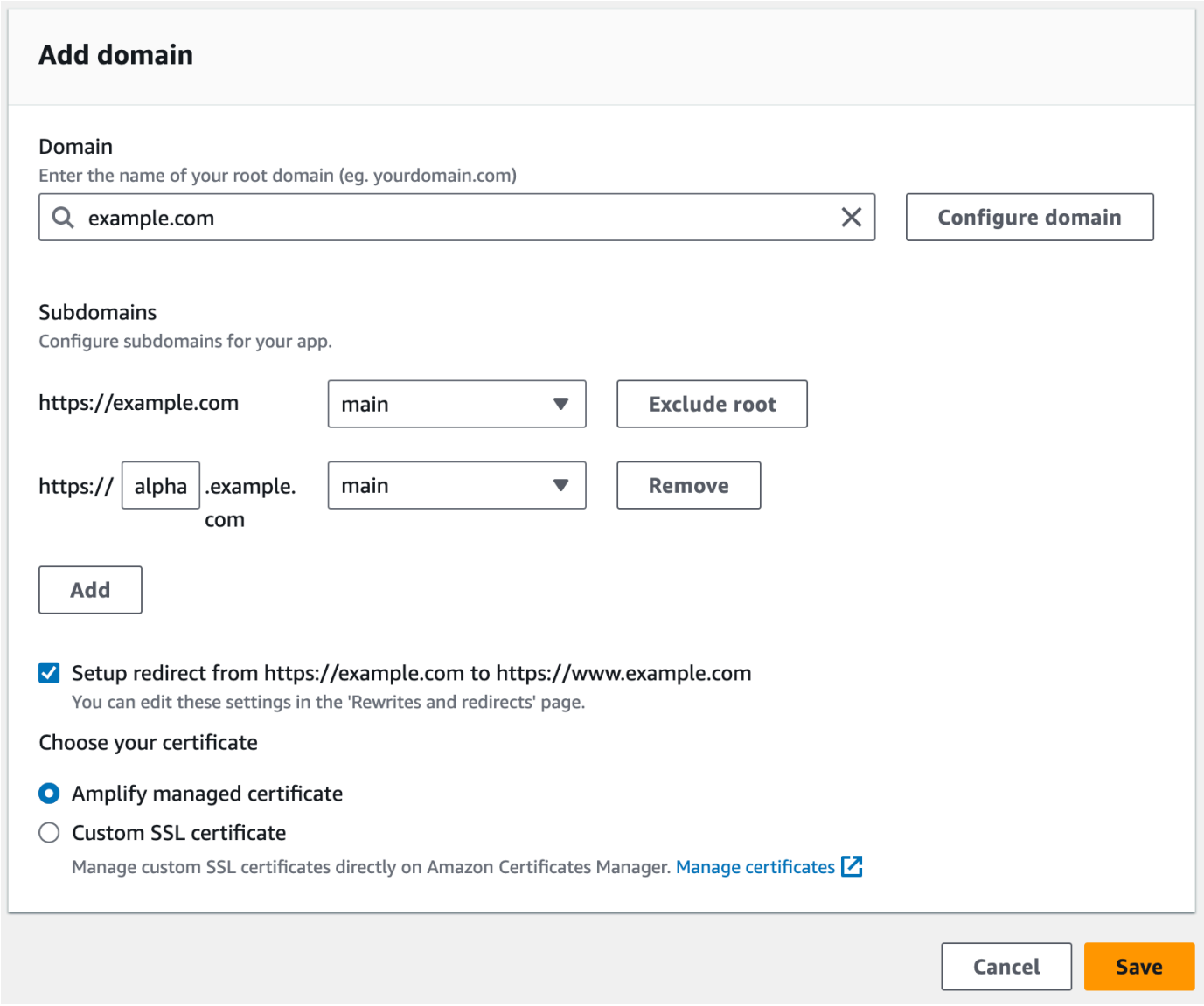
6. 选择更新。该域的状态详细信息将表明 Amplify 已启动托管证书的 SSL 创建过程或自定义证书的配置过程。

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

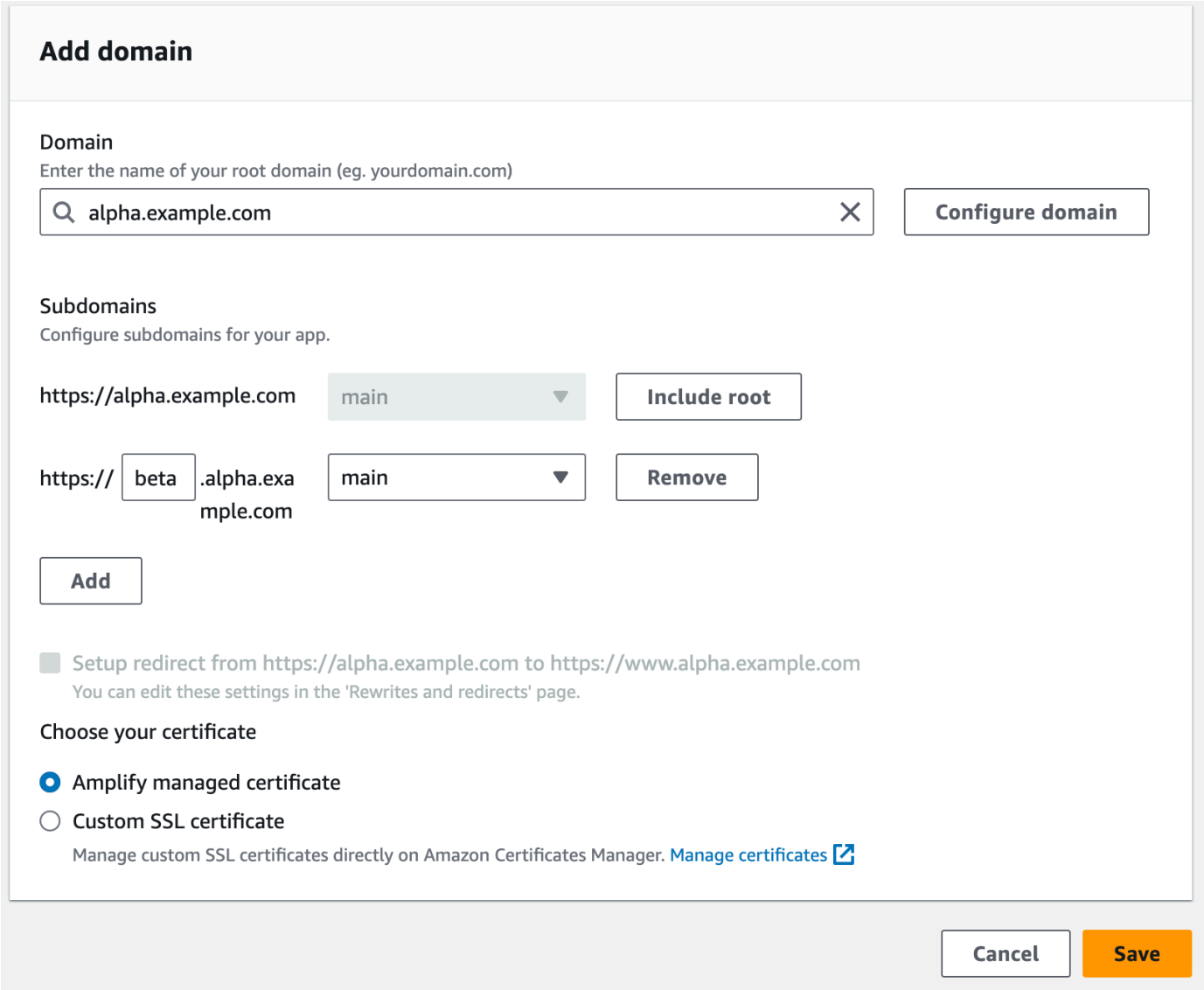
1. 登录 AWS Management Console 并打开 Amplify 控制台。
2. 选择要向其中添加子域的应用程序。
3. 在导航窗格中，依次选择应用程序设置、域管理。
4. 在域管理页面上，选择添加域。
5. 对于域，输入您的根域名称，然后选择配置域。例如，如果您的域名为 https://example.com，请在域 中输入 example.com。
6. 选择排除根目录并修改子域名称。例如，如果域名是 e xample.com，则可以将其修改为仅添加子域名 alpha，如以下屏幕截图所示。



添加多级子域

1. 登录 AWS Management Console 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，依次选择应用程序设置、域管理。
4. 在域管理页面上，选择添加域。
5. 对于域，请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，如以下屏幕截图所示。



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 Management Console 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，依次选择应用程序设置、域管理。
4. 在域名管理页面上，选择管理域名。
5. 在编辑域中，您可以根据需要编辑现有的子域。
6. （可选）要添加新的子域，请选择添加。

7. 单击更新以保存您的更改。

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 Management Console 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，依次选择应用程序设置、域管理。
4. 在域名管理页面上，选择管理域名。
5. 在编辑域中，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 添加。
     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。

在以下示例屏幕截图中，已为应用程序的 dev 分支创建了 *.example.com 通配符子域。

d. 单击更新以保存您的更改。

- 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. 单击更新以保存您的更改。

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 Management Console 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，依次选择应用程序设置、域管理。
4. 在域名管理页面上，选择管理域名。
5. 选中左下角的子域自动检测复选框。

> **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，您应用程序的拉取请求网页预览也可以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，请查阅本节中的以下主题以获取问题排查帮助。

如果您在这里找不到问题的解决方案，请联系 AWS Support。有关更多信息，请参阅 AWS Support 用户指南中的创建支持案例。

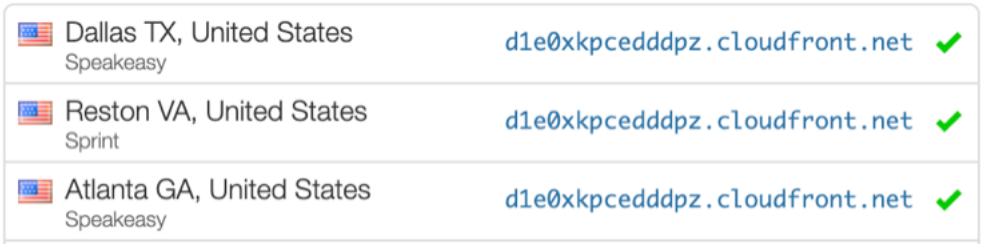
主题

- 如何验证我的别名记录是否已解析？
- 我在第三方托管的域卡在了等待验证状态
- 我在 Amazon Route 53 托管的域卡在了等待验证状态
- 我收到 CNAME 错误 AlreadyExistsException
- 我收到需要额外验证错误
- 我在网址上收到 404 错误 CloudFront
- 我在访问我的域时收到 SSL 证书或 HTTPS 错误

如何验证我的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。

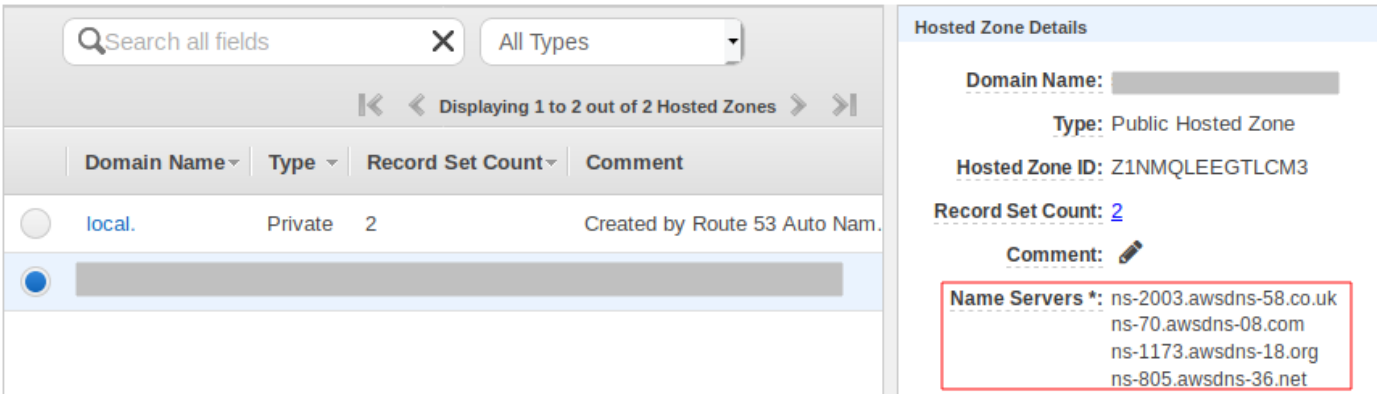
> **Important**
>
> 创建您的自定义域后，请务必立即更新 CNAME 记录。在 Amplify 控制台中创建应用程序后，每隔数分钟就会检查一次 CNAME 记录，以确定它是否已解析。如果一小时后仍未解决，则每隔数小时进行一次检查，这可能会导致您的域名在准备好使用方面会有所延迟。如果您在创建应用程序几小时后添加或更新了 CNAME 记录，则这很可能是您的应用程序陷入待验证状态的原因。

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。

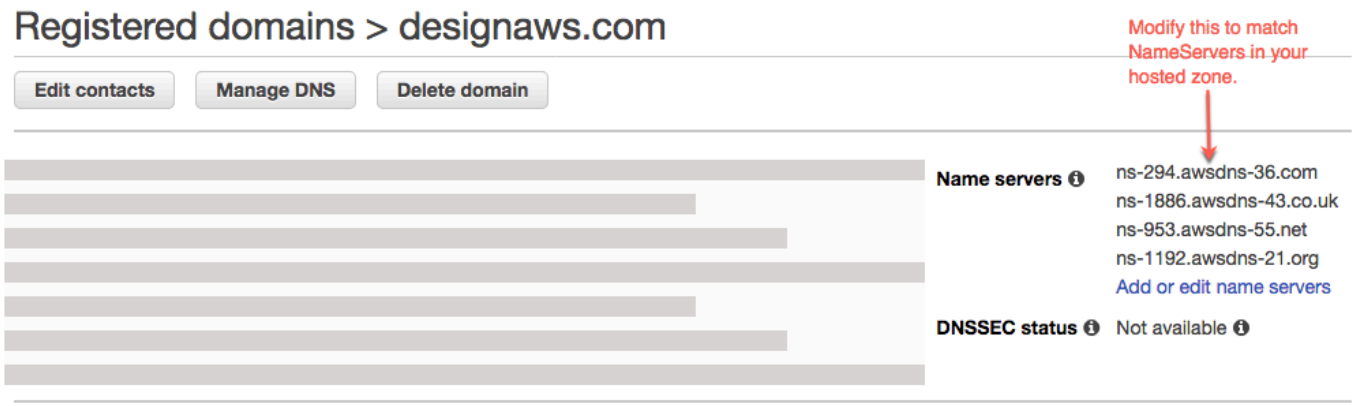
我在 Amazon Route 53 托管的域卡在了等待验证状态

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 AWS Support。有关更多信息，请参阅 AWS Support 用户指南中的创建支持案例。

我收到 CNAME 错误 AlreadyExistsException

如果您收到 CNAME AlreadyExistsException 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 Amazon 分发中。 CloudFront 执行以下步骤诊断出错的原因。

1. 登录 Amazon CloudFront 控制台并确认您没有将此域部署到任何其他分配。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 Settings（设置）部分中，选择 Edit（编辑）。

d. 从备用域名 (ANAME) 中移除此域名。然后选择保存更改。

3. 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。如果您将 www.example.com 用于其他应用程序，则 www.example.com 不能与当前正在连接的应用程序一起使用。您可以使用其他子域，例如 blog.example.com。

4. 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 AWS Support。有关更多信息，请参阅 AWS Support 用户指南中的创建支持案例。

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 AWS Support。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 Management Console 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。

        - 添加在 Amazon Route 53 中管理的自定义域.
        - 添加由第三方 DNS 提供商管理的自定义域.
        - 添加由管理的自定义域名 GoDaddy.
        - 添加由 Google Domains 管理的自定义域.
   - 请联系 AWS Support 以重新颁发您的 SSL 证书。

配置构建设置

当您使用 Amplify Hosting 部署应用程序时，它会通过检查存储库中的 `package.json` 文件来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。

您可以在 Amplify 控制台中编辑应用程序的构建设置，方法是选择应用程序设置和构建设置。这些构建设置将应用于您的应用程序中的所有分支，在存储库中保存了 `amplify.yml` 文件的分支除外。

> **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。

构建规范的命令和设置

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。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。了解如何使用您的前端部署后端。

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

构建规范的缓存字段用于缓存构建时的依赖项，例如 node_modules 文件夹，并根据客户应用程序内置的软件包管理器和框架自动推荐。第一次构建时会缓存这里的任何路径，在随后的构建中，我们会重新填充缓存，并尽可能使用这些缓存的依赖项来加快构建时间。

以下构建规范示例演示了基本的 YAML 语法：

构建规范的 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
        - path
test:
  phases:
    preTest:
      commands:
        - *enter command*
    test:
      commands:
        - *enter command*
    postTest:
      commands:
        - *enter command*
  artifacts:
    files:
        - location
        - location
    configFilePath: *location*
```

```
    baseDirectory: *location*
```

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 执行一组命令；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
```

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
```

使用您的前端部署后端

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 镜像使用名为的非特权用户运行你的代码。amplifyAmplify 授予该用户使用 Linux sudo 命令运行操作系统命令的权限。如果要为缺少的依赖项安装操作系统包，则可以使用诸如yum和之类rpm的命令sudo。

以下示例构建部分演示了使用sudo命令安装操作系统软件包的语法。

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

每个构建的键/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>
```

跳过某个提交的构建

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]。

禁用自动构建

您可以配置 Amplify，禁用在每次提交代码时自动构建。要进行设置，请选择应用程序设置 > 常规，然后滚动到包含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。以后提交到该分支的操作将不再触发新的构建。

启用或禁用基于 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 Management Console 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，依次选择应用程序设置、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`。

启用或禁用基于 diff 的后端构建

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 Management Console 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，依次选择应用程序设置、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。

单一存储库构建设置

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。

单一存储库构建规范的 YAML 语法

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。

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*
```

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

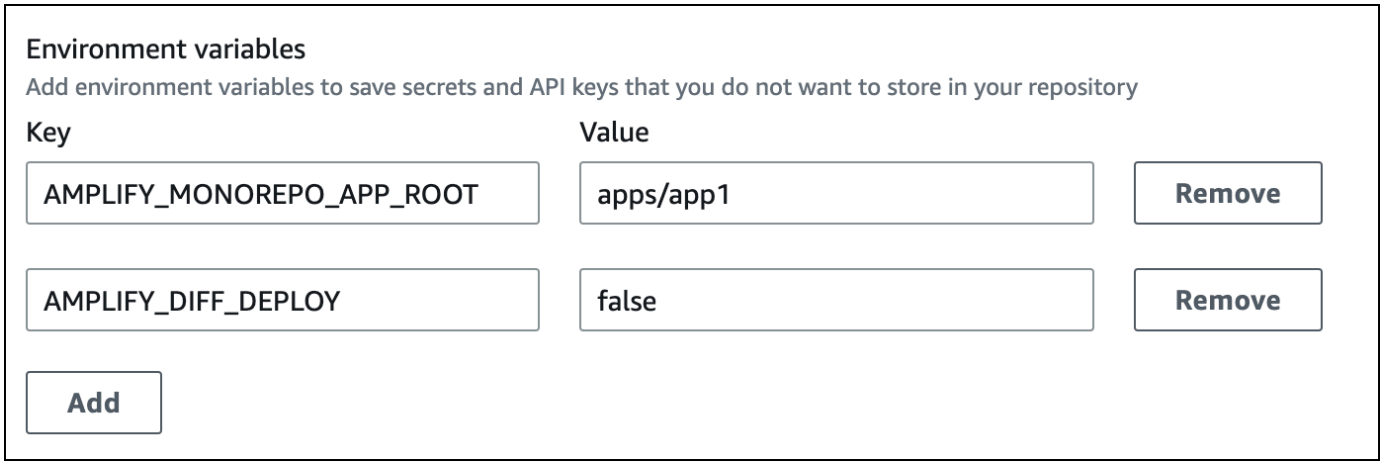
1. 登录 AWS Management Console 并打开 Amplify 控制台。
2. 选择右上角的新建应用程序、托管 Web 应用程序。
3. 在托管 Web 应用程序页面上，选择您的 Git 提供程序，然后选择继续。
4. 在添加存储库分支页面上，执行以下操作：
    a. 从最近更新的存储库列表中选择存储库的名称。
    b. 对于分支，请选择要使用的分支名称。

c. 选择连接单一存储库？ 选择一个文件夹。

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. 在构建设置页面上，您可以使用默认设置，也可以自定义应用程序的构建设置。在以下示例屏幕截图中，在环境变量部分，Amplify 使用您在步骤 4d 中指定的路径将 AMPLIFY_MONOREPO_APP_ROOT 设置为 apps/app1。



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 Management Console 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，依次选择应用程序设置、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. 选择添加变量。

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。

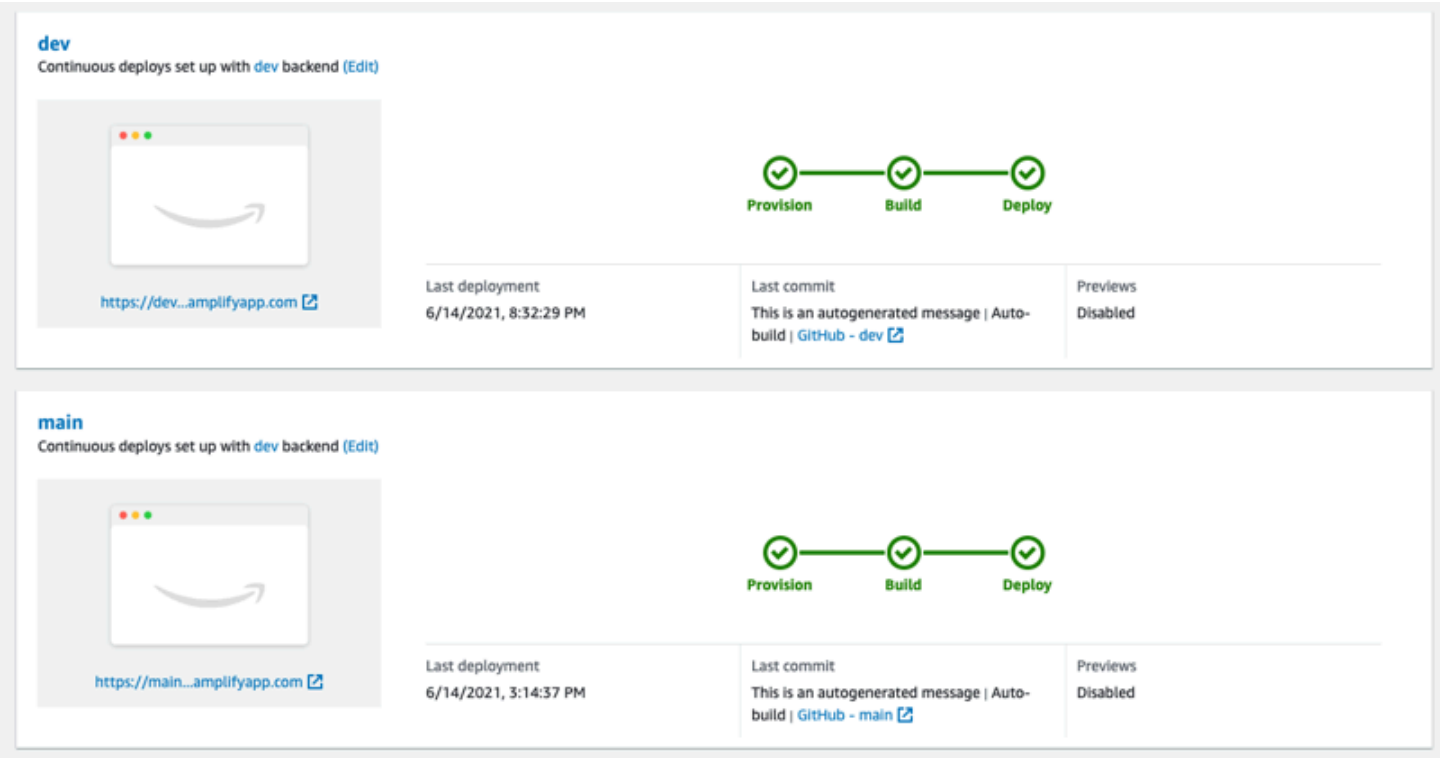
```
version: 1
applications:
  - frontend:
      phases:
        preBuild:
          commands:
            - npm install -g pnpm
```

功能分支部署和团队工作流程

Amplify 控制台的设计目的是与功能分支和 GitFlow 工作流程兼容。每当开发人员连接自己的存储库中的新分支时，Amplify 都会利用 Git 分支创建新的部署。连接您的第一个分支后，可以通过添加分支来创建新的功能分支部署，如下所示：

1. 在分支列表页面上，选择 Connect branch (连接分支)。
2. 从您的存储库中选择一个分支。
3. 保存您的应用程序，然后部署它。

您的应用程序现在具有位于 https://main.appid.amplifyapp.com 和 https://dev.appid.amplifyapp.com 上的两个部署。虽然这可能因团队而异，但通常由主分支跟踪发布代码，并其为您的生产分支。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

- 使用 Amplify 后端环境的团队工作流程
- 基于模式的功能分支部署
- 构建时自动生成 Amplify 配置

- 有条件后端构建
- 跨应用程序使用 Amplify 后端

使用 Amplify 后端环境的团队工作流程

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。有关此部署的更多信息，请参阅 开始使用全栈持续部署。

> **Note**
>
> 您可以轻松地在 Amplify 应用程序中重复使用 Amplify 后端环境。有关更多信息，请参阅跨应用程序使用 Amplify 后端。

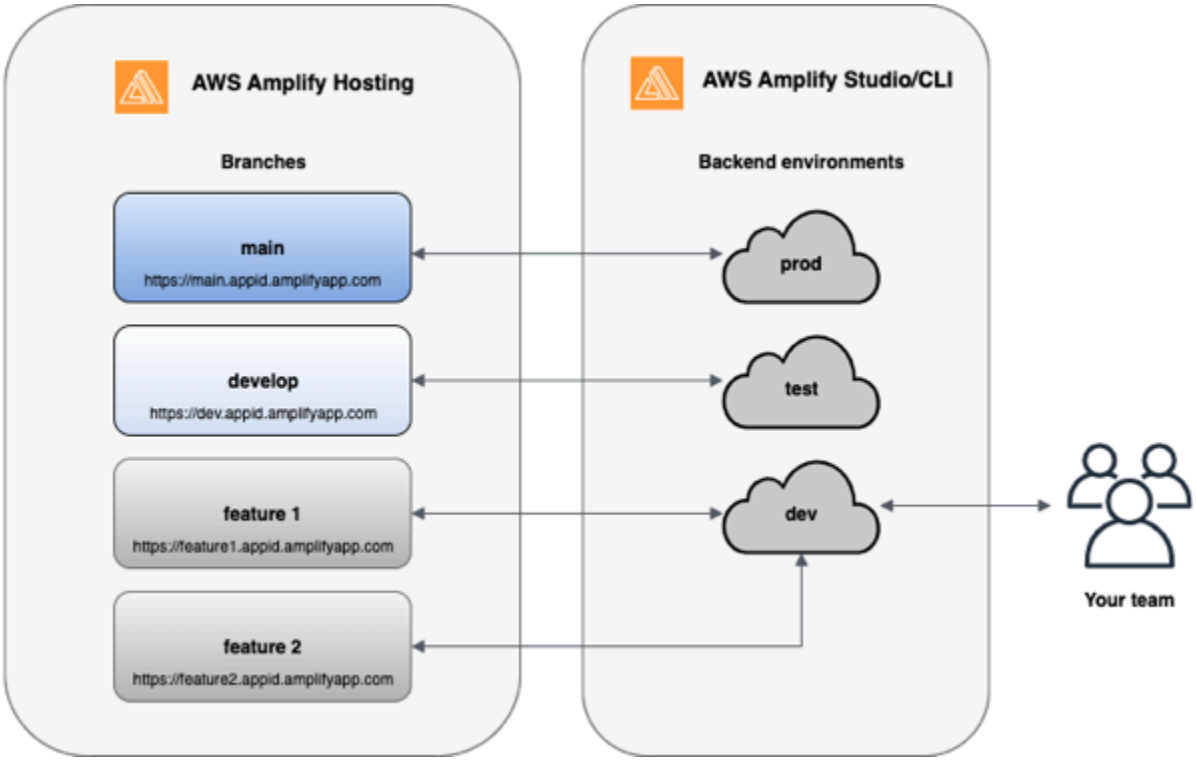
Amplify Hosting 会在功能分支部署时持续部署后端资源，如 GraphQL API 和 Lambda 函数。您可以使用以下分支模型在 Amplify Hosting 上部署后端和前端。

主题

- 功能分支工作流程
- GitFlow 工作流程
- 每个开发人员的沙盒

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 create-react-app 或 Gatsby 等引导工具创建一个项目。

```
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。

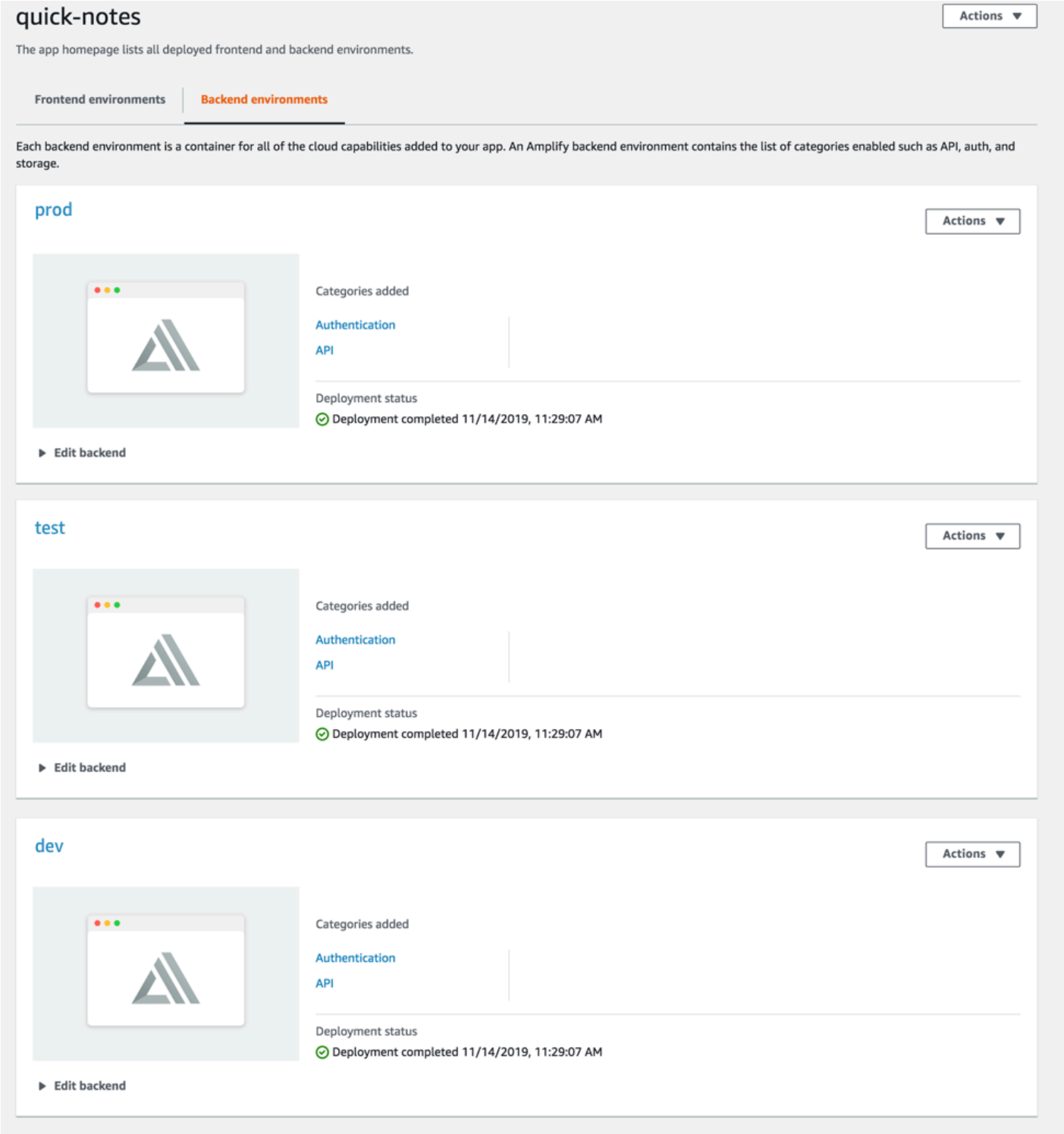
```
amplify env add
 ? Do you want to use an existing environment? (Y/n): n
 ? Enter a name for the environment: test
...
amplify push
```

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

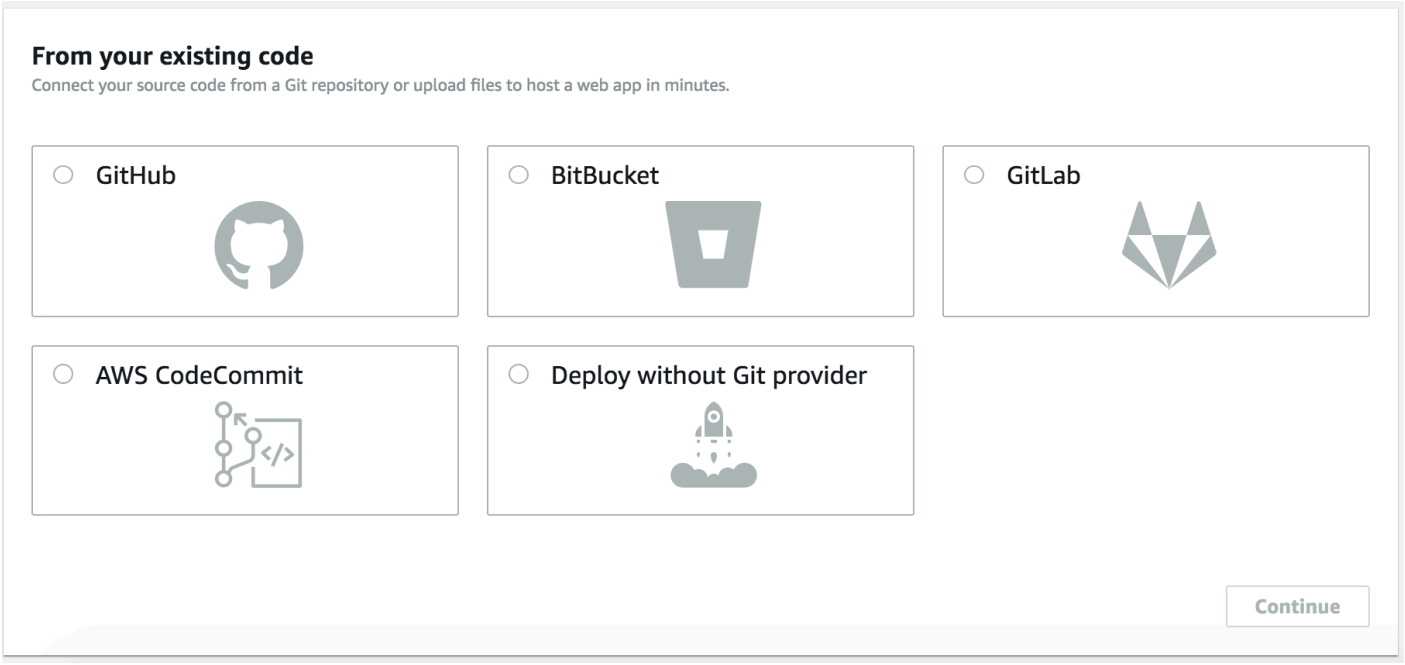
4. 将代码推送到您选择的 Git 存储库（在此示例中，我们假设您已推送到主存储库）。

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

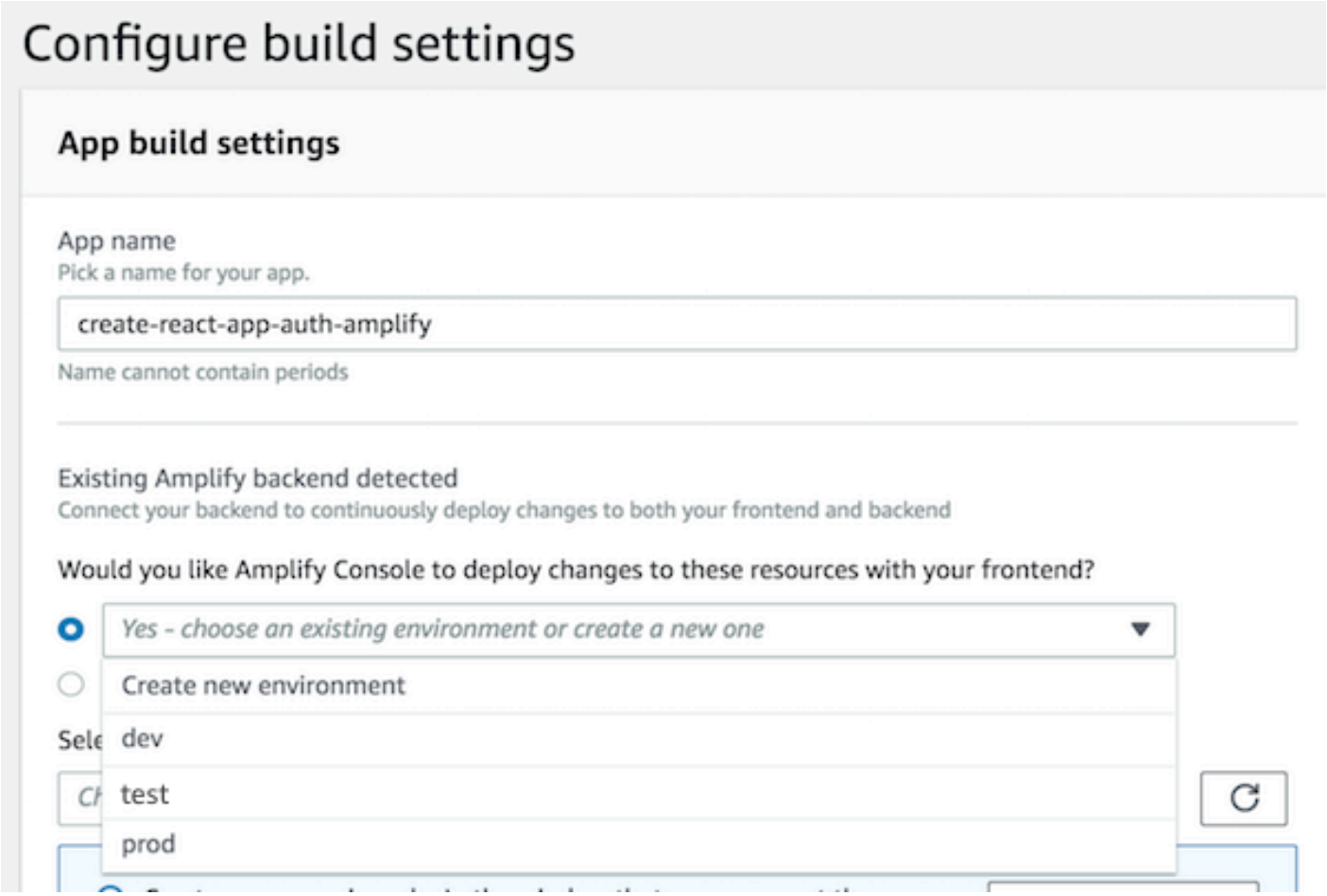
5. 访问 AWS Management Console 中的 Amplify，查看您当前的后端环境。从导航位置向上导航一级，查看在后端环境选项卡中创建的所有后端环境的列表。



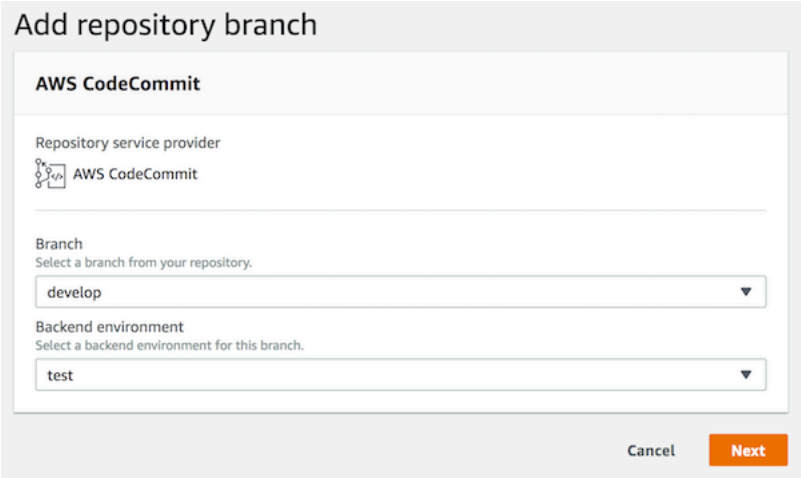
6. 切换到前端环境选项卡，连接存储库提供程序和主分支。



7. 在构建设置屏幕中，选择现有的后端环境在主分支上设置持续部署。从下拉列表中选择 生产 并将服务角色授予 Amplify。选择 Save and deploy (保存并部署)。构建完成后，您将在 https://main.appid.amplifyapp.com 上获得主分支部署。



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：如果您的系统上安装了 AWS CLI（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您可以通过 https://newinternet.appid.amplifyapp.com 访问您的功能，与您的团队成员共享。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，它将通过 https://dev.appid.amplifyapp.com 上的分支部署更新 Amplify 中的后端以及前端。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。

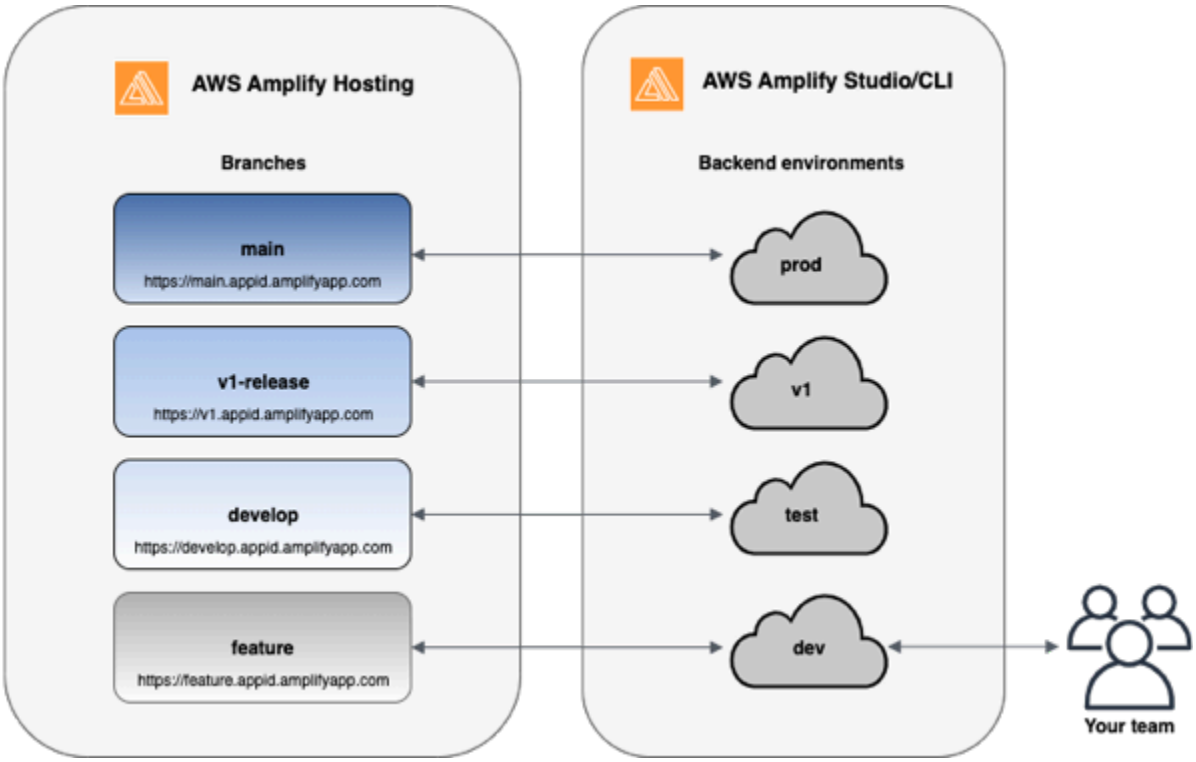
14从 Git 和 Amplify 中删除您的功能分支，并从云中删除后端环境（您始终可以通过运行“amplify env checkout prod”并运行“amplify env add”来启动新的环境）。

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

GitFlow 工作流程

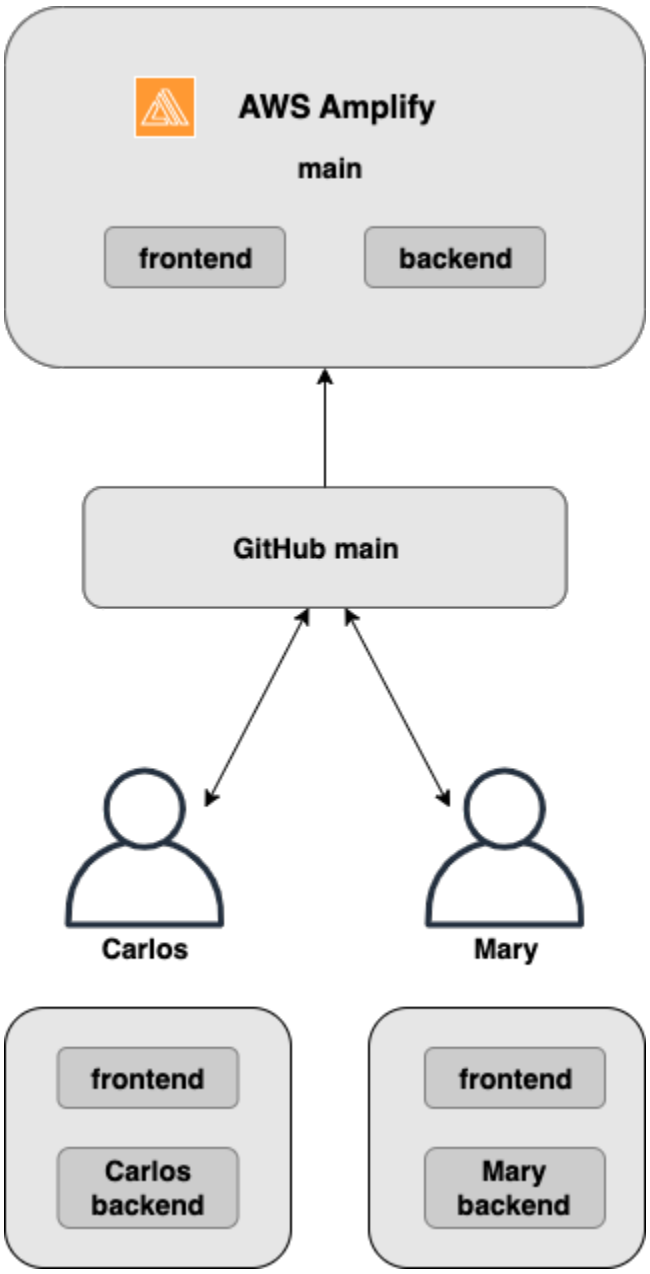
GitFlow 使用两个分支来记录项目的历史记录。主分支仅跟踪发布代码，开发分支用作新功能的集成分支。通过将新开发与已完成的工作隔离，GitFlow 简化了并行开发。在功能分支中完成新开发（例如功能和非紧急错误修复）。当开发人员确信代码已准备好发布时，功能分支将合并回集成开发分支。仅将来自发布分支和修补程序分支的合并提交到主分支（以修复紧急错误）。

下图显示了使用 GitFlow 的推荐设置。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 create-react-app 或 Gatsby 等引导工具创建一个项目。

```
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）。

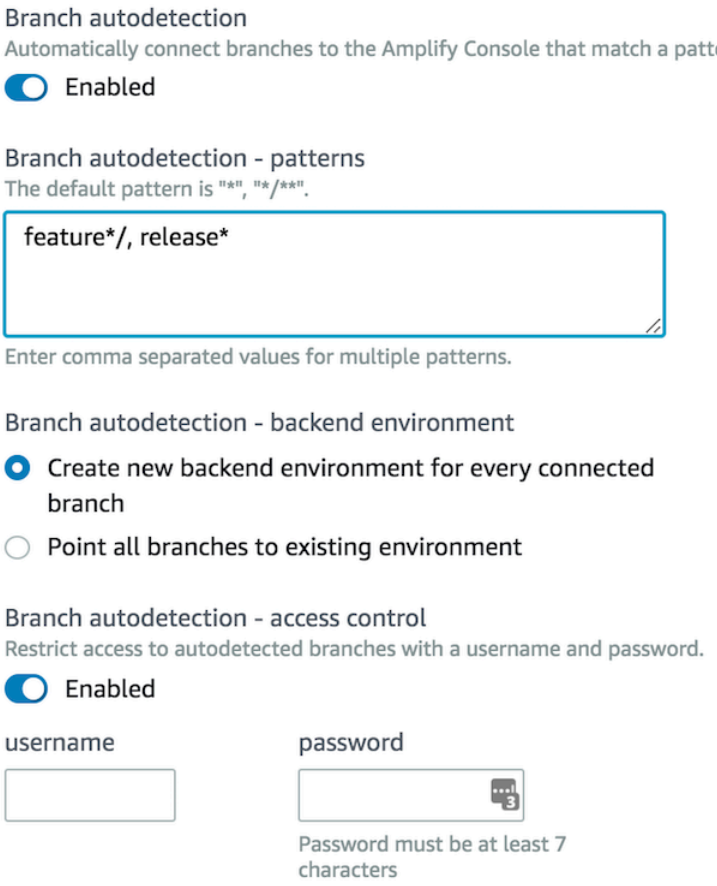
```
git commit -am 'Added mary sandbox'
git push origin main
```

4. 将您的存储库 > 主分支连接到 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 Save and deploy (保存并部署)。构建完成后，您将在 https://main.appid.amplifyapp.com 上获得主分支部署，一个新的后端环境会链接到该分支。
6. 在 Amplify 中连接开发分支（此时假设开发和主分支相同），然后选择创建新环境。构建完成后，您将在 https://develop.appid.amplifyapp.com 上获得开发分支部署，一个新的后端环境会链接到该分支。

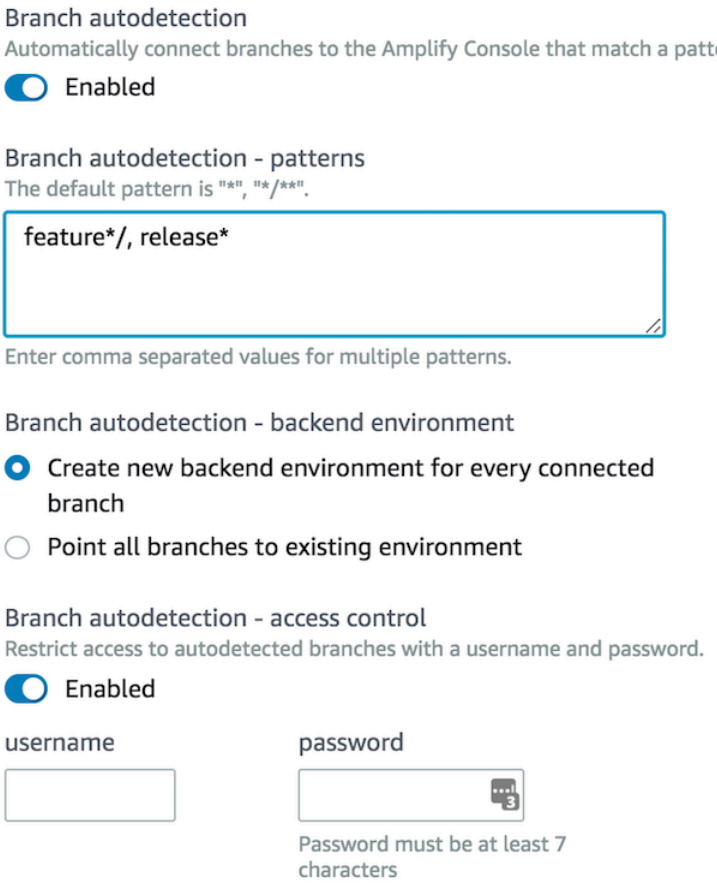
基于模式的功能分支部署

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 GitFlow 工作流程的产品团队现在可以定义诸如“release**”之类的模式，从而自动将以“release”开头的 Git 分支部署到可共享的 URL。此博客文章介绍了如何将此功能与不同的团队工作流程结合使用。

1. 选择 App settings (应用程序设置) > General (常规) > Edit (编辑)。
2. 将分支自动检测开关切换为已启用。



1. 定义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“发布”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*`。
2. 通过将 Branch autodetection - access control (分支自动检测 - 访问控制) 设置为 Enabled (已启用)，为自动创建的所有分支设置自动密码保护。
3. 对于使用 Amplify 后端构建的应用程序，您可以选择创建新环境或者将所有分支指向现有后端。



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 添加在 Amazon Route 53 中管理的自定义域
- 有关使用 Route 53 的更多信息，请参阅什么是 Amazon Route 53？。

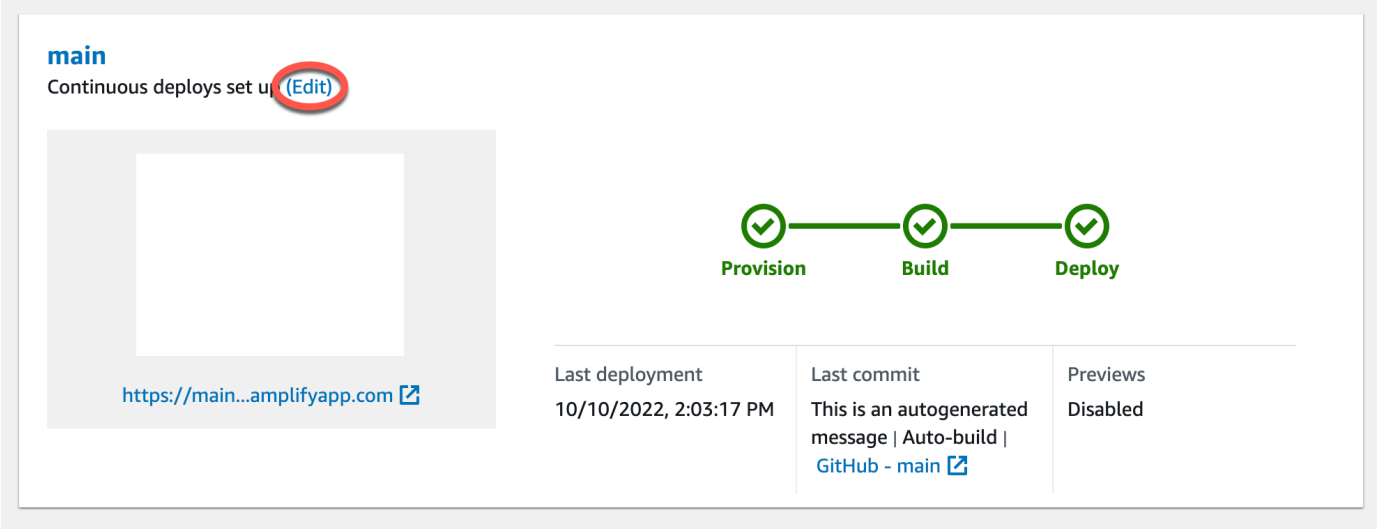
构建时自动生成 Amplify 配置

Amplify 支持在构建时自动生成 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。

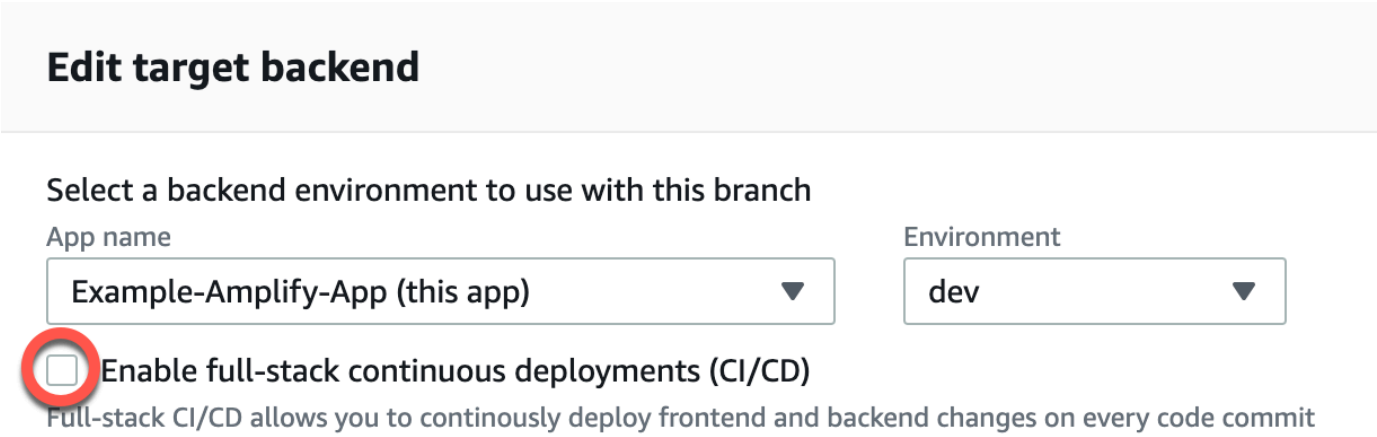
在构建时自动生成 **`aws-exports.js`**

1. 登录 AWS Management Console 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，请参阅添加服务角色。
7. 选择 Save（保存）。Amplify 会在下次构建应用程序时应用这些更改。

有条件后端构建

Amplify 支持在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 启用或禁用基于 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 构建规范的命令和设置。

跨应用程序使用 Amplify 后端

Amplify 使您能够在给定区域的所有应用程序中轻松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 Management Console 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. 在服务导航窗格中，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，请参阅添加服务角色。

e. 选择 Next（下一步）。

7. 选择 Save and deploy (保存并部署)。

将分支连接到现有应用程序时重复使用后端

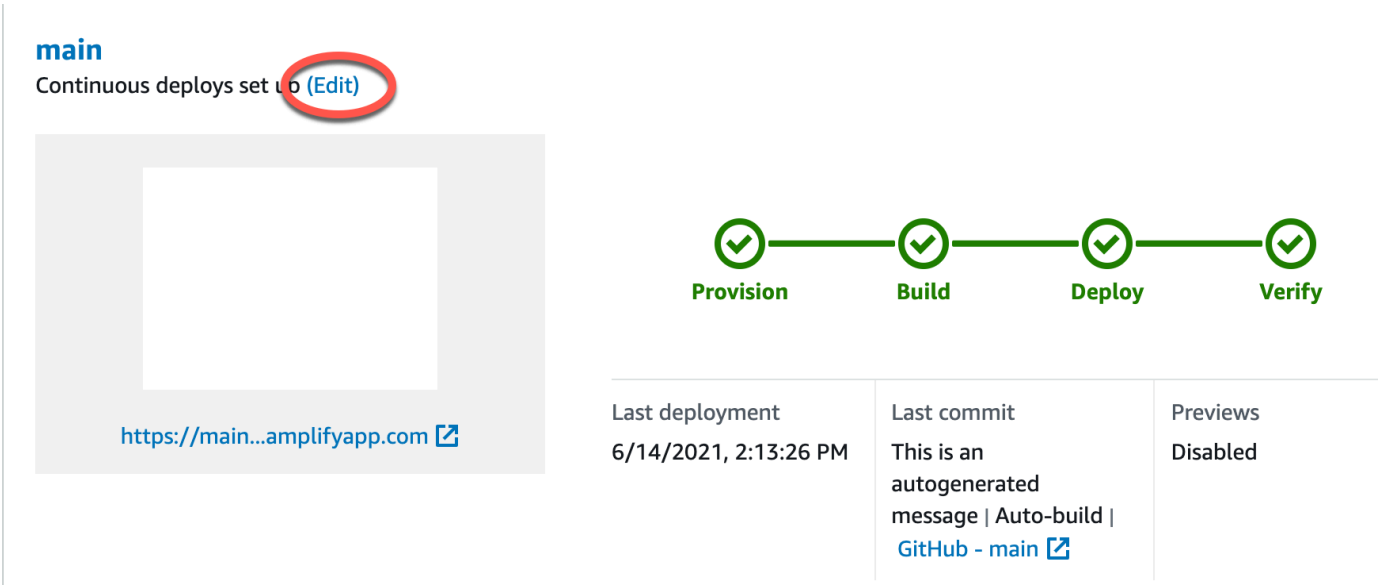
在将分支连接到现有 Amplify 应用时重复使用后端

1. 登录 AWS Management Console 并打开 Amplify 控制台。
2. 选择要连接新分支的应用程序。
3. 在导航窗格中的应用程序设置下，选择常规。
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。
10. 选择 Next（下一步）。
11. 选择 Save and deploy (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 Management Console 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的全栈 CI/CD。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，而不修改后端环境。
8. 选择 Save（保存）。Amplify 会在下次构建应用程序时应用这些更改。

手动部署

通过手动部署，您可以在不连接 Git 提供程序的情况下，用 Amplify Hosting 发布您的 Web 应用程序。您可以从桌面拖放文件夹，在几秒钟内托管您的网站。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 URL。

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 部署服务器端渲染的应用程序。

拖放手动部署

使用拖放方式手动部署应用程序

1. 登录到 AWS Management Console 并打开 Amplify 控制台。
2. 如何访问托管您的 Web 应用程序页面取决于您是从 Amplify 主页还是从所有应用程序页面开始。
   - 从 Amplify 主页
     a. 选择入门。
     b. 在 送达 部分中，选择入门。
   - 从所有应用程序页面
     - 在右上角，选择新应用程序和托管 Web 应用程序
3. 在托管您的 Web 应用程序页面上，选择不使用 Git 提供程序部署。然后，选择 Continue（继续）。
4. 在开始手动部署部分的应用程序名称中，输入您的应用程序的名称。
5. 在环境名称中，为环境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. 您可以将文件从桌面拖放到拖放区域，也可以使用选择文件从计算机中选择文件。您拖放或选择的文件可以是包含网站根目录的文件夹或 zip 文件。
8. 选择 Save and deploy (保存并部署)。

Amazon S3 或 URL 手动部署

从 Amazon S3 或公共 URL 手动部署应用程序

1. 登录到 AWS Management Console 并打开 Amplify 控制台。
2. 在页面顶部，选择 入门。
3. 在 送达 部分中，选择入门。
4. 在托管您的 Web 应用程序页面上，选择不使用 Git 提供程序部署。然后，选择 Continue（继续）。
5. 在开始手动部署部分的应用程序名称中，输入您的应用程序的名称。
6. 在环境名称中，为环境输入一个有意义的名称，例如 **`development`** 或 **`production`**。
7. 对于方法，选择 Amazon S3 或任何 URL。
8. 上传文件的过程取决于上传方法。
   - Amazon S3
     a. 对于存储桶，请从列表中选择 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（ACL）。有关更多信息，请参阅 Amazon S3 存储桶访问故障排除。
     b. 对于 Zip 文件，选择要部署的 zip 文件的名称。
   - 任何 URL
     - 在资源 URL 中，输入要部署的压缩文件的 URL。
9. 选择 Save and deploy (保存并部署)。

> **Note**
>
> 创建 zip 文件夹时，请务必压缩生成输出的内容，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。

Amazon S3 存储桶访问故障排除

在创建 Amazon S3 存储桶时，您使用其 Amazon S3 对象所有权设置来控制是为该存储桶启用还是禁用访问控制列表（ACL）。要将应用程序从 Amazon S3 存储桶手动部署到 Amplify，必须在该存储桶上启用 ACL。

如果您在从 Amazon S3 存储桶部署时遇到 `AccessControlList` 错误，则表示该存储桶是在禁用 ACL 的情况下创建的，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。

“部署到 Amplify”按钮

使用 部署到 Amplify Hosting 按钮可以公开或在您的团队中共享 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

将此按钮添加到 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. Amplify 控制台 URL，带有指向 GitHub 存储库的链接。您可以复制存储库的 URL（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 URL。

```
[![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 URL。

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

设置 Amplify 对 GitHub 存储库的访问权限

Amplify 现在使用 GitHub 应用程序功能来授权 Amplify 以只读形式访问 GitHub 存储库。使用 Amplify GitHub 应用程序，就可以对权限进行更精细的调整，使您能够授予 Amplify 仅访问指定存储库的权限。要了解有关 GitHub 应用程序的详细信息，请参阅 GitHub 网站上的 About GitHub Apps。

当您连接存储在 GitHub 存储库中的新应用程序时，默认情况下 Amplify 使用 GitHub 应用程序来访问存储库。但是，您之前从 GitHub 存储库连接的现有 Amplify 应用程序使用 OAuth 进行访问。CI/CD 将继续适用于这些应用程序，但我们强烈建议您对其进行迁移以使用新的 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 GitHub 应用程序的安装位置。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`，将`##`替换为要部署 Amplify 应用程序的区域。例如，要在美国西部（俄勒冈州）区域安装 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 Amplify GitHub 应用程序
- 将现有 OAuth 应用程序迁移到 Amplify GitHub 应用程序
- 为 AWS CloudFormation、CLI 和 SDK 部署设置 Amplify GitHub 应用程序
- 使用 Amplify GitHub 应用程序设置网页预览

为新部署安装和授权 Amplify GitHub 应用程序

当您利用 GitHub 存储库中的现有代码向 Amplify 部署新应用程序时，请按照以下说明安装和授权 GitHub 应用程序。

安装和授权 Amplify GitHub 应用程序

1. 登录 AWS Management Console 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 Amplify 托管入门页面上，选择 GitHub，然后选择继续。
4. 如果这是首次连接 GitHub 存储库，则浏览器会在 GitHub.com 上打开一个新页面，请求允许在您的 GitHub 帐户中进行 AWS Amplify 授权。选择授权。
5. 接下来，必须在您的 GitHub 账户中安装 Amplify GitHub 应用程序。在 GitHub.com 上打开一个页面，请求允许在您的 GitHub 账户中安装和授权 AWS Amplify。

6. 选择要安装 Amplify GitHub 应用程序的 GitHub 账户。
7. 请执行下列操作之一：
   - 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 Next（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，选择部署。

将现有 OAuth 应用程序迁移到 Amplify GitHub 应用程序

您之前从 GitHub 存储库连接的现有 Amplify 应用程序使用 OAuth 来访问存储库。我们强烈建议您迁移这些应用程序，使用 Amplify GitHub 应用程序。

按照以下说明迁移应用程序，并删除您的 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和授权 GitHub 应用程序后，您只需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 OAuth 迁移到 GitHub 应用程序

1. 登录 AWS Management Console 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. 浏览器会在 GitHub.com 上打开一个新页面，请求允许在您的 GitHub 帐户进行 AWS Amplify 授权。选择授权。
6. 选择要安装 Amplify GitHub 应用程序的 GitHub 账户。
7. 请执行下列操作之一：
   - 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安装和授权 GitHub 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 AWS 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 应用程序使用 OAuth 来访问存储库。这可能包括您使用 Amplify Command Line (CLI)、AWS CloudFormation 或开发工具包部署的应用程序。我们强烈建议您迁移这些应用程序，使用新的 Amplify GitHub 应用程序。迁移必须在 AWS Management Console 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 Amplify GitHub 应用程序。

您可以使用 AWS CloudFormation、Amplify CLI 和开发工具包来部署一个新的 Amplify 应用程序，该应用使用 GitHub 应用程序访问存储库。此过程要求您首先在您的 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。

在您的账户中安装 Amplify GitHub 应用程序

1. 打开 Web 浏览器，导航到 Amplify GitHub 应用程序在您要部署应用程序的 AWS 区域的安装位置。

   使用格式 `https://github.com/apps/aws-amplify-REGION/installations/new`，将*##*替换为您自己的输入。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. 选择要安装 Amplify GitHub 应用程序的 GitHub 账户。
3. 请执行下列操作之一：
   - 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. 登录到您的 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。当您使用 CLI、AWS CloudFormation 或开发工具包部署 Amplify 应用程序时，您需要提供令牌。

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、AWS CloudFormation 或开发工具包部署新应用程序。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 Amplify API 参考中的 CreateApp 和 AWS CloudFormation 用户指南中的 AWS:: Amplify::App。

以下 CLI 命令部署了一个新的 Amplify 应用程序，该应用使用 GitHub 应用程序访问存储库。用您自己的信息替换 *myapp-using-githubapp*、*https://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

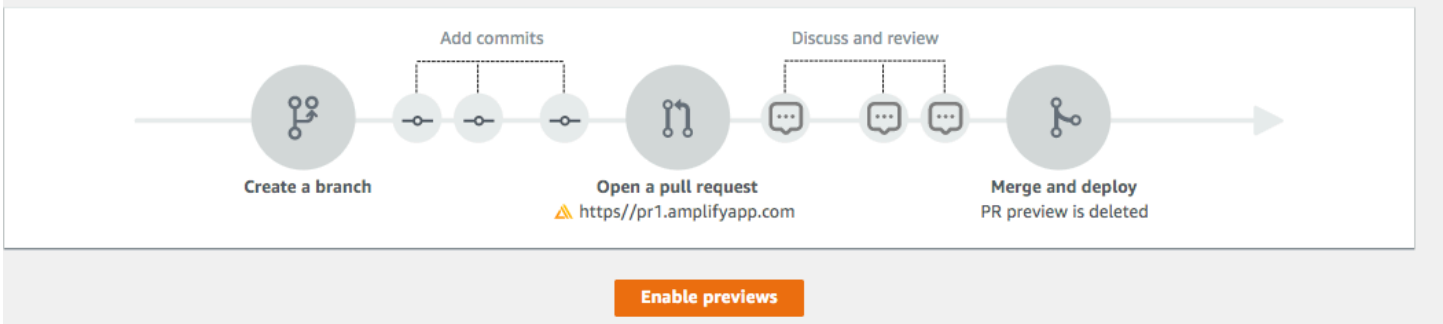
网页预览会将每个向 GitHub 存储库发出的拉取请求 (PR) 部署到一个唯一的预览网址。预览现在使用 Amplify GitHub 应用程序访问您的 GitHub 存储库。有关安装和授权 GitHub 应用程序进行网页预览的说明，请参阅 启用 Web 预览。

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 (PR) 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。

> **Note**
>
> 目前，Amplify 的预览分支可以支持 GitLab、BitBucket 和 AWS CodeCommit，并且支持功能与 GitHub 不完全相同。`AWS_PULL_REQUEST_ID` 环境变量仅在使用 GitHub 作为存储库提供程序时可用。

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 URL，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）都会启动一个临时后端，该后端在 PR 关闭时会被删除。



> **Important**
>
> 出于安全考虑，您可以在所有带有私有存储库的应用程序上启用 Web 预览，但不能在所有带有公共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。

启用 Web 预览

对于存储在 GitHub 存储库中的应用程序，预览使用 Amplify GitHub 应用程序访问存储库。如果您要在先前使用 OAuth 访问从 GitHub 存储库部署的现有 Amplify 应用程序上启用 Web 预览，则必须先迁移该应用程序，以使用 Amplify GitHub 应用程序。有关迁移说明，请参阅 将现有 OAuth 应用程序迁移到 Amplify GitHub 应用程序。

为拉取请求启用 Web 预览

1. 选择应用程序设置、预览，然后选择 启用预览。

   > **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. 仅适用于 GitHub 存储库，请执行以下操作，以便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 GitHub 应用程序的 GitHub 账户。

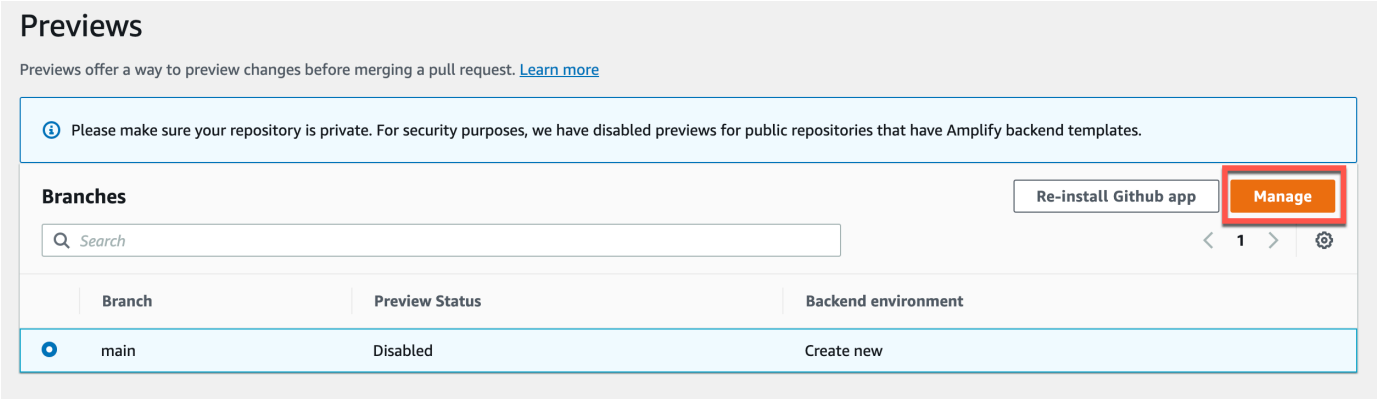
   c. GitHub.com 打开一个页面，用于为您的账户配置存储库权限。

   d. 请执行下列操作之一：

      - 要将安装应用于所有存储库，请选择所有存储库。
      - 要将安装限制在您选择的特定存储库中，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。

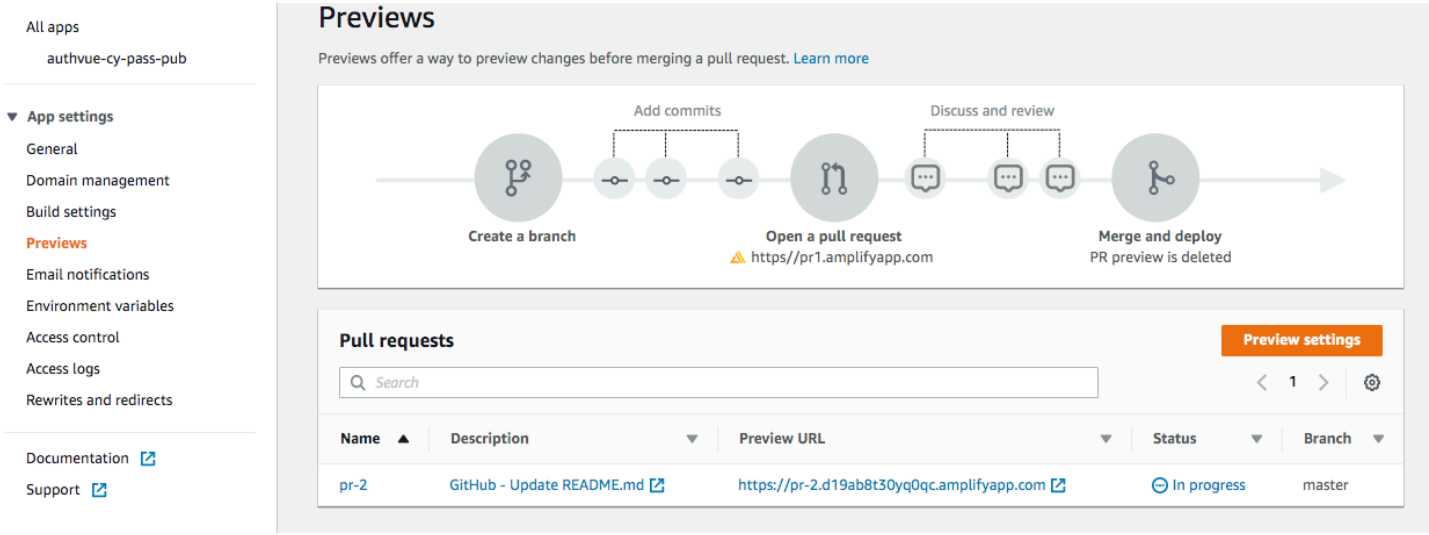
   e. 选择 保存

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，然后选择管理。

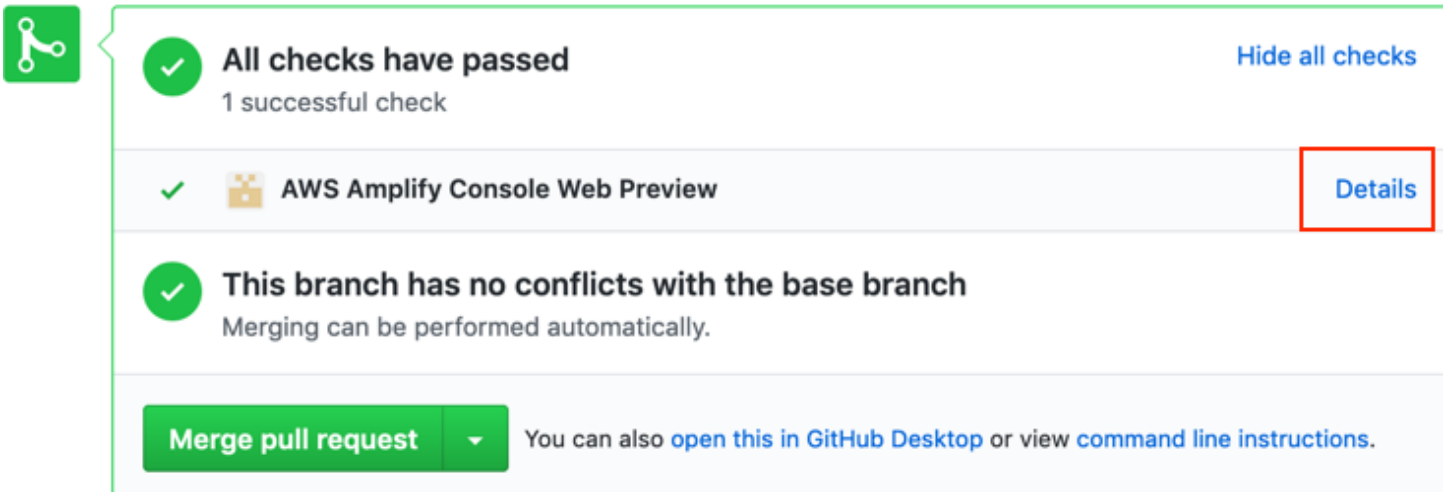


4. 在管理分支的预览设置 窗口中，打开拉取请求预览。
5. 对于全栈应用程序，执行以下操作之一：
   - 选择为每个拉取请求创建新的后端环境。此选项使您能够在不影响生产的情况下测试更改。
   - 选择将此分支的所有拉取请求指向现有环境。
6. 选择 Confirm（确认）。

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。



仅适用于 GitHub 存储库，您可以直接从 GitHub 账户中的拉取请求访问您的 URL 的预览。



拉取请求关闭后，将删除预览 URL，并删除与拉取请求关联的所有临时后端环境。

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来自拉取请求的 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。

向您的 Amplify 应用程序添加端到端 Cypress 测试

您可以在 Amplify 应用程序的测试阶段运行端到端 (E2E) 测试，以便在将代码推送到生产环境之前捕获回归情况。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。

教程：设置 Cypress 端到端测试

Cypress 是一个基于 JavaScript 的测试框架，您可以借此在浏览器上运行 E2E 测试。有关演示如何设置 E2E 测试的教程，请参阅题为为使用 Amplify 的全栈 CI/CD 部署运行端到端的 Cypress 测试的博客文章。

向您的现有 Amplify 应用程序添加测试

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。

构件 > 配置文件路径

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）。

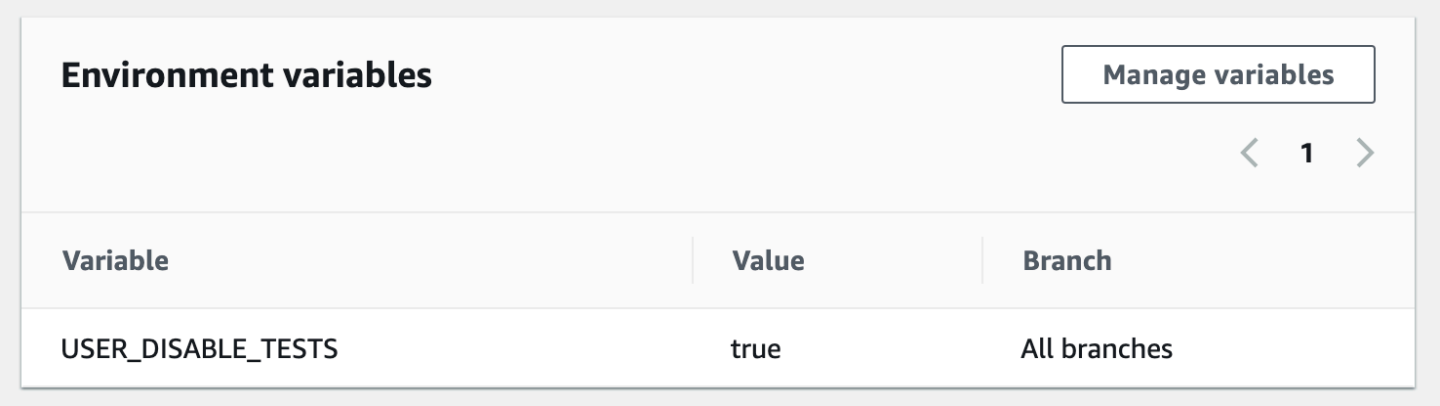
以下示例摘自构建规范 amplify.yml 文件，其中展示了如何向您的应用程序添加 Cypress 测试。

```
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'
```

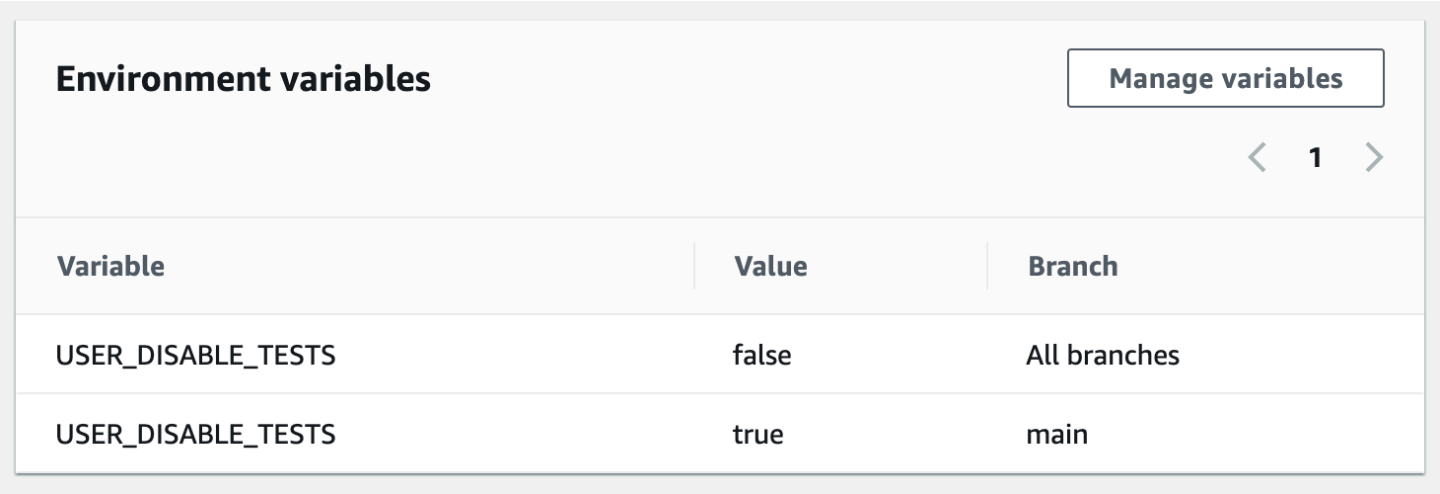
禁用测试

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。

使用重定向

重定向使 Web 服务器能够将导航从一个 URL 重新路由到另一个 URL。使用重定向的常见原因包括：自定义 URL 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 URL 更改为 Web 应用程序需要的形式。

重定向类型

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。
- 为用户界面的 A/B 比较提供测试页面。

> **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 URL 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。

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 Management Console 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，选择应用程序设置，然后选择重写和重定向。
4. 在重写和重定向部分，请选择编辑。
5. 添加重定向的过程会有所不同，具体取决于您是要单独添加规则还是要进行批量编辑：
   - 要创建单个重定向，请选择添加规则。
     a. 对于原始地址，输入用户请求的地址。
     b. 对于目标地址，请输入向用户呈现内容的目标地址。
     c. 对于类型，请从列表中选择重定向的类型。
     d. （可选）对于国家/地区代码，请输入一个由两个字母组成的国家/地区代码条件。
   - 要在 JSON 编辑器中批量编辑重定向，请选择打开文本编辑器。
     - 在批量添加重写和重定向 JSON 编辑器中，手动添加或更新重定向。
6. 选择保存。

重定向顺序

从列表顶部向下执行重定向。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
```

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。

简单重定向和重写

本节中包括常见重定向情景的示例代码。

> **Note**
>
> 原始地址域匹配不区分大小写。

您可以使用以下示例代码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 [{"source": "/original.html", "status": "301", "target": "/destination.html", "condition": null}]

您可以使用以下示例代码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 [{"source": "/docs/<*>", "status": "301", "target": "/documents/<*>", "condition": null}]

您可以使用以下示例代码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 [{"source": "/<*>", "status": "200", "target": "/index.html", "condition": null}]

您可以使用以下示例代码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 [{"source": "https://mydomain.com", "status": "200", "target": "https://www.mydomain.com", "condition": null}]

您可以使用以下示例代码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 [{"source": "https://mydomain.com", "status": "302", "target": "https://www.mydomain.com/documents/", "condition": null}]

您可以使用以下示例代码将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 [{"source": "/<*>", "status": "404", "target": "/404.html", "condition": null}]

单页 Web 应用程序 (SPA) 的重定向

大多数 SPA 框架支持 HTML5 history.pushState() 更改浏览器位置而不触发服务器请求。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 [{"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。

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

结尾斜杠和清洁 URL

为创建清洁 URL 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 通过在需要时添加结尾斜杠来自动创建清洁 URL。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 URL	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代码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 [{"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

您可以使用以下示例代码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 [{"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代码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 [{"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代码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 [{"source": "/documents", "status": "302", "target": "/documents/us/", "condition": "<US>"}]

重定向和重写中的通配符表达式

您可以在原始地址中使用通配符表达式进行重定向或重写。<*>必须将表达式放在原始地址的末尾，而且该表达式必须是唯一的。Amplify 会忽略包含多个通配符表达式的原始地址，或者将其用于其他位置。

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)	

以下两个示例演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)	

限制对分支的访问

如果您正在开发未发布的功能，则可以对尚未准备好公开访问的功能分支进行密码保护。在分支上设置访问控制后，当用户尝试访问分支的 URL 时，系统会提示他们输入用户名和密码。

在功能分支上设置密码

1. 登录到 AWS Management Console 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，选择 应用程序设置，然后选择访问控制。
4. 在访问控制设置部分，选择管理访问。
5. 在访问控制设置 中执行以下任一操作：
   - 要设置适用于所有连接分支的用户名和密码，请打开应用全局密码。例如，如果您连接了 主要、开发 和 功能 分支，则可以使用全局密码来为所有分支设置相同的用户名和密码。
   - 要将用户名和密码应用于单个分支，请关闭应用全局密码。对于要为其设置唯一用户名和密码的分支，请选择访问权限设置的所需的限制密码，然后输入用户名和密码。
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。

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 不允许您创建带有前缀的环境变量。AWS此前缀仅供 Amplify 内部使用。

> **Important**
>
> 请勿使用环境变量存储密钥。将密钥存储在使用 AWS Systems Manager 参数存储库创建的环境密钥中。有关更多信息，请参阅环境密钥。

Amplify 环境变量

Amplify 控制台默认可访问以下环境变量。

变量名称	描述	示例值
_BUILD_TIMEOUT	构建超时时间（分钟）	`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"}]`
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 禁用测试。	`true`
AWS_APP_ID	当前构建的应用程序 ID	`abcd1234`

变量名称	描述	示例值
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 URL	`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`
AWS_PULL_REQUEST_ID	Web 预览版本的拉取请求 ID。 此环境变量仅在 GitHub 用作存储库提供程序时可用。	1
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`

变量名称	描述	示例值
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，请参阅启用或禁用基于 diff 的前端构建和部署。	true
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。有关更多信息，请参阅启用或禁用基于 diff 的后端构建。	true
AMPLIFY_BACKEND_PULL_ONLY	Amplify 管理此环境变量。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。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。	true
AMPLIFY_ENABLE_DEBUG_OUTPUT	将此变量设置true为可在日志中打印堆栈跟踪。这对于调试后端构建错误很有帮助。	true

变量名称	描述	示例值
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`
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。有关详细信息，请参阅 Amplify 生成角色的 IAM 权限边界。	`arn:aws:iam::123456789012:policy/example-policy`

变量名称	描述	示例值
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。有关更多信息，请参阅更新架构。	`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。

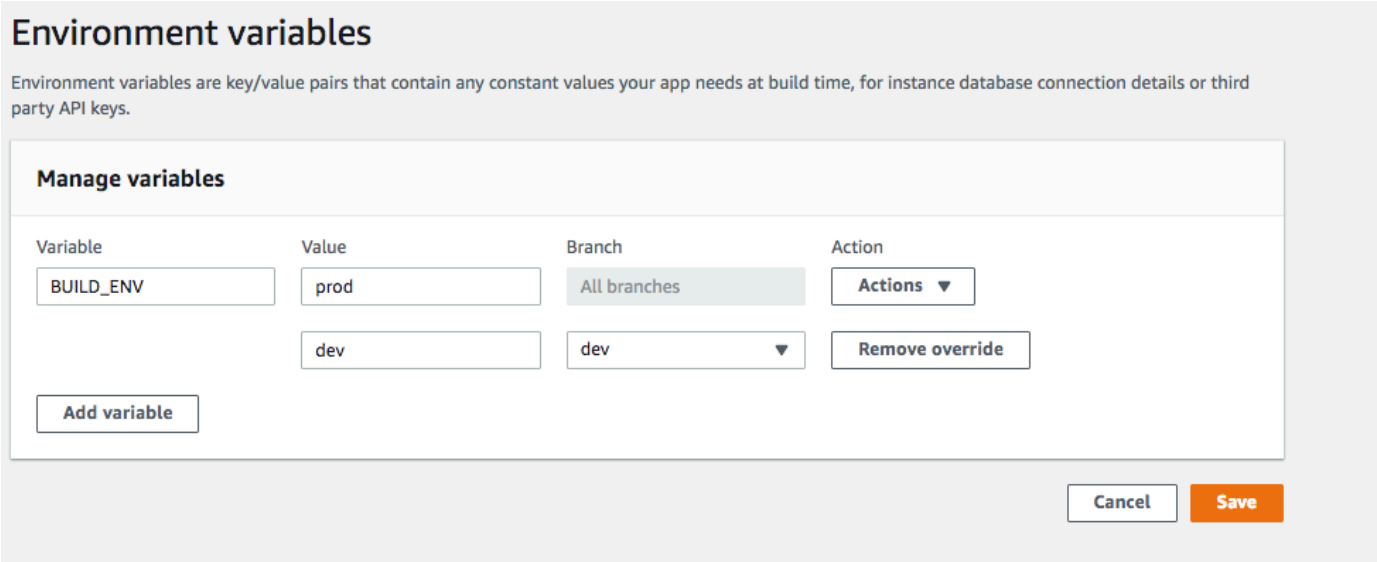
设置环境变量

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

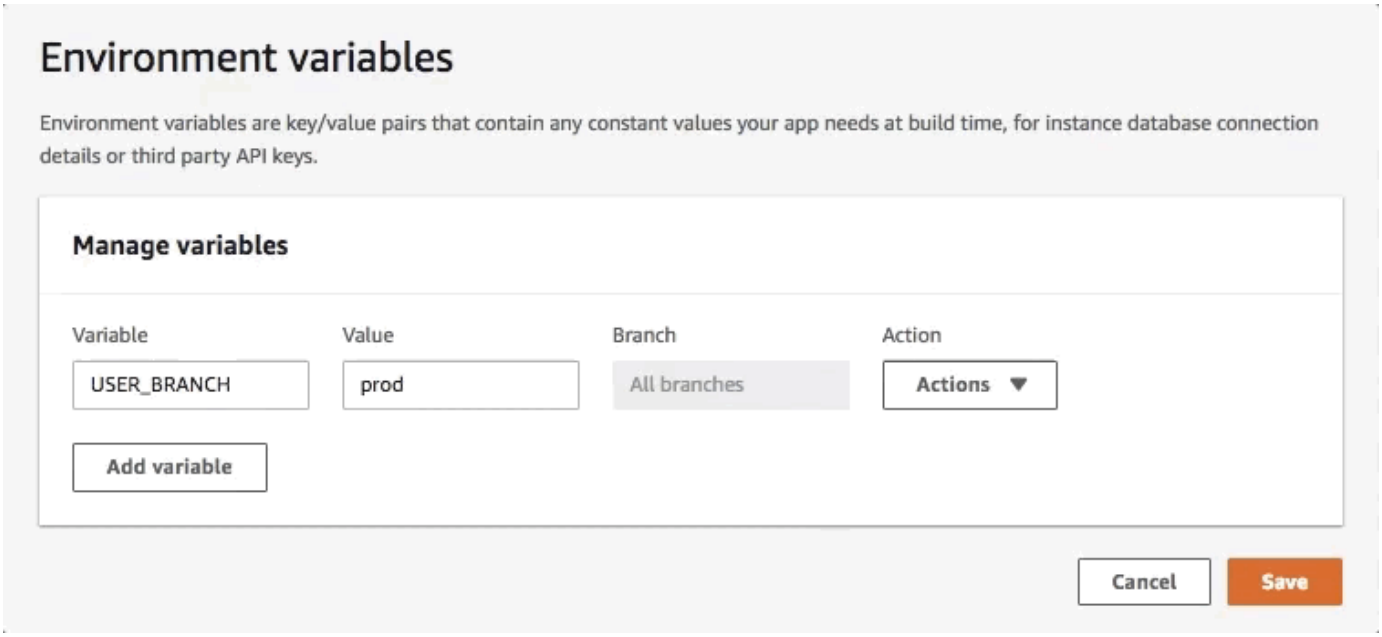
1. 登录 AWS Management Console 并打开 Amplify 控制台。
2. 在 Amplify 控制台中，选择应用程序设置，然后选择环境变量。
3. 在环境变量部分中，选择管理变量。
4. 在管理变量部分，请在变量下输入您的密钥。对于值，请输入您的值。默认情况下，Amplify 会将环境变量应用于所有分支，因此在连接新分支时不必重新输入变量。



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。

在构建时访问环境变量

要在构建过程中访问环境变量，请编辑构建设置以在构建命令中包括环境变量。

编辑构建设置以包含环境变量

1. 登录 AWS Management Console 并打开 Amplify 控制台。
2. 在 Amplify 控制台中，选择应用程序设置，然后选择构建设置。
3. 在应用程序构建规范部分，选择编辑。
4. 将环境变量添加到您的构建命令中。您现在应该能够在下次构建时访问环境变量。此示例更改了 npm 的行为 (BUILD_ENV)，并将外部服务的 API 令牌 (TWITCH_CLIENT_ID) 添加到环境文件中以供日后使用。

```
build:
  commands:
    - npm run build:$BUILD_ENV
    - echo "TWITCH_CLIENT_ID=$TWITCH_CLIENT_ID" >> backend/.env
```

构建配置中的每条命令都运行在 Bash shell 中。有关在 Bash 中使用环境变量的更多信息，请参阅 GNU Bash 手册中的 Shell 扩展。

令服务器端运行时可以访问环境变量

默认情况下，Next.js 服务器组件无权访问您应用程序的环境变量。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。

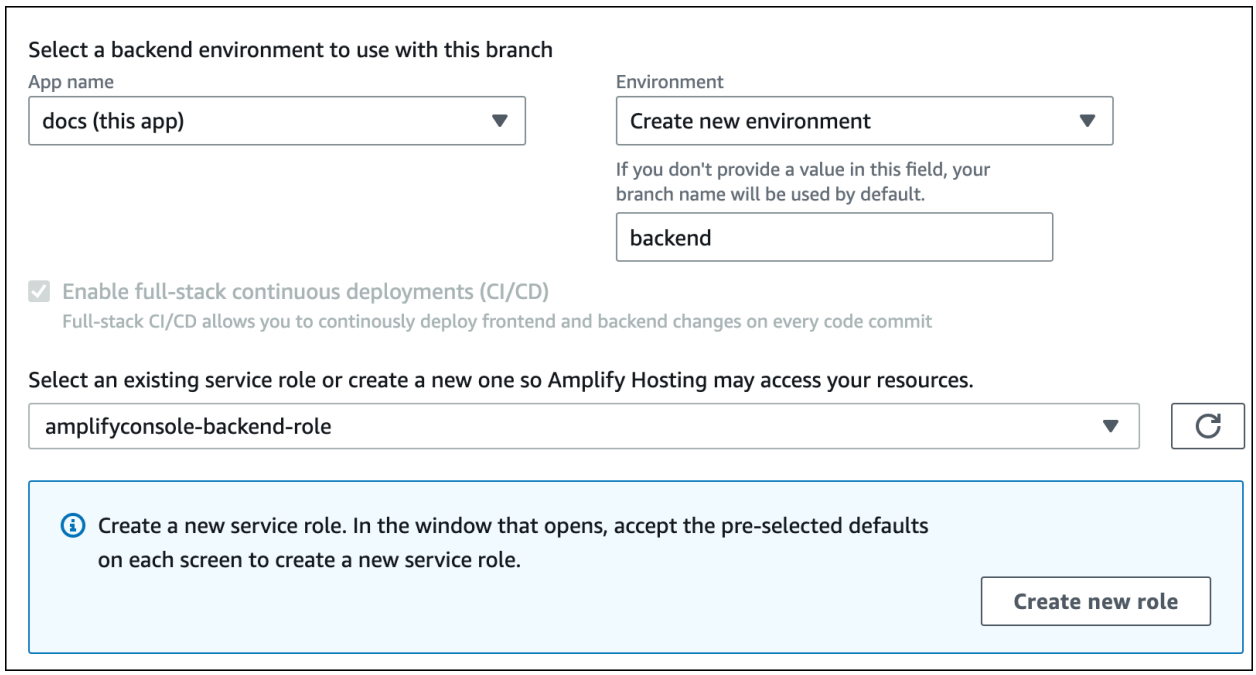
要使 Next.js 可以访问特定的环境变量，您必须修改 Amplify 构建规范文件，以便在环境文件中设置 Next.js 可以识别的环境变量。这样 Amplify 就可以在构建应用程序之前加载环境变量。有关修改构建规范的更多信息，请参阅如何在构建命令部分添加环境变量的示例。

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 Management Console 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 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，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。

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。有关就此目的使用 Amplify 的构建时环境变量的示例，请参阅 在构建时访问环境变量。

环境密钥

环境密钥与环境变量类似，但它们是可以加密的 AWS Systems Manager (SSM) 参数存储密钥值对。某些值必须加密，例如 Amplify 的“使用 Apple 私钥登录”。

设置环境密钥

按照以下说明使用控制台为 Amplify 应用设置环境密钥。 AWS Systems Manager

设置环境密钥

1. 登录 AWS Management Console 并打开AWS Systems Manager 控制台。
2. 在导航窗格中，选择应用程序管理，然后选择 Parameter Store。
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。

访问环境密钥

在构建期间访问环境密钥与访问环境变量类似，不同之处在于环境密钥以 JSON 字符串的形式存储在 `process.env.secrets` 中。

Amplify 环境密钥

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	描述	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-----

自定义标头

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 AWS Management Console 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 AWS Management Console 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。以这种方式指定的自定义标头应从 buildspec 和 `amplify.yml` 文件中迁移出来。有关说明，请参阅 迁移自定义标头。

自定义标头 YAML 格式

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 AWS Amplify 应用程序指定自定义 HTTP 标头。您可以在 AWS Management Console 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 AWS Management Console 中为应用程序设置自定义标头

1. 登录到 AWS Management Console 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，选择应用程序设置、自定义标头。
4. 在自定义标头规范部分中，选择编辑。
5. 在编辑窗口中，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 Save（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。

使用 customhttp.yml 文件设置自定义标头

1. 登录到 AWS Management Console 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，选择应用程序设置、自定义标头。

4. 在自定义标头规范部分中，选择下载。

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 AWS Management Console 中的自定义标头中定义的自定义标头。

迁移自定义标头

以前，要为应用程序指定自定义 HTTP 标头，要么在 AWS Management Console 中编辑 buildspec，要么下载并更新 `amplify.yml` 文件并将其保存在项目的根目录中。强烈建议您将自定义标头从 buildspec 和 `amplify.yml` 文件中迁移出来。

在 AWS Management Console 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. 登录到 AWS Management Console 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，依次选择应用程序设置、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，依次选择应用程序设置、自定义标头。
8. 在自定义标头规范部分中，选择编辑。
9. 在编辑窗口中，输入您在步骤 6 中删除的自定义标头的信息。
10. 选择 Save（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. 登录到 AWS Management Console 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，依次选择应用程序设置、自定义标头。

7. 在自定义标头规范部分中，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 AWS Management Console 的自定义标头部分中定义的自定义标头。

Monorepo 自定义标头

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 格式。
- 您可以使用 AWS Management Console 的自定义标头部分为 monorepo 中的应用程序指定自定义标头。请注意，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 AWS Management Console 的自定义标头部分指定的任何自定义标头。

安全标头示例

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'"
```

缓存控制标头示例

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存活时间 (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'
```

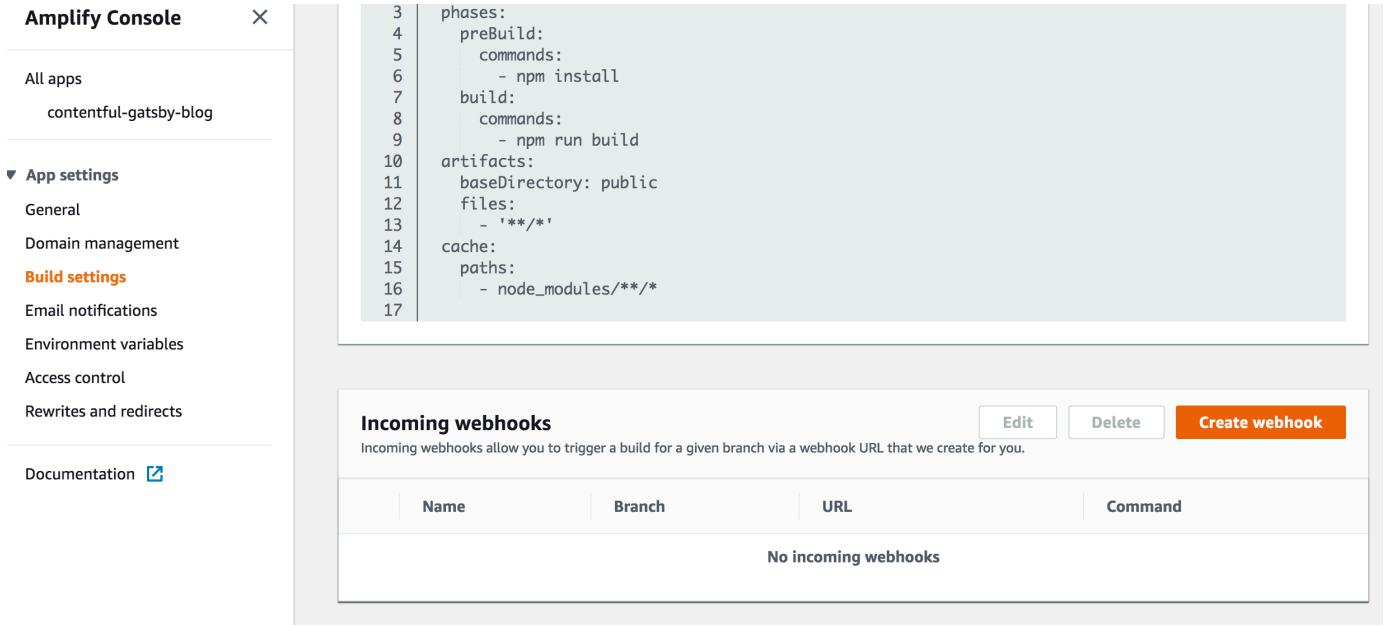
有关使用标头控制应用程序性能的更多信息，请参阅 使用标头控制缓存时间长度。

传入 webhook

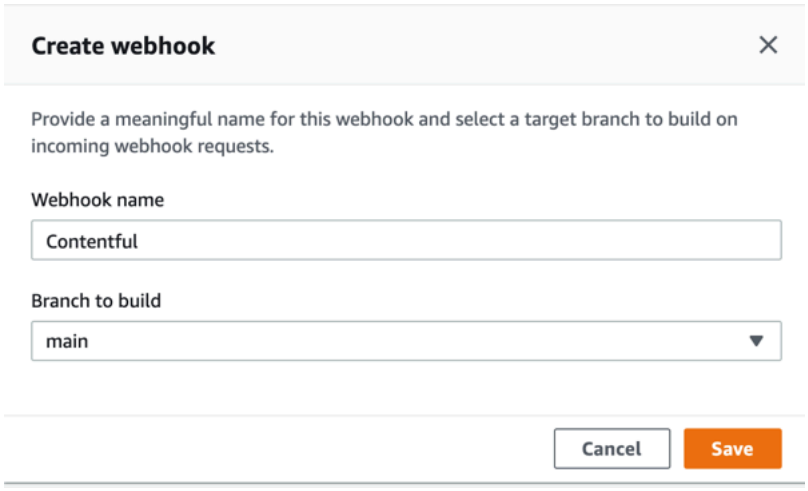
在 Amplify 控制台中设置传入的 webhook，无需将代码提交到 Git 仓库即可触发构建。您可以将无管控 CMS 工具（例如 Contentful 或 GraphCMS）结合使用 webhook 触发器以触发内容更改时的构建，或者使用 Zapier 等服务执行每日构建。

创建传入 webhook

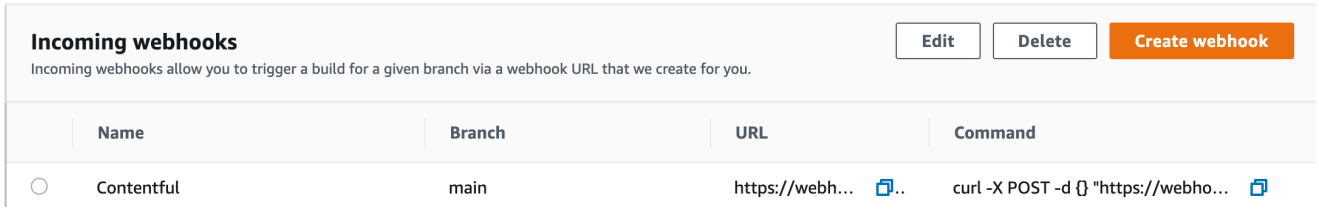
1. 登录 AWS Management Console 并打开 Amplify 控制台。
2. 选择要为其创建 webhook 的应用程序。
3. 在导航窗格中，选择构建设置。
4. 在构建设）页面上，向下滚动到传入 webhook 部分，然后选择创建 webhook。



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. 选择保存。



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管控 CMS 工具或其他服务，以触发构建。
   - 在终端窗口中运行 curl 命令，以触发新的构建。



监控

AWS Amplify 通过 Amazon 发布指标， CloudWatch 并提供访问日志，其中包含有关向您的应用程序发出的请求的详细信息。使用本节中的主题可了解如何使用这些指标和日志来监控您的应用程序。

主题

- 使用监控 CloudWatch
- 访问日志

使用监控 CloudWatch

AWS Amplify 已与亚马逊集成 CloudWatch，允许您近乎实时地监控 Amplify 应用程序的指标。您可以创建警报，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。

指标

Amplify 在`AWS/AmplifyHosting`命名空间中支持六个 CloudWatch 指标，用于监控应用程序的流量、错误、数据传输和延迟。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。

并非所有可用的统计数据都适用于每个指标。在下表中，每个指标的描述中列出了最相关的统计数据。

指标	描述
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。
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	使用 POST 和 PUT 请求传输到您的应用程序（上传）的数据总量，以字节为单位。

指标	描述
	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。

Amplify 提供以下 CloudWatch 指标维度。

维度	描述
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。

你可以在 AWS Management Console https://console.aws.amazon.com/cloudwatch/ 中访问 CloudWatch 指标。或者，您可以使用以下程序在 Amplify 控制台中访问指标。

在 Amplify 控制台中访问指标

1. 登录 AWS Management Console 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 应用程序设置 和 监控。
4. 在 监控页面上，选择指标。

告警

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 CloudWatch API 创建更高级的警报。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 Management Console 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 应用程序设置 和 监控。
4. 在监控页面上，选择警报。
5. 选择创建警报。
6. 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. 指定要触发警报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. 选择创建警报。

> **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。

SSR 应用程序的 Amazon CloudWatch 日志

Amplify 将有关你的 Next.js 运行时的信息发送到你的 Amazon CloudWatch Logs。 AWS 账户部署 SSR 应用程序时，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担任该角色。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 部署服务器端渲染的应用程序。

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 会保留该应用程序的所有访问日志。Amplify 控制台中提供了应用程序的所有访问日志。但是，访问日志的每个单独请求都限制在您指定的两周时间段内。

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。

使用以下步骤检索应用程序的访问日志。

查看访问日志

1. 登录 AWS Management Console 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 应用程序设置 和 监控。
4. 在监控页面上，选择访问日志。
5. 选择编辑时间范围。
6. 在编辑时间范围窗口中，为开始日期指定要检索日志的两周间隔的第一天。在开始时间 中，选择第一天开始检索日志的时间。
7. 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;
```

通知

您可以为 AWS Amplify 应用程序设置通知，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。

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 Management Console 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，选择应用程序设置、通知，然后在电子邮件通知部分，选择添加通知。
4. 在管理通知部分执行以下操作：
   - 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。
5. 完成后，选择 Save (保存)。

自定义构建映像和实时程序包更新

主题

- 自定义构建映像
- 实时程序包更新

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。

> **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。

自定义构建映像要求

要使自定义构建映像作为 Amplify 构建镜像使用，它必须满足以下要求：

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。

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 Management Console 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，选择应用程序设置，然后选择构建设置。
5. 在构建设置页面上，请在构建映像设置中选择编辑。
6. 在 编辑构建映像设置对话框中，展开构建映像菜单，然后选择构建映像。
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 Save（保存）。

实时程序包更新

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。

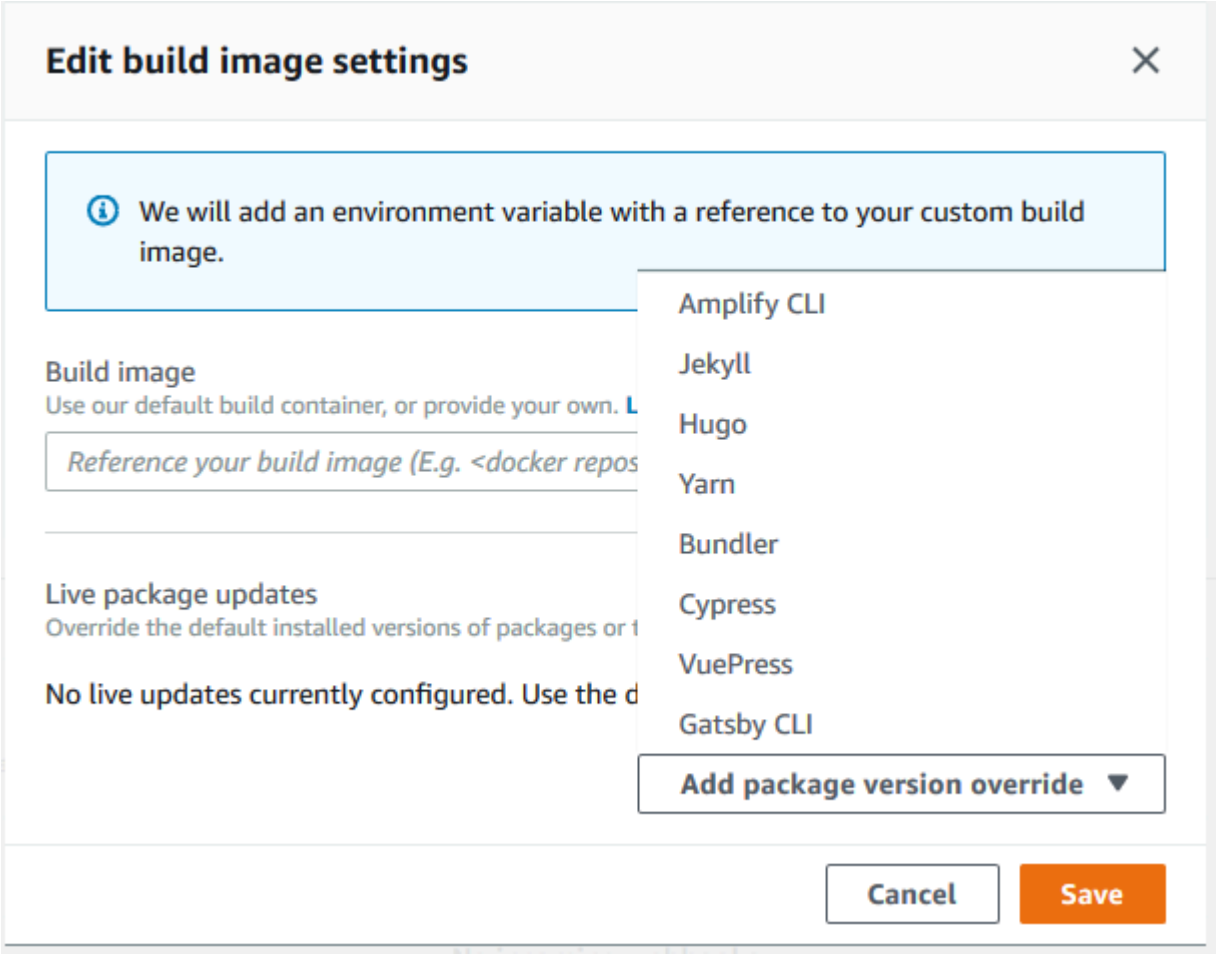
> **Warning**
>
> 将 Node.js 版本设置为最新会导致构建失败。相反，您必须指定确切的 Node.js 版本，例如 `18`、`21.5` 或 `v0.1.2`。

配置实时程序包更新

配置实时软件包更新

1. 登录 AWS Management Console 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，选择应用程序设置，然后选择构建设置。
4. 在构建设置页面上，请在构建映像设置中选择编辑。
5. 在 编辑构建映像设置对话框中，展开添加软件包版本覆盖列表，然后选择要更改的软件包。

   以下屏幕截图显示了编辑构建映像设置对话框，其中包含展开的添加软件包版本覆盖列表。



6. 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。
7. 选择 Save（保存）。

添加服务角色

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 Amplify 控制台在代表您调用其他服务时代入的 AWS Identity and Access Management (IAM) 角色。在本指南中，您将创建一个 Amplify 服务角色，该角色具有账户管理权限，并明确允许直接访问 Amplify 应用程序部署任何 Amplify Studio 或 CLI 资源以及创建和管理后端所需的资源。有关 Amplify Studio 的更多信息，请参阅 Amplify 文档中的入门。有关 Amplify CLI 的更多信息，请参阅 Amplify 文档中的 Amplify CLI。

步骤 1：登录 IAM 控制台

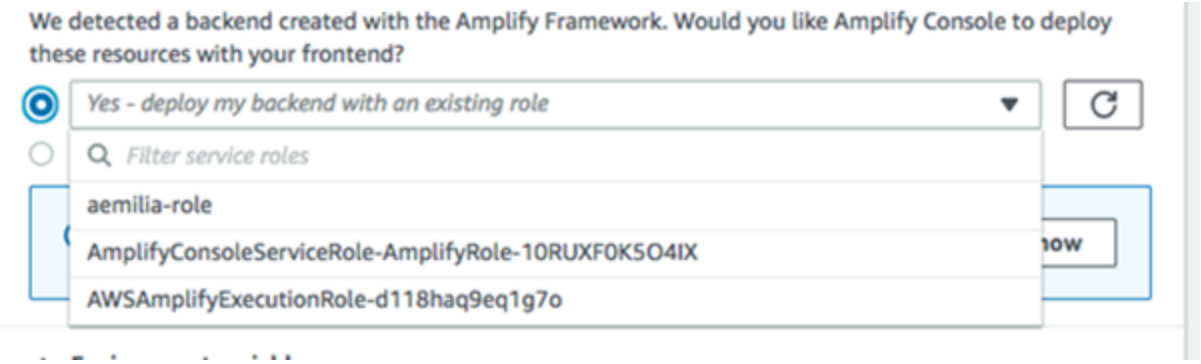
打开 IAM 控制台并从左侧导航栏中选择角色，然后选择创建角色。

步骤 2：创建 Amplify 角色

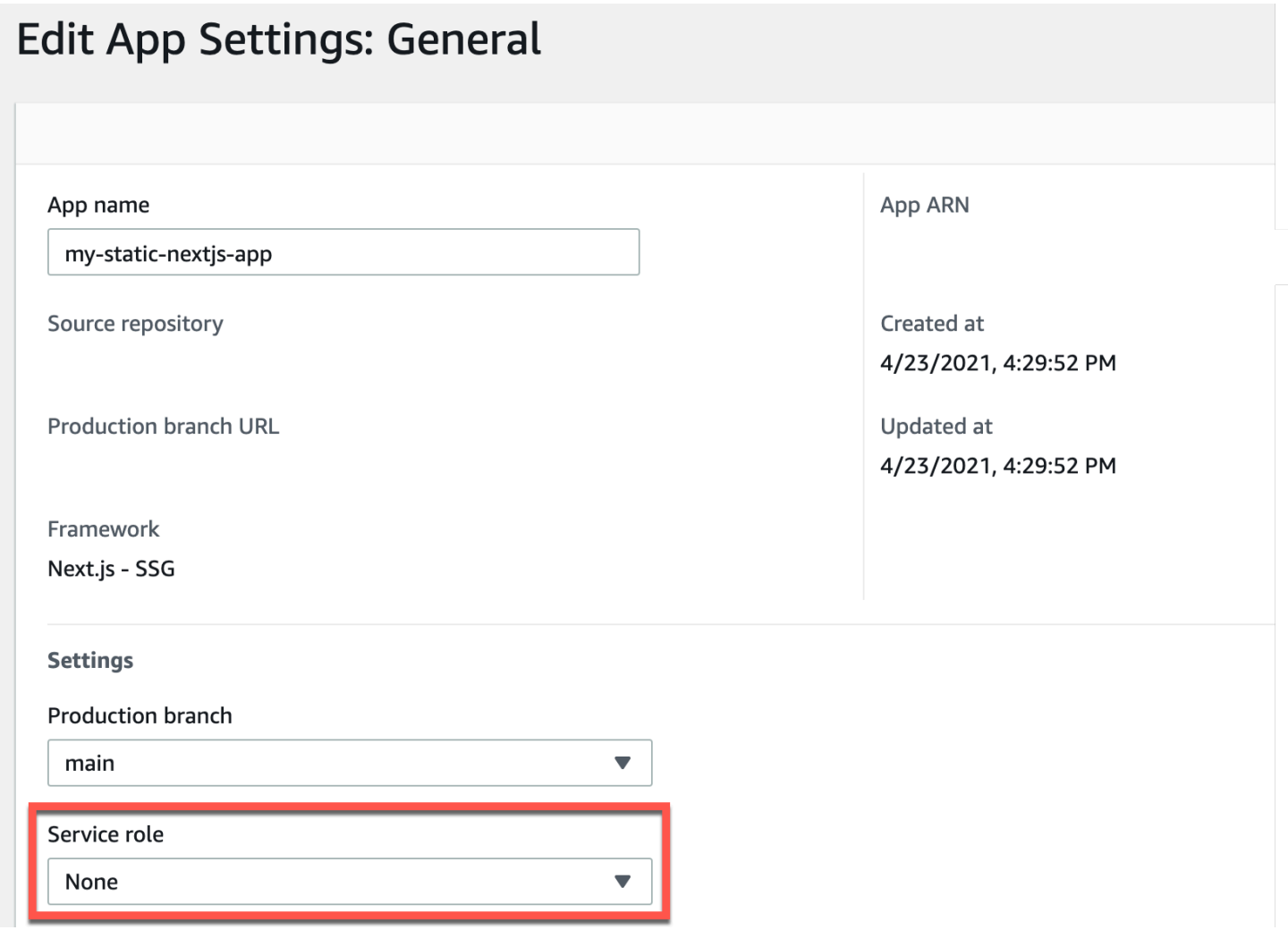
在角色选择屏幕中，找到 Amplify 并选择Amplify 后端部署角色。接受所有默认值，并为您的角色选择一个名称，例如 AmplifyConsoleServiceRole-AmplifyRole。

步骤 3：返回到 Amplify 控制台

打开 Amplify 控制台。如果您正在部署新的应用程序，请选择刷新，然后选择您刚刚创建的角色。它应该类似于 AmplifyConsoleServiceRole-AmplifyRole。



如果您有一个现有应用程序，则可以在 App settings (应用程序设置) > General (常规) 中找到服务角色设置，然后从该框的右上角选择 Edit (编辑)。从下拉列表中选择刚刚创建的服务角色，然后选择 Save (保存)。



Amplify 控制台现在有权部署后端资源。

混淆代理问题防范

混淆座席问题是一个安全性问题，即不具有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座席。

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。请将示例策略中的红色斜体文本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 AWS Management Console 编辑角色信任策略的说明，请参阅 IAM 用户指南中的修改角色（控制台）。

管理应用程序性能

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，我们建议您使用默认架构。

对于需要更好地控制应用性能的高级客户，Amplify Hosting 支持性能模式。性能模式通过将在内容分发网络（CDN）边缘处缓存的内容保留更长的时间间隔来进行优化，以实现更快的托管性能。启用性能模式后，托管配置或代码更改可能需要长达 10 分钟才能被部署或可用。有关更多信息，请参阅 the section called “开启性能模式”。

开启性能模式

使用以下步骤为部署到 Amplify Hosting 的应用程序开启性能模式。

为应用程序启用性能模式

1. 登录到 AWS Management Console 并打开 Amplify 控制台。
2. 选择要为其启用性能模式的应用程序。
3. 在导航窗格中，依次选择 应用程序设置、常规。
4. 在常规窗格中，向下滚动到分支部分。选择要为其启用性能模式的分支。
5. 依次选择操作、启用性能模式。
6. 在启用性能模式对话框中，选择启用性能模式。

使用标头控制缓存时间长度

HTTP `Cache-Control` 标头 `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 托管的应用程序会尊重并重复使用客户端发送的`Cache-Control`请求标头，除非它们被您定义的自定义标头覆盖。

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存活时间 (TTL)。

> **Note**
>
> 当应用程序开启性能模式时，Amplify 会将您可以使用自定义标题为应用程序设置的最大 TTL 从 10 分钟（600 秒）增加到一天（86,400 秒）。Amplify 对您可以使用自定义标头设置的 `s-maxage` 设为一天上限。例如，如果您设置 `s-maxage` 为一周（604,800 秒），Amplify 使用一天的最大 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 缓存控制标头示例。

使用 AWS CloudTrail 记录 Amplify API 调用

AWS Amplify 与 AWS CloudTrail 集成，该服务提供 Amplify 中的用户、角色或 AWS 服务采取的操作记录。CloudTrail 将 Amplify 的所有 API 调用作为事件捕获。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。如果创建了跟踪，就可以将 CloudTrail 事件持续交付到 Amazon S3 存储桶，包括 Amplify 的事件。如果您不配置跟踪，则仍可在 CloudTrail 控制台中的 Event history（事件历史记录）中查看最新事件。通过使用 CloudTrail 收集的信息，您可以确定向 Amplify 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，请参阅《AWS CloudTrail 用户指南》。

CloudTrail 中的 Amplify 信息

您的 AWS 账户上会默认启用 CloudTrail。当 Amazon EKS 中发生任何活动时，该活动将记录在 Amplify 事件中，并与其他 AWS 服务事件一同保存在事件历史记录中。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 AWS 账户中的事件（包括 Amazon EMR 的事件），请创建跟踪。通过跟踪记录，CloudTrail 可将日志文件传送至 Simple Storage Service（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。此跟踪记录在 AWS 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Simple Storage Service（Amazon S3）桶。此外，您可以配置其它AWS服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。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
- 为 CloudTrail 配置 Amazon SNS 通知
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

所有 Amplify 操作均由 CloudTrail 记录，并在AWS Amplify控制台 API 参考、AWSAmplify 管理员用户界面 API 参考和 Amplify UI Builder API 参考中进行书面化。例如，对 `CreateApp`、`DeleteApp` 和 `DeleteBackendEnvironment` 操作的调用将在 CloudTrail 日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 AWS Identity and Access Management (IAM) 用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅 AWS CloudTrail 用户指南中的 CloudTrail userIdentity 元素。

了解 Amplify 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 桶。CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

以下示例显示了 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"
}
```

以下示例显示了一个 CloudTrail 日志条目，该条目演示了 AWS Amplify 管理者 UI API 参考 [ListBackendJobs](#) 操作。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::444455556666:user/Mary_Major",
        "accountId": "444455556666",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Mary_Major",
        "sessionContext": {
            "sessionIssuer": {},
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-01-13T00:47:25Z"
            }
        }
    },
    "eventTime": "2021-01-13T01:15:43Z",
    "eventSource": "amplifybackend.amazonaws.com",
    "eventName": "ListBackendJobs",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.255",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.898
 Linux/4.9.230-0.1.ac.223.84.332.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.275-b01
 java/1.8.0_275 vendor/Oracle_Corporation",
    "requestParameters": {
```

```
        "appId": "d23mv2oexample",
        "backendEnvironmentName": "staging"
    },
    "responseElements": {
        "jobs": [
            {
                "appId": "d23mv2oexample",
                "backendEnvironmentName": "staging",
                "jobId": "ed63e9b2-dd1b-4bf2-895b-3d5dcexample",
                "operation": "CreateBackendAuth",
                "status": "COMPLETED",
                "createTime": "1610499932490",
                "updateTime": "1610500140053"
            },
            {
                "appId": "d23mv2oexample",
                "backendEnvironmentName": "staging",
                "jobId": "06904b10-a795-49c1-92b7-185dfexample",
                "operation": "CreateBackend",
                "status": "COMPLETED",
                "createTime": "1610499657938",
                "updateTime": "1610499704458"
            }
        ],
        "appId": "d23mv2oexample",
        "backendEnvironmentName": "staging"
    },
    "requestID": "7adfabd6-98d5-4b11-bd39-c7deaexample",
    "eventID": "68769310-c96c-4789-a6bb-68b52example",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "444455556666"
}
```

Amplify 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
- Amplify 的数据保护
- 合规性验证 AWS Amplify
- 中的基础设施安全 AWS Amplify
- 在 Amplify 中记录和监控安全事件
- 防止跨服务混淆座席
- Amplify 的安全最佳实践

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
- 使用身份进行身份验证
- 使用策略管理访问
- Amplify 如何与 IAM 协同工作
- 适用于 Amplify 的基于身份的策略示例
- AWS 的托管策略 AWS Amplify
- Amplify 身份和访问问题排查

受众

您的使用方式 AWS Identity and Access Management (IAM) 会有所不同，具体取决于您在 Amplify 中所做的工作。

服务用户：如果您使用 Amplify 服务来完成工作，您的管理员向为您提供所需的凭证和权限。随着您使用更多 Amplify 功能来完成工作，您可能需要额外权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问 Amplify 中的功能，请参阅 Amplify 身份和访问问题排查。

服务管理员：如果您负责公司的 Amplify 资源，您可能具有 Amplify 的完全访问权限。您有责任确定您的服务用户应访问哪些 Amplify 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 Amplify 搭配使用的更多信息，请参阅 Amplify 如何与 IAM 协同工作。

IAM 管理员：如果您是 IAM 管理员，您可能希望了解如何编写策略以管理对 Amplify 的访问的详细信息。要查看您可在 IAM 中使用的基于身份的 Amplify 示例策略，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 用户身份或通过担 AWS 账户根用户任 IAM 角色进行身份验证（登录 AWS）。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。 AWS IAM Identity Center （IAM Identity Center）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的如何登录到您 AWS 账户的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅 IAM 用户指南中的签署 AWS API 请求。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如， AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center 用户指南》中的 多重身份验证 和《IAM 用户指南》中的 在 AWS中使用多重身份验证（MFA）。

AWS 账户 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，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（包括需要管理员访问权限的用户）使用与身份提供商的联合身份验证 AWS 服务 通过临时证书进行访问。

联合身份是指您的企业用户目录、Web 身份提供商、Identity C enter 目录中的用户，或者任何使用 AWS 服务 通过身份源提供的凭据进行访问的用户。 AWS Directory Service当联合身份访问时 AWS 账户，他们将扮演角色，角色提供临时证书。

要集中管理访问权限，我们建议您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中创建用户和群组，也可以连接并同步到您自己的身份源中的一组用户和群组，以便在您的所有 AWS 账户和应用程序中使用。有关 IAM Identity Center 的信息，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

I AM 用户是您 AWS 账户 内部对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，我们建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅 IAM 用户指南中的何时创建 IAM 用户（而不是角色）。

IAM 角色

I AM 角色是您内部具有特定权限 AWS 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。您可以 AWS Management Console 通过切换角色在中临时担任 IAM 角色。您可以通过调用 AWS CLI 或 AWS API 操作或使用自定义 URL 来代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的 使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- Federated user access（联合用户访问）– 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的 为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的权限集。
- 临时 IAM 用户权限 – IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。
- 跨服务访问 — 有些 AWS 服务 使用其他 AWS 服务服务中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon QLDB 中运行应用程序或在 Simple Storage Service（Amazon S3）中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话 (FAS) — 当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，此操作然后在不同服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

- 服务角色 - 服务角色是服务代表您在您的账户中执行操作而分派的 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。
- 服务相关角色-服务相关角色是一种链接到的服务角色。 AWS 服务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相关角色的权限。

- 在 A@@ mazon EC2 上运行的应用程序 — 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时证书。这优先于在 EC2 实例中存储访问密钥。要向 EC2 实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建附加到该实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的 使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅 IAM 用户指南中的何时创建 IAM 角色（而不是用户）。

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。策略是其中的一个对象 AWS ，当与身份或资源关联时，它会定义其权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的结构和内容的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概览。

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。

默认情况下，用户和角色没有权限。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 policy。管理员随后可以向角色添加 IAM policy，用户可以代入角色。

IAM policy 定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 AWS Management Console AWS CLI、或 AWS API 获取角色信息。

基于身份的策略

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 用户指南中的创建 IAM policy。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管式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。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service（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。

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。

访问控制列表 (ACL)

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3 和 Amazon VPC 就是支持 ACL 的服务示例。 AWS WAF要了解有关 ACL 的更多信息，请参阅 Amazon Simple Storage Service 开发人员指南 中的访问控制列表（ACL）概览。

其它策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 – 权限边界是一个高级特征，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅 IAM 用户指南中的 IAM 实体的权限边界。
- 服务控制策略 (SCP)-SCP 是 JSON 策略，用于指定组织或组织单位 (OU) 的最大权限。 AWS Organizations AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管理的服务。如果在组织内启用了所有特征，则可对任意或全部账户应用服务控制策略（SCP）。SCP 限制成员账户中的实体（包括每个 AWS 账户根用户实体）的权限。有关 Organizations 和 SCP 的更多信息，请参阅 AWS Organizations 用户指南中的 SCP 的工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合身份用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资

源的策略。任一项策略中的显式拒绝将覆盖允许。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	支持
ACL	否
ABAC（策略中的标签）	部分
临时凭证	支持
转发访问会话（FAS）	支持
服务角色	支持
服务相关角色	不支持

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 用户指南中的创建 IAM policy。

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。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。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 Simple Storage Service（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。

要启用跨账户存取，您可以将整个账户或其它账户中的 IAM 实体指定为基于资源的策略中的主体。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当委托人和资源处于不同位置时 AWS 账户，可信账户中的 IAM 管理员还必须向委托人实体（用户或角色）授予访问资源的权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅IAM 用户指南中的 IAM 角色与基于资源的策略有何不同。

Amplify 的策略行动

支持策略操作	支持

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。策略操作通常与关联的 AWS API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

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	支持

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。语句必须包含 Resource 或 NotResource 元素。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），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 ARN 的列表，请参阅服务授权参考中 AWS Amplify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	支持

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。

在 `Condition` 元素（或 `Condition` 块）中，可以指定语句生效的条件。`Condition` 元素是可选的。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。

如果您在一个语句中指定多个 `Condition` 元素，或在单个 `Condition` 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则使用逻辑OR运算来 AWS 评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅 IAM 用户指南中的 IAM policy 元素：变量和标签。

AWS 支持全局条件密钥和特定于服务的条件密钥。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 (ACL)

支持 ACL	否

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 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

基于属性的访问权限控制（ABAC）是一种授权策略，该策略基于属性来定义权限。在中 AWS，这些属性称为标签。您可以向 IAM 实体（用户或角色）和许多 AWS 资源附加标签。标记实体和资源是 ABAC 的第一步。然后设计 ABAC 策略，以在主体的标签与他们尝试访问的资源标签匹配时允许操作。

ABAC 在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。

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 Yes（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 Partial（部分）。

有关 ABAC 的更多信息,请参阅《IAM 用户指南》中的什么是 ABAC？。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	支持

当你使用临时证书登录时，有些 AWS 服务 不起作用。有关更多信息，包括哪些 AWS 服务 适用于临时证书，请参阅 IAM 用户指南中的AWS 服务 与 IAM 配合使用的信息。

如果您使用除用户名和密码之外的任何方法登录，则 AWS Management Console 使用的是临时证书。例如，当您 AWS 使用公司的单点登录 (SSO) 链接进行访问时，该过程会自动创建临时证书。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM 用户指南》中的 切换到角色（控制台）。

您可以使用 AWS CLI 或 AWS API 手动创建临时证书。然后，您可以使用这些临时证书进行访问 AWS。 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 中的临时安全凭证。

Amplify 的转发访问会话

支持转发访问会话 (FAS)	支持

当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，此操作然后在不同服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。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	支持

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	不支持

服务相关角色是一种链接到的服务角色。 AWS 服务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相关角色的权限。

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 额表。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。他们也无法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 执行任务。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 policy。管理员随后可以向角色添加 IAM policy，用户可以代入角色。

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 policy。

有关 Amplify 定义的操作和资源类型的详细信息，包括每种资源类型的 ARN 格式，请参阅服务授权参考中的 AWS Amplify的操作、资源和条件键。

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，请遵循以下准则和建议：

- 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 托管式策略 或 工作职能的AWS 托管式策略。
- 应用最低权限 – 在使用 IAM policy 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 policy 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 AWS 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 policy，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 policy 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 IAM Acess Analyzer 策略验证。

- 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 配置受 MFA 保护的 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 中的安全最佳实操。

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，请确定正确的权限以授予需要执行删除操作的用户。

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您的使用场景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。

本政策授予 Amplify 通过部署 Amplify 后端资源（Amazon AWS AppSync Cognito、Amazon S3 和其他相关服务）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。

要查看此策略的权限，请参阅《AWS 托管策略参考》AmplifyBackendDeployFullAccess中的。

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cloudformation:DeleteStack 策略操作以支持在调DeleteBranch 用 API 时删除堆栈。	2024年4月5日

更改	描述	日期
	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 函数的更新。	
AdministratorAccess-Amplify — 更新现有政策	添加`cloudformation:TagResource` 和`cloudformation:UnTagResource` 权限以支持对 AWS CloudFormation API 的调用。	2024年4月4日
AmplifyBackendDeployFullAccess – 更新了现有策略	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 – 更新了现有策略	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
AmplifyBackendDeployFullAccess – 更新了现有策略	添加支持 Amplify 应用程序部署的`amplify:GetBackendEnvironment` 权限。	2023 年 11 月 6 日

更改	描述	日期
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 AWS 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 AWS 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 AWS 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 AWS 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
- 我无权执行 iam：PassRole
- 我想允许 AWS 账户之外的人访问我的 Amplify 资源

我无权在 Amplify 中执行操作

如果您收到错误提示，表明您无权执行某个操作，则您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，可以担任角色。对于支持基于资源的策略或访问控制列表（ACL）的服务，您可以使用这些策略向人员授予对您的资源的访问权。

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
- 要了解如何通过联合身份验证提供访问权限，请参阅 IAM 用户指南中的为经过外部身份验证的用户（联合身份验证）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存取之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。

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 multi-factor authentication（MFA）。
- 使用 SSL/TLS 与资源通信。 AWS
- 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 或软件开发工具包使用 Amplify 或其他 AWS 服务时。 AWS CLI AWS 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 使用针对边缘接入点 (POP) 加密的固态硬盘，为区域边缘缓存 (REC) 使用加密的 EBS 卷。Functions 中的 CloudFront 功能代码和配置始终以加密格式存储在边缘站点 PoP 上的加密 SSD 上以及使用的 CloudFront其他存储位置。

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 Certificate Manager 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 ACM PCA。

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。 AWS 提供了以下资源来帮助实现合规性：

- 安全与合规性快速入门指南 — 这些部署指南讨论了架构注意事项，并提供了在这些基础上 AWS 部署以安全性和合规性为重点的基准环境的步骤。
- 在 A@@ mazon Web Services 上构建 HIPAA 安全与合规架构 — 本白皮书描述了各公司如何使用 AWS 来创建符合 HIPAA 资格的应用程序。

> **Note**
>
> 并非所有 AWS 服务 人都符合 HIPAA 资格。有关更多信息，请参阅符合 HIPAA 要求的服务参考。

- AWS 合规资源AWS — 此工作簿和指南集可能适用于您所在的行业和所在地区。
- AWS 客户合规指南 — 从合规角度了解责任共担模式。这些指南总结了保护的最佳实践， AWS 服务并将指南映射到跨多个框架（包括美国国家标准与技术研究院 (NIST)、支付卡行业安全标准委员会 (PCI) 和国际标准化组织 (ISO)）的安全控制。
- 使用AWS Config 开发人员指南中的规则评估资源 — 该 AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- AWS Security Hub— 这 AWS 服务 可以全面了解您的安全状态 AWS。Security Hub 通过安全控件评估您的 AWS 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控件的列表，请参阅 Security Hub 控件参考。
- AWS Audit Manager— 这 AWS 服务 可以帮助您持续审计 AWS 使用情况，从而简化风险管理以及对法规和行业标准的合规性。

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。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service（AWS STS）生成临时安全凭证来对请求进行签名。

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

- 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 AWS CloudTrail 记录 Amplify API 调用。
- 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、S oftware-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。

防止跨服务混淆座席

混淆代理问题是一个安全性问题，即不具有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。

```
{
  "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。

```
{
  "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，请参阅服务限额用户指南中的请求增加限额。

名称	默认值	可调整	描述
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	不支持	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	不支持	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	不支持	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。

名称	默认值	可调整	描述
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	不支持	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	不支持	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	不支持	在当前区域内，您每小时可使用此账户在 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	应用程序每秒请求令牌的最大数量。Amplify Hosting 根据请求消耗的资源量（处理时间和数据传输）为请求分配令牌。
每个域的子域数	每个受支持的区域：50 个	不支持	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。

对 Amplify 托管进行故障排除

如果您在使用 Amplify Hosting 时遇到错误或部署问题，请查阅本节的主题。

主题

- 对 Amplify 的一般问题进行故障排除
- 对亚马逊 Linux 2023 版本映像问题进行故障排除
- 自定义域问题排查
- 对服务器端渲染的应用程序进行故障排除

对 Amplify 的一般问题进行故障排除

以下信息可以帮助您解决 Amplify 托管的一般问题。

主题

- HTTP 429 状态码（请求太多）

HTTP 429 状态码（请求太多）

Amplify 根据传入请求消耗的处理时间和数据传输来控制每秒向您的网站发送的请求数 (RPS)。如果您的应用程序返回 HTTP 429 状态码，则传入的请求将超过分配给您的应用程序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。

要解决此问题，我们建议您优化您的应用程序以缩短请求持续时间和数据传输以提高应用程序的 RPS。例如，使用相同的 20,000 个令牌，与延迟超过 200 毫秒的页面相比，高度优化、在 100 毫秒内响应的 SSR 页面可以支持更高的 RPS。

同样，返回响应大小为 1 MB 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令牌。

我们还建议您通过配置 CloudFront 缓存控制标头来利用 Amazon 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。

对亚马逊 Linux 2023 版本映像问题进行故障排除

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

- 如何使用 Python 运行时运行 Amplify 函数
- 如何运行需要超级用户或 root 权限的命令

如何使用 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 映像，AL2023 构建镜像预装了旧版本 Python 的符号链接。因此，您不再需要按照 A mplify Host GitHub ing 常见问题解答中的说明更新应用程序编译规范中的构建命令。

默认情况下，全局使用 Python 版本 3.10。要使用特定的 Python 版本构建函数，请在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
```

如何运行需要超级用户或 root 权限的命令

如果您使用的是 Amazon Linux 2023 构建镜像，但在运行需要超级用户或根权限的系统命令时出现错误，则必须使用 Linux sudo 命令运行这些命令。例如，如果您在运行时遇到错误`yum install -y gcc`，请使用`sudo yum install -y gcc`。

亚马逊 Linux 2 构建镜像使用根用户，但是 Amplify 的 AL2023 映像使用自定义`amplify`用户运行你的代码。Amplify 授予该用户使用 Linux sudo 命令运行命令的权限。`sudo`对于需要超级用户权限的命令，这是最佳做法。

自定义域问题排查

如果您在将自定义域名连接到 Amplify 应用程序时遇到问题，请参阅自定义域问题排查获取帮助。

对服务器端渲染的应用程序进行故障排除

如果您在将 SSR 应用部署到 Amplify 时遇到问题，SSR 部署的问题排查请参阅获取帮助。

AWS Amplify Hosting 参考

使用本节中的主题查找 AWS Amplify 各方面的详细参考资料。

主题

- AWS CloudFormation 支持
- AWS Command Line Interface 支持
- 资源标记支持
- Amplify Hosting API

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，只需点击几下即可简化跨多个 AWS 账户和/或区域的部署。

有关 Amplify Hosting，请参阅 Amplify CloudFormation 文档。有关 Amplify Studio，请参阅 Amplify UI Builder CloudFormation 文档。

AWS Command Line Interface 支持

使用 AWS Command Line Interface，通过命令行以编程方式创建 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。

- 最新文档更新：2024 年 4 月 5 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
新的故障排除章节	添加了对 Amplify 托管进行故障排除本章以描述如何修复部署到 Amplify Hosting 的应用程序时遇到的问题。	2024年4月2日
对自定义 SSL/TLS 证书的新支持	在本设置自定义域章中添加了使用 SSL/TLS 证书主题，描述了在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	添加了“对 SSR 框架的 Amplify 支持”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

更改	描述	日期
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 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
对 monorepo 框架的新支持功能上线	更新了 单一存储库构建设置 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

更改	描述	日期
更新了服务器端渲染章节	更新了 使用 Amplify Hosting 部署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 开始使用全栈持续部署 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
新 GitHub 应用程序功能上线	添加了设置 Amplify 对 GitHub 存储库的访问权限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 Hosting 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
更新了配置构建设置章节	添加了 单一存储库构建设置 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

更改	描述	日期
更新了功能分支部署章节	添加了 构建时自动生成 Amplify 配置 主题，以描述如何在构建时自动生成 `aws-exports.js` 文件。添加了 有条件后端构建 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 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
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

更改	描述	日期
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 Hosting 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 Management Console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

更改	描述	日期
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	更新了设置自定义域主题，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 释放	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。